THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

A Refinement Theory For μ-Charts

Greg Reeve

This thesis is submitted in partial fulfillment of the requirements for the Degree of Doctor Of Philosophy at the University Of Waikato.

2001 - 2005

Abstract

The language μ-Charts is one of many Statechart-like languages, a family of visual languages that are used for designing reactive systems. We introduce a logic for reasoning about and constructing refinements for μ-charts. The logic itself is interesting and important because it allows reasoning about μ-charts in terms of partial relations rather than the more traditional traces approach. The method of derivation of the logic is also worthy of report. A Z-based model for the language μ-Charts is constructed and the existing logic and refinement calculus of Z is used as the basis for the logic of μ-Charts. As well as describing the logic we introduce some of the ways such a logic can be used to reason about properties of μ-Charts and the refinement of abstract specifications into concrete realisations of reactive systems.

A refinement theory for Statechart-like languages is an important contribution because it allows us to formally investigate and reason about properties of the object language μ-Charts. In particular, we can conjecture and prove general properties required of the object language. This allows us to contrast possible language design decisions and comment on their consequences with respect to the design of Statechart-like languages.

This thesis gives a comprehensive description of the μ-Charts language and details the development of a partial relations based logic and refinement calculus for the language. The logic and refinement calculus are presented as natural deduction style proof rules that allow us to give formal proofs of language properties and provide the basis for a formal program development framework. The notion of refinement that is encoded by the refinement rules is also extensively investigated.

Acknowledgements

I would like to take this opportunity to pay special thanks to my partner Julie and supervisor Steve Reeves, without whom I would not have had the courage to start, let alone finish this work. Also a warm thank you goes out to all my friends and colleagues from both the University of Waikato and other similar institutions that I have come to know and appreciate.

A special thanks to Lindsay Groves from Victoria University of Wellington for your extensive input during writing up.

Finally, I would like to recognise the funding that helped to support this research, both from the Computer Science Department, University of Waikato, and from a New Zealand Government, Foundation for Research, Science & Technology (FRST) grant.

Contents

Chapter 1

Introduction

The language μ-Charts [65, 67, 66, 72, 74, 77, 78] is one of many Statechart-like languages: a family of visual languages that are used for designing reactive systems.

In the following, we introduce and investigate a logic for reasoning about and constructing refinements for μ-Charts. We give a "proof-theoretic" treatment of the language, in that we present the logic in terms of a series of introduction and elimination rules that can be used in proofs about reactive system specifications.

The reported work fulfils two primary tasks. Firstly, we aim to provide a complete core framework that will stand alone as the latest definition of the evolving language μ-Charts. This includes the core logic that should be used for future conservative extensions to the language. In this sense, one might view the work as a rational reconstruction of the language which was originally introduced by Scholz, most recently in [78]. The logic and therefore the meaning of the language have been reconstructed using a methodology more common to the world of state-based system specification than to the usual traces of behaviour semantics for reactive systems. The second purpose of this work is to show how we can use an existing logical framework and methodology to induce a logic for another language, the significant advantage of this approach being the ability to investigate properties of the language in terms of other well-known formalisms.

Investigating refinement theories for Statechart-like languages is an important contribution because it allows us to formally investigate and reason about properties of the object language μ-Charts. In particular, we can conjecture and prove general properties required of the object language. This allows us to contrast possible language design decisions and comment on

their consequences with respect to the design of Statechart-like languages.

Developing systems and providing correctness proofs hand-in-hand has not yet been widely adopted in practice, though the results from attempting to build such frameworks have. For example, the idea of using formal tools for checking an implementation's behaviour with respect to a specification is becoming common in practical software development. Adding specification-like statements, *e.g.* "assert" statements (or assertions), to software code and therefore allowing more sophisticated correctness checking is also becoming commonplace. This practice is often attributed to the work of Hoare [40] in the late 1960s.

In the future, a robust framework for using refinement to guide design is necessary, at least, to provide proper empirical evidence to support or deny the claim that using such a design methodology is a practical alternative. The most common way to use refinement theories to date is in a "posit and prove" rôle; *i.e.*, specification and implementation are developed and then refinement is used to show, after the fact, that the implementation implements the specification.

In the remainder of this chapter we provide a broad outline of the context in which the research belongs and the additional contributions presented here. Section 1.1 gives more general motivation as to why we consider this research to be important and interesting. Sections 1.2 through 1.4 give an overview of relevant literature. In particular: Section 1.2 describes the part of computer science covering *reactive systems*; Section 1.3 introduces the specification language Z and gives a detailed introduction to the work that formulated the methodology employed here; Section 1.4 describes the foundations of the notion of *refinement*. Finally, sections 1.5 and 1.6 list the significant contributions of the work presented and outline the contents of the remaining chapters.

1.1 Why formalise?

In computer science we often use the term *refinement* to refer to a step-wise process of transforming an abstract specification of desired system behaviour into an executable implementation of that behaviour. The resulting program must embody or implement the functionality described by the specification, by definition.

To allow formal refinement, *i.e.*, refinement by application of formal rules, guaranteeing the retention of specified behaviour, *i.e.* system *verification*, we

must start with a suitable formal specification. To obtain a formal specification, a formal mathematical language must be used that allows unambiguous descriptions of requirements to be formulated at an appropriate level of abstraction. Not only does this facilitate formal refinement, but it can also help to minimise misunderstanding between the client and developer. Importantly, a formal specification allows the developer to have a rigorous methodology for analysing the specification and help to detect differences between a client's requirements and the formal specification early in the development life-cycle; that is, it allows more rigorous specification *validation.* Though unambiguous in its statements, typically we require a language that allows partial or "loose" specification, generally through the use of nondeterminism or under-specification. This enables details which are unimportant to the client or outside of the problem domain that requires a formal treatment to be left unspecified and the choice of their implementation can be left to later stages in the program derivation. Also, the ability to abstract from low-level implementation details allows a lucid description of required behaviour without the complexity and constraints of implementation details.

While the above description of a formal development should be considered the ideal for software engineers, it is difficult to deny that this theoretically appealing idea of development is still not widely adopted in practical software development. It is clear, however, that some of the results from the extensive formal methods research are becoming common practice in software development, such as the use of assert statements and model-checking technology. There is also an increasing number of well-known successful examples of large-scale uses of highly formal development, such as the driverless line 14 of the Paris Metro where the "driver" is a fully-proved 87,000 line Ada control program. This development included roughly 100,000 lines of specification and refinement, together with 28,000 proofs all fully proved (about 10 percent of them interactively). Another example is the software developed for smart cards.

There are several accounts that outline the benefits of a formal approach to development available in the literature, for examples see [9, 10, 24, 34, 44, 85]. It also appears that there is significantly less effort expended in refuting these arguments; *e.g.*, see [27]. However, it still appears that the uptake of formal methods-proper is minimal, particularly considering the effort and people involved in the research developing these formal methods. Rather than reconsidering the existing arguments that discuss why the adoption of formal methods has been surprisingly slow, we outline why we find this

particularly surprising. That is, we consider some of the observed problems with existing alternative software development methods.

One such common development approach is to adopt an informal approach to gathering and documenting requirements. For example consider the following scenario. A project begins with a specification written in natural language developed through consultation with a client. This specification is taken and refined by an informal design process and "coded" into a program written in a popular programming language. The program is debugged and tested, and then presented to the client (often still containing unfound errors) for their acceptance. Some of the significant problems with this process stem from the fact that the client and the developer typically have different expectations and motivations. The informal nature of the specification allows it to be ambiguous; that is, it can be interpreted differently depending on "how you look at it". Significantly, the first formal description of the functionality is typically the program itself (*i.e.* described in the programming language of the eventual implementation). An adverse effect of this is that the complexity of implementing the functionality described in the informal specification is not realised until near the end of the first implementation.[1] The resources expended by this stage of development are generally too substantial to allow major changes in the specification. Therefore, if the client and developer have different expectations of the requirements, it is difficult to resolve who will bear the additional costs required to align their expectations.

Again considering natural language specifications as an example, most natural language is prone to being ambiguous and open to interpretation. The more precise the specification, the more wordy the specification, for example consider prose used in legal jargon—it is often particularly verbose in an attempt to be unambiguous. The verbose nature of unambiguous requirements specifications is one of the likely factors that contributes to insufficient requirements documentation. This, in turn, contributes to the phenomenon common to software development known as "project explosion", where a software development project gets to the implementation stage of development, or further, before it is realised that unexpected complexity of the required solution has caused or will cause expenditure of resources well exceeding the

[1]We use the term "first implementation" here because it is commonly believed that most current program development practices spend a large percentage of the development time "debugging" or rewriting the initial program code. For example, during an invited talk at FME97 Robin Bloomfield (of Adelard [70]) said that their long-range data showed that testing accounts for about 50% of development costs

budgeted amount. This is the point at which a lot of program developments currently fail [43].

Given this situation it is not surprising that there is increasingly more effort being made to integrate more rigorous methods into the formal abstract specification of requirements. When viewed in the context of being able to help prevent "project explosion", the often criticised "additional" effort required to produce formal specifications of requirements no longer appear economically infeasible. The complexity of formal specification can be justified because the complexity of the eventual solution is displayed from the very beginning for both developer and client [22]. According to Turski [83], "the specification's 'resistance' to change (expressed in the effort needed to do so properly) is not an *obstacle*, but a *warning* about the real magnitude of effort required to accommodate a change in the application domain."

In [45], Chapter 4, Horrocks gives a good description and supporting example that outlines some of the problems with natural language specifications for graphical user interface design.

Another adverse consequence of the program code being the first formal description of the required functionality[2] is the inability to guarantee program correctness given the size and complexity of most realistic code-based programs. This situation generally leads to the client, and often even the developer, relying on the operational characteristics of programs to test whether or not their functionality respects the functionality that is required. Again this relies on having a specification of the requirements to test against. Also, it is well-known that testing can only confirm the presence of program errors, not their absence [23].

Jacky [49] suggested that one of the barriers to the introduction of more formalisation at the beginning of a software project may be that the formalisation of behaviour from the outset is harder than the alternatives, for example "letting testers or users figure it out for themselves". Jacky also suggests several significant benefits that are offered by elucidating complexity from the beginning of project development. One such advantage is that a significant investment at the start of a software project can produce significant savings in the time required for tasks, such as debugging, testing and

[2]In fact it could be argued that some of the popular programming languages currently used to describe programs are not completely formal, considering formal to at least mean completely unambiguous, because they often lack a proper formal semantics. Therefore their operational semantics depends on which (of many) compilers are used to interpret the program statements.

refactoring, that are almost always required after implementation.

In [44], Holloway presents a compelling argument that supports the suggestion that "formality will eventually become the norm in software development". Briefly his rationale is as follows:

> Software engineers strive to be true engineers; true engineers use appropriate mathematics; therefore, software engineers should use appropriate mathematics. Thus, given that formal methods is the mathematics of software, software engineers should use appropriate formal methods.

Examples such as the recent Model Driven Architecture (MDA) initiative of the Object Management Group (OMG) [63] demonstrate that it is true that formalism is becoming an important consideration of practising engineers.

1.2 Specification languages for reactive systems

When designing computer systems, it is common (if not essential) to break down the task that we wish the system to perform into subtasks. This is generally done repeatedly until we are left with manageable pieces of "the puzzle" to solve. Once the pieces have been solved then they are recombined to solve the whole problem.

Obviously this process requires considerable care and a careful eye to ensure that the parts can be smoothly recombined. This type of activity has led to at least two conceptual models of computing systems. These models were derived by considering the way in which the parts or processes interact with one another.

The first model is that of sequential systems—where the individual parts that make up a program are assumed to carry out computation in some predefined order; *i.e.*, one after the other or one calls another waiting for termination before continuing. The second is the concurrent model. In this model the processes that combine to create the system are assumed to be able to carry out their computations at the same time as other processes.

The concurrent notion may be considered more powerful because it allows the introduction of parallelism, and also sequential computation can easily be characterised in a concurrent model but not necessarily *vice versa*. As usual

there is a trade off between expressibility and complexity: the addition of concurrency may allow increases in the efficiency of solutions; however these solutions are generally more complex and so harder to think about. Hence the need for formal modelling to deal with the added complexity is even more prevalent.

A common characterisation of computing systems is to say they are one of the following types: *transformational*—the system has all of its input at the beginning of execution; *reactive*—the system's computation is determined nondeterministically (in time) by input from its environment; or *interactive*—the system prompts the environment for input as necessary. The term *reactive system* was introduced by Harel and Pnueli and originally was used to refer to what are now commonly divided into reactive and interactive systems [33]. Most modern computer-driven systems can be broken down into a mixture of reactive and interactive concurrent parts [5].

While the concurrent model of computation was initially introduced to help with the complexity of modelling parallelism it was later realised that the same model was useful, if not essential, for modelling reactive systems. The sequential model is good for describing transformational systems but is difficult to extend to reactive systems whose timing characteristics are determined by their environment.

This thesis will concentrate on reactive systems and therefore this section concentrates on formalisms that have been introduced that allow us to describe, reason about and implement systems using a concurrent model. Similar results may hold for interactive systems, however these are outside of the scope of this work.

We discuss a further choice of model that is introduced when considering concurrency and describe how the literature motivates the choice of one model over the other for describing and constructing reactive systems. Also we briefly introduce and contrast some of the existing formalisms for describing reactive systems using a concurrent model.

When considering a concurrent model of systems, often the processes described above are termed *agents* [57]. Such a process can be thought of as an autonomous deterministic "black box" that has some input and output interface. A *reaction* [6] is some finite collection of interactions between these processes that performs some desired task in the system. From these assumptions it seems natural that we should model the behaviour of the system or program by considering only the way in which these processes interact, *i.e.* their communication.

Milner [57] presents this model of the communication between concurrent processes as describing each subtask of a system by describing its observable characteristics, where to observe a process is exactly to communicate with it. He formalised this notion when describing the concurrent formalism *process calculus* which was based on the earlier and well known Calculus of Communicating Processes (CCS) [56]. Milner also defines several equivalence relations for agents: the two main equivalences that are introduced are *strong bisimilarity* and *weak bisimilarity.* Strong bisimilarity requires not only input/output equivalence, *i.e.* two agents have the same external behaviour, but also requires the internal transitions of the two agents to simulate one another exactly. Weak bisimilarity is more liberal in that an internal transition of one agent can be matched by zero or more internal transitions of the other agent.

As we shall see in Section 1.4, having a notion of equivalence between processes is important to allow us to prove that replacing an abstract description of a process with a more concrete description is a valid refinement of the overall system. Milner [57] also introduces instantaneous communication between processes which will be explored below.

The *process calculus* encodes asynchronous agents or processes that synchronise via communication. As in [57], we will use the term *asynchronous processes* to refer to concurrent processes or agents that proceed at indeterminate relative speeds, and *synchronous processes* to refer to concurrent processes that proceed in lock-step.

In [6], Berry and Gonthier describe why the synchronous processes model is essential for the construction of reactive systems by giving the failings of the alternative asynchronous approach. These criticisms, which were formulated through practical considerations whilst developing the language Esterel for reactive programs, are broadly as follows: processes or reactions can compete with each other, *e.g.* new inputs can arrive from another process before the current reaction is finished and therefore the behaviour of the reaction is not atomic, *i.e.* it can depend on the timing characteristics of concurrent reactions; reasoning about temporal properties of a program can be arduous and never completely rigorous—there is no completely accurate way (in terms of time) to determine when a process is finished; finally, each process has its own perception of the entire system, *i.e.* processors may see the system in different states during the same reaction: for example, a single sensor, read by two concurrent processes in the same reaction, could differ in value because it is read at different times. These problems are claimed to be resolved by

using a strong synchronous model, which claim Berry and Gonthier describe as the *synchrony hypothesis.*

In [78] Scholz gives a different categorisation of the notion of synchrony where he describes three distinct types of synchrony. Firstly, *clock synchrony* refers to a system where any concurrent components or parallel processes advance in lock-step with a common clock, *e.g.* a "tick" of the processors' global system clock. *Clock synchrony* is the same as the *synchronous processes* described above. The second form is termed *I/O synchrony* and refers to a system where it is assumed that no time elapses between receiving input and sending output, that is the processes' internal computation is assumed to take no time, and the output is available at the same instant as the input. Finally *message synchrony* refers to a system in which the initiating source of a message is blocked until the recipient is ready-to-receive, that is, *message synchrony* is what we have referred to as 'synchronising via communication' above but it is also commonly referred to in the process algebra world as *synchronous communication.* The term *perfect synchrony*, which is related to the *synchrony hypothesis* described above, refers to a system that obeys both *clock synchrony* and *I/O synchrony.*

Another notable formalisation based on an asynchronous processes model is Communicating Sequential Processes (CSP) that was introduced by Hoare in [42]. CSP is also based on the communicating process model of concurrency. The work that led to the creation of CSP was born out of the realisation that the traditional method of implementing concurrency through the use of a shared-store was inadequate because it could lead to severe implementation problems with the prevailing hardware technology. Also, Hoare identified [41] that a shared store implementation made attempting to construct correct programs difficult. CSP uses handshake communication as does CCS, described above, as a means of enforcing synchronisation via communication, therefore processes are asynchronous but the communication between them can force synchronisation.

The difference between CCS and CSP is so subtle that a detailed explanation is outside of the scope of this work. Briefly, the CCS composition operator implements private point to point communication and has a specific operator that enforces synchronisation whereas the CSP composition operator enforces synchronisation between like-labelled transitions and has a separate operator that makes that communication private.

Clearly, there are significant differences between the asynchronous processes approach of CCS and CSP and the perfect synchrony approach of

Esterel. The synchrony hypothesis described in [6], *i.e.* assuming reactions take no time and that processors adhere to a global clock, was developed as a high-level abstraction technique for describing low-level reactive systems. Some of the observations in [6], which justify the synchrony hypothesis as a reasonable abstraction, include: "synchrony is a natural abstraction from a user's point of view: the user of a watch does not worry about internal reaction times, as long as he *perceives* that his watch reacts instantly to his commands."; "that synchrony hypotheses are very classical in physics: instantaneous body interaction is the basis of Newtonian Mechanics, instantaneous propagation of electricity is the basis of Kirchoff's laws."

More specifically Scholz [78] explains that the abstraction made by *I/O synchrony*, *i.e.* that input and output are available at exactly the same time, can be reduced to assuming that the computation of any reaction can be carried out before the next input arrives. This argument is less convincing when one is considering how to ensure the correct implementation behaviour of executing arbitrary sequential communicating programs in parallel on different processors. In [56] Section 9.3, however, Milner also investigates giving a calculus similar to that for CCS that assumes process synchrony. Some interesting observations that he makes are: that this synchronous process calculus, *i.e.* based on the assumption that processes proceed in lock-step, "has a greater claim to completeness of expression [than CCS itself]—and hence a claim to be less arbitrary"; and that the assumption of synchronous processes "somewhat offends the popular relativistic view, since it suggests the idea of a global clock"; also "the synchronous view leads to a delightfully simple calculus, and in fact asynchrony can be achieved within it just by introducing an explicit *waiting* action." Milner also points out that the asynchronous calculus does suffer from excessive expansion in the context of n-way composition, see [56] for details.

According to Huizing and Gerth [46], some of the benefits that the principle of perfect synchrony give when specifying reactive systems are that: concrete reaction times (*i.e.* 0) are known and therefore we can still correctly model the important timing characteristics of a reactive system; we do not need to fix any of the eventual implementation reaction times at this level of abstraction, therefore the reaction time is as short as possible, and we do not need to introduce artificial delays; reactions can be refined to several sub-reactions without changing the timing behaviour of the specification, *i.e.* $0 + 0 = 0$. Huizing and Gerth also formalise three properties or criteria that describe useful semantic properties of languages for describing reactive

systems. They have labelled these properties *responsiveness*, *modularisation* and *causality*. We will briefly summarise each of these in turn and then present a more detailed description of causality.

Responsiveness refers to the language having the abstraction that allows a reaction to take no time. A system's output is available simultaneously with the input that caused it. When breaking the description of a reactive system into parts the parts or processes should be symmetric in their view of events, *i.e.* they should be *clock synchronous* and communication between a system and its environment happens in the same way as communication between the parts of the system. This property is labelled *modularisation*. The final formal property, *causality*, insists that a collection of events that constitute a system reaction must be causal. That is, there must be some external event that directly or indirectly causes a reaction and there must be no causal loops, *e.g.* the output from one process in the reaction must not eventually cause another transition in the same process or prevent the initial transition from happening. Huizing and Gerth go on to prove that these three properties cannot be present together in one semantics for a language for describing reactive systems.

Because of this result it is common for languages designed for describing reactive systems to incorporate responsiveness and modularisation and to deal with causality separately. Causality is often identified [33, 46, 65, 84] as a problem with the synchronous model of concurrency, but because it is a static semantic problem, that is it can be identified by static analysis of the description of the reactive system, it is tolerable. One suggested method for dealing with reactive system descriptions containing causality problems is to perform static analysis on the description of a system and reject it if it is possible that it contains a causality problem, *e.g.* Esterel takes this approach [6]. The problem with this is that it may rule out some descriptions of reactive systems that do not actually contain causality problems and static analysis requires a thorough state exploration which, even with current model checking technology, could be intractable [65]. Another method described in [78] is to ignore causality in the description of reactive systems and remove it during refinement. This is also the approach that we take here. For a more in-depth discussion about how causality affects the semantics of μ-Charts given by the Z model presented here, the interested reader should consult [73].

Apart from the formalisms CCS, Esterel and CSP, another method for

describing reactive systems is Statecharts.[3] Statecharts were introduced by Harel in [14] and according to Berry [5] it is a "quasi-synchronous formalism", *i.e.* based on a model for reactive systems that assumes processes behave periodically, they all have nearly the same period but no common clock and they communicate by means of shared memory [2, 12]. Because Statecharts are based on finite-state automata, which are easily represented in a graphical form, they were the first of many "visual" specification languages. It is commonly claimed (*e.g.* [31]) that the ability to use diagrams to represent specifications has made Statecharts a popular specification tool for software engineers.

The well-known Unified Modelling Language (UML) [64] adopted "object-based variants of Harel Statecharts that incorporate several concepts defined in ROOMcharts, a variant of the Statechart defined in the real-time object-oriented modelling (ROOM) language" [79] known as *UML statecharts* or *UML state machines* [69].

The invention of Statecharts marked the beginning of extensive research into making precise their semantics, given informally in [14]. In particular, some of this research led to the creation of another visual specification language for reactive systems called μ-Charts. μ-Charts is a simplified version of Statecharts and is the language that is the basis of the reported research. Section 1.2.1 introduces the formalism μ-Charts and Section 1.2.2 gives a brief comparison of μ-Charts with respect to the original *Harel-Statecharts.*

1.2.1 μ-Charts

The pedigree of μ-Charts is a chain of formalisms starting with Statecharts, as described above, and includes Mini-Statecharts [62], an extended version of Mini-Statecharts [76], the μ-Charts of [65], the μ-Charts of [78] and the μ-Charts of [74]. More specifically, μ-Charts was created and developed by Philipps and Scholz in [65, 66] with the aim of showing how to give it a proper denotational semantics, and also to use these results to generalise to Statecharts, hence showing how to give a denotational semantics to Statecharts and where problems occur with the original description. Later in [78], Scholz continued to develop μ-Charts and, as expected, gives some program development rules and a notion of implementation. Subsequently a transla-

[3]Hereafter we follow a common naming convention, similar to that used by André in [1], where we use "Statecharts" and "μ-charts" to refer both to several charts and (with a capital 'C' in the case of μ-Charts) to name the language in which the charts are written.

tion method from μ-Charts into the Z specification language has been developed [72, 73, 74] (also included in Appendices) in conjunction with the ISuRF and $\mathcal{Z}_\lambda$ research projects [48, 55].

Like Statecharts, μ-Charts allows the description of a reactive system via a Mealy-like finite-state automaton such as that of Figure 1.1. The description of the chart contains states denoted by ovals, including an initial state which is distinguished by a double ring, and transitions denoted by arrowed lines. The transitions are labelled with a trigger and an action separated by a "/". The trigger is a boolean expression over the input set. For example, in Figure 1.1, the transition label s/p means the labelled transition can occur if the chart is in its initial state A and the signal s is in the input. When this transition occurs its action is to output the signal p and move into state B. For completeness, the other trigger featured in Figure 1.1, *i.e.* $-s$, means the appropriate transition can occur when the chart is in state B and the signal s is **not** in the input set.

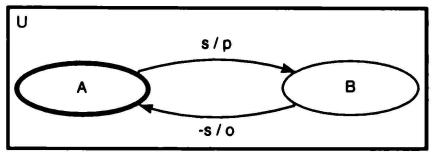


Figure 1.1: Simple sequential μ-chart

Given the basic building block for charts, complex specifications of reactive systems are constructed by composing charts together and allowing them to communicate by the instantaneous broadcast of particular output signals. The most general type of composition is demonstrated in Figure 1.2 where the charts named A and B are assumed to react in perfect synchrony and communicate by broadcasting the signal b. The other type of composition allowed is a special case that is often referred to as *decomposition.* Decomposition is visually denoted by replacing an atomic state of one chart with another μ-chart. The meaning of decomposition is the same as composition except that the "embedded" chart has additional conditions implicitly on all of its transitions that will only allow them to fire when the chart in which it is embedded is in the "enclosing" state.

A detailed account of μ-Charts' syntax and semantics is given in chapters 2 and 3.

It is important to remember that when we write specifications describing systems we may want to write them in an abstract manner so as to remove

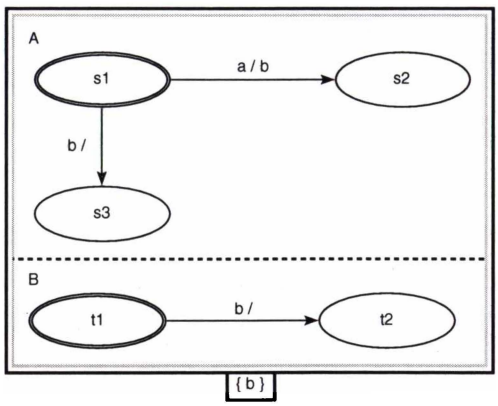


Figure 1.2: Composed μ-charts

low-level implementation details such as architecture dependent information, communication timing considerations, *etc.* We want to concentrate on fully capturing user requirements and developing a high-level knowledge of the functionality required in the proposed reactive system. μ-Charts provides such a specification language partly by allowing nondeterminism—choices can be left unspecified or nondeterministic in order that decisions can be made later in the design process. Also μ-Charts allows the abstraction mechanism often identified as a strength of Statecharts where we can describe an entire system as an abstract process and then decompose each of the states as though they are subprocesses of the system.

The semantics for μ-Charts was designed to obey perfect synchrony, that is, it employs the *synchrony hypothesis* and therefore, according to Berry [6], is suited to describing reactive systems. The synchrony hypothesis means that a μ-chart's transitions are assumed to take no time, *i.e.* time only passes when a μ-chart is in a particular state, and instantaneous feedback is used as the model of signal broadcasting. Because output signals are available at the same instant as input, broadcast outputs from one chart are available to all other charts in scope.[4] Importantly, subscribing to the principle of perfect synchrony allows μ-charts to capture the required timing characteristics of a reactive system, as identified above, by abstracting on the communication timing characteristics of the eventual implementation. This abstraction also allows us to give a (reasonably) eloquent Z model to charts based on Z descriptions of their transitions. This Z model of charts provides the foun-

[4]Signals that can be broadcast from a μ-chart are represented in the graphical representation (see Figure 1.2) by attaching a rectangle, containing the set of feedback signals, to the box surrounding a μ-chart. Any broadcast output from any sequential chart enclosed by this box is in scope, *i.e.* can see the output and consider it as input.

dations of the Z semantics for charts which includes a logic and refinement rules.

Other important reasons for choosing the μ-Charts formalism include: μ-charts are simple in construction and allow structuring of specifications through composition operators; the formalism was originally conceived with the aim of providing both a formal denotational semantics and a notion of formal refinement; finally, there was an existing link between μ-Charts and the specification language Z.

Initially this link between μ-Charts and Z [72, 74] was considered as a translation of the existing language μ-Charts into the language Z. Now we consider the Z model of μ-Charts and the resulting logic and refinement notion as giving the semantics of the language.

1.2.2 μ-Charts and Statecharts

We give a brief overview of the difference between the language μ-Charts that is the topic of this dissertation and the original Statecharts language (that is, the account, see [35], of the semantics of the orignal Statecharts developed by Harel in [14]). From [35], the interested reader can find numerous papers that describe all of the Statecharts variants and how they differ. This includes the *state machines* that are part of the Unified Modelling Language (UML) and a survey paper by von der Beeck [84] that discusses around twenty variants of Statecharts.

First we make the important observation that the language μ-Charts is, by design [78], assumed to be an extremely cut-down, and therefore simple, language based on the original Statecharts language. This decision was intentionally made (initially by Scholz) to allow a formal semantic investigation of the defined language, without the complication of considering an excessive number of language constructs. The investigation presented in this thesis does not go any further in terms of defining new "syntactic sugar" for the simplified language, instead the emphasis is placed on an extensive investigation of the refinement notion that is part of the language definition. The complication of considering the refinement notion for just the simplified language suggests that a formal investigation should indeed begin with this small core language. The idea being that once the core language is understood and has a suitable set of formal tools allowing us to reason formally about language design decisions then the core language can be extended by a series of conservative extensions towards something that may, or may not,

allow all of the constructs that are part of the language *Statecharts* and its associated variants. This is the point at which all of the useful practical considerations that have been made for these languages can be considered for the future development of μ-Charts.

We note that one essential extension that will be needed, in order that the core language μ-Charts scales to allow specification of realistic reactive systems, is the addition of local variables, along with the associated notions of assignment and variable-dependent guards. Work to this effect can be found in [7]. Also, [29] describes an approach that may be taken towards constructing other high-level language constructs.

An attribute that is common to Statecharts-like languages is the duality between "states" and "charts" that commonly occurs when describing the behaviour of a Statechart. We note that the description of Statecharts given in [35] deals with this duality by considering all "charts" as being themselves states; one of which is denoted the *root* state and therefore could be considered "the" Statechart. In Statecharts-speak [35], μ-Charts does implement the three core language constructs of Statecharts, which are: *OR-states*—a state of a "chart" that is itself a "chart" description of behaviour, *i.e.* a decomposed μ-chart; *AND-states*—"states that have orthogonal components that are related by *and*" [35], *i.e.* composed μ-charts; and basic states—states that contain no substates, *i.e.* sequential μ-charts.

Statecharts have several additional features that are not part of the language μ-Charts presented here (referred to simply as μ-Charts henceforth). We briefly discuss some of these differences in the following.

The label on a transition in a Statecharts state has the form $e[c]/a$, where e is an *event* that triggers the transition, c is a *condition* that must evaluate to true before the transition can happen, and a is the *action* that happens when the transition is taken. μ-Charts implements transition triggers whose truth value depends just on an evaluation of the signals that are present (and absent) at the moment that a step is evaluated. In the μ-Charts there are no conditional statements, at least in the sense of the expression $[c]$ of the Statecharts transition trigger above, because there is no state-based information on which such a condition could be evaluated. However, the previously mentioned work of [7] adds a similar conditional expression language to the transition triggers of μ-Charts. Also, Statecharts implements more sophisticated events, for example, the special event $timeout(e, d)$ occurs d time units after the most recent occurrence of the event e.

Similarly, the transition *actions* implemented by Statecharts allow a more

expressive language of actions, for example, allowing a transition to update local variables and more sophisticated actions like *schedule*(a, d)—action a can be scheduled for d time units later.

Statecharts allows *activities* to be associated with states, for example, *static reactions* take the form of a transition label but "are carried out (whenever enabled) as long as the system is in (and is not exiting) the state in question" [35]. *Static reactions* are described as syntactic sugar that are described semantically by orthogonality. Other state-based *activities* can, however, be explicitly associated to be active *throughout* state S or active *within* state S.

Statecharts allow transitions to cross state boundaries, that is, a transition can go directly from a state in one chart C to a particular state (or states) of a subchart (or substate) of chart C. Related to this, Statecharts defines several types of transition connectors, for example *joint*, *fork* and two types of *history* connectors, that allow the user to specify different ways to join transitions. *fork* allow a transition to split to have multiple destination states (in an orthogonal substate). Two transitions can *join* together to have an equal destination in the parent of an orthogonal substate. And a transition can enter the history connector for a substate which is taken to mean that the last state, in which the substate resided, is entered.

The differences discussed so far appear to be additional functionality that could be defined as conservative extensions to the language μ-Charts. However, more significantly there is a fundamental difference between the semantics of Statecharts and the semantics of μ-Charts. The step semantics of μ-Charts is based on the *perfect synchrony hypothesis* as discussed. Importantly a μ-Charts step is restricted so that each chart (or in Statecharts-speak, each state) must make one and only one transition. Also, the time between instantaneous transitions is not necessarily uniform. The Statecharts semantics, according to [35], has two models of time. The first is named the *asynchronous model* and obeys the perfect synchrony hypothesis. However, in this *asynchronous model* any chart (or state) can make several transitions in a single step—until there are no longer any valid transitions left–and is therefore different to the μ-Charts semantics. The second model of time is name the *synchrony model* and does not obey the *perfect synchrony hypothesis*. Therefore, the Statecharts semantics and the μ-Charts semantics must be fundamentally different.

The obvious question of whether the different semantic models of Statecharts and μ-Charts can be defined in terms of one another is outside of

the scope of this thesis. To prove this hypothesis will require defining both semantic models in the same formal framework, or inter-defining them.

1.3 Z and $\mathcal{Z}_C$

The specification language Z is based upon typed set theory and first-order predicate calculus and includes the notion of schemas used to encapsulate mathematical objects and their properties by declaration and constraint. Z is a model-based notation usually used to represent abstract specifications of systems by describing observations of their state and some operations that can change that state. One of many (see [36, 49, 68, 71, 81, 88, 90]) examples of using Z is given in [8].

Though Spivey [80] demonstrated the use of a logic for Z, the first attempt to give a provably sound logic for Z was the account of the Z-logic $\mathcal{W}$ by Brien and Woodcock in [89]. Martin (incorporating changes due to the criticisms of Henson [37]) gave a logic for standard Z in [54]. An international Z standardisation effort was completed in 2002 (see [47] and [82]). This includes a denotational semantics for Z but not the proposed standard logic for Z; to reason about "standard" Z specifications requires reasoning at the semantic level.

Henson and Reeves [38] provide a rational reconstruction of Z that they called $\mathcal{Z}_C$ (standing for "Z core"), which provides a sound logic for Z. This is the foundation of the Z-logic that we use to derive a logic for μ-Charts.

1.3.1 $\mathcal{Z}_C$

The language $\mathcal{Z}_C$ is a typed set theory that includes *schema types* which describe unordered, label-indexed tuples called *bindings*. The logic given for $\mathcal{Z}_C$ (see [38]) is presented as a natural deduction system. A proof of the soundness of the logic is given by supplying an interpretation of $\mathcal{Z}_C$ in ZF, where schemas are considered to represent sets of bindings. The schema calculus of traditional Z is then reconstructed by conservative extensions to $\mathcal{Z}_C$. The rules of the logic were developed by taking all of the usual rules for set theory and adding rules for schemas. An overview of the $\mathcal{Z}_C$ logic that is prevalent throughout the following is given in Appendix A.

Now, given a sound logic for $\mathcal{Z}_C$, correct reasoning about $\mathcal{Z}_C$ specifications, and therefore Z specifications, can be carried out in $\mathcal{Z}_C$ using rules at the level of the language itself. This has several advantages among which is

the ability for a user of $\mathcal{Z}_C$ to reason about a specification syntactically, *i.e.* without needing to know its semantics. After reconstructing Z from a formal foundation Henson and Reeves go on to develop refinement and program derivation rules for $\mathcal{Z}_C$.

1.4 Refinement

The principal idea of formal refinement is to allow program development which, by definition, guarantees that an implementation respects the behaviour described by a formal specification.

In Section 1.4.1 we describe some of the historical research that focused on giving meanings to programs and which led to the idea of program development via stepwise refinement. Section 1.4.2 deals specifically with refinement as it relates to the Z specification language, that is, we describe the context of the refinement notion that is employed in this work.

1.4.1 From proving programs to stepwise refinement

As early as 1966, Naur [61] described the lack of influence of mathematics and proof in the way that computer programs are constructed. He illustrated a method of proving the correctness of algorithms by taking snapshots of the state on which the algorithm is being executed. These snapshots allow the description of static properties that exist whenever the execution of an algorithm reaches a particular point and therefore allows us to prove that an algorithm respects the expected behaviour (that it describes) correctly.

In 1967, Floyd [25] published work on giving meanings to programs. Floyd showed that we could give a proper meaning to a program by describing, for each statement of a program, a relation that is true of the program variables or program state before an operation and a similar relation after the operation. Together these relations give a mathematical model that exactly defines the meaning of program statements and allows the proof of program properties. This was demonstrated by annotating flow charts with assertions.

Closely following this work, Hoare [40] gave a set of axioms and rules of inference that can be used, following methods first applied in the study of geometry, *i.e.* axiomatic methods, to prove the correctness of programs. This paper presented the axiomatic system as a way to prove properties of existing programs, and it gave the foundation which can be seen to be one of the bases for formal methods of program development as it is known today.

This axiomatisation of computation led to the idea of axiomatic semantics for programming languages, as opposed to the traditional operational semantics, and Hoare later described using the proof of a program in conjunction with the program's construction as a sound program development method. A particularly insightful comment in [40] describes how the proof of a program in a machine-independent axiomatic system can illustrate machine-dependent features of an algorithm and alleviate the burden of porting a program from one platform to another, a problem which is to this day being addressed by the software development community.

In [23], Dijkstra introduced the notion of predicate transformers and gave the *weakest precondition* meaning to a small programming language. Predicate transformers are defined to give the weakest precondition for which a program will terminate and satisfy a stated postcondition. He also shows how predicate transformers can be used as a calculus to aid in program development. While Floyd introduced the idea of invariant based programming, *i.e.* defining the invariant property for loops in the program, and Hoare emphasised its usefulness, Gries [30] states that it was Dijkstra [23] who presented the first useful technique for developing loop invariants before loops.

These early works, together with others not mentioned and all that followed, made it possible to give a proper, *i.e.* not operational, semantics to programs. It now seems an obvious progression that leads to abstract specification followed by provably consistent implementation.

In the early 1970s, Dijkstra [22] and Wirth [86] proposed the idea of stepwise refinement as a way to develop programs. Dijkstra [22] describes stepwise refinement to be the cognitive composition process of a well-structured program, given in great detail. In particular he points out the advantages of allowing the programmer to make one decision at a time and to be able to leave other choices to a later stage of refinement. This work introduces the inspirational idea of using the proof of a program in conjunction with its creation as a way to ensure correct program construction. Dijkstra described this idea as stemming from the formalisation of the process undertaken by a programmer whilst convincing themselves of program correctness during an informal development of code.

This notion is acknowledged in similar work by Hoare mentioned previously. Importantly Dijkstra identifies notions of data refinement, where the programmer must make more concrete the data structures to be used to store the program's data, and operation refinement, where the programmer refines the actions that are taken on that data. Wirth [86] also describes by example

the process of developing a program by a sequence of refinement steps where at each step one or several instructions are decomposed into more detailed instructions. Wirth also makes the distinction between data refinement and program refinement and states that it is natural to refine program and data specifications in parallel. Wirth's treatment of stepwise refinement is much less formal than that of Dijkstra.

In the mid 1970s, Burstall and Darlington [11] and Gerhart [26] identified a trade off between writing comprehensible programs and efficient programs. This lead them to introduce a *transformational* approach to programming. They give rules for transforming inefficient programs into more efficient programs correctly, thus allowing a programmer to write their program without considering efficiency concerns but rather with an eye to eloquence and understandability.

In [3], Back combines the ideas of stepwise refinement, the program transformation approach and the invariant-based approach to program construction to define the Refinement Calculus. The Refinement Calculus is based on earlier work by Back in which he added the notion of a specification statement to Dijkstra's Guarded Command Language. Hence the Refinement Calculus then allows the specification of a program to be given by possibly nondeterministic program statements that are not necessarily executable. Also the calculus defines rules that allow the (possibly non-executable) specification to be transformed, stepwise, into a program that is necessarily executable and by definition correctly refines the initial specification. Back describes the notion of correct refinement by a satisfaction relation. This relation allows the definition of correct refinement to be as follows:

> A program S is said to be correctly refined by another program S' if S' *preserves the correctness of* S, in the sense that S' satisfies any specification that S satisfies.

Morgan [58] and Morris [60] also developed a Refinement Calculus. Morgan published a comprehensive guide to programming using the Refinement Calculus [59] in which he emphasises the view that all specifications and implementations are programs and that there is a refinement order between programs. Like Back's, Morgan's Refinement Calculus uses Dijkstra's Guarded Command Language, extended to include specification constructs, to write programs.

In [4], Back describes two differences between Morgan's and his own work. The first is that the Morgan's Refinement Calculus introduces different spec-

ification statements and the second is that the calculus presented in [59] reformulates the original refinement calculus in a way that is influenced by the Z specification language.

To summarise: this section described the conception of formal stepwise refinement. Firstly we described the inspiring work of researchers such as (in no particular order) Floyd, Naur, Hoare and Dijkstra, whose work, among many others not mentioned, made it possible to reason about programs using mathematical logic as opposed to the insufficient methods of testing the operational characteristics of programs. We then described a change in traditional thinking which introduced the idea of a satisfaction relation, as opposed to a stricter equivalence relation, between programs. This change led to the culmination of the notion of formal program semantics and the idea of a formal refinement calculus. Most importantly, the satisfaction relation allows the partial specification of a problem to be refined in manageable steps into a computable function. Presuming correct instantiation of the calculus rules, this function or algorithm is guaranteed to satisfy all functional constraints of the specification.

Clearly this overview acknowledges only a few of the core publications whose work contributed to the formulation of the notion of program refinement.

1.4.2 Specification and stepwise refinement

We start the second thread of the development of refinement with the conception of the Z specification language (see Section 1.3). The reasons that we divide the discussion in this way is because Z is used as the meta-notation for the semantics of μ-Charts given here, and Z was developed as a dedicated specification language for which it was later realised a notion of refinement was required [13]. In comparison, the Refinement Calculus notion of specification was derived from studying the transformations that allow successful implementation, as described in the previous section. The Refinement Calculus is not renowned for having an algebra of constructs that allows high level structuring of specifications (though it does in fact have sequential composition, demonic and angelic choice, if-then-else statements and loops as part of the programming constructs of the language—these along with the monotonic algebra could be used to structure and reason about specifications).

When the Z specification language was created the connection between the specification and the implementation was informal. In [80] Spivey gives a

semantics to Z using set-theoretic structures and describes a Z specification as giving a description (possibly a "loose" description) of the behaviour to which the eventual implementation must adhere, *i.e.* the Z specification describes a nondeterministic mechanism or abstract data type whose behaviour can be interpreted from the set-theoretic model given by the semantics. The onus is then on the implementor to ensure that the eventual implementation is one which satisfies the mechanism described by the specification.

Later, in [81] Spivey describes the notions of operation refinement and data refinement as separate design decisions that must be made during program development. Rules are given that allow the simplest form of operation refinement, which is to show that one operation (given in Z) is a refinement of another Z operation. He also points out that operation refinement must be extended in two ways to be useful in program development, one of which is adding programming constructs (which is not attempted), the other is data refinement. The simple operation refinement given reduces nondeterminism in the specification by weakening preconditions and/or strengthening postconditions while data refinement can be used to change the data structures described in the specification.

The precondition/postcondition approach described above can be considered *proof-theoretic* [20] because we use the predicates defining operations to show that one operation is a refinement of another, *i.e.* to define the refinement relation. Another approach is the *model-theoretic* approach, *i.e.* the refinement relation is defined in terms of the relationship between the set-theoretic models that give the semantics of the operations. This approach is suggested in [88] and [16]. Here the relations represented by a specification are lifted with the special element $\perp$, which represents undefined or nontermination, and totalised (turned from a partial relation to a total relation) to represent chaotic behaviour outside of the precondition. Now the data refinement part of refining operations is defined in terms of set containment, *i.e.* one operation is considered a refinement of another if and only if the lifted relation describing the first operation, *i.e.* its model, is a subset of the lifted relation of the original operation. In [20], it is shown that the proof-theoretic and model-theoretic approaches described are the same.

Again, no Z-based code introduction method is given in [88] or [16], though the program derivation process suggested in [88] is to perform data refinement as described above until a concrete Z specification is reached and then translate that specification into a Refinement Calculus program statement. This translation is based on work presented by King in [50] and later

extended by Cavalcanti and Woodcock [13]. [50] gives rules for translating Z specifications into a Z Refinement Calculus program statement from which implementations can then be derived. This paper gives a general rule for the translation and also attempts to utilise the structure provided by the Z specification calculus to derive more sophisticated rules of translation. Unlike King, Cavalcanti and Woodcock have defined a new language called ZRC that has a notation compatible with Z and encompasses many of the rules of the Refinement Calculus. More recently, this work has been extended to include executable commands from Dijkstra's guarded command language, like assignments, conditionals, and loops, and with reactive behaviour from CSP, including communication, parallelism, and choice, to define a new programming language known as Circus [87].

1.4.3 $\mathcal{Z}_\lambda$ and program derivation

The language $\mathcal{Z}_\lambda$ [39] is an extension of the language $\mathcal{Z}_\mathcal{C}$. $\mathcal{Z}_\lambda$ incorporates a notion of implementation of operation schemas into the logic for $\mathcal{Z}_\mathcal{C}$ in order to provide program derivation and refinement rules for the existing logic. The notion of implementation that is the basis of these rules is to model a specification as a set of *legitimate implementations*, *i.e.* the meaning of a specification is given by the set of all of the programs that correctly implement it. Now, given a program p and a specification U, the relationship where p correctly implements U is expressed as:

$$p \in U$$

where the meaning of this proposition is given by:[5]

$$[\![p \in U]\!]_i =_{df} [\![p]\!]_f \in [\![U]\!]_s$$

that is p correctly implements U when the meaning of p (a function in the λ-notation) is one of the functions that correctly implements the specification U.

Given this notion of implementation, refinement is defined simply as set containment, *i.e.* a specification U_0 is a refinement of a specification U_1 as long as the set of functions $[\![U_0]\!]_s$ is a subset of the set of functions $[\![U_1]\!]_s$. That is, we have:

[5]Note that the semantic functions, *i.e.* $[\![.]\!]$, are annotated to denote the presence of three different semantic functions, where $[\![.]\!]_i$ is the semantics of implementation, $[\![.]\!]_f$ is the semantics of functions, and $[\![.]\!]_s$ is the semantics of specification.

$$[\![U_0 \sqsupseteq U_1]\!] =_{df} [\![U_0]\!]_s \subseteq [\![U_1]\!]_s$$

Therefore a development from specification to implementation is a finite sequence of design steps or refinements where each nontrivial step eliminates some of the possible implementations of the previous specification. Each step in the development process is verified as a correct refinement using the rules of the logic for $\mathcal{Z}_\lambda$. The development ends when the set of implementations has been reduced sufficiently so that all remaining implementations are acceptable. If the set of implementations is empty then the developer must review the design steps, possibly all the way back to the original specification. In some cases the original specification itself may have been infeasible and therefore changes may be required of the specification itself.

Contrary to the other program derivation methods discussed, when deriving programs in $\mathcal{Z}_\lambda$ each step happens at the level of the specification language, therefore one must conceptualise a two-dimensional process of program derivation. One dimension consists of the sequence of refinements from abstract specifications to more concrete specifications (*i.e.* no program statements). This process often entails introducing more refined structure to the specification as well as reducing nondeterminism. For example, splitting an operation schema into the composition of more specific operation schemas. Another example is making the described data types more concrete, *i.e.* the activities associated with traditional data refinement. On the other dimension, at any point in the refinement sequence there may be a program that clearly implements part of the specification. For instance consider the refinement sequence $S_n \sqsupseteq \ldots \sqsupseteq S_1$ that results in the specification S_n that implements the specification S_1. Now assuming $S_n =_{def} S_{n.1} \fatsemi S_{n.2}$, where the specification $S_{n.1}$ represents a program (provable by the implementation relation), then we know what part of the eventual program will be, *i.e.* the implementation of $S_{n.1}$. This type of program development is possible because $\mathcal{Z}_\lambda$ refinement is monotonic and it allows the structure of the specification to be adopted in the implementation. The program derivation is documented, and shown correct, by a proof that includes the use of rules that are derived from the implementation relation.

The notion of refinement used in the $\mathcal{Z}_\lambda$ method of program derivation (*i.e.* subset of implementations) has been shown equivalent [20] to both the proof-theoretic and the model-theoretic notions of refinement introduced above.

1.4.4 Program development for μ-Charts

Scholz [77, 78] describes refinement rules for μ-charts that are applicable as syntactic transformations and are equipped with syntax-based context conditions. He argues that these properties allow a designer to apply a refinement rule providing its specific syntactic requirements are met and therefore the designer does not need to be aware of the formal semantics of μ-Charts to apply the rules. This highlights the emphasis Scholz puts on the practical considerations of using these rules in development.

The refinement rules given are categorised into *Behavioural Refinement* and *Interface Refinement* which are outlined in the following and discussed in detail in Chapter 5.

Behavioural refinement allows the refinement of the input/output behaviour of a μ-Charts specification. A specification S_2 is a behavioural refinement of a specification S_1 if and only if any observable behaviour of S_2 can also be observed of S_1. The set of possible inputs, or the input interface, of S_1 is equal to the input interface of S_2 and the output interfaces of S_1 and S_2 are also equal. Note this allows (and we would generally expect) S_2 to remove behaviour of S_1, therefore making S_2 more deterministic. In other words, behavioural refinement is the restriction of nondeterministic behaviour without changing the input/output interfaces for a specification. Of course, a chart is trivially a refinement of itself. Formally, behavioural refinement is expressed by the predicate,

$$[\![S_2]\!]_{io} \subseteq [\![S_1]\!]_{io} \wedge In(S_1) = In(S_2) \wedge Out(S_1) = Out(S_2)$$

where $[\![S]\!]_{io}$ denotes the trace semantics, with respect to the observable input and output behaviour, for the specification S. $In(S)$ denotes the input interface, *i.e.* the set of possible input signals, for S and $Out(S)$ denotes the output interface for S.

Interface refinement is the refinement of the input/output interfaces of a specification by the adding of new signals. Another way to consider interface refinement is that it allows a designer to describe new observable behaviour that was not described at the previous level of abstraction. In the Scholz semantics, a specification S_2 is an interface refinement of a specification S_1 if and only if the input and output interfaces for S_1 are proper subsets of the respective interfaces for S_2. Any behaviour in S_2, affecting signals present in the interfaces of S_1, is behaviour in S_1. In other words an interface refinement entails adding signals to the input and/or output interfaces without creating new I/O behaviour for previously existing signals. Formally, this can be

expressed by the predicate,

$$\{(i|_{In(S_1)}, o|_{Out(S_1)}) | (i,o) \in [\![S_2]\!]_{io}\} = [\![S_1]\!]_{io} \wedge \\ In(S_1) \subseteq In(S_2) \wedge Out(S_1) \subseteq Out(S_2)$$

where $s|_X$ denotes the restriction of elements (events) in the trace s to the signals in X.

These two notions of refinement are combined into one single notion of refinement. This formal definition now states a specification S_2 is a refinement of a specification S_1 (represented symbolically by $S_1 \rightsquigarrow S_2$) if and only if:

$$\{(i|_{In(S_1)}, o|_{Out(S_1)} \mid (i,o) \in [\![S_2]\!]_{io}\} \subseteq [\![S_1]\!]_{io} \wedge \\ In(S_1) \subseteq In(S_2) \wedge Out(S_1) \subseteq Out(S_2)$$

Two important properties that Scholz proves for the refinement rules given are transitivity and the monotonicity of refinement with respect to composition (Scholz chooses to call this *compositionality*—we prefer the term *monotonicity*). Transitivity allows stepwise refinement to be carried out, hence allowing incremental system development. Transitivity gives us, for arbitrary specifications S_1, S_2 and S_3, if $S_1 \rightsquigarrow S_2$ and $S_2 \rightsquigarrow S_3$ then $S_1 \rightsquigarrow S_3$.

Monotonicity ensures that the refinement of part of a system is also a refinement of the overall system. Scholz refers to this property as compositionality. To show the refinement rules are monotonic Scholz had to show that given arbitrary specifications S_1, S_2 and S_3, such that $S_1 \rightsquigarrow S_2$ then the composition of the μ-charts S_1 and S_3 is refined by the composition of the μ-charts S_2 and S_3.

In his conclusions Scholz states that his notion of refinement for μ-Charts is based on the restriction of nondeterministic behaviour. A refinement step in this framework is guaranteed to make a μ-Charts specification more concrete and never more abstract. The construction of the rules given was based on practical, rather than theoretical, considerations. Therefore the calculus presented is not complete in a mathematical sense. For example there exist pairs of specifications, say S_1 and S_2, such that S_2 is a legitimate refinement of S_1, however, S_2 cannot be derived from S_1, using the syntactic refinement rules given.

1.5 Contributions

The aim of this thesis is to provide and investigate a new semantics and logic for μ-Charts. We extend the logic to include a calculus that allows us

to reason about the structure of charts and refinement. We reiterate that the work presented has two motivations: to represent a complete description of the latest version of μ-Charts; and to report on the process of using an existing logical framework to induce a logical framework for μ-Charts.

We give an extensive description of the language μ-Charts that we assume. This is necessary because it is different from that presented by Scholz in [78]. Essentially, where Scholz chose to reduce the language description to a minimum by showing that the decomposition operator can be defined, as syntactic sugar, in terms of the composition operator, we have chosen to include each of the language operators, composition, decomposition and interface description, as semantically defined operators, the primary reason being that it is as easy to define the decomposition operator directly in the semantics as it is to introduce it in terms of composition. There are also semantic differences between the language definitions. The definition given here was guided both by what Scholz described as the language μ-Charts (recall there are several versions of this work also, *e.g.*, [62, 65, 78, 76]) and by the task of encoding the semantics in Z.

For the logic and refinement rules we closely follow the methodology introduced by Deutsch, Henson and Reeves. That is, we begin by formally modelling μ-Charts in an appropriate meta-language, $\mathcal{Z}_C$, and then use the logic of $\mathcal{Z}_C$ to derive a logic for μ-Charts. Because $\mathcal{Z}_C$ itself was developed in this fashion, *i.e.* via a model in ZF set theory, we have shown how a semantics and logic for μ-Charts can be constructed based on set theory. A chart's semantics is denoted by a set of bindings. Each binding represents an allowable transition of the chart.

Given this semantics we develop a calculus, in the form of introduction and elimination rules, that allows us to reason (via proof) over the structure of the language. That is, given a composite chart we can deduce properties of its parts, and given propositions about parts of a chart we can reason about a chart as the structured composition of those parts.

We extend the logic further to include rules that allow us to reason about the refinement of μ-Charts. Together, the rules of the logic and those for refinement give us a refinement calculus that is a basis for a formal framework allowing formal developments of reactive systems. Naturally, the refinement rules for charts are given in the same style and language (*i.e.* natural deduction-style introduction and elimination rules using the language $\mathcal{Z}_C$) as the core logic. The derivation of these rules (given in detail in Chapter 6) closely follows similar derivations of refinement rules for Z presented by

Woodcock and Davies in [88] and later in a similar presentation by Boiten and Derrick in [16]. We rely on the extensive work of Deutsch, showing how each of the existing refinement notions for Z—that appear to differ significantly—are in fact strongly related, to ensure that the result of our derivation gives a sensible notion of refinement. A significant difference between the refinement that we derive for charts and the derivations on which the refinement notion is based is that we allow refinements that change the input/output characteristics of the respective abstract and concrete specifications.

This extension to the traditional Z data refinement treatments is due to the notion of *interface refinement* (as opposed to the more traditional *behavioural refinement*) that was suggested by Scholz in [78]. We discuss this distinction in detail in Chapter 5. Apart from introducing this distinction, the detailed account of the refinement notion for μ-Charts serves another significant purpose. This account of refinement is given using a more traditional trace semantics approach. That is, we discuss the meaning of refinement in terms of the "traces of input/output behaviour" of charts. While we consider that a logic based framework is superior for reasoning about refinements, it is still the case that the more traditional trace semantic interpretation of reactive systems is a useful tool for describing what the provable refinements actually mean.

Part of the construction of the model for charts is the description of charts in Z; the semantics for charts is derived via the $\mathcal{Z}_C$ semantics of that Z. A side-effect of this process is that the presentation of the required Z in Chapter 3 provides a concise description of the Z-model, in general, for μ-Charts. This provides a definitive account of what in previous published work (formally considered a translation from μ-Charts into Z) was introduced via specific examples. This ties in another body of work that shows how traditional Z techniques and tools can be used to reason about μ-Chart specifications of reactive systems.

Finally, the "experimental" work presented is a brief investigation of the monotonicity properties of the resulting notion of refinement for charts presented in Chapter 7. We consider this a form of experiment because it is the first example of integrated proofs using the rules of the developed logic. The development of this chapter provided significant insight into decisions about the form of the rules of the logic.

1.6 Outline

Chapter 2 presents the syntax of μ-Charts. A textual syntactic definition is given in parallel with describing the graphical representation of the language. The textual syntax is defined formally using a context free grammar. The description of the syntax is broken down into sections based on the structuring operators of the language.

Chapter 3 introduces the formal semantics for μ-Charts. The structure of the semantic definition follows that of the syntactic definition. For each of the language constructs we give an informal description of the language, the Z that is used to model the construct and the introduction and elimination rules that follow from the Z model and form the logic for μ-Charts. Also, some general rules (that hold over all charts) are presented.

In Chapter 4, we introduce and contrast some possible trace semantic interpretations that can be assigned to the language. The definition of each of these trace semantic notions provides a necessary link between the natural traces of observable behaviour semantics for charts and the rule-based logic that is the core of this thesis.

Chapter 5 provides an extensive investigation of the refinement notion for μ-Charts in the traces-of-behaviour model that is common to such formalisms. This includes a lattice-based analysis of what refinement for μ-Charts means.

The derivation of a refinement calculus for μ-Charts is detailed in Chapter 6. The derivation of the calculus is presented for two of the trace refinement notions of Chapter 5. Along with the derivation this chapter contrasts the refinement for charts with more traditional notions of refinement for Z.

The important monotonicity properties of the resulting refinement calculus are outlined in Chapter 7. Finally, Chapter 8 summarises the contributions of this thesis and suggests future work that follows naturally from the investigation reported here.

Chapter 2

Introduction to μ-Charts

This chapter defines the syntax for the language μ-Charts. We present a textual syntactic definition in parallel with describing the graphical representation of the language. First we define the basic building block for μ-Charts specifications which are sequential charts. Then we introduce the syntactic operators that can be used to combine sequential charts into more sophisticated μ-charts. In particular we give the operators for allowing the *composition* of two charts, the *decomposition* of a chart, *i.e.* embedding an arbitrary chart into a state of a sequential chart, and the *interface operator* that allows a chart designer to specify the context of the chart, that is, the signals with which a chart interacts with its environment. Unlike the previous definition of the language by Scholz in [78], we introduce decomposition of a μ-chart as a language operator in its own right rather than defining it as syntactic sugar using composition. This was done because, from the outset, decomposition appeared to warrant a separate semantic treatment, though in the end we manage only to treat composition fully. A similar case can be made for the interface operator. Both of these operators are key to μ-Charts and therefore require a clear and distinct definition.

We formally define the textual syntax of the language using a context free grammar. The graphical syntax is presented using examples.

The set of all possible syntactically correct μ-charts is given by the set $\mu Charts$ as defined by the following grammar:

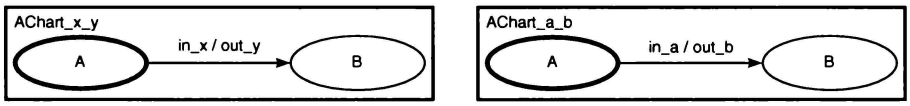
$$
\begin{aligned}
\mu Charts &::= Charts \mid (\ Charts\ ) \\
Charts &::= \mu Chart \mid \mu CompChart \\
&\quad \mid \mu DecChart \mid \mu Chartio
\end{aligned}
$$

In the following we give productions for the non-terminals $\mu Chart$, $\mu CompChart$, $\mu DecChart$ and $\mu Chartio$:

2.1 Sequential μ-Charts

The smallest unit of μ-Charts is a sequential μ-chart. In general a simple sequential chart is the specification of the allowable behaviour of a named finite state automaton that takes input from its environment and instantaneously produces output accordingly.

A sequential chart contains the name of the automaton, a nonempty set of control states, including a special start state (denoted by a double outline in the graphical notation), a feedback set (described in Section 2.2), and labelled transitions between states. For example, Figure 2.1(a) shows a sequential chart named *AChart_x_y*. It contains two control states *A* and *B*, of which *A* is uniquely identified as the initial state. Chart *AChart_x_y* contains one transition which has the guard *in_x* and action *out_y*. Informally we could describe this transition as follows: "if the chart *AChart_x_y* is in state *A* and the current input event contains the signal *in_x* then we instantaneously move to control state *B* outputting signal *out_y*".



(a) (b)

Figure 2.1: Examples of simple sequential μ-charts

Notice that we have also introduced a naming convention for charts. Due to the form of the name *AChart_x_y* it is understood that the chart is parameterised on the labels x and y. Because we never use or refer to the abstraction *AChart* itself, it is not necessary to define this mechanism in terms of usual abstraction, that is using variables, definitions and instantiations. Rather we assume the convention that, when given a chart definition such as *AChart_x_y*, we are free to consider the chart *AChart_a_b* with the understanding that it is defined as *AChart_x_y* where all occurrences of x are replaced by a and similarly y by b. The chart *AChart_a_b* is pictured in Figure 2.1(b).

To describe the attributes of charts formally, we introduce the following lexical classes: *Signal*, *Name*, *State* and *Param*, each of which denotes an abstract set of terminals that appear in the syntax grammar. These sets are assumed to contain all labels used to identify signals, chart names, control states, variables and parameters respectively.

We also introduce the sets *Chartname* and *μSignal* whose elements are defined using the following grammar:[1]

Chartname	::=	*class-name*[_*param-list*]
μSignal	::=	*signal-name*[_*param-list*]
param-list	::=	*Param*{_*Param*}
class-name	::=	*Name*
signal-name	::=	*Signal*

Now, consider the reactive systems that we might want or be able to specify using a μ-chart. The set *Chartname* is the set of labels used to name sequential charts. The set *μSignal* contains labels that name (possible parameterised) signals that these reactive systems can use to interact with their environment, and/or use for internal communication between their components.[2] Similarly, *State* is the set of labels used to name control states.

Each sequential chart contains a set of transitions. Each transition is a member of the set *μTransition* and is made up of a tuple of the form $(S_f, S_t, label)$ where S_f represents the state from which the transition originated, S_t represents the destination state of the transition and *label* defines under what conditions the transition can be taken and the resulting action of that transition. The label of the transition is an element of the set *Transition*. We define the necessary sets as follows:

μTransition	::=	(*State* , *State* , *Transition*)
Transition	::=	*guard* / *action*
guard	::=	*signal-expr* {**&** *signal-expr*}
signal-expr	::=	[**-**]*Signal*
action	::=	**{**[*signal-list*]**}**
signal-list	::=	*Signal* {, *Signal*}

Now we can define the set *μChart* of all allowable sequential charts:

---

[1]The context free grammar presented here uses the bracket notation [...] to represent the optional inclusion of the term enclosed in the bracket. The repetition construct {...}, *e.g.* {_*Param*}, means zero or more occurrences of the enclosed expression, *e.g.* _*Param*. Also, the tokens of μ-Charts are given in bold font, in particular the construct {...} is distinct from the language tokens '{' and '}' used for example to enclose a list of signals in a transitions action.

[2]Context permitting, we use the elements of each of these sets to refer to both the label itself and the object that the label represents.

$$
\begin{array}{lll}
\mu Chart & ::= & (\ \textit{Chartname},\ \textit{control-states},\ \textit{initial-state},\\
 & & \quad \textit{feedback-set},\ \textit{transitions}[,\ \textit{signal-list}]\ ) \\
\textit{control-states} & ::= & \{\ \textit{initial-state}\ [,\ \textit{states-list}]\ \} \\
\textit{initial-state} & ::= & \textit{State} \\
\textit{states-list} & ::= & \textit{State}\ [,\ \textit{states-list}] \\
\textit{feedback-set} & ::= & \{[\textit{signal-list}]\} \\
\textit{transitions} & ::= & \{\ [\textit{transition-list}]\ \} \\
\textit{transition-list} & ::= & \mu\textit{Transition}\ \{,\ \mu\textit{Transition}\}
\end{array}
$$

According to this definition, sequential charts, *i.e.* elements of the set $\mu Chart$ above, are described as a tuple that has the fields $(C, \Sigma, \sigma, \Psi, \delta)$ where:

- C is the name of the chart
- Σ represents the finite set of control states
- σ names the initial state, noting $\sigma \in \Sigma$
- Ψ represents the finite set of feedback signals
- δ represents the finite set of transitions that are defined in the chart

Notice that every sequential chart must have at least one state (the initial state) but can have empty sets of feedback signals and transitions.

Each sequential chart has an input and output interface. These interfaces represent the sets of signals that the chart can react to and output respectively. From the definition of a chart (either textual or graphical) we can typically derive these interfaces from the signals that appear in the triggers of transitions and the signals that appear in the actions of the transitions respectively. Henceforth, we refer to these derived interfaces as the *natural input/output interface* of a chart. We do so because it proves useful if the designer of a chart can specify the interfaces of a chart that differs from the natural interface. The interface definition operator is described in Section 2.5.

There is a clear relation between the textual form of the syntax that is presented here and the graphical presentation of sequential charts, for example Figure 2.1(b). The chart name is presented in the upper-left corner of the chart and each of the control state are labelled ovals *etc.* However it is worth noting that the textual definition of charts does not suppose to encode any spatial layout of a graphical chart. Hence, for any "textual chart" there are a great number of equivalent but different looking "graphical charts".

Though we have not yet discussed how the input and output of a chart can interact, it is possible for a sequential chart to contain a feedback loop that causes some of the output of the chart to be instantaneously fed back and act as input. As introduced, the description of sequential charts contains a feedback set Ψ. This is used to specify which output signals can be instantaneously fed back and hence act as input. The following section introduces sequential charts with feedback.

2.2 Feedback signals

The feedback set in sequential charts allows us to model the instantaneous feedback of designated output signals in sequential charts. For example the chart $(AChart\_a\_b, \{A, B\}, A, \{in\_a, out\_b\}, \delta)$ for $\delta = \{(A, B, in\_a/out\_b)\}$ is pictured in Figure 2.2.

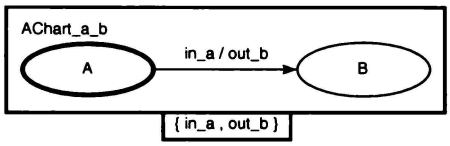


Figure 2.2: Sequential chart $AChart\_a\_b$ with feedback set $\{in\_a, out\_b\}$

If a signal is in the feedback set then whenever that signal is output by the chart it is also instantaneously available as input. If we consider the input/output behaviour that a sequential chart describes as a black box that has a wire carrying input and a wire carrying output then the feedback of signals can be considered as a loop-back wire that connects the output to the input and carries only the signals designated by the feedback set.

In the graphical representation of charts the feedback set is presented in a box attached to the outline that encloses the chart. Notice that the diagrammatic representation, *e.g.* Figure 2.2, allows the shorthand which is to omit the feedback box to represent a symbolic chart with the empty set in place of feedback Ψ. On the other hand Figure 2.2 shows a sequential chart $AChart\_a\_b$ that feeds back all of its signals, *i.e.* it has the feedback set containing in_a and out_b.

It is also worth noting that, in a fashion similar to CSP, the output signals that are feedback are not hidden from the environment, *i.e.* they can still control the environment like normal output.

2.3 Composition of μ-charts

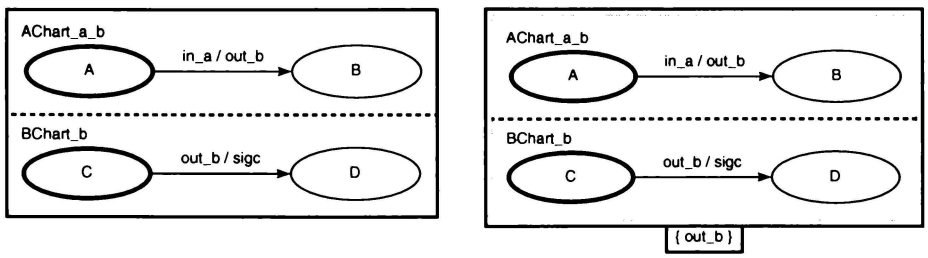
Now that we have introduced the basic building block of the language, *i.e.* sequential charts, we present the operators that can be used to combine arbitrary charts into more sophisticated μ-charts.

Given any two charts $C_1 \in \mu Charts$ and $C_2 \in \mu Charts$ and a set of signals Ψ, we denote the composition of these charts in parallel by an expression of the form $C_1 \mid \Psi \mid C_2$. The set of all such charts formed using the composition operator is defined by the set $\mu CompChart$:

$$\mu CompChart \quad ::= \quad \mu Charts \mid \{\ [signal\text{-}list]\ \} \mid \mu Charts$$

From this definition we can see, as one might expect, the feedback of signals is intimately coupled with the composition of charts. When you compose two charts you need to specify how they interact; because of this, the feedback set is included in the composition operator. The feedback set can be empty, in which case the composition would represent two charts that proceed in lock-step but do not affect one another in any way.

The graphical notation, like the case for sequential charts, uses two conventions for feedback between composed charts, *i.e.* a composed chart with no feedback box represents a composition with the empty feedback set. Consider the two composed charts in Figure 2.3.



(a) (b)

Figure 2.3: Composed charts with and without feedback

The chart in Figure 2.3(a) represents the chart $AChart\_a\_b \mid \{\} \mid BChart\_b$ where $AChart\_a\_b$ is $(AChart\_a\_b,\ \{A, B\},\ A,\ \{\},\ \{(A, B, in\_a/out\_b)\})$ and $BChart\_b$ is similarly defined. The chart of Figure 2.3(b) represents the chart $AChart\_a\_b \mid \{out\_b\} \mid BChart\_b$.

2.4 Decomposition of a μ-chart

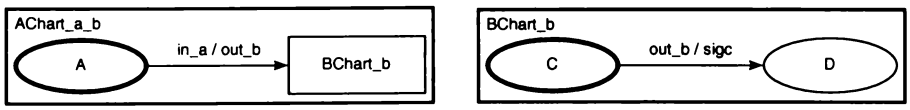
A decomposed μ-chart is a sequential chart that has other charts embedded into some or possibly all of its states. The purpose of decomposing a state in a sequential chart is to mimic a master/slave relationship between the charts. The sequential chart that has its states decomposed can be considered the master in the relationship because it effectively allows the decomposing chart, *i.e.* the slave, to react to input from the environment only when the master is in the state that the slave decomposes. Of course a master can have several slaves. A slave, however, has only one master which is always a sequential chart. The master may itself be a slave, *i.e.* there can be more than one level of decomposition.

The syntax for decomposition is defined by the following grammar:

$$\begin{aligned} \mu DecChart &::= \textbf{Dec } \mu Chart \textbf{ by } \{ \textit{ dec-states } \} \\ \textit{dec-states} &::= (\textit{ State } , \mu Charts) [, \textit{dec-states }] \end{aligned}$$

In the graphical representation of a decomposed chart the decomposed states of the master are written as rectangles (rather than the usual oval state representation). The slave chart is then given in a separate chart diagram. Figure 2.4 presents an example of a decomposed chart given by the textual expression

$$\begin{aligned} &\textbf{Dec } (AChart\_a\_b,\ \{A, BChart\_b\},\ A,\ \{(A, BChart\_b, in\_a/out\_b)\}) \\ &\quad \textbf{by } \{(BChart\_b\ ,\ (BChart\_b,\ \{C, D\},\ C,\ \{(C, D, out\_b/sigc)\}))\} \end{aligned}$$



(a) Master (b) Slave

Figure 2.4: Simple decomposed μ-chart

Notice that unlike Statecharts the μ-chart that decomposes a state is always given in a separate diagram. This is due to the tool AMuZed [55] used to create the diagrams rather than being a significant property of μ-Charts. Drawing chart *BChart_b* inside of its enclosing state *BChart_b* would be an acceptable graphical presentation of a μ-chart.

2.5 The interface operator

We refer to the final chart operator that we introduce as the *interface operator*. As we discussed above each sequential chart has a natural input/output interface associated with it. That is, the signals that occur on the triggers of transitions together make up the input interface while the signals that appear in the actions of transitions form the natural output interface.

However, it is often the case that the natural interfaces of a chart are not those that are intended by the designer. The most common situations where explicit interfaces are required is when a chart designer wishes to hide signals. A complex specification is (in general) constructed by combining several sub-charts using the operators that we have already introduced in this chapter. We can consider each sub-chart as sharing an observable interface with its environment. For example consider the composition (with feedback) of the sequential charts *AChart_a_b* and *BChart_b* pictured in Figure 2.3(b). The "effective" environment, *i.e.* what a chart sees and allows to be seen, for the respective charts *AChart_a_b*, *BChart_b* and $AChart\_a\_b \mid \{out\_b\} \mid BChart\_b$ are all different things. It is usually the case that we wish to distinguish between the way that one part of the specification, *e.g.* *AChart_a_b*, interacts with its environment and the way that the overall specification, *e.g.* $AChart\_a\_b \mid \{out\_b\} \mid BChart\_b$ interacts with its environment. In this example we may want the signal *out_b* to be used only for internal communication between *AChart_a_b* and *BChart_b*. Hence the interface that these charts share individually must contain signal *out_b*, however we want to hide this signal from the observable interface of the specification $AChart\_a\_b \mid \{out\_b\} \mid BChart\_b$. This can be achieved using the interface operator.

Another way that the interface operator can be used is to specify that the output interface of the chart is larger than its natural output interface. The reason a designer would wish to do this becomes clearer when we consider the entire definition of the language, in particular the refinement calculus for μ-Charts. In brief, the μ-Charts language makes the assumption that the output interface of a chart (whether left unspecified, *i.e.* the natural interface, or explicitly defined) specifies the context in which the chart designer expects the reactive system to reside. The refinement calculus for charts allows us to extend this output context using refinement. Such refinements allow new output behaviour over the added signals of the new context. Hence, specifying an output context that is larger than the natural output interface of the

chart allows a designer to indicate that a chart chooses not to output some signals that control its environment.

As we have seen, there are two interfaces for any chart, the input interface and the output interface. The interface operator allows us to explicitly specify either or both of these sets of signals. Restricting the input interface, *i.e.* defining an explicit input interface that is a subset of the natural interface, can be considered as filtering input from the environment, that is, these filtered signals are never received as input from the environment. Restricting the output interface means the hidden signals, *i.e.* signals that are in the natural output interface but not in the explicit output interface, are not observable as output from the specified reactive system.

The set of syntactically correct charts that contain hiding are given by $\mu Chartio$ which is defined as follows:

$$
\begin{array}{lcl}
\mu Chartio & ::= & {}_{[inputI]}\,[\ \mu Charts\ ]\,_{[outputI]} \\
inputI & ::= & \{\ [signal\text{-}list]\ \} \\
outputI & ::= & \{\ [signal\text{-}list]\ \} \\
signal\text{-}list & ::= & Signal\ [,\ signal\text{-}list]
\end{array}
$$

Notice that both the explicit input interface (on the left) and the explicit output interface (right) are optional. This allows the obvious shorthand notation such that $_X[C] =_{def} {}_X[C]_{out_C}$ and $[C]_Y =_{def} {}_{in_C}[C]_Y$.

The graphical notation includes the explicit definition of interfaces in the box attached for feedback. The feedback box for a chart that employs the interface operator can be written as "*in = input-interface* [*feedback-set*] *out = output-interface*". For example the chart ${}_{\{in\_a\}}[AChart\_a\_b \mid \{o\_b\} \mid BChart\_b]$ pictured in Figure 2.5 hides the signal *o_b* from the input interface of the chart. This means that the only way that chart *BChart_b* can be affected by the input signal *o_b* is if that signal is output from the chart *AChart_a_b*. Note, however, that the signal *o_b* is still observable output from this specification.

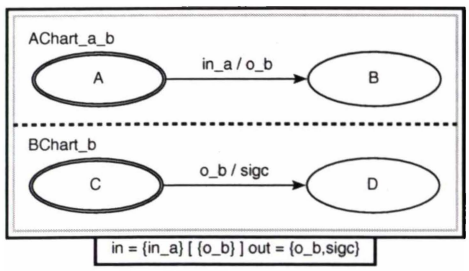


Figure 2.5: Example use of the interface operator

Again, we define some common shorthand notation that is useful when using the graphical notation for charts. Given in_C and out_C are the respective natural input/output interfaces for some chart C then we are free to use the following shorthand conventions:

$$\begin{aligned}
in = \iota \,[\, \Psi \,] &=_{def} in = \iota \,[\, \Psi \,]\, out = out_C \\
[\, \Psi \,]\, out = o &=_{def} in = in_C \,[\, \Psi \,]\, out = o \\
\Psi &=_{def} in = in_C \,[\, \Psi \,]\, out = out_C \\
in = \iota \;\; out = o &=_{def} in = \iota \,[\, \varnothing]\, out = o
\end{aligned}$$

where Ψ, ι, and o represent arbitrary sets of signals.

2.6 μ-chart names

We often find that we wish to refer both to a chart's name and the chart itself using the same identifier. When the context does not make this distinction precise we use the function ω that has the following signature.

$$\omega : Chartname \nrightarrow \mu Charts$$

We assume that ω is defined such that when it is applied to some chart name it returns the full symbolic definition of the chart with the same name. Also, we will often introduce a complex μ-Charts specification and give it a simpler name, for example the expression $C_{12} = C_1 \mid F \mid C_2$ defines the function ω such that $\omega\ C_{12} =_{def} \omega\ C_1 \mid F \mid \omega\ C_2$. Again context permitting we use the name C_{12} to refer to the chart $\omega\ C_{12}$.

Chapter 3

Language Semantics

Our formal semantics for μ-Charts is given by a method for constructing a Z model for an arbitrary chart. The kernel logic $\mathcal{Z}_C$ for Z [38], introduced in Appendix A, gives the Z model a meaning grounded in typed set theory. Hence, the combination of these gives us a semantics for μ-Charts grounded in typed set theory.

As we have already seen a μ-chart, specifying the behaviour of a reactive system, can be (and in most interesting cases is) made up by using language operators to combine simple sequential charts into more complex specifications. We present the semantics by giving an informal account of the meaning of the language constructs in parallel with describing the general method for creating the Z model and hence the formal semantics.

A core assumption of the semantics for μ-charts is that all constituent sequential sub-charts, that are combined to describe a reactive system, proceed in lock-step. When a step happens is determined by the environment in which the chart resides. That is, when the environment produces input each sequential chart makes a transition. Note that we distinguish between an input signal and input (or an input event), which is a set of input signals.

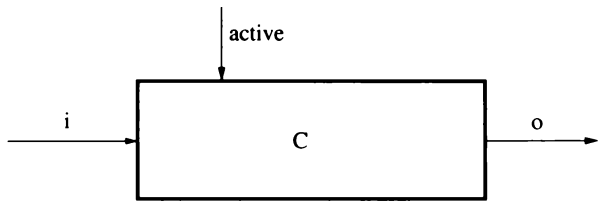
Initially, we give a step semantics for charts. This describes the behaviour of a chart in terms of the output that the chart produces in response to input from the environment assuming a given state. The step semantics essentially relates the current configuration of a chart[1] and input to a new configuration and the resulting output. This relation describes every possible step that a chart can take.

Later we show how this step semantics can be lifted to give a trace se-

[1]The term configuration is used to refer to the "state" of a chart because a μ-chart that has several sequential sub-charts can be in several states at the same time, that is, each sequential sub-chart has its own current state.

mantics that abstracts away the information about the chart's configuration. This gives the meaning of charts in terms of sets of observable input/output traces that an implementation of the specification can exhibit over time. The trace semantics is given (see Chapter 4) primarily to relate the account of μ-Charts refinement given here back to the previous work describing refinement [78]. The notion of chart refinement presented here is characterised in terms of the step semantics. Hence it is not strictly necessary to have the trace semantics at all, though the trace semantics helps to give a more intuitive understanding of the resulting refinement relation.

Building the Z model for charts requires two separate tasks. The first part of the process is to create a general model for each of the separate constituent sub-charts in the μ-chart being described. We call this the *transition model.* As one might expect given the language, this process is recursive in nature and follows exactly the structure of the chart being modelled. The overall transition model is built up by initially describing the model for each of the sequential sub-charts followed by the description of each of the new sub-charts created using the language operators. The final model describes the entire chart which is the specification of a reactive system. The modular nature of the language is such that the model of an arbitrary sub-chart can be considered as a "black-box". The following diagram demonstrates both the informal "circuit diagrams" that we use to motivate the semantic description and the "black-box" characteristics of the transition model for an arbitrary chart C.



The circuit diagram gives the structure of the general transition model for a chart. That is, we can consider the model of any chart with arbitrary structure as a circuit that has two inputs and one output. The chart model defines a relation between the two inputs and the output along with the associated change of state.

Sections 3.1 to 3.6 give both an informal description and a formal treatment that introduces the transition model for each of the language constructs. The formal treatment has two parts. The first gives a generic process for creating a Z model for an arbitrary μ-chart. The Z description of the chart can be used in the standard Z fashion to investigate the behaviour of the

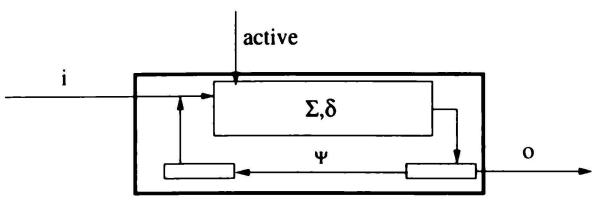
chart. Using available proof assistant and Z animation tools is a useful way to perform some validation of the μ-chart's behaviour. The second part of the formal treatment uses the meaning of the Z itself, via the Z-logic of [38], to give a logic for reasoning about μ-charts. This logic is presented in terms of introduction and elimination rules for each of the language constructs. It can be used to reason about the model of a μ-chart. The rules are presented in a natural deduction style, closely following the work of Deutsch, Henson and Reeves [20, 21].

The final step (Section 3.7) in building the model for a μ-chart is to hide the machinery that is present in the transition model, that is, the additional constructs required to model the inherent structure of charts. We refer to this model of a chart as the *step semantics* or *partial relations semantics* for a chart.

3.1 Sequential charts

A sequential chart is essentially a finite state automaton that describes the set of output signals that results from reacting to a set of inputs in a given state. The general transition model of a sequential chart has two notions of state, the first being the usual automaton notion of state which is determined by the transitions that have happened due to input since the automaton was initialised. This state is denoted in a chart diagram by labelled ovals or rectangles. The second notion of state represents whether or not the sequential chart is active. The reason that a chart can be active or inactive is because it may be the slave of another chart in the specification via decomposition. Finally, a sequential chart can instantaneously feed back output signals. This means that, during any step of the chart, fed back output is also considered as input.

The structure of the transition model of a sequential chart $(C, \Sigma, \sigma, \Psi, \delta)$ is demonstrated by the following circuit diagram.



This diagram can be separated into three logical parts as described above. The finite state automaton specification of the input/output relationship is

denoted by the component labelled Σ, δ. The active/inactive state mechanism is denoted by the input labelled *active*. And the feedback mechanism is represented by the line labelled Ψ.

Consider the chart $(C, \{A, B\}, A, \{\}, \delta)$ (where δ is the appropriate transition description) pictured in Figure 3.1.

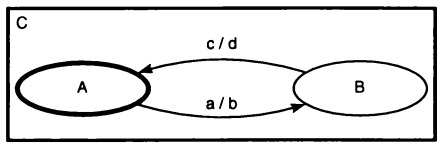


Figure 3.1: A simple sequential μ-chart

We can clearly see the finite state description of the input and output behaviour from this typical sequential μ-chart. Informally the behaviour that this chart captures can be described as: the chart starts in state A; if it is in state A and the signal a is input then signal b is output and the chart changes to state B; and similarly if it is in state B and c is input then d is output and the new state is A. Notice that we have not yet described what happens when the actions on the transitions are not satisfied. Hence the term *partial relations semantics*. We consider the meaning of a chart outside of its defined transitions after we introduce the Z semantics.

The ability of a sub-chart to be active or inactive is necessary for a chart that is embedded in the state of (*i.e.* a slave of) some other chart in a decomposition. When a chart is active it reacts to input as specified by its finite state description. When it is inactive it ignores input, remains in the same state and gives no output. In fact, no output actually means the output is the empty set of signals.

The behaviour of the feedback mechanism is uninteresting in this example because there are no signals fed back. However, in general, a sequential chart can feed back its output signals which then instantaneously act as input. This is represented in the circuit diagram by the link labelled Ψ that creates a loop between output and input and is assumed to carry only the specified feedback signals. We discuss in more detail exactly what feedback in sequential charts means semantically in Section 3.3.

3.2 The Z transition model for sequential μ-Charts

The essence of the general transition model for a sequential μ-chart is the description of each of its transitions using a separate Z operation schema. These operation schemas (one for each transition in the chart) are combined using Z schema disjunction to give one schema that describes the abstract transition behaviour of the chart. The Z state of the model has an observation that determines the current configuration of the chart. The operation schema describing a transition describes how and when that state changes. We proceed by introducing the Z that is given as the transition semantics of a chart and then discussing the meaning of that Z.

For a general sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$ we introduce the following definitions (left) that encode the chart's states, input interface and output interface in Z.

$$
\begin{array}{|l}
states_C : \mathbb{P}\, \mu_{State} \\
in_C : \mathbb{P}\, \mu Signal \\
out_C : \mathbb{P}\, \mu Signal \\
\Psi : \mathbb{P}\, \mu Signal \\
\hline
states_C = \Sigma \\
in_C = in\ C \\
out_C = out\ C
\end{array}
\qquad
Chart_C == [c_C : states_C]
$$

$$
\begin{array}{|l}
Init_C \\
\hline
Chart_C \\
\hline
c_C = \sigma_0
\end{array}
$$

The general state schema $Chart_C$ (top right) models the automaton state for the chart C. And the initial state of the chart is modelled by the schema $Init_C$ (bottom right).

A separate state schema is also given for each automaton state in the chart. So for all $\sigma \in \Sigma$ there exists a schema such that

$$C\sigma == [Chart_C \mid c_C = \sigma]$$

Now we give an operation schema for each chart transition. That is, for all $(S_f, S_t, guard/action) \in \delta$ we define an operation schema that has the following structure.

$\delta_{S_f S_t}$

CS_f
CS'_t
$i_C : \mathbb{P}\, in_C$
$act : \mathbb{P}\, \mu_{State}$
$o'_C : \mathbb{P}\, out_C$

$C \in act$
$\rho(guard)$
$o'_C = action$

Note that part of the definition of atomic charts—the observation *act*—allows for the definition of the decomposition operator. Here it is sufficient to understand that a chart can be active or inactive. Defined transitions only happen when an atomic chart is active, hence the predicate $C \in act$ is part of the precondition of the operation schema $\delta_{S_f S_t}$.

The predicate $\rho(guard)$, introduced in schema $\delta_{S_f S_t}$, stands for the Z predicate that models the syntactic guard of a chart transition. If we consider a transition's guard in general as a (possibly empty) list of signal expressions, separated by the conjunction symbol &, then each of the elements in the list can be classified into two categories: either a positive signal expression—simply the name of a signal; or a negative signal expression—the signal name is prefixed with a minus sign. A positive signal expression, say *sig* where $sig \in in_C$, is denoted by the Z expression $sig \in i_C \cup (o'_C \cap \Psi)$. A negative signal expression, say - *sig*, is denoted by the Z expression $sig \notin i_C \cup (o'_C \cap \Psi)$. The syntactic construction process denoted by ρ determines the appropriate predicate for each signal expression and connects them together using the Z logical conjunction operator $\wedge$. If the list is empty the predicate (produced by the process ρ) would be *true*. So, for the transition labelled a/b in chart C of Figure 3.1 the predicate produced would be $a \in i_C \cup (o'_C \cap \{\})$ since the guard of this transition is just a and there is no feedback associated with chart C.

This general scheme for giving the Z for a transition defines the semantic function $[\![.]\!]_{Z_t}$ such that for an arbitrary transition $(S_f, S_t, guard/action)$

$$[\![(S_f, S_t, guard/action)]\!]_{Z_t} =_{def} \delta_{S_f S_t}$$

where the schema $\delta_{S_f S_t}$ results from the method described above.

Along with the schemas for each transition we also need a single schema that models the behaviour of the chart when it is inactive. For the general sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, the inactive schema is given as follows.

$Inactive_C$

$\Xi Chart_C$
$i_C : \mathbb{P}\, in_C$
$act : \mathbb{P}\, \mu_{State}$
$o'_C : \mathbb{P}\, out_C$

$C \notin act$
$o'_C = \{\}$

The transition model for a sequential chart can now be given by the following definition.[2]

$$[\![(C, \Sigma, \sigma_0, \Psi, \delta)]\!]_{Z_t} =_{def} \delta_C$$

where

$$\delta_C == (\bigvee \{[\![t]\!]_{Z_t} \mid t \in \delta\}) \vee Inactive_C$$

This concludes the presentation of the Z that is used to give the transition model for a sequential chart. We now consider the meaning of this Z in terms of the particular example that is pictured in Figure 3.1.

The following Z schemas result from giving the transition model for this chart.[3]

δ_{AB}

CA
CB'
$i_C : \mathbb{P}\, in_C$
$act : \mathbb{P}\, \mu_{State}$
$o'_C : \mathbb{P}\, out_C$

$C \in act$
$a \in i_C \cup (o'_C \cap \Psi)$
$o'_C = \{b\}$

$Inactive_C$

$\Xi Chart_C$
$i_C : \mathbb{P}\, in_C$
$act : \mathbb{P}\, \mu_{State}$
$o'_C : \mathbb{P}\, out_C$

$C \notin act$
$o'_C = \{\}$

The transition in chart C from state A to B results in the Z schema δ_{AB}. Rather than describing the schema in terms of its syntax we proceed directly to describing the set $[\![\delta_{AB}]\!]_{Z_C}$; in the theory $\mathcal{Z}_C$ [38] the meaning of a schema is given as a set of bindings (see Appendix A). The schema δ_{AB} is denoted by the following set comprehension.[4]

[2]We use the notation $\bigvee X$ to denote the schema disjunction of all the schemas in the set X.

[3]We assume that all of the entities given in the presented Z whose definitions are omitted have the appropriate type and obvious definitions.

[4]The schema type T^a follows the type notation conventions outlined in Section A.2.

$$[\![\delta_{AB}]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C} =_{def} \{(\!|\, c_C \Rrightarrow A,\ i_C \Rrightarrow i,\ act \Rrightarrow active,\ c'_C \Rrightarrow B,\ o'_C \Rrightarrow \{b\} \,|\!) \mid i \subseteq in_C \wedge active \subseteq \mu_{State} \bullet C \in active \wedge a \in i\}$$

We can see from this definition that each binding in the semantic set has five labelled observations. The meanings of these are:

- c_C—the state of the chart before the transition happens, in this case the state A;
- i_C—the set of input signals which are offered by the environment that are in the input inteerface of the chart;
- act—a set that denotes all currently active charts;
- c'_C—the state of the chart after the transition happens, in this case state B;
- o'_C—the output generated by this sequential chart, in this case the set containing the signal b.

Hence each transition description is "parameterised" on the current configuration of the chart c_C, the input from the environment i_C and whether or not this chart is active. If the precondition of the schema holds, *i.e.* the chart is active and a transition's guard evaluates to true, then the output contributed by this sequential chart is defined by the set o'_C. Note that in the schema itself the expression $(o'_C \cap \Psi)$ models the feedback mechanism for the sequential chart. In this case $\Psi = \{\}$, that is the feedback set is empty. A similar schema to δ_{AB}, that models the transition from state B to A, called δ_{BA}, would also be given.

The second schema $Inactive_C$ describes the behaviour of the sequential chart when it is inactive. We can see from the following set of bindings that the schema $Inactive_C$ faithfully models the inactive behaviour of the chart as described above.

$$[\![Inactive_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C} = \{(\!|\, c_C \Rrightarrow s,\ i_C \Rrightarrow i,\ act \Rrightarrow active,\ c'_C \Rrightarrow s,\ o'_C \Rrightarrow \{\} \,|\!) \mid s \in \{A, B\} \wedge active \subseteq \mu_{State} \wedge i \subseteq in_C \bullet C \notin active\}$$

Now the complete transition model for chart C is defined as the disjunction of each of the individual transition schemas to be $\delta_C == \delta_{AB} \vee \delta_{BA} \vee Inactive_C$. By definition of schema disjunction the set $[\![\delta_C]\!]_{\mathcal{Z}_C}$ contains all of the bindings from the sets $[\![\delta_{AB}]\!]_{\mathcal{Z}_C}$, $[\![\delta_{BA}]\!]_{\mathcal{Z}_C}$ and $[\![Inactive_C]\!]_{\mathcal{Z}_C}$.

This method of investigating the semantics of the transition model for charts, that is, considering the meaning of schemas as sets of bindings, is somewhat informative in the simple example presented above. However, in general we do not want to resort to complex set definitions and manipulations to reason about the meaning of charts. Therefore we give a set of rules that characterise and allow us to reason about the Z semantic model for charts. These and other rules, presented later, give us a logic for charts.

First, we formalise what it means for a binding of the transition model to satisfy a specific (syntactic) transition of the chart. Given an arbitrary transition of the form $t = (S_f, S_t, guard/action)$, from the chart C, and assuming $[\![[\![t]\!]_{z_t}]\!]_{z_C}$ has type $\mathbb{P}\, T^a$, we have,

$$Trans\ t\ z^{T^a} =_{def} z.c_C = t.S_f \wedge \rho(t.guard)[\alpha T/z.\alpha T] \wedge z.c'_C = t.S_t \wedge z.o'_C = t.action$$

The terms $t.S_f$ *etc.* are assumed to be defined in the obvious way such that $t.S_f$ gives the "from state" of a transition, $t.guard$ gives the guard component of a transition, $t.S_t$ gives the "to state" and $t.action$ returns the action component. ρ is as defined above, *i.e.* it constructs an appropriate predicate from the transition's guard.

Now we give the formal definition of the transition model for charts directly in terms of the meaning of the Z model.[5]

Definition 3.2.1 For the arbitrary sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, we have,

$$[\![\delta_C]\!]^{\mathbb{P}\, T^a}_{z_C} =_{def} \{z^{T^a} \mid C \notin z.act \wedge z \mathrel{\dot\in} \Xi Chart_C \wedge z.o'_C = \{\} \vee C \in z.act \wedge \exists t \in \delta \bullet Trans\ t\ z\}$$

From this definition we derive the following introduction and elimination rules. Note that, while these rules may initially appear to be non symmetrical, when taken in conjunction with the rules of Proposition 3.2.2, they are symmetrical as expected.

Proposition 3.2.1 Given the sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, where $[\![\delta_C]\!]^{\mathbb{P}\, T^a}_{z_C}$, for arbitrary binding z^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad actv\ C\ z \quad t \in \delta, Trans\ t\ z \vdash P}{P}\ (Z_t{}^-)$$

$$\frac{actv\ C\ z \quad t \in \delta \quad Trans\ t\ z}{z \mathrel{\dot\in} \delta_C}\ (Z_t{}^+)$$

where $T^a \preceq T_3$ and assuming the usual conditions for t and P (due to the elimination of an existential quantifier).

[5]The notation $\dot\in$ is defined in Appendix A.2, page 172.

Proofs of these rules are given in Appendix B.1. For an atomic chart C the predicates $actv\ C\ z$ and $inactv\ C\ z$ are defined as follows:

$$actv\ C\ z =_{def} C \in z.act$$
$$inactv\ C\ z =_{def} \neg\ actv\ C\ z$$

We also introduce useful rules that allow us to reason about inactive charts. Eventually we show that these rules hold in general for charts with arbitrary structure.

Proposition 3.2.2 Given the sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$, for arbitrary binding z^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z \dot{\in} \Xi Chart_C}\ (iact_I^-) \qquad \frac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\ (iact_{II}^-)$$

$$\frac{inactv\ C\ z \quad z \dot{\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \dot{\in} \delta_C}\ (iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$ and $T^a \preceq T_3$.

Given these rules we can start to investigate some of the properties of the transition model. In particular, the following propositions allow us to reason about which bindings are in the transition model and which are not.

Firstly, we make the observation that the transition model given makes a distinction between two sources of input that can contribute to the signals that trigger a transition. The external input to the chart generated by the environment i_C and the fed back output from the chart itself, *i.e.* for the chart $(C, \Sigma, \sigma_0, \Psi, \delta)$ the set denoted by $o'_C \cap \Psi$. We formalise this observation in the following proposition.[6]

Proposition 3.2.3 For the arbitrary sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$ and bindings z^{T_3} and x^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \dot{\in} \delta_C}\ (Z_t^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

The term fb_z is a shorthand for the expression denoting the feedback that is applicable to a chart. That is, $fb_z = z.o'_C \cap \Psi$ in the case above.

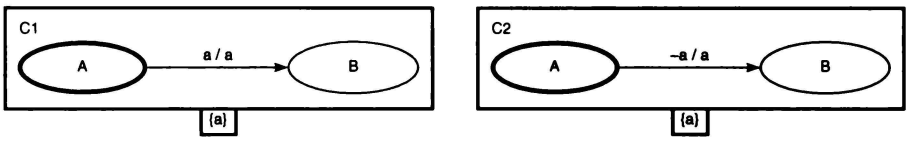
[6]We use the expression $T^a - V^i$ to represent the exclusion of the input observation i_C from the schema type T^a. This is equivalent to defining $T_i = U \curlyvee V^o$, except the former is more indicative of the notion of this proposition.

Here we give the rule $(z_t^{\in})$ just for sequential charts. In the following sections we prove the same rule for charts that are constructed using each of the chart operators. Together these proofs are an example of a proof by induction over the structure of the language. The proof for sequential charts here, provides the base case for the induction. Hence, eventually, we have that this property holds in general for charts.

3.3 Feedback of signals in μ-Charts

The semantics of feedback in sequential charts is encoded in the transition model as given in the previous section. In this section we give two pathological examples that demonstrate exactly the semantics of sequential charts with feedback.

The examples $C_1 = (C_1, \Sigma, \sigma, \{a\}, \delta_1)$ and $C_2 = (C_2, \Sigma, \sigma, \{a\}, \delta_2)$ (for appropriate Σ, σ, δ_1 and δ_2) are given in Figure 3.2. The only syntactic difference between these two μ-charts is that the transition in the chart C_1 is triggered by the presence of the signal a whereas the transition in chart C_2 is triggered by the absence of the signal a. Both examples output and feed back a.



(a) (b)

Figure 3.2: Pathological sequential charts with feedback

The encoding of feedback is present in the schemas that describe the respective transitions in these charts.

δ^1_{AB}

$C_1 A$
$C_1 B'$
$i_{C_1} : \mathbb{P}\, in_{C_1}$
$act : \mathbb{P}\, \mu_{State}$
${o_{C_1}}' : \mathbb{P}\, out_{C_1}$

$C_1 \in act$
$a \in i_{C_1} \cup ({o_{C_1}}' \cap \Psi)$
${o_{C_1}}' = \{a\}$

δ^2_{AB}

$C_2 A$
$C_2 B'$
$i_{C_2} : \mathbb{P}\, in_{C_2}$
$act : \mathbb{P}\, \mu_{State}$
${o_{C_2}}' : \mathbb{P}\, out_{C_2}$

$C_2 \in act$
$a \notin i_{C_2} \cup ({o_{C_2}}' \cap \Psi)$
${o_{C_2}}' = \{a\}$

Given $\Psi = \{a\}$, the respective predicates $a \in i_{C_1} \cup ({o_{C_1}}' \cap \Psi)$ and $a \notin i_{C_2} \cup$

($o_{C_2}' \cap \Psi$) mean that both of the transitions' guards rely not only on the environment's input, *e.g.* i_{C_1}, but also on the fed back output, *e.g.* ($o_{C_1}' \cap \{a\}$).

Now we can state and prove lemmas that describe exactly the meaning of the two examples.

Firstly, for the chart pictured in Figure 3.2(a), we show that the transition from state A to B happens regardless of the input from the environment. For simplicity we make the assumption that the charts are active.[7]

Lemma 3.3.1 Given $(C_1, \Sigma, \sigma, \Psi, \delta_1)$, where $\Sigma = \{A, B\}$, $\sigma = A$, $\Psi = \{a\}$ and $\delta_1 = \{(A, B, a/a)\}$, for arbitrary z^{T_3} and input $i \subseteq in_C$ we have,

$$\frac{actv\ C_1\ z \quad z \doteq (\!|\ c_{C_1} \Rrightarrow A, i_{C_1} \Rrightarrow i, c_{C_1}' \Rrightarrow B, o_{C_1}' \Rrightarrow \{a\}\ |\!)}{z \dot{\in} \delta_{C_1}}$$

where $[\![\delta_{C_1}]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$

Therefore, the transition from state A to state B always happens in the first step of the chart C_1, regardless of the input that the environment offers—the signal a is output and fed back. This also implies that this chart has identical behaviour to a similar chart in which the transition is labelled $/a$, *i.e.* it has an always true trigger.

Likewise, we show that the transition model for the chart C_2 pictured in Figure 3.2(b) does not contain any bindings, and therefore does not model any transitions from state A to B. This is because the single candidate transition in chart C_2 never happens in the presence of feed back. Essentially, the presence of signal a, instantaneously fed back from the output, means that the trigger $-a$ is always false. This is expressed by the following lemma.

Lemma 3.3.2 Given $(C_2, \Sigma, \sigma, \Psi, \delta_2)$, where $\Sigma = \{A, B\}$, $\sigma = A$, $\Psi = \{a\}$ and $\delta_2 = \{(A, B, -a/a)\}$, for all z^{T^a},

$$\frac{actv\ C_2\ z}{z \notin \delta_{C_2}}$$

where $[\![\delta_{C_2}]\!]^{\mathbb{P}\,T^a}_{z_C}$

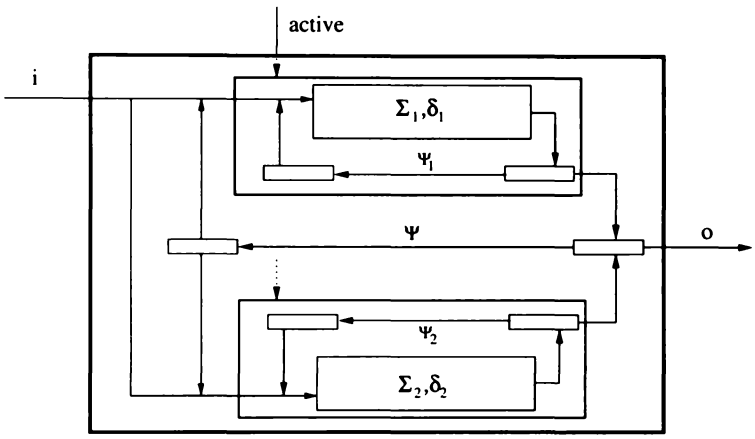
In this simple example the set of bindings, modelling the chart's transitions, is empty because the only candidate transition's guard is inconsistent with its own fed back output, and therefore it is not part of the transition model.

[7]The notation $\doteq$ is defined in Appendix A.2, page 172.

3.4 The composition operator

The composition operator allows us to take two μ-charts C_1 and C_2 and join them together to form a new more complex chart $C_1 \mid \Psi \mid C_2$ where Ψ is a set of signals. As mentioned, we assume that the charts run separately but synchronously, *i.e.* in lock-step with one and other. Their only medium of communication is asynchronous via the multicast of signals. The set Ψ denotes the signals that the charts C_1 and C_2 can use to communicate. The communication is asynchronous in that output is always enabled; a chart can always broadcast signals. However there is no guarantee that the other chart in the composition is listening (that is, ready to react on the signals broadcast). Signals persist only during one step of the chart.

The following diagram demonstrates the structure of the composed chart $(C_1, \Sigma_1, \sigma_1, \Psi_1, \delta_1) \mid \Psi \mid (C_2, \Sigma_2, \sigma_2, \Psi_2, \delta_2)$.



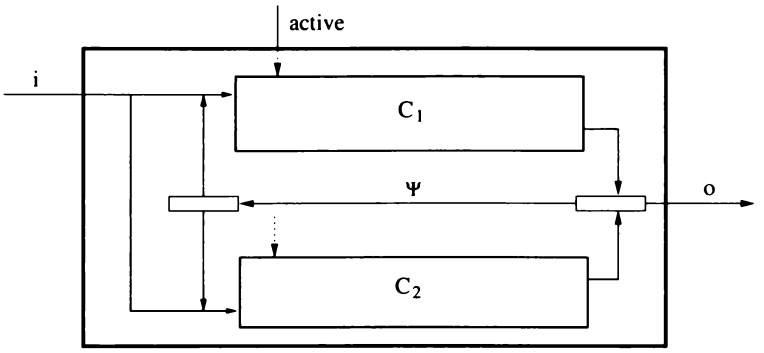
Notice that any signals that the sets Ψ and Ψ_1 (respectively Ψ and Ψ_2) have in common will be fed back on two separate paths in this diagram. Also, the composition operator not only allows C_1 to communicate with C_2 using the signals in Ψ but effectively changes the feedback characteristics of C_1 itself. The output that C_1 treats as input via feedback is now all of the signals in $\Psi_1 \cup \Psi$.

Both charts in the composition have an equivalent active state, *i.e.* one chart is active if and only if the other is active. In the circuit diagram we use dotted lines as a shorthand to indicate that the single active input to the composition is linked to both of the active inputs from the parts of the composition.

This example of the composition operator assumes that C_1 and C_2 are sequential charts. The operator in general, however, is defined over arbitrary charts. That the transition model of a composed chart takes the same type of inputs and produces the same type of outputs as that of a sequential chart

demonstrates the modular nature of the model; the model of a composed chart can be "plugged in" to replace one of the sequential charts in the above diagram.

The general picture for a composed chart $C_1 \mid \Psi \mid C_2$, where C_1 and C_2 are arbitrary charts, is as follows:



The transition model for composed charts is constructed by recursively constructing transition models for the two constituent parts of the composition. The base case being when the parts are themselves sequential charts. The models of the parts are then combined to create the transition model for the composition.

The transition model for the composed chart $C = (C_1 \mid \Psi \mid C_2)$ contains the following Z definitions and schemas. This Z makes the obvious assumption that any entity subscripted with C_1 comes from the transition model of the chart C_1 and similarly for C_2.

$$
\begin{array}{|l}
states_C : \mathbb{P}\, \mu_{State} \\
in_C : \mathbb{P}\, \mu Signal \\
out_C : \mathbb{P}\, \mu Signal \\
\Psi : \mathbb{P}\, \mu Signal \\
\hline
states_C = \\
\quad states_{C_1} \cup states_{C_2} \\
in_C = in_{C_1} \cup in_{C_2} \\
out_C = out_{C_1} \cup out_{C_2}
\end{array}
$$

$$
\begin{array}{|l}
\hline
Chart_C \\
Chart_{C_1} \\
Chart_{C_2} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
Init_C \\
Init_{C_1} \\
Init_{C_2} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\delta_C \\
\Delta Chart_C \\
i_C : \mathbb{P}\, in_C \\
act : \mathbb{P}\, \mu_{State} \\
o'_C : \mathbb{P}\, out_C \\
\hline
C_1 \in act \Leftrightarrow C \in act \\
C_2 \in act \Leftrightarrow C \in act \\
\exists\, i_{C_1}, i_{C_2}, {o_{C_1}}', {o_{C_2}}' : \mathbb{P}\, \mu Signal \bullet i_{C_1} = (i_C \cup (o'_C \cap \Psi)) \cap in_{C_1} \wedge \\
\quad i_{C_2} = (i_C \cup (o'_C \cap \Psi)) \cap in_{C_2} \wedge o'_C = {o_{C_1}}' \cup {o_{C_2}}' \wedge \delta_{C_1} \wedge \delta_{C_2} \\
\hline
\end{array}
$$

Importantly, the bindings that inhabit the set $[\![\delta_C]\!]_{z_C}$ (*i.e.* the semantics of the schema δ_C) have a similar signature to those in both $[\![\delta_{C_1}]\!]_{z_C}$ and $[\![\delta_{C_2}]\!]_{z_C}$. This shows that the Z model is consistent with the modular nature of charts demonstrated by the informal circuit diagrams.

As with sequential charts we give the definition of the transition model for composed charts directly in terms of the meaning of the Z model.[8]

Definition 3.4.1 Given an arbitrary composed chart $C = C_1 \mid \Psi \mid C_2$ we have,

$$
\begin{aligned}
[\![\delta_C]\!]^{\mathbb{P}\, T^a}_{z_C} =_{def} \{ z^{T^a} \mid\ & C_1 \in z.act \Leftrightarrow C \in z.act \wedge \\
& C_2 \in z.act \Leftrightarrow C \in z.act \wedge \\
& \exists\, o_1, o_2 \bullet z.o'_C = o_1 \cup o_2 \wedge \\
& \quad z \star (\!| \, i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}' \Rrightarrow o_1 \, |\!) \in \dot{\delta}_{C_1} \wedge \\
& \quad z \star (\!| \, i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}' \Rrightarrow o_2 \, |\!) \in \dot{\delta}_{C_2} \}
\end{aligned}
$$

where $fb_z = z.o'_C \cap \Psi$.

The following introduction and elimination rules for composed charts are derived from this definition. The proofs of these rules are given in Appendix B.3.

Proposition 3.4.1 Given $C = C_1 \mid \Psi \mid C_2$, for the binding z^{T_3} and arbitrary sets o_3 and o_4, we have,[9]

$$
\frac{\begin{array}{ll} & z.o'_C = o_1 \cup o_2, \\ z \in \delta_C & z \star (\!| \, i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}' \Rrightarrow o_1 \, |\!) \in \dot{\delta}_{C_1}, \\ & z \star (\!| \, i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}' \Rrightarrow o_2 \, |\!) \in \dot{\delta}_{C_2} \vdash Q \end{array}}{Q} \; (|\,-\,|^-)
$$

$$
\frac{\begin{array}{l} z.o'_C = o_3 \cup o_4 \\ z \star (\!| \, i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}' \Rrightarrow o_3 \, |\!) \in \dot{\delta}_{C_1} \\ z \star (\!| \, i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}' \Rrightarrow o_4 \, |\!) \in \dot{\delta}_{C_2} \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \in \delta_C} \; (|\,-\,|^+)
$$

where the usual conditions hold for o_1, o_2 and Q, $[\![\delta_C]\!]^{T_a}_{z_C}$ and $T^a \preceq T_3$.

[8]The binding, binding concatenation and typing notation used in the following definitions is defined in Appendix A.2, page 171.

[9]Note that there is an implicit typing constraint for rule $(|\,-\,|^-)$. The type T_3 must be disjoint with respect to each of the types V_1^i, V_1^o, V_2^i and V_2^o—the operands of the binding concatenation operator $\star$ must be of disjoint type.

The predicates $actv\ C\ z$ and $inactv\ C\ z$ are defined for the composed chart $C = C_1 \mid \Psi \mid C_2$ as:

$$actv\ C\ z =_{def} actv\ C_1\ z \wedge actv\ C_2\ z \wedge C \in z.act$$
$$inactv\ C\ z =_{def} inactv\ C_1\ z \wedge inactv\ C_2\ z \wedge C \notin z.act$$

Notice that the property $inactv\ C\ z \Leftrightarrow \neg\ (actv\ C\ z)$ does not hold in general for all bindings z in the type of the semantics of composed chart C. This is why the fourth assumption is required in the introduction rule—we cannot use the *law of the excluded middle* to show $actv\ C\ z \vee inactv\ C\ z$. However, we do have that the weaker property, $z \in \delta_C \vdash inactv\ C\ z \vee actv\ C\ z$ holds. This weaker property is stated and proved in Lemma B.3.3 of Appendix B.3. The proof relies on other properties (lemmas B.3.1 and B.3.2), which are interesting because they demonstrate more examples of rules that are (eventually) proved in general for any chart regardless of structure. Like for the rule $(z_t^{\in})$ of Proposition 3.2.3, this is achieved using *structural induction* over each of the μ-Chart operators where the base case is again that the property holds for sequential charts.

Now, from $(|\_|^-)$ and $(|\_|^+)$, we give other useful introduction and elimination rules. Again these rules are proved in general using structural induction.

Proposition 3.4.2 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary binding z^{T_3}, we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z \mathrel{\dot\in} \Xi Chart_C}\ (iact_I^-) \qquad \frac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\ (iact_{II}^-)$$

$$\frac{inactv\ C\ z \quad z \mathrel{\dot\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \mathrel{\dot\in} \delta_C}\ (iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Proposition 3.4.3 shows that the basic requirement of symmetry holds for the introduced composition operator.

Proposition 3.4.3

$$\frac{}{[\![\delta_{C_1|\Psi|C_2}]\!]_{z_C} = [\![\delta_{C_2|\Psi|C_1}]\!]_{z_C}}\ (|\_|_{sym})$$

Proposition 3.4.4 shows that the show property $(z_t^{\in})$ holds for composed charts.

Proposition 3.4.4 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary bindings z^{T_3} and x^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \dot{\in} \delta_C} \; (Z_i^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{Z_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

As expected, given the instantaneous feed back of signals in charts, whenever the fed back output from one part of the composition, say C_1, triggers a transition in the other part, say C_2, there is a corresponding binding in the transition model of the composition. This binding represents a transition of the composition that is triggered by the same input as the transition in C_1 and outputs the combination of the outputs from the respective parts. That this is the case follows directly from Proposition 3.4.4.

We give a concrete example of using this rule. Consider the following μ-chart.

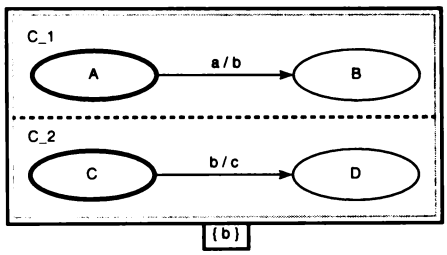


Figure 3.3: Composed μ-chart with feedback $\{b\}$

Given this chart we can easily show that

(3.1) $\langle\!|\, c_{C_1} \Rrightarrow A,\ c'_{C_1} \Rrightarrow B,\ i_{C_1} \Rrightarrow \{a\},\ act \Rrightarrow \{C_1, C_2\},\ o_{C_1}{}' \Rrightarrow \{b\} \,|\!\rangle \in \delta_{C_1}$

(3.2) $\langle\!|\, c_{C_2} \Rrightarrow C,\ c'_{C_2} \Rrightarrow D,\ i_{C_2} \Rrightarrow \{b\},\ act \Rrightarrow \{C_1, C_2\},\ o_{C_2}{}' \Rrightarrow \{c\} \,|\!\rangle \in \delta_{C_2}$

From (3.1) and (3.2) by $(|\_|^+)$ we have,

(3.3) $$\langle\!|\, c_{C_1} \Rrightarrow A,\ c_{C_2} \Rrightarrow C,\ c'_{C_1} \Rrightarrow B,\ c'_{C_2} \Rrightarrow D,\ i_C \Rrightarrow \{a, b\},\ act \Rrightarrow \{C_1, C_2\},\ o'_C \Rrightarrow \{b, c\} \,|\!\rangle \in \delta_C.$$

Now from (3.3) and Proposition 3.4.4, given that $\{a\} \cup \{b\} = \{a, b\} \cup \{b\}$, we can show that

(3.4) $$\langle\!|\, c_{C_1} \Rrightarrow A,\ c_{C_2} \Rrightarrow C,\ c'_{C_1} \Rrightarrow B,\ c'_{C_2} \Rrightarrow D,\ i_C \Rrightarrow \{a\},\ act \Rrightarrow \{C_1, C_2\},\ o'_C \Rrightarrow \{b, c\} \,|\!\rangle \in \delta_C.$$

also holds. This binding realises the behaviour of the composed chart C, which is: assuming initial states, on input a, both charts C_1 and C_2 make a transition. Clearly the transition in C_1 is triggered by the input a. The transition in C_2, however, is triggered by the instantaneous feedback of the output signal b.

This example demonstrates that the transition model essentially recursively "sequentialises" composed charts. By "sequentialise" we refer to the process of taking the cross product of all of the transitions in each part of the composition and creating a transition in the composition model that represents both of these transitions happening together. This process is recursive whenever the parts of the composition are not sequential μ-charts. Consider the following illustration of this process.

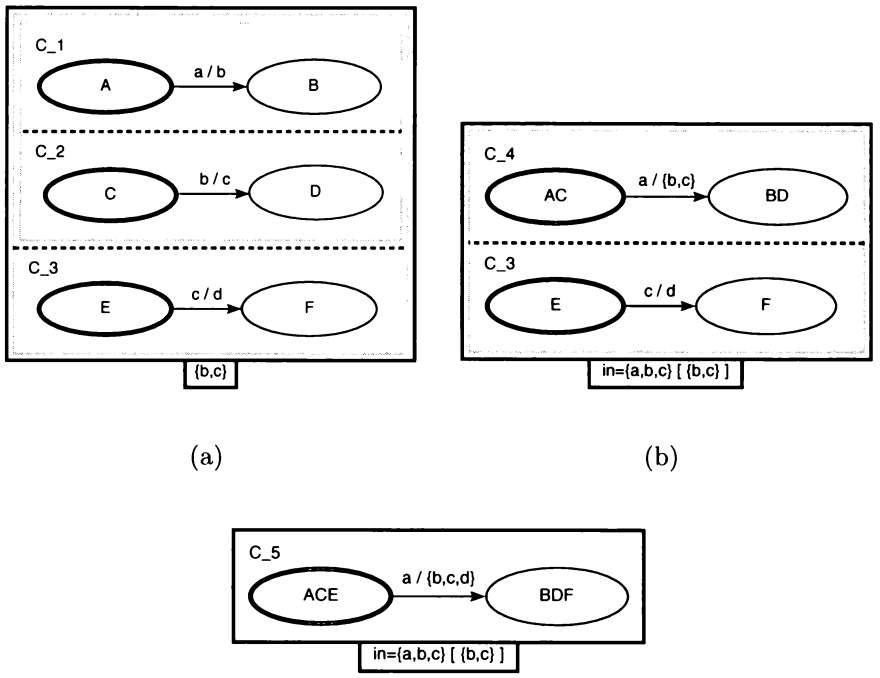


Figure 3.4: The sequentialisation of a composed chart

The chart C_4 of Figure 3.4(b) is the sequential representation of $C_1 \mid \{a, b\} \mid C_2$. Chart C_5 is the sequential representation of $C_4 \mid \{a, b\} \mid C_3$. That the following holds,

$$\langle\!| \, c_{C_1} \Rrightarrow A,\ c_{C_2} \Rrightarrow C,\ c_{C_3} \Rrightarrow E,\ c'_{C_1} \Rrightarrow B,\ c'_{C_2} \Rrightarrow D,\ c_{C_3} \Rrightarrow F,\ i_C \Rrightarrow \{a, b, c\},\ act \Rrightarrow \{C_1, C_2, C_3\},\ o'_C \Rrightarrow \{b, c, d\} \, |\!\rangle \in \delta_C.$$

and hence,

$$\langle\!| \, c_{C_1} \Rrightarrow A,\ c_{C_2} \Rrightarrow C,\ c_{C_3} \Rrightarrow E,\ c'_{C_1} \Rrightarrow B,\ c'_{C_2} \Rrightarrow D,\ c'_{C_3} \Rrightarrow F,\ i_C \Rrightarrow \{a\},\ act \Rrightarrow \{C_1, C_2, C_3\},\ o'_C \Rrightarrow \{b, c, d\} \, |\!\rangle \in \delta_C.$$

is trivial to show, given $C = (C_1 \mid \{b, c\} \mid C_2) \mid \{b, c\} \mid C_3)$ (*i.e.* the chart in Figure 3.4(a)), using the same proof method as that demonstrated for the example chart in Figure 3.3.

Of course, that this process works correctly, that is models our intuition

for charts that contain negated signals, requires that the combination of inconsistent transitions is not represented in the transition model. Take for example the following chart:

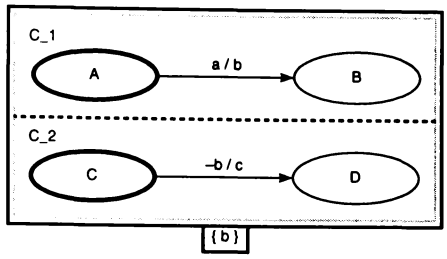


Figure 3.5: Composed chart with inconsistent transitions

The following lemma states, assuming the charts C_1 and C_2 are active, that the transition model for this chart contains no bindings that represent a transition from the configuration A, C. Hence, even though there are explicitly defined transitions in charts C_1 and C_2 respectively, the result of combining these two charts using composition is that the composed chart has no transitions that can be made.

Lemma 3.4.5 Given $C = C_1 \mid \{b\} \mid C_2$, where C_1 and C_2 are the sequential charts of Figure 3.5, for arbitrary z^{T^a} we have,

$$\frac{actv\ C\ z}{z \notin \delta_C}$$

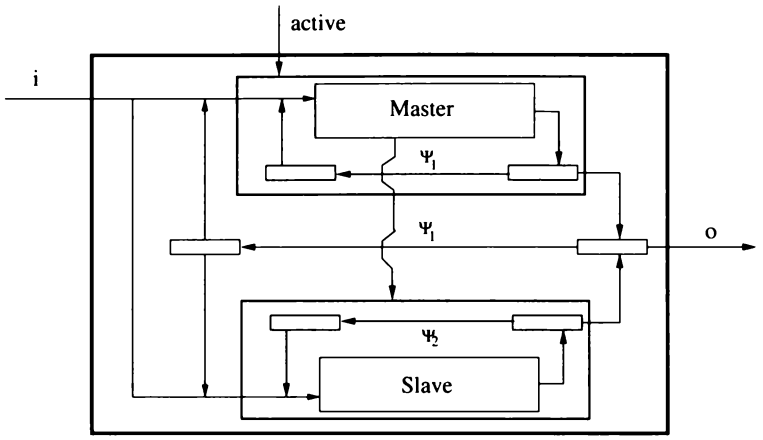
where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$

In this example we prove that there are no bindings in the model that represent active transitions. This is because the example has only two candidate transitions that can combine to become an explicit transition that the composition can make. Because these transitions cannot occur together in the presence of feedback, *i.e.* the output b from C_1 makes the trigger of the transition in C_2 false, means the resulting model contains no bindings representing active transitions.

3.5 The decomposition operator

Another of the useful structuring mechanisms of μ-charts is the decomposition operator. The decomposition of a sequential chart refers to replacing a state in the chart with another μ-chart. This creates a master/slave relationship between a sequential chart (master) and an arbitrary chart (slave) that replaces a state in the master.

Intuitively, we can consider the behaviour of a master and a slave in such a chart as though they are composed in parallel. Consider the following circuit diagram representing the chart:
Dec $(Master, \Sigma_1, \sigma_1, \Psi_1, \delta_1)$ *by* $\{(\sigma_1, (Slave, \Sigma_2, \sigma_2, \Psi_2, \delta_2))\}$, that is, a decomposed chart called *Master* with a state called σ_1, which itself contains a chart called *Slave*



There are two subtle but significant differences between this diagram and the corresponding circuit diagram for the composition operator. The first being that the active state of the slave is determined by the master and not by the active input to the overall decomposition. This is because the state of the master determines whether or not the slave is active. The second difference is that the feedback signals that the master and slave share or communicate on are determined by the feedback set that is present in the definition of the master, *i.e.* denoted by the links labelled Ψ_1 in the diagram.

Hence when the master is in the state decomposed by the slave, that is both charts are active, then the decomposition is exactly the same as the composition of the master and slave with the feedback set equivalent to that of the master chart.

Again, the structure of the circuit diagram assumes both the master and slave are sequential charts; in general the slave can have arbitrary structure. However, unlike composition, the master must be a sequential chart.

The semantics that results from adhering exactly to this model of decomposition produces some "unexpected" behaviour (at least behaviour that may be considered somewhat counter to intuition). For example consider the decomposed chart $C = Dec\ (\omega\ C_1)\ by\ \{(C_2, \omega\ C_2)\}$ pictured in Figure 3.6.

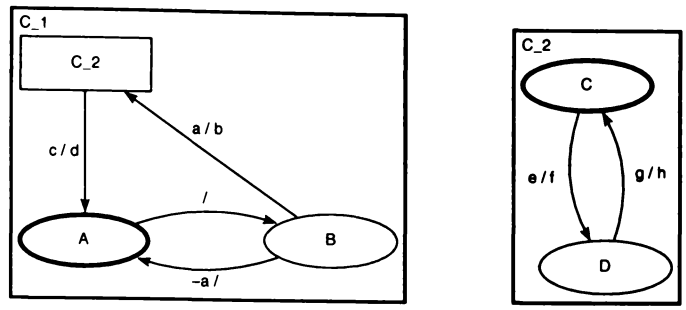
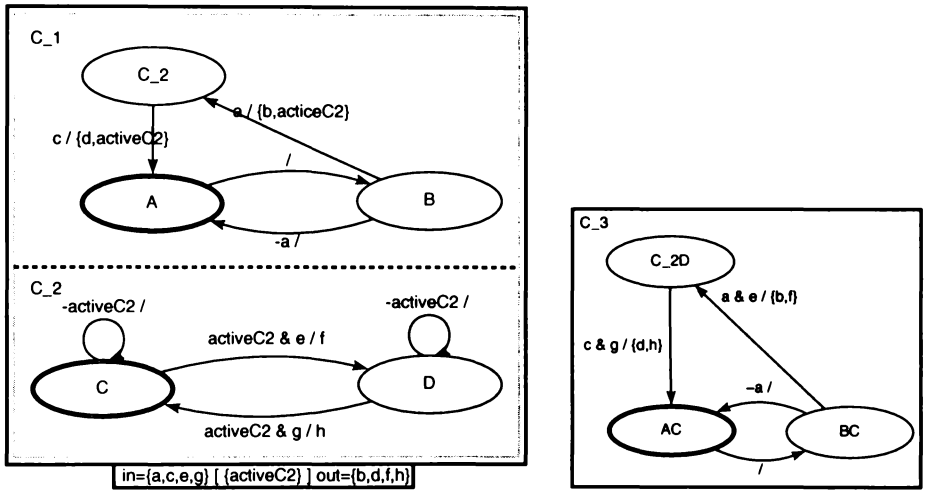


Figure 3.6: Decomposed chart C_1

The semantics of the example decomposed chart is demonstrated by the composed chart $C' =_{\{a,c,e,g\}} [C_1 \mid \{activeC2\} \mid C_2]_{\{b,d,f,h\}}$ pictured in Figure 3.7(a). The chart C_3 (Figure 3.7(b)) is the sequential equivalent of C'. Therefore, the chart C_3 exhibits identical behaviour to the original decomposed chart C.



(a) (b)

Figure 3.7: Sequential equivalent of composition

The behaviour of the chart C_3 does not appear to capture the intention of the decomposed chart C under any standard interpretation. However, consider the following examples of complimentary interpretations that we may wish, in the future, to capture using the decomposition operator. Under some situations we may wish that the slave chart C_2 is initialised each time a transition in the master enters the slave. On the other hand we may wish that the slave remembers what state it was in when it was last exited by the

master. It may be the case that the master should be idle, that is remain in the slave's state and produce no output, while the slave is free to make transitions or we may wish to allow the master to perform some task in parallel with the slave, for example counting the number of transitions the slave makes. Importantly, each of these interpretations can be implemented using explicit transitions in the original decomposed chart, such that, the new behaviour is a refinement of the behaviour that is assigned to chart C. To make explicit in the semantic model one of these interpretations for the chart C would have the side-effect that the alternate interpretation is no longer expressible at all. By giving the most general semantics we allow different interpretations to be assigned in the future without modifying the core language.

The transition model for the decomposed chart C where $C = Dec\ (\omega\ M)\ by\ \{(S, \omega\ S)\}$ and $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and arbitrary S, contains the following Z definitions and schemas:

$$
\begin{array}{|l}
states_C : \mathbb{P}\, \mu_{State} \\
in_C : \mathbb{P}\, \mu Signal \\
out_C : \mathbb{P}\, \mu Signal \\
\hline
states_C = \\
\quad states_M \cup states_S \\
in_C = in_M \cup in_S \\
out_C = out_M \cup out_S
\end{array}
$$

$$
\begin{array}{|l}
Chart_C \\
\hline
Chart_M \\
Chart_S \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
Init_C \\
\hline
Init_M \\
Init_S \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\delta_C \\
\hline
\Delta Chart_C \\
i_C : \mathbb{P}\, in_C \\
act : \mathbb{P}\, \mu_{State} \\
o'_C : \mathbb{P}\, out_C \\
\hline
M \in act \Leftrightarrow C \in act \\
S \in act \Leftrightarrow (C \in act \land (MS \lor MS')) \\
\exists\, i_M, i_S, o'_M, o'_S : \mathbb{P}\, \mu Signal \bullet i_M = (i_C \cup (o'_C \cap \Psi)) \cap in_M \land \\
\qquad i_S = (i_C \cup (o'_C \cap \Psi)) \cap in_S \land o'_C = o'_M \cup o'_S \land \delta_M \land \delta_S \\
\hline
\end{array}
$$

Again we give the definition of the transition model in terms of sets of bindings.

Definition 3.5.1 Given the decomposed chart $C = Dec\ (\omega\ M)\ by\ \{(S, \omega\ S)\}$, where $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and arbitrary S we have,

$$[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C} =_{def} \{z^{T^a} \mid M \in z.act \Leftrightarrow C \in z.act \wedge$$
$$S \in z.act \Leftrightarrow (C \in z.act \wedge (z.c_M = S \vee z.c'_M = S)) \wedge$$
$$\exists\, o_m, o_s \bullet z.o'_C = o_m \cup o_s \wedge$$
$$z \star \langle\!|\, i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_m \,|\!\rangle \mathrel{\dot\in} \delta_M \wedge$$
$$z \star \langle\!|\, i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_s \,|\!\rangle \mathrel{\dot\in} \delta_S\}$$

Now the introduction and elimination rules for decomposed charts are as follows.

Proposition 3.5.1 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, where $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, for the binding z^{T_3} and arbitrary sets o_1 and o_2 we have,[10]

$$\frac{\begin{array}{l} z \mathrel{\dot\in} \delta_C \quad z.o'_C = o_m \cup o_s, \\ \quad\quad z \star \langle\!|\, i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_m \,|\!\rangle \mathrel{\dot\in} \delta_M, \\ \quad\quad z \star \langle\!|\, i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_s \,|\!\rangle \mathrel{\dot\in} \delta_S, \\ \quad\quad actv\ C\ z \vee inactv\ C\ z \qquad\qquad \vdash Q \end{array}}{Q}\ (M_S^-)$$

$$\frac{\begin{array}{l} z.o'_C = o_1 \cup o_2 \\ z \star \langle\!|\, i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_1 \,|\!\rangle \mathrel{\dot\in} \delta_M \\ z \star \langle\!|\, i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_2 \,|\!\rangle \mathrel{\dot\in} \delta_S \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \mathrel{\dot\in} \delta_C}\ (M_S^+)$$

where the usual conditions hold for o_m, o_s and Q, $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

The predicates $actv\ C\ z$ and $inactv\ C\ z$, for decomposed charts, are defined as follows:

$$actv\ C\ z =_{def} actv\ M\ z \wedge (actv\ S\ z \Leftrightarrow (z.c_M = S \vee z.c'_M = S)) \wedge C \in z.act$$
$$inactv\ C\ z =_{def} inactv\ M\ z \wedge inactv\ S\ z \wedge C \notin z.act$$

Appendix B.4 presents proofs for all propositions in this section, including showing that the property act_{LEM} of Lemma B.3.3 and other associated properties hold for decomposed charts.

When the master is not in a decomposed state the slave contributes no output and does not change state, that is, the slave makes no transition. Hence we derive further elimination rules as follows.

[10] As for composed charts, there is again an implicit typing constraint on type T_3 for rule (M_S^-).

Proposition 3.5.2 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary S and $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and z^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{inactv\ S\ z}\ (M^-_{S\,II})$$

$$\frac{z \dot{\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{z \star (\!|\ i_M \Rrightarrow z.i_C \cap in_M, o'_M \Rrightarrow z.o'_C\ |\!) \dot{\in} \delta_M}\ (M^-_{S\,III})$$

$$\frac{z \dot{\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{z \star (\!|\ i_S \Rrightarrow z.i_C \cap in_S, o'_S \Rrightarrow \{\}\ |\!) \dot{\in} \delta_S}\ (M^-_{S\,IV})$$

where $T^a \preceq T_3$.

The rules $(iact^-_I)$, $(iact^-_{II})$ and $(iact^+)$ hold for decomposed charts.

Proposition 3.5.3 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary S, $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, and binding z^{T_3} we have

$$\frac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z \dot{\in} \Xi Chart_C}\ (iact^-_I) \qquad \frac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\ (iact^-_{II})$$

$$\frac{inactv\ C\ z \quad z \dot{\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \dot{\in} \delta_C}\ (iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Also, like sequential and composed charts, simple decompositions do not distinguish between the source of input, that is, the rule $(z^{\in}_t)$ holds for decomposed charts.

Proposition 3.5.4 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary S, $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and bindings z^{T_3} and x^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \dot{\in} \delta_C}\ (Z^{\in}_t)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

Until now we have assumed that all decomposed charts have just one state replaced by another chart. Of course, in general any subset of a chart's states can be decomposed. This more general case of the decomposition operator requires additional definition in the Z model of decomposition.

First, the Z definition of a general chart requires additional definitions and schemas. Suppose that we have the generic decomposed chart $C = Dec\ (\omega\ M)$ *by* $\{(S_1, \omega\ S_1), (S_2, \omega\ S_2), ..., (S_n, \omega\ S_n)\}$ where $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$

and $\{S_1, S_2, ..., S_n\} \subseteq \Sigma$. From the simple transition model above we can generate the transition model for the master chart M paired individually with each of the respective slaves. That is, we get the schemas $\delta_{M_{S_1}}, \delta_{M_{S_2}}, ..., \delta_{M_{S_n}}$ by modelling each of the respective charts $M_{S_1} = Dec\ (\omega\ M)\ by\ \{(S_1, \omega\ S_1)\}, M_{S_2} = Dec\ (\omega\ M)\ by\ \{(S_2, \omega\ S_2)\}, ..., M_{S_n} = Dec\ (\omega\ M)\ by\ \{(S_n, \omega\ S_n)\}$. Now the generic Z definition of the transition model for decomposed chart C is as follows:

$$
\begin{array}{|l}
states_C : \mathbb{P}\, \mu_{State} \\
in_C : \mathbb{P}\, \mu Signal \\
out_C : \mathbb{P}\, \mu Signal \\
\hline
states_C = states_{M_{S_1}} \cup states_{M_{S_2}} \cup ... \cup states_{M_{S_n}} \\
in_C = in_{M_{S_1}} \cup in_{M_{S_2}} \cup ... \cup in_{M_{S_n}} \\
out_C = out_{M_{S_1}} \cup out_{M_{S_2}} \cup ... \cup out_{M_{S_n}}
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{Chart_C} \\
\hline
Chart_{M_{S_1}} \\
Chart_{M_{S_2}} \\
... \\
Chart_{M_{S_n}} \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\multicolumn{1}{l}{Init_C} \\
\hline
Init_{M_{S_1}} \\
Init_{M_{S_2}} \\
... \\
Init_{M_{S_n}} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\delta_C} \\
\hline
\Delta Chart_C \\
i_C : \mathbb{P}\, in_C \\
act : \mathbb{P}\, \mu_{State} \\
o'_C : \mathbb{P}\, out_C \\
\hline
C \in act \Leftrightarrow M \in act \\
\exists\, i_{M_{S_1}}, i_{M_{S_2}}, ..., i_{M_{S_n}}, o'_{M_{S_1}}, o'_{M_{S_2}}, ..., o'_{M_{S_n}} : \mathbb{P}\, \mu Signal \bullet \\
\quad i_{M_{S_1}} = (i_C \cup (o'_C \cap \Psi)) \cap in_{M_{S_1}} \wedge \\
\quad i_{M_{S_2}} = (i_C \cup (o'_C \cap \Psi)) \cap in_{M_{S_2}} \wedge ... \wedge \\
\quad i_{M_{S_n}} = (i_C \cup (o'_C \cap \Psi)) \cap in_{M_{S_n}} \wedge \\
\quad o'_C = o'_{M_{S_1}} \cup o'_{M_{S_2}} \cup ... \cup o'_{M_{S_n}} \wedge \\
\quad \delta_{M_{S_1}} \wedge \delta_{M_{S_2}} \wedge ... \wedge \delta_{M_{S_n}}
\end{array}
$$

The semantic definition for the general case of the decomposition operator is given in terms of a recursive scheme. The relationship between the definition and the generic schema given above is not trivially equivalent. However, unwinding the recursion, using the specific case of one decomposed state as the base case, will always result in a definition that looks like the encoding given by the generic Z schema δ_C above.

Definition 3.5.2 Given the decomposed chart $M_\Pi = (Dec\ (\omega\ M)\ by\ \Pi)$, where $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, $\Pi_1 \subseteq \Sigma$ and $\#\Pi \geq 2$, for arbitrary $S \in \Pi$ and $M_\Phi = (Dec\ (\omega\ M)\ by\ (\Pi \setminus \{(S, \omega S)\})$ we have,[11]

$$
\begin{aligned}
[\![\delta_{M_\Pi}]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C} =_{def} \{ z^{T^a} \mid\ & M_\Phi \in z.act \Leftrightarrow M_\Pi \in z.act \wedge \\
& S \in z.act \Leftrightarrow (M_\Pi \in z.act \wedge (z.c_M = S \vee z.c'_M = S)) \wedge \\
& \exists\, o_{m_\Phi}, o_s \bullet z.o'_{M_\Pi} = o_{m_\Phi} \cup o_s \wedge \\
& \quad z \star \{\!|\ i_{M_\Phi} \Rrightarrow (z.i_{M_\Pi} \cup fb_z) \cap in_{M_\Phi}, o'_{M_\Phi} \Rrightarrow o_{m_\Phi}\ |\!\} \mathrel{\dot\in} \delta_{M_\Phi} \wedge \\
& \quad z \star \{\!|\ i_S \Rrightarrow (z.i_{M_\Pi} \cup fb_z) \cap in_S, o'_S \Rrightarrow o_s\ |\!\} \mathrel{\dot\in} \delta_S \}
\end{aligned}
$$

The chart M_Φ is the result of removing the decomposition of just one state, namely state S, from the decomposed chart M_Π. Thus applying this definition recursively results in the case where just one state is decomposed, that is, the base case described earlier.

Because this recursive definition for charts with multiple decomposed states has the same form as the definition of the base case, that is Definition 3.5.1, all of the rules described in propositions 3.5.1 to 3.5.4, and their respective proofs, generalise to decomposed charts in which multiple states are decomposed.

3.6 Chart context and signal hiding

The final structuring mechanism of μ-Charts is the *interface operator*. As we have already mentioned, the interface operator allows the designer to specify the assumed context for a chart. This operator allows two conceptually different types of behaviour to be specified. The first is signal filtering and signal hiding. The second is choosing not to output particular signals.

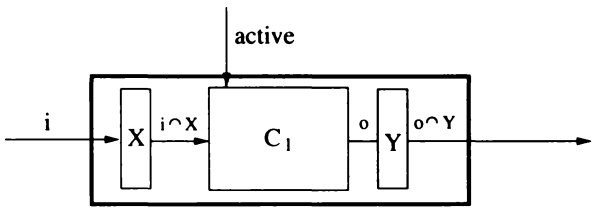
Signal filtering and signal hiding is typically used to internalise the communication between parts of a complex specification. Often it is a requirement that two sub-charts composed together communicate with each other privately. Private communication between the parts can neither be observed or interfered with from outside of the composition. This type of behaviour can be modelled by specifying the internal communication between the sub-charts and then using the interface operator to localise the signals used for the internal communication.

Alternately, a designer may wish to identify that the context can be controlled by some signal but choose not to output that signal. This is modelled

[11] The set Π_1 represents the set containing all of the first elements from the set of pairs Π.

using μ-Charts by specifying an output interface that contains more signals than the natural output interface for a chart. It is not until we consider refinement of μ-charts that the justification for this subtle distinction between the natural and explicit output interfaces of a chart becomes apparent. Given the finite state machine nature of charts, it is tempting to think that the natural output interface of a μ-chart describes all of the signals that the chart can output regardless of what context it is placed in. In fact, the ability to specify an explicit interfaces for a chart allows a designer to specify both the assumed context of a chart plus the input output behaviour of the reactive system in that context. The refinement calculus that we define for charts then allows refinements that both refine the behaviour described as well as change the assumed context of the chart.

The respective rôles of the input and output interfaces for charts is discussed in detail in Section 5.3.3. The following diagram demonstrates the general structure of the model for a chart $C =_X [C_1]_Y$, that is, chart C_1 with explicitly defined input interface X and output interface Y.



The following Z definitions and schemas provide the transition model for the chart $C =_X [C_1]_Y$.

$$
\begin{array}{|l}
states_C : \mathbb{P}\, \mu_{State} \\
in_C : \mathbb{P}\, \mu Signal \\
out_C : \mathbb{P}\, \mu Signal \\
\hline
states_C = states_{C_1} \\
in_C = X \\
out_C = Y
\end{array}
\qquad
\begin{array}{l}
Chart_C == Chart_{C_1} \\
\\
Init_C == Init_{C_1}
\end{array}
$$

$$
\begin{array}{|l}
\delta_C \\
\hline
\Delta Chart_C \\
i_C : \mathbb{P}\, in_C \\
act : \mathbb{P}\, \mu_{State} \\
o_C' : \mathbb{P}\, out_C \\
\hline
C \in act \Leftrightarrow C_1 \in act \\
\exists\, i_{C_1}, o_{C_1}' : \mathbb{P}\, \mu Signal \bullet i_C \cap in_{C_1} = i_{C_1} \wedge o_C' = o_{C_1}' \cap out_C \wedge \delta_{C_1}
\end{array}
$$

Notice that apart from the obvious distinction between the schema δ_C and the existing δ_{C_1}, *i.e.* the new predicate that restricts the output, there are some more subtle differences introduced using the Z type system. In particular the input observation i_C has the type $\mathbb{P}\, in_C$ rather than $\mathbb{P}\, in_{C_1}$. When reasoning about the schema δ_C this observation gets normalised so that its type, as for the input observation i_{C_1} in δ_{C_1}, is $\mathbb{P}\, \mu Signal$. However, the normalisation also adds the additional constraint $i_C \subseteq in_C$ to the predicate part of the normalised version of the schema δ_C. Thus restricting the set of binding from δ_{C_1} to those that take input from the new input interface for the chart C and hence modelling the hiding of input.

Interestingly, the process of hiding input restricts the set of binding that is the model of the original chart, whereas, output hiding requires that we modify the bindings that inhabit the model of the original chart. When we restrict the input interface of a chart we are making the specified reactive system less reactive. That is, there are less signals that the environment can use to effect a reaction from the system in question. On the other hand, hiding output does not affect the reactivity of the specified system, rather it lessens the ability of system to affect its environment.

Once again, we give the definition of the transition model in terms of the meaning of the Z. Given the chart $C =_X [C_1]_Y$ we have,

$$\begin{aligned}[\![\delta_C]\!]^{\mathbb{P}\, T^a}_{\mathcal{Z}_C} =_{def} \{ z^{T^a} \mid\ & C_1 \in z.act \Leftrightarrow C \in z.act \wedge \\ & \exists\, o_1 \bullet z.o'_C = o_1 \cap out_C \wedge \\ & z \star (\!|\, i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 \,|\!) \in \delta_{C_1} \}\end{aligned}$$

Notice that the definition of the type T^a implies that $z.i_C \subseteq in_C$. This demonstrates how important the type system of $\mathcal{Z}_C$ is in the description of the logic for μ-charts presented.

The following introduction and elimination rules follow trivially from this definition.

Proposition 3.6.1 Given $C =_X [C_1]_Y$, for the binding z^{T_3} we have,[12]

$$\frac{z \in \delta_C \qquad \begin{array}{l} z.o'_C = o_1 \cap out_C, \\ z \star (\!|\, i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 \,|\!) \in \delta_{C_1} \vdash P \end{array}}{P}\ (x[]_Y)$$

[12]Again type T_3 is implicitly constrained, such that it is disjoint with respect to each of the types V_1^i, V_1^o, so that the use of the operator $\star$ is well defined.

$$\frac{\begin{array}{l} z.o'_C = o_1 \cap out_C, \\ z \star \langle\!| \, i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}' \Rrightarrow o_1 \, |\!\rangle \mathrel{\dot\in} \delta_{C_1}, \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \mathrel{\dot\in} \delta_C} \ (x[]^+_Y)$$

where the usual conditions hold for o_1 and P, $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

The predicates $actv\ C\ z$ and $inactv\ C\ z$ are defined for charts with hiding as,

$$actv\ C\ z =_{def} actv\ C_1\ z \wedge C \in z.act$$
$$inactv\ C\ z =_{def} inactv\ C_1\ z \wedge C \notin z.act$$

The rules $(iact_I^-)$, $(iact_{II}^-)$ and $(iact^+)$ hold for charts that contain hiding.

Proposition 3.6.2 Given $C =_X [C_1]_Y$, for arbitrary binding z^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z \mathrel{\dot\in} \Xi Chart_C}\ (iact_I^-) \qquad \frac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\ (iact_{II}^-)$$

$$\frac{inactv\ C\ z \quad z \mathrel{\dot\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \mathrel{\dot\in} \delta_C}\ (iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

And finally the rule $(z_t^\in)$ holds for charts with hiding.

Proposition 3.6.3 Given $C =_X [C_1]_Y$ for arbitrary bindings z^{T_3} and x^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \mathrel{\dot\in} \delta_C}\ (z_t^\in)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

3.7 Partial relations semantics

In this section we give the general method for defining the step semantics for a chart.

The step semantics is no more than the transition model of the top-most subchart of a μ-chart with the active state machinery hidden. Given an arbitrary μ-chart called C, the step behaviour of C is defined by another schema which, by convention, we call *CSys*.

CSys

$$\Delta Chart_C$$
$$i_C : \mathbb{P}\, in_C$$
$$o'_C : \mathbb{P}\, out_C$$

$$\exists\, act : \mathbb{P}\, \mu_{State} \bullet C \in act \wedge \delta_C$$

The schema *CSys* hides the active state observation and specifies that the top-most chart of any hierarchical structure is active. Now we have that the Z model for an arbitrary chart C is defined as:

$$[\![C]\!]_Z =_{def} CSys$$

The meaning of the model is given as usual as a set of bindings.

Definition 3.7.1 For arbitrary chart C we have,

$$[\![[\![C]\!]_Z]\!]^{\mathbb{P}\,T}_{Z_C} =_{def} \{z^T \mid \exists z_1^{T^a} \bullet z_1 =_T z \wedge actv\ C\ z_1 \wedge z_1 \in \delta_C\}$$

Note that, for the sake of presentation, we typically refer to $[\![[\![C]\!]_Z]\!]_{Z_C}$ simply as C whenever the context makes clear the intended meaning. From this definition we derive introduction and elimination rules.

Proposition 3.7.1 For arbitrary chart C, and bindings z^{T_3}, we have,

$$\frac{z \dot{\in} C \quad z \star z_a \dot{\in} \delta_C,\ actv\ C\ z \star z_a \vdash P}{P}\ (Z_s^-)$$

$$\frac{z \star x_a \dot{\in} \delta_C \quad actv\ C\ z \star x_a}{z \dot{\in} C}\ (Z_s^+)$$

where $[\![C]\!]^T_{Z_C}$, $[\![\delta_C]\!]^{T^a}_{Z_C}$, $T \preceq T_3$, $T^{act} \not\preceq T_3$, and the usual conditions hold for $z_a^{T^{act}}$ and P.

The schema *CSys* and its meaning describes the step semantics for μ-Charts. We often refer to the step semantics as the *partial relations semantics*. This is because the meaning of the schema *CSys* can be considered

as a relation that maps the before state of a chart and input to its after state and output. This relation is often partial because an abstract μ-Chart specification describes the reaction to some input events and not others.

We now give lifted versions of the introduction and elimination rules for each of the chart operators.[13] Using these lifted rules we can reason about complex charts using just their partial relations semantics, *i.e.* without reverting to the transition model.

For the composition operator we have,

Proposition 3.7.2 Given arbitrary charts C_1, C_2 and $C = C_1 \mid \Psi \mid C_2$, and bindings $z_1^{U_1}$, $z_2^{U_2}$, $x_c^{V^{io}}$, $u_1^{V_1^{io}}$, $u_2^{V_2^{io}}$, $y_1^{U_1}$, $y_2^{U_2}$, $v_c^{V^{io}}$, $w_1^{V_1^{io}}$ and $w_2^{V_2^{io}}$, for arbitrary o_1 and o_2 we have,

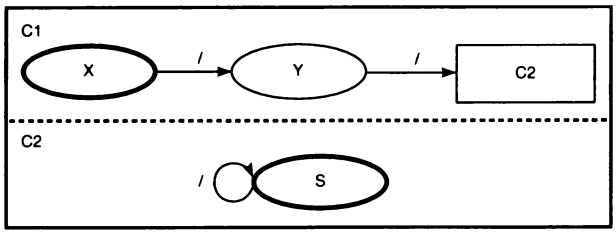
$$\frac{\begin{array}{l} z_1 \star x_1 \star y_1' \star v_1' \mathrel{\dot\in} C_1, \\ z_2 \star x_2 \star y_2' \star v_2' \mathrel{\dot\in} C_2, \\ x_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1}, \\ x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}, \\ v_c.o_C = v_1.o_{C_1} \cup v_2.o_{C_2} \vdash P \end{array}}{z_1 \star z_2 \star x_c \star y_1' \star y_2' \star v_c' \mathrel{\dot\in} C \quad P} \; (Z^-_{|-|})$$

$$\frac{\begin{array}{l} z_1 \star u_1 \star y_1' \star w_1' \mathrel{\dot\in} C_1 \\ z_2 \star u_2 \star y_2' \star w_2' \mathrel{\dot\in} C_2 \\ u_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1} \\ u_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2} \\ v_c.o_C = w_1.o_{C_1} \cup w_2.o_{C_2} \end{array}}{z_1 \star z_2 \star x_c \star y_1' \star y_2' \star v_c' \mathrel{\dot\in} C} \; (Z^+_{|-|}) \; (\dagger)$$

where $fb_{v_c} =_{def} v_c.o_C \cap \Psi$, $[\![C]\!]^{\mathbb{P}\,T}_{z_C}$, $[\![C_1]\!]^{\mathbb{P}\,T_1}_{z_C}$, $[\![C_2]\!]^{\mathbb{P}\,T_2}_{z_C}$, and the usual conditions hold for $x_1^{V_1^{io}}$, $x_2^{V_2^{io}}$, $v_1^{V_1^{io}}$, $v_2^{V_1^{io}}$ and P. This rule contains a side-condition (labelled †) which requires that we can show that $\forall\, z^{T^{io}} \bullet \exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a$.

The form of these introduction and elimination rules is motivated by the refinement rules introduced later. Also, notice that the side-condition of this rule is not given as one of the assumptions of the rule itself. This presentation of the rule is chosen because this side-condition is considered a syntactic restraint that should be placed on μ-Charts. It describes a desirable global property of a specification which ensures that the Z model is sensible rather than a property related to individual transitions. Consider the following example chart which illustrates the category of charts for which this side-condition fails.

[13] See Appendix B.6 for the proofs of the introduction and elimination rules in this section.



This is an example where the chart C_2 appears both as part of the composite chart $C = \omega\ C_1 \mid \{\} \mid \omega\ C_2$ and as the decomposition of state C_2 in chart $\omega\ C_1$. The Z model of this chart contains no bindings that represent the chart starting a transition from its initial states. This is because it is not possible for chart C_2 to be both active (as part of the composition) and inactive (as part of the decomposition) at the same time. The reason that we present the property as a side-condition rather than an assumption of the rule is because this situation is likely to never arise in practice. In terms of "charts as an engineering tool" it would be easier to restrict the language with the stronger condition requiring that one of the examples of chart C_2 be renamed, *i.e.* distinguished from the other. However, in terms of "charts as an eloquent formalism" we may wish to prove properties of the language such as $S \sqsupseteq_{\tau f} S \mid \{\} \mid S$ and $S \mid \{\} \mid S \sqsupseteq_{\tau f} S$.

For the interface operator we have,

Proposition 3.7.3 Given chart $C =_X [C_1]_Y$ for arbitrary C_1 and bindings z^U, $x^{V^{io}}$, y^U, $v^{V^{io}}$ and $x_1^{V_1^{io}}$, we have,

$$\frac{z \star x \star y' \star v' \mathrel{\dot\in} C \quad x_1.i_{C_1} = x.i_C \cap in_{C_1} \quad \begin{array}{l} z \star x_1 \star y' \star u_1' \mathrel{\dot\in} C_1, \\ v.o_C = u_1.o_{C_1} \cap out_C \vdash P \end{array}}{P}\ (Z_{[\!]}^-)$$

$$\frac{z \star u_1 \star y' \star w_1' \mathrel{\dot\in} C_1 \quad u_1.i_{C_1} = x.i_C \cap in_{C_1} \quad v.o_C = w_1.o_{C_1} \cap out_C}{z \star x \star y' \star v' \mathrel{\dot\in} C}\ (Z_{[\!]}^+)\ (\dagger)$$

where $[\![C]\!]^{\mathbb{P}\,T}_{z_C}$ and $[\![C_1]\!]^{\mathbb{P}\,T_1}_{z_C}$ the usual conditions hold for $u_1^{V_1^{io}}$, and P. The side-condition † requires that we can show that $\forall z^{T^{io}} \bullet \exists z_a^{T^{act}} \bullet actv\ C\ z \star z_a$.

We conclude this chapter by examining the relationship between composition and decomposition. As expected, given the definition of decomposed charts, we can show that two charts that share a master/slave relationship react in the same way as if they are composed in parallel whenever the master is in the decomposed state and vice versa. This gives us another useful introduction and elimination rule.

Proposition 3.7.4 Given $M_S = (Dec\ (\omega\ M)\ by\ \{(S, \omega\ S)\})$, for arbitrary S, $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, and $MS_\Psi = [\![\omega\ M \mid \Psi \mid \omega\ S]\!]_{Z_\iota}$, then for arbitrary z^{T_3} we have,

$$\frac{z.c_M = S \vee z.c'_M = S \quad z \dot{\in} \delta_{M_S}}{z \dot{\in} \delta_{MS_\Psi}}\ (M^-_{S\,V})$$

$$\frac{z.c_M = S \vee z.c'_M = S \quad z \dot{\in} \delta_{MS_\Psi}}{z \dot{\in} \delta_{M_S}}\ (M^+_{S\,II})$$

where $[\![\delta_{M_S}]\!]^{\mathbb{P}\,T^a}_{Z_C}$ and $T^a \preceq T_3$.

The Z model presented in this chapter gives a full account of the assumed semantics of μ-Charts. The introduction and elimination rules for each of the language constructs give a logic that can be used to reason about μ-Charts.

In Chapter 4, we investigate further the semantics of charts using the more common-to-reactive-systems view of a trace semantics.

Chapter 4

Trace Semantics

This chapter introduces a trace semantics for μ-Charts. First we consider what we mean by "trace semantics". A trace semantics for μ-Charts gives a meaning to charts in terms of their observable input/output behaviour. Hence it is an abstraction on a state-based view, in that the state information explicit in a chart diagram is not present at all in the trace semantics. The trace semantics considers only the interactions of the chart with its environment and therefore is a record of the sequence of outputs that a chart produces given a sequence of inputs.

We use infinite traces to model the behaviour of charts. The reason for using infinite traces is solely because this work closely follows that of Scholz [78] where infinite traces are used. An alternative approach, introduced by Hoare in [42], is to use finite traces without saying how long they are. This entails constructing subsequence complete (or "prefix complete") sets of finite traces. For example, the well known CSP process $CLOCK = (tick \rightarrow CLOCK)$ has a finite trace behaviour described by the set $\{\langle\rangle, \langle tick\rangle, \langle tick, tick\rangle, ...\}$. We could, and in some respects do, use this method to encode each of the infinite traces that together give a chart its trace meaning. However, there is no simple congruence between the "trace semantics" that one would associate with a process algebra such as CSP and the trace semantics that we give here for charts.

Roscoe ([75], page 172) points out that the only situation under which an infinite trace model conveys more information than the alternate finite trace model for CSP is when a process can exhibit all of the finite prefixes of an infinite trace but not the infinite trace itself. By the same reasoning, we could use prefix closed sets of finite traces in place of infinite sequences. Charts are finite in state and transition and therefore cannot rule out an infinite trace

for which all of the finite prefixes are possible. In fact, in Section 4.6 we use the limit of the prefix closed sets of finite traces to define infinite traces.

For an example of a trace semantics for μ-Charts that closely follows Hoare's approach see [28].

In fact, the trace semantics presented plays a reasonably minor rôle in this work—our ultimate goal is to investigate the possibility of giving a partial relations-based refinement framework for μ-Charts.

However, so that we can attempt to relate the investigation presented here back to previous work done on charts [78], we describe, not one, but several, trace semantics. The difference between them is not due to differences in the objects that we use to formalise the meaning but rather differences in the meaning assigned to charts. Specifically, we consider the implicit behaviour of a chart when explicit behaviour is not defined for some input sequence.

Also, we introduce the different trace semantics so that we can use the intuitive operational nature of traces to informally discuss the different notions of refinement. This gives us another view from which we can consider the defined notions of refinement. That the traces view of these types of formalisms provide an intuitive way to talk about reactive systems is best described by Hoare's description [42] of what a trace semantics represents: consider an observer with a note book recording the behaviour of the system as it is running.

In the following we show that the choice of how we totalise the partial relations that describe charts can be considered synonymous with choosing a different trace semantic interpretation for charts. In particular, if we define refinement via the partial relations semantics, the necessary choice of totalisation is related to the choice of trace interpretation. We introduce four different trace interpretations to help motivate where each of the commonly-known partial relation completion models, and hence refinement notions, fits into our treatment of μ-Charts.

The ability to use the same partial relations to describe different trace interpretations via refinement could be considered one of the significant benefits of this type of partial definition of behaviour. The idea being that, given the partial relations semantics, the user simply picks the set of tools (*i.e.*, the appropriate refinement rules) to suit their required trace interpretation. The denotational semantics of charts is actually the partial relations semantics plus a set of refinement rules. Each of the different sets of rules assign a different denotation to charts.

We begin to outline the alternatives by considering the example chart

pictured in Figure 4.1.

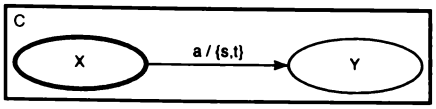


Figure 4.1: μ-chart description of a simple reactive system

The question is: what output should this chart give when reacting to the input $\langle\{a\},\{a\},\{a\},\ldots\rangle$? To answer this we outline four different choices. For each choice we give examples of: the traces that will inhabit the semantics of the chart; a chart that makes explicit the transitions encoded by the given trace semantics; and the necessary completion of the partial relation semantics so that the partial relations based refinement and traces notion of refinement match. To accord with the formalisation of the semantics that follows, the trace semantics of a chart can be considered a total relation between input traces and output traces.

Finally, Section 4.6 gives formal definitions for each of the discussed trace semantic interpretations.

4.1 Do-nothing semantics

The first choice we consider is the *do-nothing semantics* that we denote as $[\![C]\!]^{\omega}_{dn}$ for the chart C. The do-nothing semantics states that if no transition is defined for an input then the chart outputs nothing, *i.e.* outputs the empty set, and remains in its current state. Hence the resulting output trace for the input $\langle\{a\},\{a\},\{a\},\ldots\rangle$ is $\langle\{s,t\},\{\},\{\},\ldots\rangle$. Alternatively, we could write,[1]

$$(\langle\{a\},\{a\},\{a\},\ldots\rangle,\ \langle\{s,t\},\{\},\{\},\ldots\rangle) \in [\![C]\!]^{\omega}_{dn}$$

Recalling that the trace semantics must contain at least one trace for all input sequences, we can give a more general scheme that illustrates the do-nothing trace semantics of chart C. Assuming that the sequence v is an

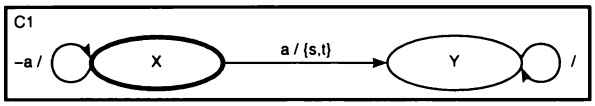
[1]Note that the ... notation used to demonstrate trace behaviour means the trace continues indefinitely as it was most recently, rather than repeating all of the previous sets of signals.

infinite sequence of inputs i such that each $i \subseteq \{a\}$ then we have,

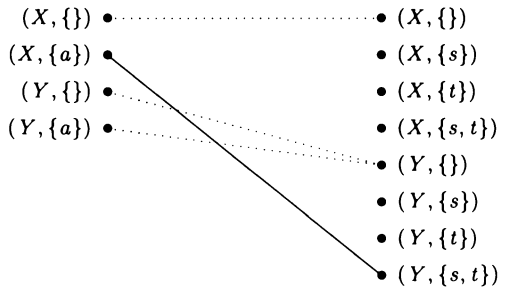
$$
\begin{aligned}
[\![C]\!]^{\omega}_{dn} = \{&(\langle\{\},\{\},...\rangle,\ \langle\{\},\{\},...\rangle),\\
&(\langle\{a\}\rangle^\frown v,\ \langle\{s,t\},\{\},\{\},...\rangle),\\
&(\langle\{\},\{a\}\rangle^\frown v,\ \langle\{\},\{s,t\},\{\},\{\},...\rangle),\\
&(\langle\{\},\{\},\{a\}\rangle^\frown v,\ \langle\{\},\{\},\{s,t\},\{\},\{\},...\rangle),\\
&...\\
&\}
\end{aligned}
$$

Note that the first pair of sequences in this set demonstrates that we cannot assume that the environment will ever produce signal a as input.

The explicit encoding of this semantics is demonstrated by the chart C_1 as follows where the two additional loop-transitions explicitly encode the do-nothing interpretation for the original chart C in Figure 4.1.



The following diagram represents (using solid black lines) the partial relation interpretation of the chart C and (using dotted lines) the necessary completion of the partial relation to give the required semantics. Note that the domain of the relation is a tuple consisting of the current state of the chart and the input, the range is the tuple containing the resulting state and the output.



Again, the reason we give the relational view of the different semantic encodings is that it begins to make obvious the link between the choice of trace semantics and the partial relation completions, some of which, are related to well-known state-based refinement techniques.

4.2 Partial-chaotic semantics

Another possible interpretation, that we will call the *partial-chaotic semantics*, $[\![C]\!]^{\omega}_{p\text{-}chaos}$, is the first of the so-called chaotic interpretations. Under the

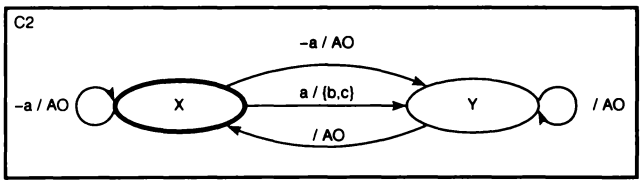
do-nothing semantics, when there is no defined transition for an input, the only acceptable reaction of an implementation is to do nothing. In other words, ignore unexpected input until an appropriate input is present. Hence leaving behaviour undefined in the specification is saying something about the suitable implementations. Under a chaotic interpretation any reaction that an implementation chooses, when a specific reaction is not stated, is acceptable. That is, leaving behaviour undefined should be taken to mean the specifier does not care what happens in this situation.

The partiality of this chaotic semantics is due to the fact that during the step in which chaotic behaviour is permitted, the implementation is free to choose the reaction, including the state to which the chart moves. However, at the next step the implementation is again constrained to react as the specified chart.

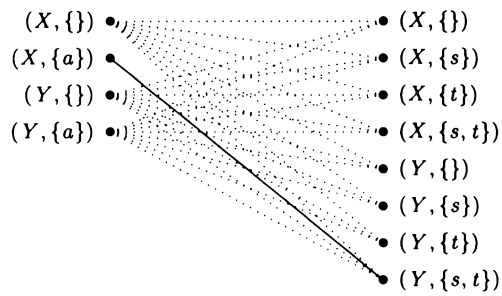
Some of the possible output traces for the example chart C resulting from the input $\langle\{a\},\{a\},\{a\},...\rangle$ include,

$\langle\{s,t\},\{\},\{\},...\rangle$
$\langle\{s,t\},\{s\},\{\},...\rangle$
$\langle\{s,t\},\{t\},\{\},...\rangle$
$\langle\{s,t\},\{s,t\},\{\},...\rangle$
$\langle\{s,t\},\{\},\{s,t\},...\rangle$
...

The explicit encoding of this semantics is demonstrated by the chart C_2 as follows. Note that the output label AO is used as a shorthand to represent an arbitrary output such that $AO \subseteq \{s,t\}$, *i.e.* some subset of out_{C_2}; the output interface of the chart. Each transition that has output AO actually represents four separate transitions.



The appropriate completion of the partial relation semantics is as follows.



4.3 Total chaotic semantics

Another form of chaotic semantics for charts is the *total chaotic semantics* denoted by $[\![C]\!]^{\omega}_{\tau\text{-}chaos}$. This semantic interpretation encodes behaviour that is not only chaotic at one step where no behaviour is described but is chaotic for that step and any steps following.

The actual difference between this and the partial-chaotic semantics is very subtle but significant none the less. Sections 5.2 and 6.3 give a detailed account of this difference and how it arises. From the outset the given encoding of the total chaotic semantics makes the difference between it and the partial-chaotic semantics obvious. It may appear that the difference is itself caused by the choice of model for the respective trace semantics rather than more fundamental differences between the two models. In fact, the opposite is true. The original encoding was modified to properly capture the difference. First we explain what the *total chaotic semantics* is and how it is modelled, then we explain the trace model for the total chaotic semantics.

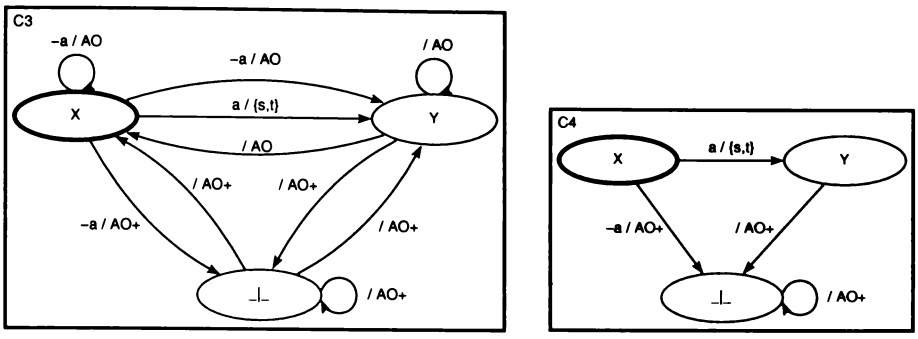
Consider again chart C of Figure 4.1 with input $\langle\{a\},\{a\},\{a\},...\rangle$. The output traces that are part of the total chaotic model for this chart include all of those from the partial-chaotic model and some new output traces such as the following.

$\langle\{s,t\},\{\bot\},\{\bot\},...\rangle$
$\langle\{s,t\},\{s,\bot\},\{\bot\},...\rangle$
$\langle\{s,t\},\{t,\bot\},\{\bot\},...\rangle$
$\langle\{s,t\},\{s,t,\bot\},\{\bot\},...\rangle$
$\langle\{s,t\},\{\bot\},\{s,t,\bot\},...\rangle$
...

The signal $\bot$ denotes a special output value that is used implicitly in the semantic model to indicate that the chart has begun acting chaotically. It cannot be explicitly used as an output signal in the chart itself.

We give two charts that explicitly encode this semantic interpretation in Figure 4.2, both of which exhibit the appropriate traces. Note we again use the label AO as a shorthand for arbitrary output. We also introduce $AO_{\bot}$ which is shorthand for arbitrary output from the set $out_C \cup \{\bot\}$, that is $AO_{\bot} \subseteq \{s,t,\bot\}$. Regardless of which completion we choose the explicit encoding of a total chaotic semantics requires an additional state. This state is necessary in general to model the behaviour where the chart continues to act chaotically from the first undefined step on. The encoding demonstrated in Figure 4.2(a) realises both the partial chaos, as described above, as well as

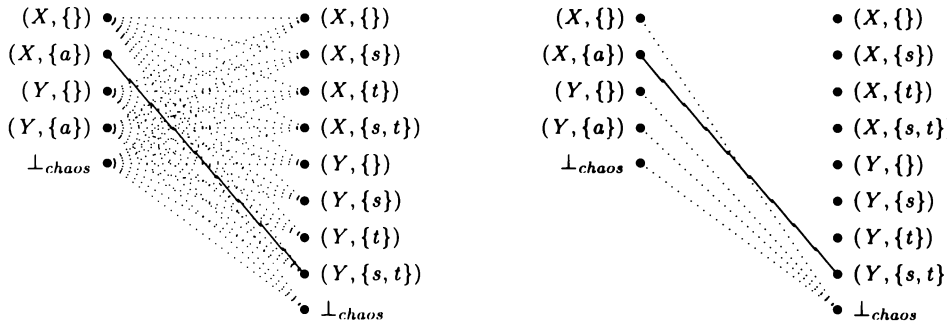
the ability to continue reacting chaotically indefinitely. The second encoding, demonstrated in Figure 4.2(b), shows that, when we ignore state (*i.e.* consider only traces), the observable behaviour of a chart reacting chaotically is a superset of the behaviour that is to recover and resume reacting in a defined manner.



(a) (b)

Figure 4.2: Alternative explicit encodings of the total chaotic semantic model

Not surprisingly, the completion of the partial relation semantics also has two possibilities.



That these two alternatives for completing the partial relation semantics coincide may initially seem surprising. Given the existing literature that describes the completion of partial relation semantics for abstract data types (ADTs) described in Z, one might expect these different completions to give significantly different semantics. However, the apparent difference can be explained by the interpretation that we assign to the special value $\perp_{chaos}$. We can see from the two charts above that the state $\perp$ *simulates* all of the other states in the chart. That is, if we ignore state information, any behaviour that could be observed in any of the defined states can also be observed from the state $\perp$. Hence, the fact that these two completion models

coincide is not unexpected when we consider chart refinement in terms of simulation relations, as we will see later.

Notice that we have made an assumption in the relational diagrams. The value $\perp_{chaos}$ not only stands for a state but also represents arbitrary input and arbitrary output. Moreover, this arbitrary output can contain the special output $\perp$. From a trace semantic point of view it is difficult to give a satisfactory argument that justifies the inclusion of this special output $\perp$.

The necessity of $\perp$ stems from allowing chaotic behaviour to be implicit in the chart description of a reactive system. Implicit chaos is an important abstraction mechanism of μ-Charts. The user can choose to leave parts of the design undefined (or rather "to be defined later") in a chart specification. Also recall that the chart semantics defined in Chapter 3 is designed such that any μ-chart which is made up by the composition of other charts can be equally described by a non-composed chart. Due to implicit chaos, a property of this semantics is that chaotic behaviour in one part of a composition affects the other part of the composition. That is, if one part of the composition acts chaotically then the composition itself also acts chaotically. Now consider the following two charts.

Given the "intrusive" nature of chaos, for arbitrary chart C and set of signals Ψ, the following properties hold.

$$(C \mid \Psi \mid Chaos) = Chaos$$
$$(C \mid \Psi \mid True) = C$$

From this observation it is clear that $[\![Chaos]\!]^{\omega}_{\tau\text{-}chaos} \neq [\![True]\!]^{\omega}_{\tau\text{-}chaos}$, at least its clear that the language would have undesirable compositionality properties if $[\![Chaos]\!]^{\omega}_{\tau\text{-}chaos}$ and $[\![True]\!]^{\omega}_{\tau\text{-}chaos}$ were equated. Hence the output value $\perp$ is used to distinguish *Chaos* from *True*.

4.4 Firing conditions semantics

If we want to describe the total chaotic semantics for charts, as we do here, then we need the stated interpretation of $\perp_{chaos}$. That is, the state $\perp$ simulates being in any of the defined states as we describe above. Alternatively, there is another possible semantic model. When the specifier does not define

a specific behaviour for an input in a given state, the interpretation is neither "do-nothing" (*i.e.* ignore the input until another significant event occurs) nor is it that the specifier does not care what happens (*i.e.* chaotic behaviour). Rather the interpretation is that this behaviour is disallowed. While this interpretation may not seem natural for specifying reactive systems—where the underlying assumption is that the behaviour of the environment is outside of the control of the specified system—it is a natural interpretation for abstract data types where allowing the environment to apply an operation may be deemed unsuitable at the time of specification. This interpretation of ADT specifications is often termed the *firing condition* semantics or the *guarded interpretation* of specified operations. That is, the precondition of an operation determines when it can be applied.

Despite the fact that this semantic interpretation may not seem natural for the specification of reactive systems, we describe it here as a possible semantic model. This is partly for completeness and partly because it may in the future prove to be another useful way of using chart specifications. We call it the *firing condition semantics* for chart C, which is denoted $[\![C]\!]^{\omega}_{fc}$.

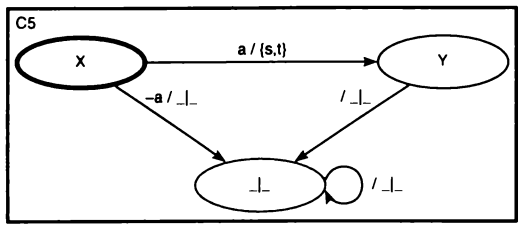
This model may be considered as a way of modelling termination of a reactive system. That is, after an event occurs that was not expected, *i.e.* not defined in the specification, the reactive system stops reacting. To encode this semantics using the same infinite trace framework that we have for the other semantics requires that we decide what it means for a reactive system to stop reacting. When considering a chart itself, changing the previous interpretation of the state $\perp$ to represent a trap state from which, the chart cannot leave, nor output any further signals, appears to be a reasonable way to encode termination. If one was able to observe the state of a reactive system as it was running it would be obvious when the system had terminated. The trace semantics of a chart, however, describes the behaviour of a reactive system without any reference to state. Encoding termination as a continuous trace of empty output does not capture enough information to properly encode termination either. In particular, a chart that is intentionally outputting no signals cannot be distinguished from a chart that has terminated. Therefore, to encode termination of charts in the trace semantics we again use the special output value $\perp$. The output $\perp$ is used in this case to indicate that a chart has terminated.

Now the allowable behaviour of our original example chart C of Figure 4.1 can be defined in a similar (but not equal) fashion to the do-nothing trace semantics. Assuming again that the sequence v is an infinite sequence of

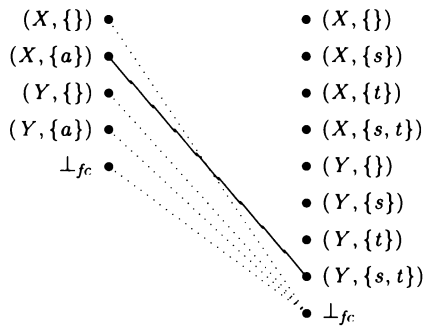
inputs i such that each $i \subseteq \{a\}$ then we have,

$$[\![C]\!]^{\omega}_{fc} = \{(\langle\{\}\rangle^\frown v,\ \langle\{\bot\},\{\bot\},...\rangle),\\ (\langle\{a\}\rangle^\frown v,\ \langle\{s,t\},\{\bot\},\{\bot\},...\rangle)\}$$

The explicit encoding of this semantics is demonstrated by the chart C_5.



The completion of the partial relation semantics looks the same as one of the alternatives for the total chaotic semantics (Figure 4.2(b)). However, we reiterate that the value $\bot_{fc}$ used here is to be read as a special value that indicates termination, that is, the output resulting from a transition to state $\bot$ is the special signal $\bot$.



Notice that the completion is strict. That is, once an undefined event has occurred, the chart will never again output any useful control signals. A chart that makes a transition to state $\bot$ then it must remain in state $\bot$. Unlike the total chaotic completion for charts, here the state $\bot$ does not *simulate* all other states in the chart. It represents a special state that, when entered, stops the chart from partaking in any further interesting interaction with its environment.

4.5 Discussion

From the different possible semantics that we have described in this section we can see that two quite separate views of what a specification actually is have emerged. This is not only true for charts but for formal specification in general. One view is that a specification is partial. That is, it describes just

the parts of the system that the specifier is interested in. An assumption that appears to underlie this approach is that some sort of formal step-wise design will follow specification. This design process ends either in a specification that is deterministic and therefore easily implemented or, in the case of some languages, with the implementation itself. In each step more and more of the previously unspecified behaviour is decided. It appears to fit with this view that refinement will be used as a tool to guide design. Also, this view appears to coincide with the assumption that the environment in which the specified system will reside is outside of the control of the system, that is the environment is considered to be completely nondeterministic in its behaviour. The total chaotic semantics described above is an example of this paradigm.

The alternative view generally treats a specification as a total description of the behaviour of the resulting system. That is not to say that nondeterministic specifications are necessarily disallowed, but any nondeterminism is explicit. This view seems to coincide with the assumption that the system under consideration can inflict control upon its environment, *i.e.* rather than being completely passive it can choose to which of the environment's requests it will react. Here it is typical for design to be a process of constructing alternate, possibly more possibly less implementable, specifications and using refinement to check that one is valid with respect to the other. The do-nothing and firing conditions semantic interpretations given here fit well into this form of formal system design.

4.6 Defining the trace interpretations

Finally, we give formal definitions of each of the trace semantic interpretations that we have outlined above. We have already identified that our semantics will map all infinite input sequences to infinite output sequences (assuming the appropriate interfaces for the chart).

In order to use and prove results about infinite sequences we define them in a manner similar to that suggested by de Bakker and de Vink in [15], namely using the theory of *ultrametric spaces*. As is usual we assume that the set A^* denotes the set of finite sequences over the alphabet A while A^ω denotes the set of infinite sequences over the alphabet A. Of course, in our case the alphabet A will represent all the subsets of some finite set of signals. The set of all sequences, *i.e.* of finite and infinite length, is then denoted by A^∞. Like [15], we assume all sequences in A^∞ are functions that map contiguous natural numbers (excluding 0) to elements of A. That is we have,

Definition 4.6.1 Given a set of signals S,

$$\begin{aligned} A^{\omega} &=_{def} \{\mathbb{N}_1 \rightarrow A \mid \mathbb{N}_1 = \mathbb{N} \setminus \{0\} \wedge A = \mathbb{P}\, S\} \\ A^{*} &=_{def} \{(1\,..\,n) \rightarrow A \mid n \in \mathbb{N} \wedge A = \mathbb{P}\, S\} \\ A^{\infty} &=_{def} A^{*} \cup A^{\omega} \end{aligned}$$

We continue to use the literal $\langle\rangle$ for the empty sequence, $\langle e_1, e_2, ..., e_n\rangle$ for a sequence containing n elements, and $\langle e_1, e_2, ...\rangle$ for an infinite sequence. The *truncation* of an infinite sequence $s \in A^{\omega}$, denoted $s \upharpoonright n$, gives a finite sequence of length n, *i.e.* $s \upharpoonright n \in A^{*}$. The truncation operator is defined as follows:

Definition 4.6.2 Given a set of signals S, $e \subseteq S$ and the sequences $w \in A^{\infty}$ and $\langle e\rangle^{\frown} w \in A^{\infty}$, we have,

$$\begin{aligned} w \upharpoonright 0 &=_{def} \langle\rangle \\ \langle\rangle \upharpoonright n &=_{def} \langle\rangle \\ (\langle e\rangle^{\frown} w) \upharpoonright (n+1) &=_{def} \langle e\rangle^{\frown} (w \upharpoonright n) \end{aligned}$$

assuming the concatenation operation $^{\frown}$ is defined as usual and $n \in \mathbb{N}$.

We define the required *ultrametric* d using the so-called Baire-distance [15].

Definition 4.6.3 Given two sequences $v, w \in A^{\infty}$,

$$d(v,w) =_{def} \begin{cases} 0 & \textit{if } v = w \\ 2^{-n} & \textit{where } n = max\{k \mid v \upharpoonright k = w \upharpoonright k\} \end{cases}$$

Now we have the important result, as shown in [15], that the ultrametric space (A^{∞}, d) is complete. From this we can guarantee that any prefix complete set of finite sequences has a *limit* or a *least upper bound* in A^{∞}. In fact, the sequence that is the least upper bound of such a set will be infinite, *i.e.* a member of A^{ω}. Hence, we describe infinite sequences as the least upper bounds of sets of finite approximations. For example, consider the finite sequence x_n of length n defined as,

$$\begin{aligned} x_0 &= \langle\rangle \\ x_{n+1} &= x_n{}^{\frown}\langle e_n\rangle \end{aligned}$$

where e_n represents some set of signals. We can now describe the infinite sequence $\langle e_0, e_1, e_2, ..., e_n, ...\rangle = \bigsqcup\{x_n \mid n \in \mathbb{N}\}$, that is, the limit of the set of finite sequences that converge to the infinite sequence. Furthermore, we can give this type of description for all infinite sequences $w \in A^{\omega}$ as $\bigsqcup\{w \upharpoonright n \mid n \in \mathbb{N}\}$.

Henceforth, we freely use the notation $\{x_n \mid n \in \mathbb{N}\}_W$ to represent such converging chains of finite sequences. Note that the subscripted W represents a set of signals, each element in the respective sequences (denoted x_n above) is some subset of the signals W. Now we have that the set of infinite sequences $A^\omega = \bigsqcup\{X \mid X \subseteq A^* \wedge X = \{x_n \mid n \in \mathbb{N}\}_{\bigcup A}\}$. This assumes that each element in the alphabet A is a set of signals and that, in this case, $\bigsqcup\{e_1, e_2, ...\} = \{\bigsqcup e_1, \bigsqcup e_2, ...\}$.

Recall that the input interface of chart C is the set of signals denoted by in_C and similarly the output interface as out_C. For convenience, assuming an alphabet $I = \mathbb{P}(in_C)$, we define the additional sets: $\mathcal{I}_C^\omega$—the set of infinite input traces over I; $\mathcal{I}_C^n$—the set of finite sequences of length n over I; $\mathcal{I}_C^*$—the set of finite sequences over I and $\mathcal{I}_C^\infty = \mathcal{I}_C^\omega \cup \mathcal{I}_C^n$. The sets $\mathcal{O}_C^\omega$, $\mathcal{O}_C^n$, $\mathcal{O}_C^*$ and $\mathcal{O}_C^\infty$ are similarly defined over the output interface. For convenience we introduce the set $out_C^\perp =_{def} out_C \cup \{\perp\}$. Also, $\mathcal{O}_{C_\perp}^\omega$, $\mathcal{O}_{C_\perp}^n$, $\mathcal{O}_{C_\perp}^*$ and $\mathcal{O}_{C_\perp}^\infty$ defined over the alphabet $out_C^\perp$.

Notice, while the sets in_C and out_C can be empty, the respective sets $\mathcal{I}_C^\omega$, $\mathcal{O}_C^\omega$ and $\mathcal{O}_{C_\perp}^\omega$ cannot. We use another result from [15], given $(\mathcal{I}_C^\infty, d_1)$ and $(\mathcal{O}_C^\infty, d_2)$ are complete ultrametric spaces then so is $(\mathcal{I}_C^\infty \times \mathcal{O}_C^\infty, d_p)$, where $d_p((x, y), (x', y')) =_{def} max\{d_1(x, x'), d_2(y, y')\}$.

We extend the definition of the truncation operator over infinite relations.

Definition 4.6.4 Given a relation $R \subseteq \mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$ and $n \in \mathbb{N}$ then $R \restriction n \subseteq \mathcal{I}_C^* \times \mathcal{O}_{C_\perp}^*$ such that,

$$R \restriction n =_{def} \{(i \restriction n, o \restriction n) \mid (i, o) \in R\}$$

Using all of the introduced machinery we give the general form of the trace semantics for charts $[\![C]\!]_x^\omega$ as follows:

$$[\![C]\!]_x^\omega \subseteq \mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega \wedge \mathrm{dom}\,[\![C]\!]_x^\omega = \mathcal{I}_C^\omega$$

To give the definitions of the four alternate trace semantics in terms of relations between infinite sequences we use the same method as above. That is, we describe how to build the semantic relation for a chart over equi-length finite sequences and use the least upper bound of a chain of such relations as the meaning for infinite sequences. Intuitively, this can be considered as describing the input/output behaviour of a chart after one step, and then after two steps and then after three steps, *etc.* Clearly, the infinite behaviour can be considered as the limit to which the relation tends as we take more and more steps. That such a limit exists is guaranteed by the metric space results. Essentially, since the sequences (representing a chart's behaviour)

do not change once fixed "in the past" they get "closer and closer" towards a limit. This "closer and closer" is what the Baire-distance of the metric space measures. That the distances tends towards zero demonstrates that the chain is tending towards a limit.

As with the linear case all appropriate chains over finite relations $\{A \subseteq \mathcal{I}_C^n \times \mathcal{O}_{C_\perp}^n \mid n \in \mathbb{N}\}$ have a least upper bound which is a member of $\mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$.

Now, given the finite semantics $[\![C]\!]_x^n \subseteq \mathcal{I}_C^n \times \mathcal{O}_{C_\perp}^n \times Chart_C$ (which will be defined shortly)[2] we have,

Definition 4.6.5 Assuming $s_n = \{(i, o) \mid \exists\, c \bullet (i, o, c) \in [\![C]\!]_x^n\}$,

$$[\![C]\!]_x^\omega \;=_{def}\; \bigsqcup \{s_n \mid n \in \mathbb{N}\}$$

Finally we define each of the alternate trace semantic relations in the finite case. Note we use the precondition operator as defined in Appendix A.3.

4.6.1 Do-nothing semantics

Definition 4.6.6 For arbitrary $si \in \mathcal{I}_C^n$, $so \in \mathcal{O}_C^n$, $i \subseteq in_C$, $o \subseteq out_C$ and $c \in Chart_C$,

$$\begin{array}{ll} [\![C]\!]_{dn}^0 & =_{def}\; \{(\langle\rangle, \langle\rangle, c') \mid c \in Init_C\} \\ [\![C]\!]_{dn}^{n+1} & =_{def}\; \{(si^\frown i, so^\frown o, c') \mid \exists\, ci \bullet (si, so, ci') \in [\![C]\!]_{dn}^n \wedge \\ & \qquad (Pre\ C\ z_i \wedge z_i \star z_o' \in C \vee \neg\, Pre\ C\ z_i \wedge o = \{\})\} \end{array}$$

where $z_i = ci \star (\!|\, i_C \Rrightarrow i \,|\!)$ and $z_o = (\!|\, o_C \Rrightarrow o \,|\!) \star c$

4.6.2 Partial chaotic semantics

Definition 4.6.7 For arbitrary $si \in \mathcal{I}_C^n$, $so \in \mathcal{O}_C^n$, $i \subseteq in_C$, $o \subseteq out_C$ and $c \in Chart_C$,

$$\begin{array}{ll} [\![C]\!]_{\rho\text{-}chaos}^0 & =_{def}\; \{(\langle\rangle, \langle\rangle, c') \mid c \in Init_C\} \\ [\![C]\!]_{\rho\text{-}chaos}^{n+1} & =_{def}\; \{(si^\frown i, so^\frown o, c) \mid \exists\, ci \bullet (si, so, ci') \in [\![C]\!]_{\rho\text{-}chaos}^n \wedge \\ & \qquad\qquad Pre\ C\ z_i \Rightarrow z_i \star z_o' \in C\} \end{array}$$

where $z_i = ci \star (\!|\, i_C \Rrightarrow i \,|\!)$ and $z_o = (\!|\, o_C \Rrightarrow o \,|\!) \star c$

[2]The x in $[\![C]\!]_x^n$ represents a place holder for any of *dn* for *do-nothing semantics*, *ρ-chaos* for *partial-chaotic semantics*, *τ-chaos*—for *total chaotic semantics*, or *fc* for the *firing conditions semantics*.

4.6.3 Total chaotic semantics

As we identified earlier on page 80, the encoding of the total chaotic semantics in terms of partial relations requires the addition of a distinguished state $\perp$ and distinguished output $\perp$. Adding state $\perp$ is commonly referred to as lifting the relation.

We extend the type $Chart_C$ to include the distinguished binding $\perp$ as follows:

Definition 4.6.8

$$Chart_C^{\perp} =_{def} Chart_C \cup \{\perp\}$$

Now the total chaotic definition is the same as that for the partial chaotic semantics with the exception that elements outside of the precondition of the partial relation (including state $\perp$ with any input) are mapped to the target of the relation (including both the state and output $\perp$). Hence the type of $[\![C]\!]^n_{\tau\text{-}chaos} \subseteq \mathcal{I}^n_C \times \mathcal{O}^n_{C_\perp} \times Chart_C^{\perp}$.

Definition 4.6.9 For arbitrary $si \in \mathcal{I}^n_C$, $so \in \mathcal{O}^n_{C_\perp}$, $i \subseteq in_C$, $o \subseteq out_C^{\perp}$ and $c \in Chart_C^{\perp}$,

$$\begin{array}{lll} [\![C]\!]^0_{\tau\text{-}chaos} & =_{def} & \{(\langle\rangle, \langle\rangle, c') \mid c \in Init_C\} \\ [\![C]\!]^{n+1}_{\tau\text{-}chaos} & =_{def} & \{(si^\frown i, so^\frown o, c) \mid \exists\, ci \bullet (si, so, ci') \in [\![C]\!]^n_{\tau\text{-}chaos} \wedge \\ & & \qquad Pre\ C\ z_i \Rightarrow z_i \star z'_o \in C\} \end{array}$$

where $z_i = ci \star (\!|\, i_C \Rrightarrow i \,|\!)$ and $z_o = (\!|\, o_C \Rrightarrow o \,|\!) \star c$

4.6.4 Firing conditions semantics

Definition 4.6.10 For arbitrary $si \in \mathcal{I}^n_C$, $so \in \mathcal{O}^n_{C_\perp}$, $i \subseteq in_C$, $o \subseteq out_C^{\perp}$) and $c \in Chart_C^{\perp}$,

$$\begin{array}{lll} [\![C]\!]^0_{fc} & =_{def} & \{(\langle\rangle, \langle\rangle, c') \mid c \in Init_C\} \\ [\![C]\!]^{n+1}_{fc} & =_{def} & \{(si^\frown i, so^\frown o, c) \mid \exists\, ci \bullet (si, so, ci') \in [\![C]\!]^n_{fc} \wedge \\ & & \qquad (Pre\ C\ z_i \wedge z_i \star z'_o \in C \vee \\ & & \qquad \neg\ Pre\ C\ z_i \wedge o = \{\perp\} \wedge c = \perp)\} \end{array}$$

where $z_i = ci \star (\!|\, i_C \Rrightarrow i \,|\!)$ and $z_o = (\!|\, o_C \Rrightarrow o \,|\!) \star c$

This completes the description and definition of the trace semantics for μ-Charts. In the following chapter, we introduce the notion of formal refinement and use the trace semantics introduced here to investigate a notion of formal refinement for μ-Charts.

Chapter 5

Trace Refinement

This chapter describes the different notions of refinement for μ-charts. First we describe exactly what we mean by the term "refinement". Unlike some formal languages, such as the refinement calculus [59], that contain syntax to describe both specifications and implementations, μ-Charts contains only one type of construct which is a μ-chart (or simply a chart). One way to view a μ-chart is to consider it as the description of a function (or several functions in the nondeterministic case). Each function is a description of the output trace that results from each possible input trace or in other words a reactive system.

When we talk about a chart specification we typically mean a high-level description of the behaviour of a reactive system. It is considered high-level because it will describe some set of important behaviours and ignore other behaviours; it is a partial description of the required reactive system. A chart that describes several functions is typically considered a specification; there are many reactive systems that implement the specification. At the other end of the spectrum a chart that describes just one function could be considered an implementation. Though μ-Charts is not an implementation language as such, when a chart describes just one trace function then there is no longer any choice involved in implementing the correct reactive system. Of course, in practice it may not be necessary to reduce the set of functions described to just one. It may be the case that there are in fact several reactive systems that are suitable, in which case the implementor simply chooses one.

μ-Charts refinement is a relation between a specification and an implementation. In fact, there is likely to be a long chain of charts between a specification and an implementation. Each chart in this chain is in turn linked by the refinement relation. The transitivity of the relation then guar-

antees that the implementation is correct with respect to the specification. This is a formalisation of the common software engineering notion of stepwise development; each step progresses down this chain. If we consider the specification as describing a set of trace functions then the refinement relation is one of subset. That is, at each step in the development we reduce the number of functions that implement the specification. The development is complete when the set of trace functions is sufficiently small that we can choose one of them as the implementation of the specified reactive system.

It is one thing to define the refinement relation informally but our goal is to define tools that we can use to reason about refinements between charts. That is, we wish to define a refinement calculus for μ-Charts. As we alluded to in Chapter 4, there are two ways that this calculus will typically be used. The first is to use the rules to guide the design decisions made when transforming or refining a chart specification into an implementation. This form of using the refinement calculus closely follows Dijkstra's notion that "we develop program and correctness proof hand in hand" [23]. From the point of view of developing such rules, this is the ideal because the usefulness of the rules can be measured by their eloquence rather than their completeness. The rules guide the developer in the design of a reactive system rather than each particular development needing a new set of rules.

The second type of use, and probably more common, is to take two specifications and show that one is a refinement of the other. This form of development follows more closely the practice against which Dijkstra warns, that is, "first designing the program and then trying to prove its correctness". Given two arbitrary specifications, finding the appropriate proof is likely to be difficult. Also if a proof can be found it is less likely to give insight into or help document design decisions.

Before introducing formal definitions of refinement we split refinement into three distinguishable types—behaviour, input and output interface refinement.

5.1 Behaviour refinement and interface refinement

There are two distinct methods of abstraction that can be employed in a chart specification. The first is the common notion of using nondeterminism to represent choices that have yet to be made about system behaviour. The

second is the description of some important subset of the total behaviour of the reactive system. It is from this observation that the distinction between behaviour and interface refinement arises.

In [78], Scholz introduces this useful distinction but does little by way of investigating or describing exactly what interface refinement is. Interface refinement allows the designer to introduce new interaction between a chart and its environment via refinement. That is, where a specification identifies the set of signals used to interact with the environment (the input and output interface of a chart), there is an assumption that part of the design process may be to extend these signals. Another way to consider this is that the context for which the specification is designed may not be the same as the context in which the eventual system will reside. The specification may have chosen to ignore some interaction as an abstraction method for dealing with complexity or the design process may wish to introduce new features that were not specified in the early stages of design.

We begin by investigating the two separate types of refinement, that is, removing nondeterminism and changing the interface with the environment, separately. Then, like Scholz, we give one definition of the refinement relation that incorporates both activities. As we will see it is almost always the case that interface refinement by itself is uninteresting until we consider it in conjunction with removing nondeterminism.

In fact, the interface refinement that we define is, by itself, not really refinement at all. Changing the interface of a chart in isolation is defined in terms of observational equivalence in terms of context. We will use the symbol $\approx$ to denote that an observational equivalence exists between two charts. We still use the phrase interface refinement to describe such observational equivalences.

Scholz comments that "in general, interface refinements could allow an arbitrary modification of both input and output interfaces", that is, to allow the interfaces to get larger or smaller. The interface refinement that we define here does allow both increasing and decreasing of interfaces, though not in general. Unlike Scholz we do not use the definition to restrict refinements to just those that increase the interfaces of a chart. Rather the definition requires that refinements only ever increase the reactivity of a chart. From such definitions, we show that this does impose some restrictions on how the interfaces of a chart can be changed via refinement. In particular, given an abstract chart A, it is always possible to increase A's interfaces. However, there are only limited situations when the interfaces can be decreased whilst

maintaining A's reactivity.

Two of the reasons that Scholz gives for allowing only increasing interfaces are: refinement via combining sequential charts using the chart operators only ever increases the interface, and allowing both increasing and decreasing interfaces does not result in a transitive refinement relation. We will comment on these reasons in Section 5.4.

Now that we have informally introduced our notion of refinement for charts we introduce some formal structures to help describe refinement and as a tool for comparing the different types of refinement presented.

5.2 The refinement semilattice

In order to discuss the trace definition of chart refinement more formally, we consider refinement in terms of a semilattice. First we will deal with just behaviour refinement and then we will show how this framework is extended to cope with interface refinement. We split chart refinement into three separate types as follows: behaviour refinement $\sqsupseteq_b$; input refinement $\approx_{\mathcal{I}}$; and output refinement $\approx_{\mathcal{O}}$.

As defined in Section 4.6, the infinite trace semantics for a chart C is such that,[1]

$$[\![C]\!]^{\omega}_{x} \subseteq \mathcal{I}^{\omega}_{C} \times \mathcal{O}^{\omega}_{C_{\perp}} \ \wedge \ \mathrm{dom}\,[\![C]\!]^{\omega}_{x} = \mathcal{I}^{\omega}_{C}$$

Notice that the informal discussion above talks about charts as being the description of a set of trace functions but our formalisation of a chart's trace behaviour gives the meaning of a chart as a total relation between input and output traces. In fact, there is a one-to-one correspondence between such a relation and a set of total functions. When the total relation is itself functional then the corresponding set of total functions contains just one function (*i.e.* the relation itself). Hence a chart whose semantics is a function would be considered an implementation. Whenever the total relation is not functional then the corresponding set of total functions contains a different total function for each of the non-functional behaviours in the relation. Hence a non-functional relation represents a chart that is a specification, *i.e.* "not yet" an implementation. This correspondence between total relations and sets of total functions is monotonic with respect to subset. Therefore taking

[1]Note we again use x in $[\![C]\!]^{n}_{x}$ represents a place holder for any of *dn*—for *do-nothing semantics*, *ρ-chaos*—for *partial-chaotic semantics*, *τ-chaos*—for *total chaotic semantics*, or *fc* for the *firing conditions semantics*.

the subset of the relation that represents a chart is the same as reducing the number of total functions that can be said to implement that chart.

Regardless of the particular semantic model that we pick, *e.g.* the do-nothing semantics, the total chaotic sematic model *etc.*, the trace semantics of chart C will be a total relation that is a subset of $\mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$. In three of the four semantic interpretations the reactivity of a chart is inferred by the totality of its trace semantic relation. Of course, the *firing conditions semantics* introduced in Section 4.4 allows us to use charts to model a system that terminates, *i.e.* a system that is not guaranteed to be reactive. The semantic relation is still total but the behaviour, which is to output just the signal $\perp$ forever, is arguably not reactive. In any case, charts cannot refuse input under any of the semantic interpretations.

Behaviour refinement of charts is defined in Definition 5.2.1. In this definition (and most of the definitions that follow in this chapter) we have omitted mention of the parameter x from the left hand side. This is done to simplify an already cluttered notation. The omission does not cause problems since we never mix instantiations of x in in any future definitions, propositions or proofs, *e.g.* we never consider both "C $\sqsupseteq_\beta^{dn}$ A" and "C $\sqsupseteq_\beta^{fc}$ A" as distinct statements within a single definition, proposition or proof. Of course, whenever a statement like $C \sqsupseteq_b A$ is replaced by its definition all occurrences of x that then appear follow the usual rules for bound definition variables, *i.e.* all occurrences of x have to be substituted for by the same value from the set of values that x property x properly ranges over. The variable x is reintroduced in Definition 5.4.1 because this is the generalisation of all of the separate notions that we discuss in this chapter. As such the refinement relation $\sqsupseteq_x^\omega$ is used to link the trace semantics-based refinement to the partial relations-based refinement later.

Definition 5.2.1 For arbitrary charts A and C we have,

$$C \sqsupseteq_b A \;=_{def}\; \forall\, i;\; o \bullet in_C = in_A \wedge out_C = out_A \wedge (i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \Rightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega$$

where $t_{\triangleright(S)}$ restricts the range of the sequence t (pointwise) to the signals in the set S. $C \sqsupseteq_b A$ is pronounced "C refines the behaviour of A".

Informally this definition states that the chart C refines (the behaviour) of chart A if and only if chart C's observable behaviour is a subset of A's, in any input providing context. We introduce this notion of observation and context in more detail in the following. We derive the following introduction

and elimination rules for behaviour refinement.[2]

Proposition 5.2.1 For arbitrary charts A,C, and infinite sequences i and o we have,

$$\frac{C \sqsupseteq_b A}{in_C = in_A}\ (\sqsupseteq^-_{\beta\, I}) \qquad \frac{C \sqsupseteq_b A}{out_C = out_A}\ (\sqsupseteq^-_{\beta\, II})$$

$$\frac{C \sqsupseteq_b A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\beta\, III})$$

$$\frac{in_C = in_A \quad out_C = out_A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \ \vdash\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}{C \sqsupseteq_b A}\ (\sqsupseteq^+_{\beta\, I})$$

$$\frac{}{A \sqsupseteq_b A}\ (\sqsupseteq^+_{\beta\, II})$$

Given this definition, behaviour refinement of charts can now be considered in terms of the following meet semilattice. Given the set $\Theta = \{x \mid x \subseteq \mathcal{I}^{\omega}_C \times \mathcal{O}^{\omega}_{C_\perp} \ \wedge\ \mathrm{dom}\, x = \mathcal{I}^{\omega}_C\}$, the refinement semilattice $\mathcal{R}_A$ is defined by the following partial order.

$$\mathcal{R}_A =_{def} (\Theta, \supseteq)$$

Notice that this semilattice is unusual because the largest element is the bottom element. Hence the meet of any two elements in the lattice (or their greatest lower bound) is the same as their union. The reason it appears to be upside down is so that the discussion here accords with other lattice-based descriptions of refinement where it is typical that a refinement is described as moving up the lattice. That is, the least element represents the least refined chart.

Now we can consider $\mathcal{R}_A$ as a model into which all possible behaviour refinements for chart A fit. The properties that hold of $\mathcal{R}_A$ allow us to describe properties of behaviour refinement. Note that the subscript A in $\mathcal{R}_A$ is used only to denote that the semilattice is parameterised by A's input and output interface. That is, if $C \sqsupseteq_b A$ then $\mathcal{R}_A = \mathcal{R}_C$

A useful way to visualise a partially ordered set is by considering its Hasse diagram. (See [51] for a description of Hasse diagrams for partially ordered sets.)

The simple Hasse diagram of Figure 5.1 gives the general idea. Note that this diagram represents a semilattice whose elements are total relations, such

[2]See Section B.7 for the proofs of all propositions in this section.

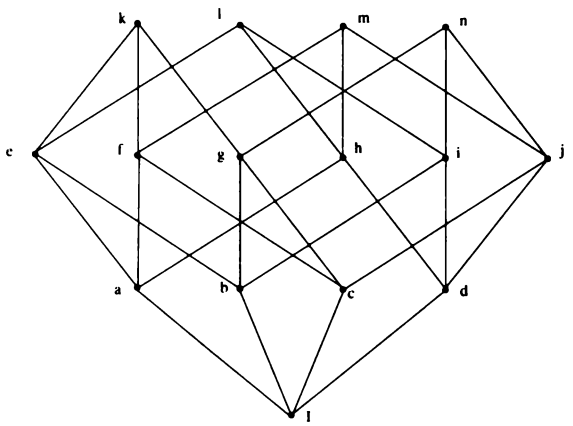


Figure 5.1: Hasse diagram for a simple subset partially ordered set

that $I = \{(x, y) \mid x \in \{e_1, e_2\}, y \in \{e_3, e_4\}\}$ for some elements $e_1 ... e_4$, and has the order $\supseteq$. Given that each element is a total relation, the elements k, l, m and n are functions. As one can imagine, actually drawing this type of diagram for a chart specification is not feasible. We use them here as an aid to description rather than suggesting that they could be used to reason about chart refinements.

For any chart C, $\mathcal{R}_C$ contains a unique least element that we will call $chaos_C$ such that $chaos_C = \mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$. All reactive systems are valid implementations of $chaos_C$. It is at this point that we need to consider again each of the different trace semantic interpretations introduced. The behaviour $chaos_C$ can only be captured by a chart under the total chaotic semantics. Consider the following chart.

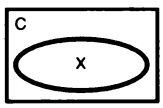


Figure 5.2: Chart C: $in_C = \{a\}$ and $out_C = \{s\}$

The total chaotic interpretation of this chart is $chaos_C$ and therefore $[\![C]\!]^\omega_{\tau\text{-}chaos} = \mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$. The partial chaotic interpretation of this chart is $True_C$, where $[\![C]\!]^\omega_{\rho\text{-}chaos} = \mathcal{I}_C^\omega \times \mathcal{O}_C^\omega$. Note that none of the output traces of C contain the output $\perp$ in this case. Under both the do-nothing and firing conditions semantics this chart represents a chart that does nothing for any inputs. That is, it either outputs nothing for ever input, or "terminates" (outputs just $\perp$) respectively. In both cases the semantics are functional relations over C's interfaces and therefore cannot be (behaviourally) refined.

Hence, the refinement semilattice $\mathcal{R}_C$ is general enough to represent each

of the semantic interpretations, but the chosen interpretation determines which element, of the lattice, any chart describes. For any chart that is totally defined the trace semantics is the same under any of the interpretations. Consider chart B of Figure 5.3.

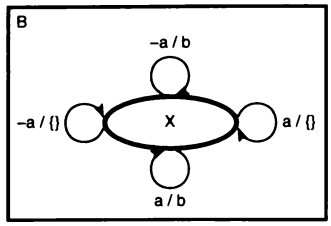


Figure 5.3: Chart B: $in_B = \{a\}$ and $out_B = \{b\}$

We have that $[\![B]\!]^{\omega}_{\tau} = \mathcal{I}^{\omega}_B \times \mathcal{O}^{\omega}_B = True_B$ under all four semantic interpretations. Any total trace function that maps all input traces over the set $\{a\}$ to arbitrary output traces over the set $\{b\}$ is a valid implementation of the chart B.

In general, we can say that specifying a reactive system using a chart C is analogous to picking a particular element in the order $\mathcal{R}_C$. Refinement of that chart is then the process of following the edges in the Hasse diagram of $\mathcal{R}_C$ from that point upwards. For example, a chart that specifies the point a in our example Hasse diagram (Figure 5.1) is implemented by any of the functional relations k, l and m. If the first design step is a refinement analogous to taking the edge from point a to point f, then the effect is to reduce the possible implementations to just those represented by k and m. The final design step is then a choice between the implementations k and m themselves.

Note that $\mathcal{R}_C$ is not a complete lattice, it does not contain a top element. If it was complete we could show that for any two specifications A and B there exists a least upper bound T. This T would itself be a specification in the refinement order and therefore any implementation of T would also be a valid implementation of A and B. In other words, if $\mathcal{R}_C$ was a complete lattice then there would be at least one implementation that satisfied any two specifications. That $\mathcal{R}_C$ is not complete highlights the fact that the chart trace refinement is quite different from the *trace refinement* of process algebras such as CSP [42]. In such refinement theories trace semantics are often defined as the prefix closed set of finite observable traces that a process can exhibit. *Trace refinement* is then defined as a subset relation over

these sets. Hence the set containing the empty trace, *i.e.* the process that does nothing, is a valid refinement of any process. This is not the case for charts. By definition, all charts are reactive and therefore their semantics are defined over all input sequences. Recall that each element in the semilattice $\mathcal{R}_C$ represents a total relation with domain equal to the set of input traces $\mathcal{I}_C^\omega$. Thus refinement allows us to reduce nondeterminism, thereby removing output sequences for an input sequence that is mapped to several output sequences, but never remove all output sequences mapped to a given input.

Another attribute of the refinement order $\mathcal{R}_C$ is that it contains points that cannot be described by any chart. While there are examples such as $chaos_C$ that can be described in one interpretation but not others there are other elements in the lattice that cannot be described in any of the semantic interpretations. This can be shown using the following arguments.

There is an infinite number of subsets of $\mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$ and therefore an infinite number of points in the refinement semilattice. Charts, however, are finite in state and transition and therefore there is only a finite number of charts one can write down. The remaining elements of the lattice cannot be described by a chart. The subset relation used to define refinement can distinguish between individual infinite sequences. Charts on the other hand can at best distinguish between a class of infinite sequences with common finite subsequences. Consider for example the following chart C.

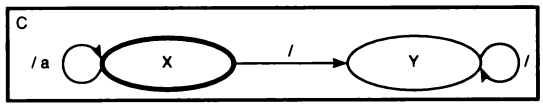


Figure 5.4: Chart C: $in_C = \{\}$ and $out_B = \{a\}$

Now using the notational convention $\varnothing^\omega = \langle\{\},\{\},...\rangle$, $a^\omega = \langle\{a\},\{a\},...\rangle$ and $a^* = \{\langle\{a\}_1,\{a\}_2,...\{a\}_n\rangle \mid n \in \mathbb{N}\}$ we have that,

$$[\![C]\!]_r^\omega = \{\varnothing^\omega\} \times (\{y^\frown\varnothing^\omega \mid y \in a^*\} \cup a^\omega)$$

Now consider the relation $R = \{\varnothing^\omega\} \times \{x^\frown\varnothing^* \mid x \in a^\omega\}$, *i.e.* a similar relation to $[\![C]\!]_r^\omega$ with the exception that there is no mapping to the infinite sequence of a's. It is the case that $R \subseteq [\![C]\!]_r^\omega$, and therefore R represents a point in $\mathcal{R}_C$. However, there does not exist a chart (with finite states) that has the same meaning as R. This is clear from the definition of the chart trace semantics and because the relation R cannot be the limit of any chain of finite relations. The only possible candidate chain would be $\{R \restriction n \mid n \in \mathbb{N}\}$.

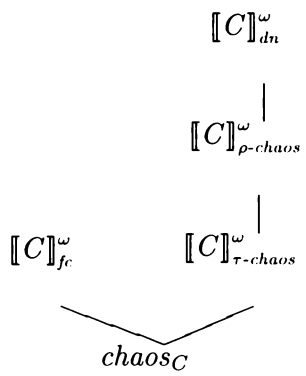
However, the least upper bound of this chain must contain $\varnothing^\omega \mapsto a^\omega$ and therefore is not equal to R itself.

Yet another way to explain these points in the refinement order is as the limits of infinite chains. That is, we can have a refinement sequence of the form $C \sqsupseteq_b C_1 \sqsupseteq_b C_2 \sqsupseteq_b \ldots$ where C_1 has more states than C and C_2 has more states than C_1 and so forth. This sequence can represent an infinite chain in the refinement order. The limit of such a chain would represent a chart with an infinite number of states and therefore is another example of a point in the refinement order that is not representable by a chart.

As we have said, assigning different semantic interpretations to the same syntactic chart C can be thought of, in general, as picking a different point in the refinement semilattice $\mathcal{R}_C$. Hence even though the definition of refinement in terms of the partial relations semantics appears very different for each of the alternate semantics, the differences can be attributed to the implicit encoding of the specific trace interpretation using a single partial relation semantics.

On the other hand, in terms of trace semantics, refinement under each of the alternative interpretations is defined by the same relation. Therefore in the following we can investigate this trace refinement relation in general. The results apply to each of the semantic interpretations because the interpretation is already encoded in the traces of the chart.

Naturally it follows that if we start from a chart for which each of the trace semantic encodings agree, such as chart B of Figure 5.3 then the refinements available under each interpretation are equivalent. More generally though, we can describe how each of the trace semantic interpretations fit into the lattice framework using the following diagram:



That $chaos_C = \mathcal{I}^\omega_C \times \mathcal{O}^\omega_{C_\perp}$ is the least element in this order demonstrates that each of the possible trace interpretations of the chart C picks an element

from the semilattice $\mathcal{R}_C$. The order represented by the diagram is the same as $\mathcal{R}_C$ itself. Hence, we can see that the do-nothing meaning of C can always be considered a refinement of the partial chaotic meaning. The partial chaotic meaning is in turn always a refinement of the total chaotic meaning of C. The firing conditions interpretation, however, is in general unrelated to any of the other interpretations.

This order between the alternate semantics demonstrates that $[\![C]\!]^{\omega}_{\tau\text{-}chaos}$ is the greatest lower bound of $[\![C]\!]^{\omega}_{dn}$ and $[\![C]\!]^{\omega}_{\rho\text{-}chaos}$ in $\mathcal{R}_C$. Thus it is the most general semantic model of the three. In particular, if we assign the total chaotic semantics to chart C, there always exists a valid refinement of C that is to change its meaning (explicitly) to have behaviour consistent with one of the other two. The firing conditions semantics, however, allows us to describe behaviour that is quite different to the other three interpretations.

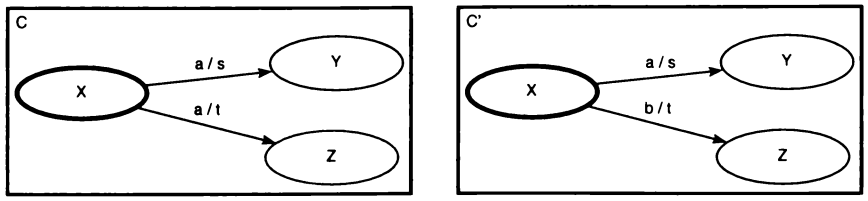
5.3 Interface refinement and $\mathcal{R}$

As we have already identified, the interface refinement of a chart changes the input and/or output interface of the chart. This is considered a useful form of refinement because the partial specification of a reactive system may ignore some signals. Hence during design the chart's behaviour with respect to these previously ignored signals is defined.

While behaviour refinement $\sqsupseteq_b$ allows only the reduction of nondeterminism, interface refinement may or may not allow the introduction of additional nondeterminism. In the following we we make more precise the concept of nondeterminism with respect to charts; we extend the semilattice framework to encompass interface refinement; and then consider each of the interface refinement relations $\approx_{\mathcal{I}}$ and $\approx_{\mathcal{O}}$ in turn.

Process algebras, such as CSP, often distinguish processes that model internal choice from processes that offer external choice. In a similar fashion we say that a chart contains internal choice (or is nondeterministic) when, given some input, the chart has a choice between different resulting states and/or different outputs. For example consider the chart in Figure 5.5(a). When chart C is in state X and is presented with an input $\{a\}$ the chart can nondeterministically choose to give the output $\{s\}$ or the output $\{t\}$. This is an example of *internal choice*. We define *external choice* to mean that the environment can make the choice between two transitions. For example consider the chart in Figure 5.5(b). When chart C' is in state X the environment can choose the transition taken by giving either the input $\{a\}$

or the input $\{b\}$. Note that in process algebras it is also common to refer to a process that exhibits no internal choice as deterministic. We use the term deterministic in the same fashion.



(a) Internal nondeterminism (b) External nondeterminism

Figure 5.5: Internal *vs* External Nondeterminism

Notice that the chart C' of Figure 5.5(b) actually demonstrates a situation where the environment can make the choice (by giving input $\{a\}$ or $\{b\}$) or allow an internal choice by giving the input $\{a, b\}$. That is, chart C' contains both internal and external choice. We use this distinction between internal and external choice in describing the effects of interface refinements.

We extend the $\sqsupseteq_b$ semilattice framework to encompass interface refinement. When the input and/or output interface of a chart is increased the size of the relation $chaos_C$ (*i.e.* the relation $\mathcal{I}_C^\omega \times \mathcal{O}_{C_\perp}^\omega$) increases. Because $\mathcal{R}_C$ is similar to a subset semilattice with respect to $chaos_C$, it is clear that both the number of elements in the refinement semilattice and the number of relationships between those elements increase when the size of $chaos_C$ increases. In simpler terms, when we increase the interface(s) of a chart the Hasse diagram gets bigger. A nice way to consider interface refinement with respect to the Hasse diagram model is to imagine the two dimensional diagram for $\mathcal{R}_C$, which we have already described, and add to this a third dimension (out of the page). As before, following an edge up the diagram is a behaviour refinement. Also, we can now follow an edge that moves further away from the page that represents an interface refinement. Hence if we have that $C \approx_{\mathcal{I}} A$ (assuming chart C increases the input interface of chart A) then this three dimensional diagram has an edge that moves us from $\mathcal{R}_A$ to $\mathcal{R}_C$ where $\mathcal{R}_C$ is a larger semilattice defined over chart C's interfaces. Hence we can imagine this more general model as a series of Hasse diagrams one behind the other. Each of these gets progressively bigger as we move further into the page. Refinement, *i.e.* of behaviour and interface, can typically be considered as moving from one point in this diagram (the specification) through a series

of upwards and forward edges to another point (the implementation). We denote this more general model, *i.e.* the set of semilattices $\mathcal{R}_C$ for all charts C, simply as $\mathcal{R}$.

This framework can be used to describe all possible μ-chart refinements. Consider yet again the chart *Chaos* of Figure 5.6 under a total chaotic semantic interpretation. This chart represents the bottom element of the refinement order $\mathcal{R}$.

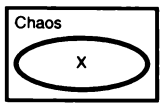


Figure 5.6: *Chaos*: $in_C = \{\}$ and $out_C = \{\}$

The reason that this appears to be a very strange type of chaos, *i.e.* its only input trace is an infinite sequence containing the empty input while its output traces represent all combinations of outputting nothing or the output $\perp$, is because it has an empty input and output interface. In fact, this chart is both refined by and a refinement of the chaotic chart over any input and output interfaces. The reason that we consider it the bottom of the refinement semilattice $\mathcal{R}$ is because it is the least description of chaos. For example we demonstrate that the chart *Chaos* is equivalent to the chart C of Figure 5.2 on page 97. Consider the refinement sequence $C \approx_{\mathcal{O}} A \approx_{\mathcal{I}} Chaos$. Here we assume that chart A refines the input interface of *Chaos* by adding the signal a to the input interface. Then chart C refines the output interface of A by adding the signal s to A's output interface. The net effect of these two interface refinements moves us from the bottom element of $\mathcal{R}$ to the point that is the bottom element of $\mathcal{R}_C$. The semilattice $\mathcal{R}_C$ has several elements and therefore several possible behaviour refinements.

Another reason why *Chaos* is considered the bottom element of $\mathcal{R}$ is that almost all practical refinement steps will entail increasing the number of signals in a chart's interface. As we will see, increasing reactivity is the fundamental requirement of chart refinement.

5.3.1 Input refinement

From the outset interface refinement is easily mistaken for the common notion of weakening preconditions that is found in other refinement theories such as that for Z. This weakening of preconditions refers to extending the number

of states (in the original state space) over which the operation is defined. In Z, this means reducing the number of states in which nondeterminism is exhibited by chaotic behaviour.

This, however, is not the correct analogy for interface refinement. The correct analogy in a Z refinement theory would be a refinement that extends the state space itself. In other words, one might say we are applying an operation in a state that has more observables than the operation originally considered. Rather than the simpler notion of weakening preconditions this type of refinement can be considered as simulating each of the original states by a set of states. In terms of the relational view of an operation, this means that the relation is uniformly expanded at each point of the domain and range. For example consider the following relations,

Assuming the left hand relation describes some operation over the observations of state S, then the right hand relation is representative of applying that operation over an extended state in which observations can be made of both states S and T, noting that S and T are assumed to be disjoint.

Returning to input refinement defined over the trace semantics of a chart we see a similar effect. Increasing the signals in the input interface of a chart C extends the relation $[\![C]\!]^{\omega}_{x}$ at each point in its domain. The shape of the relation changes in a similar fashion to the relation demonstrated in our example above, with the exception that the range remains constant. Hence, each of the new input sequences introduced by extending the input interface can be uniformly related back to one of the original input sequences by the pointwise restriction of each element of the sequence.

Input refinement is defined as follows.

Definition 5.3.1 For arbitrary charts A and C we have,

$$C \approx_{\mathcal{I}} A \;=_{def}\; \forall i;\; o \bullet (i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \\ \wedge\; out_C = out_A$$

where $i_{\triangleright(in_X)}$ restricts the range of the sequence i (pointwise) to the signals in the set in_X.

Until now we have considered the trace semantics for a chart C, formally, as a relation between input and output traces such that $[\![C]\!]^{\omega}_{x} \subseteq \mathcal{I}^{\omega}_{C} \times \mathcal{O}^{\omega}_{C_\perp}$

and $\mathrm{dom}\,[\![C]\!]^\omega_x = \mathcal{I}^\omega_C$. That is, we have really described only the extension of the relation. For the purpose of proving the following rules we point out that $[\![C]\!]^\omega_x$ is a proper relation with target and source sets that are $\mathcal{I}^\omega_C$ and $\mathcal{O}^\omega_{C_\perp}$ respectively. Therefore, it follows from the equality $[\![C]\!]^\omega_x = [\![A]\!]^\omega_x$ that $out_C = out_A$, and similarly for the respective input interfaces.

From Definition 5.3.1 we can derive the following introduction and elimination rules. The proofs for these rules are given in Section B.8.

Proposition 5.3.1 For arbitrary charts A and C, signal set ins and infinite sequences i, i' and o we have,

$$\frac{C \approx_{\mathcal{I}} A}{out_C = out_A}\ (\sqsupseteq^-_{\mathcal{I}\,I}) \qquad \frac{C \approx_{\mathcal{I}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II})$$

$$\frac{C \approx_{\mathcal{I}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,III}) \qquad \frac{C \approx_{\mathcal{I}} A \quad in_C = in_A}{[\![C]\!]^\omega_x = [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,IV})$$

$$\frac{C \approx_{\mathcal{I}} B \quad B \approx_{\mathcal{I}} A}{C \approx_{\mathcal{I}} A}\ (\sqsupseteq^-_{\mathcal{I}\,V}) \qquad \frac{C \approx_{\mathcal{I}} A \quad ins = in_A \cap in_C}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,VI})$$

$$\frac{C \approx_{\mathcal{I}} A}{A \approx_{\mathcal{I}} C}\ (\sqsupseteq^-_{\mathcal{I}\,VII}) \qquad \frac{in_A \subseteq ins \qquad \begin{array}{l} in_B = ins, \\ out_B = out_A, \\ B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{I}\,VIII})$$

$$\frac{ins \subset in_A \qquad \begin{array}{l} (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x \Leftrightarrow \\ (i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x \end{array} \qquad \begin{array}{l} in_B = ins, \\ out_B = out_A, \\ B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{I}\,IX})$$

where we assume the usual conditions for B and P in $(\sqsupseteq^-_{\mathcal{I}\,6})$ and $(\sqsupseteq^-_{\mathcal{I}\,7})$.

$$\frac{out_C = out_A \qquad (i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}{C \approx_{\mathcal{I}} A}\ (\sqsupseteq^+_{\mathcal{I}\,I})$$

$$\frac{[\![C]\!]^\omega_x = [\![A]\!]^\omega_x}{C \approx_{\mathcal{I}} A}\ (\sqsupseteq^+_{\mathcal{I}\,II})$$

Notice in our discussion we commonly refer to input refinement as though there is a proper order imposed by the definition. That is, an input refinement $A \approx_{\mathcal{I}} C$ only increases the signals in A's input interface. Yet the definition does not necessarily impose any such order. Interface refinement is defined specifically to capture an observational equivalence. The input and output interfaces define the context or environment that we assume for the chart. The definition of input refinement is given precisely to model the fact that placing a chart in a new (input providing) context should not change the chart's observable behaviour. The notional order of input refinement comes from the fact that you can always increase the input interface, and doing so allows for new interesting behaviour refinement. It is not always possible to decrease a chart's input interface and when it is, the result is typically uninteresting. Also, assuming $C \approx_{\mathcal{I}} A$ where the input interfaces of A and C are not strongly related, then there always exists another observably equivalent chart whose input interface contains just those input signals common to both A and C. We formalise these three observations in the following lemma.

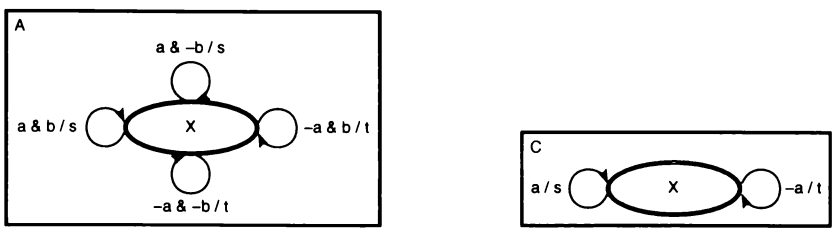
Lemma 5.3.2 For arbitrary charts A and C, traces i and o, and signal set ins,

$$\frac{in_A \subseteq ins}{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}$$

$$\frac{ins \subset in_A}{(\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A) \Leftrightarrow ((i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x})}$$

$$\frac{C \approx_{\mathcal{I}} A}{\exists B \bullet in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}$$

Consider the example in Figure 5.7 that is representative of all situations in which a valid input refinement decreases the input interface.



(a) $in_A = \{a, b\}$, $out_A = \{s, t\}$ (b) $in_C = \{a\}$, $out_C = \{s, t\}$

Figure 5.7: Decreasing an input interface

We can see from this example, in which C refines A and $in_C \subset in_A$, that decreasing the input interface of A merely captures the fact that the input signal b had no bearing on the output produced by chart A.

Now we describe some further implications of the definition of $\approx_{\mathcal{I}}$ in terms of the nondeterminism that the chart under refinement contains. The following definition formalises what it means for a chart to be deterministic.

Definition 5.3.2 For the arbitrary chart C,

$$\det C \ =_{def} \ \forall i, o, o' \bullet (i, o) \in [\![C]\!]^{\omega}_{x} \wedge (i, o') \in [\![C]\!]^{\omega}_{x} \Rightarrow o = o'$$

The first situation that we consider is taking an arbitrary chart A that is deterministic and increasing its input interface to give a new chart C, *i.e.* we have that $C \approx_{\mathcal{I}} A$. As expected the relation $\approx_{\mathcal{I}}$ guarantees that chart C will also be deterministic. Recall that a deterministic chart can be considered a reactive system implementation. That increasing the input interface of an implementation does not change its behaviour demonstrates that we can place the implementation in any (input providing) context and it remains deterministic. This is particularly important for μ-Charts because they allow negated signals in their transition guard. That this property holds guarantees that the guard $\text{-}a$ means the signal a does not occur as input. Consider again chart C pictured in Figure 5.7(b). It is clear that an implementation of C should give the output t for any input that does not include a.

We formalise this property in the following lemma.

Lemma 5.3.3 For arbitrary charts A and C we have,

$$\frac{\det A \quad C \approx_{\mathcal{I}} A}{\det C}$$

The situation in which chart A is nondeterministic is more complicated. Here the refined chart C has the same internal choice as A but has the ability to offer additional external choice (see for example Figure 5.9). In order to explain exactly what this means we need to consider input refinement and behaviour refinement together. To do so we introduce two additional types of refinement that distinguish the order in which respective refinements are performed.

The first of these is *weak input refinement* which is defined as follows.

Definition 5.3.3 For arbitrary charts A and C we have,

$$C \sqsupseteq_{\iota} A \ =_{def} \ \exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_{b} A$$

The reason we term this type of interface refinement weak is because, like behaviour refinement, it allows only the reduction of internal choice (see Lemma 5.3.10 for proof). Using input refinement in this fashion achieves no more than using just behaviour refinement itself. We can see from the definition that the chart B is observationally equivalent to chart C. That is, in any context chart B and chart C are indistinguishable—any implementation of chart B is an acceptable implementation for chart C. Moreover, given the nontrivial refinement $C \sqsupseteq_\iota A$ such that $in_A \subset in_C$, the chart B is preferable to chart C. Chart B has a simpler definition because it mentions only signals in the smaller signal set in_A.

Not surprisingly, the class of available refinement is larger if we increase a chart's input interface and then eliminate internal choice rather than eliminate choice and then increase the interface.

Figure 5.8 illustrates this relationship.

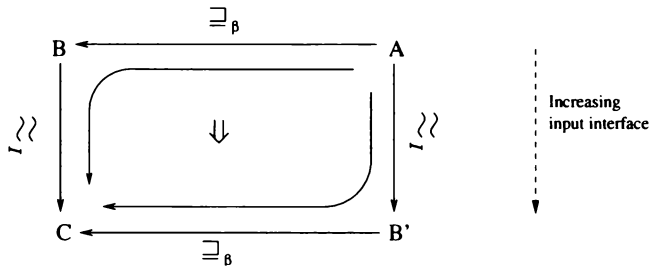


Figure 5.8: Commuting diagram of behaviour and input refinement

That this implication holds is shown by the proof of Lemma 5.3.4 in Appendex B, page 226.

Lemma 5.3.4 For arbitrary charts A, B and C we have,

$$\frac{in_A \subset in_C \quad C \approx_{\mathcal{I}} B \quad B \sqsupseteq_b A}{\exists B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{I}} A}$$

Weak input refinement describes just those refinements that can be derived by following the top path in this semi-commuting diagram. Of course, this implies that there is also an equal derivation following the bottom path.

The other category of input refinement that we introduce is *angelic refinement*. Angelic refinement describes just those refinements for which there exists a bottom path but no associated top path. Angelic refinement is defined as follows:

Definition 5.3.4 For arbitrary charts A and C we have,

$$C \sqsupseteq_{\mathcal{A}} A \;=_{def}\; (\exists B \bullet C \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A) \wedge \neg (C \sqsupseteq_\iota A)$$

The reason we term this *angelic refinement* is because the original internal nondeterminism, which is being removed, can be considered intentional. That is, the inclusion of internal choice is used intentionally in the specification as a method of abstraction. Thus one might consider its meaning as angelic in that it is assumed that any implementation of the specification will make the appropriate choice depending on the context. Angelic refinement is then the process of defining exactly how that choice is made externally.

While weak input refinement allows only the reduction of internal choice, angelic refinement transforms internal choice into external choice. For example consider the refinement sequence (which reads right to left in Figure 5.9).

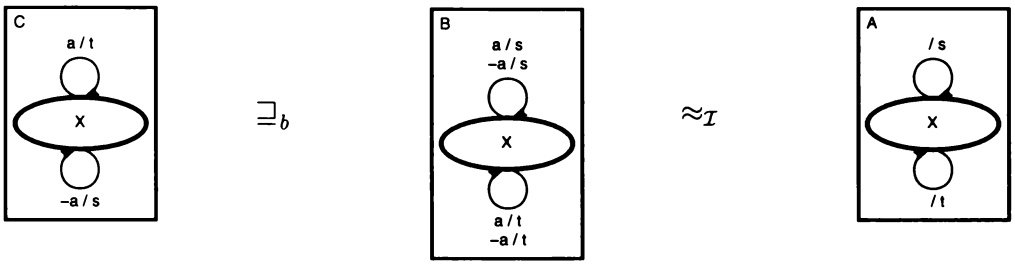


Figure 5.9: Changing internal choice into external choice

We see from this sequence that the internal choice in chart A has become an external choice in chart C. In fact, angelic refinement always treats internal choice as intentional, that is, it does not allow internal choice to be implemented by simply choosing one of the branches.

From the definition of angelic refinement we derive the following introduction and elimination rules.

Proposition 5.3.5 For arbitrary charts A and C, and infinite sequences i and o we have,

$$\frac{C \sqsupseteq_{\mathcal{A}} A \quad C \sqsupseteq_b B, B \approx_{\mathcal{I}} A \vdash P}{P} \; (\sqsupseteq^-_{\mathcal{A}\, I}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A}{\neg\,(C \sqsupseteq_\iota A)} \; (\sqsupseteq^-_{\mathcal{A}\, II})$$

$$\frac{C \sqsupseteq_{\mathcal{A}} A}{out_C = out_A} \; (\sqsupseteq^-_{\mathcal{A}\, III}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x} \; (\sqsupseteq^-_{\mathcal{A}\, IV})$$

$$\frac{C \sqsupseteq_{\mathcal{A}} A}{\neg\,(in_C \subseteq in_A)} \; (\sqsupseteq^-_{\mathcal{A}\, V}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \notin [\![A]\!]^{\omega}_x}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \notin [\![C]\!]^{\omega}_x} \; (\sqsupseteq^-_{\mathcal{A}\, VI})$$

assuming the usual conditions for B and P.

To prove that angelic refinement coincides precisely with changing internal into external choice we return to considering the semilattice $\mathcal{R}$. For all charts A and C, such that $in_A \subset in_C$, the observational equivalence relation between such charts (*i.e.* $C \approx_{\mathcal{I}} A$) can be considered a relation between the respective semilattices $\mathcal{R}_C$ and $\mathcal{R}_A$. That is, just as we described previously, all input refinements link a point in $\mathcal{R}_C$ to a point in $\mathcal{R}_A$.

First we show that this relation is injective, *i.e.* functional and one-to-one.

Lemma 5.3.6 For arbitrary charts A, A', C and C' we have,

$$\frac{in_C = in_{C'} \quad C \approx_{\mathcal{I}} A \quad C' \approx_{\mathcal{I}} A}{[\![C]\!]^{\omega}_{x} = [\![C']\!]^{\omega}_{x}} \qquad \frac{in_A = in_{A'} \quad C \approx_{\mathcal{I}} A \quad C \approx_{\mathcal{I}} A'}{[\![A]\!]^{\omega}_{x} = [\![A']\!]^{\omega}_{x}}$$

We have already shown in the first property of Lemma 5.3.2 that the relation is surjective. That is, each point in $\mathcal{R}_A$ is related to a point in $\mathcal{R}_C$. Moreover, by definition the semilattice $\mathcal{R}_C$ contains more elements than the corresponding $\mathcal{R}_A$. Therefore, it follows that $\approx_{\mathcal{I}}$ is not a total relation between the points in $\mathcal{R}_C$ and $\mathcal{R}_A$. That is, there are some points in $\mathcal{R}_C$ that are not related by $\approx_{\mathcal{I}}$ to $\mathcal{R}_A$.

Now we show that the domain of the relation $\approx_{\mathcal{I}}$ is equal to the domain of the weak input refinement relation $\sqsupseteq_{\iota}$.

Lemma 5.3.7 For the arbitrary charts A and C we have,

$$\frac{C \approx_{\mathcal{I}} A}{C \sqsupseteq_{\iota} A} \qquad \frac{C \sqsupseteq_{\iota} A}{\exists A' \bullet C \approx_{\mathcal{I}} A'}$$

Hence, $\sqsupseteq_{\iota}$ is not a total relation either. However, each chart C in $\mathcal{R}_C$, which is neither an input nor weak input refinement of some chart A, is nevertheless an angelic refinement of some chart A in $\mathcal{R}_A$. That is, A is derived from C by changing internal choice into external choice.

To prove this is the case, we start by showing that any chart C is related by refinement to some chart with a smaller interface. That is, chart refinement in general is a total relation between $\mathcal{R}_C$ and $\mathcal{R}_A$. It is trivial to show that for any chart C, $C \sqsupseteq_{b} chaos_C$ (where $chaos_C = \mathcal{I}_C^{\omega} \times \mathcal{I}_C^{\omega}$). Furthermore, for any chart A, that $chaos_C \approx_{\mathcal{I}} chaos_A$ is also easily shown. Hence, chart C always refines $chaos_A$ for any A.

Following from this we can easily show that $\sqsupseteq_{\iota} \cup \sqsupseteq_{\mathcal{A}}$ is a total relation between $\mathcal{R}_C$ and $\mathcal{R}_A$ (at least between those points that represent charts).

Lemma 5.3.8 For an arbitrary chart C and signal set ins,

$$\frac{ins \subseteq in_C}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_{\iota} A \vee C \sqsupseteq_{\mathcal{A}} A)}$$

By definition angelic refinement and weak input refinement do not coincide. Also, we have shown that $\approx_{\mathcal{I}} \cup \sqsupseteq_{\mathcal{A}}$ is a total relation from $\mathcal{R}_C$ to $\mathcal{R}_A$ where both $\approx_{\mathcal{I}}$ and $\sqsupseteq_{\iota}$ are not. Therefore, it follows that the relation $\sqsupseteq_{\mathcal{A}}$ allows refinements that cannot be achieved using either input or weak input refinement.

The set of angelic refinements is exactly those refinements that can be characterised as transforming internal choice into external choice.
Lemma 5.3.9 formalises the fact that any angelic refinement introduces additional external choice.

Lemma 5.3.9 For arbitrary charts A, C we have that,

$$\frac{C \sqsupseteq_{\mathcal{A}} A}{\exists\, i, i', o \bullet i_{\triangleright(in_A)} = {i'}_{\triangleright(in_A)} \wedge (i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_{x} \wedge ({i'}_{\triangleright(in_C)}, o) \notin [\![C]\!]^{\omega}_{x}}$$

In words, given $C \sqsupseteq_{\mathcal{A}} A$, the environment has more choice about the output from chart C than it did about the output from chart A. This is represented by the fact that two different input sequences i and i', which cause different outputs in chart C, are considered equivalent by chart A. Since, by definition $[\![C]\!]^{\omega}_{x}$, is a total relation it follows from Lemma 5.3.9 that chart C is defined over the input sequence i' but must give output that differs from o. Therefore the environment can ask the output o of chart C using input sequence i or alternatively, it can choose output o' using the input i'. This choice was not available in the chart A.

Finally, weak input refinement (and therefore input refinement) cannot introduce additional external choice.

Lemma 5.3.10 For arbitrary charts A, C, input sequences i and i', and all output sequences o we have,

$$\frac{C \sqsupseteq_{\iota} A}{i_{\triangleright(in_A)} = {i'}_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_{x} \Rightarrow ({i'}_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_{x})}$$

Now we have shown that weak input refinement and angelic refinement are both disjoint and together describe the entire class of input refinements. Therefore, we can characterise input refinement by saying that it is useful exactly (and only) when we wish to change internal choice into external choice, *i.e.* for angelic refinement. As we demonstrated with the introduction of weak input refinement, if angelic refinement is not intended then input refinement is not necessary.

5.3.2 Output refinement

Like input refinement, we give the definition of output refinement based on a notion of observational equivalence. Unlike input refinement, we cannot claim observational equivalence in all contexts; by definition the observations that one can make of a chart are precisely the outputs that the chart gives for some input. While input refinement did not change the nature of those outputs, clearly, increasing the output interface of a chart must.

The choice of the appropriate observational equivalence, on which we base our definition of output refinement, is again motivated by ensuring that refinement never decreases reactivity of a chart. Hence, for any input sequence, a refined chart must exhibit at least the same amount of output information as the original specification. This simple restriction may appear to require that refinement never decreases the output interface. There is however one special case where the defined output refinement allows the output interface to decrease. This type of refinement maintains reactivity at the level of the control information that a chart provides rather than at the level of individual signals. We give an example and explain this in more detail in the following.

Now, $C \approx_{\mathcal{O}} A$ implies charts A and C are observably equivalent in a context that can observe the signals in $out_A \cap out_C$. In words, A and C are observably equivalent if we observe just those output signals that are common to both A and C. Because refinement only allows an output signal to be removed from a chart's interface when the chart's behaviour is totally chaotic with respect to that signal—in practice, we assume that a designer will rarely write such a specification—output refinement will predominantly be used to increase the output interface. In this case, $C \approx_{\mathcal{O}} A$ implies there is no observable difference between A and C in a context expecting A. That is, C and A are indistinguishable in an environment that recognises only the signals that A could output.

In terms of existing internal choice, no subtle behaviour arises from extending the output interface of a chart. As expected, when one adds a new output signal to a chart each transition can either output that signal or not. That is, adding a new signal is the same as adding an additional copy of each existing transition to the chart and then adding the new output to each of these new transitions. The new transitions output their original signals together with the new signal. The chart can now choose either the new or old transition nondeterministically. The refined chart contains more internal

choice regardless of whether the original chart was deterministic or not.

We define output refinement as follows.

Definition 5.3.5 For arbitrary charts A and C we have,

$$C \approx_{\mathcal{O}} A \;=_{def}\; \forall i;\; o \bullet (i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega} \Leftrightarrow (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega}$$

where $o_{\triangleright(out_X^{\perp})}$ restricts the range of the sequence o (pointwise) to the signals in the set $out_X^{\perp}$.

From this definition we derive introduction and elimination rules.

Proposition 5.3.11 For arbitrary charts A and C, signal set *outs* and infinite sequences i, o and o' we have,

$$\frac{C \approx_{\mathcal{O}} A}{in_C = in_A}\ (\sqsupseteq^{-}_{\mathcal{O}\,I}) \qquad \frac{C \approx_{\mathcal{O}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega}}{(i_{\triangleright(out_A^{\perp})}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega}}\ (\sqsupseteq^{-}_{\mathcal{O}\,II})$$

$$\frac{C \approx_{\mathcal{O}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega}}{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega}}\ (\sqsupseteq^{-}_{\mathcal{O}\,III}) \qquad \frac{C \approx_{\mathcal{O}} A \quad out_C = out_A}{[\![C]\!]_x^{\omega} = [\![A]\!]_x^{\omega}}\ (\sqsupseteq^{-}_{\mathcal{O}\,IV})$$

$$\frac{C \approx_{\mathcal{O}} B \quad B \approx_{\mathcal{O}} A}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^{-}_{\mathcal{O}\,V}) \qquad \frac{C \approx_{\mathcal{O}} A \qquad outs = out_A \cap out_C}{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega} \Leftrightarrow (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]_x^{\omega}}\ (\sqsupseteq^{-}_{\mathcal{O}\,VI})$$

$$\frac{C \approx_{\mathcal{O}} A}{A \approx_{\mathcal{O}} C}\ (\sqsupseteq^{-}_{\mathcal{O}\,VII}) \qquad \frac{out_A \subseteq outs \qquad \begin{array}{l} out_B = outs, \\ in_B = in_A, \\ B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\ (\sqsupseteq^{-}_{\mathcal{O}\,VIII})$$

$$\frac{outs \subset out_A \qquad \begin{array}{l} (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega} \Leftrightarrow \\ (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]_x^{\omega} \end{array} \qquad \begin{array}{l} out_B = outs, \\ in_B = in_A, \\ B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\ (\sqsupseteq^{-}_{\mathcal{O}\,IX})$$

where we assume the usual conditions for B and P.

$$\frac{in_C = in_A \qquad \begin{array}{l} (i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega} \Leftrightarrow \\ (i_{\triangleright(in_A)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega} \end{array}}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^{+}_{\mathcal{O}\,I})$$

$$\frac{[\![C]\!]_x^{\omega} = [\![A]\!]_x^{\omega}}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^{+}_{\mathcal{O}\,II})$$

Our definition of output interface refinement again differs from that of Scholz. Similar to input refinement, the definition of output refinement has no explicit constraint that restricts refinements to those that increase the output interface of a chart. However, there is a similar implicit order imposed on output refinements. Consider the following properties that are consequences of the definition of output refinement.

Lemma 5.3.12 For any arbitrary abstract chart specification A and signal set $outs$,

$$\frac{out_A \subseteq outs}{\exists B \bullet out_B = outs \wedge B \approx_{\mathcal{O}} A}$$

$$\frac{C \approx_{\mathcal{O}} A}{\exists B \bullet out_B = out_A \cap out_C \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}$$

$$\frac{out_C \subset out_A \quad C \approx_{\mathcal{O}} A}{[\![A]\!]^{\omega}_{x} = [\![C \mid \{\} \mid True]\!]^{\omega}_{x}}$$

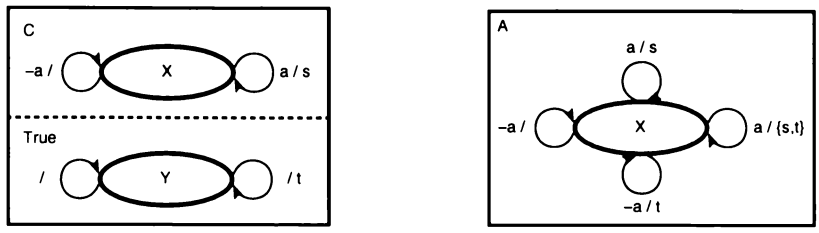
where $True$ is a chart that allows any output behaviour over the additional outputs of chart A. That is, $in_{True} = in_C$ and $out_{True} = out_A \setminus out_C$.

In words, the first property demonstrates that there always exists a valid output refinement that increases the number of signals that the chart can use to control its environment.

The second property shows that, when $C \approx_{\mathcal{O}} A$, there exists a chart B that is observationally equivalent to both charts A and C (in the context of B's output interface). Recall that output refinement, *e.g.* $C \approx_{\mathcal{O}} A$, captures observational equivalence in a context that is concerned with just those signals that are common to both charts, *i.e.* $in_A \cap in_C$.

The last property shows that two charts C and A, related by output refinement where one interface is a subset of the other, can be considered as adding or removing completely nondeterministic behaviour over some signals.

We give an example where $C \approx_{\mathcal{O}} A$ and $[\![A]\!]^{\omega}_{x} = [\![C \mid \{\} \mid True]\!]^{\omega}_{x}$ in Figure 5.10.



(a) (b)

Figure 5.10: Output interface refinement and nondeterministic behaviour

This property demonstrates the notion of maintaining reactivity at the level of control information. Output refinement can only ever be used to remove a signal from the output interface of a chart if that signal encodes no extra information. That is, the chart without that signal conveys the same amount of information to its environment as the chart with the signal.

5.3.3 Discussion

The dichotomy of interface refinement demonstrates that there is a significant conceptual difference between input and output refinement of charts. Considering both of these refinements together as one larger class of refinement appears to make the definition of an intuitive notion of chart refinement difficult. An outcome of the investigation presented is a clearer understanding of the respective rôles of the input and output interfaces of a chart in terms of both specification and refinement. We discuss these in the following subsections.

Not surprisingly we can discuss the effect of changing the input and/or output interface of a chart by considering the context or environment into which the reactive system will be placed. The only pieces of information that a chart contains about its context are its input and output interfaces. Therefore changing the signals contained in one or other of these, which is interface refinement, is analogous to considering the meaning of the chart assuming a new context. This analogy allows a plausible description of why input refinement is quite different to output refinement.

The interaction between a chart and its context, input and output, can be equally described (from a chart's or chart designer's perspective) as events that are observable and events that are controllable respectively. Following this analogy input refinement can be considered as increasing the amount of information that a chart can observe of its context. Output refinement can be consider as increasing the events that the chart can control.

5.3.3.1 The Input Interface and Refinement

The syntax of μ-Charts (Section 2.1) allows both the implicit and explicit definition of an input interface for a sequential chart. When left implicit the input interface is assumed to contain all of the signals that appear in the triggers of the chart's transitions. We will refer to this implicit input interface as the *natural input interface* for a chart. This presents three situations that we need consider in order to describe the rôle of the input interface.

The case where explicit input interface is a subset of the natural input interface. The case where the input interface is the natural input interface. And finally where the interface contains additional signals to those in the natural interface.

The first situation is an example of input filtering; that is some of the signals that are used as trigger conditions on transitions will never actually be seen as input to the chart. The simplest way to describe the semantic effect of input signal filtering is to consider a semantically equivalent chart that can be derived by replacing all positive occurrences of a filtered signal with the trigger false and all negated occurrences with the trigger true. Therefore, as expected, a chart that implements input filtering is, in general, not semantically equivalent to the same chart with its natural input interface.

The second and third situations can be considered as forming a behavioural equivalence class. That is, the behaviour of a chart that has an explicit input interface with more signals than its natural input interface is equivalent (according to our definition of input refinement) to the behaviour of the chart with its natural interface. This was initially surprising because it is trivial to show that the following property holds.

Lemma 5.3.13 For arbitrary chart C, where $C_y =_{def} {}_{in_C \cup \{y\}}[C]$, for some signal $y \notin in_C$, we have,

$$[\![C]\!]^{\omega}_{x} \neq [\![C_y]\!]^{\omega}_{x}$$

This inequality arises because the two semantic relations $[\![C]\!]^{\omega}_{x}$ and $[\![C_y]\!]^{\omega}_{x}$ are not defined over the same domain and therefore comparing them semantically using equality is not meaningful. Input refinement however, compares the two charts over arbitrary input sequences (that is, sequences that contain, at least, signals from the union of the respective interfaces).

Hence, we consider two charts A and C equivalent if they are inter-refinable, for example $C \approx_{\mathcal{I}} A$. Moreover, we can argue that the ability to define an explicit input interface for charts is necessary only for signal filtering.

In terms of refinement, recall that input refinement of a deterministic chart produces another deterministic chart. By definition, a deterministic chart has determined behaviour based on what it can observe of its context. For example a chart that has a transition with the guard a means that this transition happens if the signal a is present in the input. To be deterministic, there must also be another transition (or other transitions) that describes the

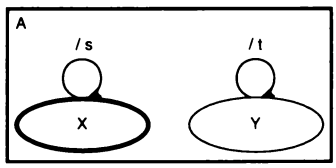
chart's action when signal a is not in the input. That signal a is either present or not present is equally determinable in a context that can only produce signal a as it is in a context that can produce signals a and b. Therefore it is reasonable that input refinement does not introduce new internal choice to a deterministic chart.

On the other hand, consider input refinement of a chart that does have internal choice. In this case, the extra information that the chart can observe of its context can be used to determine which of the previously nondeterministic transitions is taken. This refinement technique was dubbed angelic refinement earlier. Of course, if we make the choice between nondeterministic transitions before increasing the interface, that is weak input refinement, then the extra observable contextual information can have no bearing on that choice.

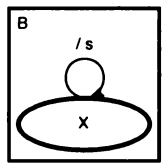
5.3.3.2 The Output Interface and Refinement

The output interface for a chart can also be stated explicitly. Again we use the term *natural output interface* to refer to an interface that contains just those signals mentioned on the actions of transitions in the chart.

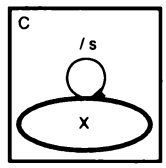
Like for the input interface, defining an explicit output interface that is a subset of the natural output interface can be considered as signal hiding. The case where the explicit output interface contains additional signals to the natural interface, however, is significantly different. In general a chart that has an expanded explicit output interface is not semantically equivalent to the same chart with the natural interface. Consider the following charts:



(a) $in_A = \{\}$, $out_A = \{s, t\}$



(b) $in_B = \{\}$, $out_B = \{s\}$



(c) $in_C = \{\}$, $out_C = \{s, t\}$

Figure 5.11: Natural and explicit output interfaces

For these particular charts we have that $A \approx_{\mathcal{O}} C$ (*i.e.*, A and C can be considered semantically equivalent), however, $B \not\approx_{\mathcal{O}} C$ (and consequently $B \not\approx_{\mathcal{O}} A$).

This demonstrates that the explicit output interface of a chart allows the designer to encode more specific behaviour for a chart than the default natural interface. In particular, the specification of chart C stipulates that the output context of C is one that can be controlled using signal t but the specified reactive system does not output t.

The original μ-Charts language [78] identifies that charts B and C should not be semantically equivalent (*i.e.*, inter-refinable). It is conjectured that the additional condition that restricted the original refinement theory to those refinements that strictly increase the interface of a chart, was introduced precisely to ensure this property holds. Like the original language, it holds of this refinement theory that $C \sqsupseteq_x^\omega B$[3] and $B \not\sqsupseteq_x^\omega C$. However, there is no explicit condition that requires refinement increase the interface of charts.

In the refinement theory presented here, refinement is a judgement made assuming a context that is controllable by (at least) the signals in the union of the respective output interfaces. That is, the charts A and C of our example above are compared assuming the environment is controlled by both signals s and t. To consider the behaviour of chart B in a context that is controlled by signals s and t requires that we find a semantically equivalent chart B' such that $B' \approx_{\mathcal{O}} B$ and $out_{B'} = \{s, t\}$. Given the presented refinement, this means that the chart B' has the additional internal choice of outputting signal t, or not, at each transition. That is, B' is a chart like B with the additional transition that outputs $\{s, t\}$. Chart C is then the behavioural refinement of this B'. Moreover, this chart B' describes the only observably equivalent behaviour to chart B in a context controllable by signals s and t. Because chart B' contains more internal choice (or contains more nondeterminism) than chart C we have that $B \not\sqsupseteq_x^\omega A$.

The separate investigation of input, output and behavioural refinement has identified three important properties of the refinement theory presented. The explicit definition of the input interface of a chart increases the expressibility of the language only in the case where the natural input interface is restricted. The explicit definition of the output interface, however, increases the expressibility of the language in both cases, *i.e.* extending or restricting the output interface. Provided we accept that semantic equality is defined by inter-refinability, and that the observational equivalences $\approx_{\mathcal{I}}$ and $\approx_{\mathcal{O}}$ are acceptable ways of considering the semantics of a chart when placed in a new context, then the (seemingly arbitrary) restriction of increasing inter-

[3]Note that the relation $\sqsupseteq_x^\omega$—the combination of each of the separate notions of refinement presented—is defined in Definition 5.4.1.

faces placed on the original refinement theory is not necessary. Similarly, the definition of the partial relations refinement theory must assume refinement is a judgement made in the broadest output context with the proviso that a chart can freely choose to output signals outside of its defined output interface.

As we see later (see Definition 6.6.2 in Section 6.6.1), it turns out that any refinement step that removes signals from a chart's output interface cannot be derived using a forward simulation relation alone.

5.4 Behaviour and interface refinement combined

While it is useful for the sake of investigation to split refinement into three categories, it is typical to use interface and behaviour refinement together. Therefore we give a definition and associated rules that unifies the two.

Definition 5.4.1 For arbitrary charts A and C we have,

$$C \sqsupseteq_x^\omega A \;=_{def}\; \forall i;\; o \bullet (i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \Rightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega$$

where $s_{\triangleright(X)}$ restricts the range of the sequence s (pointwise) to the signals in the set X.

From this definition we derive introduction and elimination rules.

Proposition 5.4.1 For arbitrary charts A and C, and infinite sequences i, o we have,

$$\frac{C \sqsupseteq_x^\omega A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega}{(i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega}\ (\sqsupseteq^-)$$

$$\frac{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \vdash (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega}{C \sqsupseteq_x^\omega A}\ (\sqsupseteq^+)$$

Aside from these two obvious rules we give additional rules that demonstrate that the refinement relation $\sqsupseteq_x^\omega$ of Definition 5.4.1 is sound and complete with respect to the three separate refinement relations $\sqsupseteq_b$, $\approx_{\mathcal{I}}$ and $\approx_{\mathcal{O}}$.

For presentation, these rules are split into four disjoint cases based on the relationship between the respective input and output interfaces of the charts.

Proposition 5.4.2 For arbitrary charts A and C we have,

Case 1: $in_A \subseteq in_C \wedge out_A \subseteq out_C$

$$\frac{C \sqsupseteq_x^\omega A \quad \begin{array}{l} C \sqsupseteq_b B', \\ B' \approx_{\mathcal{O}} B, B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq_{\uparrow\uparrow}^-) \qquad \frac{\begin{array}{l} \exists B;\ B' \bullet C \sqsupseteq_b B' \wedge \\ \qquad B' \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{I}} A \end{array}}{C \sqsupseteq_x^\omega A}\ (\sqsupseteq_{\uparrow\uparrow}^+)$$

Case 2: $in_A \subseteq in_C \wedge out_C \subset out_A$

$$\frac{C \sqsupseteq_x^\omega A \quad \begin{array}{l} C \approx_{\mathcal{O}} B', \\ B' \sqsupseteq_b B, B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq_{\uparrow\downarrow}^-) \qquad \frac{\begin{array}{l} \exists B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A \end{array}}{C \sqsupseteq_x^\omega A}\ (\sqsupseteq_{\uparrow\downarrow}^+)$$

Case 3: $in_C \subset in_A \wedge out_C \subset out_A$

$$\frac{C \sqsupseteq_x^\omega A \quad \begin{array}{l} C \approx_{\mathcal{O}} B', \\ B' \approx_{\mathcal{I}} B, B \sqsupseteq_b A \vdash P \end{array}}{P}\ (\sqsupseteq_{\downarrow\downarrow}^-) \qquad \frac{\begin{array}{l} \exists B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A \end{array}}{C \sqsupseteq_x^\omega A}\ (\sqsupseteq_{\downarrow\downarrow}^+)$$

Case 4: $in_C \subset in_A \wedge out_A \subseteq out_C$

$$\frac{C \sqsupseteq_x^\omega A \quad \begin{array}{l} C \approx_{\mathcal{I}} B', \\ B' \sqsupseteq_b B, B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\ (\sqsupseteq_{\downarrow\uparrow}^-) \qquad \frac{\begin{array}{l} \exists B;\ B' \bullet C \approx_{\mathcal{I}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{O}} A \end{array}}{C \sqsupseteq_x^\omega A}\ (\sqsupseteq_{\downarrow\uparrow}^+)$$

where we assume the usual conditions for B, B' and P.

Now we return to consider the reasons given by Scholz in [78] for only allowing, by definition, refinements that strictly increase the interface of a chart.

Firstly, that combining sequential charts using the chart operators always increases the interface of a chart is certainly something that the refinement notion cannot disallow. However, we do not consider this a motivation for unduly restricting the definition of refinement.

More importantly, that refinement is transitive is a critical property for any stepwise refinement theory. Lemma 5.4.3 (and its proof in Appendix B.9, page 244) shows that the refinement we define is indeed transitive.

Lemma 5.4.3 For arbitrary charts A, B and C,

$$\frac{C \sqsupseteq_x^\omega B \quad B \sqsupseteq_x^\omega A}{C \sqsupseteq_x^\omega A}$$

This concludes our investigation and description of μ-Charts refinement in terms of traces of behaviour. Next in Chapter 6 we show how this notion of refinement can be defined in terms of just the partial relations semantics for charts as given in Chapter 3

Chapter 6

Defining Refinement via Partial Relation Semantics

Now that we have defined exactly what refinement is in terms of the trace behaviour of a chart, we show how this refinement can be equivalently defined in terms of just the partial relations semantics of charts given in Chapter 3. Where the trace refinement described in the last chapter was defined as one notion that could be applied to each of the different trace interpretations, the partial relations refinement theory requires a separate logic (set of rules) for each of the trace interpretations. This is because, for trace refinement, the trace interpretation was assigned to a chart and then the single trace refinement notion applied to those traces. The partial relations refinement theory combines both of these tasks, that is, the refinements allowed by the particular set of rules determine the trace interpretation. The designer uses just the partial relation description of a chart and the appropriate set of refinement rules for the required trace interpretation.

In fact, the exercise of deriving these rules indicates that, for the language μ-Charts defined here, where composition is defined in terms of the partial relations of the composite parts, only two of the four trace semantics introduced are sensible interpretations. The total chaotic and firing conditions semantics for charts are sensible, the partial chaotic and do-nothing semantics are not. Therefore we derive refinement rules for the total chaotic and firing conditions semantics only. Section 6.3 outline why the partial chaotic and do-nothing semantics are not sensible interpretations for the μ-Charts presented.

Before giving the rules for chart refinement in terms of partial relations, we first describe how an existing partial relations refinement theory is related

to the trace semantics refinement. This gives us the required link to demonstrate that the rules we give for partial relations refinement are related to the different trace semantics models and the resulting trace refinement that we outline in Chapter 4. Rather than reconstructing this link from scratch, we use the link that exists between charts and Z and existing work that relates Z refinement to abstract data types (ADTs).

In [16, 88], a framework for considering Z specifications and Z refinement in terms of ADTs is introduced. The idea is to map a "standard" Z specification, *i.e.* state schema, initialisation schema and operation schemas, into a relational setting. Broadly a relational ADT is a tuple of the form (X, xi, xf, Ops) such that: X is a state space; xi is an initialisation relation; xf is a corresponding finalisation relation; and Ops is an indexed set of relational operations. The initialisation and finalisation relations map a global observable state into the ADTs private state and vice versa. A program of an ADT is defined as a particular sequence of the indexed operations upon a data type, preceded by initialisation and ended by finalisation. This mapping is used to derive a data refinement theory for Z specifications from the existing refinement notion for partial relations ADTs.

Given that the partial relation semantics for μ-Charts is defined via Z we can fit charts into the same framework. Recall from Section 3.7 that the translation of a chart into its "Z meaning" gives us a state space, an initialisation schema and one operation schema, the operation schema being the description of every step that the chart can take. So, if we view this Z description of a chart in the ADT framework we can say that any program allowed by the chart is an example of composing the step operation together with itself again and again. Of course, since we want to compare this with the trace semantics what we are really interested in is the sequences of inputs and outputs that result from such programs. If we imagine running this program over all possible input sequences and recording the resulting output sequences then we have exactly the trace semantics of the chart. In fact, because we have defined the trace semantics over infinite sequences of input and output we need to imagine composing the step operation with itself indefinitely.

In the following we show how the Z / ADT results generalise to charts. In particular we show that the ADT view of a chart can be considered as giving the trace semantics of that chart. Then we define the two notions of partial relations refinement for charts based on the existing notions of partial relation refinement for Z.

6.1 Relational ADTs with IO

In [16], Derrick and Boiten give a description of the ADT framework that allows operations to deal with inputs and outputs. The central idea is that there exists some global state G such that:

$$G = \text{seq}\, Input \times \text{seq}\, Output$$

A program of an ADT can then be considered as a relationship between two global states. The definition of refinement is given by comparing programs from two different ADTs. It is typical to talk about an abstract and a concrete ADT, where the assumption is that the concrete ADT is a refinement of the abstract.

Consider the following diagram that is common to most descriptions of ADT refinement.

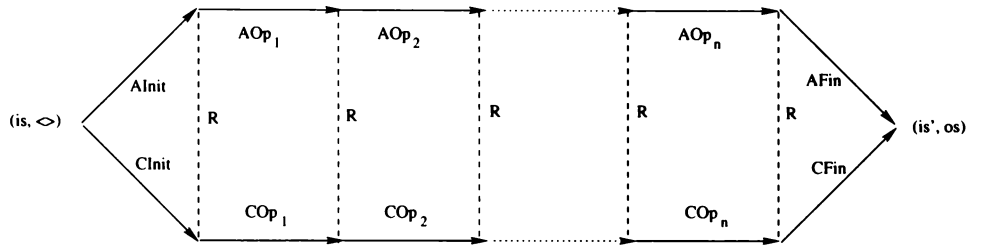


Figure 6.1: ADT Refinement Illustration

This diagram represents two ADTs, $A = (AState, AInit, \{AOp_i\}_{i \in \mathbb{N}}, AFin)$ and $C = (CState, CInit, \{COp_i\}_{i \in \mathbb{N}}, CFin)$ respectively.

We define the programs $P_a^n = AInit \fatsemi AOp_1 \fatsemi AOp_2 \fatsemi \ldots \fatsemi AOp_n \fatsemi AFin$ and $P_c^n = CInit \fatsemi COp_1 \fatsemi COp_2 \fatsemi \ldots \fatsemi COp_n \fatsemi CFin$ over the respective ADTs A and C. P_a^n and P_c^n are *conformal* programs because they have the same length and comparable operations. For example, it is assumed that the operation COp_i simulates the abstract operation AOp_i in the concrete ADT. The data type C is said to be a refinement of A when, for any pair of conformal programs, any observation that can be made of the concrete program is a possible observation of the abstract program. By observation we mean a pair of global states that consists of the global state we start the program in and the global state in which the program ends, in our case input and output sequences. In this input/output sequence model, assuming that C is a refinement of A, we have that any sequence of outputs os of length n that the program P_c^n can produce whilst consuming n elements of the input sequence is is also a sequence of outputs that the program P_a^n can produce

whilst consuming the same input sequence.

Now we represent the meaning of the programs P_c^n and P_a^n respectively as a relation between input sequences and output sequences of length n. Assuming the global input sequences range over some type *Input* (modelling the appropriate input context as described in Section 5.3) and similarly the output sequences over some type *Output*, then the meaning $[\![.]\!]_d^r$ of the program P^n is given as follows.[1]

$$[\![P^n]\!]_d^r \subseteq (\mathcal{I}^* \times \mathcal{O}^*)$$

With respect to the diagram of Figure 6.1 we have that $[\![P_a^n]\!]_d^r$ is a relation between the input sequence *is* and the resulting output sequence *os*. Notice that there is an implicit assumption in the original ADT framework which is that the sequence *is* has at least n elements. That is, applying the program P_a^n (assuming $n > 0$) from a state that represents an empty sequence of inputs is not well defined. Also, the output sequence *os* always has exactly n elements. We assume that the sequences in both the domain and range of the relation $[\![P^n]\!]_d^r$ have exactly length n. In the original ADT framework applying a program from a state that has more than n inputs takes us to the same state as applying the program from a state that has exactly the first n inputs only. Hence, that the relation $[\![P^n]\!]_d^r$ only considers input sequences of exactly length n captures all of the necessary information of the original framework. The final assumption that we make is that $[\![P^n]\!]_d^r$ is a total relation. That is, it is defined over all sequences of inputs of length n.

From the standard account of data refinement we have the following definition.

Definition 6.1.1 Given two abstract data types A and C as described above, for all pairs of conformal programs P_a^n and P_c^n of length n we have,

$$C \sqsupseteq_d A \ =_{def} \ [\![P_c^n]\!]_d^r \subseteq [\![P_a^n]\!]_d^r$$

In words, this definition states that the ADT A is a data refinement of the ADT C if and only if the observable behaviour of running any program in the concrete ADT is a subset of the behaviour observed when running the conformal program in the abstract ADT. This definition follows directly from the definition of data refinement where each different input sequence represents a different state from which the program is started.

[1] We generalise the sequences that we introduced in Section 4.6 such that $\mathcal{I}^*$ denotes the set of all finite sequences ranging over elements of the type *Input* and similarly $\mathcal{O}^*$ denotes all finite sequences ranging over *Output*. The infinite sequences $\mathcal{I}^\omega$ and $\mathcal{O}^\omega$ are similarly defined.

We have omitted an important detail so far, which is that the data refinement framework presented in both [88] and [16] assumes that the relational semantics of operations are total. Embedding Z-based ADTs into this data refinement framework therefore requires the definition of a totalisation scheme to make the partial relations semantics for Z schemas total. There are several well-known [16, 88] and well-investigated [18, 19, 53, 20, 17] schemes for totalising the partial relations. The two that we are interested in here are the *total chaotic* completion and the *firing conditions*[2] totalisation models as discussed in Chapter 4. Importantly, this means that we are using the d in the relation $\sqsupseteq_d$ of Definition 6.1.1 and in the operator $[\![.]\!]^r_d$ as a place holder for two different notions of refinement. In particular, the d can be replaced with τ-*chaos* to represent the total chaotic totalisation or *fc* to represent the firing conditions model of data refinement.

Here we chose to deal with infinite input and output sequences. Recall that the length of the output sequence is determined by the number of operations in the program, that is, output is defined by performing one of the program's operations on an input. Therefore, to consider infinite output sequences, we need also consider programs that contain an infinite number of operations. Fortunately, we only require programs that apply one of a finite set of operations infinitely many times. In fact, our case is even simpler in that there is only the one step operation of a chart to consider. Moreover, an assumption of the relational ADT framework is that each relational operation is total. These conditions mean that there is no unbounded nondeterminism present in the programs and that we do not need to consider nontermination of an operation. Henceforth, we will refer to such programs as infinite programs.

Now, given an arbitrary infinite program $P = Init \fatsemi Op_1 \fatsemi Op_2 \fatsemi ...$, such that for all i Op_i is an element of a finite set of operations, we have that the meaning $[\![.]\!]^{r\omega}_d$ of P is such that,

$$[\![P]\!]^{r\omega}_d \subseteq (\mathcal{I}^\omega \times \mathcal{O}^\omega)$$

where parameter d is used again as described above.

We introduce notation to represent the finite truncation of an infinite program. For the arbitrary infinite program $P = Init \fatsemi Op_1 \fatsemi Op_2 \fatsemi ...$ the expression $P \upharpoonright n = Init \fatsemi Op_1 \fatsemi Op_2 \fatsemi ... \fatsemi Op_n \fatsemi Fin$, that is, the program $P \upharpoonright n$ represents interrupting the infinite program P (*i.e.*, applying finalisation)

[2] In the work presented by Derrick and Boiten [16] the firing conditions semantic interpretation is referred to as the *behavioural interpretation.*

after the first n operations. Because of this finalisation we also have that $P \upharpoonright n = P^n$ for all infinite programs P and associated finite programs P^n.

Now we can define $[\![.]\!]^{r\omega}_d$ in terms of $[\![.]\!]^r_d$ as follows.

Definition 6.1.2 For the arbitrary infinite sequences i and o, and infinite program $P = Init \fatsemi Op_1 \fatsemi Op_2 \fatsemi ...$,

$$(i, o) \in [\![P]\!]^{r\omega}_d \ =_{def} \ \forall n \bullet (i \upharpoonright n, o \upharpoonright n) \in [\![P \upharpoonright n]\!]^r_d$$

where the truncation operator $s \upharpoonright n$ is defined as in Definition 4.6.2 of Section 4.6.

Given this definition for the relation $[\![P]\!]^{r\omega}_d$, we can show that the Proposition 6.1.1 holds for all infinite sequences of input and output, *i.e.* Definition 6.1.2 is monotonic with respect to subset.[3]

Proposition 6.1.1 For all conformal infinite programs $P_a = AInit \fatsemi AOp_1 \fatsemi AOp_2 \fatsemi ...$ and $P_c = CInit \fatsemi COp_1 \fatsemi COp_2 \fatsemi ...$ we have,

$$[\![P_c]\!]^{r\omega}_d \subseteq [\![P_a]\!]^{r\omega}_d \ \Leftrightarrow \ \forall n \bullet [\![P_c \upharpoonright n]\!]^r_d \subseteq [\![P_a \upharpoonright n]\!]^r_d$$

Proposition 6.1.2 follows trivially from Definition 6.1.1 and Proposition 6.1.1.

Proposition 6.1.2 Given two abstract data types A and C as described above, for all pairs of conformal infinite programs P_a and P_c we have,

$$C \sqsupseteq_d A \ \Leftrightarrow \ [\![P_c]\!]^{r\omega}_d \subseteq [\![P_a]\!]^{r\omega}_d$$

Given Proposition 6.1.2, if we can show a data refinement holds between an abstract and concrete ADT then we have shown that any observation of an infinite program over the concrete ADT must also be a possible observation of its conformal program in the abstract ADT.

6.2 Charts and ADTs

We show how the partial relations semantics for charts fits into the relational data type model.

The account of ADT refinement that we gave in the previous section made the simplifying assumption that the types of inputs and outputs associated with the two programs P^n_a and P^n_c are the same. For our purposes this simplifying assumption is too strict. Recall that the input and output interface of

[3]See Appendix B.10, page 245 for proof.

a chart can be changed via refinement. Weakening the assumption of equivalently typed input and output for both abstract and concrete programs is achieved using the respective initialisation and finalisation relations in conjunction with the notion of an observable context for charts. Recall that our exposition of input and output refinement in Section 5.3 identified that refinement is a judgement made in the broadest input/output context with the proviso that a chart can choose nondeterministically to output signals outside of its defined output interface.

We use the respective initialisation and finalisation relations to make the ADT's global state model the appropriate input/output context. The observable behaviour of the ADT (*i.e.*, the global state) is a context in which the abstract and concrete charts share the same inputs and outputs. The initialisation relation maps the global input sequences into appropriate input sequences for the respective charts. Similarly, the finalisation relation maps the outputs from the respective charts into the global output sequences.

From this we make the necessary link between the semantics for a chart C given by embedding the Z model of a chart in an ADT framework, that is, $[\![P_c]\!]_d^{r\omega}$ where $P_c = Init_C^{\perp} \fatsemi CSys^{\perp} \fatsemi CSys^{\perp} \fatsemi ...$,[4] and the trace semantics that we defined in Section 4.6 (page 85), *i.e.* $[\![C]\!]_x^{\omega}$.

Proposition 6.2.1 For arbitrary chart C and sequences $i \in \mathcal{I}^{\omega}$ and $o \in \mathcal{O}^{\omega}$,

$$(i, o) \in [\![P_c]\!]_d^{r\omega} \Leftrightarrow (i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_d^{\omega}$$

Note that we use the variable d in the operator $[\![.]\!]_d^{\omega}$ instead of the x that was introduced in chapters 4 and 5 (in particular in Definition 5.4.1, page 119) to ensure that both the left and right hand side of Proposition 6.2.1 are parameterised by the same semantic interpretation, *i.e.* the variable d binds properly. We rely on the careful construction of the framework presented and the demonstration of the different possible trace semantic interpretations of charts given in Chapter 4 as proof of proposition 6.2.1.

Now using Definition 5.4.1 and propositions 6.1.2 and 6.2.1 it is trivial to show that the trace refinement defined in Section 5.4 (page 119) is equivalent to data refinement between the appropriate ADTs for two charts.

Proposition 6.2.2 For arbitrary charts A and C,

$$C \sqsupseteq_d^{\omega} A \Leftrightarrow C \sqsupseteq_d A$$

[4]The notation $Init_C^{\perp}$ and $CSys^{\perp}$ is used to denote a suitable totalisation of the partial relations described by the schemas $Init_C$ and $CSys$, respectively. The appropriate totalisation is determined by the semantic model assumed for the charts as described in sections 4.3 and 4.4.

As demonstrated in Chapter 4, the choice of the ADT refinement notion (*i.e.*, depending on how the partial relations-based operations and simulations are made total) determines which of the *total chaotic* or *firing conditions* trace interpretations is assumed.

The remainder of this Chapter shows how we are able to follow the well-known relational ADT approach (for example see [16, 88]) to derive refinement rules for charts in terms of their partial relations semantics.

6.3 Implicit chaos *vs.* explicit permission to behave

It is well known that the special value $\perp$ is required for the standard embedding of a partial relations semantics in a relational ADT framework for refinement. This is also true for the μ-charts partial relations semantics. While the need for the special value $\perp$ is well documented in the existing literature, the exact rôle of this value is less clear.

In [88], and more recently [20], the use of $\perp$ to lift the partial relations semantics of Z is introduced and investigated.

Woodcock and Davies [88] illustrate that *lifted totalisation* (totalisation after adding $\perp$) ensures that undefinedness is propagated through relational composition. Conversely, *non-lifted totalisation* (totalisation without adding $\perp$) can lead to *non-strict recovery* from chaos. That is, if we consider relational composition in computing terms as applying one operation after another, then chaotic behaviour by one operation can be recovered by applying the next operation. This property is formalised in the following proposition.

Proposition 6.3.1 There exists a schema κ such that:

$$\overset{\bullet}{Chaos} \fatsemi \overset{\bullet}{\kappa} = \overset{\bullet}{Chaos}$$
$$\overset{\diamond}{Chaos} \fatsemi \overset{\diamond}{\kappa} = \overset{\diamond}{\kappa}$$

where $\overset{\diamond}{S}$ is the non-lifted totalisation of the relation S and $\overset{\bullet}{S}$ represents the lifted (with $\perp$) and totalised relation S.

From [88] it appears that this is the significant rôle of $\perp$ in giving a partial relations refinement theory. This rôle of $\perp$ is clearly significant when we consider refinement of conformal programs over respective ADTs. However, it is not made clear that the additional refinements allowed by $\perp$, *i.e.* those that allow chaos to persist, require that the concrete state space is extended

to include a state that essentially simulates $\perp$. That is, the generality of data refinement is required to extend the abstract domain of the ADT. We explain the notion of simulating $\perp$ in terms of charts in the following. Also, the Woodcock and Davies account [88] does not identify the other significant rôle played by $\perp$.

Deutsch, Henson and Reeves [20] show that introducing $\perp$ to a partial relations semantics allows a distinction to be made between implicit chaotic behaviour and the explicit permission to behave chaotically. Using the notation of [20], this can be formalised as:

Proposition 6.3.2

$$\overset{\bullet}{True} \neq \overset{\bullet}{Chaos}$$

where $True = [T \mid true]$, $Chaos = [T \mid false]$ and $\overset{\bullet}{S}$ represents the lifted (with $\perp$) and totalised relation S.

The investigation of [19] is concerned with the rôle of $\perp$ in defining a partial relations theory for *operation refinement* (*i.e.* a subset of data refinement where all simulations are identity relations). In operation refinement the essential rôle of $\perp$ is clearly not the propagation of undefinedness because refinements that realise this rôle of $\perp$ require the more general framework of data refinement. For operation refinement the essential rôle of $\perp$ is related to the preconditions of operations, in particular distinguishing implicit chaos from explicit permission to behave chaotically.

In terms of encoding chaotic behaviour in μ-Charts, the value $\perp$ also plays two rôles. The first is to allow chaotic behaviour to persist in general. That is, after seeing an input for which no specific behaviour is defined, a chart is free to behave in a chaotic manner indefinitely. This rôle can be seen clearly in Section 4.3 where the name $\perp$ is used for the additional state required to encode total chaotic behaviour in charts. This rôle of $\perp$ as a special "undefined" state corresponds closely to the justification for $\perp$ given by Woodcock and Davies [88] outlined above. In chart terms, the state $\perp$ models a state that is reached when no defined transition can be taken. This state can then be simulated by a new chart state during refinement. This new state allows the refined chart to have behaviour that represents a refinement of persistent chaos. In practice, undefined behaviour in a chart often represents exceptional circumstances. The state $\perp$ allows this exceptional or undefined behaviour to persist. For example, consider charts A and C of Figure 6.2.

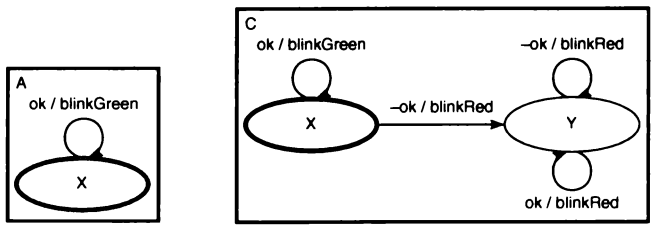


Figure 6.2: Refining Persistent Chaos

In this example we have that $C \sqsupseteq_{\tau f} A$ (that is, C refines A using a forward simulation). This refinement holds because the state Y simulates the "state" $\perp$.

A model without the state $\perp$ would allow only refinements that resolve this unexpected behaviour in one step. This would rule out the refinement demonstrated in the example in Figure 6.2. In particular, any refinement would be constrained to output *blinkGreen* whenever the input *ok* was present.

The partial chaotic trace semantics for charts is an example of a semantic interpretation that allows just *one-step* resolution of chaotic behaviour. This *one-step chaos* was identified by Scholz [78] when attempting to give a relational semantics to charts for the purpose of using model checking for verifying chart properties. Scholz did not consider using any mechanism such as $\perp$.

The second rôle of $\perp$ in charts is to distinguish between undefined behaviour and explicit permission to behave chaotically. This is a requirement of μ-Charts because of the definition of the language operators. The definition of composition is given in terms of the partial relations semantics of the composed charts. Chaotic (*i.e.* undefined) behaviour in one part of a composite chart causes chaotic behaviour in the composition itself. However, composition of defined behaviour (in particular explicit permission to behave chaotically) in one part of a composite chart does not force the composition to behave chaotically. Because the semantics differentiates between *Chaos* and *True* then the refinement notion for charts must also. Proposition 6.3.3 demonstrates that the composition operator dictates that *Chaos* and *True* are distinguishable.

Proposition 6.3.3 For the arbitrary chart C and signal set Ψ, we have

$$[\![Chaos \mid \Psi \mid C]\!]^{\omega}_{x} = [\![Chaos_C]\!]^{\omega}_{x}$$
$$[\![True \mid \Psi \mid C]\!]^{\omega}_{x} = [\![C]\!]^{\omega}_{x}$$

where $Chaos_C$ is similar to *Chaos* except that $in_{Chaos_C} = in_C$ and $out_{Chaos_C} =$

out_C.

Therefore, the mechanism $\perp$ ensures that the refinement notion for charts accords with the partial relations semantic definition. In particular, it ensures that *True* $\sqsupseteq_{\tau f}$ *Chaos* (*i.e. True* refines *Chaos*) but *Chaos* $\not\sqsupseteq_{\tau f}$ *True* (*Chaos* is **not** a refinement of *True*).

Of course, it is possible to give an alternative definition of the semantics of composition that does not require this distinction between *True* and *Chaos*. We discuss this possibility and the associated trade-offs in Section 7.2.

Now we consider this observation about the rôle of $\perp$ in terms of the four semantic interpretations for charts that we outlined in Chapter 4. For the total chaos and firing conditions semantics, implicit (*Chaos*) is distinguishable from explicit (*True*). That is by definition,

$$[\![Chaos]\!]^{\omega}_{\tau\text{-}chaos} \neq [\![True]\!]^{\omega}_{\tau\text{-}chaos} \quad \text{and,}$$
$$[\![Chaos]\!]^{\omega}_{fc} \neq [\![True]\!]^{\omega}_{fc}$$

Moreover, in both of these models the composition operator behaves sensibly. That is, the following lemma holds in general.

Lemma 6.3.4 For arbitrary charts A, B and C, and signal set Ψ we have,

$$[\![A]\!]^{\omega}_{\tau\text{-}chaos} = [\![B]\!]^{\omega}_{\tau\text{-}chaos} \Rightarrow [\![A \mid \Psi \mid C]\!]^{\omega}_{\tau\text{-}chaos} = [\![B \mid \Psi \mid C]\!]^{\omega}_{\tau\text{-}chaos}$$

However, for the partial chaotic trace semantics we have,

Definition 6.3.1

$$[\![Chaos]\!]^{\omega}_{\rho\text{-}chaos} = [\![True]\!]^{\omega}_{\rho\text{-}chaos}$$

Hence, it follows from our general argument that the composition operator for charts does not always behave sensibly. Under the do-nothing interpretation we have a similar situation because the following property holds.

Definition 6.3.2

$$[\![Chaos]\!]^{\omega}_{dn} = [\![DoNothing]\!]^{\omega}_{dn}$$

where the chart *DoNothing* explicitly encodes a chart that never outputs any signals.

An interesting observation is that, given the defined semantics of composition, the firing conditions semantics relates to the do-nothing semantics in the same way that the total chaotic semantics relates to the partial chaotic

semantics. The chart *Chaos* introduced above behaves like *Abort* under the firing conditions semantics. In particular, if the behaviour of one part of a composite chart "aborts" then the composed chart itself "aborts". In the same situation, the do-nothing semantics behaves like a "one-step" abortive process. That is, undefined behaviour (and therefore do-nothing behaviour) in one part of a composite chart causes the composed chart itself to do-nothing, but just for that particular input or step.

Two significant outcomes result from the investigation presented in this section. The first is the identification of the purpose of the (somewhat mystical) value $\perp$ in the definition of the μ-Charts language presented. That is, the value $\perp$ is firstly a technical device used so that implicit chaos is distinguishable from explicit permission to behave chaotically. This is necessary because the language operators encode an "intrusive" chaos due to their definition in terms of partial relations. Secondly, $\perp$ is required (in the more general data refinement framework) so that charts can model persistent chaotic behaviour.

We also note that the partial chaotic and do-nothing semantic interpretations are not useful under the given definition of charts. In particular, following the "usual" ADT embedding will *not* result in a refinement theory that is comparable to the respective trace semantics. Any such refinement theory will distinguish implicit (*True*) from explicit (*Chaos*) where these trace semantics do not. Therefore, attempting to derive such refinement theories for these two semantic interpretations is not useful.

6.4 Simulation and corresponding states

Before we derive the refinement rules for the two relevant trace refinement notions we briefly introduce and discuss the concept of simulation. When comparing two charts based on input and output traces, that is, checking for or calculating trace refinements, the state information of the charts is already abstracted away. This is not the case, however, when working with the partial relations semantics. We must have a way of comparing the states of one chart with another. This is exactly the task of simulations, sometimes also known as *retrieve relations*, *abstraction relations*, or *coupling invariants* [16]. Something as simple as changing the names of the states from the abstract chart to the concrete one requires that we have a simulation relation that maps the abstract state names into the new concrete state names.

Figure 6.1 of Section 6.1 (page 125) illustrates a simulation relation which

is labelled R. A simulation relation encodes the relationship between the states of the abstract specification and the states of the concrete specification. As this figure illustrates, the simulation R creates a series of commuting squares. This allows us to prove the necessary refinement properties for each of the associated operations (in our case there is only one) and use an inductive argument to show that the refinement holds when we compose (in an appropriate order) operations together into programs. Because the simulation R is a relation, it can be used in either direction. That is, we can assume that R relates the abstract states to the concrete states or vice versa. Except to say that we use the terms *forward simulation* and *backward simulation* respectively to represent this notional order, we refer the reader to [16] for a detailed description of the concepts of data type refinement.

According to the theme that has been common throughout this dissertation, there are two ways in which simulation relations are used or created. A simulation relation can be created in order to guide a refinement of one chart into another. This type of simulation is common in the ADT world where it is typical to consider data refinement as a way of refining some data representation into another, for example refining sets into lists.

Alternatively, a designer may have created two charts and wish to show that one refines the other. In this case, the importance of the simulation relation is its existence. That is, given the two existing charts, we can show that they are related by refinement by showing that a simulation relation exists between them.

In the first case the relation is designed for a specific transformation that the designer wishes to make to a specification. The new corresponding operations can often then be calculated to ensure that programs (in our case input/output traces) over the new state space have a refinement relationship to those over the existing state space.

In the case of pre-existing abstract and concrete charts it is typical to calculate the relation and then show that it is indeed a simulation. This again means the relation itself is less likely be illustrative of the design decisions that it captures. This type of use of simulation, however, is predominant in process algebra refinement theories, for example the use of simulation in IO Automata [52].

As discussed in Section 6.2, the initialisation and finalisation relations are used to modify the observable input and output sequences to allow for the interface refinement introduced in Section 5.3 (page 101). The simulation relation must also account for interface refinement. We split the definition

of the simulation relation into two separate parts. The first part is the simulation between states of the respective abstract and concrete charts. We will refer to this part of the simulation as the *corresponding relation* or $Corr_C^A$ for a simulation between charts A and C. The following Z schema gives the general scheme for the corresponding relation:

$$
\begin{array}{|l}
\hline
Corr_C^A \\
Chart_A \\
Chart'_C \\
\hline
P \\
\hline
\end{array}
$$

The predicate P defines the simulation relationship between the states of the respective charts A and C.

Note that we use the notational convention $Corr_A^C$ assuming:

$$Corr_A^C =_{def} [Chart_C;\ Chart'_A \mid P[\alpha Chart_A/\alpha Chart'_A, \alpha Chart'_C/\alpha Chart_C]]$$

The second part of the simulation relation accounts for interface refinement as described in Section 5.3. Unlike the previous part of the simulation relation, where the predicate P differs for each refinement application, the relationship between input and output can be given in general for all refinements. Using this general relationship in the framework presented guarantees that the allowable interface refinements accord with those described in Section 5.3.

For arbitrary input interfaces in_A and in_C, and output interfaces out_A and out_C, we have,

$$
\begin{array}{|l}
\hline
IO_C^A \\
i_A : in_A \\
i'_C : in_C \\
o_A : out_A \\
o'_C : out_C \\
\hline
i_A \cap in_C = i'_C \cap in_A \\
o_A \cap out_C = o'_C \cap out_A \\
\hline
\end{array}
$$

The schema IO is constructed so that $[\![IO]\!]_{Z_C}$ represents a relation from bindings of type V_A^{io} to bindings of type $V_C^{io'}$. Importantly, when this relation is combined with the corresponding relation we get a schema representing the simulation relation between charts A and C that has type $\mathbb{P}(T_A^{io} \curlyvee T_C^{io'})$.[5]

[5]The schema IO_A^C is also defined in the obvious way.

Definition 6.4.1 For charts A and C we have,

$$R =_{def} Corr_C^A \wedge IO_C^A$$

where $[\![R]\!]_{\mathcal{Z}_C}^{\mathbb{P}(T_A^{io} \curlyvee T_C^{io'})}$.

Significantly, when using the refinement theory presented the developer need only define the relationship between states of the "refining" and "refined" charts. The input/output relationship or interface refinement is always constrained by the general relationship identified by the schema IO.

Given this inherent structure that is present in all simulations required by the partial relations refinement calculus for charts we give the following useful introduction and elimination rules.

Proposition 6.4.1 Given arbitrary charts C_1, C_2, related simulation S, and bindings $z_1^{T_3}$, $z_2^{T_4}$, we have

$$\frac{z_1 \star z_2' \in S}{z_1.i_{C_1} \cap in_{C_2} = z_2.i_{C_2} \cap in_{C_1}} (S_{io1}^-) \qquad \frac{z_1 \star z_2' \in S}{z_1.o_{C_1} \cap out_{C_2} = z_2.o_{C_2} \cap out_{C_1}} (S_{io2}^-)$$

$$\frac{z_1 \star z_2' \in S}{z_1 \star z_2' \in IO_{C_2}^{C_1}} (S_{io3}^-) \qquad \frac{z_1 \star z_2' \in S}{z_1 \star z_2' \in Corr_{C_2}^{C_1}} (S_{io4}^-)$$

$$\frac{z_1 \star z_2' \in Corr_{C_2}^{C_1} \quad z_1 \star z_2' \in IO_{C_2}^{C_1}}{z_1 \star z_2' \in S} (S_{io}^+)$$

where $[\![C_2]\!]_{\mathcal{Z}_C}^{T_1}$, $[\![C_1]\!]_{\mathcal{Z}_C}^{T_2}$, $T_1 \preceq T_3$, $T_2 \preceq T_4$ and T_3 and T_4 are disjoint.

The related proofs are given in Appendix B.11 (page 247). Also, lemmas B.14.1 and B.14.2 give specialised rules for dealing with simulations between composed charts and charts that use the interface operator respectively.

6.5 Partial relation refinement

Finally, we derive partial relations refinement rules that capture the total chaotic and firing conditions trace semantic interpretations for charts. The derivation of the different sets of rules closely follows a similar treatment by Derrick and Boiten in [16].

We embed the Z-based chart ADT presented so far into a relational data type as follows:

Definition 6.5.1 For an arbitrary chart C and all sequences si and so, the Z ADT semantics $(Chart_C, Init_C, \{CSys\})$ is embedded in the relational data type $(CState, CInit, \{CStep\}, CFin)$, such that,

$$
\begin{aligned}
CState &=_{def} \mathcal{I}_C^{\omega} \times \mathcal{O}_C^{*} \times U_C \\
CInit &=_{def} \{(si \mapsto (si_{\triangleright(in_C)}, \langle\rangle, z)) \mid z \in Init_C\} \\
CFin &=_{def} \{(si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z) \mapsto so \mid z \in Chart_C\} \\
CStep &=_{def} \{(i^\frown si, so, z_1) \mapsto (si, so^\frown o, z_2) \mid \\
&\qquad\qquad z_1 \star (\!| \, i_C \Rrightarrow i, o'_C \Rrightarrow o \, |\!) \star z'_2 \in C\}
\end{aligned}
$$

The embedding of the simulation R gives the simulation relation S between the ADTs representing charts A and C.

For arbitrary sequences si and so, and bindings $z_1^{U_C}$ and $z_2^{U_A}$ we have,

$$S =_{def} \{(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \mid z_1 \star z'_2 \in Corr_C^A\}$$

Note, the sequence so may represent the empty sequence when S is used to relate the abstract ADT to the concrete ADT directly after initialisation. The sequence si is an infinite sequence and therefore cannot be empty.

We show how the simulation relation S relates to the simulation R in propositions 6.5.1, 6.5.1 and 6.5.2.

Proposition 6.5.1 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_1^{T_A^{in}}$ and $x_2^{T_C^{in}}$ we have,

$$
\begin{aligned}
&(\langle x_1.i_A\rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (\langle x_2.i_C\rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S \\
&\quad \Leftrightarrow \quad \text{[See Appendix B.12 (page 248) for details]} \\
&\forall x_3 \bullet \exists x_4 \bullet z_1 \star x_1 \star x_3 \star z'_2 \star x'_2 \star x'_4 \in R
\end{aligned}
$$

Proposition 6.5.2 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_3^{T_A^{out}}$ and $x_4^{T_C^{out}}$ we have,

$$
\begin{aligned}
&(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}{}^\frown\langle x_3.o_A\rangle, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}{}^\frown\langle x_4.o_C\rangle, z_2) \in S \\
&\quad \Leftrightarrow \quad \text{[See Appendix B.12 for details]} \\
&\forall x_2 \bullet \exists x_1 \bullet z_1 \star x_1 \star x_3 \star z'_2 \star x'_2 \star x'_4 \in R
\end{aligned}
$$

Proposition 6.5.3 For the arbitrary sequence si, and bindings $z_1^{U_C}$ and $z_2^{U_A}$ we have,

$$
\begin{aligned}
&(si_{\triangleright(in_A)}, \langle\rangle, z_1) \mapsto (si_{\triangleright(in_C)}, \langle\rangle, z_2) \in S \\
&\quad \Leftrightarrow \quad \text{[See Appendix B.12 for details]} \\
&z_1 \star z'_2 \in Corr_C^A
\end{aligned}
$$

6.6 Total chaos refinement

6.6.1 Forward simulation

We start by deriving the rules for the total chaotic interpretation. Derrick and Boiten [16] give five refinement conditions that are necessary to show that a relational data type C refines a relational data type A using a forward simulation S. They begin by lifting and totalising the relations of the respective data types. As expected, this lifting and totalising is the same as the completion of the partial relations semantics that we outlined in Chapter 5. Derrick and Boiten refer to the total chaotic interpretation as the *contract approach.* After giving the necessary lifted totalised relations they show how the five refinement conditions, referred to as *initialisation*, *finalisation*, *finalisation applicability*, *applicability* and *correctness*, can be simplified ("relaxed") to remove any reference to the introduce value $\bot$. We give the five relaxed conditions and refer to [88] for their derivations.

Definition 6.6.1 Assuming data types $A = (AState, AInit, \{AStep\}, AFin)$ and $C = (CState, CInit, \{CStep\}, CFin)$, a forward simulation S is a relation from $AState$ to $CState$ satisfying the following conditions:

$$CInit \subseteq AInit \fatsemi S \qquad \text{(initialisation)}$$

$$S \fatsemi CFin \subseteq AFin \qquad \text{(finalisation)}$$

$$\text{ran}((\text{dom } AFin) \lhd S) \subseteq \text{dom } CFin \qquad \text{(finalisation applicability)}$$

$$\text{ran}((\text{dom } AStep) \lhd S) \subseteq \text{dom } CStep \qquad \text{(applicability)}$$

$$((\text{dom } AStep) \lhd S) \fatsemi CStep \subseteq AStep \fatsemi S \qquad \text{(correctness)}$$

Now we use each of these conditions along with the relational embedding defined in Definition 6.5.1 (page 138) to derive corresponding conditions expressed in Z.

Proposition 6.6.1 gives the Z condition related to *initialisation*:

Proposition 6.6.1

$$\begin{aligned} & CInit \subseteq AInit \fatsemi S \\ & \quad \Leftrightarrow \quad \text{[See Appendix B.13 for details]} \\ & \forall y_c \bullet y_c \dot{\in} Init_C \Rightarrow \exists t_1 \bullet t_1 \dot{\in} Init_A \land t_1 \star y_c' \in R \end{aligned}$$

Unlike the derivation provided by Derrick and Boiten [16], the finalisation condition does not hold trivially for charts. This difference arises because the derivation for Z refinement makes the assumption that both the abstract

and concrete ADTs have equivalently typed input and output, whereas the derivations required here do not. In Section 5.3.3.2 (page 117) we discussed in detail the subtleties of the definition of the output interface of a chart. Also, recall that the refinement for charts is a judgement made in the broadest input/output context (*i.e.*, an environment that can be controlled by at least the signals from the output interfaces of both the abstract and concrete specifications). Here the finalisation relation is used to map the behaviour of each chart into this so-called "broad" context. This ensures that the derived refinement notion accords with the trace refinement outlined in Section 5.

Now, if the output interface out_A of the abstract chart A is *not* a subset of the output interface out_C of the chart C, then the finalisation condition fails. Take for example the case where $out_C \subset out_A$: the simulation maps the larger output interface of A into the smaller interface of C; the concrete finalisation maps this smaller interface of C back into the "broad" interface (in this case the interface of A); the composition of the simulation and the concrete finalisation results in a non-functional relation, whereas the abstract finalisation is the identity relation. That is, the case where out_C is a subset of out_A is a counter-example that demonstrates why the *finalisation* condition does not hold trivially for μ-Charts.

Proposition 6.6.2

$$
\begin{array}{l}
S \fatsemi CFin \subseteq AFin \\
\quad \Leftrightarrow \quad [\text{See Appendix B.13 for details}] \\
out_A \subseteq out_C
\end{array}
$$

Because the given finalisation relation is total over all output sequences and states of the respective charts, the *finalisation applicability* condition holds.

Proposition 6.6.3

$$
\begin{array}{l}
\mathrm{ran}(\mathrm{dom}\, AFin \lhd S) \subseteq \mathrm{dom}\, CFin \\
\quad \Leftrightarrow \quad [AFin \text{ is total: } \mathrm{dom}\, S \subseteq \mathrm{dom}\, AFin] \\
\mathrm{ran}\, S \subseteq \mathrm{dom}\, CFin \\
\quad \Leftrightarrow \quad [CFin \text{ is total over the target set of } S] \\
true
\end{array}
$$

Unlike the derivations given in [16], we can show that *Pre* and "dom" do coincide in the relational embedding given here. This is because, in this

embedding, the relational operation *CStep* relates infinite input sequences to finite output sequences. Hence it is never the case that $(\langle\rangle, os, z) \in \text{dom } CStep$. In other words, there is always an input available in the head of the input sequence. That an empty input sequence is a possibility in the derivation of the Z refinement rules (from the assumption that programs and hence their respective input and output sequences are finite) is the reason that *Pre* and dom do not coincide in the derivation given in [16].

For an arbitrary chart C we have:

Proposition 6.6.4

$$
\begin{array}{l}
(x_{ic}.i_C \frown si_{\triangleright(in_C)}, so, z) \in \text{dom } CStep \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 for details]} \\
Pre\ C\ (z \star x_{ic} \star x_{oc})
\end{array}
$$

Now for the *applicability* condition we have:

Proposition 6.6.5

$$
\begin{array}{l}
\text{ran}((\text{dom } AStep) \lhd S) \subseteq \text{dom } CStep \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 for details]} \\
\forall\, y_a, y_c \bullet Pre\ ASys\ y_a \wedge y_a \star y_c' \in R \Rightarrow Pre\ CSys\ y_c
\end{array}
$$

And finally, for correctness we have:

Proposition 6.6.6

$$
\begin{array}{l}
((\text{dom } AStep) \lhd S) \fatsemi CStep \subseteq AStep \fatsemi S \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 for details]} \\
\forall\, y_a, y_c, \vdash_{Z_C} \bullet (Pre\ A\ y_a \wedge y_a \star y_c' \in R \wedge y_c \star \vdash_{Z_C}{}' \dot{\in} C) \Rightarrow \\
\qquad\qquad \exists\, t \bullet y_a \star t' \dot{\in} A \wedge t \star \vdash_{Z_C}{}' \in R
\end{array}
$$

The derivations of propositions 6.6.1, 6.6.2, 6.6.5 and 6.6.6 give us the Z conditions necessary to show that a relation R is a forward simulation between two charts A and C under the total chaotic interpretation of the partial relations semantics. As we have shown (see Section 4.3 and Proposition 6.2), it follows that chart C refines A in the total chaotic trace interpretation for charts. In line with the natural deduction style presentation that we have adopted, Definition 6.6.2 gives introduction and elimination rules for forward simulation total chaotic refinement:

Definition 6.6.2 For arbitrary charts A and C, and bindings $y_a^{T_A^w}$, $y_c^{T_C^w}$, and $\vdash_{Z_C}{}^{T_C^w}$, we have,

$$\frac{\begin{array}{ll} & \vdash out_A \subseteq out_C \\ y_c \dot{\in} Init_C & \vdash a_1 \dot{\in} Init_A \\ y_c \dot{\in} Init_C & \vdash a_1 \star y_c' \in R \\ Pre\ A\ y_a,\ y_a \star y_c' \in R & \vdash Pre\ C\ y_c \\ Pre\ A\ y_a,\ y_a \star y_c' \in R,\ y_c \star \vdash_{Z_C}{}' \dot{\in} C & \vdash y_a \star b_1' \dot{\in} A \\ Pre\ A\ y_a,\ y_a \star y_c' \in R,\ y_c \star \vdash_{Z_C}{}' \dot{\in} C & \vdash b_1 \star \vdash_{Z_C}{}' \in R \end{array}}{C \sqsupseteq_{\tau f} A}\ (\sqsupseteq^+_{\tau f})$$

$$\frac{C \sqsupseteq_{\tau f} A}{out_A \subseteq out_C}\ (\sqsupseteq^-_{\tau f\ I}) \qquad \frac{C \sqsupseteq_{\tau f} A \quad y_c \dot{\in} Init_C \quad a_1 \dot{\in} Init_A, a_1 \star y_c' \in R \vdash P}{P}\ (\sqsupseteq^-_{\tau f\ II})$$

$$\frac{C \sqsupseteq_{\tau f} A \quad Pre\ A\ y_a \quad y_a \star y_c' \in R}{Pre\ C\ y_c}\ (\sqsupseteq^-_{\tau f\ III})$$

$$\frac{C \sqsupseteq_{\tau f} A \quad Pre\ A\ y_a \quad y_a \star y_c' \in R \quad y_c \star \vdash_{Z_C}{}' \dot{\in} C \quad \begin{array}{l} y_a \star b_1' \dot{\in} A, \\ b_1 \star \vdash_{Z_C} \in R \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\tau f\ IV})$$

where the usual conditions hold for a_1, b_1 and P.

Notice, the rules for forward simulation refinement presented here are (with the exception of the additional initialisation and finalisation conditions) very similar to the rules presented by Deutsch and Henson in [19] for *SF-refinement.*

6.6.2 Backward simulation

For backward simulation we introduce a simulation R^{-1} which is defined in the same way as the simulation R, introduced in Definition 6.4.1 (*i.e.*, using schemas $Corr_A^C$ and IO_A^C). R^{-1} is defined in the obvious way such that the following proposition holds:

Proposition 6.6.7 For arbitrary bindings z_1 and z_2 we have,

$$z_1 \star z_2' \dot{\in} R^{-1} \Leftrightarrow z_2 \star z_1' \dot{\in} R$$

where $[\![R]\!]_{Z_C}^{T_A^w \curlyvee T_C^{w'}}$ and $[\![R^{-1}]\!]_{Z_C}^{T_C^w \curlyvee T_A^{w'}}$.

Now the embedding of the simulation R^{-1} in the relational ADT gives the backward simulation relation $T \subseteq CState \times AState$.

$$T =_{def} \{(si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \mid z_1 \star z_2' \in Corr_A^C\}$$

The relationship between T and R^{-1} is described by Propositions 6.6.8 and 6.6.9. These propositions correspond closely to those of Propositions 6.5.1 and 6.5.2 for forward simulations.

Proposition 6.6.8 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_1^{T_C^{in}}$ and $x_2^{T_A^{in}}$ we have,

$$(\langle x_1.i_C\rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (\langle x_2.i_A\rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in T$$

$$\Leftrightarrow \quad \text{[See Appendix B.13 (page 255) for details]}$$

$$\forall x_3 \bullet \exists x_4 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R^{-1}$$

Proposition 6.6.9 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_3^{T_C^{out}}$ and $x_4^{T_A^{out}}$ we have,

$$(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}{}^\frown\langle x_3.o_A\rangle, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}{}^\frown\langle x_4.o_C\rangle, z_2) \in T$$

$$\Leftrightarrow \quad \text{[See Appendix B.13 (page 256) for details]}$$

$$\forall x_2 \bullet \exists x_1 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R^{-1}$$

The five conditions necessary to show that a relation is a backward simulation, again taken from [16], are as follows.

Definition 6.6.3 Assuming data types $A = (AState, AInit, \{AStep\}, AFin)$ and $C = (CState, CInit, \{CStep\}, CFin)$, a backward simulation S is a relation from $CState$ to $AState$ satisfying the following conditions:

$$CInit \fatsemi T \subseteq AInit \qquad \text{(initialisation)}$$

$$CFin \subseteq T \fatsemi AFin \qquad \text{(finalisation)}$$

$$\forall c \bullet c(\!|\ T\ |\!) \subseteq \operatorname{dom} AFin \Rightarrow c \in \operatorname{dom} CFin \qquad \text{(finalisation applicability)}$$

$$\overline{\operatorname{dom} CStep} \subseteq \operatorname{dom}(T \rhd \operatorname{dom} AStep) \qquad \text{(applicability)}$$

$$\operatorname{dom}(T \rhd \operatorname{dom} AStep) \lhd CStep \fatsemi T \subseteq T \fatsemi AStep \qquad \text{(correctness)}$$

From these conditions we derive the necessary rules in terms of the partial relations semantics for charts.

For *initialisation* we have:

Proposition 6.6.10 For arbitrary charts A and C we have,

$$CInit \fatsemi T \subseteq AInit$$

$$\Leftrightarrow \quad \text{[See Appendix B.13 (page 257) for details]}$$

$$in_A \subseteq in_C \wedge \forall y, z \bullet (y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1}) \Rightarrow z \mathrel{\dot\in} Init_A$$

Again the *finalisation* condition for backward simulation refinement does not hold trivially. As expected, the derivation that follows shows that, in order for a relation to be a backward simulation between two charts, it must be total over the states of the concrete chart.

Proposition 6.6.11 For arbitrary charts A and C we have,

$$
\begin{array}{l}
CFin \subseteq T \fatsemi AFin \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 for details]} \\
\forall\, y_c \bullet \exists\, t_1 \bullet y_c \star t_1' \in R^{-1}
\end{array}
$$

Finalisation applicability, on the other hand, holds trivially because the relation $CFin$ is total.

The following lemma is used in the respective derivations for both *applicability* and *correctness*.

Lemma 6.6.12 For arbitrary sequences si and so, and bindings $x_{ic}^{T_C^{in}}$, $z_1^{U_C}$, we have,

$$
\begin{array}{l}
(x_{ic}.i_C \frown si, so, z_1) \notin \mathrm{dom}(T \rhd \mathrm{dom}\, AStep) \\
\quad \Leftrightarrow \quad \text{[See Section B.13 (page 260) for proof]} \\
\forall\, x_{im}, z_2, x_{om}, x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_2 \star x_{im} \star x_{om})
\end{array}
$$

Now for *applicability* we have:

Proposition 6.6.13 For arbitrary charts A and C we have,

$$
\begin{array}{l}
\overline{\mathrm{dom}\, CStep} \subseteq \mathrm{dom}(T \rhd \mathrm{dom}\, AStep) \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 (page 261) for details]} \\
\forall\, y_c \bullet (\forall\, v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a) \Rightarrow Pre\ C\ y_c
\end{array}
$$

And finally for *correctness*:

Proposition 6.6.14 For arbitrary charts A and C we have,

$$
\begin{array}{l}
\mathrm{dom}(T \rhd \mathrm{dom}\, AStep) \lhd CStep \fatsemi T \subseteq T \fatsemi AStep \\
\quad \Leftrightarrow \quad \text{[See Appendix B.13 (page 262) for details]} \\
\forall\, y_c, \vdash_{z_C}, y_a \bullet ((\forall\, v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a) \wedge \\
\qquad\qquad y_c \star \vdash_{z_C}{}' \dot{\in} C \wedge \vdash_{z_C} \star\, y_a' \in R^{-1}) \Rightarrow \exists\, t_2 \bullet y_c \star t_2' \in R^{-1} \wedge t_2 \star y_a' \dot{\in} A
\end{array}
$$

Following from these derivations, Definition 6.6.4 gives the introduction and elimination rules for backward simulation refinement:

Definition 6.6.4 For arbitrary charts A and C, and arbitrary bindings $y_c^{T_C^{io}}$, $y_a^{T_A^{io}}$, $v_a^{T_A^{io}}$, and $\vdash_{z_C}{}^{T_C^{io}}$, we have,

$$\frac{\begin{array}{ll} & \vdash in_A \subseteq in_C \\ & \vdash y_c \star a_1' \in R^{-1} \\ y_c \dot{\in} Init_C,\ y_c \star y_a' \in R^{-1} & \vdash y_a \dot{\in} Init_A \\ y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a & \vdash Pre\ C\ y_c \\ y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a,\ y_c \star \vdash_{z_C}{}' \dot{\in} C,\ \vdash_{z_C} \star\, y_a' \in R^{-1} & \vdash y_c \star b_1' \in R^{-1} \\ y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a,\ y_c \star \vdash_{z_C}{}' \dot{\in} C,\ \vdash_{z_C} \star\, y_a' \in R^{-1} & \vdash b_1 \star y_a' \dot{\in} A \end{array}}{C \sqsupseteq_{\tau b} A}\ (\sqsupseteq^{+}_{\tau b})$$

$$\frac{C \sqsupseteq_{\tau b} A \quad y_c \star a_1' \in R^{-1} \vdash P}{P}\ (\sqsupseteq^{-}_{\tau b\ I}) \qquad \frac{C \sqsupseteq_{\tau b} A}{in_A \subseteq in_C}\ (\sqsupseteq^{-}_{\tau b\ II})$$

$$\frac{C \sqsupseteq_{\tau b} A \quad y_c \dot{\in} Init_C \quad y_c \star y_a' \in R^{-1}}{y_a \dot{\in} Init_A}\ (\sqsupseteq^{-}_{\tau b\ III})$$

$$\frac{C \sqsupseteq_{\tau b} A \quad v_c \star v_a' \in R^{-1} \vdash Pre\ A\ v_a}{Pre\ C\ v_c}\ (\sqsupseteq^{-}_{\tau b\ IV})$$

$$\frac{C \sqsupseteq_{\tau b} A \quad \begin{array}{l} y_c \star v_a' \in R^{-1} \vdash \\ Pre\ A\ v_a \end{array} \quad y_c \star \vdash_{z_C}{}' \dot{\in} C\ \vdash_{z_C} \star\, y_a' \in R^{-1} \quad \begin{array}{l} y_c \star b_1' \in R^{-1}, \\ b_1 \star y_a' \dot{\in} A \vdash P \end{array}}{P}\ (\sqsupseteq^{-}_{\tau b\ IV})$$

where the usual conditions hold for a_1, b_1 and P.

Again, these rules are very similar to the introduction and elimination rules presented by Deutsch and Henson [19] for *SB-refinement*.

6.7 Firing conditions refinement

6.7.1 Forward simulation

To derive appropriate refinement rules for the firing condition semantic interpretation, we need only strengthen the *correctness* refinement condition. Again following the work of Derrick and Boiten [16] we strengthen the *correctness* condition of Definition 6.6.1 to the following.

$$S \fatsemi CStep \subseteq AStep \fatsemi S$$

Using a similar derivation to that for the *correctness* condition for forward simulation refinement under the total chaotic interpretation we have,

$$S \fatsemi CStep \subseteq AStep \fatsemi S$$

$$\Leftrightarrow$$

$$\forall y_a, y_c, \vdash_{z_C} \bullet (y_a \star y_c' \in R \wedge y_c \star \vdash_{z_C}{}' \dot{\in} C) \Rightarrow \exists t_2 \bullet y_a \star t_2' \dot{\in} A \wedge t_2 \star \vdash_{z_C}{}' \in R$$

Now we define the introduction and elimination rules for forward simulation refinement assuming a firing conditions semantic interpretation.

Definition 6.7.1 For arbitrary charts A and C, and arbitrary bindings $y_c^{T^{\wp}_C}$, $y_a^{T^{\wp}_A}$, $v_a^{T^{\wp}_A}$, and $\vdash_{z_C}{}^{T^{\wp}_C}$, we have,

$$\frac{\begin{array}{ll} & \vdash out_A \subseteq out_C \\ y_c \dot{\in} Init_C & \vdash a_1 \dot{\in} Init_A \\ y_c \dot{\in} Init_C & \vdash a_1 * y_c' \in R \\ Pre\ A\ y_a,\ y_a \star y_c' \in R & \vdash Pre\ C\ y_c \\ y_a \star y_c' \in R,\ y_c \star \vdash_{z_C}{}' \dot{\in} C & \vdash y_a \star b_1' \dot{\in} A \\ y_a \star y_c' \in R,\ y_c \star \vdash_{z_C}{}' \dot{\in} C & \vdash b_1 \star \vdash_{z_C}{}' \in R \end{array}}{C \sqsupseteq_{fcf} A}\ (\sqsupseteq^{+}_{fcf})$$

$$\frac{C \sqsupseteq_{fcf} A}{out_A \subseteq out_C}\ (\sqsupseteq^{-}_{fcf\ I}) \qquad \frac{C \sqsupseteq_{fcf} A \quad y_c \dot{\in} Init_C \quad a_1 \dot{\in} Init_A, a_1 \star y_c' \in R \vdash P}{P}\ (\sqsupseteq^{-}_{fcf\ II})$$

$$\frac{C \sqsupseteq_{fcf} A \quad Pre\ A\ y_a \quad y_a \star y_c' \in R}{Pre\ C\ y_c}\ (\sqsupseteq^{-}_{fcf\ III})$$

$$\frac{C \sqsupseteq_{fcf} A \quad y_a \star y_c' \in R \quad y_c \star \vdash_{z_C}{}' \dot{\in} C \quad \begin{array}{l} y_a \star b_1' \dot{\in} A, \\ b_1 \star \vdash_{z_C}{}' \in R \vdash P \end{array}}{P}\ (\sqsupseteq^{-}_{fcf\ IV})$$

where the usual conditions hold for a_1, b_1 and P.

6.7.2 Backward simulation

To derive the appropriate rules for backward simulation refinement assuming a firing conditions semantics we need again only strengthen the associated *correctness* condition of Definition 6.6.3. That is, for correctness we need to show the following holds.

$$CStep \fatsemi T \subseteq T \fatsemi AStep$$

From the obvious derivation we now have,

$$CStep \fatsemi T \subseteq T \fatsemi AStep$$

$$\Leftrightarrow$$

$$\forall y_c, y_a, \vdash_{z_c} \bullet\ y_c \star \vdash_{z_c}{}' \dot{\in} C \wedge \vdash_{z_c} \star\, y_a' \in R^{-1} \Rightarrow \exists\, t_2 \bullet\ y_c \star t_2' \in R \wedge t_2 \star y_a' \dot{\in} A$$

The introduction and elimination rules for backward simulation refinement assuming a firing conditions semantic interpretation are defined as follows.

Definition 6.7.2 For arbitrary charts A and C, and arbitrary bindings $y_c^{T_C^{vo}}$, $y_a^{T_A^{vo}}$, $v_a^{T_A^{vo}}$, and $\vdash_{z_c}{}^{T_C^{vo}}$, we have,

$$\frac{\begin{array}{ll} & \vdash in_A \subseteq in_C \\ & \vdash y_c \star a_1' \in R^{-1} \\ y_c \dot{\in} Init_C,\ y_c \star y_a' \in R^{-1} & \vdash y_a \dot{\in} Init_A \\ y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a & \vdash Pre\ C\ y_c \\ y_c \star \vdash_{z_c}{}' \dot{\in} C,\ \vdash_{z_c} \star\, y_a' \in R^{-1} & \vdash y_c \star b_1' \in R^{-1} \\ y_c \star \vdash_{z_c}{}' \dot{\in} C,\ \vdash_{z_c} \star\, y_a' \in R^{-1} & \vdash b_1 \star y_a' \dot{\in} A \end{array}}{C \sqsupseteq_{fcb} A}\ (\sqsupseteq^{+}_{fcb})$$

$$\frac{C \sqsupseteq_{fcb} A \quad y_c \star a_1' \in R^{-1} \vdash P}{P}\ (\sqsupseteq^{-}_{fcb\,I}) \qquad \frac{C \sqsupseteq_{fcb} A}{in_A \subseteq in_C}\ (\sqsupseteq^{-}_{fcb\,II})$$

$$\frac{C \sqsupseteq_{fcb} A \quad y_c \dot{\in} Init_C \quad y_c \star y_a' \in R^{-1}}{y_a \dot{\in} Init_A}\ (\sqsupseteq^{-}_{fcb\,III})$$

$$\frac{C \sqsupseteq_{fcb} A \quad y_c \star v_a' \in R^{-1} \vdash Pre\ A\ v_a}{Pre\ C\ y_c}\ (\sqsupseteq^{-}_{fcb\,IV})$$

$$\frac{C \sqsupseteq_{fcb} A \quad y_c \star \vdash_{z_c}{}' \dot{\in} C \quad \vdash_{z_c} \star\, y_a' \in R^{-1} \quad \begin{array}{l} y_c \star b_1' \in R^{-1}, \\ b_1 \star y_a' \dot{\in} A \vdash P \end{array}}{P}\ (\sqsupseteq^{-}_{fcb\,V})$$

where the usual conditions hold for a_1, b_1 and P.

Notice that, unlike the trace semantics for charts, we do not have to add any extra machinery, for example the special output signal $\perp$, to the partial relations semantics to encode the abortive behaviour of charts.

The rules for both forward and backward simulation refinement assuming a firing conditions semantic interpretation correspond respectively to the *SPF-refinement* and *SPB-refinement* of Deutsch presented in [17].

In the following section we demonstrate how the refinement rules for μ-Charts, described here, can be used to reason about properties of the language itself. In particular, we use the rules to derive the necessary side-conditions that guarantee that a refinement of a chart is monotonic with respect to composition.

Chapter 7

Monotonicity

As with any language that provides operators allowing modular specifications and a refinement calculus for step-wise development, the monotonicity properties of the μ-Charts operators needs to be considered. These monotonicity properties are important for μ-Charts so that the language supports modular development that compliments the modular specification of reactive systems. Refinement is considered monotonic with respect to a language operator if a refinement of one part of a composite specification implies a refinement of the specification as a whole.

Both Deutsch, Henson, and Reeves [21] and Groves [32] show that in general the "usual" Z schema calculus has poor monotonicity properties. Recall that the semantics of μ-Charts is defined using the Z schema calculus and the refinement calculus for μ-Charts is derived using two of the standard notions of refinement for Z. Therefore, we expect that the refinement notion for μ-Charts will also have poor monotonicity properties. Section 7.1 describes the side-conditions that are necessary to guarantee that refinement is monotonic with respect to the chart composition operator.

Even though the monotonicity side-conditions described in Proposition 7.1.1 are presented before the monotonicity result itself, the conditions were formulated and refined from the proof of the monotonicity property (see Appendix B.14 page 264). That the process of proving the monotonicity property allows us to state (and prove) these necessary side-conditions is evidence that the goals of this thesis have been met. That is, the formal framework presented allows us to formulate precise descriptions of general, and typically non-obvious, language properties. In the case of the monotonicity result presented here, the first of the three required side-conditions is particularly non-obvious and at first reading may appear incorrect. How-

ever, the proof of monotonicity and careful evaluation of what this condition actually entails, makes the significance of the restriction clear.

Section 7.2, outlines the most significant difference between the original μ-Charts [78] and the semantics and refinement notion presented here. This follows on from the observations made in Section 6.3 (page 130) where we excluded the do-nothing and partial chaotic trace semantic interpretations for the given chart semantics. Finally, Section 7.3 argues some conclusions.

7.1 Monotonicity of the μ-Charts composition operator

We begin by showing that the composition operator of μ-Charts is monotonic with respect to forward simulation refinement only when appropriate side-conditions hold. Like the investigation of [21], the monotonicity proof itself is used to establish the necessary side-conditions. After ascertaining the required side-conditions an intuitive (in chart terms) justification for their necessity is given.

Recall that, by definition (Section 6.6.1, page 142), to show that a forward simulation refinement holds between two charts requires that we show that an appropriate simulation exists between the charts. The proof of monotonicity relies heavily on splitting the definition of the simulation into two parts, the simulation between the respective charts' configurations using the *corresponding relation* and the simulation between the allowable input and output signals using the relation IO (see Section 6.4, page 136). This notion of splitting the simulation relation was introduced in Section 6.4 where we define the corresponding relation between two charts A and C as $Corr_C^A$ and the input/output relation as IO_C^A. Where previously we have denoted (total chaotic) forward simulation refinement between two charts C and A as $C \sqsupseteq_{\tau f} A$, here we supplement the relation with an explicit label that names the simulation required for refinement. So, assuming that chart C refines chart A using the simulation S, we will write $C \sqsupseteq_{\tau f}^S A$.

7.1.1 Monotonicity result for forward simulation refinement

Proposition 7.1.1 states the monotonicity result for forward simulation refinement. The proof is given in Appendix B.14, page 269.

Proposition 7.1.1 If, for arbitrary charts A_1, C_2, and signal set Ψ, we have that,

$$\frac{}{[A_1]_\Psi \sqsupseteq^T_{\tau f} [C_2]_\Psi}\ SC_1$$

$$\frac{}{out_{A_1} \cap \Psi = out_{C_2} \cap \Psi}\ SC_2$$

$$\frac{}{out_{A_1} \cap out_B = out_{C_2} \cap out_B}\ SC_3$$

where $T =_{def} Corr^{C_2}_{A_1} \wedge IO^{C_\Psi}_{A_\Psi}$ for $C_\Psi = [C_2]_\Psi$ and $A_\Psi = [A_1]_\Psi$, then for arbitrary chart B, we have the monotonicity result,

$$\frac{C_2 \ \sqsupseteq^R_{\tau f}\ A_1 \quad SC_1 \quad SC_2 \quad SC_3}{(C_2 \mid \Psi \mid B) \ \sqsupseteq^S_{\tau f}\ (A_1 \mid \Psi \mid B)}$$

where $S =_{def} Corr^{A_1}_{C_2} \wedge Corr^B_B \wedge IO^A_C$, and $R =_{def} Corr^{A_1}_{C_2} \wedge IO^{A_1}_{C_2}$.

Despite the intricate appearance of the three side-conditions required for monotonic refinement of composed charts, these conditions are not unexpected when described in terms of charts themselves.

First consider the following property that holds in general for arbitrary charts A_1 and C_2, and feedback set Ψ.

Lemma 7.1.2

$$\frac{C_2 \ \sqsupseteq^R_{\tau f}\ A_1 \quad out_{A_1} \cap \Psi = out_{C_2} \cap \Psi}{[C_2]_\Psi \ \sqsupseteq^{T'}_{\tau f}\ [A_1]_\Psi}$$

where $T' =_{def} Corr^{A_1}_{C_2} \wedge IO^{A_\Psi}_{C_\Psi}$

Given this property holds it follows that, in the context of the monotonicity proof of Proposition 7.1.1, *i.e.* where SC_1 holds, the charts A_1 and C_2 are output equivalent with respect to the signals in the set Ψ, *i.e.* $[C_2]_\Psi \approx_\mathcal{O} [A_1]_\Psi$. In words, an environment that reacts to just those signals in the set Ψ could not tell the difference between the charts A_1 and C_2. Therefore, we see that one of the properties required to guarantee monotonic refinement (with respect to composition) is that refining one part of the composition, say refining chart A_1 into C_2, cannot change the behaviour of A_1 with respect to the signals in Ψ that are used to communicate with the other part of the composition, *e.g.* chart B.

To explain the rôle of the side-condition SC_1 more specifically, with regard to the monotonicity proof (Section B.14, page 269), we describe two distinct parts that SC_1 plays in the proof.

Firstly, SC_1 enforces that the precondition of the chart, *i.e.* the set of state/input pairs for which the chart has explicitly defined behaviour, can not be weakened. Note that here we use the term weakening of the precondition in a very strict sense; side-condition SC_1 restricts any weakening of the precondition within the domain defined by the input interface of the abstract specification. Extending the domain of definition for a chart specification, *i.e.* increasing the input interface and weakening the precondition outside of the original domain, is still permitted in general.

This first aspect of the side-condition SC_1 is required for the part of the monotonicity proof related to the *correctness* property introduced in Section 6.6.1, page 139.

Figure 7.1 presents a counter-example that illustrates why this part of side-condition SC_1 is necessary in terms of charts. Given the charts A and C we clearly have that $C_2 \sqsupseteq_{\tau f} A_1$, yet it is **not** the case that $C \sqsupseteq_{\tau f} A$. That is, even though C_2 refines A_1, the composed chart C is not a valid refinement of A. The defined reaction of chart A given input $\{a\}$ is to output $\{w, t\}$, *i.e.* the two left hand transitions of chart A combine with respect to feedback to create an overall chart transition triggered by just the input $\{a\}$. However, chart C can nondeterministically choose to output $\{w, t\}$ or $\{w, s\}$ given input $\{a\}$, *i.e.* both the respective left hand and right hand transitions combine to give this nondeterministic behaviour. Therefore, C has additional nondeterministic behaviour to A and no valid refinement holds.

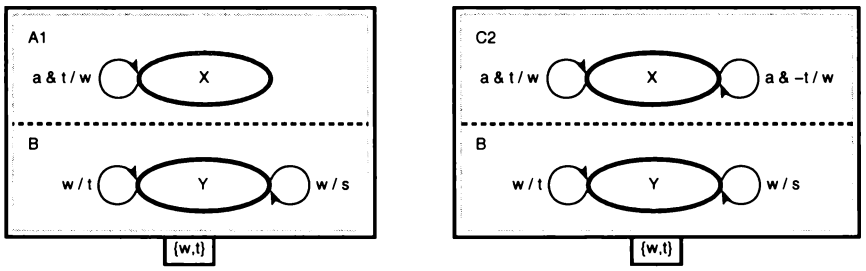


Figure 7.1: $SC_1, partI$: Charts $A = (A_1 \mid \{w, t\} \mid B)$ and $C = (C_2 \mid \{w, t\} \mid B)$

The second aspect of SC_1 is that it insists that the output behaviour, with respect to feedback, of an abstract specification is not changed via refinement. In the monotonicity proof, this aspect of SC_1 is represented by Lemma B.14.3 (see Appendix B.14, page 267). The property is required to prove the part of the monotonicity result related to the *applicability* condition.

In terms of charts, Figure 7.2 illustrates another counter-example that demonstrates why this second aspect of SC_1 is a necessary requirement for monotonic refinement. Note that the output interface of the chart C_2 is

assumed to contain the signal w, *i.e.* we assume C_2 is a behavioural refinement of A_1 rather than an interface refinement. Again we have that the composed chart C does *not* refine the chart A. This is because A is defined for input $\{a\}$ due to feedback on w where chart C is not. Therefore chart C acts chaotically for input $\{a\}$ and the resulting additional nondeterminism invalidates the refinement relation.

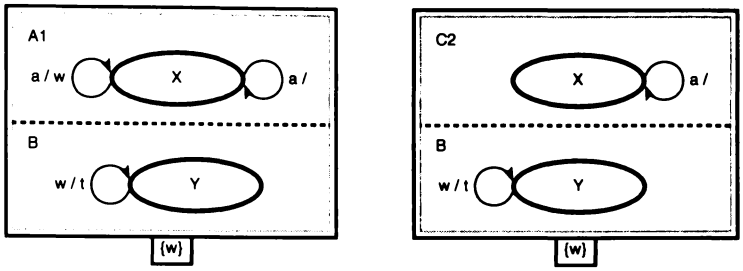


Figure 7.2: SC_1 and SC_2: Charts $A = (A_1 \mid \{w\} \mid B)$ and $C = (C_2 \mid \{w\} \mid B)$

The same charts from Figure 7.2 can be used to demonstrate why the side-condition SC_2 is required for monotonicity. In this case, however, we assume that the output interface of chart C_2 is reduced to the empty set of signals, that is, in this case C_2 is an interface refinement of A_1 rather than a behavioural refinement as above. Given this assumption SC_1 holds, that is, $[\![A_1]\!]_\Psi$ is a valid refinement of $[\![C_2]\!]_\Psi$. However, from inspection it is obvious that SC_2 does not hold in this case, that is, $out_{A_1} \cap \Psi \neq out_{C_2} \cap \Psi$, specifically, $\{w\} \cap \{w, t\} \neq \{\} \cap \{w, t\}$. The side condition SC_2 is required to prove monotonicity in relation to the *correctness* condition.

Finally, the side-condition SC_3 is required because μ-Charts refinement allows the designer to change the output context of a chart using interface refinement. If an interface refinement of one chart in a composition extends the control that the chart has over the environment using signals that were originally used just by the other part of the composition, then there is the possibility that this new behaviour, from both charts, will be inconsistent when the charts are recombined in composition. For example, consider the counter example illustrated by the charts of Figure 7.3.

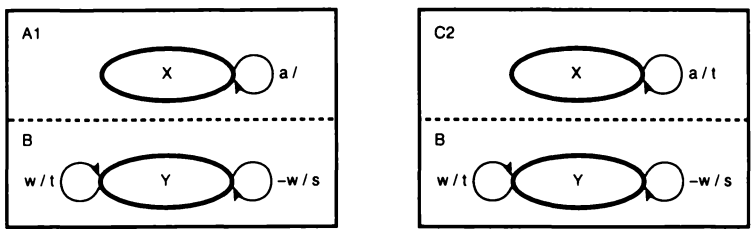


Figure 7.3: $SC_2(ii)$: Charts $A = (A_1 \parallel B)$ and $C = (C_2 \parallel B)$

Here the valid interface refinement $C_2 \sqsupseteq_{\tau f} A_1$ allows C_2 to control its environment over signals previously dealt with by the chart B, *i.e.* the signal t. The result is that the composed chart C can output $\{s, t\}$ for input $\{a\}$ where chart A could only output $\{s\}$ for input $\{a\}$. Hence, chart C has new behaviour that was not specified by chart A and therefore C is not a valid refinement of A.

Similar arguments can be used to show that the same side-conditions, SC_1, SC_2 and SC_3, are sufficient to guarantee monotonic refinement with respect to the composition operator for charts in the backward simulation case.

7.1.2 The firing conditions interpretation of μ-Charts

A requirement for monotonic refinement is that the preconditions remain unchanged over the domain of definition of an abstract chart. This requirement may cause an observant reader to question whether the *total chaotic* and *firing conditions* notions of refinement coincide in the case where refinements adhere to the monotonicity conditions. In particular, the work of Deutsch, Henson and Reeves in [17] shows that refinement based on a *firing conditions* approach can be considered as a notion that insists on the *stability of the precondition.* That is, refinement that allows the reduction of nondeterminism but insists that the precondition is neither strengthened nor weakened.

In fact, we can show that *total chaotic* refinement is both sound and complete with respect to *firing conditions* refinement when we insist that just the first condition SC_1, for monotonic refinement, is met. Any (guaranteed) monotonic refinement that we can prove using the total chaotic rules can also be proved using the rules for firing conditions refinement.

This is expressed by Proposition 7.1.3 which is proved in Section B.14, page 276.

Proposition 7.1.3 For arbitrary charts A, C and signal set Ψ we have,

$$\frac{C \sqsupseteq^S_{\tau f} A \qquad [A]_\Psi \sqsupseteq^T_{\tau f} [C]_\Psi}{C \sqsupseteq^S_{fcf} A} \qquad \frac{C \sqsupseteq^S_{fcf} A}{C \sqsupseteq^S_{\tau f} A}$$

where $S =_{def} Corr^A_C \wedge IO^A_C$ and $T =_{def} Corr^C_A \wedge IO^{C_\Psi}_{A_\Psi}$ for $C_\Psi = [C]_\Psi$ and $A_\Psi = [A]_\Psi$.

Notice that the second aspect of the side-condition SC_1 and the conditions SC_2 and SC_3 are still a necessary requirement to guarantee that firing

conditions-based refinements are monotonic with respect to composition. In particular, the property described by Lemma B.14.3 (Section B.14, page 267) still needs to be shown to ensure a monotonic firing conditions refinement.

Therefore, while it is the case that using $\sqsupseteq_{fcf}$ for chart refinement implies a "more monotonic" refinement calculus, the difference in reality is slight.

The exact difference between the two notions of refinement is that the total chaotic model allows a refinement to weaken the precondition over the abstract domain of definition where the firing conditions model does not. The choice of the appropriate model can only be determined by the context of the refinement application. We do point out, though, that the total chaotic model provides the most general refinement framework.

7.2 Total correctness *vs.* partial correctness

Here we outline and investigate the most significant difference between the original semantics for μ-Charts, in [78], and that given here. Apart from the obvious difference, that is, the original operators were defined in terms of infinite traces, the significant semantic difference is the meaning given to charts that use structuring operators in their definition. In the previous definition [78], the meaning of a composed chart is given in terms of the "totalised" meaning of the two composite charts. Here the meaning of a composed chart is given using the partial definition of the two composite charts. "Totalisation" then happens to the partial meaning of the composite chart.

First we give some examples that makes clearer exactly what is meant by totalising before or after composition, then we discuss how the original semantics might be encoded in a relational framework similar to the relational framework used here. This allows us to consider further the divergence of the two language definitions in terms of the notional charts *Chaos* and *True* introduced in Section 6.3 , page 130. Finally, we will outline the some of the ramifications, in terms of refinement, of choosing one or other language definition. This includes further consideration of monotonicity concerns.

To illustrate the difference between the respective definitions of μ-Charts, we use the three so-called pathological examples of composition, introduced by Scholz in Chapter 2 of [78]. For each of the examples we use a type of chart "meta-notation" to describe the meaning of the example compositions in terms of sequential charts. This "sequentialisation" of the composed charts makes it easier to see the differences between the two semantic definitions.

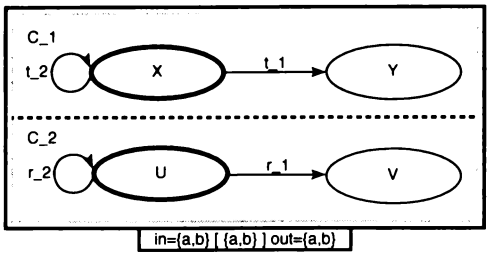


Figure 7.4: General form of "pathological" examples

The idea of the "meta-notation" is that it allows us to encode explicitly the chaotic totalisation that occurs under both of the semantic definitions. Recall that the significant difference between the two definitions is due to when this totalisation occurs.

Each of the three examples have the same form, which is described by the generic chart $S_n = C_1 \mid \{a, b\} \mid C_2$ of Figure 7.4.

The three example charts S_1, S_2 and S_3 follow from the generic chart by replacing the transition labels t_1, t_2, r_1 and r_2 as follows:

S_1: $t_1 = a/b$, $t_2 = -a/$, $r_1 = b/a$ and $r_2 = -b/$

S_2: $t_1 = -a/b$, $t_2 = a/$, $r_1 = -b/a$ and $r_2 = b/$

S_3: $t_1 = a/b$, $t_2 = -a/$, $r_1 = -b/a$ and $r_2 = b/$

In the following we refer to the semantics given here as the *total correctness semantics* and the original semantics (from Scholz [78]) as the *partial correctness semantics*. These phrases are not suggesting that one semantics is more "correct" than the other, simply that they differ in the meaning assigned to composed charts. We describe in the following exactly why one may be considered "total" and the other "partial".

Figure 7.5 demonstrates the total correctness semantic interpretation of chart S_1.

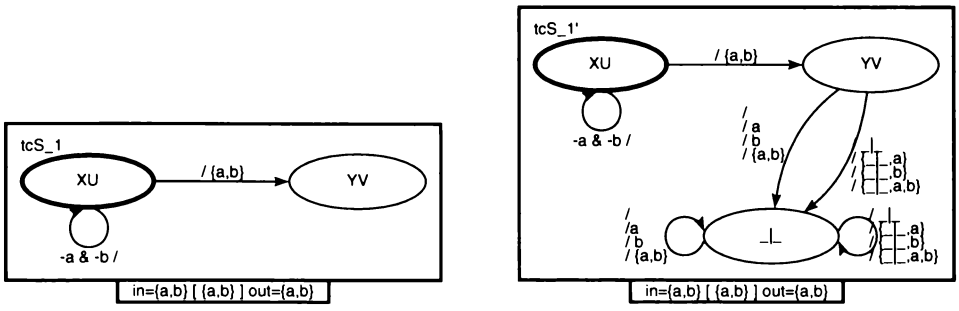


Figure 7.5: Total correctness interpretation of pathological chart SC_1

The chart tcS_1 is derived from chart S_1 by using the sequentialisation technique described in Section 3.4. Chart tcS_1' is then the total-chaotic totalisation of chart tcS_1 using our so-called meta-notation. This includes the special state labelled $\perp$, the new signal $\perp$ and four extra transitions. Note, the labels on these transitions are a shorthand for four individual nondeterministic transitions labelled " / ", " $/a$", " $/b$" and " $/\{a, b\}$" respectively. The transition label " / " denotes a transition with a true trigger condition, *i.e.* it is triggered by any input, and the command that outputs the empty set of signals.

Now consider the partial correctness interpretation of chart S_1, demonstrated in Figure 7.6. Here the partial correctness semantics is encoded by totalising the two original sequential charts to give C_1' and C_2' (left) and then using the same sequentialisation technique to give the sequential chart pcS_1' (right).

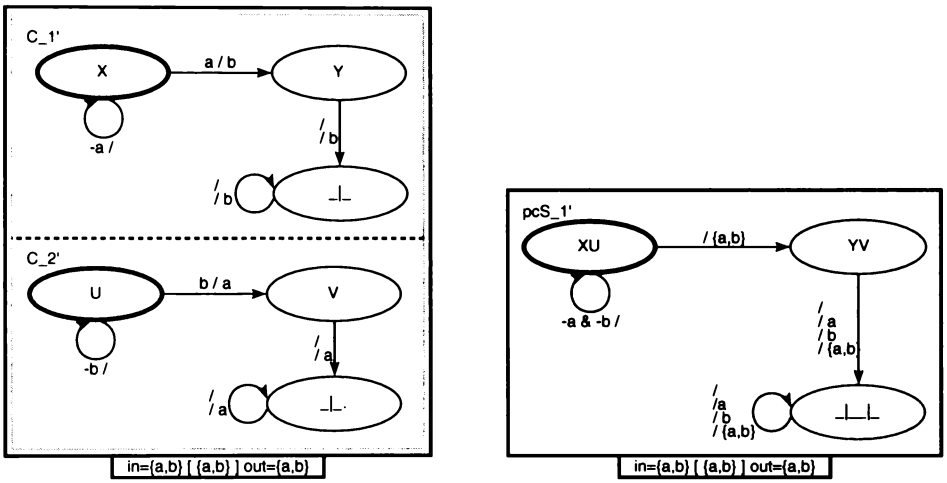


Figure 7.6: Partial correctness interpretation of pathological chart SC_1

The partial correctness meaning is essentially the same as the total correctness meaning. At least, in both cases tcS_1' and pcS_1', we could remove the introduced meta-notion constructs and rely on the implicit chaotic semantic interpretation to encode the meaning in each case.[1]

Now consider chart S_2, the second example of the chart in Figure 7.4. The total correctness interpretation of chart S_2 is demonstrated in Figure 7.7. Chart tcS_2 is the result of sequentialising S_2 and chart tcS_2' gives the total

[1]Note that the examples in [78] on which these are based appear to assume implicitly the behaviour we have encoded explicitly using the generic transitions labelled t_2 and r_2. That is, it appears that Scholz is assuming a "do-nothing" interpretation for the empty input in the initial states.

meaning of S_2.

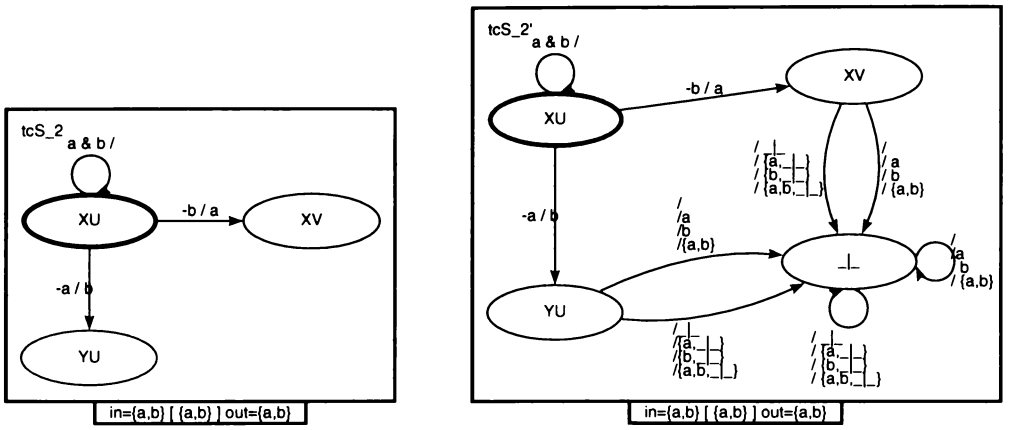


Figure 7.7: Total correctness interpretation of pathological chart SC_2

Alternatively, Figure 7.8 demonstrates the partial correctness interpretation of S_2. Here we see that pcS_2' is significantly different from tcS_2'. Where tcS_2' represents persistent chaotic behaviour for any input after the chart reaches one of states XV or YU, the chart pcS_2' does not. For instance, from state XV, given the input $\{a\}$, chart pcS_2' gives exactly the output $\{a\}$.

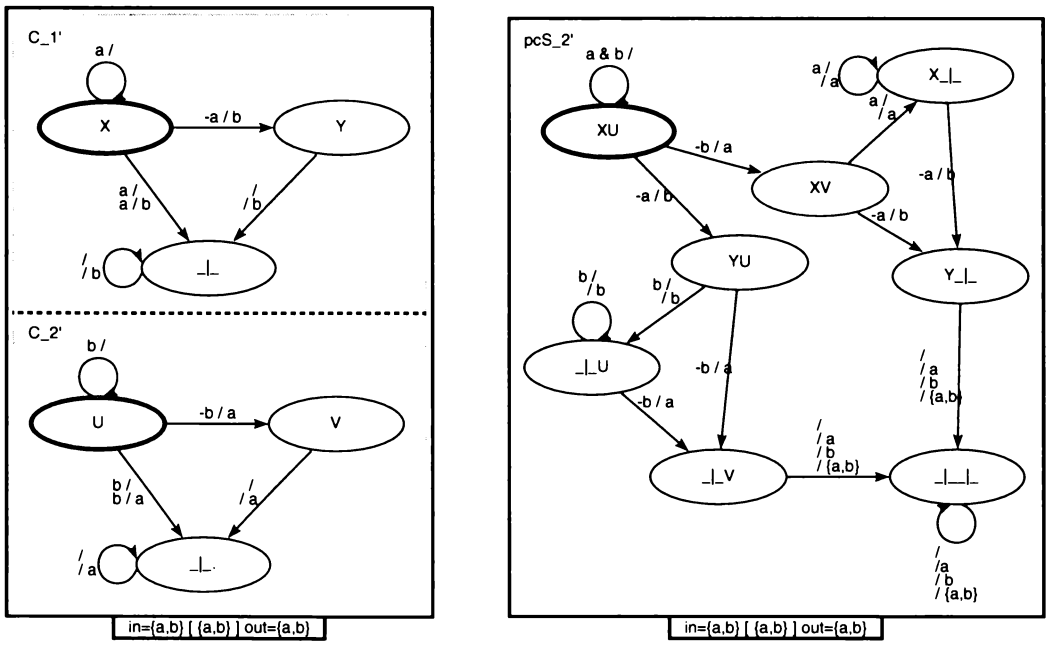


Figure 7.8: Partial correctness interpretation of pathological chart SC_2

This example demonstrates that the original partial correctness semantics encodes *unobservable chaotic behaviour*. That is, one part of the composition behaving chaotically does not affect the other part of the composition. This is in contrast to the total correctness semantics given here that encodes *in-*

trusive chaotic behaviour as described in sections 4.3 and 6.3 (pages 80 and 130).

Significantly, unlike the first example, we can no longer remove all of the meta-notion from the chart pcS_2 and rely on an implicit encoding of the required chaotic semantics. That is, if we were to remove the "meta-states", for example state $\perp U$, and the associated transitions leading to and leaving from those states, the implicit encoding of chaotic behaviour thereafter would no longer have the same meaning as the chart pcS_2' as pictured. However, given that refinement uses simulation, we could still show that the original example chart S_2 is equivalent (by a refinement relation in both directions) to a chart like pcS_2 without state $\perp\perp$.

Finally, for the example chart S_3, the total correctness interpretation is demonstrated in Figure 7.9:

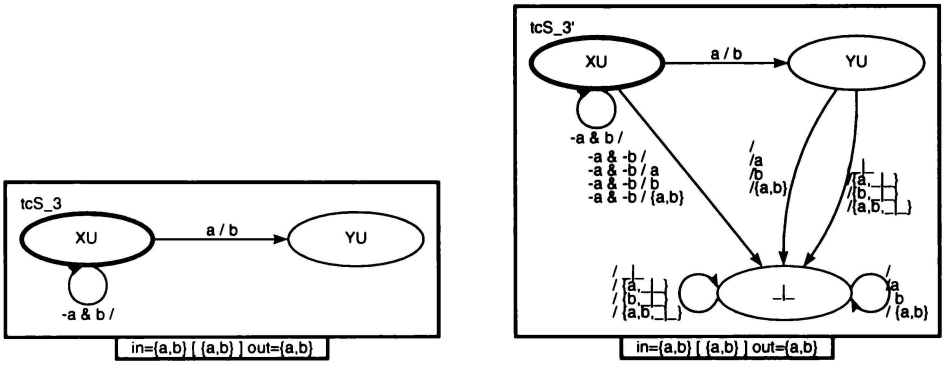


Figure 7.9: Total correctness interpretation of pathological chart SC_3

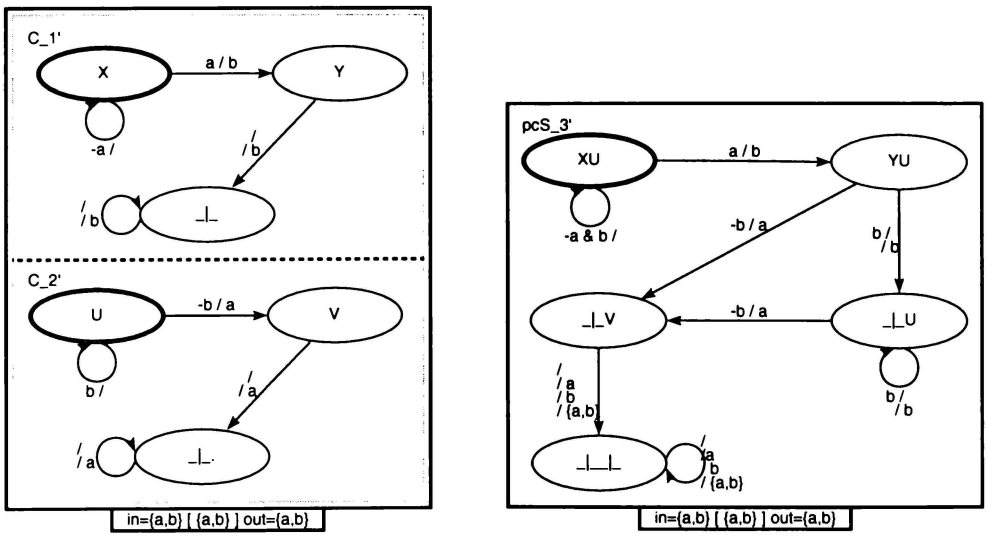


Figure 7.10: Partial correctness interpretation of pathological chart SC_3

The partial correctness interpretation however, given in Figure 7.10,

demonstrates that in cases where composed charts have transitions that conflict for some input, the specification is essentially unimplementable as a reactive system, for that input. In particular, chart pcS_3', given the empty input while in state XU, is undefined. Because the totalisation of undefined behaviour happens before the composition in the partial correctness case the point at which transitions conflict has no implicit meaning, it is simply undefined. Hence, there are no valid refinements that reintroduce behaviour for that particular input, and therefore, there are no "reactive" implementations for this specification. We discuss the ramifications of this in the following section.

7.2.1 Refining composition

It is possible to define the composition operators for the language μ-Charts, using the methodology and framework presented here, so that it encodes unobservable chaos, *i.e.* the *partial correctness* approach discussed above. A simple approach would be to explicitly encode the necessary machinery, *i.e.* the special state $\perp$ and related transitions, directly in the model of charts. Deriving refinement rules that encode this model implicitly is, however, still an open research question.

While we cannot describe the exact form of a partial relations-based refinement calculus which implements a partial correctness semantics, we have shown that one significant difference between the two approaches (partial correctness and total correctness) will be the ability to resolve conflicting behaviour introduced via the composition of two charts.

In the context of monotonicity concerns, this difference will impact on the way that reactive systems can be developed from specifications to implementations via refinement. For example, consider the case where each part of a composite specification is been developed separately using refinement, *i.e.* the situation in which monotonic refinement is of concern. When the separate developments are recombined to represent a more concrete implementation of the original specification, the choice between a partial correctness model and a total correctness model will become apparent if both of the separate developments have defined behaviours over some previously undefined input which turn out to be conflicting.

First, for the total correctness refinement model the point at which the conflicting behaviour occurs will be represented by total nondeterministic behaviour, *i.e.* chaotic behaviour. This chaotic behaviour remains to be

refined to the appropriate deterministic implementation. That the conflicting behaviour is represented as chaos means that the behaviour must have been undefined, and therefore chaotic, in the specification itself. This is why the conditions necessary for monotonicity are quite strong in the total chaotic model.

On the other hand, assuming a partial correctness model, the conflicting behaviour introduced by recombining the separate developments will be represented as an unimplementable point in the specification. This will require that the designer(s) decide and fix the appropriate development so that the conflicting behaviour is not introduced. Then the development can continue to the implementation. Given that the conflicting behaviour will be represented as an undefined point in the domain of the specification suggests that the conditions necessary to ensure monotonic refinement in the partial correctness model will be significantly weaker than those described above.

7.3 Discussion

In this section we have shown how the formal framework presented in this thesis can be used to prove general non-obvious properties that hold of μ-Charts.

In particular, we formulated and proved the side-conditions necessary to guarantee monotonic refinement with respect to one side of a composed μ-chart. Also, we discuss the significant difference between the semantic definition of μ-Charts given here and the original definition given by Scholz in [78]. That is, the semantic definition that we give encodes an "infectious chaos" or "observable chaos" through chart composition where the original semantic definition captured a "restricted-to-one-chart chaos" or "unobservable chaos".

More generally, this section demonstrates the purpose of the formal framework for μ-Charts which is the contribution of this thesis. That is, firstly, the ability to investigate properties of the language with a high level of confidence that the properties are correct in general. We can see from the justification of the monotonicity property by examples, that to formulate such properties using a case-by-case examples-based approach is at least difficult if not impossible. Obviously it is impossible to prove general properties correct by example. This frailty of formulating properties by example is reinforced again in Section 7.2 where we discuss the differences between the respective semantic definitions of μ-Charts. Here we reconstruct three specific repre-

sentative examples in an attempt to highlight these differences; however it is almost certainly the case that the subtleties involved are difficult to see using limited examples.

The second strength of a formal language is the ability to compare and comment on language design issues. Though we have done little here by way of a formal comparison of the different definitions of μ-Charts, we have shown that it is at least possible to give a similar notion of refinement for charts using a partial relations-based formulation rather than the more usual traces of behaviour approach. Also, the process of formalisation has allowed us to make some insightful comments on the significant differences between the respective language definitions; that these differences became clear was strictly due to the process of formalising the language definition (including refinement). In the literature there are several comprehensive examples of comparing the significance of language design using the respective languages' formal semantics; for examples see [20, 53].

Chapter 8

Conclusions

The aim of this thesis was to provide a rational reconstruction of μ-Charts via a "proof-theoretic" treatment of the language. The intention of the work presented was to provide a thorough account of the language definition, including a logic for reasoning over the structure of reactive system specifications and about formal (*i.e.*, including proofs) program development using refinement. The work is meant to facilitate ongoing research that results in a formal program development framework that can be used by practising engineers to develop correct implementations of reactive system specifications.

The method that we employ in this research closely follows that introduced by Deutsch, Henson and Reeves in their investigation of the specification language Z via set theory. That is, we take the existing language μ-Charts and model it in another well-known formal language Z. From the model and the existing logic for Z we deduce a logic specifically for μ-Charts. The process of using the logic to investigate properties of the language is used to guide the design (*i.e.*, the form of the logic rules required) of the logic itself. The proofs of these properties are considered to be the experiments which determine that the logic is sufficient for proving useful properties about the language.

An outline of the contributions of this thesis is as follows:

- We provide a detailed syntactic and semantic description of the language μ-Charts. This includes the formulation of a set of rules that provide a logic for μ-Charts. The rules of the logic are given as a series of introduction and elimination rules that allows natural deduction style proofs of language properties;

- We contrast several possible semantic interpretations that can be assigned to μ-charts in the case where a specific definition of required

behaviour is left unstated;

- We discuss in detail (in terms of the usual traces of behaviour semantics for communicating systems) the notion of refinement that we assign to the language via the definition of a calculus for refinement;

- We derive a calculus for the refinement of μ-charts. This calculus consists of introduction and elimination rules that allow a natural deduction-style proof of correctness to be formulated to show that the required refinement relationship holds between two μ-Charts specifications;

- We briefly consider the important monotonicity properties of the resulting notion of refinement. This includes outlining the significant difference between the language described here compared with the definition of μ-Charts given by Scholz in [78].

8.1 Outcomes

Here we discuss several outcomes of the presented research that we consider significant.

The examination of μ-Charts presented here is a comprehensively documented reconstruction of a variant of the original language. We describe in detail what μ-Chart specifications denote and what it means when we prove that a refinement relation holds between two specifications. The reconstruction of the language demonstrates the complexity of the language. This complexity is easily forgotten given the graphical nature of the formalism. That is, it is easy to convince oneself (wrongly) that required behaviour is captured by a μ-Charts specification. This emphasises the necessity of formal specification verification and formal refinement for visual representations of system requirements.

We have demonstrated how a partial relations-based semantics for charts can be constructed that captures properly the behaviour of a μ-Chart specification. This was identified as important in the original specification of the language, by Scholz, as a practical means to aid in the formal verification of reactive system specifications via model checking. However, the original relational encoding gave denotations to a limited set of μ-Chart specifications only. Here we show how we can define the language strictly in terms of the denotational semantics provided by the Z model of charts. Rather than

considering model checking as a means of formal verification, we concentrate on allowing the formulation of formal natural deduction style proofs as a method for formal verification.

We show how a well-known method for deriving data refinement rules for Z can be extended to deal with a particular type of interface refinement. That is, refinements that allow changes to the input and output characteristics of a specification, as well as the usual changes in the state representation from abstract to concrete specification.

Our investigation of the monotonicity properties of refinement highlighted the significant difference between the μ-Charts described here and the μ-Charts previously described by Scholz. The chart semantics described here encodes an intrusive form of chaotic behaviour. If one chart in a composition is considered chaotic the entire specification is considered chaotic. This is in contrast to the previous semantics of Scholz [78] that allowed isolated chaos. That is, chaos in one chart of a complex specification does not affect any other charts. Significantly, this difference in the semantic definition of the chart structuring operators affects the monotonicity of the resulting notion of refinement. The side conditions required to guarantee refinement is monotonic are stronger for the refinement described here than they are for the refinement defined in the isolated chaos semantics. On the other hand, it is possible, given the semantics defined here, to translate any complex specification into a simple sequential specification, whereas, in the Scholz semantics there exist complex charts that do not have an equivalent sequential chart. Therefore, improvements in the monotonicity properties are offset by a loss of the ability to reason using equality. We chose to retain the ability to reason with equality, *i.e.* allowing a designer to conceptualise the behaviour of composed charts as though they behave in the same manner as some sequential chart, at the expense of stronger side-conditions for monotonic refinement.

Also, related to the previous observation, the given definitions of the composition operators and refinement notion for charts presents a total correctness semantics. That is, if the composition of two charts is undefined, from some state given some input, the result is that this undefined behaviour is considered chaotic and therefore it can be correctly refined into any behaviour. On the other hand, the original semantics might be considered a partial correctness semantics because a designer can specify a reactive system that is not implementable. In some cases, given the semantics of the composition operator, composing two inconsistent specifications can result in a specification for which there exists no reactive system implementation.

Of course, this remark assumes a context where a reactive system implementation is considered to be an implementation that is prepared to react to any input provided by the environment.

8.2 What we have not done (Future work)

The following description of why the work presented is not yet complete is the motivation for the future research questions that we propose.

At best the work presented here represents a core logic for μ-Charts. Unfortunately, it does not represent a "practical" tool that could be easily adopted by practising engineers building reactive systems. The intention of this work is to provide a solid foundation from which a formal framework for reactive system development, including correctness proofs of program derivations, can be constructed.

Significantly, we note that the only "experiments" that have been conducted using the presented logic prove properties of the object language. Though important from a language construction point of view, this does not provide significant insight into how the logic should be developed further to become a practical tool for its intended purpose, *i.e.* formal program development of reliable reactive systems. While it is possible to prove that given refinement steps are valid using the logic presented, it is not yet developed enough to be considered a practical development tool. As Scholz suggested in [78], the logic for refinement needs to be lifted into the object language.

This lifting of the logic should be developed carefully based on both theoretical and practical considerations. The logic should provide an extensible "refinement toolkit" that is formed by practical attempts to develop reactive systems from formal requirements. Several case studies that involve the realistic development of reactive systems should provide the motivation for the initial rules. Common program derivation steps should be encoded in the logic. Like the well-understood notion of "software reuse"—common to the development of large software systems—a similar concept of refinement reuse should be adopted. In other words, refinement strategies can be developed for particular, common program development tasks. The core logic here can be used to ensure that the lifted logic is sound.

Another relevant question that remains to be answered is: does the formalisation of refinement itself suggest useful methods for developing reactive systems? For example, it is common in presentations of data refinement to consider the simulation relation, required to prove that a refinement exists,

as a part of the design process. That is, the simulation relation is chosen precisely to represent a particular type of refinement, for example changing abstract data types such as sets into more concrete data types such as sequences. Identifying particular types of simulations that provide useful program developments for reactive system could provide significant ways to enrich the "refinement toolkit" for reactive systems.

Tool support will be a necessity for any practical formal development framework. It is clear that tools that provide simple facilities, such as type checking for Z specifications, offer significant benefits in user satisfaction. It is almost certainly the case that even such small enhancements, offered by tool support, significantly increased the usability of the language. Before we can accurately determine whether working engineers can benefit from the process of providing proofs of correctness for their program developments, significant tool support would be required. The tools need to be built with the eventual user in mind rather than considering just the technical aspects of theorem proving. A significant requirement initially is a tool that aids in providing natural deduction-style proofs using $\mathcal{Z}_C$. It should allow the formulation of proofs in a manner natural to this style of proof. Some of the obvious facilities required include: allowing both top-down and bottom-up reasoning; managing undischarged assumptions that are introduced by application of proof rules; facilitating subproof construction, including the ability to contract or hide the details of subproofs leaving visible just the assumptions and conclusion; the ability to easily rearrange proofs to allowing whole subproofs to be selected and moved. The tools should be specific to the object language—translating the object language into a "tool-suitable" language is a significant overhead and source of mistrust and introduction of errors.

The monotonicity properties of the language need to be investigated from a user's point of view. As we outlined above, the choice of model for the language makes a significant difference to the monotonicity properties of the language. The choice of the most appropriate model needs to be made by investigating the effect of the monotonicity side-conditions and the ability to reason using equality on "typical" reactive system designs. The monotonicity side-conditions are strongly related to the interface between two parts of a composite chart. It is not clear how this interface will be managed when formal program development follows the structure of the specification, nor whether it is sensible to expect that the structure of the specification should guide the structure of the design. Using a semantics that provides monotonic

schema operators for Z to model μ-Charts and deriving a related logic, using the method demonstrated here, will be necessary to contrast the alternative monotonic chart operators in the same meta-language.

Finally, the integration of program development into the μ-Charts language needs to be considered. Currently, the language only deals with specification statements. Refinement is the process of transforming an abstract specification into a concrete specification. As it stands the formal development is complete when there is an obvious implementation of the concrete specification. However, integrating program constructs into the language should allow for formal development from specification to implementation.

In summary, necessary future work includes:

- Lifting the $\mathcal{Z}_C$-based logic into a logic that allows useful "high-level" refinements specifically designed for the development of reactive systems using the object language μ-Charts. The logic, for the formal program development of reactive systems, should be developed using case study-based experiments to guide the design of appropriate logic rules and completeness of the language. That the language is sound can be proved using the logic presented here.

- An investigation into what parts of the formal design of the logic for refinement, based on existing data refinement techniques, can be used to identify useful formal development strategies for reactive systems. In particular, the possibility that the simulation relation may be used to guide and document design;

- Tool support should be developed for the formal program development framework. The tools should obviously encode the formal semantics of the framework properly, but also be constructed with the user in mind. Integration with the object language should be seamless, and usability concerns a significant objective.

- An investigation into the way in which the monotonicity properties of refinement affect formal design. In particular, what strategies are useful for dealing with the interfaces (communication) between parts of complex reactive system specifications during formal program development. Providing another logic for μ-Charts, based on a monotonic logic for Z, will be necessary for comparison.

- Investigating the integration of reactive system program constructs,

and related program derivation rules, into the specification language μ-Charts.

8.3 Concluding the conclusions

The task of providing a reconstruction of the language μ-Charts itself provides significant insight into the task of developing complex systems. Several analogies can be made between the task of a formal investigation of the language, presented here, and the task for which the language is meant to be used, that is, formal development of reactive systems. The specification of the language μ-Charts was taken primarily from the work of Scholz [78]. While this language specification was concise and well-investigated, the subtleties of the language itself were not fully understood until the task of formalising (or implementing a logic for) the specification was undertaken. A rigorous formal investigation of the language specification enforces a clear understanding of the problem and resolution of unclear characteristics of the system. Also, the formalisation provides clear documentation of design decisions.

If we view the task of implementing complex systems as ultimately a formal encoding of an informal specification, then it is clear that the process of implementing reactive systems will entail the same complexities as the formal definition of the language itself. Formal program development must then offer the same benefits as those identified in this investigation of the language μ-Charts.

Appendix A

$\mathcal{Z}_C$ and Type Conventions

In this appendix we give a brief introduction to the kernel Z-logic $\mathcal{Z}_C$. The introduction is very similar to that given in [19]. For a detailed description of the logic $\mathcal{Z}_C$ and examples of its use we suggest that the reader consults [19] and [38]. We also introduce several conventions that are used throughout this thesis. In particular, due to the uniform nature of the Z that is used to give the semantics to μ-Charts, we introduce several notational coventions that are assumed when using the type system of $\mathcal{Z}_C$.

A.1 The logic $\mathcal{Z}_C$

$\mathcal{Z}_C$ is a typed theory in which the types of higher-order logic are extended with *schema types.* Schema type values are unordered, label-indexed tuples called *bindings.* For example, given the indexed types T_i and labels l_i then schema types take the form:

$$[...l_i : T_i...]$$

The bindings that inhabit this type are of the form:

$$\langle\!| \; ...l_i \Rrightarrow t_i{}^{T_i} ... \; |\!\rangle$$

where each labelled observation l_i takes the value denoted by the term t_i of appropriate type T_i.

The symbols $\preceq$, $\curlywedge$ and $\curlyvee$ denote the *schema subtype* relation and the *schema type intersection* and *schema type union* operators. The $\mathcal{Z}_C$ meta-operator α returns the alphabet set of labels for a schema type. That is, given the schema types $T_0 = [...l_i : T_i...]$ and $T_1 = [...m_i : T_j...]$, the expression $\alpha(T_0 \curlyvee T_1)$ returns the set of labels $\{...l_i...m_i...\}$. In particular, we write $[\alpha T/z.\alpha T]$ to indicate the family of substitutions $...[l_i/z.li]...$ where z is some

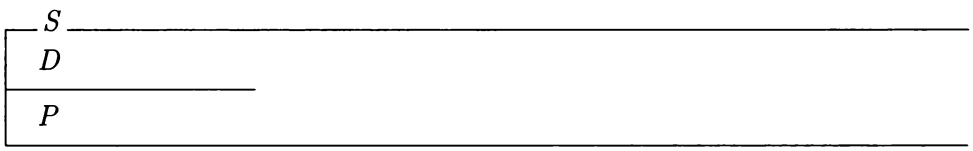
term and $\alpha T = \{...l_i...\}$. For example, given the schema type $T = [m : \mathbb{N}]$ and binding z^T, we have $(m < 0)[\alpha T/z.\alpha T] =_{def} z.m < 0$.

The binding operator *select*, denoted by '.', allows selection of the value of a specific labelled observation from a binding instance, that is $(\!| l \Rightarrow t |\!).l =_{def} t$. The operator *filter*, denoted by '$\restriction$' restricts a binding to include just those observations in a specified schema type. For arbitrary binding z and schema types $T_2 \preceq T_1$ we have, $z^{T_1} \restriction T_2 =_{def} y^{T_2}$ where $z.t_i = y.t_i$ for all $t_i \in \alpha T_2$. We also make extensive use of the *binding concatenation* operator, written $x \star y$ with the assumption that the respective bindings x and y are of disjoint type.

Now the interpretation of traditional Z schemas in $\mathcal{Z}_C$ is simply sets of bindings as follows:

$$[\![[S \mid P]]\!]_{\mathcal{Z}_C} =_{def} \{z \in [\![S]\!]_{\mathcal{Z}_C} \mid [\![P[\alpha S/z.\alpha S]]\!]_{\mathcal{Z}_C}\}$$

In using the logic $\mathcal{Z}_C$ we omit typing information and the semantic brackets whenever the context permits. That is, we often write predicates of the form $z \in S$, where z is a binding and S is the name of a schema such that $S \mathrel{\widehat{=}} [D \mid P]$ or more commonly:



That the semantic brackets are omitted is clear from the use of the membership relation $\in$. The typing information can be recovered from the declaration part of the schema S (in our example the meta-variable D). That is, assuming the schema type $T = [D]$ then $[\![S]\!]^{\mathbb{P}\,T}_{\mathcal{Z}_C}$ and z^T.

In order to reduce excessive use of the binding operator filter, we use the $\mathcal{Z}_C$ notational conventions *dotted membership*, *dotted equality* and *typed equality* which are defined respectively as follows:

$$z^{T_0} \mathrel{\dot{\in}} S^{\mathbb{P}\,T_1} =_{def} z \restriction T_1 \in S \qquad [T_1 \preceq T_0]$$

$$z_0^{T_0} \mathrel{\dot{=}} z_1^{T_1} =_{def} t_0 \restriction (T_0 \rightarrowtail T_1) = t_1 \restriction (T_0 \rightarrowtail T_1)$$

$$z_0^{T_0} =_T z_1^{T_1} =_{def} t_0 \restriction T = t_1 \restriction T \qquad [T \preceq T_0 \text{ and } T \preceq T_1]$$

Finally, we recapitulate the $\mathcal{Z}_C$ definitions for schema *renaming* and hence *schema priming* and *binding priming*. Renaming refers to the systematic substitution of labels (*e.g.* $[l_0 \leftarrow l_1]$— label l_0 is renamed l_1) and is defined by the following rules:

$$\frac{t \in S}{t[l_0 \leftarrow l_1] \in S[l_0 \leftarrow l_1]}\ (S^{+}_{\leftarrow}) \qquad \frac{t[l_0 \leftarrow l_1] \in S}{t \in S[l_0 \leftarrow l_1]}\ (S^{-}_{\leftarrow})$$

All occurrences of labels are systematically replaced, for example:

(i) $(\!| \ldots l_i \Rrightarrow t_i \ldots |\!)[l \leftarrow m] = (\!| \ldots l_i[l \leftarrow m] \Rrightarrow t_i[l \leftarrow m] |\!)$

(ii) $t.l_0[l \leftarrow m] = t[l \leftarrow m].l_0[l \leftarrow m]$

(iii) $l[l \leftarrow m] = m$

(iv) $l_0[l_1 \leftarrow m] = l_0$ when $l_0 \neq l_1$

Now, schema and binding priming are defined simply in terms of renaming. That is, given the generic schema type $T = [\ldots l_i \ldots]$, for schema $S^{\mathbb{P}\,T}$ and binding z^T we have:

$$S' =_{def} S[\ldots l_i \leftarrow l_i' \ldots]$$
$$z' =_{def} z[\ldots l_i \leftarrow l_i' \ldots]$$

The corresponding typing convention, that is $T' = [\ldots l_i' \ldots]$, is also assumed.

Importantly, the Proposition A.1.1 can be proved trivially using the definition of priming and the rule $(S\!\pm\!)$.

Proposition A.1.1 For the arbitrary schema S and binding z we have,

$$z \in S \Leftrightarrow z' \in S'$$

In the following section we introduce μ-Chart specific notational conventions, including a restricted form of the priming operator.

A.2 Type notation conventions

The previous section introduced some of the Z-logic $\mathcal{Z}_C$ in all its generality. The use of the logic in this dissertation, however, is much more restricted due to the uniform nature of the types of schemas required to give the meaning of μ-Charts. Therefore, we introduce typing conventions here that are assumed throughout the investigation of the Z-based semantics for charts.

Firstly, the schemas that describe the meaning of a chart can be considered, in general, as a relation between the before-state and input, and the after-state and output of each transition. Therefore, we are able to introduce the following typing conventions.

For arbitrary chart C we assume that the schema type U contains the observation(s) that describe the chart's state information. That is, we assume,

$$[\![Chart_C]\!]^{\mathbb{P}\,U}_{\mathcal{Z}_C}$$

where the schema $Chart_C$ is defined, as in Chapter 3, to capture the state information for chart C with arbitrary structure.

The schema types V^i, V^o, T^{act}, T^i, T^o and T^{io}, in relation to a chart C, are then defined as:

$$
\begin{aligned}
V^i &=_{def} [i_C : in_C] \\
V^o &=_{def} [o_C : out_C] \\
V^{io} &=_{def} V^i \curlyvee V^o \\
T^{act} &=_{def} [act : \mathbb{P}\, \mu_{State}] \\
T^i &=_{def} U \curlyvee V^i \\
T^o &=_{def} U \curlyvee V^o \\
T^{io} &=_{def} U \curlyvee T^i \curlyvee T^o
\end{aligned}
$$

Now we can easily describe the uniform relational type of the Z-meaning of both the transition semantics (of sections 3.1 to 3.6) and the partial relation semantics (of sections 3.7 onwards) for μ-Chart C. We have:

$$
\begin{aligned}
&[\![[\![C]\!]_{Z_t}]\!]^{\mathbb{P}(T^{act} \curlyvee T^i \curlyvee T^{o\prime})}_{\mathcal{Z}_C} \\
&[\![[\![C]\!]]\!]^{\mathbb{P}(T^i \curlyvee T^{o\prime})}_{\mathcal{Z}_C}
\end{aligned}
$$

We abbreviate the two uniform relational types further by defining types T^a and T as:

$$
\begin{aligned}
T^a &=_{def} T^{act} \curlyvee T^i \curlyvee T^{o\prime} \\
T &=_{def} T^i \curlyvee T^{o\prime}
\end{aligned}
$$

Other, common notational conventions include using indexed meta-labels δ_C in place of $[\![C]\!]_{Z_t}$ and C in place of $[\![\delta_C]\!]_{\mathcal{Z}_C}$, for chart C.

Now, context permitting, we omit the type superscripts in most places and assume freely that types of the form T, V^i, V^o, T^i and T^o are defined appropriately. For example, given $[\![C]\!]^{\mathbb{P}\, T}_{\mathcal{Z}_C}$, the Z-semantics for chart C, the predicate $x \star z' \in C$ is a well-typed abbreviation for:

$$x^{T^i} \star z^{T^{o\prime}} \in [\![[\![C]\!]_{Z_t}]\!]^{\mathbb{P}\, T}_{\mathcal{Z}_C}$$

where the binding concatenation operator $\star$ is as previously defined on page 172.

Finally, we define the schema operator Δ for schema types such that:

$$\Delta U =_{def} U \curlyvee U'$$

A.3 The precondition operator

The notion of the *precondition* of an operation schema U is formalised in a similar fashion to [20]. That is, the precondition of and operation schema

$U^{\mathbb{P}(T^i \Upsilon T^{o\prime})}$ is a set of bindings whose type contains (at least) the input observations of U, *i.e.* those observations in the type T^i. If we again consider the operation U as a relation, then the set representing the precondition captures all input values of type T^i for which the relation is defined. The operator is defined as follows.

Definition A.3.1 Given a schema $U^{\mathbb{P}\,T}$, we have,

$$Pre\ U\ x^{T_3} =_{def} \exists z \bullet x \star z' \dot{\in} U \qquad [T^i \preceq T_3]$$

Note that, as in [20], the operator *Pre* is generalised such that the set of bindings that represents the precondition contains both binding of type T^i and related bindings whose type is an extension of T^i. The type of the binding z is restricted only so that it is disjoint from the type T_3 of binding x (by application of $\star$). Binding z may even be a binding with an empty type, in which case it must hold that $T \preceq T_3$ and therefore that $x \dot{\in} U$. Like for the operators $\dot{\in}$ and $\doteq$ defined above, this generalisation is made to reduce excessive use of the filtering operator in the presentation of proofs.

The precondition of the Z-semantics of a μ-chart represents the set of all before-state, input event pairs for which a transition is defined in the given chart. While this is simple for the specific case of sequential charts, the precondition of charts that contain structure and interaction via signals becomes more cumbersome. We introduce the following rules for preconditions that follow directly from Definition A.3.1.

Proposition A.3.1 Given the arbitrary chart C and the binding x^{T_3}, we have,

$$\frac{Pre\ C\ x \quad x \star z' \dot{\in} C \vdash P}{P}\ (Pre^-)$$

$$\frac{x \star z' \dot{\in} C}{Pre\ C\ x}\ (Pre^+)$$

where $[\![C]\!]^{\mathbb{P}\,T}_{Z_t}$, $T^i \preceq T_3$, $T^o \preceq T_4$ and the usual conditions apply to z^{T_4} and P.

We give the following specialised rules that deal with the preconditions of composed charts.

Proposition A.3.2 Given arbitrary charts C_1, C_2 and $C = C_1 \mid \Psi \mid C_2$, and bindings $z_1^{U_1}$, $z_2^{U_2}$, $x^{V^{io}}$, we have,

$$\frac{\begin{array}{ll} & z_1 \star x_1 \star y_1' \star v_1' \dot{\in} C_1, \\ & z_2 \star x_2 \star y_2' \star v_2' \dot{\in} C_2, \\ Pre\ C\ (z_1 \star z_2 \star x) & x_1.i_{C_1} = (x.i_C \cup fb_v) \cap in_{C_1}, \\ & x_2.i_{C_2} = (x.i_C \cup fb_v) \cap in_{C_2}, \\ & v.o_C = v_1.o_{C_1} \cup v_2.o_{C_2} \vdash P \end{array}}{P}\ (Pre^-_{|-|})$$

$$\frac{\begin{array}{l} z_1 \star x_1 \star y_1' \star v_1' \dot{\in} C_1 \\ z_2 \star x_2 \star y_2' \star v_2' \dot{\in} C_2 \\ x_1.i_{C_1} = (x.i_C \cup fb_v) \cap in_{C_1} \\ x_2.i_{C_2} = (x.i_C \cup fb_v) \cap in_{C_2} \\ v.o_C = v_1.o_{C_1} \cup v_2.o_{C_2} \end{array}}{Pre\ C\ (z_1 \star z_2 \star x)}\ (Pre^+_{|-|})$$

where $fb_v =_{def} v.o_C \cap \Psi$, $[\![C]\!]^{\mathbb{P}\,T}_{z_C}$, $[\![C_1]\!]^{\mathbb{P}\,T_1}_{z_C}$, $[\![C_2]\!]^{\mathbb{P}\,T_2}_{z_C}$, and the usual conditions hold for $x_1^{V_1^{io}}$, $x_2^{V_2^{io}}$, $y_1^{U_1}$, $y_2^{U_2}$, $v_1^{V_1^{io}}$, $v_2^{V_2^{io}}$, $v^{V^{io}}$ and P.

Proof A.3.1

The proof for $(Pre^-_{|-|})$ is trivial using (Pre^-) and $(Z^-_{|-|})$. Similarly, $(Pre^+_{|-|})$ follows directly using $(Z^+_{|-|})$ and (Pre^+).

Also, rules $(Pre^-_{[|]})$ and $(Pre^+_{[|]})$ are used to reason about the preconditions of charts that use the interface operator.

Proposition A.3.3 Given arbitrary chart C, bindings $\vdash_{z_C}{}^{U_\Psi}$, $u^{V^{io}_\Psi}$, $x^{V^{io}}$, and signal set Ψ, for $C_\Psi =_{def} [C]_\Psi$ we have,

$$\frac{Pre\ C_\Psi \vdash_{z_C} \star\, u \quad x.i_C = u.i_{C_\Psi}}{Pre\ C \vdash_{z_C} \star\, x}\ (Pre^-_{[|]})$$

$$\frac{Pre\ C \vdash_{z_C} \star\, x \quad u.i_{C_\Psi} = x.i_C}{Pre\ C_\Psi \vdash_{z_C} \star\, u}\ (Pre^+_{[|]})$$

where $[\![C]\!]^{\mathbb{P}\,T}_{z_C}$, $[\![C_\Psi]\!]^{\mathbb{P}\,T_\Psi}_{z_C}$.

Proof A.3.2

For $(Pre^-_{[|]})$ we have,

$$\dfrac{Pre\ C_\Psi \vdash_{z_C} \star\, u \qquad \dfrac{\dfrac{}{\vdash_{z_C} \star\, u \star y'_\psi \star w' \dot{\in} C_\Psi}\,1 \qquad \begin{array}{c}\zeta_1 \\ \vdots \\ x.i_C = u.i_{C_\Psi} \cap in_C\end{array} \qquad \dfrac{}{\vdash_{z_C} \star\, x \star y'_\psi \star a' \dot{\in} C}\,2}{Pre\ C \vdash_{z_C} \star\, x}\ (Z^-_{[|]})(2)}{Pre\ C \vdash_{z_C} \star\, x}\ (Pre^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{in_{C_\Psi} = in_C}\,(df) \quad \dfrac{\dfrac{}{u.i_{C_\Psi} \subseteq in_{C_\Psi}}\,(V^{io}_{C_\Psi}\text{-}df) \quad x.i_C = u.i_{C_\Psi}}{x.i_C = u.i_{C_\Psi} \cap in_{C_\Psi}}}{x.i_C = u.i_{C_\Psi} \cap in_C}$$

For $(Pre^+_{||})$ we have,

$$\dfrac{Pre\ C \vdash z_c \star x \qquad \dfrac{\dfrac{}{\exists\, w^{V^{io}_\Psi} \bullet w.o_{C_\Psi} = a.o_C \cap out_{C_\Psi}}\,(V^{io}_\Psi\text{-}df) \qquad \dfrac{\vdots\ \zeta_1}{Pre\ C_\Psi \vdash z_c \star u \star y'_c \star w' \mathrel{\dot\in} C_\Psi}\,(Pre^+)}{Pre\ C_\Psi \vdash z_c \star u}\,(\exists^-)(2)}{Pre\ C_\Psi \vdash z_c \star u}\,(Pre^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{\vdash z_c \star x \star y'_c \star a' \in C}\,1 \qquad \dfrac{\dfrac{\dfrac{}{u.i_{C_\Psi} = x.i_C} \quad \dfrac{}{u.i_{C_\Psi} \subseteq in_{C_\Psi}}\,(V^{io}_\Psi\text{-}df)}{u.i_{C_\Psi} \cap in_{C_\Psi} = x.i_C} \quad \dfrac{}{in_{C_\Psi} = in_C}\,(df)}{u.i_{C_\Psi} \cap in_C = x.i_C} \quad \dfrac{}{w.o_{C_\Psi} = a.o_C \cap out_{C_\Psi}}\,2}{\vdash z_c \star u \star y'_c \star w' \mathrel{\dot\in} C_\Psi}\,(Z^+_{||})$$

Appendix B

Proofs

This section presents the proofs for the various propositions and lemmas presented throughout this thesis.

B.1 Proofs for Section 3.2: The Z transition model for sequential μ-Charts

In order to prove Proposition 3.2.3 we introduce and prove the following lemmas.

Lemma B.1.1 Given the arbitrary sequential chart $(C, \Sigma, \sigma, \Psi, \delta)$ and transition $t = (S_f, S_t, guard/action)$, such that $t \in \delta$. For all bindings z^{T_3} and x^{T_3} we have,

$$\frac{\rho(guard)[\alpha T^a / z.\alpha T^a] \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{\rho(guard)[\alpha T^a / x.\alpha T^a]}$$

where $[\![t]\!]^{\mathbb{P}\, T^a}_{Z_t}$, $T^a \preceq T_3$ and $T_i = T^a - [i_C : \mathbb{P}\, in_C]$

Recall that the function ρ (see Section 3.2) is defined recursively over the syntactic structure of transition guards. A transition guard (as defined in Section 2.1) is a list of signal expressions (either positive or negative) separated by the symbol &. Hence we prove this lemma by an induction over the number of signals in the guard.

When the guard is empty the function ρ returns the predicate *true*. Hence lemma B.1.1 holds trivially.

Given an arbitrary signal expression *sig_expr* and a well formed guard *sigs* containing zero or more signal expressions, assuming lemma B.1.1 holds for the guard *sigs*, then we prove it holds for the guard *sigs* & *sig_expr*. From the definition of ρ we know that $\rho(sigs \ \& \ sig\_expr) =_{def} \rho(sigs) \wedge$

$\rho(sig\_expr)$. Hence, using the induction hypothesis, the proof is reduce to showing lemma B.1.1 holds where $guard = sig\_expr$. We split this into two cases: one for a positive signal expression sig_expr; and one for a negative signal expression.

For a positive signal expression we have,

$$\dfrac{\dfrac{z.i_C \cup fb_z = x.i_C \cup fb_x \qquad \dfrac{\dfrac{\rho(sig\_expr)[\alpha T^a / z.\alpha T^a]}{(sig\_expr \in i_C \cup (o'_C \cap \Psi))[\alpha T^a / z.\alpha T^a]}\,(\rho\text{-}df)}{sig\_expr \in z.i_C \cup (z.o'_C \cap \Psi)}\,(\alpha\text{-}df)}{sig\_expr \in x.i_C \cup (x.o'_C \cap \Psi)}\,(fb_z, fb_x\text{-}df)}{\dfrac{(sig\_expr \in i_C \cup (o'_C \cap \Psi))[\alpha T^a / x.\alpha T^a]}{\rho(sig\_expr)[\alpha T^a / x.\alpha T^a]}\,(\rho\text{-}df)}\,(\alpha\text{-}df)$$

The case where the signal expression is negative can be proved in the same way where each occurrence of $\in$ is replaced with $\notin$.

Therefore, by induction, we have shown that lemma B.1.1 holds. Now using Lemma B.1.1 we can show the following lemma holds.

Lemma B.1.2 Given the arbitrary chart $(C, \Sigma, \sigma, \Psi, \delta)$, and bindings z^{T_3} and x^{T_3}, for all $t \in \delta$ we have,

$$\dfrac{Trans\ t\ z \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{Trans\ t\ x}$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $T^a \preceq T_3$ and $T_i = T^a - [i_C : \mathbb{P}\,\mu Signal]$

Proof B.1.2

$$\dfrac{\dfrac{\dfrac{\dfrac{Trans\ t\ z}{z.c_C = t.S_f \wedge z.c'_C = t.S_t \wedge z.o'_C = t.action}\,(Trans\text{-}df)(\wedge^-) \qquad x =_{T_i} z}{x.c_C = t.S_f \wedge x.c'_C = t.S_t \wedge x.o'_C = t.action}\,\zeta_1}{x.c_C = t.S_f \wedge \rho(t.guard)[\alpha T / x.\alpha T] \wedge x.c'_C = t.S_t \wedge x.o'_C = t.action}\,(\wedge^+)}{Trans\ t\ x}\,(Trans\text{-}df)$$

where ζ_1 is:

$$\dfrac{\dfrac{Trans\ t\ z}{\rho(t.guard)[\alpha T^a / z.\alpha T^a]}\,(Trans\text{-}df)(\wedge^-) \qquad z.i_C \cup fb_z = x.i_C \cup fb_x}{\rho(t.guard)[\alpha T^a / x.\alpha T^a]}\,(lem\ B.1.1)$$

We also introduce the following properties for dealing with the predicates act and $inact$.

Lemma B.1.3 Given the arbitrary sequential chart $(C, \Sigma, \sigma, \Psi, \delta)$, for arbitrary z and z_1 we have,

$$\frac{z =_{T_3} z_1 \quad actv\ C\ z}{actv\ C\ z_1}\ (act_I) \qquad \frac{z =_{T_3} z_1 \quad inactv\ C\ z}{inactv\ C\ z_1}\ (inact_I)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $\Delta U \preceq T_3$ and $T^{act} \preceq T_3$.

Proof B.1.3

$$\frac{\dfrac{actv\ C\ z}{C \in z.act}\ (df) \quad \dfrac{\dfrac{z =_{T_3} z_1}{z =_{T^{act}} z_1}\ (T^{act} \preceq T_3)}{z_1.act = z.act}}{\dfrac{C \in z_1.act}{actv\ C\ z_1}\ (df)} \qquad \frac{\dfrac{\dfrac{inactv\ C\ z}{\neg\ actv\ C\ z}\ (df)}{C \notin z.act} \quad \dfrac{\dfrac{z =_{T_3} z_1}{z =_{T^{act}} z_1}\ (T^{act} \preceq T_3)}{z_1.act = z.act}}{\dfrac{C \notin z_1.act}{\neg\ actv\ C\ z_1}\ (df)}}{inactv\ C\ z_1}\ (df)$$

Lemma B.1.4 Given the chart $(C, \Sigma, \sigma, \Psi, \delta)$, for arbitrary z^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad C \in z.act}{actv\ C\ z}\ (act_{II}) \qquad \frac{z \mathrel{\dot\in} \delta_C \quad C \notin z.act}{inactv\ C\ z}\ (inact_{II})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.1.4

Trivial from the definitions of $actv\ C\ z$ and $inactv\ C\ z$.

Lemma B.1.5 Given the chart $(C, \Sigma, \sigma, \Psi, \delta)$, for arbitrary z^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\ (act_{LEM})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.1.5

$$\frac{\dfrac{}{C \in z.act \vee C \notin z.act}\ (LEM) \quad \dfrac{\dfrac{\overline{C \in z.act}^{\,1}}{actv\ C\ z}\ (act_{II})}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^+) \quad \dfrac{\dfrac{\overline{C \notin z.act}^{\,1}}{inactv\ C\ z}\ (act_{II})}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^+)}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^-)(1)$$

Lemma B.1.6 Given the arbitrary sequential chart $(C, \Sigma, \sigma, \Psi, \delta)$, for arbitrary z we have,

$$\frac{actv\ C\ z}{C \in z.act}\ (act_{III}) \qquad \frac{inactv\ C\ z}{C \notin z.act}\ (inact_{III})$$

Proof B.1.6

Trivial from the definitions of $actv\ C\ z$ and $inactv\ C\ z$.

Proposition 3.2.1 Given the sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, for arbitrary binding z^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \quad actv\ C\ z \quad t \in \delta, Trans\ t\ z \vdash P}{P}\ (Z_t{}^-)$$

$$\frac{actv\ C\ z \quad t \in \delta \quad Trans\ t\ z}{z \dot{\in} \delta_C}\ (Z_t{}^+)$$

where $T^a \preceq T_3$ and assuming the usual conditions for t and P (due to the elimination of an existential quantifier).

Proof 3.2.1

For $(Z_t{}^-)$ we have,

$$\frac{\dfrac{\dfrac{z \dot{\in} \delta_C}{x \in \delta_C}\ (df)}{(C \notin x.act \wedge x \dot{\in} \Xi Chart_C \wedge x.o'_C = \{\}) \vee (C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x)}\ (df) \quad \begin{matrix}\zeta_1\\ \vdots \\ P\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots \\ P\end{matrix}}{P}\ (\vee^-)(1)$$

where $x =_{def} z \restriction T^a$.

ζ_1 is:

$$\frac{\dfrac{C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x}{\exists t \in \delta \bullet Trans\ t\ x}\ (\wedge^-) \quad \begin{matrix}\overline{t \in \delta}\ 2\\ \vdots \\ P\end{matrix}}{P}\ (\exists^-)(2) \qquad \begin{matrix}\overline{Trans\ t\ x}\ 2 \quad \overline{z =_{T^a} x}\ (df)\\ \overline{Trans\ t\ z}\ (Trans\text{-}df)\\ \vdots \\ P\end{matrix}\ 1$$

and ζ_2 is:

$$\frac{\dfrac{actv\ C\ z}{C \in z.act}\ (df) \quad \dfrac{\dfrac{\overline{C \notin x.act \wedge x \dot{\in} \Xi Chart_C \wedge x.o'_C = \{\}}\ 1}{C \notin x.act}\ (\wedge^-) \quad \dfrac{\overline{z =_{T^a} x}\ (df)}{z.act = x.act}}{C \notin z.act}}{\dfrac{\perp}{P}\ (\perp^-)}$$

For $(Z_t{}^+)$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{actv\ C\ z}{C \in z.act}\ (act_{III}) \quad \dfrac{\dfrac{t \in \delta \quad Trans\ t\ z}{t \in \delta \wedge Trans\ t\ z}\ (\wedge^+)}{\exists t \in \delta \bullet Trans\ t\ z}\ (\exists^+)}{C \in z.act \wedge \exists t \in \delta \bullet Trans\ t\ z}\ (\wedge^+)}{(C \notin z.act \wedge z \dot{\in} \Xi Chart_C \wedge z.o'_C = \{\}) \vee (C \in z.act \wedge \exists t \in \delta \bullet Trans\ t\ z)}\ (\vee^+)}{\dfrac{z \restriction T^a \in \delta_C}{z \dot{\in} \delta_C}\ (df)}$$

Proposition 3.2.2 Given the sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$, where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, for arbitrary binding z^{T_3} we have,

$$\dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z \dot{\in} \Xi Chart_C}\ (iact_I^-) \qquad \dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\ (iact_{II}^-)$$

$$\dfrac{inactv\ C\ z \quad z \dot{\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \dot{\in} \delta_C}\ (iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof 3.2.2

For $(iact_I^-)$ we have,

$$\dfrac{\dfrac{\dfrac{z \dot{\in} \delta_C}{x \in \delta_C}\ (df)}{(C \notin x.act \wedge x \dot{\in} \Xi Chart_C \wedge x.o'_C = \{\}) \vee (C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x)}\ (df) \quad \begin{matrix}\zeta_1 \\ \vdots \\ z \dot{\in} \Xi Chart_C\end{matrix} \quad \begin{matrix}\zeta_2 \\ \vdots \\ z \dot{\in} \Xi Chart_C\end{matrix}}{z \dot{\in} \Xi Chart_C}\ (\vee^-)(1)$$

where $x =_{def} z \restriction T^a$.

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{C \notin x.act \wedge x \dot{\in} \Xi Chart_C \wedge x.o'_C = \{\}}\ 1}{x \dot{\in} \Xi Chart_C}\ (\wedge^-) \quad \dfrac{\dfrac{}{x =_{T^a} z}\ (df)}{x =_{\Delta U} z}\ (\Delta U \preceq T^a)}{z \dot{\in} \Xi Chart_C}$$

and ζ_2 is:

$$\dfrac{\dfrac{inactv\ C\ z}{C \notin z.act}\ (df) \quad \dfrac{\dfrac{\dfrac{}{C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x}\ 1}{C \in x.act}\ (\wedge^-) \quad \dfrac{}{z =_{T^a} x}\ (df)}{z.act = x.act}}{C \in z.act}}{\dfrac{\bot}{z \dot{\in} \Xi Chart_C}\ (\bot^-)}$$

Similarly, for $(iact_{II}^{-})$ we have,

$$\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{(C \notin x.act \wedge x \mathrel{\dot\in} \Xi Chart_C \wedge x.o'_C = \{\}) \vee (C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x)}\,(df) \quad \begin{matrix}\zeta_1\\ \vdots \\ z.o'_C = \{\}\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots \\ z.o'_C = \{\}\end{matrix}}{z.o'_C = \{\}}\,(\vee^{-})(1)$$

where $x =_{def} z \upharpoonright T^a$.

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{C \notin x.act \wedge x \mathrel{\dot\in} \Xi Chart_C \wedge x.o'_C = \{\}}\,1}{x.o'_C = \{\}}\,(\wedge^{-}) \quad \dfrac{\overline{z =_{T^a} x}\,(df)}{z.o'_C = x.o'_C}}{z.o'_C = \{\}}$$

and ζ_2 is:

$$\dfrac{\dfrac{inactv\ C\ z}{C \notin z.act}\,(act_{III}) \quad \dfrac{\dfrac{\dfrac{}{C \in x.act \wedge \exists t \in \delta \bullet Trans\ t\ x}\,1}{C \in x.act}\,(\wedge^{-}) \quad \dfrac{\overline{z =_{T^a} x}\,(df)}{z.act = x.act}}{C \in z.act}}{\dfrac{\bot}{z.o'_C = \{\}}\,(\bot^{-})}$$

For $(iact^{+})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{inactv\ C\ z}{C \notin z.act}\,(act_{III}) \quad \dfrac{z \mathrel{\dot\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \mathrel{\dot\in} \Xi Chart_C \wedge z.o'_C = \{\}}\,(\wedge^{+})}{C \notin z.act \wedge z \mathrel{\dot\in} \Xi Chart_C \wedge z.o'_C = \{\}}\,(\wedge^{+})}{(C \notin z.act \wedge z \mathrel{\dot\in} \Xi Chart_C \wedge z.o'_C = \{\}) \vee (C \in z.act \wedge \exists t \in \delta \bullet Trans\ t\ z)}\,(\vee^{+})}{\dfrac{z \upharpoonright T^a \in \delta_C}{z \mathrel{\dot\in} \delta_C}}\,(df)$$

Proposition 3.2.3 For the arbitrary sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$ and bindings z^{T_3} and x^{T_3} we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \mathrel{\dot\in} \delta_C}\,(Z_t^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

Proof 3.2.3

$$\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\,(act_{LEM}) \quad \dfrac{z \mathrel{\dot\in} \delta_C \quad \overline{actv\ C\ z}\,1 \quad \begin{matrix}\zeta_1\\ \vdots \\ x \mathrel{\dot\in} \delta_C\end{matrix}}{x \mathrel{\dot\in} \delta_C}\,(Z_t^{-})(2) \quad \begin{matrix}\zeta_2\\ \vdots \\ x \mathrel{\dot\in} \delta_C\end{matrix}}{x \mathrel{\dot\in} \delta_C}\,(\vee^{-})(1)$$

ζ_1 is:

$$\dfrac{\begin{array}{c}\zeta_{1.1}\\ \vdots\end{array}\quad \dfrac{}{Trans\ t_1\ x}\quad \dfrac{}{t_1 \in \delta}\ 2 \qquad \dfrac{\dfrac{}{actv\ C\ z}\ 1 \quad x =_{T_i} z}{actv\ C\ x}\ (act_I)}{x \mathrel{\dot\in} \delta_C}\ (Z_t{}^+)$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{}{Trans\ t_1\ z}\ 2 \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{Trans\ t_1\ x}\ (lem\ B.1.2)$$

ζ_2 is:

$$\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C \quad \dfrac{}{inactv\ C\ z}\ 1}{z.o'_C = \{\}}\ (iact_{II}^-) \qquad \dfrac{x =_{T_i} z}{x =_{T^{out}} z}\ (T^{out} \preceq T_i) \quad \begin{array}{c}\zeta_{2.1}\\ \vdots\end{array} \quad \dfrac{\dfrac{}{inactv\ C\ z}\ 1 \quad x =_{T_i} z}{inactv\ C\ x}\ (inact_I)}{x.o'_C = \{\}}\ (iact^+)}{x \in \delta_C}$$

$\zeta_{2.1}$ is:

$$\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C \quad \dfrac{}{inactv\ C\ z}\ 1}{z \mathrel{\dot\in} \Xi Chart_C}\ (iact_I^-) \qquad \dfrac{x =_{T_i} z}{x =_{(\Delta U)} z}\ ((\Delta U \preceq T_i)}{x \mathrel{\dot\in} \Xi Chart_C}$$

B.2 Proofs for Section 3.3: Feedback of signals in μ-Charts

Lemma 3.3.1 Given $(C_1, \Sigma, \sigma, \Psi, \delta_1)$, where $\Sigma = \{A, B\}$, $\sigma = A$, $\Psi = \{a\}$ and $\delta_1 = \{(A, B, a/a)\}$, for arbitrary z^{T_3} and input $i \subseteq in_C$ we have,

$$\frac{actv\ C_1\ z \quad z \doteq \langle\!|\ c_{C_1} \Rrightarrow A, i_{C_1} \Rrightarrow i, c'_{C_1} \Rrightarrow B, o_{C_1}{}' \Rrightarrow \{a\}\ |\!\rangle}{z \dot{\in} \delta_{C_1}}$$

where $[\![\delta_{C_1}]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$ and $T^a \preceq T_3$

Proof 3.3.1

$$\frac{actv\ C_1\ z \quad \dfrac{}{(A, B, a/a) \in \delta_1}\ (df) \qquad \begin{array}{c}\zeta_1\\ \vdots\\ Trans\ (A, B, a/a)\ z\end{array}}{z \dot{\in} \delta_{C_1}}\ (Z_t{}^+)$$

where ζ_1 is:

$$\frac{\begin{array}{c}\zeta_1\\ \vdots\\ z \doteq \langle\!|\ c_{C_1} \Rrightarrow A, i_{C_1} \Rrightarrow i, c'_{C_1} \Rrightarrow B, o_{C_1}{}' \Rrightarrow \{a\}\ |\!\rangle\end{array}}{\dfrac{z.c_{C_1} = A \wedge z.c'_{C_1} = B \wedge z.o_{C_1}{}' = \{a\} \wedge \rho(a)[\alpha T^a / z.\alpha T^a]}{Trans\ (A, B, a/a)\ z}\ (Trans\text{-}df)}$$

where ζ_2 is:

$$\frac{\dfrac{}{a \in z.i_{C_1} \cup (\{a\} \cap \{a\})} \qquad \dfrac{z \doteq \langle\!|\ c_{C_1} \Rrightarrow A, i_{C_1} \Rrightarrow i, c'_{C_1} \Rrightarrow B, o_{C_1}{}' \Rrightarrow \{a\}\ |\!\rangle}{z.o_{C_1}{}' = \{a\}}}{\dfrac{a \in z.i_{C_1} \cup (z.o_{C_1}{}' \cap \{a\})}{\rho(a)[\alpha T^a / z.\alpha T^a]}}$$

Lemma 3.3.2 Given $(C_2, \Sigma, \sigma, \Psi, \delta_2)$, where $\Sigma = \{A, B\}$, $\sigma = A$, $\Psi = \{a\}$ and $\delta_2 = \{(A, B, -a/a)\}$, for all z^{T^a},

$$\frac{actv\ C_2\ z}{z \notin \delta_{C_2}}$$

where $[\![\delta_{C_2}]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$

Proof 3.3.2

$$\frac{\dfrac{\dfrac{}{z \in \delta_{C_2}}\ 1}{z \dot{\in} \delta_{C_2}}\ (df) \qquad actv\ C\ z \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{t \in \delta_2}\ 2}{t \in \{(A, B, -a/a)\}} \quad \dfrac{}{Trans\ t\ z}\ 2}{Trans\ (A, B, -a/a)\ z}\ (Trans\text{-}df)}{\dfrac{\rho(-a)[\alpha T^a / z.\alpha T^a]}{a \notin z.i_C \cup (\{a\} \cap \{a\})}\ (\rho\text{-}df)} \qquad \bot}{\bot}\ (Z_t{}^-)(2)}{z \notin \delta_{C_2}}\ (\bot^-)(1)$$

B.3 Proofs for Section 3.4: The composition operator

Before giving the proofs of the propositions of Section 3.4 we introduce and prove some useful lemmas.

Lemma B.3.1 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary z and z_1 we have,

$$\frac{z =_{T_3} z_1 \quad actv\ C\ z}{actv\ C\ z_1}\ (act_I) \qquad \frac{z =_{T_3} z_1 \quad inactv\ C\ z}{inactv\ C\ z_1}\ (inact_I)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $\Delta U \preceq T_3$ and $T^{act} \preceq T_3$.

Proof B.3.1

The base case for properties (act_I) and $(inact_I)$, that is, that they hold for any sequential chart, is given in Lemma B.1.3.[1]

The induction case for the left hand property is,

$$\frac{\dfrac{\dfrac{actv\ C\ z}{actv\ C_1\ z}\ (df) \quad z =_{T_3} z_1}{actv\ C_1\ z_1}\ (I.H.) \quad \dfrac{\dfrac{actv\ C\ z}{actv\ C_2\ z}\ (df) \quad z =_{T_3} z_1}{actv\ C_2\ z_1}\ (I.H.) \quad \begin{matrix}\zeta_1\\ \vdots \\ C \in z_1.act\end{matrix}}{actv\ C\ z_1}\ (df)$$

where ζ_1 is:

$$\frac{\dfrac{actv\ C\ z}{C \in z.act}\ (df) \quad \dfrac{z =_{T_3} z_1}{z.act = z_1.act}\ (T^{act} \preceq T_3)}{C \in z_1.act}$$

Similarly, for the right hand property we have,

$$\frac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\ (df) \quad z =_{T_3} z_1}{inactv\ C_1\ z_1}\ (I.H.) \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_2\ z}\ (df) \quad z =_{T_3} z_1}{inactv\ C_2\ z_1}\ (I.H.) \quad \begin{matrix}\zeta_2\\ \vdots \\ C \notin z_1.act\end{matrix}}{inactv\ C\ z_1}\ (df)$$

where ζ_1 is:

$$\frac{\dfrac{inactv\ C\ z}{C \notin z.act}\ (inact_{III}) \quad \dfrac{z =_{T^{act}} z_1}{z.act = z_1.act}}{C \notin z_1.act}$$

Lemma B.3.2 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary z^{T_3}, we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad C \in z.act}{actv\ C\ z}\ (act_{II}) \qquad \frac{z \mathrel{\dot\in} \delta_C \quad C \notin z.act}{inactv\ C\ z}\ (inact_{II})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

[1] Note that each use of the induction hypothesis $(I.H.)$ is well typed because, by definition, $\Delta U_1 \preceq \Delta U$ and $T_1^{act} \preceq T^{act}$. Therefore it follows from $\Delta U \preceq T_3$ and $T^{act} \preceq T_3$ that $\Delta U_1 \preceq T_3$ and $T_1^{act} \preceq T_3$. A similar argument holds for the types related to δ_{C_2}.

Proof B.3.2

The base case for the properties (act_{II}) and $(inact_{II})$ is given in Lemma B.1.4.

The induction case for the property on the left is,

$$\dfrac{\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{\exists\, o_1 \subseteq out_{C_1};\ o_2 \subseteq out_{C_2} \bullet \quad x.o'_C = o_1 \cup o_2 \wedge \quad x_1 \mathrel{\dot\in} \delta_{C_1} \wedge x_2 \mathrel{\dot\in} \delta_{C_2}}\,(df) \qquad \dfrac{\begin{array}{ccc}\vdots\,\zeta_1 & \vdots\,\zeta_2 & \dfrac{\overline{z =_{T^a} x}\,(df)}{z.act = x.act} \quad C \in z.act \\ actv\ C_1\ x & actv\ C_2\ x & C \in x.act\end{array}}{actv\ C\ x}\,(df)}{actv\ C\ x}\,(\exists^-)(1) \qquad \dfrac{\overline{z =_{T^a} x}\,(df)}{actv\ C\ z}\,(act_I)$$

where $x =_{def} z \restriction T^a$, $x_1 =_{def} x \star \langle\!|\, i_{C_1} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_1},\ o_{C_1}{}' \Rrightarrow o_1 \,|\!\rangle$ and $x_2 =_{def} x \star \langle\!|\, i_{C_2} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_2},\ o_{C_2}{}' \Rrightarrow o_2 \,|\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\overline{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \qquad \dfrac{\dfrac{C \in x.act \quad \dfrac{\overline{x =_{T^a} z}\,(df)}{x.act = z.act}}{C \in x.act} \qquad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{C_1 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_1 \in x.act}}{C_1 \in x_1.act} \qquad \dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\,1}{actv\ C_1\ x_1}\,(I.H.) \qquad \dfrac{\overline{x_1 =_{T^a} x}\,(df)}{actv\ C_1\ x}\,(act_I)$$

and ζ_2 is:

$$\dfrac{\dfrac{\dfrac{\overline{x_2 =_{T^a} x}\,(df)}{x_2.act = x.act} \qquad \dfrac{\dfrac{C \in z.act \quad \dfrac{\overline{x =_{T^a} z}\,(df)}{x.act = z.act}}{C \in x.act} \qquad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{C_2 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_2 \in x.act}}{C_2 \in x_2.act} \qquad \dfrac{}{x_2 \mathrel{\dot\in} \delta_{C_2}}\,1}{actv\ C_2\ x_2}\,(I.H.) \qquad \dfrac{\overline{x_2 =_{T^{act}} x}\,(df)}{actv\ C_2\ x}\,(act_I)$$

Similarly, for the right hand property, assuming the same definitions for x, x_1 and x_2, we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{\exists\, o_1 \subseteq out_{C_1};\ o_2 \subseteq out_{C_2} \bullet \quad x.o'_C = o_1 \cup o_2 \wedge \quad x_1 \mathrel{\dot\in} \delta_{C_1} \wedge x_2 \mathrel{\dot\in} \delta_{C_2}}\,(df) \qquad \dfrac{\begin{array}{ccc}\vdots\,\zeta_1 & \vdots\,\zeta_2 & \\ inactv\ C_1\ x & inactv\ C_2\ x & C \notin x.act\end{array}}{inactv\ C\ x}\,(df)}{inactv\ C\ x}\,(\exists^-)(1) \qquad \dfrac{\overline{x =_{T^{act}} z}\,(df)}{inactv\ C\ z}\,(inact_I)$$

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \qquad \dfrac{\dfrac{C \notin z.act \quad \dfrac{\dfrac{}{z =_{T^a} x}\,(df)}{z.act = x.act}}{C \notin x.act} \qquad \dfrac{\dfrac{z \dot{\in} \delta_C}{x \in \delta_C}\,(df)}{C_1 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_1 \notin x.act}}{C_1 \notin x_1.act} \qquad \dfrac{}{x_1 \dot{\in} \delta_{C_1}}\,1}{inactv\ C_1\ x_1}\,(I.H.) \qquad \dfrac{}{x_1 =_{T^a} x}\,(df)}{inactv\ C_1\ x}\,(inact_I)$$

and ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{x_2 =_{T^a} x}\,(df)}{x_2.act = x.act} \qquad \dfrac{\dfrac{C \notin z.act \quad \dfrac{\dfrac{}{z =_{T^a} x}\,(df)}{z.act = x.act}}{C \notin x.act} \qquad \dfrac{\dfrac{z \dot{\in} \delta_C}{x \in \delta_C}\,(df)}{C_2 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_2 \notin x.act}}{C_2 \notin x_2.act} \qquad \dfrac{}{x_2 \dot{\in} \delta_{C_2}}\,1}{inactv\ C_2\ x_2}\,(I.H.) \qquad \dfrac{}{x_2 =_{T^a} x}\,(df)}{inactv\ C_2\ x}\,(inact_I)$$

Lemma B.3.3 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary z^{T_3}, we have,

$$\dfrac{z \dot{\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\,(act_{LEM})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.3.3

$$\dfrac{\dfrac{}{C \in z.act \vee C \notin z.act}\,(LEM) \qquad \dfrac{\dfrac{z \dot{\in} \delta_C \quad \dfrac{}{C \in z.act}\,1}{actv\ C\ z}\,(act_{II})}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^+) \qquad \dfrac{\dfrac{z \dot{\in} \delta_C \quad \dfrac{}{C \notin z.act}\,1}{inactv\ C\ z}\,(inact_{II})}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^+)}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^-)(1)$$

Lemma B.3.4 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary binding z^{T_3} we have,

$$\dfrac{actv\ C\ z}{C \in z.act}\,(act_{III}) \qquad \dfrac{inactv\ C\ z}{C \notin z.act}\,(inact_{III})$$

Proof B.3.4

Trivial from the respective definitions of *act* and *inact*.

Proposition 3.4.1 Given $C = C_1 \mid \Psi \mid C_2$, for the binding z^{T_3} and arbitrary sets o_3 and o_4, we have,

$$\frac{\begin{array}{l} z.o'_C = o_1 \cup o_2, \\ z \star \{\!|\, i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 \,|\!\} \mathrel{\dot\in} \delta_{C_1}, \\ z \star \{\!|\, i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_2 \,|\!\} \mathrel{\dot\in} \delta_{C_2} \vdash Q \end{array}}{Q}\ (|\_|^-)$$

$$\frac{\begin{array}{l} z.o'_C = o_3 \cup o_4 \\ z \star \{\!|\, i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_3 \,|\!\} \mathrel{\dot\in} \delta_{C_1} \\ z \star \{\!|\, i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_4 \,|\!\} \mathrel{\dot\in} \delta_{C_2} \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \mathrel{\dot\in} \delta_C}\ (|\_|^+)$$

where the usual conditions hold for o_1, o_2 and Q, $[\![\delta_C]\!]^{T_a}_{z_C}$ and $T^a \preceq T_3$.

Proof 3.4.1

For $(|\_|^-)$ we have,

$$\frac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\ (df)}{\exists\, o_1, o_2 \bullet x.o'_C = o_1 \cup o_2 \wedge x_1 \mathrel{\dot\in} \delta_{C_1} \wedge x_2 \mathrel{\dot\in} \delta_{C_2}}\ (df) \qquad \begin{array}{c}\zeta_1\ \ \zeta_2\ \ \zeta_3 \\ \vdots\ \ \vdots\ \ \vdots \\ Q\end{array}}{Q}\ (\exists^-)(1)$$

ζ_1 is:

$$\frac{\dfrac{}{x.o'_C = o_1 \cup o_2}\ 1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z.o'_C = x.o'_C}}{z.o'_C = o_1 \cup o_2}$$

ζ_2 is:

$$\frac{\dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\ 1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z_1 =_{T^a_1} x_1}\ (df)}{z_1 \mathrel{\dot\in} \delta_{C_1}}$$

ζ_3 is:

$$\frac{\dfrac{}{x_2 \mathrel{\dot\in} \delta_{C_2}}\ 1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z_2 =_{T^a_2} x_2}\ (df)}{z_2 \mathrel{\dot\in} \delta_{C_2}}$$

where $x =_{def} z \restriction T^a$, $z_n =_{def} z \star \{\!|\, i_{C_n} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_n}, o_{C_n}{}' \Rrightarrow o_n \,|\!\}$. and $x_n =_{def} x \star \{\!|\, i_{C_n} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_n}, o_{C_n}{}' \Rrightarrow o_n \,|\!\}$ for $n = 1, 2$.

For $(|\_|^+)$ we have,

$$\frac{\dfrac{\dfrac{\begin{array}{c}\zeta_1\ \ \zeta_2 \\ \vdots\ \ \vdots \\ actv\ C\ z \vee inactv\ C\ z\end{array}}{\begin{array}{l} C_1 \in z.act \Leftrightarrow C \in z.act \wedge \\ C_2 \in z.act \Leftrightarrow C \in z.act \end{array}}\ (\vee^-)(1) \qquad \dfrac{z.o'_C = o_3 \cup o_4 \quad z_1 \mathrel{\dot\in} \delta_{C_1} \quad z_2 \mathrel{\dot\in} \delta_{C_2}}{\exists\, o_3, o_4 \bullet z.o'_C = o_3 \cup o_4 \wedge z_1 \mathrel{\dot\in} \delta_{C_1} \wedge z_2 \mathrel{\dot\in} \delta_{C_2}}\ (\exists^+)}{z \restriction T^a \in \delta_C}\ (df)}{z \mathrel{\dot\in} \delta_C}\ (df)$$

where $z_1 =_{def} z \star \langle\!| \ i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}' \Rrightarrow o_3 \ |\!\rangle$ and
$z_2 =_{def} z \star \langle\!| \ i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}' \Rrightarrow o_4 \ |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{actv\ C\ z}\,1}{actv\ C_1\ z}\,(df)}{C_1 \in z.act}\,(act_{III}) \quad \dfrac{\dfrac{\dfrac{}{actv\ C\ z}\,1}{actv\ C_2\ z}\,(df)}{C_2 \in z.act}\,(act_{III}) \quad \dfrac{\dfrac{}{actv\ C\ z}\,1}{C \in z.act}\,(df)}{C_1 \in z.act \wedge C_2 \in z.act \wedge C \in z.act}\,(\wedge^+)}{(C_1 \in z.act \wedge C_2 \in z.act \wedge C \in z.act) \vee (C_1 \notin z.act \wedge C_2 \notin z.act \wedge C \notin z.act)}\,(\vee^+)}{C_1 \in z.act \Leftrightarrow C \in z.act \wedge C_2 \in z.act \Leftrightarrow C \in z.act}\,(\textit{prop logic})$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{inactv\ C\ z}\,1}{inactv\ C_1\ z}\,(df)}{C_1 \notin z.act}\,(inact_{III}) \quad \dfrac{\dfrac{\dfrac{}{inactv\ C\ z}\,1}{inactv\ C_2\ z}\,(df)}{C_2 \notin z.act}\,(inact_{III}) \quad \dfrac{\dfrac{}{inactv\ C\ z}\,1}{C \notin z.act}\,(df)}{C_1 \notin z.act \wedge C_2 \notin z.act \wedge C \notin z.act}\,(\wedge^+)}{(C_1 \in z.act \wedge C_2 \in z.act \wedge C \in z.act) \vee (C_1 \notin z.act \wedge C_2 \notin z.act \wedge C \notin z.act)}\,(\vee^+)}{C_1 \in z.act \Leftrightarrow C \in z.act \wedge C_2 \in z.act \Leftrightarrow C \in z.act}\,(\textit{prop logic})$$

Proposition 3.4.2 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary binding z^{T_3}, we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z \mathrel{\dot\in} \Xi Chart_C}\,(iact_I^-) \qquad \dfrac{z \mathrel{\dot\in} \delta_C \quad inactv\ C\ z}{z.o_C' = \{\}}\,(iact_{II}^-)$$

$$\dfrac{inactv\ C\ z \quad z \mathrel{\dot\in} \Xi Chart_C \quad z.o_C' = \{\}}{z \mathrel{\dot\in} \delta_C}\,(iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof 3.4.2

For $(iact_I^-)$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z_1 \mathrel{\dot\in} \delta_{C_1}}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\,(df) \quad \dfrac{}{z_1 =_{T_3} z}\,(df)}{inactv\ C_1\ z_1}\,(inact_I)}{z_1 \mathrel{\dot\in} \Xi Chart_{C_1}}\,(I.H.) \quad \dfrac{\dfrac{}{z_1 =_{T_3} z}\,(df)}{z_1 =_{(\Delta U_1)} z}\,((\Delta U_1) \preceq T_3)}{z \mathrel{\dot\in} \Xi Chart_{C_1}} \quad \begin{matrix}\zeta_1\\ \vdots \\ z \mathrel{\dot\in} \Xi Chart_{C_2}\end{matrix}\,(S_\wedge^+)}{z \mathrel{\dot\in} (\Xi Chart_{C_1} \wedge \Xi Chart_{C_2})}\,(df)}{z \mathrel{\dot\in} \Xi Chart_C}\,(|\_|^-)(1)$$

where $z_1 =_{def} z \star \langle\!| \ i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}' \Rrightarrow o_1 \ |\!\rangle$ and
$z_2 =_{def} z \star \langle\!| \ i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}' \Rrightarrow o_2 \ |\!\rangle$.

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{z_2 \dot{\in} \delta_{C_2}}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_2\ z}\,(df) \quad \dfrac{}{z_2 =_{T_3} z}\,(df)}{inactv\ C_2\ z_2}\,(inact_I)}{z_2 \dot{\in} \Xi Chart_{C_2}}\,(I.H.) \quad \dfrac{\dfrac{}{z_2 =_{T_3} z}\,(df)}{z_2 =_{(\Delta U_2)} z}\,((\Delta U_2) \preceq T_3)}{z \dot{\in} \Xi Chart_{C_2}}$$

For $(iact_{II}^{-})$ we have,

$$\dfrac{z \dot{\in} \delta_C \quad \dfrac{\dfrac{}{z.o'_C = o_1 \cup o_2}\,1}{z.o'_C = z_1.o_{C_1}{}' \cup z_2.o_{C_2}{}'}\,(df) \quad \overset{\zeta_1}{\vdots}\ z_1.o_{C_1}{}' = \{\} \quad \overset{\zeta_2}{\vdots}\ z_2.o_{C_2}{}' = \{\}}{z.o'_C = \{\}}\,(|\_|^{-})(1)$$

where $z_1 =_{def} z \star \langle\!| i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 |\!\rangle$ and
$z_2 =_{def} z \star \langle\!| i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_2 |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{}{z_1 \dot{\in} \delta_{C_1}}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\,(df) \quad \dfrac{}{z_1 =_{T_3} z}\,(df)}{inactv\ C_1\ z_1}\,(inact_I)}{z_1.o_{C_1}{}' = \{\}}\,(I.H.)$$

ζ_2 is:

$$\dfrac{\dfrac{}{z_2 \dot{\in} \delta_{C_2}}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_2\ z}\,(df) \quad \dfrac{}{z_2 =_{T_3} z}\,(df)}{inactv\ C_2\ z_2}\,(inact_I)}{z_2.o_{C_2}{}' = \{\}}\,(I.H.)$$

And for $(iact^{+})$ we have,

$$\dfrac{\dfrac{z.o'_C = \{\}}{z.o'_C = \{\} \cup \{\}} \quad \overset{\zeta_1}{\vdots}\ z_1 \dot{\in} \delta_{C_1} \quad \overset{\zeta_2}{\vdots}\ z_2 \dot{\in} \delta_{C_2} \quad \dfrac{inactv\ C\ z}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^{+})}{z \dot{\in} \delta_C}\,(|\_|^{+})$$

where $z_1 =_{def} z \star \langle\!| i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow \{\} |\!\rangle$ and
$z_2 =_{def} z \star \langle\!| i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow \{\} |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{z \dot{\in} \Xi Chart_C}{z_1 \dot{\in} \Xi Chart_{C_1}}\,((\Delta U_1) \preceq (\Delta U)) \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\,(df) \quad \dfrac{}{z =_{T_3} z_1}\,(df)}{inactv\ C_1\ z_1}\,(inact_I) \quad \dfrac{}{z_1.o_{C_1}{}' = \{\}}\,(df)}{z_1 \dot{\in} \delta_{C_1}}\,(I.H.)$$

and ζ_2 is:

$$\dfrac{\dfrac{z \dot{\in} \Xi Chart_C}{z_2 \dot{\in} \Xi Chart_{C_2}}\,((\Delta U_2) \preceq (\Delta U)) \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_2\ z}\,(df) \quad \dfrac{}{z =_{T_3} z_2}\,(df)}{inactv\ C_2\ z_2}\,(inact_I) \quad \dfrac{}{z_2.o_{C_2}{}' = \{\}}\,(df)}{z_2 \dot{\in} \delta_{C_2}}\,(I.H.)$$

where $[\![\delta_{C_1}]\!]^{\mathbb{P}\,T_1^a}_{\mathcal{Z}_C}$, $[\![\delta_{C_2}]\!]^{\mathbb{P}\,T_2^a}_{\mathcal{Z}_C}$.

Proposition 3.4.3

$$\frac{}{[\![\delta_{C_1|\Psi|C_2}]\!]_{z_C} = [\![\delta_{C_2|\Psi|C_1}]\!]_{z_C}}\ (|\_|_{sym})$$

Proof 3.4.3

Trivial given the bindings in the respective sets $[\![\delta_{C_1|\Psi|C_2}]\!]_{z_C}$ and $[\![\delta_{C_2|\Psi|C_1}]\!]_{z_C}$ are unordered.

Proposition 3.4.4 Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary bindings z^{T_3} and x^{T_3} we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \mathrel{\dot\in} \delta_C}\ (Z_t^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

Proof 3.4.4

The base case for this structural induction, that $(Z_t^{\in})$ holds for sequential charts, is given in Proposition 3.2.3.

For the inductive case we have,

$$\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C \qquad \dfrac{\dfrac{}{z.o_C' = o_1 \cup o_2}\,1 \quad \dfrac{x =_{T_i} z}{x.o_C' = z.o_C'}}{x.o_C' = o_1 \cup o_2} \qquad \begin{matrix}\zeta_1\\ \vdots \\ x_1 \mathrel{\dot\in} \delta_{C_1}\end{matrix} \qquad \begin{matrix}\zeta_2\\ \vdots \\ x_2 \mathrel{\dot\in} \delta_{C_2}\end{matrix} \qquad \begin{matrix}\zeta_3\\ \vdots \\ actv\ C\ x \vee inactv\ C\ x\end{matrix}}{x \mathrel{\dot\in} \delta_C}\ (|\_|^{+})}{x \mathrel{\dot\in} \delta_C}\ (|\_|^{-})(1)$$

ζ_1 is:

$$\dfrac{\dfrac{}{z_1 \mathrel{\dot\in} \delta_{C_1}}\,1 \qquad \dfrac{\dfrac{x =_{T_i} z \qquad \dfrac{}{z_1.o_{C_1}' = x_1.o_{C_1}'}\,(df)}{x_1 =_{(T_i \curlyvee T_1^{out})} z_1} \quad (T_{1_i} \preceq (T_i \curlyvee T_1^{out}))}{x_1 =_{T_{1_i}} z_1} \qquad \begin{matrix}\zeta_{1.1}\\ \vdots \\ z_1.i_{C_1} \cup fb_{z_1} = \\ x_1.i_{C_1} \cup fb_{x_1}\end{matrix}}{x_1 \mathrel{\dot\in} \delta_{C_1}}\ (I.H.)$$

$\zeta_{1.1}$ is,

$$\dfrac{\dfrac{\dfrac{z.i_C \cup fb_z = x.i_C \cup fb_x}{(z.i_C \cup fb_z) \cap in_{C_1} = (x.i_C \cup fb_x) \cap in_{C_1}}\ (df) \qquad \dfrac{\dfrac{}{z_1.o_{C_1}' = x_1.o_{C_1}'}\,(df)}{z_1.o_{C_1}' \cap \Psi = x_1.o_{C_1}' \cap \Psi}}{z_1.i_{C_1} = x_1.i_{C_1}} \qquad \dfrac{}{fb_{z_1} = fb_{x_1}}}{z_1.i_{C_1} \cup fb_{z_1} = x_1.i_{C_1} \cup fb_{x_1}}$$

where $x_1 =_{def} x \star \langle\!| \ i_{C_1} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 \ |\!\rangle$, $[\![\delta_{C_1}]\!]^{\mathbb{P}\, T_1^a}_{Z_C}$, $T_{1_i} = T_1^a - T_1^{in}$ and $z_1 =_{def} z \star \langle\!| \ i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1 \ |\!\rangle$.
ζ_2 is:

$$\dfrac{\dfrac{}{z_2 \in \delta_{C_2}}\,1 \qquad \dfrac{\dfrac{x =_{T_i} z \quad \dfrac{}{z_2.o_{C_2}{}' = x_2.o_{C_2}{}'}\,(df)}{x_2 =_{(T_i \curlyvee T_2^{out})} z_2}}{x_2 =_{T_{2_i}} z_2}\,(T_{2_i} \preceq (T_i \curlyvee T_2^{out})) \qquad \begin{array}{c}\zeta_{2.1}\\ \vdots \\ z_2.i_{C_2} \cup fb_{z_2} = \\ x_2.i_{C_2} \cup fb_{x_2}\end{array}}{x_2 \mathrel{\dot\in} \delta_{C_2}}\,(I.H.)$$

$\zeta_{2.1}$ is,

$$\dfrac{\dfrac{\dfrac{z.i_C \cup fb_z = x.i_C \cup fb_x}{(z.i_C \cup fb_z) \cap in_{C_2} = (x.i_C \cup fb_x) \cap in_{C_2}}\,(df) \qquad \dfrac{\dfrac{}{z_2.o_{C_2}{}' = x_2.o_{C_2}{}'}\,(df)}{z_2.o_{C_2}{}' \cap \Psi = x_2.o_{C_2}{}' \cap \Psi}}{z_2.i_{C_2} = x_2.i_{C_2} \qquad fb_{z_2} = fb_{x_2}}}{z_2.i_{C_2} \cup fb_{z_2} = x_2.i_{C_2} \cup fb_{x_2}}$$

where $x_2 =_{def} x \star \langle\!| \ i_{C_2} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_2 \ |\!\rangle$, $[\![\delta_{C_2}]\!]^{\mathbb{P}\, T_2^a}_{Z_C}$, $T_{2_i} = T_2^a - T_2^{in}$ and $z_2 =_{def} z \star \langle\!| \ i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_2 \ |\!\rangle$.
ζ_3 is:

$$\dfrac{\dfrac{}{actv\ C\ z \vee inactv\ C\ z}\,1 \qquad \dfrac{\dfrac{\dfrac{}{actv\ C\ z}\,2 \quad x =_{T_i} z}{actv\ C\ x}\,(act_I)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^+) \qquad \dfrac{\dfrac{\dfrac{}{inactv\ C\ z}\,2 \quad x =_{T_i} z}{inactv\ C\ x}\,(inact_I)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^+)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^-)(2)$$

Lemma 3.4.5 Given $C = C_1 \mid \{b\} \mid C_2$, where C_1 and C_2 are the sequential charts of Figure 3.5, for arbitrary z^{T^a} we have,

$$\frac{actv\ C\ z}{z \notin \delta_C}$$

where $[\![\delta_C]\!]^{\mathbb{P}\, T^a}_{Z_C}$

Proof 3.4.5

Assuming z_1 and z_2 are defined as expected, we have,

$$\dfrac{\dfrac{\dfrac{}{z \in \delta_C}\,1}{z \mathrel{\dot\in} \delta_C} \qquad \dfrac{\dfrac{}{z_1 \mathrel{\dot\in} \delta_{C_1}}\,2 \qquad \dfrac{\dfrac{actv\ C\ z}{actv\ C_1\ z}\,(df) \qquad \dfrac{}{z_1 =_{T^a} z}\,(df)}{actv\ C_1\ z_1}\,(act_I) \qquad \begin{array}{c}\zeta_1\\ \vdots \\ \bot\end{array}}{\bot}\,(Z_t{}^-)(3)}{\dfrac{\bot}{z \notin \delta_C}\,(\bot^-)(1)}\,(|\_|^-)(2)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{t_1 \in \{(A, B, a/b)\}}\,3 \qquad \dfrac{}{Trans\ t_1\ z_1}\,3}{Trans\ (A, B, a/b)\ z_1}}{z_1.o_{C_1}{}' = \{b\}}\,(\rho\text{-}df) \qquad \begin{array}{c}\zeta_2\\ \vdots \\ b \notin z_1.o_{C_1}{}'\end{array}$$
$$\bot$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{z_2 \mathrel{\dot\in} \delta_{C_2}}\,2 \quad \dfrac{\dfrac{actv\ C\ z}{actv\ C_2\ z}\,(df) \quad \dfrac{}{z_1 =_{T^a} z}\,(df)}{actv\ C_2\ z_2}\,(act_I)}{b \notin z_1.o_{C_1}'}\,(Z_t^{-})(4) \quad \dfrac{\begin{array}{c}\zeta_3 \\ \vdots \\ b \notin z.o_C' \end{array} \quad \dfrac{}{z.o_C' = o_1 \cup o_2}\,2}{\dfrac{b \notin o_1}{b \notin z_1.o_{C_1}'}\,(df)}}{}$$

and ζ_3 is:

$$\dfrac{\dfrac{\dfrac{}{z_2.i_{C_2} = (z.i_C \cup (z.o_C' \cap \Psi)) \cap in_{C_2}}\,(df)}{z_2.i_{C_2} = (z.i_C \cup (z.o_C' \cap \{b\})) \cap \{b\}} \quad \dfrac{\dfrac{\dfrac{}{t_2 \in \{(C, D, -b/c)\}}\,4 \quad \dfrac{}{Trans\ t_2\ z_2}\,4}{Trans\ (C, D, -b/c)\ z_2}}{\dfrac{b \notin z_2.i_{C_2} \cup (z_2.o_{C_2}' \cap \{\})}{b \notin z_2.i_{C_2}}\,(\rho\text{-}df)}}{\dfrac{b \notin (z.i_C \cup (z.o_C' \cap \{b\})) \cap \{b\}}{b \notin z.o_C'}}$$

B.4 Proofs for Section 3.5: The decomposition operator

Before giving the proofs of the propositions of Section 3.5 we show that the properties introduced in lemmas B.3.1 to B.3.4 of the previous section also hold for decomposed charts.

Lemma B.4.1 Given $C = Dec\ (\omega\ C_1)$ *by* $\{(C_2, \omega\ C_2)\}$, for arbitrary $C_2, \omega\ C_1 = (C_1, \Sigma, \sigma, \Psi, \delta)$, and bindings z and z_1, we have,

$$\frac{z =_{T_3} z_1 \quad actv\ C\ z}{actv\ C\ z_1}\ (act_I) \qquad \frac{z =_{T_3} z_1 \quad inactv\ C\ z}{inactv\ C\ z_1}\ (inact_I)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $\Delta U \preceq T_3$ and $T^{act} \preceq T_3$.

Proof B.4.1

Lemma B.1.3 shows that (act_I) and $(inact_I)$ hold for sequential charts.[2]
The induction case for the left hand property is,

$$\frac{\dfrac{\dfrac{actv\ C\ z}{actv\ C_1\ z}\ (df) \quad z =_{T_3} z_1}{actv\ C_1\ z_1}\ (I.H.) \quad \dfrac{\dfrac{actv\ C\ z}{C \in z.act}\ (df) \quad \dfrac{z =_{T_3} z_1}{z.act = z_1.act}\ (T^{act} \preceq T_3) \quad \zeta_1}{C \in z_1.act}}{actv\ C\ z_1}\ (df)$$

where ζ_1 is:

$$\frac{\dfrac{\dfrac{\dfrac{actv\ C\ z}{actv\ C_2\ z \Leftrightarrow (z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2)}\ (df) \quad \dfrac{\overline{actv\ C_2\ z_1}\ ^1 \quad z =_{T_3} z_1}{actv\ C_2\ z}\ (I.H.)}{z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2} \quad \zeta_{1.1} \quad \zeta_{1.2}}{z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2}\ (\vee^-)(2) \quad \zeta_2}{actv\ C_2\ z_1 \Rightarrow (z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2)}\ (\Rightarrow^+)(1)}{actv\ C_2\ z_1 \Leftrightarrow (z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2)}$$

$\zeta_{1.1}$ is:

$$\frac{\dfrac{\overline{z.c_{C_1} = C_2}\ ^2 \quad \dfrac{z =_{T_3} z_1}{z.c_{C_1} = z_1.c_{C_1}}\ (\Delta U \preceq T_3)}{z_1.c_{C_1} = C_2}}{z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2}\ (\vee^+)$$

$\zeta_{1.2}$ is:

$$\frac{\dfrac{\overline{z.c'_{C_1} = C_2}\ ^2 \quad \dfrac{z =_{T_3} z_1}{z.c'_{C_1} = z_1.c'_{C_1}}\ (\Delta U \preceq T_3)}{z_1.c'_{C_1} = C_2}}{z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2}\ (\vee^+)$$

[2] As with the proof of Lemma B.3.1 each use of the induction hypothesis $(I.H.)$ is well typed.

ζ_2 is:

$$\dfrac{\begin{array}{c}\zeta_{2.1}\\ \vdots\\ z.c_{C_1} = C_2 \vee \\ z.c'_{C_1} = C_2\end{array} \qquad \dfrac{actv\ C\ z}{actv\ C_2\ z \Leftrightarrow (z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2)}\ (df)}{\dfrac{actv\ C_2\ z \qquad z_2 =_{T_3} z}{\dfrac{actv\ C_2\ z_1}{(z_1.c_{C_1} = C_2 \vee z_1.c'_{C_1} = C_2) \Rightarrow actv\ C_2\ z_1}\ (\Rightarrow^+)(2)}\ (I.H.)}$$

$\zeta_{2.1}$ is:

$$\dfrac{\dfrac{}{\begin{array}{l}z_1.c_{C_1} = C_2 \vee \\ z_1.c'_{C_1} = C_2\end{array}}\ 2 \qquad \dfrac{\dfrac{\dfrac{}{z_1.c_{C_1} = C_2}\ 3 \qquad \dfrac{z =_{T_3} z_1}{z.c_{C_1} = z_1.c_{C_1}}\ ((\Delta U \curlyvee T^{act}) \preceq T_3)}{z.c_{C_1} = C_2}}{z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2}\ (\vee^+) \qquad \begin{array}{c}\zeta_{2.2}\\ \vdots\end{array}}{z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2}\ (\vee^-)(3)$$

$\zeta_{2.2}$ is:

$$\dfrac{\dfrac{\dfrac{}{z_1.c'_{C_1} = C_2}\ 3 \qquad \dfrac{z =_{T_3} z_1}{z.c'_{C_1} = z_1.c'_{C_1}}\ (\Delta U \preceq T_3)}{z.c'_{C_1} = C_2}}{z.c_{C_1} = C_2 \vee z.c'_{C_1} = C_2}\ (\vee^+)$$

For the right hand property we have,

$$\dfrac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\ (df) \qquad z =_{T_3} z_1}{inactv\ C_1\ z_1}\ (I.H.) \qquad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_2\ z}\ (df) \qquad z =_{T_3} z_1}{inactv\ C_2\ z_1}\ (I.H.) \qquad \begin{array}{c}\zeta_1\\ \vdots\\ C \notin z_1.act\end{array}}{inactv\ C\ z_1}\ (df)$$

where ζ_1 is:

$$\dfrac{\dfrac{inactv\ C\ z}{C \notin z.act}\ (inact_{III}) \qquad \dfrac{z =_{T_3} z_1}{z.act = z_1.act}\ (T^{act} \preceq T_3)}{C \notin z_1.act}$$

Lemma B.4.2 Given $C = Dec\ (\omega\ C_1)$ *by* $\{(C_2, \omega\ C_2)\}$, for arbitrary $C_2, \omega\ C_1 = (C_1, \Sigma, \sigma, \Psi, \delta)$, and binding x^{T_3}, we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad C \in z.act}{actv\ C\ z}\ (act_{II}) \qquad \dfrac{z \mathrel{\dot\in} \delta_C \quad C \notin z.act}{inactv\ C\ z}\ (inact_{II})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.4.2

Again the base case for both properties follows trivially from the definition of $actv\ C_1\ z$ and $inactv\ C_1\ z$ for any sequential chart C_1.

The induction case for the left hand property is,

$$\dfrac{\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{\exists\, o_1 \subseteq out_{C_1};\ o_2 \subseteq out_{C_2} \bullet \\ x.o'_C = o_1 \cup o_2 \wedge \\ x_1 \mathrel{\dot\in} \delta_{C_1} \wedge x_2 \mathrel{\dot\in} \delta_{C_2}}\,(\exists^-)(1) \qquad \dfrac{\begin{array}{c}\zeta_1 \\ \vdots \\ actv\ C_1\ x\end{array} \qquad \begin{array}{c}\zeta_2 \\ \vdots \\ C \in x.act\end{array}}{actv\ C\ x}\,(df)}{actv\ C\ x} \qquad \dfrac{}{z =_{T^a} x}\,(df)}{actv\ C\ z}\,(act_I)$$

where $x =_{def} z \restriction T^a$, $x_1 =_{def} x \star \langle\!|\ i_{C_1} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle$ and $x_2 =_{def} x \star \langle\!|\ i_{C_2} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_2}, o_{C_2}{}' \Rrightarrow o_2\ |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \qquad \dfrac{\dfrac{\dfrac{}{x =_{T^a} z}\,(df) \qquad \dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{C \in x.act \qquad x.act = z.act}}{C \in x.act} \qquad \dfrac{}{C_1 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_1 \in x.act}}{C_1 \in x_1.act} \qquad \dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\,1}{actv\ C_1\ x_1}\,(I.H.) \qquad \dfrac{}{x_1 =_{T^a} x}\,(df)}{actv\ C_1\ x}\,(act_I)$$

ζ_2 is:

$$\dfrac{\dfrac{\begin{array}{c}\zeta_{2.1} \\ \vdots \\ (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)\end{array}}{actv\ C_2\ x \Rightarrow \\ (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)}\,(\Rightarrow^+)(2) \qquad \dfrac{\begin{array}{c}\zeta_{2.2} \\ \vdots \\ actv\ C_2\ x\end{array}}{(x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2) \Rightarrow \\ actv\ C_2\ x}\,(\Rightarrow^+)(3)}{actv\ C_2\ x \Leftrightarrow (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)}$$

and $\zeta_{2.1}$ is:

$$\dfrac{\begin{array}{c}\zeta_3 \\ \vdots \\ C_2 \in x.act \Leftrightarrow (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)\end{array} \qquad \dfrac{\dfrac{}{actv\ C_2\ x}\,2}{C_2 \in x.act}\,(act_{III})}{(x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)}$$

and $\zeta_{2.2}$ is:

$$\dfrac{\dfrac{\dfrac{}{x_2 =_{T^a} x}\,(df)}{x_2.act = x.act} \qquad \dfrac{\dfrac{}{(x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)}\,3 \qquad \begin{array}{c}\zeta_3 \\ \vdots \\ actv\ C_2\ x \Leftrightarrow \\ (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)\end{array}}{C_2 \in x.act}}{C_2 \in x_2.act} \qquad \dfrac{}{x_2 \mathrel{\dot\in} \delta_{C_2}}\,1}{actv\ C_2\ x_2}\,(I.H.) \qquad \dfrac{}{x_2 =_{T^a} x}\,(df)}{actv\ C_2\ x}\,(act_I)$$

ζ_3 is:

$$\dfrac{\dfrac{C \in z.act \quad \dfrac{}{x.act = z.act}\,(df)}{C \in x.act} \qquad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \mathrel{\dot\in} \delta_C}\,(df)}{\begin{array}{l} C_2 \in x.act \Leftrightarrow \\ (C \in x.act \wedge (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)) \end{array}}\,(df)}{C_2 \in x.act \Leftrightarrow (x.c_{C_1} = C_2 \vee x.c'_{C_1} = C_2)}$$

For the right hand property, assuming the same definitions for x, x_1 and x_2, we have,

$$\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \mathrel{\dot\in} \delta_C}\,(df)}{\begin{array}{r} \exists\, o_1 \subseteq out_{C_1};\ o_2 \subseteq out_{C_2} \bullet \\ x.o'_C = o_1 \cup o_2 \wedge \\ x_1 \mathrel{\dot\in} \delta_{C_1} \wedge x_2 \mathrel{\dot\in} \delta_{C_2} \end{array}}\,(df) \qquad \dfrac{\begin{array}{c} \zeta_1 \\ \vdots \\ inactv\ C_1\ x \end{array} \quad \begin{array}{c} \zeta_2 \\ \vdots \\ inactv\ C_2\ x \end{array} \quad C \notin x.act}{inactv\ C\ x}\,(df)}{inactv\ C\ x}\,(\exists^-)(1) \qquad \dfrac{}{x =_{T^{act}} z}\,(df)}{inactv\ C\ z}\,(inact_I)$$

ζ_1 is:

$$\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \qquad \dfrac{\dfrac{\dfrac{}{z =_{T^a} x}\,(df)}{C \notin z.act \quad z.act = x.act}}{C \notin x.act} \qquad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \mathrel{\dot\in} \delta_C}\,(df)}{\begin{array}{r} C_1 \in x.act \Leftrightarrow \\ C \in x.act \end{array}}\,(df)}{C_1 \notin x.act} \qquad \dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\,1$$

$$\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{C_1 \notin x_1.act} \qquad \dfrac{}{inactv\ C_1\ x_1}\,(I.H.)$$

$$\dfrac{}{inactv\ C_1\ x}\,(inact_I)$$

and ζ_2 is:

$$\dfrac{\dfrac{}{x_2 =_{T^a} x}\,(df)}{x_2.act = x.act} \qquad \dfrac{\dfrac{\dfrac{}{z =_{T^a} x}\,(df)}{C \notin z.act \quad z.act = x.act}}{C \notin x.act} \qquad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \mathrel{\dot\in} \delta_C}\,(df)}{\begin{array}{r} C_2 \in x.act \Leftrightarrow \\ C \in x.act \end{array}}\,(df)}{C_2 \notin x.act} \qquad \dfrac{}{x_2 \mathrel{\dot\in} \delta_{C_2}}\,1$$

$$\dfrac{\dfrac{}{x_2 =_{T^a} x}\,(df)}{C_2 \notin x_2.act} \qquad \dfrac{}{inactv\ C_2\ x_2}\,(I.H.)$$

$$\dfrac{}{inactv\ C_2\ x}\,(inact_I)$$

Lemma B.4.3 Given $C = Dec\ (\omega\ C_1)$ *by* $\{(C_2, \omega\ C_2)\}$, for arbitrary $C_2, \omega\ C_1 = (C_1, \Sigma, \sigma, \Psi, \delta)$, and binding z^{T^a}, we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\,(act_{LEM})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.4.3

For an arbitrary decomposed chart $C = Dec\ (\omega\ C_1)\ by\ \{(C_2, \omega\ C_2)\}$, we have,

$$\dfrac{\dfrac{}{C \in z.act \vee C \notin z.act}\ (LEM) \qquad \dfrac{\dfrac{z \in \delta_C \qquad \overline{C \in z.act}^{\,1}}{actv\ C\ z}\ (act_{II})}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^+) \qquad \dfrac{\dfrac{z \in \delta_C \qquad \overline{C \notin z.act}^{\,1}}{inactv\ C\ z}\ (inact_{II})}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^+)}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^-)(1)$$

Lemma B.4.4 Given $C = Dec\ (\omega\ C_1)\ by\ \{(C_2, \omega\ C_2)\}$, for arbitrary $C_2, \omega\ C_1 = (C_1, \Sigma, \sigma, \Psi, \delta)$, and binding z^{T^a}, for $n \in \{1, 2\}$ we have,

$$\dfrac{actv\ C\ z}{C \in z.act}\ (act_{III}) \qquad \dfrac{inactv\ C\ z}{C \notin z.act}\ (inact_{III})$$

Proof B.4.4

Trivial from the respective definitions of $actv\ C\ z$ and $inactv\ C\ z$.

Proposition 3.5.1 Given $C = Dec\ (\omega\ M)\ by\ \{(S, \omega\ S)\}$, where $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, for the binding z^{T_3} and arbitrary sets o_1 and o_2 we have,

$$\dfrac{\begin{array}{l} z \dot{\in} \delta_C \\ \\ \end{array} \qquad \begin{array}{l} z.o'_C = o_m \cup o_s, \\ z \star \{\!|\ i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_m\ |\!\} \dot{\in} \delta_M, \\ z \star \{\!|\ i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_s\ |\!\} \dot{\in} \delta_S, \\ actv\ C\ z \vee inactv\ C\ z \end{array} \qquad \begin{array}{l} \\ \\ \vdash Q \end{array}}{Q}\ (M_S^-)$$

$$\dfrac{\begin{array}{l} z.o'_C = o_1 \cup o_2 \\ z \star \{\!|\ i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_1\ |\!\} \dot{\in} \delta_M \\ z \star \{\!|\ i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_2\ |\!\} \dot{\in} \delta_S \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \dot{\in} \delta_C}\ (M_S^+)$$

where the usual conditions hold for o_m, o_s and Q, $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$ and $T^a \preceq T_3$.

Proof 3.5.1

For (M_S^-) we have,

$$\dfrac{\dfrac{\dfrac{z \dot{\in} \delta_C}{x \in \delta_C}\ (df)}{\exists\, o_1, o_2 \bullet\ x.o'_C = o_1 \cup o_2 \wedge x_1 \dot{\in} \delta_M \wedge x_2 \dot{\in} \delta_S}\ (df) \qquad \dfrac{\dfrac{z \dot{\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\ (act_{LEM}) \qquad \begin{array}{c} \zeta_1 \quad \zeta_2 \quad \zeta_3 \\ \vdots \qquad \vdots \qquad \vdots \\ Q \end{array}}{Q}\ (\exists^-)(1)$$

ζ_1 is:

$$\dfrac{\dfrac{}{x.o'_C = o_1 \cup o_2}\,1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z.o'_C = x.o'_C}}{z.o'_C = o_1 \cup o_2}$$

ζ_2 is:

$$\dfrac{\dfrac{}{x_1 \dot{\in} \delta_M}\,1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z_1 =_{T^a_m} x_1}\ (df)}{z_1 \dot{\in} \delta_M}$$

ζ_3 is:

$$\dfrac{\dfrac{}{x_2 \dot{\in} \delta_S}\,1 \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z_2 =_{T^a_s} x_2}\ (df)}{z_2 \dot{\in} \delta_S}$$

where $x =_{def} z \restriction T^a$, $z_n =_{def} z \star \langle\!|\ i_{C_n} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_n}, o_{C_n}{}' \Rrightarrow o_n\ |\!\rangle$. and $x_n =_{def} x \star \langle\!|\ i_{C_n} \Rrightarrow (x.i_C \cup fb_x) \cap in_{C_n}, o_{C_n}{}' \Rrightarrow o_n\ |\!\rangle$ for $n = 1, 2$.
For (M_S^+) we have,

$$\dfrac{\dfrac{\begin{array}{l} M \in x.act \Leftrightarrow C \in x.act \wedge \\ S \in x.act \Leftrightarrow (C \in x.act \wedge \\ \quad (x.c_M = S \vee x.c'_M = S)) \end{array} \qquad \dfrac{x.o'_C = o_1 \cup o_2 \quad x_1 \dot{\in} \delta_M \quad x_2 \dot{\in} \delta_S}{\exists\, o_1, o_2 \bullet x.o'_C = o_1 \cup o_2 \wedge x_1 \dot{\in} \delta_M \wedge x_2 \dot{\in} \delta_S}\ (\exists^+)}{\dfrac{x \in \delta_C}{z \dot{\in} \delta_C}\ (df)}\ (df)$$

where $x =_{def} z \restriction T^a$, $x_1 =_{def} x \star \langle\!|\ i_M \Rrightarrow (x.i_C \cup fb_x) \cap in_M, o'_M \Rrightarrow o_1\ |\!\rangle$ and $x_2 =_{def} x \star \langle\!|\ i_S \Rrightarrow (x.i_C \cup fb_x) \cap in_S, o'_S \Rrightarrow o_2\ |\!\rangle$.
ζ_1 is:

$$\dfrac{actv\ C\ x \vee inactv\ C\ x \qquad \begin{array}{cc} \zeta_{1.1} & \zeta_{1.2} \\ \vdots & \vdots \end{array}}{\begin{array}{l} M \in x.act \Leftrightarrow C \in x.act \wedge \\ S \in x.act \Leftrightarrow (C \in x.act \wedge (x.c_M = S \vee x.c'_M = S)) \end{array}}\ (\vee^-)(1)$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{actv\ C\ x}\,1}{actv\ M\ x}\ (df)}{M \in x.act}\ (act_{III}) \qquad \begin{array}{c} \zeta_{1.1.1} \\ \vdots \\ S \in x.act \Leftrightarrow \\ (x.c_M = S \vee x.c'_M = S) \end{array} \qquad \dfrac{\dfrac{}{actv\ C\ x}\,1}{C \in x.act}\ (df)}{M \in x.act \wedge (S \in x.act \Leftrightarrow (x.c_M = S \vee x.c'_M = S)) \wedge C \in x.act}\ (\wedge^+)}{\begin{array}{r} (M \in x.act \wedge (S \in x.act \Leftrightarrow (x.c_M = S \vee x.c'_M = S)) \wedge C \in x.act) \vee \\ (M \notin x.act \wedge S \notin x.act \wedge C \notin x.act) \end{array}}\ (\vee^+)$$

$$\dfrac{}{\begin{array}{l} M \in x.act \Leftrightarrow C \in x.act \wedge \\ S \in x.act \Leftrightarrow (C \in x.act \wedge (x.c_M = S \vee x.c'_M = S)) \end{array}}\ (prop\ logic)$$

where $\zeta_{1.1.1}$ is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\ x}\,1}{S \in x.act \Leftrightarrow (C \in x.act \wedge (x.c_M = S \vee x.c'_M = S))}\ (df) \qquad \dfrac{\dfrac{}{actv\ C\ x}\,1}{C \in x.act}\ (act_{III})}{S \in x.act \Leftrightarrow (x.c_M = S \vee x.c'_M = S)}$$

$\zeta_{1.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{inactv\ C\ x}\,1}{inactv\ M\ x}\,(df)}{M \notin x.act}\,(inact_{III}) \quad \dfrac{\dfrac{\dfrac{}{inactv\ C\ x}\,1}{inactv\ S\ x}\,(df)}{S \notin x.act}\,(inact_{III}) \quad \dfrac{\dfrac{}{inactv\ C\ x}\,1}{C \notin x.act}\,(df)}{M \notin x.act \wedge S \notin x.act \wedge C \notin x.act}\,(\wedge^+)}{(M \in x.act \wedge (S \in x.act \Leftrightarrow (x.c_M = S \vee x.c'_M = S)) \wedge C \in x.act) \vee (M \notin x.act \wedge S \notin x.act \wedge C \notin x.act)}\,(\vee^+)$$

$$\dfrac{(M \in x.act \wedge (S \in x.act \Leftrightarrow (x.c_M = S \vee x.c'_M = S)) \wedge C \in x.act) \vee (M \notin x.act \wedge S \notin x.act \wedge C \notin x.act)}{\begin{array}{l} M \in x.act \Leftrightarrow C \in x.act \wedge \\ S \in x.act \Leftrightarrow (C \in x.act \wedge (x.c_M = S \vee x.c'_M = S)) \end{array}}\,(prop\ logic)$$

Proposition 3.5.2 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary S and $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and z^{T_3} we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{inactv\ S\ z}\,(M^-_{S\,II})$$

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{z \star \langle\!|\ i_M \Rrightarrow z.i_C \cap in_M, o'_M \Rrightarrow z.o'_C\ |\!\rangle \mathrel{\dot\in} \delta_M}\,(M^-_{S\,III})$$

$$\dfrac{z \mathrel{\dot\in} \delta_C \quad z.c_M \neq S \quad z.c'_M \neq S}{z \star \langle\!|\ i_S \Rrightarrow z.i_C \cap in_S, o'_S \Rrightarrow \{\}\ |\!\rangle \mathrel{\dot\in} \delta_S}\,(M^-_{S\,IV})$$

where $T^a \preceq T_3$.

Proof 3.5.2

For $(M^-_{S\,II})$ we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C \qquad \dfrac{\dfrac{}{actv\ C\ z \vee inactv\ C\ z}\,1 \quad \dfrac{\dfrac{}{inactv\ C\ z}\,2}{inactv\ S\ z}\,(df) \quad \begin{array}{c}\zeta_1 \\ \vdots \\ inactv\ S\ z\end{array}}{inactv\ S\ z}\,(\vee^-)(2)}{inactv\ S\ z}\,(M^-_S)(1)$$

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{z_2 \mathrel{\dot\in} \delta_S}\,1}{z_2 \restriction T^a_2 \in \delta_S} \quad \dfrac{\begin{array}{c}\zeta_2 \\ \vdots \\ \bot\end{array}}{S \notin (z_2 \restriction T^a_2).act}\,(\bot^-)(3)}{inactv\ S\ (z_2 \restriction T^a_2)}\,(inact_{II}) \quad \dfrac{}{z_2 \restriction T^a_2 =_{T^a_2} z}\,(df)}{inactv\ S\ z}\,(inact_I)$$

where $z_1 =_{def} z \star \langle\!|\ i_M \Rrightarrow (z.i_C \cup fb_z) \cap in_M, o'_M \Rrightarrow o_m\ |\!\rangle$ and
$z_2 =_{def} z \star \langle\!|\ i_S \Rrightarrow (z.i_C \cup fb_z) \cap in_S, o'_S \Rrightarrow o_s\ |\!\rangle$.

ζ_2 is:

$$\dfrac{\dfrac{}{z_2 =_{T_3} z}\,(df) \quad \dfrac{\dfrac{}{z_2 \mathrel{\dot\in} \delta_S}\,1 \quad \dfrac{}{S \in z_2.act}\,3}{actv\ S\ z_2}\,(act_{II})}{actv\ S\ z} \quad \begin{array}{c}\zeta_3 \\ \vdots \\ \neg\ actv\ S\ z\end{array}$$

$$\bot$$

ζ_3 is:

$$\dfrac{\dfrac{\overline{actv\ C\ z}^{\,2}}{actv\ S\ z \Leftrightarrow (z.c_M = S \vee z.c'_M = S)}\ (df) \qquad \dfrac{\dfrac{\overline{z.c_M \neq S} \quad \overline{z.c'_M \neq S}}{z.c_M \neq S \wedge z.c'_M \neq S}\ (\wedge^+)}{\neg\,(z.c_M = S \vee z.c'_M = S)}\ (\textit{prop logic})}{\neg\ actv\ S\ z}$$

For $(M^-_{S\,III})$, assuming the usual definitions for bindings z_1 and z_2 (as above), we have,

$$\dfrac{\dfrac{\overline{z_1 \dot{\in} \delta_M}^{\,1} \qquad \dfrac{\dfrac{\overline{z_2 \dot{\in} \delta_S}^{\,1} \quad \begin{array}{c}\zeta_1\\ \vdots\\ inactv\ S\ z_2\end{array}}{z_2.o'_S = \{\}}\ (iact^-_{II}) \qquad \dfrac{\overline{z.o'_C = o_1 \cup o_2}^{\,1}}{z.o'_C = z_1.o'_M \cup z_2.o'_S}\ (df)}{z.o'_C = z_1.o'_M}\ (df)}{z \star \langle\!|\ i_M \Rightarrow z.i_C \cap in_M, o'_M \Rightarrow z.o'_C\ |\!\rangle \dot{\in} \delta_M} \qquad z \dot{\in} \delta_C}{z \star \langle\!|\ i_M \Rightarrow z.i_C \cap in_M, o'_M \Rightarrow z.o'_C\ |\!\rangle \dot{\in} \delta_M}\ (M^-_S)(1)$$

where ζ_1 is:

$$\dfrac{\overline{actv\ C\ z \vee inactv\ C\ z}^{\,1} \qquad \dfrac{\begin{array}{c}\zeta_2\\ \vdots\\ inactv\ S\ z_2\end{array} \qquad \dfrac{\dfrac{\overline{inactv\ C\ z}^{\,2}}{inactv\ S\ z}\ (df) \qquad \overline{z =_{T_3} z_2}\ (df)}{inactv\ S\ z_2}\ (inact_I)}{inactv\ S\ z_2}\ (\vee^-)(2)}{inactv\ S\ z_2}$$

ζ_2 is:[3]

$$\dfrac{\overline{z_2 \dot{\in} \delta_S}^{\,1} \qquad \dfrac{\begin{array}{c}\zeta_3\\ \vdots\\ \neg\ actv\ S\ z\end{array} \quad \overline{z =_{T_3} z_2}\ (df)}{\neg\ actv\ S\ z_2}\ (inact_{I-contra})}{inactv\ S\ z_2}\ (act_{LEM})$$

and ζ_3 is:

$$\dfrac{\dfrac{\overline{actv\ C\ z}^{\,2}}{actv\ S\ z \Leftrightarrow (z.c_M = S \vee z.c'_M = S)}\ (df) \qquad \dfrac{\dfrac{z.c_M \neq S \quad z.c'_M \neq S}{z.c_M \neq S \wedge z.c'_M \neq S}}{\neg\,(z.c_M = S \vee z.c'_M = S)}}{\neg\ actv\ S\ z}$$

Similarly, for $(M^-_{S\,IV})$ we have,

$$\dfrac{\dfrac{\overline{z_2 \dot{\in} \delta_S}^{\,1} \qquad \dfrac{\overline{actv\ C\ z \vee inactv\ C\ z}^{\,1} \qquad \dfrac{\begin{array}{c}\zeta_1\\ \vdots\\ inactv\ S\ z_2\end{array} \qquad \dfrac{\dfrac{\overline{inactv\ C\ z}^{\,2}}{inactv\ S\ z}\ (df) \qquad \overline{z =_{T_3} z_2}\ (df)}{inactv\ S\ z_2}\ (inact_I)}{inactv\ S\ z_2}\ (\vee^-)(2)}{z_2.o'_S = \{\}}\ (iact^-_{II}) \qquad \overline{z_2 \dot{\in} \delta_M}^{\,1}}{z \star \langle\!|\ i_S \Rightarrow z.i_C \cap in_S, o'_S \Rightarrow \{\}\ |\!\rangle \dot{\in} \delta_S}\ (df)}{z \dot{\in} \delta_C} \qquad}{z \star \langle\!|\ i_S \Rightarrow z.i_C \cap in_S, o'_S \Rightarrow \{\}\ |\!\rangle \dot{\in} \delta_S}\ (M^-_S)(1)$$

[3]Here, and in the following, we freely use an alternate form for the rule (act_{LEM}). Given the original rule has the form $\Gamma \vdash act \vee inact$ it follows that $\Gamma, \neg\ act \vdash inact$. Also, the rule labelled $(inact_{I-contra})$ represents using the contrapositive proposition $\Gamma, \neg\ B \vdash \neg\ A$ where proposition $(inact_I)$ has the form $\Gamma, A \vdash B$.

ζ_1 is:

$$\dfrac{\dfrac{}{z_2 \dot{\in} \delta_S}\,1 \qquad \dfrac{\begin{array}{c}\vdots\,\zeta_2\\ \neg\ actv\ S\ z\end{array} \quad \dfrac{}{z =_{T_3} z_2}\,(df)}{\neg\ actv\ S\ z_2}\,(inact_{I-contra})}{inactv\ S\ z_2}\,(act_{LEM})$$

and ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\ z}\,2}{actv\ S\ z \Leftrightarrow (z.c_M = S \vee z.c'_M = S)}\,(df) \qquad \dfrac{\dfrac{z.c_M \neq S \quad z.c'_M \neq S}{z.c_M \neq S \wedge z.c'_M \neq S}}{\neg\,(z.c_M = S \vee z.c'_M = S)}}{\neg\ actv\ S\ z}$$

Proposition 3.5.3 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary S, $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, and binding z^{T_3} we have

$$\dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z \dot{\in} \Xi Chart_C}\,(iact_I^-) \qquad \dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\,(iact_{II}^-)$$

$$\dfrac{inactv\ C\ z \quad z \dot{\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \dot{\in} \delta_C}\,(iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$ and $T^a \preceq T_3$.

Proof 3.5.3

For $(iact_I^-)$, assuming the usual definitions for z_1 and z_2, we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z_1 \dot{\in} \delta_M}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ M\ z}\,(df) \quad \dfrac{}{z_1 =_{T_3} z}\,(df)}{inactv\ M\ z_1}\,(inact_I)}{z_1 \dot{\in} \Xi Chart_M}\,(I.H.) \quad \dfrac{\dfrac{}{z_1 =_{T_3} z}\,(df)}{z_1 =_{(\Delta U_m)} z}\,((\Delta U_m) \preceq T_3)}{z \dot{\in} \Xi Chart_M} \quad \dfrac{\begin{array}{c}\vdots\,\zeta_1\\ z \dot{\in} \Xi Chart_S\end{array}}{}\,(S^+_\wedge)}{z \dot{\in} (\Xi Chart_M \wedge \Xi Chart_S)}\,(Chart_C\text{-}df)$$

$$\dfrac{z \dot{\in} \delta_C \qquad z \dot{\in} \Xi Chart_C}{z \dot{\in} \Xi Chart_C}\,(M_S^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{z_2 \dot{\in} \delta_S}\,1 \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ S\ z}\,(df) \quad \dfrac{}{z_2 =_{T_3} z}\,(df)}{inactv\ S\ z_2}\,(inact_I)}{z_2 \dot{\in} \Xi Chart_S}\,(I.H.) \quad \dfrac{\dfrac{}{z_2 =_{T_3} z}\,(df)}{z_2 =_{(\Delta U_s)} z}\,((\Delta U_s) \preceq T_3)}{z \dot{\in} \Xi Chart_S}$$

For $(iact_{II}^-)$ we have,

$$\dfrac{z \dot{\in} \delta_C \qquad \dfrac{\dfrac{\dfrac{}{z.o'_C = o_1 \cup o_2}\,1}{z.o'_C = z_1.o'_M \cup z_2.o'_S}\,(df) \quad \begin{array}{c}\vdots\,\zeta_1\\ z_1.o'_M = \{\}\end{array} \quad \begin{array}{c}\vdots\,\zeta_2\\ z_1.o'_M = \{\}\end{array}}{z.o'_C = \{\}}\,(|\_|^-)(1)}{z.o'_C = \{\}}$$

where ζ_1 is:

$$\dfrac{\dfrac{}{z_1 \dot{\in} \delta_M}\,1 \qquad \dfrac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ M\ z}\,(df) \qquad \dfrac{}{z_1 =_{T_3} z}\,(df)}{inactv\ M\ z_1}\,(inact_I)}{z_1.o'_M = \{\}}\,(I.H.)$$

ζ_2 is:

$$\dfrac{\dfrac{}{z_2 \dot{\in} \delta_S}\,1 \qquad \dfrac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ S\ z}\,(df) \qquad \dfrac{}{z_2 =_{T_3} z}\,(df)}{inactv\ S\ z_2}\,(inact_I)}{z_2.o'_S = \{\}}\,(I.H.)$$

And for $(iact^+)$, we have,

$$\dfrac{\dfrac{z.o'_C = \{\}}{z.o'_C = \{\} \cup \{\}} \qquad \overset{\zeta_1}{\vdots}\ z_1 \dot{\in} \delta_M \qquad \overset{\zeta_2}{\vdots}\ z_2 \dot{\in} \delta_S \qquad \dfrac{inactv\ C\ z}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^+)}{z \dot{\in} \delta_C}\,(|\_|^+)$$

ζ_1 is:

$$\dfrac{\dfrac{z \dot{\in} \Xi Chart_C}{z_1 \dot{\in} \Xi Chart_M}\,((\Delta U_m) \preceq (\Delta U)) \qquad \dfrac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ M\ z}\,(df) \qquad \dfrac{}{z =_{T_3} z_1}\,(df)}{inactv\ M\ z_1}\,(inact_I) \qquad \dfrac{}{z_1.o'_M = \{\}}\,(df)}{z_1 \dot{\in} \delta_M}\,(I.H.)$$

and ζ_2 is:

$$\dfrac{\dfrac{z \dot{\in} \Xi Chart_C}{z_2 \dot{\in} \Xi Chart_S}\,((\Delta U_s) \preceq (\Delta U)) \qquad \dfrac{\dfrac{\dfrac{inactv\ C\ z}{inactv\ S\ z}\,(df) \qquad \dfrac{}{z =_{T_3} z_2}\,(df)}{inactv\ S\ z_2}\,(inact_I) \qquad \dfrac{}{z_2.o'_S = \{\}}\,(df)}{z_2 \dot{\in} \delta_S}\,(I.H.)$$

where $[\![\delta_M]\!]^{\mathbb{P}\,T_m}_{z_C}$, $[\![\delta_S]\!]^{\mathbb{P}\,T_s}_{z_C}$.

Proposition 3.5.4 Given $C = Dec\ (\omega\ M)$ *by* $\{(S, \omega\ S)\}$, for arbitrary $S, \omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$ and bindings z^{T_3} and x^{T_3} we have,

$$\dfrac{z \dot{\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \dot{\in} \delta_C}\,(Z_t^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

Proof 3.5.4

The proof of this proposition is identical in structure to the proof of Proposition 3.4.4 (*i.e.* that $(Z_t^{\in})$ holds for composed charts), replacing the applications of the rules $(|\_|^-)$ and $(|\_|^+)$ with (M_S^-) and (M_S^+) respectively. Apart from this, the significant difference between the two proofs is hidden by the application of the rule (act_I). This divergence amounts to the difference between the proof of Lemma B.3.1 (that (act_I) holds for composed charts) and Lemma B.4.1 (that (act_I) holds for decomposed charts).

B.5 Proofs for Section 3.6: Chart context and signal hiding

Again we show that the properties introduced in lemmas B.3.1 to B.3.4 hold for charts that contain signal hiding.

Lemma B.5.1 Given $C =_X [C_1]_Y$, for arbitrary z and z_1 we have,

$$\frac{z =_{T_3} z_1 \quad actv\ C\ z}{actv\ C\ z_1}\ (act_I) \qquad \frac{z =_{T_3} z_1 \quad inactv\ C\ z}{inactv\ C\ z_1}\ (inact_I)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$, $\Delta U \preceq T_3$ and $T^{act} \preceq T_3$.

Proof B.5.1

For the left hand property we have,

$$\frac{\dfrac{actv\ C\ z}{actv\ C_1\ z}\ (df) \qquad z =_{T_3} z_1}{actv\ C_1\ z_1}\ (I.H.) \qquad \frac{\dfrac{actv\ C\ z}{C \in z.act}\ (df) \qquad \dfrac{z =_{T_3} z_1}{z.act = z_1.act}\ (T^{act} \preceq T_3)}{C \in z_1.act}\ (df)$$
$$actv\ C\ z_1$$

Similarly, for the right hand property we have,

$$\frac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\ (df) \qquad z =_{T_3} z_1}{inactv\ C_1\ z_1}\ (I.H.) \qquad \frac{\dfrac{inactv\ C\ z}{C \notin z.act}\ (inact_{III}) \qquad \dfrac{z =_{T^{act}} z_1}{z.act = z_1.act}}{C \notin z_1.act}\ (df)$$
$$inactv\ C\ z_1$$

Lemma B.5.2 Given $C =_X [C_1]_Y$, for arbitrary z^{T_3}, we have,

$$\frac{z \mathrel{\dot\in} \delta_C \quad C \in z.act}{actv\ C\ z}\ (act_{II}) \qquad \frac{z \mathrel{\dot\in} \delta_C \quad C \notin z.act}{inactv\ C\ z}\ (inact_{II})$$

where $[\![\delta_C]\!]^{T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof B.5.2

For the property on the left we have,

$$\frac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\ (df)}{\exists\, o_1 \bullet x.o'_C = o_1 \cap out_C \wedge x_1 \mathrel{\dot\in} \delta_{C_1}}\ (df) \qquad \dfrac{\dfrac{\begin{array}{c}\zeta_1\\ \vdots\\ actv\ C_1\ x\end{array} \qquad \dfrac{\overline{z =_{T^a} x}\ (df)}{z.act = x.act} \qquad C \in z.act}{C \in x.act}\ (df)}{actv\ C\ x}\ (\exists^-)(1) \qquad \dfrac{}{z =_{T^a} x}\ (df)}{actv\ C\ x}\ (act_I)$$
$$actv\ C\ z$$

where $x =_{def} z \restriction T^a$, $x_1 =_{def} x \star \langle\!| \, i_{C_1} \Rrightarrow x.i_C, o_{C_1}' \Rrightarrow o_1 \, |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \quad \dfrac{\dfrac{\dfrac{}{x =_{T^a} z}\,(df) \quad \dfrac{C \in x.act \quad \dfrac{}{x.act = z.act}}{C \in x.act}}{C_1 \in x.act} \quad \dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{C_1 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_1 \in x.act}}{C_1 \in x_1.act} \quad \dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\,1}{actv\ C_1\ x_1}\,(I.H.)}{actv\ C_1\ x}\,(act_I)$$

For the right hand property, with x and x_1 defined as above, we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{\exists\, o_1 \bullet x.o_C' = o_1 \cap out_C \wedge x_1 \mathrel{\dot\in} \delta_{C_1}}\,(df) \quad \dfrac{\begin{array}{c}\zeta_1 \\ \vdots \\ inactv\ C_1\ x \quad C \notin x.act\end{array}}{inactv\ C\ x}\,(df)}{inactv\ C\ x}\,(\exists^-)(1) \quad \dfrac{}{x =_{T^{act}} z}\,(df)}{inactv\ C\ z}\,(inact_I)$$

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x_1 =_{T^a} x}\,(df)}{x_1.act = x.act} \quad \dfrac{\dfrac{\dfrac{}{z =_{T^a} x}\,(df)}{C \notin x.act} \quad \dfrac{\dfrac{\dfrac{z \mathrel{\dot\in} \delta_C}{x \in \delta_C}\,(df)}{C_1 \in x.act \Leftrightarrow C \in x.act}\,(df)}{C_1 \notin x.act}}{C_1 \notin x_1.act} \quad \dfrac{}{x_1 \mathrel{\dot\in} \delta_{C_1}}\,1}{inactv\ C_1\ x_1}\,(I.H.)}{inactv\ C_1\ x}\,(inact_I)$$

Lemma B.5.3 Given $C =_X [C_1]_Y$, for arbitrary z^{T_3}, we have,

$$\dfrac{z \mathrel{\dot\in} \delta_C}{actv\ C\ z \vee inactv\ C\ z}\,(act_{LEM})$$

where $[\![\delta_C]\!]^{T^a}_{Z_C}$ and $T^a \preceq T_3$.

Proof B.5.3

$$\dfrac{\dfrac{}{C \in z.act \vee C \notin z.act}\,(LEM) \quad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C \quad \overline{C \in z.act}^{\,1}}{actv\ C\ z}\,(act_{II})}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^+) \quad \dfrac{\dfrac{z \mathrel{\dot\in} \delta_C \quad \overline{C \notin z.act}^{\,1}}{inactv\ C\ z}\,(inact_{II})}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^+)}{actv\ C\ z \vee inactv\ C\ z}\,(\vee^-)(1)$$

Lemma B.5.4 Given $C =_X [C_1]_Y$, for arbitrary binding z^{T_3} we have,

$$\frac{actv\ C\ z}{C \in z.act}\ (act_{III}) \qquad \frac{inactv\ C\ z}{C \notin z.act}\ (inact_{III})$$

Proof B.5.4

Trivial from the respective definitions of *act* and *inact*.

Proposition 3.6.1 Given $C =_X [C_1]_Y$, for the binding z^{T_3} we have,

$$\frac{z \dot{\in} \delta_C \qquad \begin{array}{l} z.o'_C = o_1 \cap out_C, \\ z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1} \vdash P \end{array}}{P}\ (x[]^-_Y)$$

$$\frac{\begin{array}{l} z.o'_C = o_1 \cap out_C, \\ z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}, \\ actv\ C\ z \vee inactv\ C\ z \end{array}}{z \dot{\in} \delta_C}\ (x[]^+_Y)$$

where the usual conditions hold for o_1 and P, $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof 3.6.1

For $(x[]^-_Y)$ we have,

$$\dfrac{\dfrac{\dfrac{z \dot{\in} \delta_C}{z \restriction T^a \in \delta_C}}{\begin{array}{l} \exists\, o_1 \bullet z.o'_C = o_1 \cap out_C \wedge \\ z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1} \end{array}}(df) \qquad \begin{array}{c} \zeta_1 \\ \vdots \\ P \end{array}}{P}\ (\exists^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}}\,1 \qquad \dfrac{}{z.o'_C = o_1 \cap out_C}\,1}{P}\ (ass)$$

For $(x[]^+_Y)$ we have,

$$\dfrac{\dfrac{\begin{array}{c} \zeta_1 \\ \vdots \end{array} \qquad \dfrac{\dfrac{z.o'_C = o_1 \cap out_C \quad \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}}{z.o'_C = o_1 \cap out_C \wedge \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}}(\wedge^+)}{\exists\, o_1 \bullet z.o'_C = o_1 \cap out_C \wedge \langle\!|\ i_{C_1} \Rrightarrow z.i_C \cap in_{C_1}, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}}(\exists^+)}{C_1 \in z.act \Leftrightarrow C \in z.act \wedge \exists\, o_1 \bullet z.o'_C = o_1 \cap out_C \wedge \langle\!|\ i_{C_1} \Rrightarrow z.i_C, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle \dot{\in} \delta_{C_1}}(\wedge^+)}{z \dot{\in} \delta_C}(df)$$

where ζ_1 is:

$$\dfrac{actv\ C\ z \vee inactv\ C\ z \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{actv\ C\ z}^{\,1}}{actv\ C_1\ z \wedge C \in z.act}\,(df)}{actv\ C_1\ z}\,(act_{III})}{C_1 \in z.act}}{C_1 \in z.act \wedge C \in z.act} \qquad \dfrac{\dfrac{\overline{actv\ C\ z}^{\,1}}{actv\ C_1\ z \wedge C \in z.act}\,(df)}{C \in z.act}\,(\wedge^-)}{C_1 \in z.act \Leftrightarrow C \in z.act} \qquad \begin{array}{c}\zeta_{1.2}\\ \vdots\end{array}}{C_1 \in z.act \Leftrightarrow C \in z.act}\,(\vee^-)(1)$$

$\zeta_{1.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{inactv\ C\ z}^{\,1}}{inactv\ C_1\ z \wedge C \notin z.act}\,(df)}{inactv\ C_1\ z}\,(act_{III})}{C_1 \notin z.act} \qquad \dfrac{\dfrac{\overline{inactv\ C\ z}^{\,1}}{inactv\ C_1\ z \wedge C \notin z.act}\,(df)}{C \notin z.act}\,(\wedge^-)}{C_1 \notin z.act \wedge C \notin z.act}}{C_1 \notin z.act \Leftrightarrow C \notin z.act}}{C_1 \in z.act \Leftrightarrow C \in z.act}\,(contrapositive)$$

Proposition 3.6.2 Given $C =_X [C_1]_Y$, for arbitrary binding z^{T_3} we have,

$$\dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z \dot{\in} \Xi Chart_C}\,(iact_I^-) \qquad \dfrac{z \dot{\in} \delta_C \quad inactv\ C\ z}{z.o'_C = \{\}}\,(iact_{II}^-)$$

$$\dfrac{inactv\ C\ z \quad z \dot{\in} \Xi Chart_C \quad z.o'_C = \{\}}{z \dot{\in} \delta_C}\,(iact^+)$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$ and $T^a \preceq T_3$.

Proof 3.6.2

For $(iact_I^-)$ we have,

$$\dfrac{\dfrac{z \dot{\in} \delta_C \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{}^{\,1}}{z_1 \dot{\in} \delta_{C_1}} \quad \dfrac{inactv\ C\ z}{inactv\ C_1\ z}\,(df)}{z_1 \dot{\in} \Xi Chart_{C_1}}\,(I.H.) \quad \dfrac{\overline{z_1 =_{T_3} z}}{z_1 =_{(\Delta U_1)} z}\,(df)}{z \dot{\in} \Xi Chart_{C_1}}\,((\Delta U_1) \preceq T_3) \qquad \overline{Chart_{C_1} = Chart_C}\,(df)}{z \dot{\in} \Xi Chart_C}\,(x[]_Y^-)(1)}{z \dot{\in} \Xi Chart_C}$$

where $z_1 =_{def} z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle$.

For $(iact_{II}^-)$ we have,

$$\dfrac{z \dot{\in} \delta_C \qquad \dfrac{\dfrac{\dfrac{\overline{z.o'_C = o_1 \cap out_C}^{\,1}}{z.o'_C = z_1.o_{C_1}{}' \cap out_C}\,(df) \qquad \dfrac{\dfrac{\overline{}^{\,1}}{z_1 \dot{\in} \delta_{C_1}} \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\,(df) \quad \dfrac{\overline{z_1 =_{T_3} z}}{inactv\ C_1\ z_1}\,(df)(inact_I)}{z_1.o_{C_1}{}' = \{\}}\,(I.H.)}{z.o'_C = \{\}}}{z.o'_C = \{\}}\,(x[]_Y^-)(1)}{z.o'_C = \{\}}$$

where $z_1 =_{def} z \star \langle\!| \, i_{C_1} \Rrightarrow z.i_C, o_{C_1}{}' \Rrightarrow o_1 \, |\!\rangle$.

And for $(iact^+)$ we have,

$$\dfrac{\dfrac{z.o'_C = \{\}}{z.o'_C = \{\} \cap out_C} \quad \begin{matrix}\zeta_1 \\ \vdots \\ z_1 \dot{\in} \delta_{C_1}\end{matrix} \quad \dfrac{inactv\ C\ z}{actv\ C\ z \vee inactv\ C\ z}\ (\vee^+)}{z \dot{\in} \delta_C}\ (x[]^+_Y)$$

where $z_1 =_{def} z \star \langle\!| \, i_{C_1} \Rrightarrow z.i_C, o_{C_1}{}' \Rrightarrow \{\} \, |\!\rangle$.
ζ_1 is:

$$\dfrac{\dfrac{z \dot{\in} \Xi Chart_C}{z_1 \dot{\in} \Xi Chart_{C_1}}\ ((\Delta U_1) \preceq (\Delta U)) \quad \dfrac{\dfrac{inactv\ C\ z}{inactv\ C_1\ z}\ (df) \quad \dfrac{}{z =_{T_3} z_1}\ (df)}{inactv\ C_1\ z_1}\ (inact_I) \quad \dfrac{}{z_1.o_{C_1}{}' = \{\}}\ (df)}{z_1 \dot{\in} \delta_{C_1}}\ (I.H.)$$

Proposition 3.6.3 Given $C =_X [C_1]_Y$ for arbitrary bindings z^{T_3} and x^{T_3} we have,

$$\dfrac{z \dot{\in} \delta_C \quad x =_{T_i} z \quad z.i_C \cup fb_z = x.i_C \cup fb_x}{x \dot{\in} \delta_C}\ (Z_t^{\in})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T^a}_{\mathcal{Z}_C}$, $T_i =_{def} T^a - V^i$ and $T^a \preceq T_3$.

Proof 3.6.3

$$\dfrac{\dfrac{}{z.o'_C = o_1 \cap out_C}\ 1 \quad \dfrac{x =_{T_i} z}{x.o'_C = z.o'_C} \quad \dfrac{\dfrac{}{x.o'_C = o_1 \cap out_C} \quad \begin{matrix}\zeta_1 \\ \vdots \\ x_1 \dot{\in} \delta_{C_1}\end{matrix} \quad \begin{matrix}\zeta_2 \\ \vdots \\ actv\ C\ x \vee inactv\ C\ x\end{matrix}}{z \dot{\in} \delta_C \qquad x \dot{\in} \delta_C}\ (x[]^+_Y)}{x \dot{\in} \delta_C}\ (x[]^-_Y)(1)$$

ζ_1 is:

$$\dfrac{\dfrac{}{z_1 \dot{\in} \delta_{C_1}}\ 1 \quad \dfrac{\dfrac{x =_{T_i} z \quad \dfrac{}{z_1.o_{C_1}{}' = x_1.o_{C_1}{}'}\ (df)}{x_1 =_{(T_i \curlyvee T_1^{out})} z_1}}{x_1 =_{T_{1_i}} z_1}\ (T_{1_i} \preceq (T_i \curlyvee T_1^{out})) \quad \begin{matrix}\zeta_{1.1} \\ \vdots \\ z_1.i_{C_1} \cup fb_{z_1} = \\ x_1.i_{C_1} \cup fb_{x_1}\end{matrix}}{x_1 \dot{\in} \delta_{C_1}}\ (I.H.)$$

$\zeta_{1.1}$ is:

$$\dfrac{\begin{matrix}\zeta_{1.1.1} \\ \vdots \\ z.i_C \cup fb_{z_1} = x.i_C \cup fb_{x_1}\end{matrix}\ \dfrac{\dfrac{}{z_1.o_{C_1}{}' = x_1.o_{C_1}{}'}\ (df)}{\dfrac{z_1.o_{C_1}{}' \cap \Psi = x_1.o_{C_1}{}' \cap \Psi}{fb_{z_1} = fb_{x_1}}}\ (\cup\text{-}df) \quad \dfrac{}{z_1.i_{C_1} = z.i_C}\ (df) \quad \dfrac{}{x_1.i_{C_1} = x.i_C}\ (df)}{z_1.i_{C_1} \cup fb_{z_1} = x_1.i_{C_1} \cup fb_{x_1}}$$

$\zeta_{1.1.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x =_{T_i} z}{x =_{T^{out}} z}\,(T^{out} \preceq T_i)}{x.o'_C = z.o'_C}}{x.o'_C \cap \Psi = z.o'_C \cap \Psi}}{fb_x = fb_z} \qquad z.i_C \cup fb_z = x.i_C \cup fb_z}{z.i_C \cup fb_z = x.i_C \cup fb_z} \qquad \dfrac{\dfrac{\dfrac{}{o_1 \cap out_C \cap \Psi \subseteq o_1 \cap \Psi}\,(\cap\text{-}df) \qquad \dfrac{}{z.o'_C = o_1 \cap out_C}\,1}{z.o'_C \cap \Psi \subseteq o_1 \cap \Psi}}{z.o'_C \cap \Psi \subseteq z_1.o_{C_1}{}' \cap \Psi}\,(df)}{fb_z \subseteq fb_{z_1}}$$

$$z.i_C \cup fb_{z_1} = x.i_C \cup fb_{z_1}$$

where $x_1 =_{def} x \star \langle\!|\ i_{C_1} \Rrightarrow x.i_C, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle$, $z_1 =_{def} z \star \langle\!|\ i_{C_1} \Rrightarrow z.i_C, o_{C_1}{}' \Rrightarrow o_1\ |\!\rangle$, $[\![\delta_{C_1}]\!]^{\mathbb{P}\,T_1^a}_{z_C}$ and $T_{1_1} = T_1^a - T_1^{in}$.

ζ_2 is:

$$\dfrac{\dfrac{}{actv\ C\ z \vee inactv\ C\ z}\,1 \qquad \dfrac{\dfrac{\dfrac{}{actv\ C\ z}\,2 \quad x =_{T_i} z}{actv\ C\ x}\,(act_I)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^+) \qquad \dfrac{\dfrac{\dfrac{}{inactv\ C\ z}\,2 \quad x =_{T_i} z}{inactv\ C\ x}\,(inact_I)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^+)}{actv\ C\ x \vee inactv\ C\ x}\,(\vee^-)(2)$$

B.6 Proofs for Section 3.7: Partial relations semantics

Proposition 3.7.1 For arbitrary chart C, and bindings z^{T_3}, we have,

$$\frac{z \mathrel{\dot\in} C \quad z \star z_a \mathrel{\dot\in} \delta_C,\ actv\ C\ z \star z_a \vdash P}{P}\ (Z_s^-)$$

$$\frac{z \star x_a \mathrel{\dot\in} \delta_C \quad actv\ C\ z \star x_a}{z \mathrel{\dot\in} C}\ (Z_s^+)$$

where $[\![C]\!]^T_{z_C}$, $[\![\delta_C]\!]^{T^a}_{z_C}$, $T \preceq T_3$, $T^{act} \not\preceq T_3$, and the usual conditions hold for $z_a^{T^{act}}$ and P.

Proof 3.7.1

For (Z_s^-) we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x = z \restriction T}\ (df) \quad \dfrac{\dfrac{}{x_1 \star z_a \in \delta_C}\ 1 \quad \dfrac{}{x_1 \star z_a =_T x}\ 1}{x \star z_a \in \delta_C}}{z \restriction T \star z_a \in \delta_C} \quad T^{act} \not\preceq T_3}{z \star z_a \mathrel{\dot\in} \delta_C} \quad \overset{\zeta_1}{\vdots}\ actv\ C\ z \star z_a}{P}\ (ass)}{\exists\, z_1^{T^a} \bullet z_1 =_T x \wedge actv\ C\ z_1 \wedge z_1 \in \delta_C}\ (df)}{P}\ (\exists^-)(1)$$

where $x =_{def} z \restriction T$.

ζ_1 is:

$$\dfrac{\dfrac{}{actv\ C\ x \star z_a}\ 1 \quad \dfrac{\dfrac{}{x = z \restriction T}\ (df)}{x \star z_a = z \restriction T \star z_a} \quad T^{act} \not\preceq T_3}{x \star z_a =_{T^a} z \star z_a} \quad \dfrac{}{\Delta U \preceq T^a}\ (df) \quad \dfrac{}{T^{act} \preceq T^a}\ (df)}{actv\ C\ z \star z_a}\ (act_I)$$

For (Z_s^+) we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z \restriction T \star x_a =_T z}\ (df) \quad \overset{\zeta_1}{\vdots}\ actv\ C\ (z \restriction T \star x_a) \quad \dfrac{z \star x_a \mathrel{\dot\in} \delta_C}{z \restriction T \star x_a \in \delta_C}}{z \restriction T \star x_a =_T z \wedge actv\ C\ (z \restriction T \star x_a) \wedge z \restriction T \star x_a \in \delta_C}\ (\wedge^+)}{\exists\, z_1^{T^a} \bullet z_1 =_T z \wedge actv\ C\ z_1 \wedge z_1 \in \delta_C}\ (df)}{z \mathrel{\dot\in} C}$$

where ζ_1 is:

$$\dfrac{actv\ C\ z \star x_a \quad \dfrac{}{z \restriction T \star x_a =_{\Delta U \curlyvee T^{act}} z \star x_a}\ (df)}{actv\ C\ (z \restriction T \star x_a)}\ (act_I)$$

Before continuing the proofs of the logic rules for the partial relations semantics we introduce another property of the transition model. In the given transition model, for an arbitrary chart C, there is typically several bindings that model a single transition that a chart can make. This is an important property of the model that allows us to build a general model of the composition operators. One of the reasons for this is that there is typically several bindings for which the predicate *actv* $C\ z$ evaluates to true. That is, there are several possible values for the observation *act* (*i.e.* sets of chart names) for which a chart is considered active. Essentially, the property allows us to show that there is a separate binding in the transition model for each set of chart names that represent a state in which chart C is active. The similar property holds for the inactive state.

Lemma B.6.2 For arbitrary chart C and bindings z^{T_3}, $z_a^{T^{act}}$ and $z_b^{T^{act}}$, we have,

$$\frac{z \star z_a \mathrel{\dot\in} \delta_C \quad actv\ C\ z \star z_a \quad actv\ C\ z \star z_b}{z \star z_b \mathrel{\dot\in} \delta_C}\ (act_{IV})$$

$$\frac{z \star z_a \mathrel{\dot\in} \delta_C \quad inactv\ C\ z \star z_a \quad inactv\ C\ z \star z_b}{z \star z_b \mathrel{\dot\in} \delta_C}\ (inact_{IV})$$

where $[\![\delta_C]\!]^{\mathbb{P}\,T}_{z_C}$ and $T \preceq T_3$

Proof B.6.2

Like the respective proofs for (act_I) to (act_{III}) this property is easily proved using structural induction over the language operators.

Proposition 3.7.2 Given arbitrary charts C_1, C_2 and $C = C_1 \mid \Psi \mid C_2$, and bindings $z_1^{U_1}$, $z_2^{U_2}$, $x_c^{V^{io}}$, $u_1^{V_1^{io}}$, $u_2^{V_2^{io}}$, $y_1^{U_1}$, $y_2^{U_2}$, $v_c^{V^{io}}$, $w_1^{V_1^{io}}$ and $w_2^{V_2^{io}}$, for arbitrary o_1 and o_2 we have,

$$\frac{\begin{array}{ll} & z_1 \star x_1 \star y_1' \star v_1' \mathrel{\dot\in} C_1, \\ & z_2 \star x_2 \star y_2' \star v_2' \mathrel{\dot\in} C_2, \\ z_1 \star z_2 \star x_c \star y_1' \star y_2' \star v_c' \mathrel{\dot\in} C & x_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1}, \\ & x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}, \\ & v_c.o_C = v_1.o_{C_1} \cup v_2.o_{C_2} \vdash P \end{array}}{P}\ (Z_{|-|}^{-})$$

$$\frac{\begin{array}{l} z_1 \star u_1 \star y_1' \star w_1' \mathrel{\dot\in} C_1 \\ z_2 \star u_2 \star y_2' \star w_2' \mathrel{\dot\in} C_2 \\ u_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1} \\ u_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2} \\ v_c.o_C = w_1.o_{C_1} \cup w_2.o_{C_2} \end{array}}{z_1 \star z_2 \star x_c \star y_1' \star y_2' \star v_c' \mathrel{\dot\in} C}\ (Z^+_{|-|})\ (\dagger)$$

where $fb_{v_c} =_{def} v_c.o_C \cap \Psi$, $[\![C]\!]^{\mathbb{P}\,T}_{z_C}$, $[\![C_1]\!]^{\mathbb{P}\,T_1}_{z_C}$, $[\![C_2]\!]^{\mathbb{P}\,T_2}_{z_C}$, and the usual conditions hold for $x_1^{V_1^{io}}$, $x_2^{V_2^{io}}$, $v_1^{V_1^{io}}$, $v_2^{V_1^{io}}$ and P. This rule contains a side-condition (labelled †) which requires that we can show that $\forall\, z^{T^{io}} \bullet \exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a$.

Proof 3.7.2

For $(Z^-_{|-|})$, assuming
$v_1^{V_1^{io}} \doteq \langle\!|\ i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}\ |\!\rangle$, $u_1^{V_1^{io}} \doteq \langle\!|\ o_{C_1} \Rrightarrow o_1\ |\!\rangle$,
$v_2^{V_2^{io}} \doteq \langle\!|\ i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}\ |\!\rangle$, $u_2^{V_2^{io}} \doteq \langle\!|\ o_{C_2} \Rrightarrow o_2\ |\!\rangle$,
$x_1^{T_1^{io}} = z_1 \star v_1 \star y_1' \star u_1'$, $x_2^{T_2^{io}} = z_2 \star v_2 \star y_2' \star u_2'$ and
$z^{T^{io}} = z_1 \star z_2 \star x_c \star y_1' \star y_2' \star v_c'$, we have,

$$\dfrac{\dfrac{\dfrac{}{x_1 \mathrel{\dot\in} C_1}\,\zeta_1 \quad \dfrac{}{x_2 \mathrel{\dot\in} C_2}\,\zeta_2 \quad \dfrac{}{v_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1}}\,\zeta_3}{P}\,(ass) \quad \dfrac{\dfrac{}{v_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}}\,\zeta_4 \quad \dfrac{}{v_c.o_C = u_1.o_{C_1} \cup u_2.o_{C_2}}\,\zeta_5}{z \star z_a \mathrel{\dot\in} \delta_C}\,1}{P}\,(|\,-\,|^-)(2)}{P}\,(Z^-_s)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{z \star z_a \star v_1 \restriction V_1^i \star u_1' \restriction V_1^{o\prime} \mathrel{\dot\in} \delta_{C_1}}\,2}{x_1 \restriction T_1 \star z_a \in \delta_{C_1}}}{x_1 \star z_a \mathrel{\dot\in} \delta_{C_1}} \quad \dfrac{\dfrac{\dfrac{}{actv\ C\ z \star z_a}\,1}{actv\ C_1\ z \star z_a}\,(df) \quad \dfrac{}{x_1 \star z_a =_{(\Delta U_1 \curlyvee T^{act})} z \star z_a}\,(df)}{actv\ C_1\ x_1 \star z_a}\,(act_I)}{x_1 \mathrel{\dot\in} C_1}\,(Z^+_s)$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{z \star z_a \star v_2 \restriction V_2^i \star u_2' \restriction V_2^{o\prime} \mathrel{\dot\in} \delta_{C_2}}\,2}{x_2 \restriction T_2 \star z_a \in \delta_{C_2}}}{x_2 \star z_a \mathrel{\dot\in} \delta_{C_2}} \quad \dfrac{\dfrac{\dfrac{}{actv\ C\ z \star z_a}\,1}{actv\ C_2\ z \star z_a}\,(df) \quad \dfrac{}{x_2 \star z_a =_{(\Delta U_2 \curlyvee T^{act})} z \star z_a}\,(df)}{actv\ C_2\ x_2 \star z_a}\,(act_I)}{x_2 \mathrel{\dot\in} C_2}\,(Z^+_s)$$

ζ_3 is:

$$\dfrac{\dfrac{}{fb_z = fb_{v_c}}\,(df) \quad \dfrac{}{v_1.i_{C_1} = (z.i_C \cup fb_z) \cap in_{C_1}}\,(df) \quad \dfrac{}{z.i_C = x_c.i_C}\,(df)}{v_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1}}$$

ζ_4 is:

$$\dfrac{\dfrac{}{fb_z = fb_{v_c}}\,(df) \quad \dfrac{}{v_2.i_{C_2} = (z.i_C \cup fb_z) \cap in_{C_2}}\,(df) \quad \dfrac{}{z.i_C = x_c.i_C}\,(df)}{v_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}}$$

ζ_5 is:

$$\dfrac{\dfrac{\dfrac{}{z.o'_C = o_1 \cup o_2}\,2}{v_c.o_C = o_1 \cup o_2}\,(V^{io}\text{-}df)}{v_c.o_C = u_1.o_{C_1} \cup u_2.o_{C_2}}\,(df)$$

For $(Z^+_{|-|})$, assuming,
$v_1^{V_1^i} = \langle\!|\ i_{C_1} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_1}\ |\!\rangle$,
$v_2^{V_2^i} = \langle\!|\ i_{C_2} \Rrightarrow (z.i_C \cup fb_z) \cap in_{C_2}\ |\!\rangle$,
$x_1^{T_1^{io}} = z_1 \star u_1 \star y'_1 \star w'_1$,
$x_2^{T_2^{io}} = z_2 \star u_2 \star y'_2 \star w'_2$ and
$z^{T^{io}} = z_1 \star z_2 \star x_c \star y'_1 \star y'_2 \star v'_c$, we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ \zeta}{z \star z_a \mathrel{\dot\in} \delta_C} \quad \dfrac{}{actv\ C\ z \star z_a}\,3}{z \mathrel{\dot\in} C}\,(Z_s^+) \quad \dfrac{\forall\, z^{T^{io}} \bullet \exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a}{\exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a}\,(\forall^-)}{x_2 \mathrel{\dot\in} C_2 \qquad z \mathrel{\dot\in} C}\,(\exists^-)(3)}{z \mathrel{\dot\in} C}\,(Z_s^-)(2) \qquad x_1 \mathrel{\dot\in} C_1}{z \mathrel{\dot\in} C}\,(Z_s^-)(1)$$

where ζ is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\ z \star z_a}\,3}{actv\ C\ z \star z_a \vee inactv\ C\ z \star z_a}\,(\vee^+) \quad \begin{matrix}\zeta_1\\ \vdots\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots\end{matrix} \quad \dfrac{v_c.o_C = w_1.o_{C_1} \cup w_2.o_{C_2}}{z \star z_a.o_C = w_1.o_{C_1} \cup w_2.o_{C_2}}\,(df)}{z \star z_a \mathrel{\dot\in} \delta_C}\,(|\_|^+)$$

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{x_1 \star z_{a_1} \mathrel{\dot\in} \delta_{C_1}}\,1 \quad \dfrac{\dfrac{u_1.i_{C_1} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_1}}{u_1.i_{C_1} = (z.i_C \cup fb_z) \cap in_{C_1}} \quad \dfrac{}{x_c.i_C = z.i_C}\,(df) \quad \dfrac{}{fb_{v_c} = fb_z}\,(df)}{z \star z_{a_1} \star v_1 \star w'_1 \mathrel{\dot\in} \delta_{C_1}}}{z \star z_a \star v_1 \star w'_1 \mathrel{\dot\in} \delta_{C_1}}\,(act_{IV})$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{}{actv\ C_1\ (x_1 \star z_{a_1})}\,1 \quad \dfrac{}{x_1 \star z_{a_1} =_{(\Delta U_1\ \curlyvee\ T^{act})} z \star z_{a_1} \star v_1 \star w'_1}\,(df)}{actv\ C_1\ (z \star z_{a_1} \star v_1 \star w'_1)}$$

$\zeta_{1.2}$ is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\ (z \restriction T \star z_a)}\,3}{actv\ C_1\ (z \restriction T \star z_a)}\,(df) \quad \dfrac{}{z \restriction T \star z_a =_{(\Delta U_1\ \curlyvee\ T^{act})} z \star z_a \star v_1 \star w'_1}\,(df)}{actv\ C_1\ (z \star z_a \star v_1 \star w'_1)}\,(act_I)$$

ζ_2 is:

$$\dfrac{\dfrac{}{x_2 \star z_{a_2} \dot{\in} \delta_{C_2}}\,2 \quad \dfrac{\dfrac{u_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2} \qquad \dfrac{}{x_c.i_C = z.i_C}\,(df)}{u_2.i_{C_2} = (z.i_C \cup fb_{v_c}) \cap in_{C_2}} \qquad \dfrac{}{fb_{v_c} = fb_z}\,(df)}{u_2.i_{C_2} = (z.i_C \cup fb_z) \cap in_{C_2}} \qquad \begin{matrix}\zeta_{2.1}\\ \vdots\end{matrix} \quad \begin{matrix}\zeta_{2.2}\\ \vdots\end{matrix}}{z \star z_{a_2} \star v_2 \star w_2' \dot{\in} \delta_{C_2}}\,(act_{IV})$$

$$z \star z_a \star v_2 \star w_2' \dot{\in} \delta_{C_2}$$

$\zeta_{2.1}$ is:

$$\dfrac{\dfrac{}{actv\ C_2\,(x_2 \star z_{a_2})}\,2 \quad \dfrac{}{x_2 \star z_{a_2} =_{(\Delta U_2 \curlyvee T^{act})} z \star z_{a_2} \star v_2 \star w_2'}\,(df)}{actv\ C_2\,(z \star z_{a_2} \star v_2 \star w_2')}$$

$\zeta_{2.2}$ is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\,(z \restriction T \star z_a)}\,3}{actv\ C_2\,(z \restriction T \star z_a)}\,(df) \quad \dfrac{}{z \restriction T \star z_a =_{(\Delta U_2 \curlyvee T^{act})} z \star z_a \star v_2 \star w_2'}\,(df)}{actv\ C_2\,(z \star z_a \star v_2 \star w_2')}\,(act_I)$$

Proposition 3.7.3 Given chart $C =_X [C_1]_Y$ for arbitrary C_1 and bindings z^U, $x^{V^{io}}$, y^U, $v^{V^{io}}$ and $x_1^{V_1^{io}}$, we have,

$$\dfrac{z \star x \star y' \star v' \dot{\in} C \quad x_1.i_{C_1} = x.i_C \cap in_{C_1} \quad \begin{matrix} z \star x_1 \star y' \star u_1' \dot{\in} C_1, \\ v.o_C = u_1.o_{C_1} \cap out_C \vdash P \end{matrix}}{P}\,(Z_{[]}^-)$$

$$\dfrac{z \star u_1 \star y' \star w_1' \dot{\in} C_1 \quad u_1.i_{C_1} = x.i_C \cap in_{C_1} \quad v.o_C = w_1.o_{C_1} \cap out_C}{z \star x \star y' \star v' \dot{\in} C}\,(Z_{[]}^+)\ (\dagger)$$

where $[\![C]\!]_{z_C}^{\mathbb{P}\,T}$ and $[\![C_1]\!]_{z_C}^{\mathbb{P}\,T_1}$ the usual conditions hold for $u_1^{V_1^{io}}$, and P. The side-condition $\dagger$ requires that we can show that $\forall\, z^{T^{io}} \bullet \exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a$.

Proof 3.7.3

For $(Z_{[]}^-)$, assuming
$\vdash_{z_C}^{T^{io}} = z \star x \star y' \star v'$,
$x_1^{V_1^{io}} \doteq \langle\!|\ i_{C_1} \Rrightarrow \vdash_{z_C}.i_C \cap in_{C_1}\ |\!\rangle$,
$u_1^{V_1^{io}} \doteq \langle\!|\ o_{C_1} \Rrightarrow o_1\ |\!\rangle$ and
$z_1^{T_1^{io}} = z \star x_1 \star y' \star u'$, we have,

$$\dfrac{\vdash_{z_C} \dot{\in} C \qquad \dfrac{\dfrac{}{\vdash_{z_C}.i_C = x.i_C}\,(df) \qquad \dfrac{}{x_1.i_{C_1} = \vdash_{z_C}.i_C \cap in_{C_1}}\,(df)}{x_1.i_{C_1} = x.i_C \cap in_{C_1}} \qquad \dfrac{\dfrac{}{\vdash_{z_C} \star z_a \dot{\in} \delta_C}\,1 \qquad \begin{matrix}\zeta_1\\ \vdots\\ z_1 \dot{\in} C_1\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots\\ v.o_C = u_1.o_{C_1} \cap out_C\end{matrix}}{P}\,(ass)}{P}\,(x[]_Y^-)(2)}{P}\,(Z_s^-)(1)$$

where ζ_1:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash z_c \star z_a \star \langle\!|\ i_{C_1} \Rrightarrow (\vdash z_c \star z_a).i_C \cap in_{C_1}, o_{C_1}' \Rrightarrow o_1\ |\!\rangle \mathrel{\dot\in} \delta_{C_1}}\,2}{z \star z_a \star x_1 \restriction T_1^i \star y' \star u_1' \restriction T_1^{o\prime} \in \delta_{C_1}}}{z_1 \star z_a \mathrel{\dot\in} \delta_{C_1}} \qquad \begin{array}{c}\zeta_{1.1}\\ \vdots \\ actv\ C_1\ z_1 \star z_a\end{array}}{z_1 \star z_a \mathrel{\dot\in} \delta_{C_1} \wedge actv\ C_1\ z_1 \star z_a}\,(\exists^+)}{\exists\, z_a^{T^{act}} \bullet z_1 \star z_a \mathrel{\dot\in} \delta_{C_1} \wedge actv\ C_1\ z_1 \star z_a}\,(Z_s^+)}{z_1 \mathrel{\dot\in} C_1}$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C \vdash z_c \star z_a}\,1}{actv\ C_1 \vdash z_c \star z_a}\,(df) \qquad \dfrac{}{\vdash z_c \star z_a =_{(\Delta U_1 \curlyvee T^{act})} z_1 \star z_a}\,(df)}{actv\ C_1\ z_1 \star z_a}\,(act_I)$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{\vdash z_c.o_C = o_1 \cap out_C}\,2}{v.o_C = o_1 \cap out_C}\,(df) \qquad \dfrac{}{u_1.o_{C_1} = o_1}\,(df)}{v.o_C = u_1.o_{C_1} \cap out_C}$$

For $(Z_{[]}^+)$, assuming
$z_1^{T^{io}} = z \star u_1 \star y' \star w_1'$,
$v_1^{V_1^i} = \langle\!|\ i_{C_1} \Rrightarrow x.i_C \cap in_{C_1}\ |\!\rangle$,
$u^{T^{io}} = z \star x \star y' \star v'$, we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{actv\ C\ u \star z_a}\,2 \qquad \begin{array}{c}\zeta_1\\ \vdots \\ u \star z_a \mathrel{\dot\in} \delta_C\end{array}}{u \mathrel{\dot\in} C}\,(Z_s^+) \qquad \dfrac{\forall\, u^{T^{io}} \bullet \exists\, z_a^{T^{act}} \bullet actv\ C\ u \star z_a}{\exists\, z_a^{T^{act}} \bullet actv\ C\ z \star z_a}\,(\forall^-)}{z_1 \mathrel{\dot\in} C_1 \qquad u \mathrel{\dot\in} C}\,(\exists^-)(2)}{u \mathrel{\dot\in} C}\,(Z_s^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{actv\ C\ u \star z_a}\,2}{actv\ C\ u \star z_a \vee inactv\ C\ u \star z_a}\,(\vee^+) \qquad \begin{array}{c}\zeta_2\\ \vdots \\ v.o_C = w_1.o_{C_1} \cap out_C\end{array} \qquad \dfrac{}{v.o_C = u \star z_a.o_C}\,(df)}{u \star z_a.o_C = w_1.o_{C_1} \cap out_C}\,(x[]_Y^+)}{u \star z_a \mathrel{\dot\in} \delta_C}$$

ζ_2 is:

$$\dfrac{\dfrac{}{z_1 \star z_b \mathrel{\dot\in} \delta_{C_1}}\,1 \qquad \dfrac{}{actv\ C_1\ z_1 \star z_b}\,1 \qquad \dfrac{\dfrac{\dfrac{}{actv\ C\ u \star z_a}\,2}{actv\ C_1\ u \star z_a}\,(df) \qquad \dfrac{}{u \star z_a =_{(\Delta U_1 \curlyvee T_1^{act})} z_1 \star z_a}\,(df)}{actv\ C_1\ z_1 \star z_a}\,(act_I)}{z_1 \star z_a \mathrel{\dot\in} \delta_{C_1}}\,(act_{IV}) \qquad \begin{array}{c}u_1.i_{C_1} = \\ x.i_C \cap in_{C_1}\end{array}$$

$$\dfrac{z \star v_1 \star y' \star w_1' \star z_a \mathrel{\dot\in} \delta_{C_1}}{u \star z_a \star v_1 \star w_1' \mathrel{\dot\in} \delta_{C_1}}$$

Proposition 3.7.4 Given $M_S = (Dec\ (\omega\ M)\ by\ \{(S, \omega\ S)\})$, for arbitrary S, $\omega\ M = (M, \Sigma, \sigma, \Psi, \delta)$, and $MS_\Psi = [\![\omega\ M \mid \Psi \mid \omega\ S]\!]_{Z_t}$, then for arbitrary z^{T_3} we have,

$$\frac{z.c_M = S \vee z.c'_M = S \quad z \mathrel{\dot\in} \delta_{M_S}}{z \mathrel{\dot\in} \delta_{MS_\Psi}}\ (M^-_{S\,V})$$

$$\frac{z.c_M = S \vee z.c'_M = S \quad z \mathrel{\dot\in} \delta_{MS_\Psi}}{z \mathrel{\dot\in} \delta_{M_S}}\ (M^+_{S\,II})$$

where $[\![\delta_{M_S}]\!]^{\mathbb{P}\,T^a}_{z_C}$ and $T^a \preceq T_3$.

Proof 3.7.4

Given that the following holds from the respective definitions of the predicates *act* and *inact*,

$$\frac{z.c_M = S \vee z.c'_M = S \quad actv\ MS_\Psi\ z \vee inactv\ MS_\Psi\ z}{actv\ M_S\ z \vee inactv\ M_S\ z}$$

The proof of $(M^-_{S\,II})$ follows trivially using the rules (M^-_S) and $(|\_|^+)$.
Similarly given,

$$\frac{z.c_M = S \vee z.c'_M = S \quad actv\ M_S\ z \vee inactv\ M_S\ z}{actv\ MS_\Psi\ z \vee inactv\ MS_\Psi\ z}$$

The rule $(M^+_{S\,II})$ holds trivially using $(|\_|^-)$ and (M^+_S).

B.7 Proofs for Section 5.2: The refinement semilattice

Proposition 5.2.1 For arbitrary charts A,C, and infinite sequences i and o we have,

$$\frac{C \sqsupseteq_b A}{in_C = in_A}\ (\sqsupseteq^-_{\beta\, I}) \qquad \frac{C \sqsupseteq_b A}{out_C = out_A}\ (\sqsupseteq^-_{\beta\, II})$$

$$\frac{C \sqsupseteq_b A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\beta\, III})$$

$$\frac{in_C = in_A \quad out_C = out_A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \ \vdash\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}{C \sqsupseteq_b A}\ (\sqsupseteq^+_{\beta\, I})$$

$$\frac{}{A \sqsupseteq_b A}\ (\sqsupseteq^+_{\beta\, II})$$

Proof 5.2.1

The proofs of $(\sqsupseteq^-_{\beta\, I})$, $(\sqsupseteq^-_{\beta\, II})$ and $(\sqsupseteq^-_{\beta\, III})$ are trivial from Definition 5.2.1 using $(\wedge^-)$. Now for the introduction rule. For $(\sqsupseteq^+_{\beta\, I})$ we have,

$$\frac{\dfrac{in_C = in_A \quad out_C = out_A \quad \dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,1}{\vdots}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\Rightarrow^+)(1)}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\wedge^+)}{in_C = in_A \wedge out_C = out_A \wedge (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (df)}{C \sqsupseteq_b A}$$

And for $(\sqsupseteq^+_{\beta\, II})$ we have,

$$\frac{\dfrac{}{in_A = in_A} \quad \dfrac{}{out_A = out_A} \quad \dfrac{\dfrac{}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,1}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}}{A \sqsupseteq_b A}\ (\sqsupseteq^+_{\beta\, I})(1)$$

B.8 Proofs for Section 5.3: Interface refinement and $\mathcal{R}$

B.8.1 Input interface refinement

Proposition 5.3.1 For arbitrary charts A and C, signal set ins and infinite sequences i, i' and o we have,

$$\frac{C \approx_{\mathcal{I}} A}{out_C = out_A}\ (\sqsupseteq^-_{\mathcal{I}\,I}) \qquad \frac{C \approx_{\mathcal{I}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{I}\,II})$$

$$\frac{C \approx_{\mathcal{I}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{I}\,III}) \qquad \frac{C \approx_{\mathcal{I}} A \quad in_C = in_A}{[\![C]\!]^{\omega}_x = [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{I}\,IV})$$

$$\frac{C \approx_{\mathcal{I}} B \quad B \approx_{\mathcal{I}} A}{C \approx_{\mathcal{I}} A}\ (\sqsupseteq^-_{\mathcal{I}\,V}) \qquad \frac{C \approx_{\mathcal{I}} A \quad ins = in_A \cap in_C}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{I}\,VI})$$

$$\frac{C \approx_{\mathcal{I}} A}{A \approx_{\mathcal{I}} C}\ (\sqsupseteq^-_{\mathcal{I}\,VII}) \qquad \frac{in_A \subseteq ins \qquad \begin{array}{l} in_B = ins, \\ out_B = out_A, \\ B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{I}\,VIII})$$

$$\frac{ins \subset in_A \qquad \begin{array}{l} (i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x \Leftrightarrow \\ (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_x \end{array} \qquad \begin{array}{l} in_B = ins, \\ out_B = out_A, \\ B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{I}\,IX})$$

where we assume the usual conditions for B and P in $(\sqsupseteq^-_{\mathcal{I}\,6})$ and $(\sqsupseteq^-_{\mathcal{I}\,7})$.

Proof 5.3.1

The rule $(\sqsupseteq^-_{\mathcal{I}\,I})$ follows trivially from Definition 5.3.1 using $(\wedge^-)$.

For $(\sqsupseteq^-_{\mathcal{I}\,II})$ we have,

$$\frac{\begin{array}{c} \zeta_1 \\ \vdots \\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_C)}) \in [\![A]\!]^{\omega}_x \end{array} \quad \frac{C \approx_{\mathcal{I}} A}{out_C = out_A}\ (\sqsupseteq^-_{\mathcal{I}\,I})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}$$

where ζ_1 is:

$$\frac{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \quad \frac{C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_C)}) \in [\![A]\!]^{\omega}_x}\ (\approx_{\mathcal{I}}\text{-}df)}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_C)}) \in [\![A]\!]^{\omega}_x}\ (\Rightarrow^-)$$

For $(\sqsupseteq^-_{\mathcal{I}\,III})$ we have,

$$\dfrac{\begin{array}{c}\zeta_1\\ \vdots\\ (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_A)}) \in [\![C]\!]^{\omega}_x\end{array} \quad \dfrac{C \approx_{\mathcal{I}} A}{out_C = out_A}\,(\sqsupseteq^-_{\mathcal{I}\,I})}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}$$

where ζ_1 is:

$$\dfrac{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x \quad \dfrac{C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_A)}) \in [\![C]\!]^{\omega}_x}\,(\approx_{\mathcal{I}}\text{-}df)}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_A)}) \in [\![C]\!]^{\omega}_x}\,(\Rightarrow^-)$$

Along with the rules $(\sqsupseteq^-_{\mathcal{I}\,II})$ and $(\sqsupseteq^-_{\mathcal{I}\,III})$ we introduce and prove two useful variants of these rules. We give a proof of these rule here in order to freely use these variants in the following.

Firstly, we have that,

$$\dfrac{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_x \quad \dfrac{C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x}\,(\approx_{\mathcal{I}}\text{-}df)}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x}\,(\Rightarrow^-)$$

Correspondingly,

$$\dfrac{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x \quad \dfrac{C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_x \Rightarrow (i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_x}\,(\approx_{\mathcal{I}}\text{-}df)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_x}\,(\Rightarrow^-)$$

Now for $(\sqsupseteq^-_{\mathcal{I}\,IV})$ we have,

$$\dfrac{\dfrac{\dfrac{\begin{array}{cc}\zeta_1 & \zeta_3\\ \vdots & \vdots\end{array}}{(i,o) \in [\![C]\!]^{\omega}_x \Leftrightarrow (i,o) \in [\![A]\!]^{\omega}_x}\,(\Rightarrow^+)(1)}{\forall\, i, o \bullet (i,o) \in [\![C]\!]^{\omega}_x \Leftrightarrow (i,o) \in [\![A]\!]^{\omega}_x}\,(\forall^+)}{[\![C]\!]^{\omega}_x = [\![A]\!]^{\omega}_x}$$

where ζ_1 is:

$$\dfrac{\begin{array}{c}\zeta_2\\ \vdots\end{array} \quad \dfrac{\dfrac{\dfrac{}{(i,o) \in [\![C]\!]^{\omega}_x}\,1}{i = i_{\triangleright(in_C)}}\,([\![\cdot]\!]^{\omega}_x\text{-}df) \quad in_A = in_C}{i = i_{\triangleright(in_A)}} \quad \dfrac{\dfrac{\dfrac{}{(i,o) \in [\![C]\!]^{\omega}_x}\,1}{o = o_{\triangleright(out^{\perp}_C)}}\,([\![\cdot]\!]^{\omega}_x\text{-}df) \quad \dfrac{C \approx_{\mathcal{I}} A}{out_A = out_C}\,(\sqsupseteq^-_{\mathcal{I}\,I})}{out^{\perp}_A = out^{\perp}_C}}{o = o_{\triangleright(out^{\perp}_A)}}}{(i,o) \in [\![A]\!]^{\omega}_x}$$

$$\dfrac{}{(i,o) \in [\![C]\!]^{\omega}_x \Rightarrow (i,o) \in [\![A]\!]^{\omega}_x}\,(\Rightarrow^+)(1)$$

and ζ_2 is:

$$\dfrac{C \approx_{\mathcal{I}} A \quad \dfrac{\dfrac{}{(i,o) \in [\![C]\!]^{\omega}_x}\,1}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,([\![\cdot]\!]^{\omega}_x\text{-}df)}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{I}\,II})$$

The proof labelled ζ_3 is symmetric to ζ_1 using $(\sqsupseteq^-_{\mathcal{I}\,III})$.

For $(\sqsupseteq^-_{\mathcal{I}\,V})$ we have,

$$\dfrac{\dfrac{\begin{array}{cc}\zeta_1 & \zeta_3\\ \vdots & \vdots\\ (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow & (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x \Rightarrow\\ (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x & (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x\end{array}}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (\wedge^+)}{C \approx_{\mathcal{I}} A}\ ([\![.]\!]^\omega_x\text{-}df)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\,1}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\ ([\![.]\!]^\omega_x\text{-}df) \qquad C \approx_{\mathcal{I}} B}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II}) \qquad B \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II}) \qquad \begin{array}{c}\zeta_2\\ \vdots\end{array}}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (\Rightarrow^+)(1)$$

where ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\,1}{o = o_{\triangleright(out^\perp_C)}}\ ([\![.]\!]^\omega_x\text{-}df) \qquad \dfrac{\dfrac{C \approx_{\mathcal{I}} B}{out_C = out_B}\ (\sqsupseteq^-_{\mathcal{I}\,I}) \quad \dfrac{B \approx_{\mathcal{I}} A}{out_B = out_A}\ (\sqsupseteq^-_{\mathcal{I}\,I})}{\dfrac{out_C = out_A}{out^\perp_C = out^\perp_A}}}{o = o_{\triangleright(out^\perp_A)}}$$

Once again, the proof labelled ζ_3 is symmetric to ζ_1 using $(\sqsupseteq^-_{\mathcal{I}\,III})$.

For $(\sqsupseteq^-_{\mathcal{I}\,VI})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x}\,1}{((i_{\triangleright(in_C)})_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (_{\triangleright(.)}\text{-}df) \qquad C \approx_{\mathcal{I}} A}{((i_{\triangleright(in_C)})_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,III})}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\ (_{\triangleright(.)}\text{-}df) \qquad C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II})}{(i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x \Rightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (\Rightarrow^+)(1) \qquad \begin{array}{c}\zeta_1\\ \vdots\end{array}}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x}$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{ins = in_A \cap in_C}{ins \subseteq in_A} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{c}\zeta_2\\ \vdots\\ (i_{\triangleright(ins)}, o) \in [\![C]\!]^\omega_x\end{array} \qquad \dfrac{ins = in_A \cap in_C}{ins \subseteq in_C}}{((i_{\triangleright(ins)})_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\ (_{\triangleright(.)}\text{-}df) \qquad C \approx_{\mathcal{I}} A}{((i_{\triangleright(ins)})_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II})}{(i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x}\ (_{\triangleright(.)}\text{-}df)}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^\omega_x \Rightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^\omega_x}\ (\Rightarrow^+)(2)$$

and ζ_2 is:

$$\dfrac{ins = in_A \cap in_C \qquad \dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x}}\,{}^{2}}{((i_{\triangleright(in_A)})_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x}}\,(\triangleright(.)\text{-}df) \qquad C \approx_{\mathcal{I}} A}{((i_{\triangleright(in_A)})_{\triangleright(in_C)}, o) \in [\![C]\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\mathcal{I}}\,III)}{(i_{\triangleright(ins)}, o) \in [\![C]\!]^{\omega}_{x}}\,(\triangleright(.)\text{-}df)$$

The proof for $(\sqsupseteq^{-}_{\mathcal{I}}\,VII)$ is trivial given the definition of $\approx_{\mathcal{I}}$ (Definition 5.3.1) and that $\Leftrightarrow$ is commutative.

For $(\sqsupseteq^{-}_{\mathcal{I}}\,VIII)$ we have,

$$\dfrac{\dfrac{in_A \subseteq ins}{\exists B \bullet in_B = ins \wedge out_B = out_A \wedge B \approx_{\mathcal{I}} A}\,\dagger \qquad \dfrac{\overline{in_B = ins}\,{}^{1} \quad \overline{out_B = out_A}\,{}^{1} \quad \overline{B \approx_{\mathcal{I}} A}\,{}^{1}}{P}}{P}\,(\exists^{-})(1)$$

We still need to prove the step labelled † above. That is, for any chart A and set of signals *ins*, where $in_A \subseteq ins$, there exists a chart B that has input interface *ins*, an output interface equal to out_A and is an interface refinement of chart A. To do so we give an informal account of an algorithm that takes an arbitrary chart and extends its interface one signal at a time to give a new chart that has exactly the properties required of B. This algorithm is guaranteed to terminate because of the assumption that the set *ins* is finite.

Briefly this algorithm is as follows. For the first new signal, say x where x is not already in in_A, we take chart A and add to it a copy of each of the existing transitions. We then add to the trigger of one of these transitions the signal x and to the other the negation of the signal -x. This gives us a new chart A_1 such that $A_1 \approx_{\mathcal{I}} A$. We then proceed to add the next new signal, in the same fashion, to the chart A_1 giving yet another chart A_2. The transitivity of $\approx_{\mathcal{I}}$ gives us that $A_2 \approx_{\mathcal{I}} A$. Continuing this chain until each of the new signals in *ins* are accounted for, will give us the chart B as required.

For $(\sqsupseteq^{-}_{\mathcal{I}}\,IX)$ we have,

$$\dfrac{\dfrac{ins \subset in_A \qquad (i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x}}{\exists B \bullet in_B = ins \wedge out_B = out_A \wedge B \approx_{\mathcal{I}} A}\,\dagger \qquad \dfrac{\overline{in_B = ins}\,{}^{1} \quad \overline{out_B = out_A}\,{}^{1} \quad \overline{B \approx_{\mathcal{I}} A}\,{}^{1}}{P}}{P}\,(\exists^{-})(1)$$

Again we prove the step labelled † by giving an informal account of an algorithm that creates an appropriate chart B from the given chart A. This algorithm is as follows. Take the specific chart A and remove from it any reference to the signals (*i.e.* positive and / or negative) that are not in the set *ins*. Ensure the input interface of this new chart is equal to *ins*. Because $(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x}$ the resulting chart will have exactly the attributes required to prove the existence of the chart B.

Now for the introduction rules:

$$\frac{out_C = out_A \qquad (i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega}{C \approx_{\mathcal{I}} A} \; (\sqsupseteq^+_{\mathcal{I}\,I})$$

$$\frac{[\![C]\!]_x^\omega = [\![A]\!]_x^\omega}{C \approx_{\mathcal{I}} A} \; (\sqsupseteq^+_{\mathcal{I}\,II})$$

Proof 5.3.1 (*continued*)

The proof of $(\sqsupseteq^+_{\mathcal{I}\,I})$ is split into two cases as follows.

Case 1: Assuming $o = o_{\triangleright(out_C^\perp)}$ we have,

$$\dfrac{\dfrac{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega \quad \dfrac{out_C = out_A}{out_C^\perp = out_A^\perp}}{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_C^\perp)}) \in [\![A]\!]_x^\omega \qquad out_C = out_A}\,(\wedge^+)}{\dfrac{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega \wedge out_C = out_A}{C \approx_{\mathcal{I}} A}\,(\approx_{\mathcal{I}}\text{-}df)}$$

Case 2: Assuming $o \neq o_{\triangleright(out_C^\perp)}$ we have,

$$\dfrac{\dfrac{\dfrac{o \neq o_{\triangleright(out_C^\perp)}}{(i_{\triangleright(in_C)}, o) \notin [\![C]\!]_x^\omega}\,([\![.]\!]_x^\omega\text{-}df) \qquad \dfrac{o \neq o_{\triangleright(out_C^\perp)} \quad out_C = out_A}{o \neq o_{\triangleright(out_A^\perp)}}}{(i_{\triangleright(in_A)}, o) \notin [\![A]\!]_x^\omega}\,([\![.]\!]_x^\omega\text{-}df)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega}\,(\wedge^+) \qquad out_C = out_A}{\dfrac{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega \wedge out_C = out_A}{C \approx_{\mathcal{I}} A}\,(\approx_{\mathcal{I}}\text{-}df)}\,(\wedge^+)$$

And finally for $(\sqsupseteq^+_{\mathcal{I}\,II})$,

$$\dfrac{\dfrac{\dfrac{[\![C]\!]_x^\omega = [\![A]\!]_x^\omega}{in_C = in_A} \qquad \dfrac{\dfrac{[\![C]\!]_x^\omega = [\![A]\!]_x^\omega}{\forall\, i, o \bullet (i, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i, o) \in [\![A]\!]_x^\omega}\,(\forall^-)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_C)}, o) \in [\![A]\!]_x^\omega} \qquad \dfrac{[\![C]\!]_x^\omega = [\![A]\!]_x^\omega}{out_C^\perp = out_A^\perp}\,([\![.]\!]_x^\omega\text{-}df)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega \qquad out_C = out_A}\,(\wedge^+)}{\dfrac{(i_{\triangleright(in_C)}, o) \in [\![C]\!]_x^\omega \Leftrightarrow (i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega \wedge out_C = out_A}{C \approx_{\mathcal{I}} A}\,(\approx_{\mathcal{I}}\text{-}df)}$$

Lemma 5.3.2 For arbitrary charts A and C, traces i and o, and signal set ins,

$$\frac{in_A \subseteq ins}{\exists\, B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}$$

$$\frac{ins \subset in_A}{(\exists\, B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A) \Leftrightarrow ((i_{\triangleright(in_A)}, o) \in [\![A]\!]_x^\omega \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]_x^\omega)}$$

$$\frac{C \approx_{\mathcal{I}} A}{\exists\, B \bullet in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}$$

Proof 5.3.2

For the first property in this lemma we have:

$$\dfrac{in_A \subseteq ins \quad \dfrac{\dfrac{\overline{in_B = ins}^{\,1} \quad \overline{B \approx_{\mathcal{I}} A}^{\,1}}{in_B = ins \wedge B \approx_{\mathcal{I}} A}\,(\wedge^+)}{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}\,(\exists^+)}{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}\,(\sqsupseteq^-_{\mathcal{I}\,VIII})(1)$$

The second property requires that we prove an implication in both directions. For the $\Rightarrow$ direction we have:

$$\dfrac{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A \quad \dfrac{\dfrac{\dfrac{\overline{ins = in_B}^{\,1} \quad ins \subset in_A}{in_B \subset in_A}}{in_B = in_A \cap in_B} \quad \overline{ins = in_B}^{\,1}}{ins = in_A \cap in_B}}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x}}\,(\sqsupseteq^-_{\mathcal{I}\,VI})}{(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x}}\,(\exists^-)(1)$$

For the $\Leftarrow$ direction we have:

$$\dfrac{ins \subset in_A \quad (i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(ins)}, o) \in [\![A]\!]^{\omega}_{x} \quad \dfrac{\overline{in_B = ins}^{\,1} \quad \overline{B \approx_{\mathcal{I}} A}^{\,1}}{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}}{\exists B \bullet in_B = ins \wedge B \approx_{\mathcal{I}} A}\,(\sqsupseteq^-_{\mathcal{I}\,IX})(1)$$

Finally for the third property we split the proof into two cases.

Case 1: Assuming $in_A \subseteq in_C$ we have,

$$\dfrac{\dfrac{\dfrac{in_A \subseteq in_C}{in_A = (in_C \cap in_A)} \quad C \approx_{\mathcal{I}} A \quad \dfrac{\overline{[\![A]\!]^{\omega}_{x} = [\![A]\!]^{\omega}_{x}}}{A \approx_{\mathcal{I}} A}\,(\sqsupseteq^+_{\mathcal{I}\,II})}{in_A = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} A \wedge A \approx_{\mathcal{I}} A}\,(\wedge^+)}{\exists B \bullet in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}\,(\exists^+)$$

Case 2: Assuming $\neg\,(in_A \subseteq in_C)$,

$$\dfrac{\neg\,(in_A \subseteq in_C)}{in_C \cap in_A \subset in_A} \quad \dfrac{C \approx_{\mathcal{I}} A \quad \overline{in_C \cap in_A = in_C \cap in_A}\,(\sqsupseteq^-_{\mathcal{I}\,VI}) \quad \zeta_1}{\begin{array}{c}(i_{\triangleright(in_A)}, o) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow \\ (i_{\triangleright(in_A \cap in_C)}, o) \in [\![A]\!]^{\omega}_{x}\end{array}}}{\exists B \bullet in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}\,(\sqsupseteq^-_{\mathcal{I}\,IX})(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\overline{in_B = (in_C \cap in_A)}^{\,1} \quad \dfrac{C \approx_{\mathcal{I}} A \quad \dfrac{\overline{B \approx_{\mathcal{I}} A}^{\,1}}{A \approx_{\mathcal{I}} B}\,(\sqsupseteq^-_{\mathcal{I}\,VII})}{C \approx_{\mathcal{I}} B}\,(\sqsupseteq^-_{\mathcal{I}\,V}) \quad \overline{B \approx_{\mathcal{I}} A}^{\,1}}{in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}\,(\wedge^+)}{\exists B \bullet in_B = (in_C \cap in_A) \wedge C \approx_{\mathcal{I}} B \wedge B \approx_{\mathcal{I}} A}\,(\exists^+)$$

Lemma 5.3.3 For arbitrary charts A and C we have,

$$\frac{\det A \quad C \approx_{\mathcal{I}} A}{\det C}$$

Proof 5.3.3

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\det A}{\forall i,o,o' \bullet (i,o) \in [\![A]\!]^{\omega}_{x} \wedge (i,o') \in [\![A]\!]^{\omega}_{x} \Rightarrow o = o'}\,(df)}{(i_{\triangleright(in_A)},o) \in [\![A]\!]^{\omega}_{x} \wedge (i_{\triangleright(in_A)},o') \in [\![A]\!]^{\omega}_{x} \Rightarrow o = o'}\,(\forall^{-})}\,\zeta_1}{o = o'}\,(\Rightarrow^{-})}{(i,o) \in [\![C]\!]^{\omega}_{x} \wedge (i,o') \in [\![C]\!]^{\omega}_{x} \Rightarrow o = o'}\,(\Rightarrow^{+})(1)}{\forall i,o,o' \bullet (i,o) \in [\![C]\!]^{\omega}_{x} \wedge (i,o') \in [\![C]\!]^{\omega}_{x} \Rightarrow o = o'}\,(\forall^{+})}{\det C}\,(df)$$

where ζ_1 is,

$$\dfrac{\dfrac{\dfrac{\overline{(i,o) \in [\![C]\!]^{\omega}_{x}}\,1}{(i_{\triangleright(in_C)},o) \in [\![C]\!]^{\omega}_{x}} \quad C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)},o) \in [\![A]\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\mathcal{I}}\,II) \quad \dfrac{\dfrac{\overline{(i,o') \in [\![C]\!]^{\omega}_{x}}\,1}{(i_{\triangleright(in_C)},o') \in [\![C]\!]^{\omega}_{x}} \quad C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)},o') \in [\![A]\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\mathcal{I}}\,II)}{(i_{\triangleright(in_A)},o) \in [\![A]\!]^{\omega}_{x} \wedge (i_{\triangleright(in_A)},o') \in [\![A]\!]^{\omega}_{x}}$$

Lemma 5.3.4 For arbitrary charts A, B and C we have,

$$\frac{in_A \subset in_C \quad C \approx_{\mathcal{I}} B \quad B \sqsupseteq_b A}{\exists B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{I}} A}$$

Proof 5.3.4

$$\dfrac{\dfrac{\dfrac{in_A \subset in_C}{in_A \subseteq in_C}\,(\subseteq\text{-}df) \quad \dfrac{\dfrac{\dfrac{\overline{in_{B'} = in_C}\,1 \quad \begin{matrix}\zeta_1\\ \vdots\\ out_{B'} = out_C\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots\\ (i_{\triangleright(in_{B'})},o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_{x}\end{matrix}}{C \sqsupseteq_b B'}\,(\sqsupseteq^{+}_{\beta}\,I)(2) \quad \overline{B' \approx_{\mathcal{I}} A}\,1}{C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{I}} A}\,(\wedge^{+})}{\exists B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{I}} A}\,(\exists^{+})}{\exists B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{I}} A}\,(\sqsupseteq^{-}_{\mathcal{I}}\,VIII)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{B \sqsupseteq_b A}{out_B = out_A}\,(\sqsupseteq^{-}_{\beta}\,II) \quad \dfrac{C \approx_{\mathcal{I}} B}{out_C = out_B}\,(\sqsupseteq^{-}_{\mathcal{I}}\,I)}{out_C = out_A} \quad \overline{out_{B'} = out_A}\,1}{out_{B'} = out_C}$$

and ζ_2 is:

$$\dfrac{\dfrac{\dfrac{\overline{(i_{\triangleright(in_C)},o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}}\,2 \quad C \approx_{\mathcal{I}} B}{(i_{\triangleright(in_B)},o_{\triangleright(out^{\perp}_{B})}) \in [\![B]\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\mathcal{I}}\,II) \quad B \sqsupseteq_b A}{(i_{\triangleright(in_A)},o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\beta}\,III) \quad \overline{B' \approx_{\mathcal{I}} A}\,1}{(i_{\triangleright(in_{B'})},o_{\triangleright(out^{\perp}_{A})}) \in [\![B']\!]^{\omega}_{x}}\,(\sqsupseteq^{-}_{\mathcal{I}}\,III) \quad \dfrac{\overline{B' \approx_{\mathcal{I}} A}\,1}{out_{B'} = out_A}}{(i_{\triangleright(in_{B'})},o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_{x}}$$

Proposition 5.3.5 For arbitrary charts A and C, and infinite sequences i and o we have,

$$\frac{C \sqsupseteq_{\mathcal{A}} A \quad C \sqsupseteq_b B, B \approx_{\mathcal{I}} A \vdash P}{P}\ (\sqsupseteq^-_{\mathcal{A}\,I}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A}{\neg\,(C \sqsupseteq_\iota A)}\ (\sqsupseteq^-_{\mathcal{A}\,II})$$

$$\frac{C \sqsupseteq_{\mathcal{A}} A}{out_C = out_A}\ (\sqsupseteq^-_{\mathcal{A}\,III}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{A}\,IV})$$

$$\frac{C \sqsupseteq_{\mathcal{A}} A}{\neg\,(in_C \subseteq in_A)}\ (\sqsupseteq^-_{\mathcal{A}\,V}) \qquad \frac{C \sqsupseteq_{\mathcal{A}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \notin [\![A]\!]^\omega_x}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \notin [\![C]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{A}\,VI})$$

assuming the usual conditions for B and P.

Proof 5.3.5

For $(\sqsupseteq^-_{\mathcal{A}\,I})$ we have,

$$\dfrac{\dfrac{\dfrac{C \sqsupseteq_{\mathcal{A}} A}{(\exists B \bullet C \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A) \wedge \neg\, C \sqsupseteq_\iota A}\ (\sqsupseteq_{\mathcal{A}}\text{-}df) \qquad \dfrac{\overline{C \sqsupseteq_b B}^{\,1} \quad \overline{B \approx_{\mathcal{I}} A}^{\,1}}{\vdots \qquad \vdots}}{\exists B \bullet C \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A}\ (\wedge^-) \qquad P}{P}\ (\exists^-)(1)$$

The proof of $(\sqsupseteq^-_{\mathcal{A}\,II})$ is trivial from the definition of $\sqsupseteq_{\mathcal{A}}$ using $(\wedge^-)$.

For $(\sqsupseteq^-_{\mathcal{A}\,III})$ we have,

$$\dfrac{C \sqsupseteq_{\mathcal{A}} A \qquad \dfrac{\dfrac{\overline{C \sqsupseteq_b B}^{\,1}}{out_C = out_B}\ (\sqsupseteq^-_{\beta\,II}) \qquad \dfrac{\overline{B \approx_{\mathcal{I}} A}^{\,1}}{out_B = out_A}\ (\sqsupseteq^-_{\mathcal{I}\,I})}{out_C = out_A}}{out_C = out_A}\ (\sqsupseteq^-_{\mathcal{A}\,I})(1)$$

For $(\sqsupseteq^-_{\mathcal{A}\,IV})$ we have,

$$\dfrac{C \sqsupseteq_{\mathcal{A}} A \qquad \dfrac{\overline{B \approx_{\mathcal{I}} A}^{\,1} \qquad \dfrac{\overline{C \sqsupseteq_b B}^{\,1} \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\ (\sqsupseteq^-_{\beta\,III})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\,II})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{A}\,I})(1)$$

For $(\sqsupseteq^-_{\mathcal{A}\,V})$ we have,

$$\dfrac{\dfrac{C \sqsupseteq_{\mathcal{A}} A}{\neg\,(C \sqsupseteq_{\iota} A)}\,(\sqsupseteq^{-}_{\mathcal{A}\,II}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \approx_{\mathcal{I}} C}\,2}{C \approx_{\mathcal{I}} B}\,(\sqsupseteq^{-}_{\mathcal{I}\,VII}) \quad \begin{array}{c}\zeta_1\\ \vdots\\ B \sqsupseteq_b A\end{array}}{C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}\,(\wedge^{+}) \quad \dfrac{}{in_C \subseteq in_A}\,1}{\exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}\,(\exists^{+})}{C \sqsupseteq_{\iota} A}\,(\sqsupseteq^{-}_{\mathcal{I}\,VIII})(2)}{\neg\,(C \sqsupseteq_{\iota} A)}\,(\sqsupseteq^{-}_{\mathcal{A}\,II})}{\dfrac{\bot}{\neg\,(in_C \subseteq in_A)}\,(\bot^{-})(1)}$$

where ζ_1 is:

$$\dfrac{\begin{array}{c}\zeta_2\\ \vdots\\ in_B = in_A\end{array}\,2 \qquad \dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_B)}, o_{\triangleright(out^{\perp}_B)}) \in [\![B]\!]^{\omega}_x}\,3 \quad \dfrac{}{B \approx_{\mathcal{I}} C}\,2}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{I}\,II}) \quad C \sqsupseteq_{\mathcal{A}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{A}\,IV})}{B \sqsupseteq_b A}\,(\sqsupseteq^{+}_{\beta\,I})(3)$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{B \approx_{\mathcal{I}} C}\,2}{out_B = out_C}\,(\sqsupseteq^{-}_{\mathcal{I}\,I}) \quad \dfrac{C \sqsupseteq_{\mathcal{A}} A}{out_C = out_A}\,(\sqsupseteq^{-}_{\mathcal{A}\,III})}{out_B = out_A}$$

For $(\sqsupseteq^{-}_{\mathcal{A}\,VI})$ we have,

$$\dfrac{\dfrac{C \sqsupseteq_{\mathcal{A}} A \quad \dfrac{\dfrac{}{\neg\,(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \notin [\![C]\!]^{\omega}_x}\,1}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{A}\,IV}) \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \notin [\![A]\!]^{\omega}_x}{\dfrac{\bot}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \notin [\![C]\!]^{\omega}_x}\,(\bot^{-})(1)}$$

Lemma 5.3.6 For arbitrary charts A, A', C and C' we have,

$$\dfrac{in_C = in_{C'} \quad C \approx_{\mathcal{I}} A \quad C' \approx_{\mathcal{I}} A}{[\![C]\!]^{\omega}_x = [\![C']\!]^{\omega}_x} \qquad \dfrac{in_A = in_{A'} \quad C \approx_{\mathcal{I}} A \quad C \approx_{\mathcal{I}} A'}{[\![A]\!]^{\omega}_x = [\![A']\!]^{\omega}_x}$$

Proof 5.3.6

For the left hand side we have:

$$\dfrac{in_C = in_{C'} \qquad \dfrac{\begin{array}{c}\zeta_1\\ \vdots\end{array} \quad \dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_{C'})}, o_{\triangleright(out^{\perp}_{C'})}) \in [\![C']\!]^{\omega}_x}\,1 \quad C' \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{I}\,II}) \quad C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{I}\,III})}{\begin{array}{l}(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x \Leftrightarrow\\ (i_{\triangleright(in_{C'})}, o_{\triangleright(out^{\perp}_{C'})}) \in [\![C']\!]^{\omega}_x\end{array}}\,(\Rightarrow^{+})(1) \quad \begin{array}{c}\zeta_2\\ \vdots\\ out_C = out_{C'}\end{array}}{C \approx_{\mathcal{I}} C'}\,(\sqsupseteq^{+}_{\mathcal{I}\,I})}{[\![C]\!]^{\omega}_x = [\![C']\!]^{\omega}_x}\,(\sqsupseteq^{-}_{\mathcal{I}\,IV})$$

where ζ_1 is:

$$\dfrac{\dfrac{\overline{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega}}^{\,2} \quad C \approx_{\mathcal{I}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]_x^{\omega}}\,(\sqsupseteq^{-}_{\mathcal{I}\,II}) \quad C' \approx_{\mathcal{I}} A}{(i_{\triangleright(in_{C'})}, o_{\triangleright(out_{C'}^{\perp})}) \in [\![C']\!]_x^{\omega}}\,(\sqsupseteq^{-}_{\mathcal{I}\,III})}{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]_x^{\omega} \Rightarrow (i_{\triangleright(in_{C'})}, o_{\triangleright(out_{C'}^{\perp})}) \in [\![C']\!]_x^{\omega}}\,(\Rightarrow^{+})(2)$$

And ζ_2 is:

$$\dfrac{\dfrac{\overline{C \approx_{\mathcal{I}} A}}{out_C = out_A}\,(\sqsupseteq^{-}_{\mathcal{I}\,I}) \quad \dfrac{\overline{C' \approx_{\mathcal{I}} A}}{out_{C'} = out_A}\,(\sqsupseteq^{-}_{\mathcal{I}\,I})}{out_C = out_{C'}}$$

The proof of the right hand rule is easily derived using the symmetry of the proof above.

Lemma 5.3.7 For the arbitrary charts A and C we have,

$$\dfrac{C \approx_{\mathcal{I}} A}{C \sqsupseteq_{\iota} A} \qquad \dfrac{C \sqsupseteq_{\iota} A}{\exists A' \bullet C \approx_{\mathcal{I}} A'}$$

Proof 5.3.7

For the left hand side we have:

$$\dfrac{\dfrac{C \approx_{\mathcal{I}} A \quad \overline{A \sqsupseteq_b A}\,(\sqsupseteq^{+}_{\beta\,II})}{\exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}}{C \sqsupseteq_{\iota} A}\,(\sqsupseteq_{\iota}\text{-}df)$$

And for the right hand side we have,

$$\dfrac{\dfrac{C \sqsupseteq_{\iota} A}{\exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}\,(\sqsupseteq_{\iota}\text{-}df) \quad \dfrac{\overline{C \approx_{\mathcal{I}} B}^{\,1}}{\exists A' \bullet C \approx_{\mathcal{I}} A'}\,(\exists^{+})}{\exists A' \bullet C \approx_{\mathcal{I}} A'}\,(\exists^{-})(1)$$

Lemma 5.3.8 For an arbitrary chart C and signal set ins,

$$\dfrac{ins \subseteq in_C}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_{\iota} A \vee C \sqsupseteq_{\mathcal{A}} A)}$$

Proof 5.3.8

$$\dfrac{\dfrac{}{\exists A \bullet in_A = ins \wedge [\![A]\!]_x^{\omega} = [\![chaos_A]\!]_x^{\omega}}\,(Chaos\text{-}df) \quad \dfrac{\dfrac{}{C \sqsupseteq_{\iota} A \vee \neg (C \sqsupseteq_{\iota} A)}\,(LEM) \quad \begin{matrix}\zeta_1\\ \vdots\end{matrix} \quad \begin{matrix}\zeta_2\\ \vdots\end{matrix}}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_{\iota} A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\vee^{-})(2)}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_{\iota} A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\exists^{-})(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{in_A = ins}\,1 \quad \dfrac{\dfrac{}{C \sqsupseteq_\iota A}\,2}{C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A}\,(\vee^+)}{in_A = ins \wedge (C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\exists^+)}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\exists^+)$$

And ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{C \sqsupseteq_b chaos_C}\,(\sqsupseteq_b\text{-}df) \quad \dfrac{chaos_C \approx_{\mathcal{I}} chaos_A}{chaos_C \approx_{\mathcal{I}} A}\,(\approx_{\mathcal{I}}\text{-}df) \quad \dfrac{\dfrac{[\![chaos_A]\!]^\omega_x = [\![A]\!]^\omega_x}{chaos_A \approx_{\mathcal{I}} A}\,1}{}\,(\sqsupseteq^+_{\mathcal{I}\,II})}{}\,(\sqsupseteq^-_{\mathcal{I}\,V})}{\exists B \bullet C \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A}\,(\exists^+) \quad \dfrac{}{\neg (C \sqsupseteq_\iota A)}\,2}{(\exists B \bullet C \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A) \wedge \neg (C \sqsupseteq_\iota A)}\,(\wedge^+)}{C \sqsupseteq_{\mathcal{A}} A}\,(\sqsupseteq_{\mathcal{A}}\text{-}df)$$

$$\dfrac{\dfrac{\dfrac{}{in_A = ins}\,1 \quad \dfrac{C \sqsupseteq_{\mathcal{A}} A}{C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A}\,(\vee^+)}{in_A = ins \wedge (C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\exists^+)}{\exists A \bullet in_A = ins \wedge (C \sqsupseteq_\iota A \vee C \sqsupseteq_{\mathcal{A}} A)}\,(\exists^+)$$

Lemma 5.3.9 For arbitrary charts A, C we have that,

$$\dfrac{C \sqsupseteq_{\mathcal{A}} A}{\exists i, i', o \bullet i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \wedge (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \wedge (i'_{\triangleright(in_C)}, o) \notin [\![C]\!]^\omega_x}$$

Proof 5.3.9

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{C \sqsupseteq_{\mathcal{A}} A}{\neg (in_C \subseteq in_A)}\,(\sqsupseteq^-_{\mathcal{A}\,V})}{in_A \cap in_C \subset in_C} \quad \dfrac{\begin{array}{c}\zeta_3 \quad \zeta_4 \\ \vdots \quad \vdots \\ (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Leftrightarrow \\ (i_{\triangleright(in_A \cap in_C)}, o) \in [\![C]\!]^\omega_x\end{array}}{}\,(\wedge^+) \quad \begin{array}{c}\zeta \\ \vdots \\ \bot\end{array}}{\bot}\,(\sqsupseteq^-_{\mathcal{I}\,IX})(2)}{\neg\, \forall i, i', o \bullet i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x) \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}\,(\bot^-)(1)}{\exists i, i', o \bullet \neg\, (i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x))}$$

$$\exists i, i', o \bullet i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \wedge (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \wedge (i'_{\triangleright(in_C)}, o) \notin [\![C]\!]^\omega_x$$

where ζ is:

$$\dfrac{\dfrac{\dfrac{}{in_{B'} = in_A \cap in_C}\,2 \quad \dfrac{}{in_A \cap in_C \subseteq in_A}\,(df)}{in_{B'} \subseteq in_A} \quad \dfrac{\dfrac{C \sqsupseteq_{\mathcal{A}} A}{\neg (C \sqsupseteq_\iota A)}\,(\sqsupseteq^-_{\mathcal{A}\,II}) \quad \begin{array}{c}\zeta_1 \\ \vdots \\ C \sqsupseteq_\iota A\end{array}}{\bot}}{\bot}\,(\sqsupseteq^-_{\mathcal{I}\,VIII})(5)$$

ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{in_B = in_A}\,5 \quad \zeta_2 \quad \dfrac{\dfrac{}{out_B = out_{B'}}\,5 \quad \dfrac{\dfrac{}{out_{B'} = out_C}\,2 \quad \dfrac{C \sqsupseteq_{\mathcal{A}} A}{out_C = out_A}\,(\sqsupseteq^-_{\mathcal{A}\,III})}{out_{B'} = out_A}}{out_B = out_A}\,(\sqsupseteq^+_{\beta\,I})(6) \quad \dfrac{}{C \approx_{\mathcal{I}} B}\,2}{B \sqsupseteq_b A}\,(\wedge^+),(\exists^+)}{\exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}\,(\sqsupseteq_\iota\text{-}df)}{C \sqsupseteq_\iota A}$$

ζ_2 is:

$$\dfrac{\dfrac{}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\,6 \quad \dfrac{\dfrac{\dfrac{}{B' \approx_{\mathcal{I}} C}\,2}{C \approx_{\mathcal{I}} B'}\,(\sqsupseteq^-_{\mathcal{I}\,VII}) \quad \dfrac{\dfrac{}{B \approx_{\mathcal{I}} B'}\,5}{B' \approx_{\mathcal{I}} B}\,(\sqsupseteq^-_{\mathcal{I}\,VII})}{C \approx_{\mathcal{I}} B}\,(\sqsupseteq^-_{\mathcal{I}\,V})}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{I}\,III}) \quad C \sqsupseteq_{\mathcal{A}} A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{A}\,IV})$$

Now for ζ_3, assuming $in_{A_C} = in_A \cap in_C$ and $i_{X_Y} = (i_{\triangleright(in_X)})_{\triangleright(in_Y)}$ we have,

$$\dfrac{\dfrac{\dfrac{}{i_{A_A} = i_{\triangleright(in_A)}}\,(\triangleright(.)\text{-}df) \quad \dfrac{\dfrac{}{\forall i, i', o \bullet i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}\,1}{i_{A_A} = i_{\triangleright(in_A)} \Rightarrow ((i_{A_C}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}\,(\forall^-)}{(i_{A_C}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x} \quad \dfrac{}{i_{A_C} = i_{\triangleright(in_{A_C})}}\,(\triangleright(.)\text{-}df)}{(i_{\triangleright(in_{A_C})}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}$$

And for ζ_4, again assuming $in_{A_C} = in_A \cap in_C$ and $i_{X_Y} = (i_{\triangleright(in_X)})_{\triangleright(in_Y)}$ we have,

$$\dfrac{\dfrac{\dfrac{}{i_{A_A} = i_{\triangleright(in_A)}}\,(\triangleright(.)\text{-}df) \quad \dfrac{\dfrac{}{\forall i, i', o \bullet i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}\,1}{i_{\triangleright(in_A)} = i_{A_A} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{A_C}, o) \in [\![C]\!]^\omega_x)}\,(\forall^-)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{A_C}, o) \in [\![C]\!]^\omega_x} \quad \dfrac{}{i_{A_C} = i_{\triangleright(in_{A_C})}}\,(\triangleright(.)\text{-}df)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i_{\triangleright(in_{A_C})}, o) \in [\![C]\!]^\omega_x}$$

Lemma 5.3.10 For arbitrary charts A, C, input sequences i and i', and all output sequences o we have,

$$\dfrac{C \sqsupseteq_\iota A}{i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}$$

Proof 5.3.10

$$\dfrac{\dfrac{\dfrac{C \sqsupseteq_\iota A}{\exists B \bullet C \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A}\,(\sqsupseteq_\iota\text{-}df) \quad \dfrac{\begin{array}{c}\vdots\ \zeta \\ (i'_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x \quad \overline{C \approx_{\mathcal{I}} B}^{\,3}\end{array}}{(i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{I}\,III})}{(i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\,(\exists^-)(3)}{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}\,(\Rightarrow^+)(2)}{i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow ((i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x \Rightarrow (i'_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x)}\,(\Rightarrow^+)(1)$$

where ζ is:

$$\dfrac{\overline{C \approx_{\mathcal{I}} B}^{\,3} \quad \dfrac{\overline{(i_{\triangleright(in_C)}, o) \in [\![C]\!]^\omega_x}^{\,2}}{(i_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{I}\,II}) \quad \begin{array}{c}\vdots\ \zeta_1 \\ (i_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x \Rightarrow \\ (i'_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x\end{array}}{(i'_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x}\,(\Rightarrow^-)$$

ζ_1 is:

$$\dfrac{\dfrac{\overline{[\![B]\!]^\omega_x = [\![B]\!]^\omega_x}}{\begin{array}{r}\forall i, o \bullet (i, o) \in [\![B]\!]^\omega_x \Leftrightarrow \\ (i, o) \in [\![B]\!]^\omega_x\end{array}} \quad \dfrac{\overline{i_{\triangleright(in_A)} = i'_{\triangleright(in_A)}}^{\,1} \quad \dfrac{\dfrac{\overline{B \sqsupseteq_b A}^{\,3}}{in_B = in_A}\,(\sqsupseteq^-_{\beta\,I})}{\begin{array}{r}i_{\triangleright(in_A)} = i'_{\triangleright(in_A)} \Rightarrow \\ i_{\triangleright(in_B)} = i'_{\triangleright(in_B)}\end{array}}}{i_{\triangleright(in_B)} = i'_{\triangleright(in_B)}}}{\begin{array}{r}(i_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x \Leftrightarrow \\ (i_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x\end{array}}}{(i_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x \Leftrightarrow (i'_{\triangleright(in_B)}, o) \in [\![B]\!]^\omega_x}$$

B.8.2 Output interface refinement

Proposition 5.3.11 For arbitrary charts A and C, signal set *outs* and infinite sequences i, o and o' we have,

$$\frac{C \approx_{\mathcal{O}} A}{in_C = in_A}\ (\sqsupseteq^-_{\mathcal{O}\,I}) \qquad \frac{C \approx_{\mathcal{O}} A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}{(i_{\triangleright(out^{\perp}_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{O}\,II})$$

$$\frac{C \approx_{\mathcal{O}} A \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{O}\,III}) \qquad \frac{C \approx_{\mathcal{O}} A \quad out_C = out_A}{[\![C]\!]^{\omega}_x = [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{O}\,IV})$$

$$\frac{C \approx_{\mathcal{O}} B \quad B \approx_{\mathcal{O}} A}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^-_{\mathcal{O}\,V}) \qquad \frac{C \approx_{\mathcal{O}} A \qquad outs = out_A \cap out_C}{(i, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x \Leftrightarrow (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^-_{\mathcal{O}\,VI})$$

$$\frac{C \approx_{\mathcal{O}} A}{A \approx_{\mathcal{O}} C}\ (\sqsupseteq^-_{\mathcal{O}\,VII}) \qquad \frac{out_A \subseteq outs \qquad \begin{array}{l} out_B = outs, \\ in_B = in_A, \\ B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{O}\,VIII})$$

$$\frac{outs \subset out_A \qquad \begin{array}{l} (i, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x \Leftrightarrow \\ (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]^{\omega}_x \end{array} \qquad \begin{array}{l} out_B = outs, \\ in_B = in_A, \\ B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\mathcal{O}\,IX})$$

where we assume the usual conditions for B and P.

Proof 5.3.11

For $(\sqsupseteq^-_{\mathcal{O}\,I})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{i \in \mathrm{dom}\,[\![C]\!]^{\omega}_x}\,1}{\exists\, o \bullet (i, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,(\mathrm{dom}\text{-}df) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{(i, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,2 \qquad C \approx_{\mathcal{O}} A}{(i, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\approx_{\mathcal{O}}\text{-}df)}{\exists\, o' \bullet (i, o') \in [\![A]\!]^{\omega}_x}\,(\exists^+)}{i \in \mathrm{dom}\,[\![A]\!]^{\omega}_x}\,(\mathrm{dom}\text{-}df)}{i \in \mathrm{dom}\,[\![A]\!]^{\omega}_x}\,(\exists^-)(2)}{i \in \mathrm{dom}\,[\![C]\!]^{\omega}_x \Rightarrow i \in \mathrm{dom}\,[\![A]\!]^{\omega}_x}\,(\Rightarrow^+)(1) \qquad \begin{array}{c}\zeta_1 \\ \vdots\end{array}}{\mathrm{dom}\,[\![C]\!]^{\omega}_x = \mathrm{dom}\,[\![A]\!]^{\omega}_x}\,([\![\cdot]\!]^{\omega}_x\text{-}df)}{\mathcal{I}^{\omega}_C = \mathcal{I}^{\omega}_A}}{in_C = in_A}$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{i \in \mathrm{dom}\,[\![A]\!]^{\omega}_{x}}\,3}{\exists\, o \bullet (i, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{(i, o_{\triangleright(out^{\perp}_{C})}) \in [\![A]\!]^{\omega}_{x}}\,4 \quad C \approx_{\mathcal{O}} A}{(i, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}}\,(\approx_{\mathcal{O}}\text{-}df)}{\exists\, o' \bullet (i, o') \in [\![C]\!]^{\omega}_{x}}\,(\exists^{+})}{i \in \mathrm{dom}\,[\![C]\!]^{\omega}_{x}}\,(\mathrm{dom}\text{-}df)}{i \in \mathrm{dom}\,[\![C]\!]^{\omega}_{x}}\,(\exists^{-})(4)}{i \in \mathrm{dom}\,[\![A]\!]^{\omega}_{x} \Rightarrow i \in \mathrm{dom}\,[\![C]\!]^{\omega}_{x}}\,(\Rightarrow^{+})(3)}$$

For $(\sqsupseteq^{-}_{\mathcal{O}\,II})$ we have,

$$\dfrac{\begin{array}{c}\zeta_1\\ \vdots\\ (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}\end{array} \quad \dfrac{C \approx_{\mathcal{O}} A}{in_C = in_A}\,(\sqsupseteq^{-}_{\mathcal{O}\,I})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}}$$

where ζ_1 is:

$$\dfrac{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x} \qquad \dfrac{C \approx_{\mathcal{O}} A}{\begin{array}{l}(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x} \Rightarrow\\ \quad (i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}\end{array}\,(\approx_{\mathcal{O}}\text{-}df), (\wedge^{-})}}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}}\,(\Rightarrow^{-})$$

For $(\sqsupseteq^{-}_{\mathcal{O}\,III})$ we have,

$$\dfrac{\begin{array}{c}\zeta_1\\ \vdots\\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}\end{array} \quad \dfrac{C \approx_{\mathcal{O}} A}{in_C = in_A}\,(\sqsupseteq^{-}_{\mathcal{O}\,I})}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}}$$

where ζ_1 is:

$$\dfrac{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x} \qquad \dfrac{C \approx_{\mathcal{I}} A}{\begin{array}{l}(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x} \Rightarrow\\ \quad (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}\end{array}\,(\approx_{\mathcal{O}}\text{-}df), (\wedge^{-})}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}}\,(\Rightarrow^{-})$$

As with the rules $(\sqsupseteq^{-}_{\mathcal{I}\,II})$ and $(\sqsupseteq^{-}_{\mathcal{I}\,III})$ we again introduce and prove two less general variants of the rules $(\sqsupseteq^{-}_{\mathcal{O}\,II})$ and $(\sqsupseteq^{-}_{\mathcal{O}\,III})$.
Firstly, we have that,

$$\dfrac{(i, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x} \qquad \dfrac{C \approx_{\mathcal{O}} A}{\begin{array}{l}(i, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x} \Rightarrow\\ \quad (i, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}\end{array}\,(\approx_{\mathcal{O}}\text{-}df), (\wedge^{-})}}{(i, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x}}\,(\Rightarrow^{-})$$

Correspondingly,

$$\dfrac{(i, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x} \qquad \dfrac{C \approx_{\mathcal{I}} A}{\begin{array}{l}(i, o_{\triangleright(out^{\perp}_{A})}) \in [\![A]\!]^{\omega}_{x} \Rightarrow\\ \quad (i, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}\end{array}\,(\approx_{\mathcal{O}}\text{-}df), (\wedge^{-})}}{(i, o_{\triangleright(out^{\perp}_{C})}) \in [\![C]\!]^{\omega}_{x}}\,(\Rightarrow^{-})$$

For $(\sqsupseteq^-_{\mathcal{O}\,IV})$ we have,

$$\dfrac{\dfrac{\begin{array}{cc}\zeta_1 & \zeta_2\\ \vdots & \vdots\end{array}}{(i',o)\in[\![C]\!]^\omega_x \Leftrightarrow (i',o)\in[\![A]\!]^\omega_x}\,(\wedge^+)}{\forall\, i,o \bullet (i,o)\in[\![C]\!]^\omega_x \Leftrightarrow (i,o)\in[\![A]\!]^\omega_x}\,(\forall^+)}{[\![C]\!]^\omega_x = [\![A]\!]^\omega_x}$$

where ζ_1 is:

$$\dfrac{C\approx_{\mathcal{O}} A \quad \dfrac{\dfrac{}{(i,o')\in[\![C]\!]^\omega_x}\,1}{(i,o'_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,([\![.]\!]^\omega_x\text{-}df) \qquad \dfrac{\dfrac{}{(i,o')\in[\![C]\!]^\omega_x}\,1}{o'=o'_{\triangleright(out^\perp_C)}}\,([\![.]\!]^\omega_x\text{-}df) \quad \dfrac{out_A=out_C}{out^\perp_A=out^\perp_C}\,(\perp\text{-}df)}{(i,o'_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,II}) \qquad \dfrac{}{o'=o'_{\triangleright(out^\perp_A)}}\,1}{\dfrac{(i,o')\in[\![A]\!]^\omega_x}{(i,o')\in[\![C]\!]^\omega_x\Rightarrow(i,o')\in[\![A]\!]^\omega_x}\,(\Rightarrow^+)(1)}$$

and ζ_2 is:

$$\dfrac{C\approx_{\mathcal{O}} A \quad \dfrac{\dfrac{}{(i,o')\in[\![A]\!]^\omega_x}\,1}{(i,o'_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,([\![.]\!]^\omega_x\text{-}df) \qquad \dfrac{\dfrac{}{(i,o')\in[\![A]\!]^\omega_x}\,1}{o'=o'_{\triangleright(out^\perp_A)}}\,([\![.]\!]^\omega_x\text{-}df) \quad \dfrac{out_A=out_C}{out^\perp_A=out^\perp_C}\,(\perp\text{-}df)}{(i,o'_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,III}) \qquad \dfrac{}{o'=o'_{\triangleright(out^\perp_C)}}}{\dfrac{(i,o')\in[\![C]\!]^\omega_x}{(i,o')\in[\![A]\!]^\omega_x\Rightarrow(i,o')\in[\![C]\!]^\omega_x}\,(\Rightarrow^+)(1)}$$

For $(\sqsupseteq^-_{\mathcal{O}\,V})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,1 \quad C\approx_{\mathcal{O}} B}{(i,o_{\triangleright(out^\perp_B)})\in[\![B]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,II}) \quad B\approx_{\mathcal{O}} A}{(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,II}) \quad \begin{array}{c}\zeta_1\\ \vdots\end{array}}{(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x\Rightarrow(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(\Rightarrow^+)(1)}{(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x\Leftrightarrow(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}}{C\approx_{\mathcal{O}} A}\,(\approx_{\mathcal{O}}\text{-}df)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,2 \quad B\approx_{\mathcal{O}} A}{(i,o_{\triangleright(out^\perp_B)})\in[\![B]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,III}) \quad C\approx_{\mathcal{O}} B}{(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,III})}{(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x\Rightarrow(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,(\Rightarrow^+)(2)$$

For $(\sqsupseteq^-_{\mathcal{O}\,VI})$ we have,

$$\dfrac{\dfrac{\dfrac{C\approx_{\mathcal{O}} A \quad \dfrac{\dfrac{\begin{array}{c}\zeta_3\\ \vdots\end{array}}{outs^\perp=out^\perp_A\cap out^\perp_C} \quad \dfrac{}{(i,o_{\triangleright(outs^\perp)})\in[\![A]\!]^\omega_x}\,1}{((i,o_{\triangleright(out^\perp_C)})_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(_{\triangleright(.)}\text{-}df)}{(i,(o_{\triangleright(out^\perp_C)})_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,III})}{(i,o_{\triangleright(out^\perp_C)})\in[\![C]\!]^\omega_x}\,(_{\triangleright(.)}\text{-}df)}{(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\,II}) \quad C\approx_{\mathcal{O}} A}{(i,o_{\triangleright(outs^\perp)})\in[\![A]\!]^\omega_x\Rightarrow(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x}\,(\Rightarrow^+)(1) \quad \begin{array}{c}\zeta_1\\ \vdots\end{array}}{(i,o_{\triangleright(out^\perp_A)})\in[\![A]\!]^\omega_x\Leftrightarrow(i,o_{\triangleright(outs^\perp)},o)\in[\![A]\!]^\omega_x}$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\zeta_3}{\vdots}}{outs^{\perp} \subseteq out_A^{\perp}} \qquad \dfrac{\dfrac{\dfrac{\zeta_2}{\vdots}}{(i, o_{\triangleright(outs^{\perp})}) \in [\![C]\!]^{\omega}_x} \qquad \dfrac{\dfrac{\dfrac{\zeta_3}{\vdots}}{outs^{\perp} = out_A^{\perp} \cap out_C^{\perp}}}{outs^{\perp} \subseteq out_C^{\perp}}}{(i, (o_{\triangleright(outs^{\perp})})_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\triangleright(.)\text{-}df) \qquad C \approx_{\mathcal{O}} A}{(i, (o_{\triangleright(outs^{\perp})})_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^{-}_{\mathcal{O}\ II})}{(i, o_{\triangleright(outs)}) \in [\![A]\!]^{\omega}_x}\ (\triangleright(.)\text{-}df)}{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x \Rightarrow (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\Rightarrow^{+})(2)$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{\zeta_3}{\vdots}}{outs^{\perp} = out_A^{\perp} \cap out_C^{\perp}} \qquad \dfrac{\dfrac{\dfrac{\overline{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ 2}{(i, (o_{\triangleright(out_A^{\perp})})_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\triangleright(.)\text{-}df) \qquad C \approx_{\mathcal{O}} A}{((o_{\triangleright(out_A^{\perp})})_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\sqsupseteq^{-}_{\mathcal{O}\ III})}{(i, o_{\triangleright(outs^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\triangleright(.)\text{-}df)$$

and ζ_3 is:

$$\dfrac{outs = out_A \cap out_C}{outs^{\perp} = out_A^{\perp} \cap out_C^{\perp}}\ (\perp\text{-}df)$$

For $(\sqsupseteq^{-}_{\mathcal{O}\ VII})$ we have,

$$\dfrac{\dfrac{\dfrac{C \approx_{\mathcal{O}} A \quad \overline{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ 1}{(i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\sqsupseteq^{-}_{\mathcal{O}\ III})}{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x \Rightarrow (i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\Rightarrow^{+})(1) \qquad \dfrac{\dfrac{C \approx_{\mathcal{O}} A \quad \overline{(i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ 2}{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\sqsupseteq^{-}_{\mathcal{O}\ II})}{(i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x \Rightarrow (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x}\ (\Rightarrow^{+})(2)}{(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x \Leftrightarrow (i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}$$
$$\dfrac{}{\forall\, i;\ o \bullet (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_x \Leftrightarrow (i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_x}\ (\forall^{+})$$
$$\dfrac{}{A \approx_{\mathcal{O}} C}\ (\approx_{\mathcal{O}}\text{-}df)$$

For $(\sqsupseteq^{-}_{\mathcal{O}\ VIII})$ we have,

$$\dfrac{\dfrac{\dfrac{out_A \subseteq outs}{\exists\, B \bullet out_B = outs \wedge in_B = in_A \wedge B \approx_{\mathcal{O}} A}\ \dagger \qquad \dfrac{\overline{out_B = outs}\ 1 \quad \overline{in_B = in_A}\ 1 \quad \overline{B \approx_{\mathcal{O}} A}\ 1}{P}}{P}\ (\exists^{-})(1)$$

As in the analogous proof of rule $(\sqsupseteq^{-}_{\mathcal{I}\ VIII})$, the existence of a suitable chart B to discharge the step labelled † is shown by describing a simple algorithm as follows. For the first new signal, say x where x is not already in out_A, we take chart A and add to it a copy of each of the existing transitions. We then add to the action (*i.e.* the set of output signals) of each of these duplicate transitions the signal x. That is, the new chart generated, say A_1, has each of A's original transitions plus an additional copy, each of which outputs the additional signal x. Now A_1 is such that $A_1 \approx_{\mathcal{O}} A$. We then proceed to add the next new signal, in the same fashion, to the chart A_1 giving yet another chart

A_2. The transitivity of $\approx_{\mathcal{O}}$ gives us that $A_2 \approx_{\mathcal{O}} A$. Continuing this chain until each of the signals in $outs \setminus out_A$ are added will give us the chart B as required.
For $(\sqsupseteq^{-}_{\mathcal{O}\,IX})$ we have,

$$\dfrac{\dfrac{outs \subset out_A \qquad (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(outs^{\perp})}) \in [\![A]\!]^{\omega}_{x}}{\exists B \bullet out_B = outs \wedge out_B = out_A \wedge B \approx_{\mathcal{O}} A}\ \dagger \qquad \dfrac{\dfrac{}{out_B = outs}\,1 \qquad \dfrac{\dfrac{}{out_B = out_A}\,1}{B \approx_{\mathcal{O}} A}\,1}{P}}{P}\ (\exists^{-})(1)$$

Again we prove the proof step labelled † by giving an informal account of an algorithm that creates an appropriate chart B from the given chart A. Take the specific chart A and remove from it any output reference to the signals that are not in the set *outs*. Change the output interface of this chart to be equal to *outs* as required. Because $(i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(outs)}) \in [\![A]\!]^{\omega}_{x}$ holds the resulting chart will have exactly the attributes required to prove the existence of the chart B.

Now for the introduction rules:

$$\dfrac{in_C = in_A \qquad (i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x}}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^{+}_{\mathcal{O}\,I})$$

$$\dfrac{[\![C]\!]^{\omega}_{x} = [\![A]\!]^{\omega}_{x}}{C \approx_{\mathcal{O}} A}\ (\sqsupseteq^{+}_{\mathcal{O}\,II})$$

Proof 5.3.11 (*cont.*)

The proof of $(\sqsupseteq^{+}_{\mathcal{O}\,I})$ is split into two cases as follows.
Case 1: Assuming $i = i_{\triangleright(in_C)}$ we have,

$$\dfrac{\dfrac{\dfrac{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(in_A)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x} \quad in_C = in_A}{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i_{\triangleright(in_C)}, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x}}}{(i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x}}}{C \approx_{\mathcal{O}} A}\ (\approx_{\mathcal{O}}\text{-}df)$$

Case 2: Assuming $i \neq i_{\triangleright(in_C)}$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{i \neq i_{\triangleright(in_C)}}{(i, o_{\triangleright(out_C^{\perp})}) \notin [\![C]\!]^{\omega}_{x}}\ ([\![\cdot]\!]^{\omega}_{x}\text{-}df) \qquad \dfrac{\dfrac{i \neq i_{\triangleright(in_C)} \quad in_C = in_A}{i \neq i_{\triangleright(in_A)}}}{(i, o_{\triangleright(out_A^{\perp})}) \notin [\![A]\!]^{\omega}_{x}}\ ([\![\cdot]\!]^{\omega}_{x}\text{-}df)}{(i, o_{\triangleright(out_C^{\perp})}) \notin [\![C]\!]^{\omega}_{x} \wedge (i, o_{\triangleright(out_A^{\perp})}) \notin [\![A]\!]^{\omega}_{x}}\ (\wedge^{+})}{(i, o_{\triangleright(out_C^{\perp})}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(out_A^{\perp})}) \in [\![A]\!]^{\omega}_{x}}\ (\textit{prop logic})}{C \approx_{\mathcal{O}} A}\ (\approx_{\mathcal{O}}\text{-}df)$$

And finally for $(\sqsupseteq^{+}_{\mathcal{O}\ II})$,

$$\dfrac{\dfrac{\dfrac{[\![C]\!]^{\omega}_{x} = [\![A]\!]^{\omega}_{x}}{out_C = out_A}\ ([\![.]\!]^{\omega}_{x}\text{-}df)}{out^{\perp}_C = out^{\perp}_A}\ (\perp\text{-}df) \qquad \dfrac{\dfrac{[\![C]\!]^{\omega}_{x} = [\![A]\!]^{\omega}_{x}}{\forall i, o \bullet (i,o) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i,o) \in [\![A]\!]^{\omega}_{x}}}{(i, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(out^{\perp}_C)}) \in [\![A]\!]^{\omega}_{x}}\ (\forall^{-})}{(i, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_{x}}}{C \approx_{\mathcal{O}} A}\ (\approx_{\mathcal{O}}\text{-}df)$$

Lemma 5.3.12 For any arbitrary abstract chart specification A and signal set $outs$,

$$\dfrac{out_A \subseteq outs}{\exists B \bullet out_B = outs \wedge B \approx_{\mathcal{O}} A}$$

$$\dfrac{C \approx_{\mathcal{O}} A}{\exists B \bullet out_B = out_A \cap out_C \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}$$

$$\dfrac{out_C \subset out_A \quad C \approx_{\mathcal{O}} A}{[\![A]\!]^{\omega}_{x} = [\![C \mid \{\} \mid True]\!]^{\omega}_{x}}$$

where $True$ is a chart that allows any output behaviour over the additional outputs of chart A. That is, $in_{True} = in_C$ and $out_{True} = out_A \setminus out_C$.

Proof 5.3.12

For the first property in this lemma we have:

$$\dfrac{out_A \subseteq outs \qquad \dfrac{\dfrac{\overline{out_B = outs}^{\,1} \quad \overline{B \approx_{\mathcal{O}} A}^{\,1}}{out_B = outs \wedge B \approx_{\mathcal{O}} A}\ (\wedge^{+})}{\exists B \bullet out_B = outs \wedge B \approx_{\mathcal{O}} A}\ (\exists^{+})}{\exists B \bullet out_B = outs \wedge B \approx_{\mathcal{O}} A}\ (\sqsupseteq^{-}_{\mathcal{O}\ VIII})(1)$$

We split the second property into two cases.

Case 1: Assuming $out_A \subseteq out_C$ we have,

$$\dfrac{\dfrac{\dfrac{out_A \subseteq out_C}{out_A = (out_C \cap out_A)} \quad C \approx_{\mathcal{O}} A \qquad \dfrac{\overline{[\![A]\!]^{\omega}_{x} = [\![A]\!]^{\omega}_{x}}}{A \approx_{\mathcal{O}} A}\ (\sqsupseteq^{+}_{\mathcal{O}\ II})}{out_A = (out_C \cap out_A) \wedge C \approx_{\mathcal{O}} A \wedge A \approx_{\mathcal{O}} A}\ (\wedge^{+})}{\exists B \bullet out_B = (out_C \cap out_A) \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}\ (\exists^{+})$$

Case 2: Assuming $\neg\,(out_A \subseteq out_C)$,

$$\dfrac{\neg\,(out_A \subseteq out_C)}{out_C \cap out_A \subset out_A} \qquad \dfrac{C \approx_{\mathcal{O}} A \qquad \dfrac{\overline{out_C \cap out_A = out_C \cap out_A}}{(i, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_{x} \Leftrightarrow (i, o_{\triangleright(out^{\perp}_A \cap out^{\perp}_C)}) \in [\![A]\!]^{\omega}_{x}}\ (\sqsupseteq^{-}_{\mathcal{O}\ VI}) \quad \begin{matrix}\zeta_1\\ \vdots\end{matrix}}{\exists B \bullet out_B = (out_C \cap out_A) \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}\ (\sqsupseteq^{-}_{\mathcal{O}\ IX})(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{out_B = (out_C \cap out_A)}\,1 \quad \dfrac{C \approx_{\mathcal{O}} A \quad \dfrac{\dfrac{}{B \approx_{\mathcal{O}} A}\,1}{A \approx_{\mathcal{O}} B}\,(\sqsupseteq^-_{\mathcal{O}\,VII})}{C \approx_{\mathcal{O}} B}\,(\sqsupseteq^-_{\mathcal{O}\,V}) \quad \dfrac{}{B \approx_{\mathcal{O}} A}\,1}{out_B = (out_C \cap out_A) \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}\,(\wedge^+)}{\exists B \bullet out_B = (out_C \cap out_A) \wedge C \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{O}} A}\,(\exists^+)$$

For the third property we give an informal proof. To give a formal proof would require lifting the semantic definition of chart composition (Section 3.4) into the the trace semantic interpretation of charts. This would require using induction over each of the trace semantic recursive definitions.

In the following we refer freely to the arbitrary charts A and C that appear in the definition of this property. We also refer to the signal set *outs* that occurs in the rule $(\sqsupseteq^-_{\mathcal{O}\,VIII})$.

Consider again the algorithm that we describe as part of the proof for the rule $(\sqsupseteq^-_{\mathcal{O}\,VIII})$. If we were to apply this algorithm to chart C, attempting to find an output refinement with an interface equal to out_A, the chart that the algorithm produces would be exactly chart A. That is, the algorithm that we described could equally be described as composing chart C with chart *True* where $out_{True} = outs \setminus out_C$.

B.9 Proofs for Section 5.4: Behavioural and interface refinement combined

Proposition 5.4.1 For arbitrary charts A and C, and infinite sequences i, o we have,

$$\frac{C \sqsupseteq_x^\omega A \quad (i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega}{(i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega} \ (\sqsupseteq^-)$$

$$\frac{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega \vdash (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega}{C \sqsupseteq_x^\omega A} \ (\sqsupseteq^+)$$

Proof 5.4.1
The proof of both $(\sqsupseteq^-)$ and $(\sqsupseteq^+)$ are trivial from definition 5.4.1.

Proposition 5.4.2 For arbitrary charts A and C we have,

Case 1: $in_A \subseteq in_C \wedge out_A \subseteq out_C$

$$\frac{C \sqsupseteq_x^\omega A \quad \begin{array}{l} C \sqsupseteq_b B', \\ B' \approx_{\mathcal{O}} B, B \approx_{\mathcal{I}} A \vdash P \end{array}}{P} \ (\sqsupseteq_{\Pi}^-) \qquad \frac{\exists B;\ B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{I}} A}{C \sqsupseteq_x^\omega A} \ (\sqsupseteq_{\Pi}^+)$$

Proof 5.4.2 (*Case 1*)
For $(\sqsupseteq_{\Pi}^-)$ we have,

$$\frac{\exists B;\ B' \bullet C \sqsupseteq_b B' \wedge B' \approx_{\mathcal{O}} B \wedge B \approx_{\mathcal{I}} A \qquad \begin{array}{c} \zeta \\ \vdots \\ (i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega \end{array}}{\dfrac{(i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega}{C \sqsupseteq_x^\omega A} \ (\sqsupseteq^+)(1)} \ (\exists^-)(2)$$

where ζ is:

$$\frac{\dfrac{\overline{B' \approx_{\mathcal{O}} B}^{\,2} \quad \dfrac{\overline{(i_{\triangleright(in_C)}, o_{\triangleright(out_C^\perp)}) \in [\![C]\!]_x^\omega}^{\,1} \quad \overline{C \sqsupseteq_b B'}^{\,2}}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out_{B'}^\perp)}) \in [\![B']\!]_x^\omega} \ (\sqsupseteq_{\beta\, III}^-)}{(i_{\triangleright(in_B)}, o_{\triangleright(out_B^\perp)}) \in [\![B]\!]_x^\omega} \ (\sqsupseteq_{\mathcal{O}\, II}^-) \qquad \overline{B \approx_{\mathcal{I}} A}^{\,2}}{(i_{\triangleright(in_A)}, o_{\triangleright(out_A^\perp)}) \in [\![A]\!]_x^\omega} \ (\sqsupseteq_{\mathcal{I}\, II}^-)$$

For $(\sqsupseteq_{\Pi}^+)$ we have,

$$\frac{\dfrac{\dfrac{\begin{array}{c} \zeta_1 \\ \vdots \\ C \sqsupseteq_b B' \end{array} \quad \overline{B' \approx_{\mathcal{O}} B}^{\,2} \quad \overline{B \approx_{\mathcal{I}} A}^{\,1}}{P} \qquad \dfrac{out_A \subseteq out_C \quad \overline{out_B = out_A}^{\,1}}{out_B \subseteq out_C}}{P} \ (\sqsupseteq_{\mathcal{O}\, VIII}^-)(2) \qquad in_A \subseteq in_C}{P} \ (\sqsupseteq_{\mathcal{I}\, VIII}^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{in_B = in_C}\,1 \quad \dfrac{}{in_{B'} = in_B}\,2}{in_C = in_{B'}} \quad \dfrac{}{out_C = out_{B'}}\,2 \quad \begin{array}{c}\zeta_2\\ \vdots\end{array}}{C \sqsupseteq_b B'}\,(\sqsupseteq^+_{\beta\, I})(3)$$

and ζ_2 is:

$$\dfrac{\dfrac{}{B \approx_{\mathcal{I}} A}\,1 \quad \dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,3 \quad C \sqsupseteq^{\omega}_x A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^-)}{(i_{\triangleright(in_B)}, o_{\triangleright(out^{\perp}_B)}) \in [\![B]\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{I}\, III}) \quad \dfrac{}{B' \approx_{\mathcal{O}} B}\,2}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{O}\, III})$$

Proposition 5.4.2 Case 2: $in_A \subseteq in_C \wedge out_C \subset out_A$

$$\dfrac{C \sqsupseteq^{\omega}_x A \quad \begin{array}{l} C \approx_{\mathcal{O}} B', \\ B' \sqsupseteq_b B, B \approx_{\mathcal{I}} A \vdash P \end{array}}{P}\,(\sqsupseteq^-_{\uparrow\downarrow}) \qquad \dfrac{\begin{array}{l}\exists\, B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A\end{array}}{C \sqsupseteq^{\omega}_x A}\,(\sqsupseteq^+_{\uparrow\downarrow})$$

Proof 5.4.2 (*Case 2*)

For $(\sqsupseteq^-_{\uparrow\downarrow})$ we have,

$$\dfrac{\begin{array}{l}\exists\, B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{I}} A\end{array} \quad \begin{array}{c}\zeta\\ \vdots\\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x\end{array}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\exists^-)(2)}{C \sqsupseteq^{\omega}_x A}\,(\sqsupseteq^+)(1)$$

where ζ is:

$$\dfrac{\dfrac{}{B' \sqsupseteq_b B}\,2 \quad \dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,1 \quad \dfrac{}{C \approx_{\mathcal{O}} B'}\,2}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{O}\, II})}{(i_{\triangleright(in_B)}, o_{\triangleright(out^{\perp}_B)}) \in [\![B]\!]^{\omega}_x}\,(\sqsupseteq^-_{\beta\, III}) \quad \dfrac{}{B \approx_{\mathcal{I}} A}\,2}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{I}\, II})$$

For $(\sqsupseteq^+_{\uparrow\downarrow})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B' \approx_{\mathcal{O}} C}\,2}{C \approx_{\mathcal{O}} B'}\,(\sqsupseteq^-_{\mathcal{O}\, VII}) \quad \begin{array}{c}\zeta_1\\ \vdots\\ B' \sqsupseteq_b B\end{array} \quad \dfrac{}{B \approx_{\mathcal{I}} A}\,1}{P} \quad \dfrac{out_C \subset out_A}{out_C \subseteq out_A}}{P}\,(\sqsupseteq^-_{\mathcal{O}\, VIII})(2)}{P}\,(\sqsupseteq^-_{\mathcal{I}\, VIII})(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{in_B = in_C}\,1 \quad \dfrac{}{in_{B'} = in_C}\,2}{in_{B'} = in_B} \qquad \dfrac{\dfrac{}{out_B = out_A}\,1 \quad \dfrac{}{out_{B'} = out_A}\,2}{out_{B'} = out_B} \qquad \begin{array}{c}\zeta_2\\ \vdots\end{array}}{B' \sqsupseteq_b B}\ (\sqsupseteq^+_{\beta\, I})(3)$$

and ζ_2 is:

$$\dfrac{C \sqsupseteq^\omega_x A \qquad \dfrac{\dfrac{}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^\perp_{B'})}) \in [\![B']\!]^\omega_x}\,3 \quad \dfrac{}{B' \approx_{\mathcal{O}} C}\,2}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{O}\, II})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-) \qquad \dfrac{}{B \approx_{\mathcal{I}} A}\,2}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\, III})$$

Proposition 5.4.2 Case 3: $in_C \subset in_A \wedge out_C \subset out_A$

$$\dfrac{C \sqsupseteq^\omega_x A \qquad \begin{array}{l} C \approx_{\mathcal{O}} B', \\ B' \approx_{\mathcal{I}} B, B \sqsupseteq_b A \vdash P \end{array}}{P}\ (\sqsupseteq^-_{\Rightarrow\!\Rightarrow}) \qquad \dfrac{\begin{array}{l}\exists B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A\end{array}}{C \sqsupseteq^\omega_x A}\ (\sqsupseteq^+_{\Rightarrow\!\Rightarrow})$$

Proof 5.4.2 (*Case 3*)

For $(\sqsupseteq^-_{\Rightarrow\!\Rightarrow})$ we have,

$$\dfrac{\begin{array}{l}\exists B;\ B' \bullet C \approx_{\mathcal{O}} B' \wedge \\ \qquad B' \approx_{\mathcal{I}} B \wedge B \sqsupseteq_b A\end{array} \qquad \begin{array}{c}\zeta\\ \vdots\\ (i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x\end{array}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\exists^-)(2)}{C \sqsupseteq^\omega_x A}\ (\sqsupseteq^+)(1)$$

where ζ is:

$$\dfrac{\dfrac{}{B' \approx_{\mathcal{I}} B}\,2 \qquad \dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\,1 \quad \dfrac{}{C \approx_{\mathcal{O}} B'}\,2}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^\perp_{B'})}) \in [\![B']\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{O}\, II})}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\ (\sqsupseteq^-_{\mathcal{I}\, II}) \qquad \dfrac{}{B \sqsupseteq_b A}\,2}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\ (\sqsupseteq^-_{\beta\, III})$$

For $(\sqsupseteq^+_{\Rightarrow\!\Rightarrow})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B' \approx_{\mathcal{O}} C}\,1}{C \approx_{\mathcal{O}} B'}\ (\sqsupseteq^-_{\mathcal{O}\, VII}) \quad \dfrac{\dfrac{}{B \approx_{\mathcal{I}} B'}\,2}{B' \approx_{\mathcal{I}} B}\ (\sqsupseteq^-_{\mathcal{I}\, VII}) \quad \begin{array}{c}\zeta_1\\ \vdots\\ B \sqsupseteq_b A\end{array}}{P} \qquad \dfrac{\dfrac{in_C \subset in_A}{in_C \subseteq in_A} \quad \dfrac{}{in_{B'} = in_C}\,1}{in_{B'} \subseteq in_A}\ (\sqsupseteq^-_{\mathcal{I}\, VIII})(2)}{P} \qquad \begin{array}{c}\vdots\\ P\end{array}}{P}\ (\sqsupseteq^-_{\mathcal{O}\, VIII})(1)$$

$$\dfrac{out_C \subset out_A}{out_C \subseteq out_A}$$

where ζ_1 is:

$$\dfrac{\dfrac{}{in_B = in_A}\,2 \quad \dfrac{\dfrac{}{out_B = out_{B'}}\,2 \quad \dfrac{}{out_{B'} = out_A}\,1}{out_B = out_A} \quad \begin{array}{c}\zeta_2\\ \vdots\end{array}}{B \sqsupseteq_b A}\,(\sqsupseteq^+_{\beta\, I})(3)$$

and ζ_2 is:

$$\dfrac{\dfrac{}{B' \approx_{\mathcal{O}} C}\,1 \quad \dfrac{\dfrac{}{(i_{\triangleright(in_B)}, o_{\triangleright(out^{\perp}_B)}) \in [\![B]\!]^{\omega}_x}\,3 \quad \dfrac{}{B \approx_{\mathcal{I}} B'}\,2}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{I}\, II})}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{O}\, II}) \quad C \sqsupseteq^{\omega}_x A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^-)$$

Proposition 5.4.2 Case 4: $in_C \subset in_A \wedge out_A \subseteq out_C$

$$\dfrac{C \sqsupseteq^{\omega}_x A \quad \begin{array}{l} C \approx_{\mathcal{I}} B', \\ B' \sqsupseteq_b B, B \approx_{\mathcal{O}} A \vdash P \end{array}}{P}\,(\sqsupseteq^-_{\downarrow\uparrow}) \qquad \dfrac{\begin{array}{l}\exists B;\ B' \bullet C \approx_{\mathcal{I}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{O}} A\end{array}}{C \sqsupseteq^{\omega}_x A}\,(\sqsupseteq^+_{\downarrow\uparrow})$$

where we assume the usual conditions for B, B' and P.

Proof 5.4.2 (*Case 4*)

For $(\sqsupseteq^-_{\downarrow\uparrow})$ we have,

$$\dfrac{\begin{array}{l}\exists B;\ B' \bullet C \approx_{\mathcal{I}} B' \wedge \\ \qquad B' \sqsupseteq_b B \wedge B \approx_{\mathcal{O}} A\end{array} \quad \begin{array}{c}\zeta\\ \vdots\\ (i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x\end{array}}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\exists^-)(2)$$
$$\dfrac{}{C \sqsupseteq^{\omega}_x A}\,(\sqsupseteq^+)(1)$$

where ζ is:

$$\dfrac{\dfrac{}{B' \sqsupseteq_b B}\,2 \quad \dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^{\perp}_C)}) \in [\![C]\!]^{\omega}_x}\,1 \quad \dfrac{}{C \approx_{\mathcal{I}} B'}\,2}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^{\perp}_{B'})}) \in [\![B']\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{I}\, II})}{(i_{\triangleright(in_B)}, o_{\triangleright(out^{\perp}_B)}) \in [\![B]\!]^{\omega}_x}\,(\sqsupseteq^-_{\beta\, III}) \quad \dfrac{}{B \approx_{\mathcal{O}} A}\,2}{(i_{\triangleright(in_A)}, o_{\triangleright(out^{\perp}_A)}) \in [\![A]\!]^{\omega}_x}\,(\sqsupseteq^-_{\mathcal{O}\, II})$$

For $(\sqsupseteq^+_{\downarrow\uparrow})$ we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B' \approx_{\mathcal{I}} C}\,2}{C \approx_{\mathcal{I}} B'}\,(\sqsupseteq^-_{\mathcal{I}\, VII}) \quad \begin{array}{c}\zeta_1\\ \vdots\\ B' \sqsupseteq_b B\end{array} \quad \dfrac{}{B \approx_{\mathcal{O}} A}\,(\sqsupseteq^-_{\mathcal{I}\, VII})}{P} \quad \dfrac{\dfrac{}{in_C \subset in_A}}{in_C \subseteq in_A}}{P}\,(\sqsupseteq^-_{\mathcal{I}\, VIII})(2) \quad \dfrac{}{out_A \subseteq out_C}}{P}\,(\sqsupseteq^-_{\mathcal{O}\, VIII})(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{in_{B'} = in_A}\,2 \quad \dfrac{}{in_B = in_A}\,1}{in_{B'} = in_B} \quad \dfrac{\dfrac{}{out_{B'} = out_C}\,2 \quad \dfrac{}{out_B = out_C}\,1}{out_{B'} = out_B} \quad \zeta_2}{B' \sqsupseteq_b B}\,(\sqsupseteq^+_{\beta\, I})(3)$$

and ζ_2 is:

$$\dfrac{C \sqsupseteq^\omega_x A \qquad \dfrac{\dfrac{}{(i_{\triangleright(in_{B'})}, o_{\triangleright(out^\perp_{B'})}) \in [\![B']\!]^\omega_x}\,3 \quad \dfrac{}{B' \approx_{\mathcal{I}} C}\,2}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{I}\, II})}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\,(\sqsupseteq^-) \qquad \dfrac{}{B \approx_{\mathcal{O}} A}\,1}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\,(\sqsupseteq^-_{\mathcal{O}\, III})$$

Lemma 5.4.3 For arbitrary charts A, B and C,

$$\dfrac{C \sqsupseteq^\omega_x B \quad B \sqsupseteq^\omega_x A}{C \sqsupseteq^\omega_x A}$$

Proof 5.4.3

$$\dfrac{\dfrac{\dfrac{}{(i_{\triangleright(in_C)}, o_{\triangleright(out^\perp_C)}) \in [\![C]\!]^\omega_x}\,1 \quad C \sqsupseteq^\omega_x B}{(i_{\triangleright(in_B)}, o_{\triangleright(out^\perp_B)}) \in [\![B]\!]^\omega_x}\,(\sqsupseteq^-) \quad B \sqsupseteq^\omega_x A}{(i_{\triangleright(in_A)}, o_{\triangleright(out^\perp_A)}) \in [\![A]\!]^\omega_x}\,(\sqsupseteq^-)}{C \sqsupseteq^\omega_x A}\,(\sqsupseteq^+)(1)$$

B.10 Proofs for Section 6.1: Relational ADTs with IO

Proposition 6.1.1 For all conformal infinite programs $P_a = AInit \fatsemi AOp_1 \fatsemi AOp_2 \fatsemi ...$ and $P_c = CInit \fatsemi COp_1 \fatsemi COp_2 \fatsemi ...$ we have,

$$[\![P_c]\!]^{r\omega}_d \subseteq [\![P_a]\!]^{r\omega}_d \Leftrightarrow \forall n \bullet [\![P_c \upharpoonright n]\!]^r_d \subseteq [\![P_a \upharpoonright n]\!]^r_d$$

Proof 6.1.1

For the $\Rightarrow$ direction we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(i \upharpoonright n, o \upharpoonright n) \in [\![P_c \upharpoonright n]\!]^r_d}\,1}{\exists s \bullet \forall m \bullet (i \upharpoonright m, ((o \upharpoonright n)^\frown s) \upharpoonright m) \in [\![P_c \upharpoonright m]\!]^r_d}\,\dagger \quad \begin{array}{c}\zeta_1\\ \vdots\end{array}}{(i \upharpoonright n, o \upharpoonright n) \in [\![P_a \upharpoonright n]\!]^r_d}\,(\exists^-)(2)}{(i \upharpoonright n, o \upharpoonright n) \in [\![P_c \upharpoonright n]\!]^r_d \Rightarrow (i \upharpoonright n, o \upharpoonright n) \in [\![P_a \upharpoonright n]\!]^r_d}\,(\Rightarrow^+)(1)}{[\![P_c \upharpoonright n]\!]^r_d \subseteq [\![P_a \upharpoonright n]\!]^r_d}}{\forall n \bullet [\![P_c \upharpoonright n]\!]^r_d \subseteq [\![P_a \upharpoonright n]\!]^r_d}\,(\forall^+)$$

We justify the proof step labelled dagger informally by considering two cases.

Case 1: Supposing $m \leq n$, we need to show that $(i \upharpoonright n, o \upharpoonright n) \in P_c \upharpoonright n \Rightarrow (i \upharpoonright m, o \upharpoonright m) \in P_c \upharpoonright m$. In this case, truncating the sequence o to length m results in a shorter sequence than truncating o to length n. Therefore, we are free to choose any sequence s. That any sequence s is suitable certainly implies there exists such an s. To show that this simpler implication holds we revisit the informal discussion presented about ADTs in Section 6.2—in particular Figure 6.1. Truncating a program after n steps is defined to be applying the finalisation operation after performing n operations. Applying finalisation after n steps of a program, *e.g.* $P_c \upharpoonright n$, can be considered in this framework as stopping the program and observing the n outputs that the program P_c has produced after consuming n inputs. Now consider stopping the program after it has consumed $n+1$ inputs, *e.g.* $P_c \upharpoonright (n+1)$. Clearly, the $n+1$ outputs that we observe from $P_c \upharpoonright (n+1)$ are made up of the n outputs that we could observe from $P_c \upharpoonright n$ plus one extra output that resulted from the $n+1^{th}$ operation. In other words, the $n+1^{th}$ operation cannot change any of the previous n outputs. Now, consider building the set $\{P_c \upharpoonright 0, P_c \upharpoonright 1, ..., P_c \upharpoonright n\}$. It follows from the argument above that this set is going to be "prefix complete". Thus, given $m \leq n$ we can see that $(i \upharpoonright n, o \upharpoonright n) \in P_c \upharpoonright n \Rightarrow (i \upharpoonright m, o \upharpoonright m) \in P_c \upharpoonright m$.

Case 2: Where $m > n$ we can infer the existence of a suitable sequence s from the assumption that the program P_c is defined over all input sequences. It follows from this assumption, that we could use the program P_c itself to calculate the sequence s (or at least a suitable prefix for s). In other words, we could run the program $P_c \upharpoonright m$ on the input $i \upharpoonright m$ and record the output o. Now, for any infinite sequence so, $o^\frown so$ is an appropriate witness to the existence of s.

Now ζ_1 is,

$$\dfrac{\dfrac{\dfrac{\dfrac{\zeta_2}{\vdots}}{(i,(o \restriction n)^\frown s) \in [\![P_a]\!]^{r\omega}_d}\,(df)}{\forall m \bullet (i \restriction m,((o \restriction n)^\frown s) \restriction m) \in [\![P_a \restriction m]\!]^{r}_d}\,(\forall^-) \qquad \dfrac{}{((o \restriction n)^\frown s) \restriction n = o \restriction n}\,(df)}{(i \restriction n,((o \restriction n)^\frown s) \restriction n) \in [\![P_a \restriction n]\!]^{r}_d}}{(i \restriction n, o \restriction n) \in [\![P_a \restriction n]\!]^{r}_d}$$

And ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{\forall m \bullet (i \restriction m,((o \restriction n)^\frown s) \restriction m) \in [\![P_c \restriction m]\!]^{r}_d}\,2}{(i,(o \restriction n)^\frown s) \in [\![P_c]\!]^{r\omega}_d}\,(df) \qquad \dfrac{\dfrac{[\![P_c]\!]^{r\omega}_d \subseteq [\![P_a]\!]^{r\omega}_d}{\forall i, o \bullet (i,o) \in [\![P_c]\!]^{r\omega}_d \Rightarrow (i,o) \in [\![P_a]\!]^{r\omega}_d}\,(\forall^-)}{(i,(o \restriction n)^\frown s) \in [\![P_c]\!]^{r\omega}_d \Rightarrow (i,(o \restriction n)^\frown s) \in [\![P_a]\!]^{r\omega}_d}}{(i,(o \restriction n)^\frown s) \in [\![P_a]\!]^{r\omega}_d}$$

For the $\Leftarrow$ direction we have,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(i,o) \in [\![P_c]\!]^{r\omega}_d}\,1}{\forall n \bullet (i \restriction n, o \restriction n) \in [\![P_c \restriction n]\!]^{r}_d}\,(df)}{(i \restriction n, o \restriction n) \in [\![P_c \restriction n]\!]^{r}_d}\,(\forall^-) \qquad \dfrac{\dfrac{\forall n \bullet [\![P_c \restriction n]\!]^{r}_d \subseteq [\![P_a \restriction n]\!]^{r}_d}{[\![P_c \restriction n]\!]^{r}_d \subseteq [\![P_a \restriction n]\!]^{r}_d}}{(i \restriction n, o \restriction n) \in [\![P_c \restriction n]\!]^{r}_d \Rightarrow (i \restriction n, o \restriction n) \in [\![P_a \restriction n]\!]^{r}_d}}{(i \restriction n, o \restriction n) \in [\![P_a \restriction n]\!]^{r}_d}\,(\Rightarrow^-)}{\forall n \bullet (i \restriction n, o \restriction n) \in [\![P_a \restriction n]\!]^{r}_d}\,(\forall^+)}{(i,o) \in [\![P_a]\!]^{r\omega}_d}\,(df)}{(i,o) \in [\![P_c]\!]^{r\omega}_d \Rightarrow (i,o) \in [\![P_a]\!]^{r\omega}_d}\,(\Rightarrow^+)(1)}{[\![P_c]\!]^{r\omega}_d \subseteq [\![P_a]\!]^{r\omega}_d}$$

B.11 Proofs for Section 6.4: Simulation and corresponding states

Proposition 6.4.1 Given arbitrary charts C_1, C_2, related simulation S, and bindings $z_1^{T_3}$, $z_2^{T_4}$, we have

$$\frac{z_1 \star z_2' \dot{\in} S}{z_1.i_{C_1} \cap in_{C_2} = z_2.i_{C_2} \cap in_{C_1}}\ (S^-_{io1}) \qquad \frac{z_1 \star z_2' \dot{\in} S}{z_1.o_{C_1} \cap out_{C_2} = z_2.o_{C_2} \cap out_{C_1}}\ (S^-_{io2})$$

$$\frac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}\ (S^-_{io3}) \qquad \frac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1}}\ (S^-_{io4})$$

$$\frac{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \quad z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}{z_1 \star z_2' \dot{\in} S}\ (S^+_{io})$$

where $[\![C_2]\!]^{T_1}_{z_C}$, $[\![C_1]\!]^{T_2}_{z_C}$, $T_1 \preceq T_3$, $T_2 \preceq T_4$ and T_3 and T_4 are disjoint.

Proof 6.4.1

For (S^-_{io1}) and (S^-_{io2}) we have,

$$\dfrac{\dfrac{\dfrac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \wedge IO_{C_2}^{C_1}}\ (S\text{-}df)}{z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}\ (S^-_{\wedge_1})}{z_1.i_{C_1} \cap in_{C_2} = z_2.i_{C_2} \cap in_{C_1}}\ (IO\text{-}df) \qquad \dfrac{\dfrac{\dfrac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \wedge IO_{C_2}^{C_1}}\ (S\text{-}df)}{z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}\ (S^-_{\wedge_1})}{z_1.o_{C_1} \cap out_{C_2} = z_2.o_{C_2} \cap out_{C_1}}\ (IO\text{-}df)$$

For (S^-_{io3}) and (S^-_{io4}) we have,

$$\dfrac{\dfrac{\dfrac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \wedge IO_{C_2}^{C_1}}\ (S\text{-}df)}{z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}\ (S^-_{\wedge_1})}{} \qquad \dfrac{\dfrac{\dfrac{z_1 \star z_2' \dot{\in} S}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \wedge IO_{C_2}^{C_1}}\ (S\text{-}df)}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1}}\ (S^-_{\wedge_0})}{}$$

And for (S^+_{io}) we have,

$$\dfrac{\dfrac{\dfrac{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \quad z_1 \star z_2' \dot{\in} IO_{C_2}^{C_1}}{z_1 \star z_2' \dot{\in} Corr_{C_2}^{C_1} \wedge IO_{C_2}^{C_1}}\ (S^+_{\wedge})}{z_1 \star z_2' \dot{\in} S}\ (S\text{-}df)}{}$$

B.12 Proofs for Section 6.5: Partial relation refinement

Proposition 6.5.1 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_1^{T_A^{in}}$ and $x_2^{T_C^{in}}$ we have,

$$(\langle x_1.i_A\rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (\langle x_2.i_C\rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S$$

$\Leftrightarrow$ [See proof below for detailed derivation]

$$\forall\, x_3 \bullet \exists\, x_4 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R$$

Proof 6.5.1

For the $\Rightarrow$ case we have:

$$(\langle x_1.i_A\rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (\langle x_2.i_C\rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S$$

$\Leftrightarrow$ [Definition of S]

$$z_1 \star z_2' \in Corr_C^A \wedge \exists\, i \bullet x_1.i_A = i \cap in_A \wedge x_2.i_C = i \cap in_C$$

$\Rightarrow$ [$\cap$-df]

$$z_1 \star z_2' \in Corr_C^A \wedge \exists\, i \bullet x_1.i_A \cap in_C = i \cap in_A \cap in_C \wedge x_2.i_C \cap in_A = i \cap in_C \cap in_A$$

$\Leftrightarrow$ [$i \cap in_A \cap in_C = i \cap in_A \cap in_C$ holds for all i]

$$z_1 \star z_2' \in Corr_C^A \wedge x_1.i_A \cap in_C = x_2.i_C \cap in_A$$

$\Leftrightarrow$ [$x_3 \in T_A^{out}$: $x_3.o_A \cap out_C = (x_3.o_A \cap out_C) \cap out_A$ holds for all out_A and out_C]

$$z_1 \star z_2' \in Corr_C^A \wedge \forall\, x_3 \bullet \exists\, x_4 \bullet x_1.i_A \cap in_C = x_2.i_C \cap in_A \wedge x_3.o_A \cap out_C = x_4.o_C \cap out_A$$

$\Leftrightarrow$ [Definition of IO_C^A]

$$z_1 \star z_2' \in Corr_C^A \wedge \forall\, x_3 \bullet \exists\, x_4 \bullet x_1 \star x_3 \star x_2' \star x_4' \in IO_C^A$$

$\Leftrightarrow$ [Definition of R]

$$\exists\, x_3, x_4 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R$$

And for the $\Leftarrow$ case we have:

$$\exists\, x_3, x_4 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R$$

$\Leftrightarrow$ [From the last three steps of the previous proof]

$$z_1 \star z_2' \in Corr_C^A \wedge x_1.i_A \cap in_C = x_2.i_C \cap in_A$$

$\Rightarrow$ [$\cap$-df, $=$-df, $\cup$-df]

$$z_1 \star z_2' \in Corr_C^A \wedge x_1.i_A = x_2.i_C \cap in_A \cup x_1.i_A \wedge x_2.i_C = x_1.i_A \cap in_C \cup x_2.i_C$$

$\Leftrightarrow$ [$x_1 \in T_A^{in} : x_1.i_A = x_1.i_A \cap in_A$, $x_2 \in T_C^{in} : x_2.i_C = x_2.i_C \cap in_C$]

$$z_1 \star z_2' \in Corr_C^A \wedge x_1.i_A = (x_2.i_C \cup x_1.i_A) \cap in_A \wedge x_2.i_C = (x_1.i_A \cup x_2.i_C) \cap in_C$$

$\Rightarrow$ [Using $(\exists^+)$]

$$z_1 \star z_2' \in Corr_C^A \wedge \exists\, i \bullet x_1.i_A = i \cap in_A \wedge x_2.i_C = i \cap in_C$$

$\Leftrightarrow$ [Definition of S]

$$(\langle x_1.i_A \rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (\langle x_2.i_C \rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S$$

Proposition 6.5.2 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_3^{T_A^{out}}$ and $x_4^{T_C^{out}}$ we have,

$$(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}{}^\frown\langle x_3.o_A \rangle, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}{}^\frown\langle x_4.o_C \rangle, z_2) \in S$$

$\Leftrightarrow$ [See proof below for detailed derivation]

$$\forall\, x_2 \bullet \exists\, x_1 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R$$

Proof 6.5.2

The proof of this proposition has the same form as that for Proposition 6.5.1 above with the appropriate changes to reflect that this proposition deals with the output interface of the respective charts rather than the input interfaces.

Proposition 6.5.3 For the arbitrary sequence si, and bindings $z_1^{U_C}$ and $z_2^{U_A}$ we have,

$$(si_{\triangleright(in_A)}, \langle\rangle, z_1) \mapsto (si_{\triangleright(in_C)}, \langle\rangle, z_2) \in S$$

$\Leftrightarrow$ [See proof below for detailed derivation]

$$z_1 \star z_2' \in Corr_C^A$$

Proof 6.5.3

Follows trivially from the definition of S.

B.13 Proofs for Section 6.6: Total chaos refinement

For the forward simulation *initialisation* condition we have:

Proposition 6.6.1

$$
\begin{array}{l}
CInit \subseteq AInit \fatsemi S \\
\quad \Leftrightarrow \quad \text{[Detailed derivation below]} \\
\forall\, y_c \bullet y_c \mathrel{\dot\in} Init_C \Rightarrow \exists\, t_1 \bullet t_1 \mathrel{\dot\in} Init_A \wedge t_1 \star y_c' \in R
\end{array}
$$

Proof 6.6.1

$$
\begin{array}{l}
CInit \subseteq AInit \fatsemi S \\
\quad \Leftrightarrow \quad [\subseteq\text{-df}] \\
\forall\, si, z, ssi, so \bullet (si, (ssi, so, z)) \in CInit \Rightarrow (si, (ssi, so, z)) \in AInit \fatsemi S \\
\quad \Leftrightarrow \quad [CInit \text{ pointwise restricts } si \text{ to signals in } in_C \text{ and deletes output}] \\
\forall\, si, z \bullet (si, (si_{\rhd (in_C)}, \langle\rangle, z)) \in CInit \Rightarrow (si, (si_{\rhd (in_C)}, \langle\rangle, z)) \in AInit \fatsemi S \\
\quad \Leftrightarrow \quad [\fatsemi\text{-df}] \\
\forall\, si, z \bullet (si, (si_{\rhd (in_C)}, \langle\rangle, z)) \in CInit \Rightarrow \exists\, ssi, t, so \bullet (si, (ssi, so, t)) \in AInit \wedge \\
\qquad\qquad ((ssi, so, t), (si_{\rhd (in_C)}, \langle\rangle, z)) \in S \\
\quad \Leftrightarrow \quad [AInit \text{ pointwise restricts } si \text{ to signals in } in_A \text{ and deletes output}] \\
\forall\, si, z \bullet (si, (si_{\rhd (in_C)}, \langle\rangle, z)) \in CInit \Rightarrow \exists\, t \bullet (si, (si_{\rhd (in_A)}, \langle\rangle, t)) \in AInit \wedge \\
\qquad\qquad ((si_{\rhd (in_A)}, \langle\rangle, t), (si_{\rhd (in_C)}, \langle\rangle, z)) \in S \\
\quad \Leftrightarrow \quad \text{[Definitions of } CInit \text{ and } AInit\text{, and Proposition 6.5.3]} \\
\forall\, z \bullet z \in Init_C \Rightarrow \exists\, t \bullet t \in Init_A \wedge t \star z' \in Corr_C^A \\
\quad \Leftrightarrow \quad \text{[Definitions of } R \text{ and } \dot\in\text{]} \\
\forall\, z, x_{ic}, x_{oc} \bullet z \star x_{ic} \star x_{oc} \mathrel{\dot\in} Init_C \Rightarrow \exists\, t, x_{ia}, x_{oa} \bullet t \star x_{ia} \star x_{oa} \mathrel{\dot\in} Init_A \wedge \\
\qquad\qquad t \star x_{ia} \star x_{oa} \star z' \star x_{ic}' \star x_{oc}' \in R \\
\quad \Leftrightarrow \quad [\text{Let } y_c = z \star x_{ic} \star x_{oc},\ t_1 = t \star x_{ia} \star x_{oa}] \\
\forall\, y_c \bullet y_c \mathrel{\dot\in} Init_C \Rightarrow \exists\, t_1 \bullet t_1 \mathrel{\dot\in} Init_A \wedge t_1 \star y_c' \in R
\end{array}
$$

The condition for *finalisation* holds *iff* $out_A \subseteq out_C$.

Proposition 6.6.2

$$S \fatsemi CFin \subseteq AFin$$

$\Leftrightarrow$ [Detailed derivation below]

$out_A \subseteq out_C$

Proof 6.6.2

We split the proof into cases and prove the contrapositive property in both cases. For the $\Rightarrow$ case we have:[4]

$\neg\ out_A \subseteq out_C$

$\Leftrightarrow$ [Definition of pointwise restriction]

$\exists\, so \bullet so_{\triangleright(out_A)} \neq so_{\triangleright(out_C)\triangleright(out_A)}$

$\Leftrightarrow$ [Definition of $AFin$]

$\forall\, si, z_1 \bullet \exists\, so \bullet ((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \notin AFin$

$\Leftrightarrow$ [Definition of CFin: $\forall\, si, z_2, so \bullet ((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2), so) \in CFin$]

$\forall\, si, z_1, z_2 \bullet \exists\, so \bullet ((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \notin AFin \wedge$

$((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2), so) \in CFin$

[Definition of S:

$\Rightarrow$ $\exists\, z_1, z_2 \bullet \forall\, si, so \bullet$

$((si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1), (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2)) \in S$]

$\exists\, si, z_1, z_2, so \bullet ((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \notin AFin \wedge$

$((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2), so) \in CFin$

$((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), (si_{\triangleright(in_C)}, (so_{\triangleright(out_C)})_{\triangleright(out_C)}, z_2)) \in S$

$\Leftrightarrow$ [Definition of $\triangleright(\_)$: $(so_{\triangleright(out_C)})_{\triangleright(out_C)} = so_{\triangleright(out_C)}$]

$\exists\, si, z_1, z_2, so \bullet ((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \notin AFin \wedge$

$((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2), so) \in CFin$

$((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2)) \in S$

$\Leftrightarrow$ [Definition of $\fatsemi$]

$\exists\, si, z_1, so \bullet ((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \notin AFin \wedge$

$((si_{\triangleright(in_A)}, (so_{\triangleright(out_C)})_{\triangleright(out_A)}, z_1), so) \in S \fatsemi CFin$

$\Rightarrow$ [$\subseteq$-df]

$\neg\ S \fatsemi CFin \subseteq AFin$

And for the $\Leftarrow$ case we have:

$\neg\ S \fatsemi CFin \subseteq AFin$

[4]Notice that the justification for the forth step in this proof assumes that the relation S is not the empty relation. By definition charts must have at least one state (an initial state). Moreover, the corresponding relation $Corr_C^A$ always relates the initial states of charts A and C. Therefore, by definition, S cannot be the empty relation.

$\Leftrightarrow$ [$\subseteq$-df, Predicate calculus]

$\exists\, si, so, sso, z_1 \bullet ((si, so, z_1), sso) \in S \fatsemi CFin \wedge ((si, so, z_1), sso) \notin AFin$

$\Leftrightarrow$ [$\fatsemi$-df]

$\exists\, si, so, sso, ti, to, z_1, z_2 \bullet ((si, so, z_1), (ti, to, z_2)) \in S \wedge ((ti, to, z_2), sso) \in CFin \wedge$
$((si, so, z_1), sso) \notin AFin$

$\Rightarrow$ [Definitions of S, $CFin$ and $AFin$]

$\exists\, vo, si, so \bullet vo_{\rhd(out_A)} = so \wedge vo_{\rhd(out_C)} = to \wedge sso_{\rhd(out_C)} = to \wedge \neg\, sso_{\rhd(out_A)} = so$

$\Rightarrow$ [Definitions of S, $CFin$ and $AFin$]

$\exists\, vo, sso \bullet vo_{\rhd(out_C)} = sso_{\rhd(out_C)} \wedge \neg\, vo_{\rhd(out_A)} = sso_{\rhd(out_A)}$

$\Leftrightarrow$ [Definition of pointwise restriction]

$\neg\, out_A \subseteq out_C$

The precondition of C coincides with the domain of relation $CStep$.[5]

Proposition 6.6.4

$(x_{ic}.i_C \frown si_{\rhd(in_C)}, so, z) \in \mathrm{dom}\ CStep$

$\Leftrightarrow$ [Detailed derivation below]

$Pre\ C\ (z \star x_{ic} \star x_{oc})$

Proof 6.6.4

$(x_{ic}. \frown si_{\rhd(in_C)}, so, z) \in \mathrm{dom}\ CStep$

$\Leftrightarrow$ [dom-df]

$\exists\, ssi, sso, x_{oe}, t \bullet (x_{ic}. \frown si_{\rhd(in_C)}, so, z) \mapsto (ssi, sso \frown x_{oe}., t) \in CStep$

$\Leftrightarrow$ [Definition of $CStep$]

$\exists\, x_{oe}, t \bullet z \star x_{ic} \star t' \star x'_{oe} \in C$

$\Leftrightarrow$ [$\dot{\in}$-df]

$\forall\, x_{oc} \bullet \exists\, x_{oe}, t \bullet z \star x_{ic} \star x_{oc} \star t' \star x'_{oe} \mathrel{\dot{\in}} C$

$\Leftrightarrow$ [Pre-df]

$Pre\ C\ (z \star x_{ic} \star x_{oc})$

[5]We introduce the notational convenience that is to use a binding followed by a full stop to represent the value of the binding's single observation, for example $x_{ic}.$ is used in place of $x_{ic}.i_C$. Also, when concatenating a single element to a sequence we omit the sequence brackets, that is $i \frown si$ represents $\langle i \rangle \frown si$.

For the *applicability* condition we have:

Proposition 6.6.5

$$\text{ran}((\text{dom } AStep) \lhd S) \subseteq \text{dom } CStep$$

$$\Leftrightarrow \quad \text{[Detailed derivation below]}$$

$$\forall y_a, y_c \bullet Pre\ ASys\ y_a \wedge y_a \star y_c' \in R \Rightarrow Pre\ CSys\ y_c$$

Proof 6.6.5

$$\text{ran}((\text{dom } AStep) \lhd S) \subseteq \text{dom } CStep$$

$$\Leftrightarrow \quad [\subseteq\text{-df}]$$

$$\forall x_{ic}, si, so, z_2 \bullet (x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{ran}((\text{dom } AStep) \lhd S) \Rightarrow$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{dom } CStep$$

$$\Leftrightarrow \quad \text{[ran-df]}$$

$$\forall x_{ic}, si, so, z_2 \bullet (\exists x_{ia}, ssi, sso, z_1 \bullet$$
$$(x_{ia}^\frown ssi_{\rhd(in_A)}, sso_{\rhd(out_A)}, z_1) \mapsto$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in (\text{dom } AStep) \lhd S) \Rightarrow$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{dom } CStep$$

$$\Leftrightarrow \quad \text{[Predicate calculus]}$$

$$\forall x_{ic}, si, so, z_2, x_{ia}, ssi, sso, z_1 \bullet$$
$$(x_{ia}^\frown ssi_{\rhd(in_A)}, sso_{\rhd(out_A)}, z_1) \mapsto$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in (\text{dom } AStep) \lhd S \Rightarrow$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{dom } CStep$$

$$\Leftrightarrow \quad \text{[Definition of } S\text{: } ssi_{\rhd(in_A)} = si_{\rhd(in_A)} \text{ and } sso_{\rhd(out_A)} = so_{\rhd(out_A)}\text{]}$$

$$\forall x_{ic}, si, so, z_2, x_{ia}, z_1 \bullet (x_{ia}^\frown si_{\rhd(in_A)}, so_{\rhd(out_A)}, z_1) \mapsto$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in (\text{dom } AStep) \lhd S \Rightarrow$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{dom } CStep$$

$$\Leftrightarrow \quad [\lhd\text{-df}]$$

$$\forall x_{ic}, si, so, z_2, x_{ia}, z_1 \bullet ((x_{ia}^\frown si_{\rhd(in_A)}, so_{\rhd(out_A)}, z_1) \in \text{dom } AStep \wedge$$
$$(x_{ia}^\frown si_{\rhd(in_A)}, so_{\rhd(out_A)}, z_1) \mapsto (x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in S) \Rightarrow$$
$$(x_{ic}^\frown si_{\rhd(in_C)}, so_{\rhd(out_C)}, z_2) \in \text{dom } CStep$$

$$\Leftrightarrow \quad \text{[}Pre \text{ and dom coincide, Proposition 6.5.1, Predicate calculus]}$$

$$\forall x_{ic}, x_{oc}, z_2, x_{ia}, x_{oa}, z_1 \bullet (Pre\ ASys\ (z_1 \star x_{ia} \star x_{oa}) \wedge$$
$$z_1 \star x_{ia} \star x_{oa} \star z_2' \star x_{ic}' \star x_{oc}' \in R) \Rightarrow Pre\ CSys\ (z_2 \star x_{ic})$$
$$\Leftrightarrow \quad [\text{Let } y_a = z_1 \star x_{ia} \star x_{oa},\ y_c = z_2 \star x_{ic} \star x_{oc}]$$
$$\forall y_a, y_c \bullet Pre\ ASys\ y_a \wedge y_a \star y_c' \in R \Rightarrow Pre\ CSys\ y_c$$

Finally, for the *correctness* condition we have:

Proposition 6.6.6

$$((\text{dom } AStep) \lhd S) \fatsemi CStep \subseteq AStep \fatsemi S$$
$$\Leftrightarrow \quad [\text{Detailed derivation below}]$$
$$\forall y_a, y_c, \vdash_{\mathcal{Z}_C} \bullet (Pre\ A\ y_a \wedge y_a \star y_c' \in R \wedge y_c \star \vdash_{\mathcal{Z}_C}{}' \dot{\in} C) \Rightarrow$$
$$\exists t \bullet y_a \star t' \dot{\in} A \wedge t \star \vdash_{\mathcal{Z}_C}{}' \in R$$

Proof 6.6.6

$$((\text{dom } AStep) \lhd S) \fatsemi CStep \subseteq AStep \fatsemi S$$
$$\Leftrightarrow \quad [\subseteq\text{-df}]$$
$$\forall x_{ia}, si, so, z_1, ssi, x_{oc}, sso, z_3 \bullet$$
$$(x_{ia}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in ((\text{dom } AStep) \lhd S) \fatsemi CStep \Rightarrow$$
$$(x_{ia}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in AStep \fatsemi S$$
$$\Leftrightarrow \quad [\fatsemi\text{-df}]$$
$$\forall x_{ia}, si, so, z_1, ssi, x_{oc}, sso, z_3 \bullet (\exists x_{im}, ti, to, z_2 \bullet$$
$$(x_{ia}^\frown si, so, z_1) \mapsto (x_{im}^\frown ti, to, z_2) \in (\text{dom } AStep) \lhd S \wedge$$
$$(x_{im}^\frown ti, to, z_2) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in CStep) \Rightarrow$$
$$\exists tti, x_{oe}, tto, y \bullet (x_{ia}^\frown si, so, z_1) \mapsto (tti, tto^\frown x_{oe}., y) \in AStep \wedge$$
$$(tti, tto^\frown x_{oe}., y') \mapsto (ssi, sso^\frown x_{oc}., z_3) \in S$$
$$\Leftrightarrow \quad [\text{Predicate calculus}]$$
$$\forall x_{ia}, si, so, z_1, ssi, x_{oc}, sso, z_3, x_{im}, ti, to, z_2 \bullet$$
$$((x_{ia}^\frown si, so, z_1) \mapsto (x_{im}^\frown ti, to, z_2) \in (\text{dom } AStep) \lhd S \wedge$$
$$(x_{im}^\frown ti, to, z_2) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in CStep) \Rightarrow$$
$$\exists tti, x_{oe}, tto, y \bullet (x_{ia}^\frown si, so, z_1) \mapsto (tti, tto^\frown x_{oe}., y) \in AStep \wedge$$
$$(tti, tto^\frown x_{oe}., y) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in S$$
$$\Leftrightarrow \quad [\lhd\text{-df}]$$

$$\forall x_{ia}, si, so, z_1, ssi, x_{oc}, sso, z_3, x_{im}, ti, to, z_2 \bullet$$

$$((x_{ia}^\frown si, so, z_1) \in \mathrm{dom}\, AStep \wedge ((x_{ia}^\frown si, so, z_1) \mapsto (x_{im}^\frown ti, to, z_2) \in S \wedge$$

$$(x_{im}^\frown ti, to, z_2) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in CStep) \Rightarrow$$

$$\exists tti, x_{oe}, tto, y \bullet (x_{ia}^\frown si, so, z_1) \mapsto (tti, tto^\frown x_{oe}., y) \in AStep \wedge$$

$$(tti, tto^\frown x_{oe}., y) \mapsto (ssi, sso^\frown x_{oc}., z_3) \in S$$

$\Leftrightarrow$ [Definitions of S and $CStep$]

$$\forall x_{ia}, si, so, z_1, x_{oc}, z_3, x_{im}, z_2 \bullet ((x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \in \mathrm{dom}\, AStep \wedge$$

$$((x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (x_{im}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S \wedge$$

$$(x_{im}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oc}., z_3) \in CStep) \Rightarrow$$

$$\exists tti, x_{oe}, tto, y \bullet (x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (tti, tto^\frown x_{oe}., y) \in AStep \wedge$$

$$(tti, tto^\frown x_{oe}., y) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oc}., z_3) \in S$$

$\Leftrightarrow$ [Definition of $AStep$]

$$\forall x_{ia}, si, so, x_{oc}, x_{im}, z_1, z_2, z_3 \bullet ((x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \in \mathrm{dom}\, AStep \wedge$$

$$(x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (x_{im}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \in S \wedge$$

$$(x_{im}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_2) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oc}., z_3) \in CStep) \Rightarrow$$

$$\exists x_{oe}, y \bullet$$

$$(x_{ia}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_1) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oe}., y) \in AStep \wedge$$

$$(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oe}., y) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oc}., z_3) \in S$$

$\Leftrightarrow$ [*Pre* and dom coincide, Proposition 6.5.1, definitions of $CStep$ and $AStep$, Proposition 6.5.2]

$$\forall x_{ia}, x_{im}, x_{ie}, x_{oc}, x_{oa}, z_1, z_2, z_3 \bullet \exists x_{om} \bullet (Pre\ A\ (z_1 \star x_{ia} \star x_{oa}) \wedge$$

$$z_1 \star x_{ia} \star x_{oa} \star z_2' \star x_{im}' \star x_{om}' \in R \wedge z_2 \star x_{im} \star z_3' \star x_{oc}' \in C) \Rightarrow$$

$$\exists x_{oe}, x_{ic}, y \bullet z_1 \star x_{ia} \star y' \star x_{oe}' \in A \wedge y \star x_{ic} \star x_{oe} \star z_3' \star x_{ie}' \star x_{oc}' \in R$$

$\Leftrightarrow$ [Predicate calculus, definition of $\dot{\in}$]

$$\forall x_{ia}, x_{im}, x_{ie}, x_{oc}, x_{oa}, x_{om}, z_1, z_2, z_3 \bullet (Pre\ A\ (z_1 \star x_{ia} \star x_{oa}) \wedge$$

$$z_1 \star x_{ia} \star x_{oa} \star z_2' \star x_{im}' \star x_{om}' \in R \wedge z_2 \star x_{im} \star x_{om} \star z_3' \star x_{ie}' \star x_{oc}' \dot{\in} C) \Rightarrow$$

$$\exists x_{oe}, x_{ic}, y \bullet z_1 \star x_{ia} \star x_{oa} \star y' \star x_{ic}' \star x_{oe}' \dot{\in} A \wedge y \star x_{ic} \star x_{oe} \star z_3' \star x_{ie}' \star x_{oc}' \in R$$

$\Leftrightarrow$ [Let $y_a = z_1 \star x_{ia} \star x_{oa}$, $y_c = z_2 \star x_{im} \star x_{om}$, $\vdash_{Z_C} = z_3 \star x_{ie} \star x_{oc}$ and $t = y \star x_{ic} \star x_{oe}$]

$$\forall y_a, y_c, \vdash_{Z_C} \bullet (Pre\ A\ y_a \wedge y_a \star y_c' \in R \wedge y_c \star \vdash_{Z_C}' \dot{\in} C) \Rightarrow$$

$$\exists t \bullet y_a \star t' \dot{\in} A \wedge t \star \vdash_{Z_C}' \in R$$

Proposition 6.6.8 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_1^{T_C^{in}}$ and $x_2^{T_A^{in}}$ we have,

$$(\langle x_1.i_C\rangle^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (\langle x_2.i_A\rangle^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in T$$

$$\Leftrightarrow \quad \text{[See proof below]}$$

$$\forall x_3 \bullet \exists x_4 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R^{-1}$$

Proof 6.6.8

The proof of this proposition has identical form to the proof for Proposition 6.5.1, replacing all occurrences of S with T, R with R^{-1}, $Corr_C^A$ with $Corr_A^C$ and IO_C^A with IO_A^C.

Proposition 6.6.9 For arbitrary sequences si and so, and bindings $z_1^{U_C}$, $z_2^{U_A}$, $x_3^{T_C^{out}}$ and $x_4^{T_A^{out}}$ we have,

$$(si_{\triangleright(in_A)}, so_{\triangleright(out_A)}{}^\frown\langle x_3.o_A\rangle, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}{}^\frown\langle x_4.o_C\rangle, z_2) \in T$$

$$\Leftrightarrow \quad []$$

$$\forall x_2 \bullet \exists x_1 \bullet z_1 \star x_1 \star x_3 \star z_2' \star x_2' \star x_4' \in R^{-1}$$

Proof 6.6.9

For the $\Rightarrow$ case we have:

$$(si_{\triangleright(out_C)}, so_{\triangleright(out_C)}{}^\frown\langle x_4.o_C\rangle, z_2) \mapsto (si_{\triangleright(out_A)}, so_{\triangleright(out_A)}{}^\frown\langle x_3.o_A\rangle, z_1) \in T$$

$$\Leftrightarrow \quad \text{[Definition of } T\text{]}$$

$$z_1 \star z_2' \in Corr_A^C \wedge \exists o \bullet x_3.o_A = o \cap out_A \wedge x_4.o_C = o \cap out_C$$

$$\Rightarrow \quad [\cap\text{-df}]$$

$$z_1 \star z_2' \in Corr_A^C \wedge$$
$$\exists o \bullet x_3.o_A \cap out_C = o \cap out_A \cap out_C \wedge x_4.o_C \cap out_A = o \cap out_C \cap out_A$$

$$\Leftrightarrow \quad [o \cap out_A \cap out_C = o \cap out_A \cap out_C \text{ holds for all } o]$$

$$z_1 \star z_2' \in Corr_A^C \wedge x_3.o_A \cap out_C = x_4.o_C \cap out_A$$

$$\Leftrightarrow \quad [x_3 \in T_A^{out}\text{: } x_3.o_A \cap out_C = (x_3.o_A \cap out_C) \cap out_A \text{ holds for all } out_A \text{ and } out_C]$$

$$z_1 \star z_2' \in Corr_A^C \wedge$$
$$\forall x_3 \bullet \exists x_4 \bullet x_3.o_A \cap out_C = x_4.o_C \cap out_A \wedge x_3.o_A \cap out_C = x_4.o_C \cap out_A$$

$$\Leftrightarrow \quad \text{[Definition of } IO_A^C\text{]}$$

$$z_1 \star z_2' \in Corr_A^C \wedge \forall x_3 \bullet \exists x_4 \bullet x_3 \star x_3 \star x_4' \star x_4' \in IO_A^C$$

$$\Leftrightarrow \quad \text{[Definition of } R^{-1}\text{]}$$

$$\exists x_3, x_4 \bullet z_1 \star x_3 \star x_3 \star z_2' \star x_4' \star x_4' \in R^{-1}$$

And for the $\Leftarrow$ case we have:

$$\exists x_3, x_4 \bullet z_1 \star x_3 \star x_3 \star z_2' \star x_4' \star x_4' \in R^{-1}$$

$\Leftrightarrow$ [From the last three steps of the previous proof]

$$z_1 \star z_2' \in Corr_A^C \wedge x_3.o_A \cap out_C = x_4.o_C \cap out_A$$

$\Rightarrow$ [$\cap$-df, $=$-df, $\cup$-df]

$$z_1 \star z_2' \in Corr_A^C \wedge x_3.o_A = x_4.o_C \cap out_A \cup x_3.o_A \wedge x_4.o_C = x_3.o_A \cap out_C \cup x_4.o_C$$

$\Leftrightarrow$ [$x_3 \in T_A^{out} : x_3.o_A = x_3.o_A \cap out_A,\ x_4 \in T_C^{out} : x_4.o_C = x_4.o_C \cap out_C$]

$$z_1 \star z_2' \in Corr_A^C \wedge x_3.o_A = (x_4.o_C \cup x_3.o_A) \cap out_A \wedge x_4.o_C = (x_3.o_A \cup x_4.o_C) \cap out_C$$

$\Rightarrow$ [Using $(\exists^+)$]

$$z_1 \star z_2' \in Corr_A^C \wedge \exists o \bullet x_3.o_A = o \cap out_A \wedge x_4.o_C = o \cap out_C$$

$\Leftrightarrow$ [Definition of T]

$$(si_{\triangleright(out_C)}, so_{\triangleright(out_C)} \frown \langle x_4.o_C \rangle, z_2) \mapsto (si_{\triangleright(out_A)}, so_{\triangleright(out_A)} \frown \langle x_3.o_A \rangle, z_1) \in T$$

Proposition 6.6.10 states that the *initialisation* condition for backwards simulation refinement, in a relational ADT embedding for charts, holds if and only if the input interface of the abstract chart is a subset of the input interface of the concrete chart.

Proposition 6.6.10 For arbitrary charts A and C we have,

$$CInit \,{}_{\S}\, T \subseteq AInit$$

$\Leftrightarrow$ [Detailed derivation below]

$$in_A \subseteq in_C \wedge \forall y, z \bullet (y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1}) \Rightarrow z \mathrel{\dot\in} Init_A$$

Proof 6.6.10

Again, the proof is split into cases and we prove the contrapositive property in both cases. For the $\Rightarrow$ case we have:

$$\neg\, (in_A \subseteq in_C \wedge \forall y, z \bullet (y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1}) \Rightarrow z \mathrel{\dot\in} Init_A)$$

$\Leftrightarrow$ [Predicate calculus]

$$\neg\, in_A \subseteq in_C \vee \exists y, z \bullet y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1} \wedge \neg\, z \mathrel{\dot\in} Init_A$$

Using a similar proof to the first case for Lemma 6.6.2, we have

$$\neg\, in_A \subseteq in_C$$

$\Leftrightarrow$ [Definition of pointwise restriction]

$$\exists\, si \bullet si_{\triangleright(in_A)} \neq (si_{\triangleright(in_C)})_{\triangleright(in_A)}$$

$\Leftrightarrow$ [Definition of $AInit$]

$$\forall\, z_1 \bullet \exists\, si \bullet (si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \notin AInit$$

$\Leftrightarrow$ [Definition of CInit]

$$\forall\, z_1, z_2 \bullet \exists\, si \bullet (si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \notin AInit \wedge (si, (si_{\triangleright(in_C)}, \langle\rangle, z_2)) \in CInit$$

$\Rightarrow$ [Definition of T]

$$\exists\, z_1, z_2, si \bullet (si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \notin AInit \wedge (si, (si_{\triangleright(in_C)}, \langle\rangle, z_2)) \in CInit \wedge$$
$$(((si_{\triangleright(in_C)})_{\triangleright(in_C)}, \langle\rangle, z_2), (((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \in T$$

$\Leftrightarrow$ [Definition of $\triangleright(\_)$: $(si_{\triangleright(in_C)})_{\triangleright(in_C)} = si_{\triangleright(in_C)}$]

$$\exists\, z_1, z_2, si \bullet (si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \notin AInit \wedge (si, (si_{\triangleright(in_C)}, \langle\rangle, z_2)) \in CInit \wedge$$
$$((si_{\triangleright(in_C)}, \langle\rangle, z_2), ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \in T$$

$\Leftrightarrow$ [Definition of $\fatsemi$]

$$\exists\, z_1, si \bullet (si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \notin AInit \wedge$$
$$(si, ((si_{\triangleright(in_C)})_{\triangleright(in_A)}, \langle\rangle, z_1)) \in CInit \fatsemi T$$

$\Rightarrow$ [$\subseteq$-df]

$$\neg\, CInit \fatsemi T \subseteq AInit$$

Also, we have:

$$\exists\, y, z \bullet y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1} \wedge \neg\, z \mathrel{\dot\in} Init_A$$

$\Rightarrow$ [Definitions of $CInit$, R^{-1}, T and $AInit$]

$$\exists\, si, y, z \bullet (si, (si_{\triangleright(in_C)}, \langle\rangle, y \upharpoonright U_C)) \in CInit \wedge$$
$$((si_{\triangleright(in_C)}, \langle\rangle, y \upharpoonright U_C), (si_{\triangleright(in_A)}, \langle\rangle, z \upharpoonright U_A)) \in T \wedge$$
$$(si, (si_{\triangleright(in_A)}, \langle\rangle, z \upharpoonright U_A)) \notin AInit$$

$\Rightarrow$ [$\fatsemi$-df and $\subseteq$-df]

$$\neg\, CInit \fatsemi T \subseteq AInit$$

Now, using disjunction elimination, we have,

$$\neg\, (in_A \subseteq in_C \wedge \forall\, y, z \bullet (y \mathrel{\dot\in} Init_C \wedge y \star z' \in R^{-1}) \Rightarrow z \mathrel{\dot\in} Init_A)$$

$\Rightarrow$

$$\neg\, CInit \fatsemi T \subseteq AInit$$

For the $\Leftarrow$ case we have:

$$\neg\, CInit \fatsemi T \subseteq AInit$$

$\Leftrightarrow$ [$\subseteq$-df]

$\exists\, si, so, ssi, z_2 \bullet (si, (ssi, so, z_2)) \in CInit \fatsemi T \wedge (si, (ssi, so, z_2)) \notin AInit$

$\Leftrightarrow$ [$\fatsemi$-df]

$\exists\, si, so, ssi, ti, to, z_1, z_2 \bullet (si, (ti, to, z_1)) \in CInit \wedge ((ti, to, z_1), (ssi, so, z_2)) \in T$
$\wedge\, (si, (ssi, so, z_2)) \notin AInit$

$\Rightarrow$ [Definitions of $CInit$, T and $AInit$]

$\exists\, vi, si, so, ssi, ti, to, z_1, z_2 \bullet$
$(si_{\triangleright(in_C)} = ti \wedge vi_{\triangleright(in_C)} = ti \wedge vi_{\triangleright(in_A)} = ssi \wedge si_{\triangleright(in_A)} \neq ssi) \vee$
$(z_1 \in Init_C \wedge z_1 \star z_2 \in Corr_A^C \wedge z_2 \notin Init_A)$

$\Rightarrow$ [Definitions of R^{-1} and $\dot{\in}$]

$\exists\, vi, si, \bullet (si_{\triangleright(in_C)} = vi_{\triangleright(in_C)} \wedge vi_{\triangleright(in_A)} \neq si_{\triangleright(in_A)}) \vee$
$\exists\, y, z \bullet y \mathrel{\dot{\in}} Init_C \wedge y \star z \in R^{-1} \wedge \neg\, z \mathrel{\dot{\in}} Init_A$

$\Leftrightarrow$

$\neg\, iniA \subseteq in_C \vee \exists\, y, z \bullet y \mathrel{\dot{\in}} Init_C \wedge y \star z \in R^{-1} \wedge \neg\, z \mathrel{\dot{\in}} Init_A$

$\Leftrightarrow$ [Predicate calculus]

$\neg\, (in_A \subseteq in_C \wedge \forall\, y, z \bullet (y \mathrel{\dot{\in}} Init_C \wedge (y, z) \in R^{-1}) \Rightarrow z \mathrel{\dot{\in}} Init_A)$

For the backwards simulation *finalisation* condition we have:

Proposition 6.6.11 For arbitrary charts A and C we have,

$CFin \subseteq T \fatsemi AFin$

$\Leftrightarrow$ [Detailed derivation below]

$\forall\, y_c \bullet \exists\, t_1 \bullet y_c \star t_1' \in R^{-1}$

Proof 6.6.11

$CFin \subseteq T \fatsemi AFin$

$\Leftrightarrow$ [$\subseteq$-df]

$\forall\, si, z, so, sso \bullet ((si, so, z), sso) \in CFin \Rightarrow ((si, so, z), sso) \in T \fatsemi AFin$

$\Leftrightarrow$ [$\fatsemi$-df]

$\forall\, si, z, so, sso \bullet ((si, so, z), sso) \in CFin \Rightarrow$
$\exists\, ti, to, y \bullet (si, so, z) \mapsto (ti, to, y) \in T \wedge ((ti, to, y), sso) \in AFin$

$\Leftrightarrow$ [Definition of $CFin$]

$\forall\, si, z, so \bullet ((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z), so) \in CFin \Rightarrow$
$\exists\, ti, to, y \bullet (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z) \mapsto (ti, to, y) \in T \wedge$
$((ti, to, y), so) \in AFin$

$\Leftrightarrow$ [Definition of T]

$\forall si, z, so \bullet ((si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z), so) \in CFin \Rightarrow$

$\quad \exists y \bullet (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, y) \in T \wedge$

$\quad\quad ((si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, y), so) \in AFin$

$\Leftrightarrow$ [$CFin$ and $AFin$ are total]

$\forall z \bullet \exists y \bullet (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, y) \in T$

$\Leftrightarrow$ [Definition of T]

$\forall z \bullet \exists y \bullet z \star y' \in Corr_A^C$

$\Leftrightarrow$ [Definition of IO]

$\forall y_1, z_1 \bullet \exists y_2, z_2 \bullet z_1 \star z_2' \in Corr_A^C \wedge y_1 \star y_2' \in IO_A^C$

$\Leftrightarrow$ [Definition of R^{-1}]

$\forall y_1, z_1 \bullet \exists y_2, z_2 \bullet z_1 \star y_1 \star z_2' \star y_2' \in R^{-1}$

$\Leftrightarrow$ [Let $y_c = z_1 \star y_1$ and $t_1 = y_2 \star z_2$]

$\forall y_c \bullet \exists t_1 \bullet y_c \star t_1' \in R^{-1}$

Lemma 6.6.12 For arbitrary sequences si and so, and bindings $x_{ic}^{T_C^{in}}$, $z_1^{U_C}$, we have,

$(x_{ic}.i_C \frown si, so, z_1) \notin \mathrm{dom}(T \rhd \mathrm{dom}\, AStep)$

$\Leftrightarrow$

$\forall x_{im}, z_2, x_{om}, x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_2 \star x_{im} \star x_{om})$

Proof 6.6.12

$(x_{ic} \frown si, so, z_1) \notin \mathrm{dom}(T \rhd \mathrm{dom}\, AStep)$

$\Leftrightarrow$ [dom-df]

$\neg\ \exists x_{im}, ssi, sso, z_2 \bullet (x_{ic} \frown si, so, z_1) \mapsto (x_{im} \frown ssi, sso, z_2) \in T \rhd \mathrm{dom}\, AStep$

$\Leftrightarrow$ [$\rhd$-df]

$\neg\ \exists x_{im}, ssi, sso, z_2 \bullet (x_{ic} \frown si, so, z_1) \mapsto (x_{im} \frown ssi, sso, z_2) \in T \wedge$

$\quad (x_{im} \frown ssi, sso, z_2) \notin \mathrm{dom}\, AStep$

$\Leftrightarrow$ [Predicate calculus]

$\forall x_{im}, ssi, sso, z_2 \bullet (x_{ic} \frown si, so, z_1) \mapsto (x_{im} \frown ssi, sso, z_2) \in T \Rightarrow$

$\quad (x_{im} \frown ssi, sso, z_2) \in \mathrm{dom}\, AStep$

$\Leftrightarrow$ [definition of T]

$\forall x_{im}, z_2 \bullet (x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (x_{im}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in T \Rightarrow$

$(x_{im}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in \mathrm{dom}\, AStep$

$\Leftrightarrow$ [Proposition 6.6.8]

$\forall x_{im}, z_2 \bullet (\forall x_{om} \bullet \exists x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1}) \Rightarrow$

$(x_{im}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in \mathrm{dom}\, AStep$

$\Leftrightarrow$ [Predicate calculus]

$\forall x_{im}, z_2, x_{om}, x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow$

$(x_{im}^\frown si_{\triangleright(in_A)}, so_{\triangleright(out_A)}, z_2) \in \mathrm{dom}\, AStep$

$\Leftrightarrow$ [*Pre* and dom coincide]

$\forall x_{im}, z_2, x_{om}, x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_2 \star x_{im} \star x_{om})$

For the backwards simulation *applicability* we have:

Proposition 6.6.13 For arbitrary charts A and C we have,

$\overline{\mathrm{dom}\, CStep} \subseteq \mathrm{dom}(T \rhd \mathrm{dom}\, AStep)$

$\Leftrightarrow$ [Detailed derivation below]

$\forall y_c \bullet (\forall v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a) \Rightarrow Pre\ C\ y_c$

Proof 6.6.13

$\overline{\mathrm{dom}\, CStep} \subseteq \mathrm{dom}(T \rhd \mathrm{dom}\, AStep)$

$\Leftrightarrow$ [$\subseteq$-df and definition of set compliment]

$\forall x_{ic}, si, so, z_1 \bullet (x_{ic}^\frown si, so, z_1) \notin \mathrm{dom}\, CStep \Rightarrow$

$(x_{ic}^\frown si, so, z_1) \in \mathrm{dom}(T \rhd \mathrm{dom}\, AStep)$

$\Leftrightarrow$ [contraposition]

$\forall x_{ic}, si, so, z_1 \bullet (x_{ic}^\frown si, so, z_1) \notin \mathrm{dom}(T \rhd \mathrm{dom}\, AStep) \Rightarrow$

$(x_{ic}^\frown si, so, z_1) \in \mathrm{dom}\, CStep$

$\Leftrightarrow$ [Lemma 6.6.12]

$\forall x_{ic}, si, so, z_1 \bullet (\forall x_{im}, z_2, x_{om}, x_{oc} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow$

$Pre\ A\ (z_2 \star x_{im} \star x_{om})) \Rightarrow (x_{ic}^\frown si, so, z_1) \in \mathrm{dom}\, CStep$

$\Leftrightarrow$ [*Pre* and dom coincide]

$\forall x_{ic}, x_{oc}, z_1 \bullet (\forall x_{im}, z_2, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_2' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow$

$Pre\ A\ (z_2 \star x_{im} \star x_{om})) \Rightarrow Pre\ C\ (z_1 \star x_{ic} \star x_{oc})$

$$\Leftrightarrow \quad [\text{Let } y_c = z_1 \star x_{ic} \star x_{oc} \text{ and } v_a = z_2 \star x_{im} \star x_{om}]$$
$$\forall\, y_c \bullet (\forall\, v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow \mathit{Pre}\ A\ v_a) \Rightarrow \mathit{Pre}\ C\ y_c$$

Finally, for the backwards simulation *correctness* we have:

Proposition 6.6.14 For arbitrary charts A and C we have,

$$\mathrm{dom}(T \rhd \mathrm{dom}\, \mathit{AStep}) \lhd \mathit{CStep} \fatsemi T \subseteq T \fatsemi \mathit{AStep}$$
$$\Leftrightarrow \quad [\text{Detailed derivation below}]$$
$$\forall\, y_c, \vdash_{z_c}, y_a \bullet ((\forall\, v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow \mathit{Pre}\ A\ v_a) \wedge$$
$$y_c \star \vdash_{z_c}{}' \dot{\in} C \wedge \vdash_{z_c} \star\, y_a' \in R^{-1}) \Rightarrow \exists\, t_2 \bullet y_c \star t_2' \in R^{-1} \wedge t_2 \star y_a' \dot{\in} A$$

Proof 6.6.14

$$\mathrm{dom}(T \rhd \mathrm{dom}\, \mathit{AStep}) \lhd \mathit{CStep} \fatsemi T \subseteq T \fatsemi \mathit{AStep}$$
$$\Leftrightarrow \quad [\subseteq\text{-df}]$$
$$\forall\, x_{ic}, si, so, z_1, ssi, x_{oa}, soo, z_2 \bullet$$
$$(x_{ic}{}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in \mathrm{dom}(T \rhd \mathrm{dom}\, \mathit{AStep}) \lhd \mathit{CStep} \fatsemi T \Rightarrow$$
$$(x_{ic}{}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T \fatsemi \mathit{AStep}$$
$$\Leftrightarrow \quad [\fatsemi\text{-df}]$$
$$\forall\, x_{ic}, si, so, z_1, ssi, x_{oa}, soo, z_2 \bullet (\exists\, ti, x_{oe}, to, z_3 \bullet$$
$$(x_{ic}{}^\frown si, so, z_1) \mapsto (ti, to^\frown x_{oe}., z_3') \in \mathrm{dom}(T \rhd \mathrm{dom}\, \mathit{AStep}) \lhd \mathit{CStep} \wedge$$
$$(ti, to^\frown x_{oe}., z_3) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T) \Rightarrow$$
$$(x_{ic}{}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T \fatsemi \mathit{AStep}$$
$$\Leftrightarrow \quad [\text{Predicate calculus, } \lhd\text{-df}]$$
$$\forall\, x_{ic}, si, so, z_1, ssi, x_{oa}, soo, z_2, ti, x_{oe}, to, z_3 \bullet$$
$$((x_{ic}{}^\frown si, so, z_1) \mapsto (ti, to^\frown x_{oe}., z_3) \in \mathit{CStep} \wedge$$
$$(x_{ic}{}^\frown si, so, z_1) \notin \mathrm{dom}(T \rhd \mathrm{dom}\, \mathit{AStep}) \wedge$$
$$(ti, to^\frown x_{oe}., z_3) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T) \Rightarrow$$
$$(x_{ic}{}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T \fatsemi \mathit{AStep}$$
$$\Leftrightarrow \quad [\text{Lemma 6.6.12, Predicate calculus}]$$

$$\forall x_{ic}, si, so, z_1, ssi, x_{oa}, soo, z_2, ti, x_{oe}, to, z_3, x_{oc} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$(x_{ic}^\frown si, so, z_1) \mapsto (ti, to^\frown x_{oe}., z_3) \in CStep \wedge$$

$$(ti, to^\frown x_{oe}., z_3) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T) \Rightarrow$$

$$(x_{ic}^\frown si, so, z_1) \mapsto (ssi, sso^\frown x_{oa}., z_2) \in T \fatsemi AStep$$

$\Leftrightarrow$ [definitions of *CStep*, *AStep* and T]

$$\forall x_{ic}, si, so, z_1, x_{oa}, z_2, x_{oe}, z_3, x_{oc} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$(x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oe}., z_3) \in CStep \wedge$$

$$(si_{\triangleright(in_C)}, so_{\triangleright(out_C)}^\frown x_{oe}., z_3) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oa}., z_2) \in T) \Rightarrow$$

$$(x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oa}., z_2) \in T \fatsemi AStep$$

$\Leftrightarrow$ [Definition of *CStep*, Proposition 6.6.9]

$$\forall x_{ic}, si, so, z_1, x_{oa}, z_2, x_{oe}, z_3, x_{oc} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$z_1 \star x_{ic} \star z_3' \star x_{oe}' \in C \wedge \forall x_{ia} \bullet \exists x_{ie} \bullet z_3 \star x_{ie} \star x_{oe} \star z_2' \star x_{ia}' \star x_{oa}' \in R^{-1}) \Rightarrow$$

$$(x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oa}., z_2) \in T \fatsemi AStep$$

$\Leftrightarrow$ [Predicate calculus]

$$\forall x_{ic}, si, so, z_1, x_{oa}, z_2, x_{oe}, z_3, x_{oc}, x_{ie}, x_{ia} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$z_1 \star x_{ic} \star z_3' \star x_{oe}' \in C \wedge z_3 \star x_{ie} \star x_{oe} \star z_2' \star x_{ia}' \star x_{oa}' \in R^{-1}) \Rightarrow$$

$$(x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oa}., z_2) \in T \fatsemi AStep$$

$\Leftrightarrow$ [$\fatsemi$-df, definition of T]

$$\forall x_{ic}, si, so, z_1, x_{oa}, z_2, x_{oe}, z_3, x_{oc}, x_{ie}, x_{ia} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$z_1 \star x_{ic} \star z_3' \star x_{oe}' \in C \wedge z_3 \star x_{ie} \star x_{oe} \star z_2' \star x_{ia}' \star x_{oa}' \in R^{-1}) \Rightarrow \exists x_{in}, y \bullet$$

$$(x_{ic}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, z_1) \mapsto (x_{in}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, y) \in T \wedge$$

$$(x_{in}^\frown si_{\triangleright(in_C)}, so_{\triangleright(out_C)}, y) \mapsto (si_{\triangleright(in_A)}, so_{\triangleright(out_A)}^\frown x_{oa}., z_2) \in AStep$$

$\Leftrightarrow$ [Proposition 6.6.8, definition of *AStep*]

$$\forall x_{ic}, z_1, x_{oa}, z_2, x_{oe}, z_3, x_{oc}, x_{ie}, x_{ia} \bullet$$

$$((\forall x_{im}, z_4, x_{om} \bullet z_1 \star x_{ic} \star x_{oc} \star z_4' \star x_{im}' \star x_{om}' \in R^{-1} \Rightarrow Pre\ A\ (z_4 \star x_{im} \star x_{om})) \wedge$$

$$z_1 \star x_{ic} \star z_3' \star x_{oe}' \in C \wedge z_3 \star x_{ie} \star x_{oe} \star z_2' \star x_{ia}' \star x_{oa}' \in R^{-1}) \Rightarrow$$

$$\exists x_{in}, y \bullet (\exists x_{on} \bullet z_1 \star x_{ic} \star x_{oc} \star y' \star x_{in}' \star x_{on}' \in R^{-1}) \wedge y \star x_{ie} \star z_2' \star x_{oa}' \in A$$

$\Leftrightarrow$ [Let $y_c = z_1 \star x_{ic} \star x_{oc}$, $v_a = z_4 \star x_{im} \star x_{om}$, $\vdash_{z_C} = z_3 \star x_{ie} \star x_{oe}$, $y_a = z_2 \star x_{ia} \star x_{oa}$ and $t_2 = y \star x_{in} \star x_{on}$, $\dot\in$-df]

$$\forall y_c, \vdash_{z_C}, y_a \bullet ((\forall v_a \bullet y_c \star v_a' \in R^{-1} \Rightarrow Pre\ A\ v_a) \wedge$$

$$y_c \star \vdash_{z_C}{}' \dot\in C \wedge \vdash_{z_C} \star\, y_a' \in R^{-1}) \Rightarrow \exists t_2 \bullet y_c \star t_2' \in R^{-1} \wedge t_2 \star y_a' \dot\in A$$

B.14 Proofs for Section 7.1: Monotonicity of the μ-Charts composition operator

Before proving the monotonicity result of Proposition 7.1.1 we prove the following elimination rules that are specific to the simulation relations used in the proof of the monotonicity property.

Lemma B.14.1 Given arbitrary charts A_1, C_2, B, $C = C_2 \mid \Psi \mid B$ and $A = A_1 \mid \Psi \mid B$; related simulations $S =_{def} Corr_{C_2}^{A_1} \wedge Corr_B^B \wedge IO_C^A$ and $R =_{def} Corr_{C_2}^{A_1} \wedge IO_{C_2}^{A_1}$; arbitrary binding $z_a^{U_1}$, $z_b^{U_2}$, $x_a^{V_a^{io}} \vdash_{\mathcal{Z}_C}{}^{U_2}$, $x_c^{V_c^{io}}$, $x_2^{V_2^{io}}$ and $v_c^{V_c^{io}}$, we have,

$$\frac{\begin{array}{ll} z_a \star z_b \star x_a \star \vdash_{\mathcal{Z}_C}{}' \star z_b' \star x_c' \in S & a_1.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1}, \\ x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2} & a_1.o_{A_1} = x_2.o_{C_2} \cap out_{A_1}, \\ & z_a \star a_1 \star \vdash_{\mathcal{Z}_C}{}' \star x_2' \in R \vdash P \end{array}}{P} \; (S^-_{|-|_1})$$

where $[\![A]\!]^{\mathbb{P}\,T_a}_{\mathcal{Z}_C}$, $[\![C]\!]^{\mathbb{P}\,T_c}_{\mathcal{Z}_C}$, $[\![A_1]\!]^{\mathbb{P}\,T_1}_{\mathcal{Z}_C}$, $[\![C_2]\!]^{\mathbb{P}\,T_2}_{\mathcal{Z}_C}$, $[\![B]\!]^{\mathbb{P}\,T_3}_{\mathcal{Z}_C}$, and the usual conditions hold for $a_1^{V_1^{io}}$ and P.

Similarly the obvious counterpart to this rule holds for arbitrary $x_1^{V_1^{io}}$, assuming the usual conditions for $b_2^{V_2^{io}}$.

$$\frac{\begin{array}{ll} z_a \star z_b \star x_a \star \vdash_{\mathcal{Z}_C}{}' \star z_b' \star x_c' \in S & b_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}, \\ x_1.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1} & b_2.o_{C_2} = x_1.o_{C_1} \cap out_{C_2}, \\ & z_a \star x_1 \star \vdash_{\mathcal{Z}_C}{}' \star b_2' \in R \vdash P \end{array}}{P} \; (S^-_{|-|_2})$$

Proof B.14.1

For $(S^-_{|-|_1})$, letting $t^{V_1^{io}} =_{def} \{\!|\, i_{A_1} \Rrightarrow (x_a.i_A \cup fb_{v_c}) \cap in_{A_1}, o_{A_1} \Rrightarrow x_2.out_{A_1} \,|\!\}$ we have,

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\zeta_1 \\ \vdots \\ t \star x_2' \in IO_{C_2}^{A_1}\end{array} \quad \overline{t.o_{A_1} = x_2.o_{C_2} \cap out_{A_1}}}{t.o_{A_1} = x_2.o_{C_2} \cap out_{A_1} \wedge t \star x_2' \in IO_{C_2}^{A_1}} \quad \dfrac{}{\begin{array}{l} t.i_{A_1} = \\ (x_a.i_A \cup fb_{v_c}) \cap in_{A_1} \end{array}} \; (df)}{t.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1} \wedge t.o_{A_1} = x_2.o_{C_2} \cap out_{A_1} \wedge t \star x_2' \in IO_{C_2}^{A_1}} \quad \begin{array}{c}\zeta_2 \\ \vdots \\ P\end{array}}{\begin{array}{c} \exists\, x_1^{V_1^{io}} \bullet x_1.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1} \wedge \\ x_1.o_{A_1} = x_2.o_{C_2} \cap out_{A_1} \wedge x_1 \star x_2' \in IO_{C_2}^{A_1} \end{array}} \; (\exists^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{}{t.o_{A_1} = x_2.o_{C_2} \cap out_{A_1}}\,(df) \quad \dfrac{\dfrac{}{x_2.o_{C_2} \subseteq out_{C_2}}\,(V_2^{io}\text{-}df)}{x_2.o_{C_2} \cap out_{A_1} \cap out_{C_2} = x_2.o_{C_2} \cap out_{A_1}}}{t.o_{A_1} \cap out_{C_2} = x_2.o_{C_2} \cap out_{A_1}} \quad \begin{array}{c}\zeta_{1.1}\\ \vdots\\ (df)\end{array}}{t \star x_2' \in IO_{C_2}^{A_1}} \quad t.i_{A_1} \cap in_{C_2} = x_2.i_{C_2} \cap in_A$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{t.i_{A_1} = x_a.i_A \cup fb_{v_c}) \cap in_{A_1}}\,(df) \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z_a \star z_b \star x_a \star \vdash_{\mathcal{Z}_C}{}' \star z_b' \star x_c' \in S}{x_a.i_A \cap in_C = x_c.i_C \cap in_A}\,(S_{io1}^-)}{x_a.i_A \cap (in_{C_2} \cup in_B) = x_c.i_C \cap (in_{A_1} \cup in_B)}\,(df)}{x_a.i_A \cap (in_{C_2} \cup in_B) \cap in_{C_2} \cap in_{A_1} = x_c.i_C \cap \cap (in_{A_1} \cup in_B) \cap in_{C_2} \cap in_{A_1}}\,(df)}{x_a.i_A \cap in_{C_2} \cap in_{A_1} = x_c.i_C \cap in_{C_2} \cap in_{A_1}}}{x_a.i_A \cap in_{A_1} \cap in_{C_2} = x_c.i_C \cap in_{C_2} \cap in_{A_1}}}{(x_a.i_A \cup fb_{v_c}) \cap in_{A_1} \cap in_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2} \cap in_{A_1}} \quad x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}}{t.i_{A_1} \cap in_{C_2} = x_2.i_{C_2} \cap in_A}\,(df)$$

ζ_2 is:

$$\dfrac{\dfrac{}{a_1.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1}}\,1 \quad \dfrac{}{a_1.o_{A_1} = x_2.o_{C_2} \cap out_{A_1}}\,1 \quad \begin{array}{c}\zeta_3\\ \vdots\\ z_a \star a_1 \star \vdash_{\mathcal{Z}_C}{}' \star x_2' \in R\end{array}}{P}$$

ζ_3 is:

$$\dfrac{\dfrac{}{a_1 \star x_2' \in IO_{C_2}^{A_1}}\,1 \quad \dfrac{z_a \star z_b \star x_a \star \vdash_{\mathcal{Z}_C}{}' \star z_b' \star x_c' \in S}{z_a \star \vdash_{\mathcal{Z}_C}{}' \in Corr_{C_2}^{A_1}}\,(S_{io4}^-)}{z_a \star a_1 \star \vdash_{\mathcal{Z}_C}{}' \star x_2' \in R}\,(S_{io}^+)$$

The proof of $(S^-_{|-|_2})$ has the same form and can be derived in the obvious way from the proof for $(S^-_{|-|_1})$ above.

Lemma B.14.2 Given arbitrary charts A and C, simulation relation $R =_{def} Corr_C^A \wedge IO_C^A$, bindings $z_a^{U_a}$, $x_1^{V_a^{io}}$, $\vdash_{\mathcal{Z}_C}{}^{U_c}$, $x_2^{V_c^{io}}$, and signal set Ψ, for $A_\Psi =_{def} [A]_\Psi$, $C_\Psi =_{def} [C]_\Psi$ and $T =_{def} Corr_C^A \wedge IO_{C_\Psi}^{A_\Psi}$ we have,

$$\dfrac{\begin{array}{l} z_a \star x_1 \star \vdash_{\mathcal{Z}_C}{}' \star x_2' \in R \quad a_\psi.i_{A_\Psi} = x_1.i_A,\\ \qquad\qquad\qquad\quad f_\phi.i_{C_\Psi} = x_2.i_C,\\ \qquad\qquad\qquad\quad f_\phi.o_{C_\Psi} = a_\psi.o_{A_\Psi},\\ \qquad\qquad\qquad\quad z_a \star a_\psi \star \vdash_{\mathcal{Z}_C}{}' \star f_\phi' \in T \vdash P \end{array}}{P}\,(S_{||}^-)$$

where $[\![A]\!]^{T_a}_{z_C}$, $[\![C]\!]^{T_c}_{z_C}$, $[\![A_\Psi]\!]^{\mathbb{P}\,T_\psi}_{z_C}$, $[\![C_\Psi]\!]^{\mathbb{P}\,T_\phi}_{z_C}$, and the usual conditions hold for $a_\psi^{V_\psi^{io}}$, $f_\phi^{V_\phi^{io}}$ and P.

Proof B.14.2

$$\dfrac{\dfrac{}{\exists\, a_\psi^{V_\psi^{io}} \bullet \; a_\psi.i_{A_\Psi} = x_1.i_A \cap in_{A_\Psi} \wedge \; a_\psi.o_{A_\Psi} = x_1.o_A \cap out_{A_\Psi}}\,(V_\psi^{io}\text{-}df) \qquad \dfrac{\zeta_1 \quad \dfrac{}{\exists f_\phi^{V_\phi^{io}} \bullet \; f_\phi.i_{C_\Psi} = x_2.i_C \cap in_{C_\Psi} \wedge \; f_\phi.o_{C_\Psi} = x_2.o_A \cap out_{C_\Psi}}\,(V_\phi^{io}\text{-}df)}{P}\,(\exists^-)(2)}{P}\,(\exists^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{z_a \star x_1 \star \vdash_{z_C}{}' \star\, x_2' \in R}{x_1.i_A \cap in_{C_\Psi} = x_2.i_C \cap in_{A_\Psi}}\,(S^-_{io1}) \quad \zeta_2}{a_\psi.i_{A_\Psi} \cap in_{C_\Psi} = x_2.i_C \cap in_{A_\Psi}} \quad \zeta_3}{a_\psi.i_{A_\Psi} \cap in_{C_\Psi} = f_\phi \cap in_{A_\Psi}} \quad \zeta_4}{a_\psi \star f_\phi' \in IO^{A_\Psi}_{C_\Psi}}\,(IO\text{-}df) \qquad \dfrac{z_a \star x_1 \star \vdash_{z_C}{}' \star\, x_2' \in R}{z_a \star \vdash_{z_C}{}' \in Corr^A_C}\,(S^-_{io4})}{z_a \star a_\psi \star \vdash_{z_C}{}' \star f_\phi' \in T}\,(S^+_{io})$$

$$\dfrac{\zeta_2 \quad \zeta_3 \quad \zeta_5 \quad \vdots}{P}$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{}{a_\psi.i_{A_\Psi} = x_1.i_A \cap in_{A_\Psi}}\,1 \qquad \dfrac{}{in_{A_\Psi} = in_A}\,(df)}{a_\psi.i_{A_\Psi} = x_1.i_A \cap in_A} \qquad \dfrac{}{x_1.i_A \subseteq in_A}\,(V_a^{io}\text{-}df)}{a_\psi.i_{A_\Psi} = x_1.i_A}$$

ζ_3 is:

$$\dfrac{\dfrac{\dfrac{}{f_\phi.i_{C_\Psi} = x_2.i_C \cap in_{C_\Psi}}\,1 \qquad \dfrac{}{in_{C_\Psi} = in_C}\,(df)}{f_\phi.i_{C_\Psi} = x_2.i_C \cap in_C} \qquad \dfrac{}{x_2.i_C \subseteq in_C}\,(V_c^{io}\text{-}df)}{f_\phi.i_{C_\Psi} = x_2.i_C}$$

ζ_4 is:

$$\dfrac{\dfrac{\dfrac{\zeta_5}{f_\phi.o_{C_\Psi} = a_\psi.o_{A_\Psi}}}{f_\phi.o_{C_\Psi} \cap out_{A_\Psi} = a_\psi.o_{A_\Psi} \cap out_{A_\Psi}} \qquad \dfrac{}{out_{A_\Psi} = out_{C_\Psi}}\,(df)}{f_\phi.o_{C_\Psi} \cap out_{A_\Psi} = a_\psi.o_{A_\Psi} \cap out_{C_\Psi}}$$

ζ_5 is:

$$\dfrac{\dfrac{\dfrac{}{a_\psi.o_{A_\Psi} = x_1.o_A \cap out_{A_\Psi}}\,1 \qquad \dfrac{}{f_\phi.o_{C_\Psi} = x_1.o_A \cap out_{C_\Psi}}\,2 \qquad \dfrac{}{out_{A_\Psi} = out_{C_\Psi}}\,(df)}{f_\phi.o_{C_\Psi} = a_\psi.o_{A_\Psi}}$$

The property of Lemma B.14.3 follows directly from the side-condition SC_1 that is part of the monotonicity result of Proposition 7.1.1. This gives us a rule that can be used more concisely in the proof of Proposition 7.1.1 that follows.

Lemma B.14.3 For arbitrary charts A_1, C_2, bindings $z_1^{U_1}$, $x_1^{V_1^{io}}$, $y_1^{U_1}$, $v_1^{V_1^{io}}$, $z_2^{U_2}$, $x_2^{V_2^{io}}$, and signal set Ψ, we have,

$$\dfrac{\dfrac{}{[A_1]_\Psi \sqsupseteq^T_{\tau f} [C_2]_\Psi}\ SC_1 \qquad \begin{array}{l} Pre\ C_2\ (z_2 \star x_2) \\ z_1 \star x_1 \star y_1' \star v_1' \mathrel{\dot\in} A_1 \\ z_1 \star x_1 \star z_2' \star x_2' \in R \end{array} \qquad \begin{array}{l} z_2 \star x_2 \star b_2' \star e_2' \mathrel{\dot\in} C_2, \\ v_1.o_{A_1} \cap \Psi = e_2.o_{C_2} \cap \Psi \vdash P \end{array}}{P}\ (SC_{1\,I}^-)$$

where $T =_{def} Corr^{C_2}_{A_1} \wedge IO^{C_\Psi}_{A_\Psi}$, $A_\Psi = [A_1]_\Psi$, $C_\Psi = [C_2]_\Psi$, $[\![A_\Psi]\!]^{T_\psi}_{z_C}$, $[\![C_\Psi]\!]^{T_\phi}_{z_C}$, $[\![A_1]\!]^{T_1}_{z_C}$, $[\![C_2]\!]^{T_2}_{z_C}$, and the usual conditions hold for $b_2^{U_2}$, $e_2^{V_2^{io}}$ and P

Proof B.14.3

Letting $s^{V_\phi^{io}} =_{def} \{\!|\ o_{C_\Psi} \Rrightarrow v.o_{C_2} \cap out_{C_\Psi}\ |\!\}$ and
$t^{V_\psi^{io}} =_{def} \{\!|\ o_{A_\Psi} \Rrightarrow v_1.o_{A_1} \cap out_{A_\Psi}\ |\!\}$,
we have,

$$\dfrac{z_1 \star x_1 \star z_2' \star x_2' \in R \qquad \begin{array}{c}\zeta_1 \\ \vdots \\ P\end{array}}{P}\ (S_{||}^-)(1)$$

where ζ_1 is:

$$\dfrac{\dfrac{}{A_\Psi \sqsupseteq^T_{\tau f} C_\Psi}\ SC_1 \qquad \begin{array}{c}\zeta_{1.1} \\ \vdots \\ z_1 \star a_\psi \star y_1' \star t' \mathrel{\dot\in} A_\Psi\end{array} \qquad \begin{array}{c}\zeta_{1.2} \\ \vdots \\ Pre\ C_\Psi\ (z_2 \star e_\psi)\end{array} \qquad \begin{array}{c}\zeta_{1.3} \\ \vdots \\ z_2 \star e_\psi \star z_1' \star a_\psi' \in T\end{array} \qquad \begin{array}{c}\zeta_2 \\ \vdots \\ P\end{array}}{P}\ (\sqsupseteq^-_{\tau f\ IV})(2)$$

$\zeta_{1.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a_\psi.i_{A_\Psi} = x_1.i_{A_1}}\ 1 \qquad \dfrac{}{a_\psi.i_{A_\Psi} \subseteq in_{A_\Psi}}\ (V^{io}_{A_\Psi}\text{-}df)}{a_\psi.i_{A_\Psi} \cap in_{A_\Psi} = x_1.i_{A_1}} \qquad \dfrac{}{in_{A_\Psi} = in_{A_1}}\ (df)}{x_1.i_{A_1} = a_\psi.i_{A_\Psi} \cap in_{A_1}} \qquad \dfrac{}{t.o_{A_\Psi} = v_1.o_{A_1} \cap out_{A_\Psi}}\ (df)}{z_1 \star x_1 \star y_1' \star v_1' \mathrel{\dot\in} A_1 \qquad z_1 \star a_\psi \star y_1' \star t' \mathrel{\dot\in} A_\Psi}\ (Z^+_{||})$$

$\zeta_{1.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{e_\psi.i_{C_\Psi} = x_2.i_{C_2}}\,1 \quad \dfrac{}{e_\psi, i_{C_\Psi} \subseteq in_{C_\Psi}}\,(df)}{e_\psi.i_{C_\Psi} \cap in_{C_\Psi} = x_2.i_{C_2}} \quad \dfrac{}{in_{C_\Psi} = in_{C_2}}\,(df)}{x_2.i_{C_2} = e_\psi.i_{C_\Psi} \cap in_{C_2}} \quad \dfrac{}{z_2 \star x_2 \star y' \star v' \dot{\in} C_2}\,3}{\dfrac{\dfrac{}{s.o_{C_\Psi} = v.o_{C_2} \cap out_{C_\Psi}}\,(df) \quad \dfrac{\dfrac{}{z_2 \star e_\psi \star y' \star s' \dot{\in} C_\Psi}}{Pre\ C_\Psi\ (z_2 \star e_\psi)}\,(Pre^+)}{Pre\ C_2\ (z_2 \star x_2)}\,(Pre^-)(3)}\,(Z_{||}^+)$$

where $\zeta_{1.3}$ is:

$$\dfrac{\dfrac{\dfrac{}{z_1 \star a_\psi \star z_2' \star e_\psi' \in Corr_{C_2}^{A_1} \wedge IO_{C_\Psi}^{A_\Psi}}\,1}{z_1 \star z_2' \in Corr_{C_2}^{A_1}}\,(\wedge^-) \quad \dfrac{\dfrac{}{z_1 \star a_\psi \star z_2' \star e_\psi' \in Corr_{C_2}^{A_1} \wedge IO_{C_\Psi}^{A_\Psi}}\,1}{a_\psi \star e_\psi' \in IO_{C_\Psi}^{A_\Psi}}\,(\wedge^-)}{z_2 \star z_1' \in Corr_{A_1}^{C_2}}\,(df) \quad \dfrac{}{e_\psi \star a_\psi' \in IO_{A_\Psi}^{C_\Psi}}\,(df)}{z_2 \star e_\psi \star z_1' \star a_\psi' \in T}\,(S_{io}^+)$$

ζ_2 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,\zeta_3}{c_\psi.o_{C_\Psi} = v_1.o_{A_1} \cap \Psi} \quad \dfrac{\dfrac{}{out_{C_\Psi} = \Psi}\,(df) \quad \dfrac{}{c_\psi.o_{C_\Psi} = v_2.o_{C_2} \cap out_{C_\Psi}}\,4}{c_\psi.o_{C_\Psi} = v_2.o_{C_2} \cap \Psi}}{v_1.o_{A_1} \cap \Psi = v_2.o_{C_2} \cap \Psi} \quad \dfrac{}{z_2 \star x_2 \star b_2' \star v_2' \dot{\in} C_2}\,4}{\dfrac{}{z_2 \star e_\psi \star b_2' \star c_\psi' \dot{\in} C_\Psi}\,2 \quad \dfrac{\vdots}{P}\,(ass)}{P}\,(Z_{||}^-)(4)$$

ζ_3 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{out_{C_\Psi} = out_{A_\Psi}}\,(df) \quad \dfrac{\dfrac{}{b_2 \star c_\psi \star y_1' \star t' \in T}\,2}{c_\psi.o_{C_\Psi} \cap out_{A_\Psi} = t.o_{A_\Psi} \cap out_{C_\Psi}}\,(S_{io2}^-)}{c_\psi.o_{C_\Psi} \cap out_{C_\Psi} = t.o_{A_\Psi} \cap out_{A_\Psi}} \quad \dfrac{}{c_\psi.o_{C_\Psi} \subseteq out_{C_\Psi}}\,(V_\Psi^{io}\text{-}df)}{c_\psi.o_{C_\Psi} = t.o_{A_\Psi} \cap out_{A_\Psi}}\,(df) \quad \dfrac{}{t.o_{A_\Psi} = v_1.o_{A_1} \cap out_{A_\Psi}}}{\dfrac{}{out_{A_\Psi} = \Psi}\,(df) \quad \dfrac{}{c_\psi.o_{C_\Psi} = v_1.o_{A_1} \cap out_{A_\Psi}}}{c_\psi.o_{C_\Psi} = v_1.o_{A_1} \cap \Psi}$$

Lemma B.14.4 For arbitrary charts A, C and signal set Ψ we have,

$$\dfrac{[A]_\Psi \sqsupseteq_{\tau f}^T [C]_\Psi \quad y_a \star y_c' \in R \quad y_c \star \vdash_{Z_C}{}' \dot{\in} C}{Pre\ A\ y_a}\,(SC_{1\,II}^-)$$

where $R =_{def} Corr_C^A \wedge IO_C^A$ and $T =_{def} Corr_A^C \wedge IO_{A_\Psi}^{C_\Psi}$ for $C_\Psi = [C]_\Psi$ and $A_\Psi = [A]_\Psi$.

Proof B.14.4

Assuming $S = Corr_C^A \wedge IO_{C_\Psi}^{A_\Psi}$ we have,

$$\dfrac{\dfrac{A_\Psi \sqsupseteq^t_{\tau f} C_\Psi \quad \dfrac{\dfrac{}{a_\phi \star f'_\phi \in S}\,1 \quad \dfrac{}{f_\phi \star a'_\phi \in T}\,(df)}{Pre\ C_\Psi\ f_\phi}\,(\sqsupseteq^-_{\tau f\ III}) \quad \dfrac{\dfrac{y_c \star \vdash_{z_c}{}' \dot{\in} C}{Pre\ C\ y_c}\,(Pre^+) \quad \dfrac{}{f_\phi.i_{C_\Psi} = y_c.i_C}\,1}{Pre\ A_\Psi\ a_\psi}\,(Pre^+_{[]}) \quad \dfrac{}{y_a.i_A = a_\psi.i_{A_\Psi}}\,1}{Pre\ A\ y_a}\,(Pre^-_{[]}) \quad y_a \star y'_c \in R}{Pre\ A\ y_a}\,(S^-_{[]})(1)$$

Proposition 7.1.1 If, for arbitrary charts A_1, C_2, and signal set Ψ, we have that,

$$\dfrac{}{[A_1]_\Psi \sqsupseteq^T_{\tau f} [C_2]_\Psi}\,SC_1$$

$$\dfrac{}{out_{A_1} \cap \Psi = out_{C_2} \cap \Psi}\,SC_2$$

$$\dfrac{}{out_{A_1} \cap out_B = out_{C_2} \cap out_B}\,SC_3$$

where $T =_{def} Corr_{A_1}^{C_2} \wedge IO_{A_\Psi}^{C_\Psi}$ for $C_\Psi = [C_2]_\Psi$ and $A_\Psi = [A_1]_\Psi$, then for arbitrary chart B, we have the monotonicity result,

$$\dfrac{C_2 \sqsupseteq^R_{\tau f} A_1 \quad SC_1 \quad SC_2 \quad SC_3}{(C_2 \mid \Psi \mid B) \sqsupseteq^S_{\tau f} (A_1 \mid \Psi \mid B)}$$

where $S =_{def} Corr_{C_2}^{A_1} \wedge Corr_B^B \wedge IO_C^A$, and $R =_{def} Corr_{C_2}^{A_1} \wedge IO_{C_2}^{A_1}$.

Proof 7.1.1

To prove Proposition 7.1.1 we give separate proofs for each of the following five properties. Assuming $C = C_2 \mid \Psi \mid B$ and $A = A_1 \mid \Psi \mid B$, for arbitrary bindings $y_a^{T^\omega_A}$, $y_c^{T^\omega_C}$, and $z_c^{T^\omega_C}$, we show that, .

$$\dfrac{C_2 \sqsupseteq^R_{\tau f} A_1 \quad y_c \dot{\in} Init_C}{\exists\, t_1 \bullet t_1 \dot{\in} Init_A \wedge t_1 \star y'_c \in S}\,init \qquad \dfrac{C_2 \sqsupseteq^R_{\tau f} A_1}{out_A \subseteq out_C}\,fin$$

$$\dfrac{SC_1 \quad Pre\ A\ y_a \quad y_a \star y'_c \in S}{Pre\ C\ y_c}\,app$$

$$\dfrac{C_2 \sqsupseteq^R_{\tau f} A_1 \quad SC_2 \quad SC_3 \quad Pre\ A\ y_a \quad y_a \star y'_c \in S \quad y_c \star \vdash_{z_c}{}' \dot{\in} C}{\exists\, t_2 \bullet y_a \star t'_2 \dot{\in} A \wedge t_2 \star \vdash_{z_c}{}' \in S}\,corr$$

Given these four properties hold, Proposition 7.1.1 follows trivially using the rule $(\sqsupseteq^+_{\tau f})$.

$$\frac{\begin{array}{cccc} init & fin & app & corr \\ \vdots & \vdots & \vdots & \vdots \end{array}}{C \sqsupseteq^{S}_{\tau f} A}\ (\sqsupseteq^{+}_{\tau f})$$

Each of the local assumptions, that is, all except $C_2 \sqsupseteq^{R}_{\tau f} A_1$, SC_1, SC_2 and SC_3, are discharged by the invocation of the rule $(\sqsupseteq^{+}_{\tau f})$.

Now for the property *init*, given arbitrary bindings $z_b^{U_3}, \vdash_{Z_C}{}^{U_2}$ and $x_c^{T^{io}_C}$, we have,

$$\frac{C_2 \sqsupseteq^{R}_{\tau f} A_1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{z_b \star \vdash_{Z_C} \star\, x_c \dot{\in} Init_C}{z_b \star \vdash_{Z_C} \in Init_C}\,(\dot{\in}\text{-}df)}{z_b \star \vdash_{Z_C} \in Init_B \wedge Init_{C_2}}\,(Init_C\text{-}df)}{\vdash_{Z_C} \in Init_{C_2}}\,(S^{-}_{\wedge_0})}{\vdash_{Z_C} \star\, x_3 \dot{\in} Init_{C_2}}\,(\dot{\in}\text{-}df) \qquad \begin{array}{c}\zeta_1 \\ \vdots \\ \exists\, t \bullet t \dot{\in} Init_A \wedge t \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S\end{array}}{\exists\, t \bullet t \dot{\in} Init_A \wedge t \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}\ (\sqsupseteq^{-}_{\tau f\ II})(1)$$

where ζ_1 is:

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{z_b \star \vdash_{Z_C} \star\, x_c \dot{\in} Init_C}{z_b \star \vdash_{Z_C} \in Init_C}\,(\dot{\in}\text{-}df)}{z_b \star \vdash_{Z_C} \in Init_B \wedge Init_{C_2}}\,(Init_C\text{-}df)}{z_b \in Init_B}\,(S^{-}_{\wedge_0}) \qquad \dfrac{\dfrac{}{t_1 \star x_1 \dot{\in} Init_{A_1}}\,1}{t_1 \in Init_{A_1}}\,(\dot{\in}\text{-}df)}{z_b \star t_1 \in Init_B \wedge Init_{A_1}}\,(S^{+}_{\wedge}) \qquad \begin{array}{c}\zeta_2 \\ \vdots \\ z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S\end{array}}{\dfrac{\dfrac{z_b \star t_1 \star x_a \dot{\in} Init_A \wedge z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}{\exists\, t \bullet t \dot{\in} Init_A \wedge t \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}\,(\exists^{+})}{z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}\,(\wedge^{+})}$$

$$\frac{\dfrac{}{\exists\, x_a^{V^{io}_A} \bullet x_a \star x_c \in IO^A_C}\,(df)}{\exists\, t \bullet t \dot{\in} Init_A \wedge t \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}\,(\exists^{-})(2)$$

ζ_2 is:

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{}{t_1 \star x_1 \star \vdash_{Z_C}{}' \star x'_3 \in R}\,1}{t_1 \star x_1 \star \vdash_{Z_C}{}' \star x'_3 \in Corr^{A_1}_{C_2} \wedge IO^{A_1}_{C_2}}\,(R\text{-}df)}{t_1 \star \vdash_{Z_C}{}' \in Corr^{A_1}_{C_2}}\,(S^{-}_{\wedge}) \qquad \dfrac{}{x_a \star x'_c \in IO^A_C}\,2}{t_1 \star x_a \star \vdash_{Z_C}{}' \star x'_c \in Corr^{A_1}_{C_2} \wedge IO^A_C}\,(S^{+}_{\wedge}) \qquad \dfrac{}{z_b \star z'_b \in Corr^B_B}\,(df)}{z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in Corr^B_B \wedge Corr^{A_1}_{C_2} \wedge IO^{A_1}_{C_2}}\,(S^{+}_{\wedge})$$

$$\frac{z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in Corr^B_B \wedge Corr^{A_1}_{C_2} \wedge IO^{A_1}_{C_2}}{z_b \star t_1 \star x_a \star z'_b \star \vdash_{Z_C}{}' \star x'_c \in S}\,(S\text{-}df)$$

For the property *fin* we have,

$$\frac{\dfrac{\dfrac{C_2 \sqsupseteq^{R}_{\tau f} A_1}{out_{A_1} \subseteq out_{C_2}}\,(\sqsupseteq^{-}_{\tau f\ I})}{out_{A_1} \cup out_B \subseteq out_{C_2} \cup out_B}\,(\subseteq\text{-}df)}{out_A \subseteq out_C}\,(out_A\text{-}df), (out_C\text{-}df)$$

For the property *app*, where $fb_{v_a} = v_a.o_A \cap \Psi$ and $fbx_c = x_c.o_C \cap \Psi$ and $[\![A_1]\!]^{T_1}_{Z_C}$,

$[\![C_2]\!]^{T_2}_{z_C}$ and $[\![B]\!]^{T_3}_{z_C}$, we have,

$$\dfrac{\dfrac{}{\exists\, v_c^{V_c^{io}} \bullet v_c.o_C = v_2.o_{C_2} \cup v_3.o_B}\,(V_c^{io}\text{-}df) \qquad \dfrac{\zeta_1 \quad \dfrac{}{\exists\, x_2^{V_2^{io}} \bullet x_2.i_{C_2} = (x_c.i_C \cup fb_{v_a}) \cap in_{C_2} \wedge x_2.o_{C_2} = x_1.o_{A_1} \cap out_{C_2}}\,(V_2^{io}\text{-}df)}{Pre\ C\ (\vdash_{z_C} \star z_b \star x_c)}\,(\exists^-)(2)}{Pre\ C\ (\vdash_{z_C} \star z_b \star x_c)}\,(\exists^-)(1)$$

where ζ_1 is:

$$\dfrac{SC_1 \qquad \dfrac{}{z_a \star x_1 \star y'_a \star v'_1 \mathrel{\dot\in} A_1}\,3 \quad \zeta_{1.1} \quad Pre\ C_2 \vdash_{z_C} \star x_2 \qquad \zeta_{1.2} \quad z_a \star x_1 \star \vdash_{z_C}{}' \star x'_2 \in R \qquad \zeta_2 \quad Pre\ C\ (\vdash_{z_C} \star z_b \star x_c)}{Pre\ C\ (\vdash_{z_C} \star z_b \star x_c)}\,(SC_{1\,I}^-)(4)$$

$$\dfrac{Pre\ A\ (z_a \star z_b \star x_a)}{Pre\ C\ (\vdash_{z_C} \star z_b \star x_c)}\,(Pre^-_{|-|}(3))$$

$\zeta_{1.1}$ is:

$$\dfrac{C_2 \sqsupseteq^R_{\tau f}\ A_1 \qquad \zeta_{1.2} \quad z_a \star x_1 \star \vdash_{z_C}{}' \star x'_2 \in R \qquad \dfrac{\dfrac{}{z_a \star x_1 \star y'_a \star v'_1 \mathrel{\dot\in} A_1}\,3}{Pre\ A_1\ z_a \star x_1}\,(Pre^+)}{Pre\ C_2 \vdash_{z_C} \star x_2}\,(\sqsupseteq^-_{\tau f\ III})$$

$\zeta_{1.2}$ is:

$$\dfrac{x_1 \star x'_2 \in IO^{A_1}_{C_2} \qquad \zeta_{1.2.1} \quad \dfrac{\dfrac{z_a \star z_b \star x_a \star \vdash_{z_C}{}' \star z'_b \star x'_c \in S}{z_a \star z_b \star \vdash_{z_C}{}' \star z'_b \in Corr^A_C}\,(S^-_{io4})}{z_a \star \vdash_{z_C}{}' \in Corr^{A_1}_{C_2}}\,(df)}{z_a \star x_1 \star \vdash_{z_C}{}' \star x'_2 \in R}\,(S^+_{io})$$

$\zeta_{1.2.1}$ is:

$$\dfrac{\dfrac{\dfrac{}{x_2.i_{C_2} = (x_c.i_C \cup fb_{v_a}) \cap in_{C_2}}\,2}{x_1.i_{A_1} \cap in_{C_2} = x_2.i_{C_2} \cap in_{A_1}} \qquad \zeta_{1.2.2} \qquad \zeta_{1.2.3} \quad x_1.o_{A_1} \cap out_{C_2} = x_2.o_{C_2} \cap out_{A_1}}{x_1 \star x'_2 \in IO^{A_1}_{C_2}}$$

$\zeta_{1.2.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z_a \star z_b \star x_a \star \vdash_{z_C}{}' \star z'_b \star x'_c \in S}{x_a.i_A \cap in_C = x_c.i_C \cap in_A}\,(S^-_{io1}) \quad \dfrac{}{x_c.i_C \subseteq in_C}\,(V^{io}_C\text{-}df)}{x_a.i_A \cap in_C = x_c.i_C \cap in_C \cap in_A} \quad \dfrac{}{x_a.i_A \subseteq in_A}\,(V^{io}_A\text{-}df)}{x_a.i_A \cap in_A \cap in_C = x_c.i_C \cap in_C \cap in_A} \quad \dfrac{}{in_{C_2} \subseteq in_C}\,(df)}{x_a.i_A \cap in_A \cap in_{C_2} = x_c.i_C \cap in_{C_2} \cap in_A} \quad \dfrac{}{in_{A_1} \subseteq in_A}\,(df)}{x_a.i_A \cap in_{A_1} \cap in_{C_2} = x_c.i_C \cap in_{C_2} \cap in_{A_1}}$$

$$\dfrac{\dfrac{}{x_1.i_{A_1} = (x_a.i_A \cup fb_{v_a}) \cap in_{A_1}}\,3 \qquad (x_a.i_A \cup fb_{v_a}) \cap in_{A_1} \cap in_{C_2} = (x_c.i_C \cup fb_{v_a}) \cap in_{C_2} \cap in_{A_1}}{x_1.i_{A_1} \cap in_{C_2} = (x_c.i_C \cup fb_{v_a}) \cap in_{C_2} \cap in_{A_1}}$$

$\zeta_{1.2.3}$ is:

$$\dfrac{\dfrac{\dfrac{}{x_2.o_{C_2} = x_1.o_{A_1} \cap out_{C_2}}\,2}{x_2.o_{C_2} \cap out_{A_1} = x_1.o_{A_1} \cap out_{C_2} \cap out_{A_1}} \quad \dfrac{}{x_1.i_{A_1} \subseteq out_{A_1}}\,(V^{io}_1\text{-}df)}{x_1.o_{A_1} \cap out_{C_2} = x_2.o_{C_2} \cap out_{A_1}}$$

ζ_2 is:

$$\dfrac{\dfrac{}{z_b \star x_3 \star y'_b \star v'_3 \dot{\in} B}\,3 \quad \overset{\zeta_{2.1}}{\vdots} \quad \dfrac{}{\vdash_{z_c} \star x_2 \star y'_c \star v'_2 \dot{\in} C_2}\,4 \quad \overset{\zeta_{2.2}}{\vdots} \quad \dfrac{}{v_c.o_C = v_2.o_{C_2} \cup v_3.o_B}\,1\ (Pre^{+}_{|-|})}{Pre\ C\ (\vdash_{z_c} \star z_b \star x_c)}$$

$\zeta_{2.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z_a \star z_b \star x_a \star \vdash_{z_c}{}' \star z'_b \star x'_c \in S}{x_a.i_A \cap in_C = x_c.i_C \cap in_A}\,(S^{-}_{io1})}{x_a.i_A \cap in_C \cap in_B = x_c.i_C \cap in_A \cap in_B} \quad \dfrac{}{in_B \subseteq in_C}\,(df)}{x_a.i_A \cap in_B = x_c.i_C \cap in_A \cap in_B} \quad \dfrac{}{in_B \subseteq in_A}\,(df)}{x_a.i_A \cap in_B = x_c.i_C \cap in_B}}{\dfrac{\dfrac{}{x_3.i_B = (x_a.i_A \cup fb_{v_a}) \cap in_B}\,3 \qquad (x_a.i_A \cup fb_{v_a}) \cap in_B = (x_c.i_C \cup fb_{v_a}) \cap in_B}{x_3.i_B = (x_c.i_C \cup fb_{v_a}) \cap in_B} \qquad \overset{\zeta_3}{\vdots}\ fb_{v_c} = fb_{v_a}}{x_3.i_B = (x_c.i_C \cup fb_{v_c}) \cap in_B}$$

and $\zeta_{2.2}$ is:

$$\dfrac{\dfrac{}{x_2.i_{C_2} = (x_c.i_C \cup fb_{v_a}) \cap in_{C_2}}\,2 \qquad \overset{\zeta_3}{\vdots}\ fb_{v_c} = fb_{v_a}}{x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}}$$

and ζ_3 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{fb_{v_c} = v_c.o_C \cap \Psi}\,(df) \quad \dfrac{}{v_c.o_C = v_2.o_{C_2} \cup v_3.o_B}\,1}{fb_{v_c} = (v_2.o_{C_2} \cup v_3.o_B) \cap \Psi}}{\dfrac{}{v_1.o_{A_1} \cap \Psi = v_2.o_{C_2} \cap \Psi}\,4 \qquad fb_{v_c} = (v_2.o_{C_2} \cap \Psi) \cup (v_3.o_B \cap \Psi)}}{fb_{v_c} = (v_1.o_{A_1} \cap \Psi) \cup (v_3.o_B \cap \Psi)}}{\dfrac{fb_{v_c} = (v_1.o_{A_1} \cup v_3.o_B) \cap \Psi \qquad \dfrac{}{v_a.o_A = v_1.o_{A_1} \cup v_3.o_B}\,3}{fb_{v_c} = v_a.o_A \cap \Psi}}{fb_{v_c} = fb_{v_a}}\,(df)}$$

And finally for the property *corr* we have,

$$\dfrac{\dfrac{}{\vdash_{z_C} \star z_b \star x_c \star y'_c \star y'_b \star v'_c \dot{\in} C} \quad \overset{\zeta_1}{\vdots}\ \exists\, t \bullet z_a \star z_b \star x_a \star t' \dot{\in} A \wedge t \star y'_c \star y'_b \star v'_c \in S}{\exists\, t \bullet z_a \star z_b \star x_a \star t' \dot{\in} A \wedge t \star y'_c \star y'_b \star v'_c \in S}\,(Z^{-}_{|-|})(1)$$

where ζ_1 is:

$$\dfrac{z_a \star z_b \star x_a \star \vdash_{z_C}{}' \star z'_b \star x'_c \in S \quad \dfrac{}{x_2.i_{C_2} = (x_c.i_C \cup fb_{v_c}) \cap in_{C_2}}\,1 \quad \overset{\zeta_2}{\vdots}\ \exists\, t \bullet z_a \star z_b \star x_a \star t \dot{\in} A \wedge t \star y'_c \star y'_b \star v'_c \in S}{\exists\, t \bullet z_a \star z_b \star x_a \star t' \dot{\in} A \wedge t \star y'_c \star y'_b \star v'_c \in S}\,(S^{-}_{|-|_1})(2)$$

ζ_2 is:

$$\dfrac{C_2\ \sqsupseteq^{R}_{\tau f}\ A_1 \quad \dfrac{}{z_a \star x_1 \star \vdash_{z_C}{}' \star x'_2 \in R}\,2 \quad \overset{\zeta_{2.1}}{\vdots}\ Pre\ A_1\ z_a \star x_1 \quad \dfrac{}{\vdash_{z_C} \star x_2 \star y'_c \star v'_2 \dot{\in} C_2}\,1 \quad \overset{\zeta_3}{\vdots}}{\exists\, t \bullet z_a \star z_b \star x_a \star t' \dot{\in} A \wedge t \star y'_c \star y'_b \star v'_c \in S}\,(\sqsupseteq^{-}_{\tau f\ IV})(3)$$

$\zeta_{2.1}$ is:

$$\dfrac{\dfrac{}{z_a \star x_1 \star \vdash_{z_c}{}' \star x_2' \in R}\,2 \quad \dfrac{}{A_\Psi \sqsupseteq^T_{\tau f} C_\Psi}\,(SC_1) \quad \dfrac{}{\vdash_{z_c} \star x_2 \star y_c' \star v_2' \in C_2}\,1}{Pre\ A_1\ z_a \star x_1}\,(lem\ B.14.4)$$

ζ_3 is:

$$\dfrac{\dfrac{}{\exists\, v_a^{V_A^{io}} \bullet v_a.o_A = v_1.o_{A_1} \cup v_3.o_B \wedge \quad v_a.i_A = v_c.i_C \cap in_A}\,(V_A^{io}\text{-}df) \quad \begin{array}{c}\zeta_4\\ \vdots\\ \exists\, t \bullet z_a \star z_b \star x_a \star t \mathrel{\dot\in} A \wedge \quad t \star y_c' \star y_b' \star v_c' \in S\end{array}}{\exists\, t \bullet z_a \star z_b \star x_a \star t' \mathrel{\dot\in} A \wedge t \star y_c' \star y_b' \star v_c' \in S}\,(\exists^-)(5)$$

ζ_4 is:

$$\dfrac{\dfrac{\begin{array}{cc}\zeta_5 & \zeta_6\\ \vdots & \vdots\\ z_a \star z_b \star x_a \star y_a' \star y_b' \star v_a' \mathrel{\dot\in} A & y_a \star y_b \star v_a \star y_c' \star y_b' \star v_c' \in S\end{array}}{z_a \star z_b \star x_a \star y_a' \star y_b' \star v_a' \mathrel{\dot\in} A \wedge y_a \star y_b \star v_a \star y_c' \star y_b' \star v_c' \in S}\,(\wedge^+)}{\exists\, t \bullet z_a \star z_b \star x_a \star t \mathrel{\dot\in} A \wedge t \star y_c' \star y_b' \star v_c' \in S}\,(\exists^+)$$

ζ_5 is:

$$\dfrac{\dfrac{}{z_a \star x_1 \star y_a' \star v_1' \mathrel{\dot\in} A_1}\,3 \quad \begin{array}{c}\dfrac{}{z_b \star x_3 \star y_b' \star v_3' \mathrel{\dot\in} B}\,1\\ \vdots\end{array} \quad \dfrac{}{v_a.o_A = v_1.o_{A_1} \cup v_3.o_B}\,5 \quad \begin{array}{cc}\zeta_{5.1} & \zeta_{5.2}\\ \vdots & \vdots\end{array}}{z_a \star z_b \star x_a \star y_a' \star y_b' \star v_a' \mathrel{\dot\in} A}\,(|\_|^+)$$

$\zeta_{5.1}$ is:

$$\dfrac{\begin{array}{c}\zeta_{5.3}\\ \vdots\\ fb_{v_a} = fb_{v_c}\end{array} \quad \dfrac{}{x_1.i_{A_1} = (x_a.i_A \cup fb_{v_c}) \cap in_{A_1}}\,2}{x_1.i_{A_1} = (x_a.i_A \cup fb_{v_a}) \cap in_{A_1}}$$

$\zeta_{5.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z_a \star z_b \star x_a \star \vdash_{z_c}{}' \star z_b' \star x_c' \in S}{x_a.i_A \cap in_C = x_c.i_C \cap in_A}\,(S^-_{io1}) \quad \dfrac{}{in_B \subseteq in_C}\,(df)}{x_a.i_A \cap in_C \cap in_B = x_c.i_C \cap in_A \cap in_B} \quad \dfrac{}{in_B \subseteq in_A}\,(df)}{x_a.i_A \cap in_B = x_c.i_C \cap in_A \cap in_B}}{x_a.i_A \cap in_B = x_c.i_C \cap in_B}}{(x_a.i_A \cup fb_{v_c}) \cap in_B = (x_c.i_C \cup fb_{v_c}) \cap in_B}$$

$$\dfrac{\dfrac{\dfrac{}{x_3.i_B = (x_c.i_C \cup fb_{v_c}) \cap in_B}\,1 \quad \begin{array}{c}\zeta_3\\ \vdots\\ fb_{v_c} = fb_{v_c}\end{array}}{x_3.i_B = (x_c.i_C \cup fb_{v_c}) \cap in_B}}{x_3.i_B = (x_a.i_A \cup fb_{v_a}) \cap in_B}$$

$\zeta_{5.3}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v_a.o_A = v_1.o_{A_1} \cup v_3.o_B}\,5}{v_a.o_A \cap \Psi = (v_1.o_{A_1} \cup v_3.o_B) \cap \Psi}}{v_a.o_A \cap \Psi = (v_1.o_{A_1} \cap \Psi) \cup (v_3.o_B \cap \Psi)} \quad \begin{array}{c}\zeta_{5.3.1}\\ \vdots\end{array}}{v_a.o_A \cap \Psi = (v_2.o_{C_2} \cap \Psi) \cup (v_3.o_B \cap \Psi)} \quad \dfrac{\dfrac{}{v_c.o_C = v_2.o_{C_2} \cup v_3.o_B}\,1}{v_c.o_C \cap \Psi = (v_2.o_{C_2} \cap \Psi) \cup (v_3.o_B \cap \Psi)}}{v_a.o_A \cap \Psi = v_c.o_C \cap \Psi}$$

$$\dfrac{}{fb_{v_a} = fb_{v_c}}\,(df)$$

$\zeta_{5.3.1}$ is:

$$\dfrac{\dfrac{}{out_{A_1} \cap \Psi = out_{C_2} \cap \Psi}\,(SC_2) \quad \dfrac{\dfrac{}{y_a \star v_1 \star y'_c \star v'_2 \in R}\,3}{v_1.o_{A_1} \cap out_{C_2} = v_2.o_{C_2} \cap out_{A_1}}\,(S^-_{io2})}{\dfrac{\dfrac{}{v_2.o_{C_2} \subseteq out_{C_2}}\,(V^{io}_2\text{-}df) \quad \dfrac{v_1.o_{A_1} \cap out_{A_1} \cap \Psi = v_2.o_{C_2} \cap out_{C_2} \cap \Psi}{v_1.o_{A_1} \cap \Psi = v_2.o_{C_2} \cap out_{C_2} \cap \Psi}}{v_1.o_{A_1} \cap \Psi = v_2.o_{C_2} \cap \Psi}}\quad \dfrac{}{v_1.o_{A_1} \subseteq out_{A_1}}\,(V^{io}_1\text{-}df)$$

ζ_6 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{y_a \star v_1 \star y'_c \star v'_2 \in R}\,3}{y_a \star y'_c \in Corr^{A_1}_{C_1}} \quad \dfrac{}{y_b \star y'_b \in Corr^{B}_{B}}\,(df)}{y_a \star y_b \star y'_c \star y'_b \in Corr^{A_1}_{C_1} \wedge Corr^{B}_{B}}}{y_a \star y_b \star y'_c \star y'_b \in Corr^{A}_{C}}\,(df) \quad \begin{matrix}\zeta_{6.1}\\ \vdots \\ v_a \star v'_c \in IO^A_C\end{matrix}}{y_a \star y_b \star v_a \star y'_c \star y'_b \star v'_c \in S}\,(S^+_{io})$$

$\zeta_{6.1}$ is:

$$\dfrac{\dfrac{\dfrac{}{v_a.i_A = v_c.i_C \cap in_A}\,5}{v_a.i_A \cap in_C = v_c.i_C \cap in_A \cap in_C} \quad \dfrac{}{v_c.i_C \subseteq in_C}\,(V^{io}_C\text{-}df)}{v_a.i_A \cap in_C = v_c.i_C \cap in_A} \quad \begin{matrix}\zeta_{6.2}\\ \vdots \\ v_a.o_A \cap out_C = v_c.o_C \cap out_A\end{matrix}}{v_a \star v'_c \in IO^A_C}$$

$\zeta_{6.2}$ is:

$$\dfrac{\begin{matrix}\zeta_{6.3}\\ \vdots \\ v_c.o_C \cap out_A = (v_1.o_{A_1} \cup v_3.o_B) \cap out_C\end{matrix} \quad \dfrac{\dfrac{}{v_a.o_A = v_1.o_{A_1} \cup v_3.o_B}\,5}{v_a.o_A \cap out_C = (v_1.o_{A_1} \cup v_3.o_B) \cap out_C}}{v_a.o_A \cap out_C = v_c.o_C \cap out_A}$$

$\zeta_{6.3}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{v_3.o_B \subseteq out_B}\,(V^{io}_B\text{-}df)}{v_3.o_B \subseteq (out_{C_2} \cup out_B)} \quad \dfrac{\begin{matrix}\zeta_{6.4}\\ \vdots \\ v_c.o_C \cap out_A = ((v_1.o_{A_1} \cap out_{C_2}) \cup (v_1.o_{A_1} \cap out_B)) \cup v_3.o_B\end{matrix}}{v_c.o_C \cap out_A = (v_1.o_{A_1} \cap (out_{C_2} \cup out_B)) \cup v_3.o_B}}{v_c.o_C \cap out_A = (v_1.o_{A_1} \cap (out_{C_2} \cup out_B)) \cup (v_3.o_B \cap (out_{C_2} \cup out_B))}}{v_c.o_C \cap out_A = (v_1.o_{A_1} \cap out_C) \cup (v_3.o_B \cap out_C)}}{v_c.o_C \cap out_A = (v_1.o_{A_1} \cup v_3.o_B) \cap out_C}$$

$\zeta_{6.4}$ is:

$$\dfrac{\dfrac{\begin{matrix}\zeta_{6.4.1}\\ \vdots \\ v_1.o_{A_1} \cap out_B = v_2.o_{C_2} \cap out_B\end{matrix} \quad \dfrac{\begin{matrix}\zeta_{6.4.2}\\ \vdots \\ v_c.o_C \cap out_A = (v_1.o_{A_1} \cap out_{C_2}) \cup (v_2.o_{C_2} \cap out_B) \cup v_3.o_B\end{matrix}}{v_1.o_{A_1} \cap out_{C_2} = v_2.o_{C_2} \cap out_{A_1}}\,(S^-_{io2})}{v_c.o_C \cap out_A = ((v_1.o_{A_1} \cap out_{C_2}) \cup (v_1.o_{A_1} \cap out_B)) \cup v_3.o_B}}{}$$

$\zeta_{6.4.1}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y_a \star v_1 \star y'_c \star v'_2 \in R}\,3}{v_1.o_{A_1} \cap out_{C_2} = v_2.o_{C_2} \cap out_{A_1}}\,(S^-_{io2}) \qquad \dfrac{}{out_{A_1} \cap out_B = out_{C_2} \cap out_B}\,(SC_3)}{v_1.o_{A_1} \cap out_{C_2} \cap out_B = v_2.o_{C_2} \cap out_{A_1} \cap out_B} \qquad \dfrac{}{v_1.o_{A_1} \subseteq out_{A_1}}\,(V_1^{io}\text{-}df)}{v_1.o_{A_1} \cap out_{A_1} \cap out_B = v_2.o_{C_2} \cap out_{C_2} \cap out_B} \qquad \dfrac{}{v_2.o_{C_2} \subseteq out_{C_2}}\,(V_2^{io}\text{-}df)}{v_1.o_{A_1} \cap out_B = v_2.o_{C_2} \cap out_{C_2} \cap out_B}}{v_1.o_{A_1} \cap out_B = v_2.o_{C_2} \cap out_B}$$

and $\zeta_{6.4.2}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v_c.o_C = v_2.o_{C_2} \cup v_3.o_B}\,1}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cup v_3.o_B) \cap out_A}\,(df)}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cap out_A) \cup (v_3.o_B \cap out_A)} \qquad \dfrac{\dfrac{}{v_3.o_B \subseteq out_B}\,(V_3^{io}\text{-}df)}{v_3.o_B \subseteq (out_{A_1} \cup out_B)}\,(df)}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cap out_A) \cup (v_3.o_B \cap (out_{A_1} \cup out_B)}}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cap out_A) \cup v_3.o_B}}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cap (out_{A_1} \cup out_B)) \cup v_3.o_B}}{v_c.o_C \cap out_A = (v_2.o_{C_2} \cap out_{A_1}) \cup (v_2.o_{C_2} \cap out_B) \cup v_3.o_B}$$

Proposition 7.1.3 For arbitrary charts A, C and signal set Ψ we have,

$$\frac{C \sqsupseteq^{S}_{\tau f} A \qquad [A]_{\Psi} \sqsupseteq^{T}_{\tau f} [C]_{\Psi}}{C \sqsupseteq^{S}_{fcf} A} \qquad \frac{C \sqsupseteq^{S}_{fcf} A}{C \sqsupseteq^{S}_{\tau f} A}$$

where $S =_{def} Corr^{A}_{C} \wedge IO^{A}_{C}$ and $T =_{def} Corr^{C}_{A} \wedge IO^{C_{\Psi}}_{A_{\Psi}}$ for $C_{\Psi} = [C]_{\Psi}$ and $A_{\Psi} = [A]_{\Psi}$.

Proof 7.1.3

We begin by showing that the firing conditions refinement relation $\sqsupseteq_{fcf}$ satisfies the total chaos elimination rule $(\sqsupseteq^{-}_{\tau f\ IV})$, *i.e.* the elimination rule due to the *correctness* condition as outlined in Section 6.6.

$$\frac{C \sqsupseteq_{fcf} A \quad y_a \star y'_c \in R \quad y_c \star \vdash z_c{}' \dot{\in} C \qquad \begin{array}{c} \overline{y_a \star b'_1 \dot{\in} A}^{\,1} \quad \overline{b_1 \star \vdash z_c \in R}^{\,1} \\ \vdots \qquad\qquad \vdots \\ P \end{array}}{P}\ (\sqsupseteq^{-}_{fcf\ IV})(1)$$

The remaining elimination rules, $(\sqsupseteq^{-}_{\tau f\ I})$, $(\sqsupseteq^{-}_{\tau f\ II})$ and $(\sqsupseteq^{-}_{\tau f\ III})$ are trivially satisfied by $(\sqsupseteq^{-}_{\tau f\ I})$. Therefore, that $C \sqsupseteq^{S}_{fcf} A \models C \sqsupseteq^{S}_{\tau f} A$ holds (*i.e.* the right-hand property of Proposition 7.1.3) follows trivially using the introduction rule $(\sqsupseteq^{+}_{\tau f})$.

We use a similar argument for the left-hand property of Proposition 7.1.3. First we show that total chaos refinement relation $\sqsupseteq_{\tau f}$, along with the side-condition SC_1, satisfies the firing conditions elimination rule $(\sqsupseteq^{-}_{fcf\ IV})$.

$$\frac{C \sqsupseteq_{\tau f} A \quad \begin{array}{c} \zeta_1 \\ \vdots \\ Pre\ A\ y_a \end{array} \quad y_a \star y'_c \in R \quad y_c \star \vdash z_c{}' \dot{\in} C \qquad \begin{array}{c} \overline{y_a \star b'_1 \dot{\in} A}^{\,1} \quad \overline{b_1 \star \vdash z_c \in R}^{\,1} \\ \vdots \qquad\qquad \vdots \\ P \end{array}}{P}\ (\sqsupseteq^{-}_{\tau f\ IV})(1)$$

where ζ_1 is:

$$\frac{[A]_{\Psi} \sqsupseteq^{T}_{\tau f} [C]_{\Psi} \quad y_a \star y'_c \in S \quad y_c \star \vdash z_c{}' \dot{\in} C}{Pre\ A\ y_a}\ (lem\ B.14.4)$$

Again the remaining elimination rules for firing conditions refinement are trivially satisfied by $\sqsupseteq_{\tau f}$. Therefore, using this and the introduction rule for firing conditions refinement $(\sqsupseteq^{+}_{fcf})$ gives us a trivial proof of the completeness property (left-hand side) of Proposition 7.1.3.

Bibliography

[1] C. André. Representation and analysis of reactive behaviours: A synchronous approach. In *Proceedings of CESA96 (Computational Engineering in Systems Applications)*, pages 19–29, Lille, 1996. IEEE.

[2] C. André, M. Peraldi-Frati, and J. Rigault. Integrating the synchronous paradigm into UML: Application to control-dominated systems. In J. Jézéquel, H. Hussmann, and Stephen Cook, editors, *UML 2002—The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference, Dresden, Germany, September/October 2002, Proceedings*, volume 2460 of *LNCS*, pages 163–178. Springer, 2002.

[3] R. J. R. Back. A calculus of refinements for program derivations. *Acta Informatica*, 25(6):593–624, August 1988.

[4] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction.* Springer-Verlag, 1998.

[5] G. Berry. The Foundations of Esterel. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner.* MIT Press, 2000.

[6] G. Berry and G. Gonthier. The Esterel synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, November 1992.

[7] J. Bowen. Extending μ-charts. Dissertation, Computer Science Department, University of Waikato, New Zealand, 2003.

[8] J. P. Bowen. *Formal Specification and Documentation using Z.* International Thompson Computer Press, 1996.

[9] J. P. Bowen and M. G. Hinchey. Ten commandments of formal methods. *Computer*, 28(4):56–63, 1995.

[10] J. P. Bowen and V. Stavridou. The industrial take-up of formal methods in safety-critical and other areas: A perspective. In *FME '93: Proceedings of the First International Symposium of Formal Methods Europe on Industrial-Strength Formal Methods*, pages 183–195. Springer-Verlag, 1993.

[11] R. M. Burstall and J. Darlington. Some transformations for developing recursive programs. *Journal of the ACM*, 24(1):44–67, 1977.

[12] P. Caspi, C. Mazuet, and N. R. Paligot. About the design of distributed control systems: The quasi-synchronous approach. In *SAFECOMP '01: Proceedings of the 20th International Conference on Computer Safety, Reliability and Security*, pages 215–226. Springer-Verlag, 2001.

[13] A. Cavalcanti and J. Woodcock. ZRC—a refinement calculus for Z. *Formal Aspects of Computing*, 10:267–289, 1998.

[14] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computing*, pages 231–274, 1987.

[15] J. de Bakker and E. de Vink. *Control Flow Semantics.* MIT Press, Cambridge, MA, 1996.

[16] J. Derrick and E. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications.* Formal Approaches to Computing and Information Technology. Springer, May 2001.

[17] M. Deutsch, M. Henson, and S. Reeves. Six theories of operation refinement for partial relation semantics. Technical Report CSM-363, Department of Computer Science Department, University of Essex, 2002.

[18] M. Deutsch and M. C. Henson. An analysis of backward simulation data-refinement for partial relation semantics. In *Proceedings of the 10th Asia-Pacific Software Engineering Conference (APSEC 2003)*, pages 38–48, Chiang Mai, Thailand, December 2003. IEEE Computer Society.

[19] M. Deutsch and M. C. Henson. An analysis of forward simulation data refinement. In D. Bert, J.P. Bowen, S. King, and M. Waldén, editors, *ZB 2003: Formal Specification and Development in Z and B / Third International Conference of B and Z Users*, volume 2651 of *Lecture Notes in Computer Science*, pages 148–167. Springer-Verlag Heidelberg, August 2003.

[20] M. Deutsch, M. C. Henson, and S. Reeves. An analysis of total correctness refinement models for partial relation semantics I. *Logic Journal of the IGPL*, 11(3):287–317, 2003.

[21] M. Deutsch, M. C. Henson, and S. Reeves. Operation refinement and monotonicity in the schema calculus. In D. Bert, J. P. Bowen, S. King, and M. Walden, editors, *ZB 2003: Formal Specification and Development in Z and B*, volume 2651 of *Lecture Notes in Computer Science*, pages 103–126. Springer-Verlag, 2003.

[22] E. W. Dijkstra. Notes on structured programming. In *Structured Programming*. Academic Press, 1969.

[23] E. W. Dijkstra. *A Discipline of Programming*, chapter 14. Prentice-Hall, Englewood Cliffs, N. J., 1976.

[24] D. W. Embley, R. B. Jackson, and S. N. Woodfield. OO systems analysis: Is it or isn't it? *IEEE Software*, 12(4):19–33, 1995.

[25] R. W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume 19 of *Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, Providence, RI, 1967.

[26] S. Gerhart. Correctness preserving program transformations. In *Conference Record of the Second annual ACM Symposium on Principles of Programming Languages*, pages 54–66. ACM, January 1975.

[27] R. L. Glass. The mystery of formal methods disuse. *Commun. ACM*, 47(8):15–17, 2004.

[28] G. Goldson, G. Reeve, and S. Reeves. μ-Chart-based specification and refinement. In *Formal Methods and Software Engineering. 4th International Conference on Formal Engineering Methods, ICFEM 2002*, LNCS 2495, pages 323–334, Shanghai, China, October 2002. Springer.

[29] D. Grant and L. Stirling, editors. *Idioms for μ-Charts*, Canberra, Australia, August 2001. IEEE Computer Society.

[30] D. Gries. *The Science of Programming*. Springer, New York, 1981.

[31] R. Grosu, T. Stauner, and M. Broy. A modular visual model for hybrid systems. *Lecture Notes in Computer Science*, 1486:75–, 1998.

[32] L. Groves. Refinement and the Z schema calculus. In J. Woodcock, J. Derrick, E. Boiten, and J. von Wright, editors, *Proc. REFINE'02, Copenhagen, July 20-21, 2002*, volume 70, number 3 of *Electronic Notes in Theoretical Computer Science*, 2002. (See `http://www.mcs.vuw.ac.nz/~lindsay/Papers`.).

[33] N. Halbwachs. Synchronous programming of reactive systems. In *Computer Aided Verification*, pages 1–16, 1998.

[34] A. Hall. Seven myths of formal methods. *IEEE Software*, 7(5):11–19, 1990.

[35] D. Harel and A. Naamad. The statemate semantics of statecharts. *ACM Trans. Softw. Eng. Methodol.*, 5(4):293–333, 1996.

[36] I. Hayes, editor. *Specification Case Studies.* Prentice-Hall, 1987.

[37] M. C. Henson. The standard logic of Z is inconsistent. *Formal Aspects of Computing Journal*, 10(3):243–247, 1998.

[38] M. C. Henson and S. Reeves. Investigating Z. *Journal of Logic and Computation*, 10(1):1–30, 2000.

[39] M. C. Henson and S. Reeves. Program development and specification refinement in the schema calculus. In J. P. Bowen, S. Dunne, A. Galloway, and S. King, editors, *ZB2000: Formal Specification and Development in Z and B*, volume 1878 of *Lecture Notes in Computer Science*, pages 344–362. Springer-Verlag, 2000.

[40] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[41] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[42] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall International Series in Computer Science, 1985.

[43] G. F. Hoffnagle. Experiences in systems evolution: Practical aspects and cautionary tales. In *Proceedings of APSEC2000.* IEEE Computer Society, 2000. Keynote Presentation.

[44] C. Michael Holloway. Why engineers should consider formal methods. In *Proceedings of the 16th AIAA/IEEE Digital Avionics Systems Conference*, volume 1, pages 1.3–16 - 1.3–22, Irvine, CA, October 1997.

[45] I. Horrocks. *Constructing the User Interface with Statecharts.* Addison-Wesley Longman Publishing Co., Inc., 1999.

[46] C. Huizing and R. Gerth. Semantics of reactive systems in abstract time. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *Proceedings of Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 291–314, Berlin, Germany, June 1992. Springer.

[47] Information technology—z formal specification notation—syntax, type system and semantics, iso/iec 13568:2002. ISO/IEC Z Standard.

[48] The ISuRF web site is `http://www.cs.waikato.ac.nz/Research/fm/isurf.html`.

[49] J. Jacky. *The Way of Z: Practical programming with formal methods.* Cambridge University Press, 1997.

[50] S. King. Z and the Refinement Calculus. In D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors, *VDM '90 VDM and Z—Formal Methods in Software Development*, volume 428 of *Lecture Notes in Computer Science*, pages 164–188. Springer-Verlag, April 1990.

[51] B. Kolman and R. C. Busby. *Discrete Mathematical Structures for Computer Science.* Prentice-Hall, Inc., 2nd edition, 1986.

[52] N. Lynch and F. Vaandrager. Forward and backward simulations, part I: Untimed systems. *Information and Computation*, 121(2):214–233, 1995.

[53] M. C. Henson M. Deutsch. An analysis of total correctness refinement models for partial relation semantics ii. *Logic Journal of the IGPL*, 11(3):319–352, 2003.

[54] A.P. Martin. A Revised Deductive System for Z. Technical report, Software Verification Research Centre, February 1998. Available from `http://svrc.it.uq.edu.au/ apm/logic/logictr.ps`.

[55] The $\mathcal{Z}_\lambda$ web site is `http://www.cs.waikato.ac.nz/Research/fm/index.html`.

[56] R. Milner. *A Calculus of Communicating Systems.* Springer, Berlin, 1 edition, 1980.

[57] R. A. Milner. *Communication and Concurrency.* Prentice Hall, New York, 1989.

[58] C. Morgan. *Programming from specifications.* Prentice-Hall, Inc., 1990.

[59] C. Morgan. *Programming from Specifications: Second Edition.* Prentice Hall International, Hempstead, UK, 1994.

[60] J. M. Morris. Programs from specifications. In E. W. Dijkstra, editor, *Formal Development of Programs and Proofs*, pages 81–115. Addison-Wesley, Reading, MA, 1990.

[61] P. Naur. Proof of algorithms by general snapshots. *Nordisk tidskrift for informationsbehandling*, 6(4):310–316, 1966.

[62] D. Nazareth, F. Regensburger, and P. Scholz. Mini-statecharts: A lean version of statecharts. Technical Report TUM-I9610, Technische Univerität München, 1996.

[63] OMG Model Driven Architecture (MDA). *OMG*'s *MDA* web site can be found at `http://www.omg.org/mda`.

[64] OMG Unified Modeling Language (UML). *OMG*'s *UML* web site can be found at `http://www.uml.org`.

[65] J. Philipps and P. Scholz. Compositional specification of embedded systems with statecharts. In M. Bidoit and M. Dauchet, editors, *TAPSOFT '97: Theory and Practice of Software Development*, number 1214 in LNCS, pages 637–651. Springer-Verlag, 1997.

[66] J. Philipps and P. Scholz. Formal verification of statecharts with instantaneous chain reaction. In E. Brinksma, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1217 of *Lecture Notes in Computer Science*, pages 224–238. Springer-Verlag, 1997.

[67] J. Philipps and P. Scholz. Formal verification and hardware design with statecharts. In B. Moller and J. V. Tucker, editors, *Prospects for hardware foundations: ESPRIT Working Group 8533: NADA–new hardware design methods, survey chapters*, volume 1546 of *Lecture Notes in Computer Science*, pages 356–389. Springer-Verlag, 1998.

[68] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z.* Prentice Hall, New York, 1991.

[69] *Quantum Leaps:* The *Quantum Leaps* glossary page can be found at `http://www.quantum-leaps.com/resources/glossary.htm`.

[70] Robin Bloomfield's Adelard web site can be found at `http://www.adelard.co.uk/info/staff/rbloomfield.htm`.

[71] B. Ratcliff. *Introducing Specification using Z: a practical case study guide.* McGraw-Hill International, 1994.

[72] G. Reeve and S. Reeves. μ-Charts and Z: Examples and extensions. In *Proceedings of APSEC2000.* IEEE Computer Society, 2000.

[73] G. Reeve and S. Reeves. μ-Charts and Z: Extending the translation. Technical Report 00/11, Department of Computer Science, University of Waikato, 2000.

[74] G. Reeve and S. Reeves. μ-Charts and Z: Hows, whys and wherefores. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *Integrated Formal Methods 2000: Proceedings of the 2nd. International Workshop on Integrated Formal Methods*, LNCS 1945. Springer-Verlag, 2000.

[75] A. W. Roscoe, C. A. R. Hoare, and R. Bird. *The Theory and Practice of Concurrency.* Prentice Hall PTR, 1997.

[76] P. Scholz. An extended version of mini-statecharts. Technical Report TUM-I9628, Technische Univerität München, 1996.

[77] P. Scholz. A refinement calculus for statecharts. In E. Estesiano, editor, *Fundamental approaches to software engineering: First International Conference, FASE'98*, volume 1382 of *Lecture Notes in Computer Science*, pages 285–301. Springer-Verlag, Berlin, 1998.

[78] P. Scholz. *Design of Reactive Systems and their Distributed Implementation with Statecharts.* PhD thesis, Institut für Informatik, Technische Universität München, August 1998. TUM-I9821.

[79] B. Selic, G. Gullekson, and P. T. Ward. *Real-time object-oriented modeling.* John Wiley & Sons, Inc., 1994.

[80] J. M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*, volume 3 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1988.

[81] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 2nd edition, 1992.

[82] The Z notation web site is `http://vl.zuser.org/`.

[83] W. M. Turski. Essay on software engineering at the turn of century. In *FASE '00: Proceedings of the Third Internationsl Conference on Fundamental Approaches to Software Engineering*, pages 1–20. Springer-Verlag, 2000.

[84] M. von der Beeck. A comparison of statecharts variants. *Lecture Notes in Computer Science*, 863:128–, 1994.

[85] J. M. Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–23, 1990.

[86] N. Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, April 1971.

[87] J. Woodcock and Ana Cavalcanti. The semantics of circus. In *ZB*, pages 184–203, 2002.

[88] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice Hall, 1996.

[89] J. C. P. Woodcock and S. M. Brien. W: A logic for Z. In *Proceedings of the Z User Workshop*, pages 77–96. Springer-Verlag, 1992.

[90] J. B. Wordsworth. *Software Development with Z: A Practical Approach to Formal Methods in Software Engineering*. Addison-Wesley Publishing Company, 1994.