Working Paper Series
ISSN 1170-487X

Revising Z: semantics and logic

by Martin C Henson and Steve Reeves

Working Paper 98/4
March 1998

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Revising Z: semantics and logic

Martin C. Henson and Steve Reeves

Department of Computer Science,
University of Essex, England
hensm@essex.ac.uk
fax: +44 1206 872788
Department of Computer Science,
University of Waikato, New Zealand
stever@lucy.cs.waikato.ac.nz
fax: +64 7 8384155

Abstract. We introduce a simple specification logic Z_C comprising a logic and semantics (in ZF set theory). We then provide an interpretation for (a rational reconstruction of) the specification language Z within Z_C. As a result we obtain a sound logic for Z, including the schema calculus. A consequence of our formalisation is a critique of a number of concepts used in Z. We demonstrate that the complications and confusions which these concepts introduce can be avoided without compromising expressibility.

Keywords: Specification language Z; Logics and semantics of specification languages

1 Introduction

1.1 Background

The specification language Z has been in existence, and has been very widely used, for more than a decade. In view of this, it is perhaps somewhat surprising to discover that there exists no definitive account of either its proof theory or semantics. The variety of distinct interpretations one can find in the wealth of textbooks (*e.g.* [PST96], [BJ95], [Bow96], [Dil90], [Dil94], [Hay87], [Har96], [Jac97], [MP93], [Rat94], [She95], [Spi92], [WD96] is a non-exhaustive list), and the informal style in which most of these are written, comes as an unwelcome and unsatisfactory observation.

On the other hand there has been a major effort to provide proof-tool support for Z. This has very recently been surveyed in [Mar97], where three approaches are found to have been used: syntactic encodings of custom logics for Z and deep versus shallow semantic encodings of the language of Z within a meta-logical framework. The latter *induce* a logic via the semantic embedding (whether shallow or deep) but this still leaves an open question regarding the correctness of the chosen encoding (see *ibid.*, section 3.1). The former, arguably the deepest encodings (*ibid.*), presuppose the existence of a Z logic and not just a language. What emerges clearly, with either semantic or syntactic encoding, is that a complete and comprehensive logic for the language of Z is a prerequisite. The problem is, however, that no completely satisfactory logic exists, although much progress has been made (*e.g.* [WB92], [Bri95], [Nic95], [HM97], [Toy97], [WD96]).

Where there have been attempts to provide Z with a logic we find an unusual absence of meta-mathematical results. Such results should be almost as important for the development of Z as the existence of the proof-theory and semantics themselves. For example, it is only by investigating the meta-mathematics of proposed systems that one can evaluate, in a controlled and precise manner, their suitability. Only by this means can one ensure a properly organised formal system that is adequate for the task. In addition, meta-mathematical analysis is also capable of revealing those concepts of the system that are cumbersome, confused, redundant or unwieldy. Usually, a language would be introduced with its logic but this was not the case with Z. As a consequence, it should not be too surprising to discover that the task of providing a logic would involve, to some extent, a reappraisal of the language.

In this paper, our aim is to begin the process of establishing a logic for Z, guided by meta-mathematical investigation. We do not claim by any means to have completed the task of providing a complete and satisfactory account, but we do feel that what follows establishes a methodologically sound trajectory for further work. Our work should be considered as complementary to, rather than as a competitor of, the work which continues on the Z standard (which is working towards completing a logic for (standard)

Z) because we feel it necessary to allow the mathematical investigation to form, in part, a *critique* of the language Z as it is generally understood. It will, in the future, be very interesting to compare the approaches.

Our work should be compared with that in the literature concerned directly with the provision of a logic for Z, most notably the logic $\mathcal{W}$ [WB92] which has informed [Nic95] and [WD96]. As [Mar97] has accurately diagnosed, the existence of such a logic is a *precondition* for the development of proof-tools for Z, and is an entirely separate research area[1]. This We will make some further remarks in section 9.

1.2 Organisation of the paper

In the first section we introduce a specification system Z_C which is essentially a typed set theory incorporating the notion of a *schema type*. This system is, when compared with Z in its notational scope, very simple indeed. Z_C is much more than a notation: it is a specification *logic* and the language is associated with both rules for determining the types of terms, rules for determining that propositions are well-formed, and rules of inference. The meta-mathematical measure we investigate is the property of *syntactic consistency*: we show that all provably true (proto-)propositions are well-formed. Z_C forms an effective bridge between Z proper and the intended model in classical extensional set theory. We go on to show how Z_C may be interpreted quite simply in ZF using, in particular, a suitable dependent product operation over a family of sets over a very small (in ordinal terms) universe of sets. The interpretation is shown to be suitable by means of soundness results of a standard and, in this case, very simple kind.

In the following section we introduce a notation which is very much closer to the Z familiar in the literature. We include more substantial mechanisms for the construction of propositions, sets and, in particular, schema: an algebra of schema operations is provided from which other common forms may be defined. This notation is also equipped with a system of rules for type assignment for terms and propositionhood, together with a logic. As before, we can show that this system is syntactically consistent with respect to type assignment and propositionhood. Following this we are able to describe an interpretation for this Z within the much simpler system Z_C. This interpretation has several significant features. We are able to provide *compositional* interpretations for the algebra of schema operations[2]. Additionally we are able to test our interpretation by showing that it preserves the type assignment and propositionhood systems of both Z and Z_C. Finally we are able to demonstrate that the interpretation of Z in Z_C is sound: it preserves entailment. By composing results we obtain a soundness theorem for Z in the intended model ZF.

Three standard concepts of Z remain at this stage, and these are without doubt the locus of much confusion in the literature. Our final technical section reviews these notions and we provide additional mechanisms suitable to either interpret or modify them. In conclusion we examine some example specifications from the literature which employ the full range of the apparatus available in Z and demonstrate how such specifications may be rendered in our system.

2 The specification logic Z_C

In this section we shall describe a simple specification logic which we call Z_C. It is based upon the notion of *schema type* which has been introduced in Z. Our strategy will be to interpret higher level features of Z within this logic. The idea of interpreting the *language* of Z within a small core *language* is not new. Our approach is novel in presenting a core specification *logic* and undertaking a systematic mathematical analysis.

2.1 The language of Z_C

We begin with types.

$$T ::= \mathbb{N} \mid \mathbb{P}\, T \mid T \times T \mid [D]$$

[1] It is however, a very important research area and the reader is encouraged to read [Mar97], which goes on, beyond the organising remarks that we have made use of here, to provide a comprehensive review of existing work to date, the most notable of which is certainly the shallow semantic embedding of [KSW96].

[2] This is in stark contrast to the usual presentations which are forced, due to the absence of an appropriate model, to describe these non-compositionally in terms of macro-expansion. Such an approach, as is usual with non-compositional definitions, is complex, and a formalisation, if available, will be correspondingly less useful.

Schema types $[D]$ are explicitly part of the type system. We take $\mathbb{N}$ rather than $\mathbb{Z}$ to be the primitive type of Z_C. This is a more natural choice from a logical point of view and we may impose conventional rules for $\mathbb{N}$ from which other derived number systems, such as $\mathbb{Z}$, may be obtained.

Declarations are very simple.

$$D ::= l : T \mid l : T;\ D$$

Declarations of the form $l : T$ are called *prime* declarations. The labels l are *constants*. We shall write $[D] \preceq [D']$ when the set of prime declarations of D is a subset of that of[3] D'. Other meta-operations we shall need over schema types: $[D] \setminus [D']$ is the schema type comprising all prime declarations of $[D]$ which do not occur in $[D']$. When we are interested only in a single label we will write this as $[D] \setminus (l : T)$. The schema type $[D] \vee [D']$ is the schema type comprising the union of the prime declarations in D and D'. It is not defined when this union contains prime declarations $l : T$ and $l : T'$ when $T \neq T'$. Finally, we shall introduce meta-notational conventions which require substitution for labels. For this we need the *alphabet* operator, defined as follows: Let $[D] = [\cdots l_i : T_i \cdot]\mathrm{i}$. Then $\alpha[D] =_{df} \{\cdots l_i \cdots\}$ and we shall write $[\alpha[D]/t(\alpha[D])]$ to represent the family of substitutions: $[\cdots][l_i/t(l_i)][\cdots]$ where $t(l_i)$ (*etc.*) is some term involving the label l_i.

The formulæ of Z_C delineate a typed predicate logic. We begin with the type-free category which forms the basis for the language of well-typed formulæ we require.

The *proto-syntax* of formulæ is given by:

$$P ::= \bot \mid t_0 = t_1 \mid t \in C \mid \neg P \mid P_0 \vee P_1 \mid \exists z : T \bullet P$$

The logic of Z_C is classical and so the remaining logical connectives and the universal quantifier can be defined in terms of the above in the usual manner.

The proto-syntax of terms[4] is as follows:

$$t ::= x \mid n \mid C \mid t.l \mid \langle\!| \cdots l_i \Rightarrow t_i \cdots |\!\rangle \mid t.1 \mid t.2 \mid (t, t) \mid t \upharpoonright [D]$$

The last term formation operator will be unexpected, as it has had no history in Z so far as we can tell. We pronounce the symbol $\upharpoonright$ "filter" and its purpose is to permit the restriction of bindings to a given schema type. We shall see, in section 5.2 and in a variety of other places following that, how important these filtered terms are for constructing compositional interpretations for schema expressions.

The subcategory of numerals is as expected:

$$n ::= 0 \mid \mathit{succ}\ n$$

Finally the proto-syntax of sets:

$$C ::= \{z : T \mid P\}$$

We only include this as a separate category because it will play a more significant role in Z and thus permits a smoother transition to that more sophisticated language.

2.2 Type assignment and propositionhood in Z_C

Definition 1. Sequents of the system have the form: $\Gamma \rhd_C J$ where J is a judgement that a term has a type or that a proto-formula is a proposition[5]. Γ is a type assignment context for variables. Such contexts are understood to be *sets* and so are extended by taking a *union*, with the proviso that variables may

[3] Here, and in many places elsewhere in the paper, we have used a *prime* as a diacritical mark on metavariables. Except for a few instances in section 7 (when we discuss priming in Z in detail), primes in this paper *have no significance beyond their usual mathematical uses.* In particular, they *never* have the significance which has become common in Z. We reject entirely the use of priming as it has been practiced in Z as unnecessary, confused and mathematically unacceptable; but the reader will have to wait until section 7 before we can make good these claims.

[4] We might include some other operators as primitive, but there is no point in cluttering up the presentation with these.

[5] We will omit the subscript, whenever the context allows, in this system and in all the other systems which follow.

occur at most once. For clarity of presentation we shall omit the entailment symbol and all components of contexts which are irrelevant to, or which remain unchanged by, any rule.

$$\frac{}{\bot\ prop}\ (C_\bot) \qquad \frac{t_0 : T \quad t_1 : T}{t_0 = t_1\ prop}\ (C_=) \qquad \frac{t : T \quad C : \mathbb{P}\, T}{t \in C\ prop}\ (C_\in)$$

$$\frac{P\ prop}{\neg P\ prop}\ (C_\neg) \qquad \frac{P_0\ prop \quad P_1\ prop}{P_0 \vee P_1\ prop}\ (C_\vee) \qquad \frac{x : T \rhd P\ prop}{\exists\, x : T \bullet P\ prop}\ (C_\exists)$$

$$\frac{}{x : T \rhd x : T}\ (C_x) \qquad \frac{}{0 : \mathbb{N}}\ (C_0) \qquad \frac{n : \mathbb{N}}{succ\ n : \mathbb{N}}\ (C_s) \qquad \frac{x : T \rhd P\ prop}{\{x : T \mid P\} : \mathbb{P}\, T}\ (C_{\{\}})$$

$$\frac{t : [\cdots l_i : T_i \cdots]}{t.l_i : T_i}\ (C_.) \qquad \frac{\cdots \quad t_i : T_i \quad \cdots}{(\!| \cdots l_i \Rrightarrow t_i \cdots |\!) : [\cdots l_i : T_i \cdots]}\ (C_{\Rrightarrow}) \qquad \frac{t : T_1 \times T_2}{t.1 : T_1}\ (C_1)$$

$$\frac{t : T_1 \times T_2}{t.2 : T_2}\ (C_2) \qquad \frac{t_0 : T_0 \quad t_1 : T_1}{(t_0, t_1) : T_0 \times T_1}\ (C_{()}) \qquad \frac{t : T' \quad T \preceq T'}{t \upharpoonright T : T}\ (C_\upharpoonright)$$

Lemma 2 The generation lemma for Z_C.
(i) If $\Gamma \rhd P_0 \vee P_1$ prop then $\Gamma \rhd P_0$ prop and $\Gamma \rhd P_1$ prop
(ii) If $\Gamma \rhd \{x : T \mid P\} : T_0$ then $\Gamma, x : T \rhd P$ prop and $T_0 = \mathbb{P}\, T$

Proof. *(i)* There is only one rule with a conclusion of the form $\Gamma \rhd P_0 \vee P_1$ *prop*, namely: $(C_\vee)$.
(ii) Similarly.

These are two cases of a general result known as the *generation lemma*. There are many similar cases for all other conclusion forms and these are all proved in exactly the same manner. □

2.3 Some consequences of typechecking

Before we move on to discuss the logic of Z_C there are number of auxiliary results which concern the rules for proposition formation and type assignment introduced in definition 1.

Lemma 3.
(i) If x does not appear free in P and $x : T \rhd P$ prop then $\rhd P$ prop
(ii) If x does not appear free in t and $x : T \rhd t : T'$ then $\rhd t : T'$
(iii) If $\rhd t : T'$ and x does not appear free in t then $x : T \rhd t : T'$
(iv) If $\rhd P$ prop and x does not appear free in P then $x : T \rhd P$ prop □

Typing of terms is unique in Z_C.

Lemma 4. *If $t : T_0$ and $t : T_1$ then $T_0 = T_1$*

Proof. By the generation lemma (lemma 2). □

Lemma 5.
(i) If $P[z/t]$ prop and $t : T$ then $z : T \rhd P$ prop
(ii) If $z : T \rhd P$ prop and $t : T$ then $P[z/t]$ prop
(iii) If $t'[z/t] : T'$ and $t : T$ then $z : T \rhd t' : T'$
(iv) If $z : T \rhd t' : T'$ and $t : T$ then $t'[z/t] : T'$

Proof. The pairs *(i)*; *(iii)* and *(ii)*; *(iv)* are each proved by simultaneous induction on the structure of the relevant derivations. □

We shall need the following definition, which extends filtering from terms to comprehensions, *intra* Z_C in order to give the inference rules for filtered terms[6].

[6] For notational simplicity we shall, in the sequel, often write t_T for a term t such that $\rhd t : T$.

Definition 6. Suppose that $T \preceq T'$. $C_{\mathbb{P}\,T'} \restriction \mathbb{P}\,T =_{df} \{z : T \mid \exists\, x : T' \bullet x \in C \wedge z = x \restriction T\}$

Lemma 7. *The following rule is derivable:*

$$\frac{C : \mathbb{P}\,T' \quad T \preceq T'}{C \restriction \mathbb{P}\,T : \mathbb{P}\,T}\ (C_{\mathbb{P}\restriction})$$

Proof. This follows by rule $(C_{\{\}})$ providing that $z : T \rhd \exists\, x : T' \bullet x \in C_{T'} \wedge z = x \restriction T$ *prop* and this follows by rule $(C_{\exists})$ providing that $z : T, x : T' \rhd x \in C_{T'} \wedge z \in x \restriction T$ *prop*. By (derived) rule $(C_{\wedge})$ this reduces to $z : T, x : T' \rhd x \in C_{T'}$ *prop* and $z : T, x : T' \rhd x \restriction T$ *prop*. The former follows by rule $(C_{\in})$ since $z : T, x : T' \rhd x : T'$ holds by axiom (C_x) and $z : T, x : T' \rhd C : \mathbb{P}\,T'$ by assumption. The latter follows by rule $(C_{=})$ since $z : T, x : T' \rhd z : T$ holds by axiom (C_x) and $z : T, x : T' \rhd x \restriction T : T$ which, itself, follows by rule $(C_{\restriction})$ since, by axiom (C_x), we have $z : T, x : T' \rhd x : T'$ and $T \preceq T'$ by assumption. □

2.4 The logic of Z_C

The proto-judgements of the logic have the form:

$$\Gamma \vdash_C P$$

where a proto-context Γ has the form $\Gamma^-;\ \Gamma^+$. Γ^- is a type assignment context (a context for the type system) and Γ^+ is a set of formulæ. These are well-formed according to the following rules.

$$\frac{}{\Gamma^-\ \textit{context}} \qquad \frac{\Gamma^- \rhd P\ \textit{prop} \quad \Gamma\ \textit{context}}{\Gamma^-;\ P, \Gamma^+\ \textit{context}}$$

Proofs introduce new putative contexts, propositions and terms and the rules must be guarded in some cases by type judgements to ensure they remain type consistent. We shall establish that these conditions of well-formedness are maintained by the rules below (proposition 11). As before, we omit all data which remains unchanged by a rule.

Definition 8 Logic of Z_C.

$$\frac{\Gamma \vdash P_0 \quad \Gamma^- \rhd P_1\ \textit{prop}}{\Gamma \vdash P_0 \vee P_1}\ (\vee^+_0) \qquad \frac{\Gamma \vdash P_1 \quad \Gamma^- \rhd P_0\ \textit{prop}}{\Gamma \vdash P_0 \vee P_1}\ (\vee^+_1) \qquad \frac{P_0 \vee P_1 \quad P_0 \vdash P_2 \quad P_1 \vdash P_2}{P_2}\ (\vee^-)$$

$$\frac{\Gamma, P \vdash \bot \quad \Gamma^- \rhd P\ \textit{prop}}{\Gamma \vdash \neg P}\ (\neg^+) \qquad \frac{\neg\neg P}{P}\ (\neg^-) \qquad \frac{P \quad \neg P}{\bot}\ (\bot^+) \qquad \frac{\Gamma \vdash \bot \quad \Gamma^- \rhd P\ \textit{prop}}{\Gamma \vdash P}\ (\bot^-)$$

$$\frac{\Gamma \vdash P[z/t] \quad \Gamma^- \rhd t : T}{\Gamma \vdash \exists z : T \bullet P}\ (\exists^+) \qquad \frac{\exists z : T \bullet P_0 \quad y : T;\ P_0[z/y] \vdash P_1}{P_1}\ (\exists^-)$$

$$\frac{\Gamma, P\ \textit{context}}{\Gamma, P \vdash P}\ (\textit{ass}) \qquad \frac{\Gamma^- \rhd t : T}{\Gamma \vdash t = t}\ (\textit{ref})$$

$$\frac{\Gamma^- \rhd (\!| \cdots l_i \Rrightarrow t_i \cdots |\!) : T}{\Gamma \vdash (\!| \cdots l_i \Rrightarrow t_i \cdots |\!).l_i = t_i}\ (\Rrightarrow^=_0) \qquad \frac{\Gamma^- \rhd t : [\cdots l_i : T_i \cdots]}{\Gamma \vdash (\!| \cdots l_i \Rrightarrow t.l_i \cdots |\!) = t}\ (\Rrightarrow^=_1)$$

$$\frac{\Gamma^- \rhd (t, t') : T}{\Gamma \vdash (t, t').1 = t}\ ((,)^=_0) \qquad \frac{\Gamma^- \rhd (t, t') : T}{\Gamma \vdash (t, t').2 = t'}\ ((,)^=_1) \qquad \frac{\Gamma^- \vdash t : T \times T'}{\Gamma \vdash (t.1, t.2) = t}\ ((,)^=_2)$$

$$\frac{t' = t}{t = t'}\ (\textit{sym}) \qquad \frac{t = t' \quad P[z/t]}{P[z/t']}\ (\textit{sub}) \qquad \frac{P[n/0] \quad n : \mathbb{N};\ P \vdash P[n/\textit{succ}\ n]}{n : \mathbb{N} \vdash P}\ (\mathbb{N}^-)$$

$$\frac{\Gamma \vdash P[z/t] \quad \Gamma^- \triangleright t : T}{\Gamma \vdash t \in \{z : T \mid P\}}\ (\{\}^+) \qquad \frac{t \in \{z : T \mid P\}}{P[z/t]}\ (\{\}^-) \qquad \frac{z : T \vdash P_0 \Leftrightarrow P_1}{\{z : T \mid P_0\} = \{z : T \mid P_1\}}\ (\{\}^=)$$

$$\frac{\Gamma \vdash t.l_i = t_i \quad \Gamma^- \triangleright t : T' \quad [\cdots l_i : T_i \cdots] \preceq T'}{\Gamma \vdash (t \restriction [\cdots l_i : T_i \cdots]).l_i = t_i}\ (\restriction^=)$$

Since contexts are sets we do not require structural rules. The following weakening rule is clearly admissible and may be usefully added:

$$\frac{\Gamma\ context \quad \vdash P}{\Gamma \vdash P}$$

2.5 Consequences of the logic for Z_C

There should also be a number of equality congruence rules for the terms of Z_C. These are not included in the system because they are all easily derivable; essentially because we have the rule (*sub*) in addition to the rules which make equality an equivalence.

The following are derived rules of Z_C:

$$\frac{t_0 = t_1 \quad t_1 = t_2}{t_0 = t_2}\ (trans) \qquad \frac{t_0 = t_2 \quad t_1 = t_3}{(t_0, t_1) = (t_2, t_3)}\ (=_{()})$$

$$\frac{\Gamma \vdash t = t' \quad \Gamma^- \triangleright t : T \times T'}{\Gamma \vdash t.1 = t'.1}\ (=_{.1}) \qquad \frac{\Gamma \vdash t = t' \quad \Gamma^- \triangleright t : T \times T'}{\Gamma \vdash t.2 = t'.2}\ (=_{.2})$$

$$\frac{\cdots \quad t_i = t_i' \quad \cdots}{\langle\!| \cdots l_i \Rightarrow t_i \cdots |\!\rangle = \langle\!| \cdots l_i \Rightarrow t_i' \cdots |\!\rangle}\ (=_{\Rightarrow}) \qquad \frac{\Gamma \vdash t = t' \quad \Gamma^- \triangleright t : [\cdots l_i : T_i \cdots]}{\Gamma \vdash t.l_i = t'.l_i}\ (=_{.l})$$

$$\frac{\Gamma \vdash t = t' \quad \Gamma^- \triangleright t : T' \quad T \preceq T'}{\Gamma \vdash t \restriction T = t' \restriction T}\ (=_{\restriction})$$

In addition we have derived rules which relate filtered terms and filtered sets.

$$\frac{\Gamma \vdash t \in C \quad \Gamma^- \triangleright t : T' \quad T \preceq T'}{\Gamma \vdash t \restriction T \in C \restriction \mathbb{P}\, T}\ (\in^+_{\restriction})$$

This follows by rules $(C_{\restriction})$, $(\{\}^+)$ and $(\exists^+)$.

$$\frac{\Gamma \vdash t \in C \restriction \mathbb{P}\, T \quad \Gamma^-, x : T;\ \Gamma^+, x \in C, t = x \restriction T \vdash P}{\Gamma \vdash P}\ (\in^-_{\restriction})$$

This follows by rules $(\{\}^-)$ and $(\exists^-)$.

Finally, in view of rule $(=_{\{\}})$, we can prove that equality between sets is extensional.

Definition 9. $C_{\mathbb{P}\,T} \subseteq C'_{\mathbb{P}\,T} =_{df} \forall z : T \bullet z \in C \Rightarrow z \in C'$

Lemma 10. $C_{\mathbb{P}\,T} \subseteq C'_{\mathbb{P}\,T} \subseteq C_{\mathbb{P}\,T} \Leftrightarrow C = C'$

Proof. ($\Rightarrow$) By rule (*sub*).
($\Leftarrow$) By rules $(\{\}^-)$ and $(\{\}^=)$. □

2.6 Syntactic consistency for Z_C

The logic should only enable us to deduce well-formed propositions from well-formed assumptions. This is the content of the next result.

Proposition 11 Syntactic consistency. *If* $\Gamma \vdash_C P$ *when* Γ *context then* $\Gamma^- \triangleright_C P$ *prop*

Proof. By induction on the structure of the derivation $\Gamma \vdash P$. Suppose that Γ *context*.
Ad Rule $(\vee_0^+)$:
We have $\Gamma^- \triangleright P_0$ *prop ex hypothesi* and this together with the second premise gives us $\Gamma^- \triangleright P_0 \vee P_1$ *prop* as required by rule $(C_\vee)$.
Ad $(\vee_1^+)$: Similarly.
Ad Rule $(\vee^-)$:
We may assume *ex hypothesi* (first premise) that $\Gamma^- \triangleright P_0 \vee P_1$ *prop* since we have Γ *context* by assumption. From the generation lemma it follows that $\Gamma^- \triangleright P_0$ *prop* and this, together with the assumption that Γ *context* is sufficient to show that Γ, P_0 *context* and hence, *ex hypothesi* (second premise), that $(\Gamma, P_0)^- \triangleright P_2$ *prop*. But $(\Gamma, P_0)^- = \Gamma^-$ and so $\Gamma^- \triangleright P_2$ *prop* as required.
Ad Rule $(\neg^+)$:
$\Gamma^- \triangleright \neg P$ *prop* follows immediately by rule $(C_\neg)$ from the second premise.
Ad Rule $(\neg^-)$:
We have $\Gamma^- \triangleright \neg\neg P$ *prop ex hypothesi* and then $\Gamma^- \triangleright P$ *prop* follows from lemma 2.
Ad Rule $(\bot^+)$:
This follows immediately by rule $(C_\bot)$.
Ad Rule $(\bot^-)$:
This follows immediately from the second premise.
Ad Rule $(\exists^+)$:
We may assume *ex hypothesi* that $\Gamma^- \triangleright P[z/t]$ *prop* since we have Γ *context* by assumption. From this and $\Gamma^- \triangleright t : T$ we have $\Gamma^-, z : T \triangleright P$ *prop* by lemma $5(i)$. But then $\Gamma^- \triangleright \exists z : T \bullet P$ *prop* follows by rule $(C_\exists)$.
Ad Rule $(\exists^-)$:
We may assume *ex hypothesi* (first premise) that $\Gamma^- \triangleright \exists z : T \bullet P_0$ *prop* since we have Γ *context* by assumption. From the generation lemma it follows that $\Gamma^-, z : T \triangleright P_0$ *prop* and this, together with the assumption that Γ *context*, is sufficient to show that $\Gamma^-, z : T;\ \Gamma^+, P_0$ *context*. Since we know that the variable y is not free in P_0 we have, by alpha conversion, $\Gamma^-, y : T;\ \Gamma^+, P_0[z/y]$ *context* and then, *ex hypothesi* (second premise), that $\Gamma^-, y : T \triangleright P_1$ *prop*. But since y is not free in P_1 this reduces to to $\Gamma^- \triangleright P_1$ *prop* by lemma $3(i)$.
Ad Rule (ass):
This follows immediately from the premise.
Ad Rule (ref):
This follows by rule $(C_=)$ from the premise.
Ad Rule $(\Rrightarrow_0^=)$:
From the premise, by lemma 2, we have $T = [\cdots l_i : T_i \cdots]$ for types T_i and, in particular, that $\Gamma^- \triangleright t_i : T_i$. From the premise, using rule $(C_.)$, we obtain $\Gamma^- \triangleright (\!| \cdots l_i \Rrightarrow t_i \cdots |\!).l_i : T_i$ whence, by rule $(C_=)$, we may conclude that $\Gamma^- \triangleright (\!| \cdots l_i \Rrightarrow t_i \cdots |\!).l_i = t_i$ *prop* as required.
Ad Rule $(\Rrightarrow_1^=)$:
From the premise, using rule $(C_.)$ we have $\Gamma^- \triangleright t.l_i : T_i$ for all i. Hence, by rule $(C_{\Rrightarrow})$, we obtain $\Gamma^- \triangleright (\!| \cdots l_i \Rrightarrow t.l_i \cdots |\!) : [\cdots l_i : T_i \cdots]$. This, together with the premise allows us to conclude, by rule $(C_=)$, that $\Gamma^- \triangleright (\!| \cdots l_i \Rrightarrow t.l_i \cdots |\!) = t$ *prop* as required.
Ad Rule $(()_0^=)$:
From the premise, by lemma 2, we have $T = T_0 \times T_1$ for types T_0 and T_1, and $\Gamma^- \triangleright t : T_0$. From the premise, using rule (C_1), we obtain $\Gamma^- \triangleright (t, t').1 : T_0$. Then, using rule $(C_=)$ we may conclude that $\Gamma^- \triangleright (t, t').1 = t$ *prop* as required.
Ad Rule $(()_1^=)$:
Similarly.
Ad Rule $(()_2^=)$:
From the premise, using rules (C_1) and (C_2) we have $\Gamma^- \triangleright t.1 : T$ and $\Gamma^- \triangleright t.2 : T'$. Hence, by rule $(C_{()})$, we obtain $\Gamma^- \triangleright (t.1, t.2) : T \times T'$. Then, using rule $(C_=)$ we may conclude that $\Gamma^- \triangleright (t.1, t.2) = t$ *prop*

as required.
Ad Rule (*sym*):
We have $\Gamma^- \triangleright t' = t$ *prop ex hypothesi* and then $\Gamma^- \triangleright t : T$ and $\Gamma^- \triangleright t' : T$ by lemma 2 for some type T. $\Gamma^- \triangleright t = t'$ *prop* then follows by rule $(C_=)$.
Ad Rule (*sub*):
From the first premise we have, by lemma 2, that $\Gamma \triangleright t : T$ and $\Gamma \triangleright t' : T$ for some type T. From the former, and the fact that $\Gamma^- \triangleright P[z/t]$ *prop* follows *ex hypothesi*, we have $\Gamma^-, z : T \triangleright P$ *prop* by lemma $5(i)$. From this and the latter we have $\Gamma^- \triangleright P[z/t']$ *prop* as required.
Ad Rule $(\mathbb{N}^-)$:
From the first premise we obtain *ex hypothesi* $\Gamma^- \triangleright P[n/0]$ *prop*. We know by axiom (C_0) that $\Gamma^- \triangleright 0 : \mathbb{N}$ consequently by lemma $3(i)$ we have $\Gamma^-, n : \mathbb{N} \triangleright P$ *prop* as required.
Ad Rule $(\{\}^+)$:
From the assumption that Γ *context* we have, from the premise, $\Gamma^- \triangleright P[z/t]$ *prop* whence $\Gamma^-, z : T \triangleright P$ *prop* from $\Gamma^- \triangleright t : T$ and lemma $5(i)$. From this we obtain $\Gamma^- \triangleright \{z : T \mid P\} : \mathbb{P}\, T$ by rule $(C_{\{\}})$. Combining this with the second premise we have, by rule $(C_\in)$, $\Gamma^- \triangleright t \in \{z : T \mid P\}$ *prop* as required.
Ad Rule $(\{\}^-)$:
From the assumption that Γ *context* we have $\Gamma^- \triangleright t \in \{z : T \mid P\}$ *prop ex hypothesi.* From lemma 2 we may conclude that $\Gamma^- \triangleright t : T$ and that $\Gamma^- \triangleright \{z : T \mid P\} : \mathbb{P}\, T$. Similarly, by lemma 2, we have $\Gamma^-, z : T \triangleright P$ *prop* from the latter and this, together with the former, yields $\Gamma^- \triangleright P[z/t]$ *prop* by lemma $5(ii)$ as required.
Ad Rule $(\{\}^=)$:
Ex hypothesi we have $\Gamma^-, z : T \triangleright P_0 \Leftrightarrow P_1$ *prop* and by lemma 2 we may conclude that $\Gamma^-, z : T \triangleright P_0$ *prop* and $\Gamma^-, z : T \triangleright P_1$ *prop*. But then $\Gamma^- \triangleright \{z : T \mid P_0\} = \{z : T \mid P_1\}$ *prop* follows as required by rules $(C_{\{\}})$ and $(C_=)$.
Ad Rule $(\restriction^=)$:
Let us write T for the schema type $[\cdots l_i : T_i \cdots]$. *Ex hypothesi* we have $\Gamma^- \triangleright t.l_i = t_i$ *prop*. From this, using lemma 2 and the third premise we obtain $\Gamma^- \triangleright t_i : T_i$. From the second and third premises, using rule $(C_\restriction)$, we infer that $\Gamma^- \triangleright t \restriction T : T$. Then, by rule $(C_=)$ we may conclude that $\Gamma^- \triangleright (t \restriction T).l_i = t_i$ *prop* as required. □

The specification logic Z_C is essentially a typed set-theory in which, in particular, we have *schema* types. There are, however, no schema in Z_C and this may seem rather odd since these are archetypical of Z. In fact, given the schema types, schema are just special cases of the comprehensions. Specifically, we may introduce schema by metanotational convention using the following definition:

$$[\, D \mid P \,] =_{df} \{z : [D] \mid P[\alpha[D]/z.\alpha[D]]\}$$

Note that this device requires us to allow the meta-variable P to range over the proto-propositions *extended with labels as terms.* The definiendum is, of course, (proto-)syntactically valid in Z_C. Given this definition we may provide the following versions of the comprehension rules using the schema notation:

$$\frac{\Gamma \vdash P[\alpha[D]/t.\alpha[D]] \quad \Gamma^- \triangleright t : [D]}{\Gamma \vdash t \in [D \mid P]}\ (S^+) \qquad \frac{t \in [D \mid P]}{t \in [D]}\ (S_0^-) \qquad \frac{t \in [D \mid P]}{P[\alpha[D]/t.\alpha[D]]}\ (S_1^-)$$

Since schema are just special sets and sets are extensional (lemma 10) it is immediate that equality for schema is also extensional. We shall return to this in far more detail in section 5.2, where we show how the algebra of schema in its entirety can be represented in Z_C.

It is also clear that, in Z_C, there is very little else one expects from Z. We are proposing that Z_C be taken as an adequate base theory within which the much higher level features of Z can be interpreted. As such it plays an intermediate role between Z and classical extensional set-theory (which is the intended model for Z). To show that Z_C can play this role we must show that it can be faithfully interpreted in ZF, and we devote the next section to that task. The remainder of the paper is devoted to showing that Z_C is adequate for the interpretation of Z.

3 A model of Z_C in ZF

In this section we provide an interpretation $[\![\_]\!]_C$ from the language of Z_C into ZF and prove a variety of results. We shall omit the subscript on the interpretation function unless this is essential. The semantics

is extremely simple. The novelty lies in our interpretation of schema types as dependent products over a family of sets from a small (in ordinal terms) cumulative universe.

3.1 Types

The language of types Z_C is given by a simple context free grammar. Such a grammar is understood mathematically to be an inductive definition over an operator which determines a set (the language of the grammar). The closure ordinal for this induction (the ordinal at which iteration of the operator reaches a fixpoint) is ω because the operator in question is continuous. In other words, the non-terminal operators may be applied finitely, but unboundedly, often. Consequently the type structures which can be described are, from a set-theoretic perspective, rather trivial. Consider, therefore, the following definition in ZF which constructs a tiny cumulative hierarchy.

$$\begin{array}{lll} (i) & F(0) & = \mathbb{N} \\ (ii) & F(\alpha + 1) & = F(\alpha) \cup \mathbb{P}\, F(\alpha) \\ (iii) & F(\omega) & = \bigcup_{\alpha < \omega} F(\alpha) \end{array}$$

This function is guaranteed to exist by transfinite induction (in fact only transfinite induction below $\omega.2$ is required) and we then take $F(\omega + 1)$ to be the universe within which the type system of Z may be interpreted[7]. This universe is a *set*.

Let $B(X)$ be an I-indexed family of sets over $F(\omega)$ (That is, $B(X) \in I \to F(\omega + 1)$). Then we can define a *dependent function space* which is suitable for our purposes as follows:

$$\Pi_{(X \in I)}.B(X) =_{df} \{f \in I \to F(\omega) \mid (\forall\, i \in I)(f(i) \in B(i))\}$$

This we can harness to interpret the types of Z_C:

$$\begin{array}{lll} (i) & [\![\mathbb{N}]\!] & =_{df} \mathbb{N} \\ (ii) & [\![T_0 \times T_1]\!] & =_{df} [\![T_0]\!] \times [\![T_1]\!] \\ (iii) & [\![\mathbb{P}\, T]\!] & =_{df} \mathbb{P}\, [\![T]\!] \\ (iv) & [\![[\cdots l_i : T_i \cdots]]\!] & =_{df} \Pi_{(X \in I)}.B(X) \end{array}$$

where $I =_{df} \{\cdots l_i \cdots\}$ and $B(l_i) =_{df} [\![T_i]\!]$. The labels l_i can be modelled in ZF in any number of ways, for example as finite ordinals. The only important point is that they be distinguishable from one another. We shall write them in ZF as we do in Z for simplicity.

3.2 Sets, logic and terms

We shall translate the entire proto-syntax of Z_C into well-formed formulæ of ZF.

$$\begin{array}{ll} [\![\bot]\!] & =_{df} (\forall\, x)(\neg x = x) \\ [\![t_0 = t_1]\!] & =_{df} [\![t_0]\!] = [\![t_1]\!] \\ [\![t_0 \in t_1]\!] & =_{df} [\![t_0]\!] \in [\![t_1]\!] \\ [\![P_0 \vee P_1]\!] & =_{df} [\![P_0]\!] \vee [\![P_1]\!] \\ [\![\neg P]\!] & =_{df} \neg [\![P]\!] \\ [\![\exists\, z : T \bullet P]\!] & =_{df} (\exists\, z)(z \in [\![T]\!] \wedge [\![P]\!]) \end{array}$$

There are no special conditions to impose with respect to the judgement of propositionhood of Z_C, since ZF is an untyped language of sets. As a consequence we may interpret the judgement forms $\Gamma \triangleright_C P\ prop$ to mean that $[\![P]\!]$ is a well-formed formula of ZF. Since this is true for any P the judgement is (*semantically*) redundant.

The terms are straightforwardly interpreted. We take the usual definition of cartesian product in ZF in which ordered pairs are defined by $\langle x, y \rangle =_{df} \{\{x\}, \{x, y\}\}$. Then we make use of the maps

[7] Note that $F(\omega + 1)$ is closed under union, so we do not need to make special provision for the cartesian product in the construction of the universe since, in ZF, $A \times B \in \mathbb{P}\mathbb{P}\mathbb{P}(A \cup B)$.

$fst \in A \times B \to A$ such that $\langle a, b\rangle \overset{fst}{\mapsto} a$ and $snd \in A \times B \to B$ such that $\langle a, b\rangle \overset{snd}{\mapsto} b$.

$$
\begin{array}{ll}
[\![x]\!] & =_{df} x \\
[\![l]\!] & =_{df} l \\
[\![n]\!] & =_{df} n \\
[\![(t_0, t_1)]\!] & =_{df} \{\{[\![t_0]\!]\}, \{[\![t_0]\!], [\![t_1]\!]\}\} \\
[\![t.1]\!] & =_{df} fst\,[\![t]\!] \\
[\![t.2]\!] & =_{df} snd\,[\![t]\!] \\
[\![t.l]\!] & =_{df} [\![t]\!]\,[\![l]\!] \\
[\![\{x : T \mid P\}]\!] & =_{df} \{[\![x]\!] \in [\![T]\!] \mid [\![P]\!]\} \\
[\![\mathbb{P}\, t]\!] & =_{df} \mathbb{P}\,[\![t]\!] \\
[\![t_0 \times t_1]\!] & =_{df} [\![t_0]\!] \times [\![t_1]\!]
\end{array}
$$

3.3 Mathematical results

As a result of careful design the two crucial semantic results for Z_C are easy to prove.

Proposition 12 Soundness of type assignment for Z_C.
If $\Gamma \rhd_C t : T$ *then* $[\![\Gamma]\!] \vdash_{zf} [\![t]\!] \in [\![T]\!]$

Proof. We proceed by induction on the structure of the derivation. Most cases are straightforward. There are two with interest:
Ad Rule $(C_{\{\}})$: We have to show that: $\{[\![x]\!] \in [\![T]\!] \mid [\![P]\!]\} \in \mathbb{P}\,[\![T]\!]$ Although this is immediate it is worth observing that the premise of this rule was *not* required since $[\![P]\!]$ is a well-formed formula of ZF for the entire protosyntax of Z_C.
Ad Rule $(C_.)$: We have to show that $[\![t]\!]\,(l_i) \in [\![T_i]\!]$. By induction we have that $[\![t]\!] \in \Pi_{(X \in I)}.B(X)$. Consequently, $[\![t]\!]\,(l_i) \in B(l_i)$ and $B(l_i) = [\![T_i]\!]$ as required. □

Proposition 13 Soundness of Z_C logic. *If* $\Gamma \vdash_C P$ *then* $[\![\Gamma]\!] \vdash_{zf} [\![P]\!]$

4 Introducing Z

The specification logic Z which we introduce in this section will seem a somewhat impoverished version of the Z one routinely finds in the literature. Our intention is to provide a basic, high-level extension of Z_C which itself may be extended, by further infrastructure, in a variety of ways. It does not seem sensible to us that Z should aim to provide every feature for every conceivable application; particularly when these may be expressed very simply as notational conventions. What we focus on here will be generalisations in which sets may occur in what are type contexts in Z_C, and on the basic operations of the schema calculus.

There remain, nonetheless, a number of particularly important notions that it would be a mistake to leave unexamined. Having provided an interpretation for the specification logic of this section we shall, first, provide a model in Z_C (hence ZF) and then devote an entire section (section 6) to the explanation of the most important derived constructs.

4.1 The language of Z

We first give the proto-syntax for the language of Z which we consider in this paper. Essentially Z extends Z_C by allowing more general forms of propositions, more general forms of sets, and a number of new forms of terms. We shall use the same names for the syntactic categories as we used for Z_C, except for the declarations, since the Z_C category appears as well. In what follows we will always write D_C for the category of Z_C declarations, permitting us to reuse the category name D for the more general Z declarations.

Types are as they were in Z_C.

$$T ::= \mathbb{N} \mid \mathbb{P}\,T \mid T \times T \mid [D_C]$$

The proto-syntax for declarations in Z is, then:

$$D ::= l \in C \mid l \in C;\ D$$

The proto-syntax of propositions:

$$P ::= \bot \mid t = t \mid t \in C \mid \neg P \mid P \vee P \mid \exists z \in C \bullet P$$

The proto-syntax for sets:

$$C ::= S \mid \{z \in C \mid P\} \mid \mathbb{P}\, C \mid C \times C \mid \mathbb{N} \mid [D] \mid \lambda z \in C \bullet t$$

In Z, then, the types appear as a sub-category, or, more precisely, the *carrier sets* of the types do. This sub-category can be formally isolated by means of the following category definitions[8]:

$$D^* ::= l \in T^* \mid l \in T^*;\ D^*$$
$$T^* ::= \mathbb{N} \mid T^* \times T^* \mid \mathbb{P}\, T^* \mid [D^*]$$

Since D^* is just the Z image of the Z_C declarations, we can take all the operations defined over D_C as inducing similar operations over D^*.

Among the set comprehensions we shall, as before, isolate the *schema* as a special case and introduce special metanotation for them:

$$[\, D \mid P \,] =_{df} \{z \in [D] \mid P[\alpha[D]/z.\alpha[D]]\}$$

We have, in Z, schema *expressions*:

$$S ::= [D \mid P] \mid S \vee S \mid S \setminus (l : T) \mid \neg S \mid S[l_1 \leftarrow l_0]$$

Note that all such expressions are included as sets in Z. We only use the unusual notation for renaming in order to prevent confusion with substitution in the meta-language. Schema hiding is, in standard approaches, equivalent to schema existential quantification (*e.g.* [WD96] p. 181). Since components of schema types are, in our approach, labels (constants) it does not make sense to write hiding in terms of a quantifier. In view of the equivalence, however, we suffer no loss of expressivity. We shall have a little more to say about this in section 6.

Disjunction, hiding, negation and renaming are sufficient to permit definitions for conjunction, implication, equivalence, pre-condition, composition and piping to be constructed using the usual definitions. We shall have more to say about this in sections 6 and 9.

The proto-syntax of terms:

$$t ::= x \mid n \mid C \mid t.l \mid (\!| \cdots l_i \Rrightarrow t_i \cdots |\!) \mid t.1 \mid t.2 \mid (t, t) \mid t \upharpoonright T$$
$$\mid \mathit{let}\ x == t\ \mathit{in}\ t \mid (\lambda z \in C \bullet t)t$$

We shall write $t_0 \mapsto t_1$ as a synonym for (t_0, t_1) when the pair is considered as a maplet, that is, as an element of a function.

[8] We have established a notational shift from conventional presentations of Z but, we feel, it is justified. Most particularly, allowing $t : C$ would suggest that we are permitting *sets to be types*, which we are not: such an approach would make typechecking undecidable. Writing $t \in T^*$ on the other hand suggests that we are permitting *types to be sets* which is precisely what Z allows. Additionally, the colon is a judgement of the type assignment and propositionhood system and this *never* assigns anything other than a type (*name*) to a term. Consequently, it would be somewhat confusing to permit more general usage for the type assignment symbol elsewhere in the language. These scruples arise here because we are dealing with logical systems and not simply the language. Perhaps we are being too fastidious, but the strict distinction between the *name* of a type and its *carrier* is well recognised (*e.g.* [Spi92] p. 24). What might be a reasonable abuse of *language* (writing the name T in place of the carrier set T^*), we feel, may not be so easily accommodated as an abuse of *logic* (writing $t : T$ in place of $t \in T^*$ (or $t \in T$)). Indeed we will not have to burden the presentation by distinguishing the name and carrier of a type precisely *because* we will insist on the correct logical relation in type judgements and in membership propositions.

4.2 Type assignment and propositionhood in Z

Definition 14. The judgements of the system again have the following forms:

$$\Gamma \rhd_Z P\ prop$$
$$\Gamma \rhd_Z t : T$$

The contexts, as usual, are sets of type assignments for variables. As before, in giving the rules, we will omit any data which is not changed by a rule.

$$\frac{}{\bot\ prop}\ (Z_\bot) \qquad \frac{t_0 : T \quad t_1 : T}{t_0 = t_1\ prop}\ (Z_=) \qquad \frac{t : T \quad C : \mathbb{P}\, T}{t \in C\ prop}\ (Z_\in)$$

$$\frac{P\ prop}{\neg\ P\ prop}\ (Z_\neg) \qquad \frac{P_0\ prop \quad P_1\ prop}{P_0 \vee P_1\ prop}\ (Z_\vee) \qquad \frac{C : \mathbb{P}\, T \quad z : T \rhd P\ prop}{\exists z \in C \bullet P\ prop}\ (Z_\exists)$$

$$\frac{}{x : T \rhd x : T}\ (Z_x) \qquad \frac{}{0 : \mathbb{N}}\ (Z_0) \qquad \frac{n : \mathbb{N}}{succ\ n : \mathbb{N}}\ (Z_s) \qquad \frac{S_0 : \mathbb{P}\, T_0 \quad S_1 : \mathbb{P}\, T_1}{S_0 \vee S_1 : \mathbb{P}(T_0 \vee T_1)}\ (Z_{\vee S})$$

$$\frac{S : \mathbb{P}\, T}{S \setminus (l : T') : \mathbb{P}\, T \setminus (l : T')}\ (Z_h) \qquad \frac{S : T}{\neg S : T}\ (Z_{\neg S}) \qquad \frac{S : T}{S[l_0 \leftarrow l_1] : T[l_0/l_1]}\ (Z_r)$$

$$\frac{C : \mathbb{P}\, T \quad z : T \rhd P\ prop}{\{z \in C \mid P\} : \mathbb{P}\, T}\ (Z_{\{\}}) \qquad \frac{C : \mathbb{P}\, T}{\mathbb{P}\, C : \mathbb{P}\mathbb{P}\, T}\ (Z_\mathbb{P}) \qquad \frac{C_0 : \mathbb{P}\, T_0 \quad C_1 : \mathbb{P}\, T_1}{C_0 \times C_1 : \mathbb{P}(T_0 \times T_1)}\ (Z_\times)$$

$$\frac{}{\mathbb{N} : \mathbb{P}\mathbb{N}}\ (Z_\mathbb{N}) \qquad \frac{\cdots \quad C : \mathbb{P}\, T \quad \cdots}{[\cdots l \in C \cdots] : \mathbb{P}[\cdots l : T \cdots]}\ (Z_{[]}) \qquad \frac{t : [\cdots l_i : T \cdots]}{t.l_i : T}\ (Z_.)$$

$$\frac{C : \mathbb{P}\, T_0 \quad z : T_0 \rhd t : T_1}{\lambda z \in C \bullet t : \mathbb{P}(T_0 \times T_1)}\ (Z_\lambda) \qquad \frac{\lambda z \in C \bullet t_0 : \mathbb{P}(T_0 \times T_1) \quad t_1 : T_0}{(\lambda z \in C \bullet t_0)\ t_1 : T_1}\ (Z_{ap})$$

$$\frac{\cdots \quad t_i : T_i \quad \cdots}{(\!|\cdots l_i \Rrightarrow t_i \cdots |\!) : [\cdots l_i : T_i \cdots]}\ (Z_\Rrightarrow) \qquad \frac{t : T_0 \times T_1}{t.1 : T_0}\ (Z_1) \qquad \frac{t : T_0 \times T_1}{t.2 : T_1}\ (Z_2)$$

$$\frac{t_0 : T_0 \quad t_1 : T_1}{(t_0, t_1) : T_0 \times T_1}\ (Z_{()}) \qquad \frac{t_0 : T_0 \quad z : T_0 \rhd t_1 : T_1}{let\ x == t_0\ in\ t_1 : T_1}\ (Z_{let})$$

$$\frac{t : T' \quad T \preceq T'}{t \restriction T : T}\ (Z_\restriction)$$

Similar results to those we exhibited for the corresponding system for Z_C hold for the system for Z.

Lemma 15. *The generation lemma for Z holds.* □

Lemma 16.
(*i*) *If x does not appear free in P and $x : T \rhd P\ prop$ then $\rhd P\ prop$*
(*ii*) *If x does not appear free in t and $x : T \rhd t : T'$ then $\rhd t : T'$*
(*iii*) *If $\rhd t : T'$ and x does not appear free in t then $x : T \rhd t : T'$*
(*iv*) *If $\rhd P\ prop$ and x does not appear free in P then $x : T \rhd P\ prop$* □

Typing of terms is also unique in Z.

Lemma 17. *If $t : T_0$ and $t : T_1$ then $T_0 = T_1$*

Proof. By the generation lemma (lemma 15). □

Lemma 18.
(i) If $P[z/t]$ prop and $t : T$ then $z : T \triangleright P$ prop
(ii) If $z : T \triangleright P$ prop and $t : T$ then $P[z/t]$ prop
(iii) If $t'[z/t] : T'$ and $t : T$ then $z : T \triangleright t' : T'$
(iv) If $z : T \triangleright t' : T'$ and $t : T$ then $t'[z/t] : T'$ □

Lemma 19. *$\triangleright T : \mathbb{P}\, T$ for all types T.*

Proof. An easy induction on the structure of types. □

Lemma 20. *The following rule is derivable:*

$$\frac{C : \mathbb{P}\, T' \quad T \preceq T'}{C \restriction \mathbb{P}\, T : \mathbb{P}\, T}\ (Z_{\mathbb{P}\restriction})$$

4.3 A logic for Z

The judgements of the logic have the form:

$$\Gamma \vdash_Z P$$

As before, contexts Γ have the form $\Gamma^-;\ \Gamma^+$, and these are well-formed by means of analogous rules introduced earlier for Z_C.

Definition 21 Logic of Z.

$$\frac{\Gamma \vdash P_0 \quad \Gamma^- \triangleright P_1\ prop}{\Gamma \vdash P_0 \vee P_1}\ (\vee_0^+) \qquad \frac{\Gamma \vdash P_1 \quad \Gamma^- \triangleright P_0\ prop}{\Gamma \vdash P_0 \vee P_1}\ (\vee_1^+) \qquad \frac{P_0 \vee P_1 \quad P_0 \vdash P_2 \quad P_1 \vdash P_2}{P_2}\ (\vee^-)$$

$$\frac{\Gamma, P \vdash \bot \quad \Gamma^- \triangleright P\ prop}{\Gamma \vdash \neg P}\ (\neg^+) \qquad \frac{\neg\neg P}{P}\ (\neg^-) \qquad \frac{P \quad \neg P}{\bot}\ (\bot^+) \qquad \frac{\Gamma \vdash \bot \quad \Gamma^- \triangleright P\ prop}{\Gamma \vdash P}\ (\bot^-)$$

$$\frac{P[z/t] \quad t \in C}{\exists z \in C \bullet P}\ (\exists^+) \qquad \frac{\Gamma \vdash \exists z \in C \bullet P_0 \quad \Gamma^- \triangleright C : \mathbb{P}\, T \quad \Gamma^-, y : T;\ \Gamma^+, P_0[z/y] \vdash P_1}{P_1}\ (\exists^-)$$

$$\frac{\Gamma, P\ context}{\Gamma, P \vdash P}\ (ass) \qquad \frac{\Gamma^- \triangleright t : T}{\Gamma \vdash t = t}\ (ref) \qquad \frac{t' = t}{t = t'}\ (sym) \qquad \frac{t = t' \quad P[z/t]}{P[z/t']}\ (sub)$$

$$\frac{\Gamma^- \triangleright (\!| \cdots l_i \Rrightarrow t_i \cdots |\!) : T}{\Gamma \vdash (\!| \cdots l_i \Rrightarrow t_i \cdots |\!).l_i = t_i}\ (\Rrightarrow^{=}_0) \qquad \frac{\Gamma^- \triangleright t : [\cdots l_i : T_i \cdots]}{\Gamma \vdash (\!| \cdots l_i \Rrightarrow t.l_i \cdots |\!) = t}\ (\Rrightarrow^{=}_1)$$

$$\frac{\Gamma^- \triangleright (t, t') : T}{\Gamma \vdash (t, t').1 = t}\ ((\,)^{=}_0) \qquad \frac{\Gamma^- \triangleright (t, t') : T}{\Gamma \vdash (t, t').2 = t'}\ ((\,)^{=}_1) \qquad \frac{\Gamma^- \vdash t : T \times T'}{\Gamma \vdash (t.1, t.2) = t}\ ((\,)^{=}_2)$$

$$\frac{P[z/t] \quad t \in C}{t \in \{z \in C \mid P\}}\ (\{\}^+) \qquad \frac{t \in \{z \in C \mid P\}}{t \in C}\ (\{\}^-_0) \qquad \frac{t \in \{z \in C \mid P\}}{P[z/t]}\ (\{\}^-_1)$$

$$\frac{t \notin S}{t \in \neg S}\ (\neg S^+) \qquad \frac{t \in \neg S}{t \notin S}\ (\neg S^-) \qquad \frac{t \in S}{t[l/l'] \in S[l \leftarrow l']}\ (S^+_{\leftarrow})$$

$$\frac{t \in S[l \leftarrow l']}{t[l'/l] \in S}\ (S^-_{\leftarrow}) \qquad \frac{\Gamma \vdash t \in S \quad \Gamma^- \triangleright t : T}{\Gamma \vdash t \restriction T \setminus (l : T') \in S \setminus (l : T')}\ (S^+_h)$$

$$\frac{\Gamma \vdash t \in S \setminus (l : T') \quad \Gamma^- \triangleright t : T \setminus (l : T') \quad \Gamma^-, y : T;\ \Gamma^+, y \in S, y \restriction T \setminus (l : T') = t \vdash P}{\Gamma \vdash P}\ (S^-_h)$$

$$\frac{\Gamma \vdash t \restriction T \in S \quad \Gamma^- \triangleright S' : \mathbb{P}\, T' \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \in S \vee S'}\ (S^+_{\vee_0})$$

$$\frac{\Gamma \vdash t \restriction T' \in S' \quad \Gamma^- \triangleright S : \mathbb{P}\, T \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \in S \vee S'}\ (S^+_{\vee_1})$$

$$\frac{t \in S \vee S' \quad t \restriction T \in S \vdash P \quad t \restriction T' \in S' \vdash P \quad t : T \vee T'}{P}\ (S^-_{\vee})$$

$$\frac{\Gamma, z \in C_0 \vdash z \in C_1 \quad \Gamma^- \triangleright z \in C_0\ prop}{\Gamma \vdash C_0 \in \mathbb{P}\, C_1}\ (\mathbb{P}^+) \qquad \frac{C_0 \in \mathbb{P}\, C_1 \quad t \in C_0}{t \in C_1}\ (\mathbb{P}^-)$$

$$\frac{t_0 \in C_0 \quad t_1 \in C_1}{(t_0, t_1) \in C_0 \times C_1}\ (\times^+) \qquad \frac{t \in C_0 \times C_1}{t.1 \in C_0}\ (\times^-_0) \qquad \frac{t \in C_0 \times C_1}{t.2 \in C_1}\ (\times^-_1)$$

$$\frac{}{0 \in \mathbb{N}}\ (\mathbb{N}^+_0) \qquad \frac{n \in \mathbb{N}}{succ\ n \in \mathbb{N}}\ (\mathbb{N}^+_1) \qquad \frac{P[n/0] \quad n : \mathbb{N};\ P \vdash P[n/succ\ n]}{n : \mathbb{N} \vdash P}\ (\mathbb{N}^-)$$

$$\frac{\cdots \quad t_i \in C_i \quad \cdots}{\langle\!|\, \cdots l_i \Rrightarrow t_i \cdots \,|\!\rangle \in [\cdots l_i \in C_i \cdots]}\ ([]^+) \qquad \frac{t \in [\cdots l_i \in C_i \cdots]}{t.l_i \in C_i}\ ([]^-)$$

$$\frac{\Gamma \vdash t_0 \in C \quad \Gamma^- \triangleright C : \mathbb{P}\, T_0 \quad \Gamma^-, z : T_0 \triangleright t_1 : T_1}{\Gamma \vdash t_0 \mapsto t_1[z/t_0] \in \lambda\, z \in C \bullet t_1}\ (\lambda^+)$$

$$\frac{t_0 \in \lambda\, z \in C \bullet t_1}{t_0.1 \in C}\ (\lambda^-_0) \qquad \frac{t_0 \in \lambda\, z \in C \bullet t_1}{t_0.2 = t_1[z/t_0.1]}\ (\lambda^-_1)$$

$$\frac{\Gamma^- \triangleright t_0 : T_0 \quad \Gamma^-, z : T_0 \triangleright t_1 : T_1}{\Gamma \vdash (\lambda\, z \in C \bullet t_0)\ t_1 = t_0[z/t_1]}\ (\lambda^=) \qquad \frac{\Gamma^- \triangleright t : T \quad \Gamma^-, z : T \triangleright t' : T'}{\Gamma \vdash let\ z == t\ in\ t' = t'[z/t]}\ (let)$$

$$\frac{z : T \vdash P_0 \Leftrightarrow P_1 \quad C : \mathbb{P}\, T}{\{z \in C \mid P_0\} = \{z \in C \mid P_1\}}\ (\{\}^=) \qquad \frac{\Gamma \vdash t.l_i = t_i \quad \Gamma^- \triangleright t : T' \quad [\cdots l_i : T_i \cdots] \preceq T'}{\Gamma \vdash (t \restriction [\cdots l_i : T_i \cdots]).l_i = t_i}\ (\restriction^=)$$

4.4 Consequences of the logic for Z

There are, as was the case with Z_C, a large number of congruence rules for equality which are all derivable using the substitution and equivalence rules of equality.

The following are all derivable in the logic for Z.

$$\frac{t_0 = t_1 \quad t_1 = t_2}{t_0 = t_2}\ (trans) \qquad \frac{t_0 = t_2 \quad t_1 = t_3}{(t_0, t_1) = (t_2, t_3)}\ (=_{()})$$

$$\frac{\Gamma \vdash t = t' \quad \Gamma^- \rhd t : T \times T'}{\Gamma \vdash t.1 = t'.1}\ (=_{.1}) \qquad \frac{\Gamma \vdash t = t' \quad \Gamma^- \rhd t : T \times T'}{\Gamma \vdash t.2 = t'.2}\ (=_{.2})$$

$$\frac{\cdots \quad t_i = t_i' \quad \cdots}{(\!| \cdots l_i \Rrightarrow t_i \cdots |\!) = (\!| \cdots l_i \Rrightarrow t_i' \cdots |\!)}\ (=_{\Rrightarrow}) \qquad \frac{\Gamma \vdash t = t' \quad \Gamma^- \rhd t : [\cdots l_i : T_i \cdots]}{\Gamma \vdash t.l_i = t'.l_i}\ (=_{.l})$$

$$\frac{C_0 = C_1 \quad \Gamma^- \rhd \{z \in C_0 \mid P\} : \mathbb{P}\, T}{\{z \in C_0 \mid P\} = \{z \in C_1 \mid P\}}\ (=_{\{\}}) \qquad \frac{C_0 = C_1}{\mathbb{P}\, C_0 = \mathbb{P}\, C_1}\ (=_{\mathbb{P}}) \qquad \frac{C_0 = C_2 \quad C_1 = C_3}{C_0 \times C_1 = C_2 \times C_3}\ (=_{\times})$$

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_0 \vee S_2 = S_1 \vee S_2}\ (sub_{\vee_0}) \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_2 \vee S_0 = S_2 \vee S_1}\ (sub_{\vee_1})$$

$$\frac{S = S'}{S \setminus (l : T) = S' \setminus (l : T)}\ (sub_{\setminus}) \qquad \frac{S = S'}{S[l \leftarrow l'] = S'[l \leftarrow l']}\ (sub_{\leftarrow})$$

$$\frac{S = S'}{\neg S = \neg S'}\ (sub_{\neg})$$

$$\frac{C_0 = C_1 \quad t_0 = t_1}{\lambda z \in C_0 \bullet t_0 = \lambda z \in C_1 \bullet t_1}\ (=_{\lambda}) \qquad \frac{\Gamma \vdash t_0 = t_2 \quad \Gamma^- \rhd t_0 : T \quad \Gamma \vdash x : T \vdash t_1 = t_3}{\Gamma \vdash let\ x == t_0\ in\ t_1 = let\ x == t_2\ in\ t_3}\ (=_{let})$$

$$\frac{\Gamma \vdash \lambda z \in C_0 \bullet t_0 = \lambda z \in C_1 \bullet t_1 \quad \Gamma^- \rhd t : T \quad \Gamma^- \rhd C_0 : \mathbb{P}\, T}{\Gamma \vdash (\lambda z \in C_0 \bullet t_0)\ t = (\lambda z \in C_1 \bullet t_1)\ t}\ (=_{app_0})$$

$$\frac{\Gamma \vdash t_0 = t_1 \quad \Gamma^- \rhd t_0 : T \quad \Gamma^- \rhd C : \mathbb{P}\, T}{\Gamma \vdash (\lambda z \in C \bullet t)\ t_0 = (\lambda z \in C \bullet t)\ t_1}\ (=_{app_1})$$

$$\frac{\Gamma \vdash t = t' \quad \Gamma^- \rhd t : T' \quad T \preceq T'}{\Gamma \vdash t \restriction T = t' \restriction T}\ (=_{\restriction})$$

Set equality in Z is, like Z_C, extensional. The necessary rules are also part of the logic for Z.

Lemma 22. $C_{\mathbb{P}\, T} \subseteq C'_{\mathbb{P}\, T} \subseteq C_{\mathbb{P}\, T} \Leftrightarrow C = C'$ □

The rule which relates filtered terms and filtered sets in Z_C also generalises to Z.

Lemma 23.

$$\frac{\Gamma \vdash t \in C \quad \Gamma^- \rhd t : T' \quad T \preceq T'}{\Gamma \vdash t \restriction T \in C \restriction \mathbb{P}\, T}\ (\in_{\restriction})$$

Proof. By rules $(Z_{\restriction})$, $(\{\}^+)$ and $(\exists^+)$. □

Lemma 24. *The following rule is admissible:*

$$\frac{t : T}{t \in T}$$

Proof. By induction on the structure of the term t. For example:
Ad $t.1$:
We may assume that $t.1 : T$ for some type T. By lemma 15 we have $t : T \times T'$ for some type T'. *ex hypothesi* we then obtain $t \in T \times T'$ and then $t.1 \in T$, by rule $(\times_0^-)$, as required. □

Two simple observations that we shall require later:

Lemma 25. $\Gamma \vdash P[\alpha T/t \upharpoonright T.\alpha T] \Leftrightarrow P[\alpha T/t.\alpha T]$

Proof. This is a simple extension of the fact that $t \upharpoonright T.l = t.l$ for any label l in the alphabet of T. □

Lemma 26. *If* $T \preceq T'$ *and no label in* $T' \setminus T$ *occurs in* P *then:* $\Gamma \vdash P[\alpha T/t.\alpha T] \Leftrightarrow P[\alpha T'/t.\alpha T]$ □

The rule (S_h^-) specialises into two useful derived rules:

$$\frac{t \in [D \mid P] \setminus (l : T)}{t \in [D] \setminus (l : T)}\ (S_{h_0}^-) \qquad \frac{t \in [D \mid P] \setminus (l : T)}{\exists z : T \bullet P[\alpha([D] \setminus (l : T))/t.\alpha([D] \setminus (l : T))][l/z]}\ (S_{h_1}^-)$$

Schema were introduced in Z, as in Z_C, by convention. This induces immediately the following rules:

$$\frac{P[\alpha[D]/t.\alpha[D]] \quad t \in [D]}{t \in [D \mid P]}\ (S^+) \qquad \frac{t \in [D \mid P]}{P[\alpha[D]/t.\alpha[D]]}\ (S_0^-) \qquad \frac{t \in [D \mid P]}{t \in [D]}\ (S_1^-)$$

It is then possible to prove those relationships which are commonly used to describe (occasionally to define) the schema operators in the literature (*e.g.* for negation see [WD96] p. 176, for disjunction *ibid.* p. 174, and for hiding *ibid.* p. 178). This begins the task of establishing an equational logic for Z which is justified by the logic. The equations all have premises which ensure that the equalities are well-formed.

Lemma 27.

$$\frac{\Gamma^- \triangleright [D^* \mid P] : \mathbb{P}[D^*]}{\Gamma \vdash \neg[D^* \mid P] = [D^* \mid \neg P]}\ (\neg^=)$$

Proof. Note that the declarations must range over types. It is well known that this equation fails if the declarations range over sets in general[9]. Assume that $\Gamma^- \triangleright [D^* \mid P] : \mathbb{P}[D^*]$. This implies that the equation is a proposition and that both sides have the type $\mathbb{P}[D^*]$.
Ad $(\subseteq)$
Let $t : [D^*]$. Suppose that $t \in \neg[D^* \mid P]$. Using rule $(\neg S^-)$ this is $t \notin [D^* \mid P]$, or equivalently $\neg(t \in [D^*] \wedge P[\alpha[D^*]/t.\alpha[D^*]])$. Using De Morgan's law, this is just $t \notin [D^*] \vee \neg P[\alpha[D^*]/t.\alpha[D^*]]$. Assume that $t \notin [D^*]$. Using lemma 24 we obtain $t \in [D^*]$ from the assumption and hence we conclude that $\bot$, whence, by rule $(\bot^-)$, $\neg P[\alpha[D^*]/t.\alpha[D^*]]$. This now also follows by rule $(\vee^-)$ from the disjunction above. From this, and $t \in [D^*]$ we finally conclude, by rule (S^+), that $t \in [D^* \mid \neg P]$ as required.
Ad $(\supseteq)$
Let $t : [D^*]$. Suppose that $t \in [D^* \mid \neg P]$. Using rules (S_0^-) and (S_1^-) we obtain: $\neg P[\alpha[D^*]/t.\alpha[D^*]]$ and $t \in [D^*]$. By rule $(\{\}^+)$ and propositional logic we then conclude that $t \notin [D^* \mid P]$ which, by rule $(\neg S^+)$ is $t \in \neg[D^* \mid P]$ as required. □

Lemma 28.

$$\frac{\Gamma^- \triangleright [D_0^* \mid P_0] : \mathbb{P}\, T_0 \quad \Gamma^- \triangleright [D_1^* \mid P_1] : \mathbb{P}\, T_1}{\Gamma \vdash [D_0^* \mid P_0] \vee [D_1^* \mid P_1] = [D_0^* \vee D_1^* \mid P_0 \vee P_1]}\ (\vee^=)$$

[9] This is expressed rather differently in the literature because the equation is often used to informally *define* the schema operations. It is expressed in the regime of [WD96] (p. 176) by indicating that the *transformation* of a negated schema only applies to *normalised* schema, that is, when the declaration part has been reduced to its unique canonical form (*ibid.* p. 159).

Proof. We write D for $D_0^* \vee D_1^*$ in what follows. *Ad* ($\subseteq$):
We proceed by rule ($S_\vee^-$). Using rules (S_0^-), (S_1^-) and lemmata 25 and 26 on each assumption we obtain $P_0[\alpha[D]/t.\alpha[D]]$, $P_1[\alpha[D]/t.\alpha[D]]$, $t \upharpoonright [D_0^*] \in [D_0^*]$ and $t \upharpoonright [D_1^*] \in [D_1^*]$. From the latter pair, using ($S_\wedge^+$) we have $t \in [D_0^* \vee D_1^*]$. From the former pair we obtain $(P_0 \vee P_1)[\alpha[D]/t.\alpha[D]]$, by rules ($\vee_0^+$) and ($\vee_1^+$) discharging the assumptions. It remains to apply rule (S^+) to this data to conclude that $t \in [D_0^* \vee D_1^* \mid P_0 \vee P_1]$ as required.
Ad ($\supseteq$):
Suppose that $t \in [D \mid P_0 \vee P_1]$. Using rules ($S_0^-$), ($S_1^-$), ($\in_\upharpoonright$) (twice: using the facts that $D_0^* \preceq D$ and $D_1^* \preceq D$) we obtain: $(P_0 \vee P_1)[\alpha[D]/t.\alpha[D]]$, $t \upharpoonright D_0^* \in [D_0^*]$ and $t \upharpoonright D_1^* \in [D_1^*]$. Using rule ($\vee^-$) on the disjunction above, we use lemma 25 and lemma 26 on each assumption to obtain $P_0[\alpha[D_0^*]/t \upharpoonright [D_0^*].\alpha[D_0^*]]$ and $P_1[\alpha[D_1^*]/t \upharpoonright [D_1^*].\alpha[D_1^*]]$. Combining these with the data above we have, by rule (S^+) (twice) $t \upharpoonright D_0^* \in [D_0^* \mid P_0]$ and $t \upharpoonright D_0^* \in [D_0^* \mid P_0]$. Using rules ($S_{\vee_0}^+$) and ($S_{\vee_1}^+$) we conclude, discharging the assumptions, that $t \in [D_0^* \mid P_0] \vee [D_1^* \mid P_1]$ as required. □

In the case of hiding there is a minor notational variation because, in our framework, labels are constants:

Lemma 29.

$$\frac{\Gamma^- \triangleright [D^* \mid P] : \mathbb{P}\, T'}{\Gamma \vdash [D^* \mid P] \setminus (l : T) = [D^* \setminus (l : T) \mid \exists z : T \bullet P[l/z]]} \ (\setminus^=)$$

Proof. *Ad* ($\subseteq$):
Let $t \in [D^* \mid P] \setminus (l : T)$. By rules ($S_{h_0}^-$) and ($S_{h_1}^-$) we obtain $t : [D^*]/\mathit{hide}(l : T)$ and $\exists z : T \bullet P[\alpha([D^*] \setminus (l : T))/t.\alpha([D^*] \setminus (l : T))][l/z]$. Then, by rule ($S^+$) we have $t \in [D \setminus (l : T) \mid P[l/z]]$ as required.
Ad ($\supseteq$):
Let $t \in [D^* \setminus (l : T) \mid \exists z : T \bullet P[l/z]]$. Using rules ($S_0^-$) and ($S_1^-$) we obtain $t \in [D^*] \setminus (l : T)$ and $\exists z : T \bullet P[l/z][\alpha([D^*] \setminus (l : T))/t.\alpha([D^*] \setminus (l : T))]$. Proceeding by rule ($\exists^-$) we may assume, for some $y : T$ that $P[l/y][\alpha([D^*] \setminus (l : T))/t.\alpha([D^*] \setminus (l : T))]$. Let t' be the unique term such that $t = t' \upharpoonright [D^*] \setminus (l : T)$ and $t'.l = y$. Then from the data above we may infer that $t' \in [D^*]$ and that $P[\alpha[D^*]/\alpha[D^*]$ whence, by rule (S^+) we have $t' \in [D^* \mid P]$. By rule (S_h^+) this allows us to conclude that $t' \upharpoonright [D^*] \setminus (l : T) \in [D^* \mid P] \setminus (l : T)$ which is, by definition, $t \in [D^* \mid P] \setminus (l : T)$ as required. □

In addition we have an equation relating general declarations over sets to declarations over types. This, by iteration, enables us to remove all non-type sets from the declarations of Z schema in the equational logic.

Lemma 30.

$$\frac{\Gamma^- \triangleright [D;\ l \in C \mid P] : \mathbb{P}\, T' \quad \Gamma^- \triangleright C : \mathbb{P}\, T}{\Gamma \vdash [D;\ l \in C \mid P] = [D;\ l \in T \mid l \in C \wedge P]} \ (\in^=)$$

Proof. Let us write D' for the declaration $D;\ l \in C$. *Ad* ($\subseteq$):
By rules (S_0^-) and (S_1^-) we may obtain: $P[\alpha[D']/t.\alpha[D']]$ and $t \in [D']$. From the latter, by rule ($[]^-$), $t.l \in C$ and, using the assumption, $t.l : T$. By lemma 24 we have $t.l \in T$ and then, by rule ($[]^+$) $t \in [D;\ l \in T]$. $t.l \in C$ is equivalently $(l \in C)[\alpha[D']/t.\alpha[D']$ hence, combining this by rule (wedge^+) with the former, gives $(P \wedge l \in C)[\alpha[D']/t.\alpha[D']]$, from which we conclude, by rule (S^+), that $t \in [D;\ l \in T \mid l \in C \wedge P]$ as required.
Ad ($\supseteq$):
By rule (S_0^-) we may obtain: $(P \wedge l \in C)[\alpha[D;\ l \in T]/t.\alpha[D;\ l \in T]]$. From this we obtain, by rule ($\wedge_0^-$) and ($\wedge_1^-$), $P[\alpha[D;\ l \in T]/t.\alpha[D;\ l \in T]]$ and $t.l \in C$. By rule (S_1^-) we also have $t \in [D;\ l \in T$. Using rules ($[]^-$) and ($[]^+$) with this data we conclude that $t \in [D;\ l \in C]$. But then, using rule (S^+) we conclude that $t \in [D;\ l \in C \mid P]$ as required. □

4.5 Syntactic consistency for Z

We must, of course, ensure that the rules of the logic are syntactically consistent.

Proposition 31. *If $\Gamma \vdash_Z P$ when Γ context then $\Gamma^- \triangleright_Z P$ prop*

Proof. By induction on the structure of the derivation $\Gamma \vdash P$. Suppose that Γ *context*.
Ad Rules $(\vee_0^+)$, $(\vee_1^+)$, $(\vee^-)$, $(\neg^+)$, $(\neg^-)$, $(\bot^+)$, $(\bot^-)$, (ass), (ref), (sym), (sub), $(\Rrightarrow_0^=)$, $(\Rrightarrow_1^=)$, $((\,)_0^=)$, $((\,)_1^=)$, $((\,)_2^=)$, $(\mathbb{N}^-)$, $(\restriction^=)$:
These are also rules of Z_C and have already been dealt with in proposition 11. Note that this also relies on the observation that the rules for propositionhood for disjunction *etc.* in Z are also rules of Z_C.
Ad Rule $(\exists^+)$:
Ex hypothesi we have $\Gamma^- \rhd t \in C$ *prop* and then, by lemma 15, we may infer that $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd C : \mathbb{P}\, T$ for some T. Using the former and the fact that $\Gamma^- \rhd P[z/t]$ *prop* holds *ex hypothesi* we conclude that $\Gamma^-, z : T \rhd P$ *prop* by lemma 18(i). Combining this with the latter typing judgement, we obtain $\Gamma^- \rhd \exists\, z \in C \bullet P$ *prop* by rule $(Z_\exists)$ as required.
Ad Rule $(\exists^-)$:
We may assume *ex hypothesi* from the first premise that $\Gamma^- \rhd \exists\, z \in C \bullet P_0$ *prop*. Using lemma 15 we may infer that $\Gamma^-, z : T' \rhd P_0$ *prop* and $\Gamma^- \rhd C : \mathbb{P}\, T'$ for some type T'. The latter, the second premise, and lemma 17 imply that $T' = T$. From the former we may conclude that $\Gamma^-, z : T;\ \Gamma^+, P_0$ *context* and then, by α-conversion, that $\Gamma^-, y : T;\ \Gamma^+, P_0[z/y]$ *context*. This fact allows us to infer that $\Gamma^-, y : T \rhd P_1$ *prop ex hypothesi*. But since y is not free in P_1 this reduces to $\Gamma^- \rhd P_1$ *prop* by lemma 16(i).
Ad Rule $(\{\}^+)$:
From the two premises we obtain, *ex hypothesi* $\Gamma^- \rhd P[z/t]$ *prop* and $\Gamma^- \rhd t \in C$ *prop*. From the latter we obtain, using lemma 15 $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd C : \mathbb{P}\, T$ for some T. Combining $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd P[z/t]$ *prop* using lemma 18(i) we infer that $\Gamma^-, z : T \rhd P$ *prop* and this together with $\Gamma^- \rhd C : \mathbb{P}\, T$ gives us $\Gamma^- \rhd \{z \in C \mid P\} : \mathbb{P}\, T$ by rule $(Z_{\{\}})$. This and $\Gamma^- \rhd t : T$ imply $\Gamma^- \rhd t \in \{z \in C \mid P\}$ *prop* as required, by rule $(Z_\in)$.
Ad Rule $(\{\}_0^-)$:
From the premise we obtain $\Gamma^- \rhd t \in \{z \in C \mid P\}$ *prop ex hypothesi*, and two applications of lemma 15 yield $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd C : \mathbb{P}\, T$ for some T. From these we may infer that $\Gamma^- \rhd t \in C$ *prop* as required, by rule $(Z_\in)$.
Ad Rule $(\{\}_1^-)$:
From the premise we obtain $\Gamma^- \rhd t \in \{z \in C \mid P\}$ *prop ex hypothesi*, and two applications of lemma 15 yield $\Gamma^- \rhd t : T$ and $\Gamma^-, z : T \rhd P$ *prop*. These combine using lemma 18(ii), to show that $\Gamma^- \rhd P[z/t]$ *prop* as required.
Ad Rule $(\neg S^+)$:
We have *ex hypothesi* that $\Gamma^- \rhd t \notin S$ *prop* from which, by lemma 15 we obtain $\Gamma^- \rhd t \in S$ *prop*. Using lemma 15 again we may infer that $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd S : \mathbb{P}\, T$ for some T. From the latter, using rule $(Z_{\neg S})$, we obtain $\Gamma^- \rhd \neg S : \mathbb{P}\, T$ and, combining this with the former, by rule $(Z_\in)$, we conclude that $\Gamma^- \rhd t \in \neg S$ *prop* as required.
Ad Rule $(\neg S^-)$:
We have *ex hypothesi* that $\Gamma^- \rhd t \in \neg S$ *prop* from which, by lemma 15 we obtain $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd \neg S : \mathbb{P}\, T$ for some T. Using lemma 15 once again we may infer from the latter that $\Gamma^- \rhd S : \mathbb{P}\, T$. Combing this with the former, by rule $(Z_\in)$, we conclude that $\Gamma^- \rhd t \in S$ *prop* as required.
Ad Rule $(S_{\leftarrow}^+)$:
We have *ex hypothesi* that $\Gamma^- \rhd t \in S$ *prop* from which, by lemma 15, we obtain both $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd S \in \mathbb{P}\, T$ for some T.
Ad Rule $(S_{\leftarrow}^-)$:
We have *ex hypothesi* that $\Gamma^- \rhd t \in S[l \leftarrow l']$ *prop*, whence, by lemma 15, $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd S[l \leftarrow l'] : \mathbb{P}\, T$ for some T. Using lemma 15 again we have $\Gamma^- \rhd S : \mathbb{P}\, T'$ for some T' which forces $T = T'[l/l']$. Then we obtain $\Gamma^- \rhd t[l/l'] : T'$ and finally, by rule $(Z_\in)$, $\Gamma^- \rhd t[l/l'] \in S$ *prop* as required.
Ad Rule (S_h^+):
We have *ex hypothesi* that $\Gamma^- \rhd t \in S$ *prop* and, given the assumption $\Gamma^- \rhd t : T$, we obtain by lemma 15, $\Gamma^- \rhd S : \mathbb{P}\, T$ whence, by rule (Z_h), $\Gamma^- \rhd S \setminus (l : T') : \mathbb{P}\, T \setminus (l : T')$. From the assumption we have, by rule $(Z_\restriction)$, $\Gamma^- \rhd t \restriction T \setminus (l : T') : T \setminus (l : T')$. Putting these together, using rule $(Z_\in)$, we conclude that $\Gamma^- \rhd t \restriction T \setminus (l : T') \in S \setminus (l : T')$ *prop* as required.
Ad Rule (S_h^-):
We have from the first premise *ex hypothesi* that $\Gamma^- \rhd t \in S \setminus (l : T')$. Using lemma 15 (twice), lemma 17 and the second premise, we may infer that $\Gamma^- \rhd S : \mathbb{P}\, T$, hence, by rule $(Z_\in)$, $\Gamma^-, y : T \rhd y \in S$ *prop* and, by rules $(Z_\restriction)$, $(Z_=)$ and the fact that $T \setminus (l : T') \preceq T$, $\Gamma^-, y : T \rhd y \restriction T \setminus (l : T') = t$ *prop*. Hence $\Gamma^-, y : T;\ \Gamma^+, y \in S, y \restriction T \setminus (l : T') = t$ *context* so from the final premise, *ex hypothesi*, we obtain

$\Gamma^- \triangleright P$ *prop* as required.
Ad Rule $(S^+_{\vee_0})$:
We have from the first premise *ex hypothesi* that $\Gamma^- \triangleright t \restriction T \in S$ *prop* from which, by lemma 15, rule $(Z_\restriction)$ and the assumption that $\Gamma^- \triangleright t : T \vee T'$, we obtain $\Gamma^- \triangleright t \restriction T : T$ and $\Gamma^- \triangleright S : \mathbb{P}\, T$. From the latter, and the second premise, we infer that $\Gamma^- \triangleright S \vee S' : \mathbb{P}(T \vee T')$ by rule $(Z_{S_\vee})$. From this data, by rule $(Z_\in)$, we conclude that $\Gamma^- \triangleright t \in S \vee S'$ *prop* as required.
Ad Rule $(S^+_{\vee_1})$:
Similarly.
Ad Rule $(S^-_\vee)$:
We have from the first premise *ex hypothesi* that $\Gamma^- \triangleright t \in S \vee S'$ *prop* and from this, using lemma 15 and the assumption that $\Gamma^- \triangleright t : T \vee T'$, we obtain $\Gamma^- \triangleright S \vee S' : \mathbb{P}(T \vee T')$ and then using lemma 15 once again, we have both $\Gamma^- \triangleright S : \mathbb{P}\, T$ and $\Gamma^- \triangleright S' : \mathbb{P}\, T'$. Since both $T \preceq T \vee T'$ and $T' \preceq T \vee T'$ it follows, by rule $(Z_\restriction)$, that $\Gamma^- \triangleright t \restriction T : T$ and $\Gamma^- \triangleright t \restriction T' : T'$. From all these, using rule $(Z_\in)$, we have $\Gamma^- \triangleright t \restriction T \in S$ *prop* and $\Gamma^- \triangleright t \restriction T' \in S'$ *prop* which are sufficient to show that $\Gamma, t \restriction T \in S$ *context* and $\Gamma, t \restriction T' \in S'$ *context*. These are what we require to discharge the antecedents of the implications we obtain *ex hypothesi* from the second and third premises, each of which permits us to conclude that $\Gamma^- \triangleright P$ *prop* as required.
Ad Rule $(\mathbb{P}^+)$:
From the second premise we obtain $\Gamma, t \in C_0$ *context* and then from the first premise *ex hypothesi* we may conclude that $\Gamma^- \triangleright t \in C_1$ *prop*. Using lemma 15 we obtain $\Gamma^- \triangleright C_0 : \mathbb{P}\, T$ and $\Gamma^- \triangleright t : T$ for some T from the second premise. Using lemma 15 and $\Gamma^- \triangleright t \in C_1$ *prop* we obtain $\Gamma^- \triangleright t : T'$ and $\Gamma^- \triangleright C_1 : \mathbb{P}\, T'$ for some T'. But we may conclude that $T = T'$ from $\Gamma^- \triangleright t : T'$ and $\Gamma^- \triangleright t : T$ using lemma 17. From $\Gamma^- \triangleright C_1 : \mathbb{P}\, T$ we have $\Gamma^- \triangleright \mathbb{P}\, C_1 : \mathbb{P}\,\mathbb{P}\, T$ by rule $(Z_\mathbb{P})$ and then, from this, and $\Gamma^- \triangleright C_0 : \mathbb{P}\, T$ we get $\Gamma^- \triangleright C_0 \in \mathbb{P}\, C_1$ *prop* by rule $(Z_\in)$ as required.
Ad Rule $(\mathbb{P}^-)$:
From the two premises we obtain, *ex hypothesi* $\Gamma^- \triangleright C_0 \in \mathbb{P}\, C_1$ *prop* and $\Gamma^- \triangleright t \in C_0$ *prop*. Using lemma 15 these yield $\Gamma^- \triangleright C_0 : T$, $\Gamma^- \triangleright \mathbb{P}\, C_1 : \mathbb{P}\, T$, $\Gamma^- \triangleright t : T'$ and $\Gamma^- \triangleright C_0 : \mathbb{P}\, T'$ for types T and T'. By lemma 17 we have $T = \mathbb{P}\, T'$ hence, in particular, we have $\Gamma^- \triangleright t : \mathbb{P}\, T'$ and $\Gamma^- \triangleright C_1 : \mathbb{P}\,\mathbb{P}\, T$. From this we obtain $\Gamma^- \triangleright t \in C_1$ *prop*, by rule $(Z_\in)$, as required.
Ad Rule $(\times^+)$:
From the premises we obtain *ex hypothesi* $\Gamma^- \triangleright t_0 \in C_0$ *prop* and $\Gamma^- \triangleright t_1 \in C_1$ *prop*. By lemma 15 we obtain $\Gamma^- \triangleright t_0 : T_0$, $\Gamma^- \triangleright C_0 : \mathbb{P}\, T_0$, $\Gamma^- \triangleright t_1 : T_1$ and $\Gamma^- \triangleright C_1 : \mathbb{P}\, T_1$ for types T_0 and T_1. The first and third of these yield $\Gamma^- \triangleright (t_0, t_1) : T_0 \times T_1$ by rule $(Z_{()})$, and the second and fourth yield $\Gamma^- \triangleright C_0 \times C_1 : \mathbb{P}(T_0 \times T_1)$ by rule $(Z_\times)$. From these we may conclude that $\Gamma^- \triangleright (t_0, t_1) \in C_0 \times C_1$ *prop*, by rule $(Z_\in)$, as required.
Ad Rule $(\times^-_0)$:
From the premise we have *ex hypothesi* $\Gamma^- \triangleright t \in C_0 \times C_1$ *prop* and then by lemma 15 we have both $\Gamma^- \triangleright t : T$ and $\Gamma^- \triangleright C_0 \times C_1 : \mathbb{P}\, T$ for some T. Using lemma 15 on the second of these we may conclude that $\Gamma^- \triangleright C_0 : \mathbb{P}\, T_0$ and $\Gamma^- \triangleright C_1 : \mathbb{P}\, T_1$ for some T_0 and T_1. This forces $T = T_0 \times T_1$, hence $\Gamma^- \triangleright t : T_0 \times T_1$. From this we have $\Gamma^- \triangleright t.1 : T_0$ by rule (Z_1) and this together with $\Gamma^- \triangleright C_0 : \mathbb{P}\, T_0$ yields $\Gamma^- \triangleright t.1 \in C_0$ *prop*, by rule $(Z_\in)$, as required.
Ad Rule $(\times^-_1)$:
Similarly.
Ad Rule $(\mathbb{N}^+_0)$:
We have to show that $\Gamma^- \triangleright 0 \in \mathbb{N}$ *prop*. But this follows from the axioms (Z_0) and $(Z_\mathbb{N})$ by rule $(Z_\in)$.
Ad Rule $(\mathbb{N}^+_1)$:
From the premise we have *ex hypothesi* $\Gamma^- \triangleright n \in \mathbb{N}$ *prop* whence, by lemma 15 $\Gamma^- \triangleright n : T$ and $\Gamma^- \triangleright \mathbb{N} : \mathbb{P}\, T$ for some T. By axiom $(Z_\mathbb{N})$ we have $T = \mathbb{N}$ and therefore, by rule (Z_s) $\Gamma^- \triangleright succ\ n : \mathbb{N}$. Combining this with $\Gamma^- \triangleright \mathbb{N} : \mathbb{P}\,\mathbb{N}$ we conclude that $\Gamma^- \triangleright succ\ n \in \mathbb{N}$, by rule $(Z_\in)$ as required.
Ad Rule $([]^+)$:
Let us write T for the schema type $[\cdots l_i : T_i \cdots]$. *Ex hypothesi* we have both $\Gamma^- \triangleright t_i : T_i$ and $\Gamma^- \triangleright C_i : \mathbb{P}\, T_i$ for all i. From the former, using rule $(Z_\Rightarrow)$, we obtain $\Gamma^- \triangleright (\!| \cdots l_i \Rightarrow t_i \cdots |\!) : T$ and then, from the latter, by rule $(Z_{[]})$, that $\Gamma^- \triangleright [\cdots l_i \in C_i \cdots] : \mathbb{P}\, T$. Hence, by rule $(Z_\in)$ we conclude that $\Gamma^- \triangleright (\!| \cdots l_i \Rightarrow t_i \cdots |\!) \in [\cdots l_i \in C_i \cdots]$ *prop* as required.
Ad Rule $([]^-)$:
From the premise we obtain *ex hypothesi* $\Gamma^- \triangleright t \in [\cdots l_i \in C_i \cdots]$ *prop* whence, by lemma 15 $\Gamma^- \triangleright t : T$

and $\Gamma^- \rhd [\cdots l_i \in C_i \cdots] : \mathbb{P}\, T$ for some T. Using lemma 15 once again on the second of these we have $\Gamma^- \rhd C_i : \mathbb{P}\, T_i$ where $T = [\cdots l_i : T_i \cdots]$. But then $\Gamma^- \rhd t : [\cdots l_i : T_i \cdots]$ which, by rule $(Z_.)$, implies that $\Gamma^- \rhd t.l_i : T_i$. Combining this with $\Gamma^- \rhd C_i : \mathbb{P}\, T_i$ yields $\Gamma^- \rhd t.l_i \in \mathbb{P}\, C_i$, by rule $(Z_\in)$ as required.
Ad Rule (λ^+):
We have *ex hypothesi* that $\Gamma^- \rhd t_0 \in C$ *prop* and using lemma 15 we obtain $\Gamma^- \rhd t : T$ and $\Gamma^- \rhd C : \mathbb{P}\, T$ for some T.
Ad Rule (λ_0^-):
We have *ex hypothesi* that $\Gamma^- \rhd t_0 \in \lambda\, z \in C \bullet t_1$ *prop* and using lemma 15 we obtain $\Gamma^- \rhd t_0 : T$ and $\Gamma^- \rhd \lambda\, z \in C \bullet t_1 : \mathbb{P}\, T$ for some T. But from the latter, using lemma 15 once again, we infer that $\Gamma^- \rhd C : \mathbb{P}\, T_0$ and $\Gamma^-, z : T_0 \rhd t_1 : T_1$ for types T_0 and T_1 which forces $T = T_0 \times T_1$. But from $\Gamma^- \rhd t_0 : T_0 \times T_1$ we may infer, by rule (Z_1), that $\Gamma^- \rhd t_0.1 : T_0$. Combining this with the type assignment for C derived above, using rule $(Z_\in)$, we conclude that $\Gamma^- \rhd t_0.1 \in C$ *prop* as required.
Ad Rule (λ_1^-):
We have *ex hypothesi* that $\Gamma^- \rhd t_0 \in \lambda\, z \in C \bullet t_1$ *prop* and using lemma 15 we obtain $\Gamma^- \rhd t_0 : T$ and $\Gamma^- \rhd \lambda\, z \in C \bullet t_1 : \mathbb{P}\, T$ for some T. But from the latter, using lemma 15 once again, we infer that $\Gamma^- \rhd C : \mathbb{P}\, T_0$ and $\Gamma^-, z : T_0 \rhd t_1 : T_1$ for types T_0 and T_1 which forces $T = T_0 \times T_1$. But from $\Gamma^- \rhd t_0 : T_0 \times T_1$ we may infer, by rule (Z_2), that $\Gamma^- \rhd t_0.2 : T_1$ and by rule (Z_1) that $\Gamma^- \rhd t_0.1 : T_0$. The latter, together with $\Gamma^-, z : T_0 \rhd t_1 : T_1$, permits us to conclude, by lemma 18(iv), that $\Gamma^- \rhd t_1[z/t_0.1] : T_1$ whence, by rule $(Z_=)$ we conclude that $\Gamma^- \rhd t_0.2 = t_1[z/t_0.1]$ *prop* as required.
Ad Rule $(=_\lambda)$:
From the first premise we have, by rule (Z_λ), that $\Gamma^- \rhd \lambda\, x \in C \bullet t_0 : \mathbb{P}(T_0 \times T_1)$, and from this and the second premise, by rule (Z_{ap}), that $\Gamma^- \rhd (\lambda\, x \in C \bullet t_0)t_1 : T_1$. But both premises imply that $\Gamma^- \rhd t_0[z/t_1] : T_1$ by lemma 18(iv), hence we may conclude that $\Gamma^- \rhd (\lambda\, x \in C \bullet t_0)t_1 = t_0[z/t_1]$ *prop*, by rule $(Z_\in)$, as required.
Ad Rule (*let*):
From the assumptions that $\Gamma^- \rhd t_0 : T_0$ and $\Gamma^-, z : T_0 \rhd t_1 : T_1$ we may infer, by rule (Z_{let}), that $\Gamma^- \rhd$ *let* $x == t_0$ *in* $t_1 : T_1$. In addition, combining the assumptions using lemma 18(iv), we obtain $\Gamma^- \rhd t_1[z/t_0] : T_1$ and then, by rule $(Z_\in)$, we conclude that $\Gamma^- \rhd$ *let* $x == t_0$ *in* $t_1 = t_1[z/t_0]$ *prop* as required.
Ad Rule $(\{\}^=)$:
From the first premise we obtain $\Gamma^-, z : T \rhd P_0 \Leftrightarrow P_1$ *prop ex hypothesi*, and then we have both $\Gamma^-, z : T \rhd P_0$ *prop* and $\Gamma^-, z : T \rhd P_1$ *prop* by lemma 15. These, together with the second premise, imply $\Gamma^- \rhd \{z \in C \mid P_0\} : \mathbb{P}\, T$ and $\Gamma^- \rhd \{z \in C \mid P_1\} : \mathbb{P}\, T$ by rule $(Z_{\{\}})$. From these, by rule $(Z_=)$, we obtain $\Gamma^- \rhd \{z \in C \mid P_0\} = \{z \in C \mid P_1\}$ *prop* as required. □

5 An interpretation of Z in Z_C

In this section we describe a translation $[\![\ \_\ ]\!]_Z$ of Z, as described above, into Z_C, our core specification logic. This translation (unlike normalisation processes for Z which are found in the literature) is compositional. This can be achieved because we have made precise the notions of propositionhood and type assignment; indeed, since well-formed Z sentences are those which satisfy the rules of definition 14, we shall make use, where necessary, the type information associated with Z terms. As before, we omit the subscript on the translation function unless it is essential.

5.1 Propositions

The language of Z propositions is only marginally more complicated than that of Z_C. The translation is, for the main part, transparent:

$$
\begin{array}{lll}
(i) & [\![\bot]\!] & =_{df} \bot \\
(ii) & [\![t_0 = t_1]\!] & =_{df} [\![t_0]\!] = [\![t_1]\!] \\
(iii) & [\![t \in C]\!] & =_{df} [\![t]\!] \in [\![C]\!] \\
(iv) & [\![\neg P]\!] & =_{df} \neg [\![P]\!] \\
(v) & [\![P_0 \vee P_1]\!] & =_{df} [\![P_0]\!] \vee [\![P_1]\!] \\
(vi) & [\![\exists\, z \in (C_{\mathbb{P}\, T}) \bullet P]\!] & =_{df} \exists\, z : T \bullet z \in [\![C]\!] \wedge [\![P]\!]
\end{array}
$$

5.2 The schema calculus

The three basic schema calculus operations can be defined in Z_C as follows:

$$
\begin{array}{lll}
(i) & \neg_T C & =_{df} \{z : T \mid z \notin C\} \\
(ii) & C_0 \vee_{(T_0,T_1)} C_1 & =_{df} \{z : T_0 \vee T_1 \mid z \upharpoonright T_0 \in C_0 \vee z \upharpoonright T_1 \in C_1\} \\
(iii) & C \setminus_T (l : T') & =_{df} \{z : T \setminus (l : T') \mid \exists x : T \bullet x \in C \wedge z = x \upharpoonright T \setminus (l : T')\}
\end{array}
$$

With these in place it is possible to translate Z schema calculus expressions into Z_C. Note that we only interpret type-correct Z and, as a consequence, we may make use of the relevant typing information.

$$
\begin{array}{lll}
(i) & [\![\neg S_T]\!] & =_{df} \neg_T [\![S]\!] \\
(ii) & [\![S_T \vee S'_{T'}]\!] & =_{df} [\![S]\!] \vee_{(T,T')} [\![S']\!] \\
(iii) & \left[\!\left[S \setminus (l : T')_{T \setminus (l:T')} \right]\!\right] & =_{df} [\![S]\!] \setminus_T (l : T')
\end{array}
$$

5.3 Sets

There are a number of new forms of set available in Z. We first introduce set operations for power sets and cartesian products in Z_C:

$$
\begin{array}{lll}
(i) & \mathbb{P}_T C & =_{df} \{z : \mathbb{P}\, T \mid \forall x : T \bullet x \in z \Rightarrow x \in C\} \\
(ii) & C_0 \times_{(T_0,T_1)} C_1 & =_{df} \{z : T_0 \times T_1 \mid z.1 \in C_0 \wedge z.2 \in C_1\}
\end{array}
$$

Then sets in Z are translated as follows:

$$
\begin{array}{lll}
(i) & [\![[D \mid P]_{\mathbb{P}\,T}]\!] & =_{df} \{z : T \mid z \in [\![[D]]\!] \wedge [\![P]\!] [\alpha T / z.\alpha T]\} \\
(ii) & [\![\{z \in C_{\mathbb{P}\,T} \mid P\}]\!] & =_{df} \{z : T \mid z \in [\![C]\!] \wedge [\![P]\!]\} \\
(iii) & [\![\mathbb{P}\, C_{\mathbb{P}\,T}]\!] & =_{df} \mathbb{P}_T [\![C]\!] \\
(iv) & \left[\!\left[C_{\mathbb{P}\,T} \times C'_{\mathbb{P}\,T'} \right]\!\right] & =_{df} [\![C]\!] \times_{(T,T')} [\![C']\!] \\
(v) & [\![\mathbb{N}]\!] & =_{df} \{z : \mathbb{N} \mid z = z\} \\
(vi) & [\![[\cdots l_i \in C_i \cdots]_{[\cdots l_i : T_i \cdots]}]\!] & =_{df} \{z : [\cdots l_i : T_i \cdots] \mid \cdots \wedge z.l_i \in [\![C_i]\!] \wedge \cdots\} \\
(vii) & [\![\lambda z \in C_{\mathbb{P}\,T} \bullet t_T]\!] & =_{df} \{x : T \times T' \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t]\!] [z / x.1]\}
\end{array}
$$

5.4 Terms

In addition to the sets, which are translated above, there are two new forms of term in Z. In full the translation is:

$$
\begin{array}{lll}
(i) & [\![x]\!] & =_{df} x \\
(ii) & [\![n]\!] & =_{df} n \\
(iii) & [\![t.l]\!] & =_{df} [\![t]\!].l \\
(iv) & [\![(\!| \cdots l_i \Rrightarrow t_i \cdots |\!)]\!] & =_{df} (\!| \cdots l_i \Rrightarrow [\![t]\!] \cdots |\!) \\
(v) & [\![t.1]\!] & =_{df} [\![t]\!].1 \\
(vi) & [\![t.2]\!] & =_{df} [\![t]\!].2 \\
(vii) & [\![(t_0, t_1)]\!] & =_{df} ([\![t_0]\!], [\![t_1]\!]) \\
(viii) & [\![let\ x == t_0\ in\ t_1]\!] & =_{df} [\![t_1]\!] [x / [\![t_0]\!]] \\
(ix) & [\![(\lambda z \in C \bullet t_0)\ t_1]\!] & =_{df} [\![t_0]\!] [z / [\![t_1]\!]] \\
(x) & [\![t \upharpoonright T]\!] & =_{df} [\![t]\!] \upharpoonright T
\end{array}
$$

Lemma 32. $[\![P[x/t]]\!] = [\![P]\!] [x / [\![t]\!]]$

Proof. Variables are unchanged by the translation and this is sufficient. □

We will use this property without further reference in the sequel.

5.5 Correctness of the translation

The syntax of the systems Z and Z_C are both given in two parts: a proto-syntax equipped with rules for type assignment and propositionhood. Our translation supposes that the Z expressions it considers are well-formed, but yields expressions which are *prima facie* only in the proto-syntax of Z_C. It is incumbent upon us to show that the translation preserves syntactic well-formedness. This is the content of the following proposition.

Proposition 33.
(*i*) *If* $\Gamma \rhd_Z P$ *prop then* $\Gamma \rhd_C [\![P]\!]$ *prop*
(*ii*) *If* $\Gamma \rhd_Z t : T$ *then* $\Gamma \rhd_C [\![t]\!] : T$

Proof. By simultaneous induction on the structure of the antecedent derivations.
Ad (*i*): Assume $\Gamma \rhd_Z P$ *prop*. We proceed by cases.
Ad Rules $(Z_\bot)$, $(Z_=)$, $(Z_\in)$, $(Z_\neg)$, $(Z_\vee)$:
These all follow immediately *ex hypothesi*.
Ad Rule $(Z_\exists)$:
We have $\Gamma \rhd_C [\![C]\!] : \mathbb{P}\, T$ *ex hypothesi* from which, by lemma 3(*iii*), we may conclude that $\Gamma, z : T \rhd_C [\![C]\!] : \mathbb{P}\, T$. From this and the instance of axiom (C_x) $\Gamma, z : T \rhd_C z : T$ we have $\Gamma, z : T \rhd_C z \in [\![C]\!]$ by rule $(C_\in)$. From this and $\Gamma, z : T \rhd_C [\![P]\!]$ *prop*, which follows *ex hypothesi*, we may conclude that $\Gamma, z : T \rhd_C z \in [\![C]\!] \wedge [\![P]\!]$ *prop* by (derived) rule $(C_\wedge)$. Then by rule $(C_\exists)$ we have $\Gamma \rhd_C \exists z : T \bullet z \in [\![C]\!] \wedge [\![P]\!]$ *prop* which is $\Gamma \rhd_C [\![\exists z \in C \bullet P]\!]$ *prop* as required.
Ad (*ii*): Assume $\Gamma \rhd_Z t : T$. We proceed by cases.
Ad Rules (Z_x), (Z_0), (Z_s), (Z_r), $(Z_.)$, $(Z_\Rightarrow)$, (Z_1), (Z_2), $(Z_()$, $(Z_\upharpoonright)$:
These all follow immediately *ex hypothesi*.
Ad Rule $(Z_{\vee S})$:
Observe that $T_0 \preceq T_0 \vee T_1$. Together with the instance of axiom (C_x) $\Gamma, z : T_0 \vee T_1 \rhd_C z : T_0 \vee T_1$ we have $\Gamma, z : T_0 \vee T_1 \rhd_C z \upharpoonright T_0 : T_0$ by rule $(C_\upharpoonright)$. From this and $\Gamma, z : T_0 \vee T_1 \rhd_C [\![S_0]\!] : \mathbb{P}\, T_0$, which follows *ex hypothesi*, we obtain $\Gamma, z : T_0 \vee T_1 \rhd_C z \upharpoonright T_0 \in [\![S_0]\!]$ *prop*. A similar argument from the other induction hypothesis allows us to conclude that $\Gamma, z : T_0 \vee T_1 \rhd_C z \upharpoonright T_1 \in [\![S_1]\!]$ *prop*. Combining these, using (derived) rule $(C_\wedge)$ we obtain $\Gamma, z : T_0 \vee T_1 \rhd_C z \upharpoonright T_0 \in [\![S_0]\!] \wedge z \upharpoonright T_1 \in [\![S_1]\!]$ from which, by rule $(C_{\{\}})$, we have $\Gamma \rhd_C \{z : T_0 \vee T_1 \mid z \upharpoonright T_0 \in [\![S_0]\!] \wedge z \upharpoonright T_1 \in [\![S_1]\!]\} : \mathbb{P}(T_0 \vee T_1)$ which is $\Gamma \rhd_C [\![S_0 \vee S_1]\!] : \mathbb{P}(T_0 \vee T_1)$ as required.
Ad Rule (Z_h):
From the axiom (C_x) we may infer that $\Gamma, z : T \setminus (l : T'), x : T \rhd_C z : T \setminus (l : T')$ and that $\Gamma, z : T \setminus (l : T'), x : T \rhd_C x : T$. From the latter, and the observation that $T \setminus (l : T') \preceq T$ we have $\Gamma, z : T \setminus (l : T'), x : T\ x \upharpoonright T \setminus (l : T') : T \setminus (l : T')$ by rule $(C_\upharpoonright)$. Combining this with the former we have $\Gamma, z : T \setminus (l : T'), x : T \rhd_C z = x \upharpoonright T \setminus (l : T')$ *prop* by rule $(C_=)$. From the latter instance of axiom (C_x) above together with $\Gamma, z : T \setminus (l : T'), x : T \rhd_C [\![S]\!] : \mathbb{P}\, T$, which we have *ex hypothesi*, we may conclude that $\Gamma, z : T \setminus (l : T'), x : T \rhd_C x \in [\![S]\!]$ *prop* by rule $(C_\in)$. Putting these two propositions together by (derived) rule $(C_\wedge)$ we obtain $\Gamma, z : T \setminus (l : T'), x : T \rhd_C x \in [\![S]\!] \wedge z = x \upharpoonright T \setminus (l : T')$ *prop*. Hence, $\Gamma, z : T \setminus (l : T') \rhd_C \exists x : T \bullet x \in [\![S]\!] \wedge z = x \upharpoonright T \setminus (l : T')$ *prop* by rule $(C_\exists)$ and then $\Gamma \rhd \{z : T \setminus (l : T') \mid \exists x : T \bullet x \in [\![S]\!] \wedge z = x \upharpoonright T \setminus (l : T')\} : \mathbb{P}\, T \setminus (l : T')$ which is $\Gamma \rhd_C [\![S \setminus (l : T')]\!] : \mathbb{P}\, T \setminus (l : T')$ as required.
Ad Rule $(Z_{\neg S})$:
Combining, using rule $(C_\in)$, the axiom instance $\Gamma, z : T \rhd_C z : T$ and $[\![S]\!] : \mathbb{P}\, T$, which we have *ex hypothesi*, we obtain $\Gamma, z : T \rhd_C z \in [\![S]\!]$ *prop*. From this we have $\Gamma, z : T \rhd_C z \notin [\![S]\!]$ *prop* by rule $(C_\neg)$. Hence, we obtain $\Gamma \rhd_C \{z : T \mid z \notin [\![S]\!]\} : \mathbb{P}\, T$ by rule $(C_{\{\}})$ which is $\Gamma \rhd_C [\![\neg S]\!] : \mathbb{P}\, T$ as required.
Ad Rule $(Z_{\{\}})$:
We have $\Gamma \rhd_C [\![C]\!] : \mathbb{P}\, T$ *ex hypothesi* from which $\Gamma, z : T \rhd_C [\![C]\!] : \mathbb{P}\, T$ by 3(*iv*). Combining this with $\Gamma, z : T \rhd_C z : T$, which is an instance of axiom (C_x), we have $\Gamma, z : T \rhd_C z \in [\![C]\!]$ *prop*. This, together with $\Gamma, z : T \rhd_C [\![P]\!]$ *prop*, which follows *ex hypothesi*, we may infer that $\Gamma, z : T \rhd_C z \in [\![C]\!] \wedge [\![P]\!]$ *prop* by (derived) rule $(C_\wedge)$. From this we obtain $\Gamma \rhd_C \{z : T \mid z \in [\![C]\!] \wedge [\![P]\!]\} : \mathbb{P}\, T$ by rule $(C_{\{\}})$, and this is $\Gamma \rhd_C [\![\{z \in C \mid P\}]\!] : \mathbb{P}\, T$ as required.
Ad Rule $(Z_\mathbb{P})$:
From the instances $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C x : T$ and $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C z : \mathbb{P}\, T$ of the axiom (C_x) we may conclude, by rule $(C_\in)$, that $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C x \in z$ *prop*. In addition from the former axiom instance and from $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C [\![C]\!] : \mathbb{P}\, T$, which is obtained *ex hypothesi* using lemma 3(*iii*), we may

conclude that $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C x \in [\![C]\!]$ *prop*. These two propositions combine, using (derived) rule $(C_{\Rightarrow})$, to yield $\Gamma, z : \mathbb{P}\, T, x : T \rhd_C x \in z \Rightarrow x \in [\![C]\!]$ *prop*. From this we obtain $\Gamma, z : \mathbb{P}\, T \rhd_C \forall x : T \bullet x \in z \Rightarrow x \in [\![C]\!]$ *prop* by (derived) rule $(C_{\forall})$ and then $\Gamma \rhd_C \{z : \mathbb{P}\, T \mid \forall x : T \bullet x \in z \Rightarrow x \in [\![C]\!]\} : \mathbb{P}\mathbb{P}\, T$ follows by rule $(C_{\{\}})$. But this is $\Gamma \rhd_C [\![\mathbb{P}\, C]\!] : \mathbb{P}\mathbb{P}\, T$ as required.

Ad Rule $(Z_{\times})$:

By axiom (C_x) we have $\Gamma, z : T_0 \times T_1 \rhd_C z : T_0 \times T_1$ and then, by rule (C_1) we may conclude that $\Gamma, z : T_0 \times T_1 \rhd_C z.1 : T_0$. We also have $\Gamma \rhd_C [\![C_0]\!] : \mathbb{P}\, T_0$ *ex hypothesi* and by lemma $3(iii)$ $\Gamma, z : T_0 \times T_1 \rhd_C [\![C_0]\!] : \mathbb{P}\, T_0$ follows. Combining these, using rule $(C_{\in})$ we obtain $\Gamma, z : T_0 \times T_1 \rhd_C z.1 \in [\![C_0]\!]$ *prop*. A similar argument shows that $\Gamma, z : T_0 \times T_1 \rhd_C z.2 \in [\![C_1]\!]$ *prop*. Together these imply $\Gamma, z : T_0 \times T_1 \rhd_C z.1 \in [\![C_0]\!] \wedge z.2 \in [\![C_1]\!]$ *prop* by (derived) rule $(C_{\wedge})$. Then, by rule $(C_{\{\}})$ we have $\Gamma \rhd_C \{z : T_0 \times T_1 \mid z.1 \in [\![C_0]\!] \wedge z.2 \in [\![C_1]\!]\} : \mathbb{P}(T_0 \times T_1)$ which is $\Gamma \rhd_C [\![C_0 \times C_1]\!] : \mathbb{P}(T_0 \times T_1)$ as required.

Ad Rule $(Z_{\mathbb{N}})$:

From rule $(C_{=})$ we obtain, from the instance of axiom (C_x) $\Gamma, z : \mathbb{N} \rhd_C z : \mathbb{N}$, $\Gamma, z : \mathbb{N} \rhd_C z = z$ *prop*. Hence, by rule $(C_{\{\}})$, we have $\Gamma \rhd_C \{z : \mathbb{N} \mid z = z\} : \mathbb{P}\mathbb{N}$. But this is $\Gamma \rhd_C [\![\mathbb{N}]\!] : \mathbb{P}\mathbb{N}$ as required.

Ad Rule $(Z_{[]})$:

We proceed by informal induction on the length of the schema type $[\cdots l_i : T_i \cdots]$. From the instance of the axiom (C_x) $\Gamma, z : [\cdots l_i : T_i \cdots] \rhd_C z : [\cdots l_i : T_i \cdots]$, we obtain $\Gamma, z : [\cdots L_i : T_i \cdots] \rhd_C z.l_i : T_i$ by rule $(C_.)$. *Ex hypothesi* we have $\Gamma \rhd_C [\![C_i]\!] : \mathbb{P}[\cdots l_i : T_i \cdots]$ whence, by lemma $3(iii)$, $\Gamma, z : [\cdots l_i : T_i \cdots] \rhd_C [\![C_i]\!] : \mathbb{P}[\cdots l_i : T_i \cdots]$. Hence, by rule $(C_{\in})$, $\Gamma, z : [\cdots l_i : T_i \cdots] \rhd_C z.l_i \in [\![C_i]\!]$ *prop*. Combining these derivations by (derived) rule $(C_{\wedge})$, we obtain $\Gamma, z : [\cdots l_i : T_i \cdots] \rhd_C \cdots \wedge z.l_i \in [\![C_i]\!] \wedge \cdots$ *prop*. Then, by rule $(C_{\{\}})$, we have $\Gamma \rhd_C \{z : [\cdots l_i : T_i \cdots] \mid \cdots \wedge z.l_i \in [\![C_i]\!] \wedge \cdots\} : \mathbb{P}[\cdots l_i : T_i \cdots]$ which is $\Gamma \rhd_C [\![[\cdots l_i \in C_i \cdots]]\!] : \mathbb{P}[\cdots l_i : T_i \cdots]$ as required.

Ad Rule (Z_{λ}):

By axiom (C_x) we have $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x : T_0 \times T_1$ whence, by rule (C_1), $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.1 : T_0$. A similar argument shows that $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.2 : T_1$. From the former together with $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C [\![C]\!] : \mathbb{P}\, T_0$ which follows *ex hypothesi* and by lemma $3(iii)$, we obtain $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.1 \in [\![C]\!]$ *prop* by rule $(C_{\in})$. *Ex hypothesi* we have $\Gamma, z : T_0 \rhd_C [\![t]\!] : T_1$. By lemma $3(iii)$ we have $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C [\![t]\!] : T_1$, and then using $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.1 : T_0$ together with lemma $5(iv)$ we have $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C [\![t]\!][z/x.1] : T_1$. From this and $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.2 : T_1$ we obtain, by rule $(C_{=})$, $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.2 = [\![t]\!][z/x.1]$ *prop*. Combining this, using (derived) rule $(C_{\wedge})$, with the earlier proposition we have $\Gamma, x : T_0 \times T_1, z : T_0 \rhd_C x.1 \in [\![C]\!] \wedge x.2 = [\![t]\!][z/x.1]$ *prop*. Hence, using rule $(C_{\{\}})$, we have $\Gamma, z : T_0 \rhd_C \{x : T_0 \times T_1 \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t]\!][z/x.1]\} : \mathbb{P}(T_0 \times T_1)$ and then $\Gamma \rhd_C \{x : T_0 \times T_1 \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t]\!][z/x.1]\} : \mathbb{P}(T_0 \times T_1)$ by lemma $3(ii)$. But this is $\Gamma \rhd_C [\![\lambda\, z \in C \bullet t]\!] : \mathbb{P}(T_0 \times T_1)$ as required.

Ad Rule (Z_{ap}):

By lemma 15 we have, from the first premise, that $\Gamma, z : T_0 \rhd_Z t_0 : T_1$ and then, *ex hypothesi* we obtain $\Gamma, z : T_0 \rhd_C [\![t_0]\!] : T_1$. From the second premise, *ex hypothesi*, we have, on the other hand, $\Gamma \rhd_C [\![t_1]\!] : T_0$. Then, by lemma $5(iv)$, we infer that $\Gamma \rhd_C [\![t_0]\!][z/[\![t_1]\!]] : T_1$ which is $\Gamma \rhd_C [\![(\lambda\, x \in C \bullet t_0)\ t_1]\!] : T_1$ as required.

Ad Rule (Z_{let}):

Ex hypothesi we have $\Gamma \rhd_C [\![t_0]\!] : T_0$ and $\Gamma, z : T_0 \rhd_C [\![t_1]\!] : T_1$. By lemma $18(iv)$ we may conclude that $\Gamma \rhd_C [\![t_1]\!][z/[\![t_0]\!]] : T_1$, but this is $\Gamma \rhd_C [\![let\ x == t_0\ in\ t_1]\!] : T_1$ as required. □

As a corollary we have the soundness of the type assignment system in *ZF*. We temporarily write $[\![\ \_\ ]\!]$ for the composition $[\![\ \_\ ]\!]_C \circ [\![\ \_\ ]\!]_Z$.

Corollary 34. *If* $\Gamma \rhd_Z t : T$ *then* $[\![\Gamma\ ]\!] \vdash_{zf} [\![t]\!] \in [\![T]\!]$

Proof. Compose propositions 12 and 33. □

Next we have the relative soundness result for the logic.

Proposition 35. *If* $\Gamma^-;\ \Gamma^+ \vdash_Z P$ *then* $\Gamma^-;\ [\![\Gamma^+]\!] \vdash_C [\![P]\!]$

Proof. By induction on the structure of the antecedent derivation. We shall suppress references to the contexts unless they play a significant role.

Ad Rules $(\vee_0^+)$, $(\vee_1^+)$, $(\vee^-)$, $(\neg^+)$, $(\neg^-)$, $(\bot^+)$, $(\bot^-)$, (ass), (ref), (sym), (sub), $(\Rrightarrow_0^=)$, $(\Rrightarrow_1^=)$, $((\,)_0^=)$,

$((\,)^{=}_1)$, $((\,)^{=}_2)$, $(\mathbb{N}^-)$, $(\restriction^{=})$:
These are also rules of Z_C.
Ad Rule $(\exists^+)$:
Ex hypothesi we have $[\![P]\!][z/[\![t]\!]]$ and $[\![t]\!] \in [\![C]\!]$. By (derived) rule $(\wedge)$ we have $(z \in [\![C]\!] \wedge [\![P]\!])[z/[\![t]\!]]$ and, using proposition 11, $[\![t]\!] : T$ for some T. Hence, by $(\exists^+)$, $\exists z : T \bullet z \in [\![C]\!] \wedge [\![P]\!]$ which is $[\![\exists z \in C_{\mathbb{P}\,T} \bullet P]\!]$ as required.
Ad Rule $(\exists^-)$:
Ex hypothesi we have $\Gamma \vdash_C [\![z \in C_{\mathbb{P}\,T} \bullet P_0]\!]$ and $\Gamma^-, y : T;\ [\![\Gamma^+]\!], [\![P_0[z/y]]\!] \vdash_C [\![P_1]\!]$. From the former we obtain, $\Gamma \vdash_C \exists z : T \bullet z \in [\![C]\!] \wedge [\![P_0]\!]$. This, and the latter are sufficient, by rule $(\exists^-)$, to conclude $\Gamma \vdash_C [\![P_1]\!]$ as required.
Ad Rule $(\{\}^+)$:
Ex hypothesi we have $[\![P[z/t]]\!]$ and $[\![t \in C_{\mathbb{P}\,T}]\!]$. From the latter, by lemmata 11 and 2, we obtain, $\triangleright_C t : T$. This, together with the former, by rule $(\{\}^+)$, yields $[\![t]\!] \in \{z : T \mid z \in [\![C]\!] \mid [\![P]\!]\}$ which is $[\![t \in \{z \in C \mid P\}]\!]$ as required.
Ad Rule $(\{\}^-_{\mathrm{o}})$:
Ex hypothesi we have $[\![t \in \{z \in C_{\mathbb{P}\,T} \mid P\}]\!]$ which amounts to $[\![t]\!] \in \{z : T \mid z \in C \wedge [\![P]\!]\}$. By rule $(\{\}^-)$ we obtain $[\![t]\!] \in [\![C]\!] \wedge [\![P]\!][z/[\![t]\!]]$ whence $[\![t]\!] \in [\![C]\!]$, by (derived) rule $(\wedge^-_{\mathrm{o}})$, as required.
Ad Rule $(\{\}^-_1)$:
Ex hypothesi we have $[\![t \in \{z \in C_{\mathbb{P}\,T} \mid P\}]\!]$ which amounts to $[\![t]\!] \in \{z : T \mid z \in C \wedge [\![P]\!]\}$. By rule $(\{\}^-)$ we obtain $[\![t]\!] \in [\![C]\!] \wedge [\![P]\!][z/[\![t]\!]]$ whence $[\![P]\!][z/[\![t]\!]]$, by (derived) rule $(\wedge^-_1)$, as required.
Ad Rule $(\neg S^+)$:
Ex hypothesi we have $[\![t \notin S]\!]$ whence $[\![t]\!] \notin [\![S]\!]$. By lemma 11, we know that $[\![t]\!] : T$ for some type T, and then, by rule $(\{\}^+)$, we obtain $[\![t]\!] \in \{z : T \mid z \notin [\![S]\!]\}$ which is $[\![t \in \neg S]\!]$ as required.
Ad Rule $(\neg S^-)$:
Ex hypothesi we have $[\![t \in \neg S_{\mathbb{P}\,T}]\!]$ which is $[\![t]\!] \in [\![\neg S]\!]$, and this is $[\![t]\!] \in \{z : T \mid z \notin [\![S]\!]\}$. Using rule $(\{\}^-)$ we obtain $[\![t]\!] \notin [\![S]\!]$ from which we conclude that $[\![t \notin S]\!]$ as required.
Ad Rule $(S^+_{\leftarrow})$:
This follows *ex hypothesi* as a substitution instance.
Ad Rule $(S^-_{\leftarrow})$:
Similarly.
Ad Rule (S^+_h):
From the first premise *ex hypothesi* we have $[\![t]\!] \in [\![S]\!]$ and from the second, by lemma 33, $[\![t]\!] : T$. From this, by rule (ref), we obtain $[\![t]\!] = [\![t]\!]$, whence, by (derived) rule $(=_{\restriction})$ and the fact that $T \preceq T \setminus (l : T')$, $[\![t]\!] \restriction T \setminus (l : T') = [\![t]\!] \restriction T \setminus (l : T')$. Then by (derived) rule $(\wedge^+)$ we infer that $[\![t]\!] \in [\![S]\!] \wedge [\![t]\!] \restriction T \setminus (l : T') = [\![t]\!] \restriction T \setminus (l : T')$, and then, by rule $(\exists^+)$, we have $\exists x : T \bullet x \in [\![S]\!] \wedge [\![t]\!] \restriction T \setminus (l : T') = x \restriction T \setminus (l : T')$. Proceeding by rule $(\{\}^+)$ we then obtain $[\![t]\!] \restriction T \setminus (l : T') \in \{z : T \setminus (l : T') \mid \exists x : T \bullet x \in [\![S]\!] \wedge z = x \restriction T \setminus (l : T')\}$ which is $[\![t \restriction T \setminus (l : T') \in S \setminus (l : T')]\!]$ as required.
Ad Rule (S^-_h):
From the first premise *ex hypothesi* we have $[\![t]\!] \in [\![S \setminus (l : T')]\!]$ which is, using rule $(\{\}^-)$, $\exists x : T \bullet x \in [\![S]\!] \wedge t = x \restriction T \setminus (l : T')$. $[\![P]\!]$ then follows, by rule $(\exists^-)$ providing that $y : T;\ y \in [\![S]\!], y \restriction T \setminus (l : T') = [\![t]\!] \vdash [\![P]\!]$ but this is the third premise.
Ad Rule $(S^+_{\vee_{\mathrm{o}}})$:
From the first premise, *ex hypothesi* we have $[\![t \restriction T \in S]\!]$ which is $[\![t]\!] \restriction T \in [\![S]\!]$. From the second and third premises we obtain, by lemma 33(ii), $[\![S']\!] : \mathbb{P}\,T'$ and $[\![t]\!] : T \vee T'$. From the latter, by rule $(C_{\restriction})$, we may infer that $[\![t]\!] \restriction T' : T'$ (since $T' \preceq T \vee T'$) and from this and the latter, by rule $(C_{\in})$, we obtain $[\![t]\!] \restriction T' \in [\![S']\!]$ *prop*. Then by rule $(\vee^+_{\mathrm{o}})$, we have $[\![t]\!] \restriction T \in [\![S]\!] \vee [\![t]\!] \restriction T' \in [\![S']\!]$ which simplifies, using rule $(\{\}^+)$, to $[\![t \in S \vee S']\!]$ as required.
Ad Rule $(S^+_{\vee_1})$:
From the first premise, *ex hypothesi* we have $[\![t \restriction T' \in S']\!]$ which is $[\![t]\!] \restriction T' \in [\![S']\!]$. From the second and third premises we obtain, by lemma 33(ii), $[\![S]\!] : \mathbb{P}\,T$ and $[\![t]\!] : T \vee T'$. From the latter, by rule $(C_{\restriction})$, we may infer that $[\![t]\!] \restriction T : T$ (since $T \preceq T \vee T'$) and from this and the latter, by rule $(C_{\in})$, we obtain $[\![t]\!] \restriction T \in [\![S]\!]$ *prop*. Then by rule $(\vee^+_1)$, we have $[\![t]\!] \restriction T \in [\![S]\!] \vee [\![t]\!] \restriction T' \in [\![S']\!]$ which simplifies, using rule $(\{\}^+)$, to $[\![t \in S \vee S']\!]$ as required.
Ad Rule $(S^-_{\vee})$:
From the first premise, *ex hypothesi* we have $[\![t]\!] \in [\![S \vee S']\!]$ which is, using rule $(\{\}^-)$, $[\![t]\!] \restriction T \in [\![S]\!] \vee [\![t]\!] \restriction T' \in [\![S']\!]$. From the second and third premises we have, *ex hypothesi*, $[\![t]\!] \restriction T \in [\![S]\!] \vdash_C [\![P]\!]$ and

$[\![t]\!] \upharpoonright T' \in [\![S']\!] \vdash_C [\![P]\!]$. From these three, using rule $(\vee^-)$, we obtain $[\![P]\!]$ as required.

Ad Rule $(\mathbb{P}^+)$:

From the first premise, *ex hypothesi* we have $z \in [\![C_0]\!] \vdash z \in [\![C_1]\!]$ and then $z \in [\![C_0]\!] \Rightarrow z \in [\![C_1]\!]$ by (derived) rule $(\Rightarrow^+)$. From the second premise, using proposition $33(i)$, $z \in [\![C_0]\!]$ *prop*. From the latter, by lemma 2 we obtain $z : T$ and $[\![C_0]\!] : \mathbb{P}\, T$. But in view of the former, we may infer, using (derived) rule $(\forall^+)$, that $\forall z : T \bullet z \in [\![C_0]\!] \Rightarrow z \in [\![C_1]\!]$ and then this simplifies, using rule $(\{\}^+)$, to $[\![C_0 \in \mathbb{P}\, C_1]\!]$ as required.

Ad Rule $(\mathbb{P}^-)$:

From the premises, *ex hypothesi* we have $[\![C_0]\!] \in \mathbb{P}[\![C_1]\!]$ and $[\![t]\!] \in [\![C_0]\!]$. By lemma 2 and proposition 11 we obtain $[\![t]\!] : T$, $[\![C_0]\!] : \mathbb{P}\, T$ and $[\![C_1]\!] : \mathbb{P}\, T$ for some T. From $[\![C_0]\!] \in \mathbb{P}[\![C_1]\!]$ we have $[\![C_0]\!] \in \{z : T \mid \forall x : T \bullet z \in z \Rightarrow x \in [\![C_1]\!]\}$ which, by rule $(\{\}^-)$, leads to $\forall x : T \bullet x \in [\![C_0]\!] \Rightarrow x \in [\![C_1]\!]$. Since $[\![t]\!] : T$ we may infer, by (derived) rule $(\forall^-)$, that $[\![t]\!] \in [\![C_0]\!] \Rightarrow [\![t]\!] \in [\![C_1]\!]$. Since $[\![t]\!] \in [\![C_0]\!]$ this yields, by (derived) rule $(\Rightarrow^-)$, $[\![t \in C_1]\!]$ as required.

Ad Rule $(\times^+)$:

From the premises, *ex hypothesi* we have $[\![t_0]\!] \in [\![C_0]\!]$ and $[\![t_1]\!] \in [\![C_1]\!]$. By rule, $(\wedge^+)$ we obtain $[\![t_0]\!] \in [\![C_0]\!] \wedge [\![t_1]\!] \in [\![C_1]\!]$ and then, by rules $((\,)^=_0)$ and $((\,)^=_1)$, this gives $([\![t_0]\!], [\![t_1]\!]).1 \in [\![C_0]\!] \wedge ([\![t_0]\!], [\![t_1]\!]).2 \in [\![C_1]\!]$. Rule $(\{\}^+)$ permits us to infer that $([\![t_0]\!], [\![t_1]\!]) \in \{z : T_0 \times T_1 \mid z.1 \in [\![C_0]\!] \wedge z.2 \in [\![C_1]\!]\}$, since we have $([\![t_0]\!], [\![t_1]\!]) : T_0 \times T_1$, for some T_0 and T_1, from the above using lemma 2 and rule $(C_{()})$. But this is $[\![(t_0, t_1) \in C_0 \times C_1]\!]$ as required.

Ad Rule $(\times^-_0)$:

From the premise, *ex hypothesi* we have $[\![t]\!] \in [\![C_0 \times C_1]\!]$ which simplifies, using rule $(\{\}^-)$, to $[\![t]\!].1 \in [\![C_0]\!] \wedge [\![t]\!].2 \in [\![C_1]\!]$. But then, by (derived) rule $(\wedge^-_0)$, we have $[\![t.1 \in C_0]\!]$ as required.

Ad Rule $(\times^-_1)$:

From the premise, *ex hypothesi* we have $[\![t]\!] \in [\![C_0 \times C_1]\!]$ which simplifies, using rule $(\{\}^-)$, to $[\![t]\!].1 \in [\![C_0]\!] \wedge [\![t]\!].2 \in [\![C_1]\!]$. But then, by (derived) rule $(\wedge^-_1)$, we have $[\![t.2 \in C_1]\!]$ as required.

Ad Rule $(\mathbb{N}^+_0)$:

By rule (C_0) we have $0 : \mathbb{N}$ and then, by rule (ref), $0 = 0$. Then, by rule $(\{\}^+)$, we obtain $0 \in \{z : \mathbb{N} \mid z = z\}$ which is $[\![0 \in \mathbb{N}]\!]$ as required.

Ad Rule $(\mathbb{N}^+_1)$:

From the premise, *ex hypothesi* we have $[\![n \in \mathbb{N}]\!]$ which is $n \in \{z : \mathbb{N} \mid z = z\}$. By rule $(\{\}^-_0)$ we obtain $n : \mathbb{N}$, hence, by rule (C_s), *succ* $n : \mathbb{N}$. By rule (C_{ref}), this leads to *succ* n = *succ* n, whence, by rule $(\{\}^+)$, *succ* $n \in \{z : \mathbb{N} \mid z = z\}$. But this is $[\![succ\ n \in \mathbb{N}]\!]$ as required.

Ad Rule $([]^+)$:

From the premises we have, *ex hypothesi* that $\cdots [\![t_i]\!] \in [\![C_i]\!] \cdots$. Using lemma 2 we may obtain $\cdots [\![t_i]\!] : T_i \cdots$ for types $\cdots T_i \cdots$. Hence, by rule $(C_{\Rrightarrow})$, we obtain $(\!| \cdots l_i \Rrightarrow t_i \cdots |\!) : [\cdots l_i : T_i \cdots]$. This permits us to use rules (*sub*) and $(\Rrightarrow^=_0)$ to obtain $\cdots (\!| \cdots l_i \Rrightarrow [\![t_i]\!] \cdots |\!).l_i \in [\![C_i]\!] \cdots$. Using (derived) rule $(\wedge^+)$ we infer that $\cdots \wedge (\!| \cdots l_i \Rrightarrow [\![t_i]\!] \cdots |\!).l_i \in [\![C_i]\!] \wedge \cdots$ and then, using the typing condition $(\!| \cdots l_i \Rrightarrow t_i \cdots |\!) : [\cdots l_i : T_i \cdots]$. once again, by $(\{\}^+)$, we conclude that $(\!| \cdots l_i \Rrightarrow [\![t_i]\!] \cdots |\!) \in \{z : [\cdots l_i : T_i \cdots] \mid \cdots \wedge z.l_i \in [\![C_i]\!] \wedge \cdots\}$ which is $(\!| \cdots l_i \Rrightarrow [\![t_i]\!] \cdots |\!) \in [\cdots l_i \in [\![C_i]\!] \cdots]$ as required.

Ad Rule $([]^-)$:

From the premise, *ex hypothesi* we have $[\![t]\!] \in [\cdots l_i \in [\![C_i]\!] \cdots]$ and then, by rule $(\{\}^-)$, we may conclude that $[\![t]\!].l_i \in [\![C_i]\!]$ as required.

Ad Rule (λ^+):

From the first premise, *ex hypothesi* we have $[\![t_0]\!] \in [\![C]\!]$ and from the second and third, using lemma 33, $[\![C]\!] : \mathbb{P}\, T_0$ and $z : T_0 \triangleright [\![t_1]\!] : T_1$. By lemmata 2 and $5(iv)$ we obtain $[\![t_1[x/t_1]]\!] : T_1$. By rules $(\times^-_0)$, (ref), $(\times^-_1)$, $(\wedge^+)$ and $(\{\}^+)$, we conclude that $([\![t_0]\!], [\![t_1[z/t_0]]\!]) \in \{x : T_0 \times T_1 \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t_1[z/t_0]]\!]\}$ which is $[\![(t_0, t_1[z/t_0]) \in \lambda\, z \in C \bullet t_1]\!]$ as required.

Ad Rule (λ^-_0):

From the premise, *ex hypothesi* we have $[\![t_0]\!] \in \{x : T_0 \times T_1 \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t_1[z/t_0]]\!]\}$ hence, by rules $(\{\}^-)$ and $(\wedge^-_0)$ we obtain $[\![t_0]\!].1 \in [\![C]\!]$ as required.

Ad Rule (λ^-_1):

From the premise, *ex hypothesi* we have $[\![t_0]\!] \in \{x : T_0 \times T_1 \mid x.1 \in [\![C]\!] \wedge x.2 = [\![t_1[z/t_0]]\!]\}$ hence, by rules $(\{\}^-)$ and $(\wedge^-_1)$ we obtain $[\![t_0]\!].2 = [\![t_1[z/t_0]]\!]$ as required.

Ad Rule (*let*):

From the premises, using lemma 33, we have $[\![t]\!] : T$ and $z : T \triangleright [\![t']\!] : T'$. By lemma $5(iv)$ we obtain $[\![t'[z/t]]\!] : T$, hence, by rule (ref), $[\![t'[z/t]]\!] = [\![t'[z/t]]\!]$. But this is $[\![let\ z == t\ in\ t']\!] = [\![t'[z/t]]\!]$ as required.

Ad Rule $(\{\}^{=})$:
From the first premise, *ex hypothesi* we have $z : T \vdash [\![P_0]\!] \Leftrightarrow [\![P_1]\!]$ and from the second, by lemma 33, $[\![C]\!] : \mathbb{P}\, T$. By rule $(C_{\in})$ we have $z : T \rhd z \in [\![C]\!]$ *prop*, hence it follows that $z : T \vdash z \in [\![C]\!] \wedge [\![P_0]\!] \Leftrightarrow z \in [\![C]\!] \wedge [\![P_1]\!]$, whence, by rule $(=_{\{\}})$, $\{z \in C \mid z \in [\![C]\!] \wedge [\![P_0]\!]\} = \{z \in C \mid z \in [\![C]\!] \wedge [\![P_1]\!]\}$ as required. □

Finally, as a corollary, we have soundness for the logic in *ZF*. Let us write $[\![\ \_\ ]\!]$ in what follows for the interpretation $[\![\ \_\ ]\!]_C \circ [\![\ \_\ ]\!]_Z$ of Z within *ZF*.

Corollary 36. *If* $\Gamma \vdash_Z P$ *then* $[\![\Gamma\ ]\!] \vdash_{zf} [\![P]\!]$

Proof. Combine propositions 35 and 13. □

This together with corollary 34 completes the process of modelling Z in *ZF*.

6 Derived constructs

We have indicated that, given the basic connectives and set (hence schema) operations we have so far, we can construct all of the others that we expect to find in Z. Since we are particularly concerned with developing the logic of Z, it is perhaps appropriate that we examine the logical consequences of the expected constructions. We need not dwell on the extension of the propositions (to include conjunction, implication and a universal quantifier) since this is all quite standard in classical logic, and the small burden of the type constraints adds no serious complexity to the task. We shall, though, spend some time extending the schema calculus, since this is less trivial and is also of major importance in Z[10].

6.1 Schema conjunction

We would expect to define conjunction over schema by analogy with operations like set intersection and logical conjunction:

$$S \wedge S' =_{df} \neg(\neg S \vee \neg S')$$

Using rules $(Z_{\neg})$ and $(Z_{\vee})$ we obtain the following derived rule for type assignment:

$$\frac{\Gamma \rhd S : \mathbb{P}\, T \quad \Gamma \rhd S' : \mathbb{P}\, T'}{\Gamma \rhd S \wedge S' : \mathbb{P}(T \vee T')}\ (Z_{\wedge})$$

The translation of this new construct into Z_C is immediate:

$$[\![S_T \wedge S'_{T'}]\!] = \neg_{\ T \vee T'}(\neg_{\ T} [\![S]\!] \vee_{(T,T')} \neg_{\ T'} [\![S']\!])$$

We can simplify the right-hand side of this:

$$\begin{aligned}
[\![S_T \wedge S'_{T'}]\!] &=_{df} \neg_{\ T \vee T'}(\neg_{\ T} [\![S]\!] \vee_{(T,T')} \neg_{\ T'} [\![S']\!]) \\
&= \{z : T \vee T' \mid z \notin \{y : T \vee T' \mid y \upharpoonright T \in \neg_{\ T} [\![S]\!] \vee y \upharpoonright T' \in \neg_{\ T'} [\![S']\!]\}\} \\
&= \{z : T \vee T' \mid z \in \{y : T \vee T' \mid y \upharpoonright T \notin \neg_{\ T} [\![S]\!] \wedge y \upharpoonright T' \notin \neg_{\ T'} [\![S']\!]\}\} \\
&= \{y : T \vee T' \mid y \upharpoonright T \notin \neg_{\ T} [\![S]\!] \wedge y \upharpoonright T' \notin \neg_{\ T'} [\![S']\!]\} \\
&= \{y : T \vee T' \mid y \upharpoonright T \notin \{z : T \mid z \notin [\![S]\!]\} \wedge y \upharpoonright T' \notin \{z : T' \mid z \notin [\![S']\!]\}\} \\
&= \{y : T \vee T' \mid y \upharpoonright T \in [\![S]\!] \wedge y \upharpoonright T' \in [\![S']\!]\}
\end{aligned}$$

[10] In fact it is not quite clear why this has traditionally been referred to as a schema *calculus* since the literature we have cited typically introduces nothing beyond a *notation* for schema expressions. It is true that there are some suggested rules for membership in compound schema, at least for the simplest cases like conjunction, in the normative source [Nic95], but these are quite clearly wrong. For example (*ibid* Section F.6.6, p. 207) we can easily derive the following rules, the first of which is too restrictive and the others are not even well-formed:

$$\frac{\Gamma \vdash b \in S \quad \Gamma \vdash b \in T}{\Gamma \vdash b \in S \wedge T} \qquad \frac{\Gamma \vdash b \in S \wedge T}{\Gamma \vdash b \in S} \qquad \frac{\Gamma \vdash b \in S \wedge T}{\Gamma \vdash b \in T}$$

using the rules $(SchBindMem)$ and $(SAnd)$. Although this is a well-known problem of the treatment in [Nic95], the suggested remedy (adding a proviso to rule $(SchBindMem)$) [Mar98] only ensures that the elimination rules above are well-defined: they are still very limited. Beyond this incorrect treatment of the simplest aspects of the schema calculus there is almost nothing, for example the incomplete rule for schema composition $(SComp)$ (*ibid* p. 208).

Given this translation the following rule follows by rules $(\wedge^+)$ and $(\{\}^+)$:

$$\frac{\Gamma \vdash t \restriction T \in S \quad \Gamma \vdash t \restriction T' \in S' \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \in S \wedge S'} \ (S^+_\wedge)$$

For the corresponding elimination rules we have the following rules using rule $(\{\}^-)$ and (derived) rules $(\wedge^-_0)$ and $(\wedge^-_1)$:

$$\frac{\Gamma \vdash t \in S \wedge S' \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \restriction T \in S} \ (S^-_{\wedge_0}) \qquad \frac{\Gamma \vdash t \in S \wedge S' \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \restriction T' \in S'} \ (S^-_{\wedge_1})$$

With these in place we can prove the expected relationship (see [WD96] p. 165-6):

Lemma 37.

$$\frac{\Gamma^- \triangleright [D^*_0 \mid P_0] : \mathbb{P}\, T_0 \quad \Gamma^- \triangleright [D^*_1 \mid P_1] : \mathbb{P}\, T_1}{\Gamma \vdash [D^*_0 \mid P_0] \wedge [D^*_1 \mid P_1] = [D^*_0 \vee D^*_1 \mid P_0 \wedge P_1]} \ (\wedge^=)$$

Proof. We can use the equational logic that we already have at our disposal. The result follows easily by rules $(\neg^=)$ and $(\vee^=)$ and De Morgan's laws. □

Finally we have substitution rules:

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \triangleright S_2 : \mathbb{P}\, T}{\Gamma \vdash S_0 \wedge S_2 = S_1 \wedge S_2} \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \triangleright S_2 : \mathbb{P}\, T}{\Gamma \vdash S_2 \wedge S_0 = S_2 \wedge S_1}$$

Again, these follow easily from the corresponding rules for disjunction and negation schema.

6.2 Schema implication

Following the pattern given above for conjunction, we would expect the definition:

$$S \Rightarrow S' =_{df} \neg S \vee S'$$

Using the rules $(Z_\neg)$ and $(Z_\vee)$ we obtain a derived rule for type assignment:

$$\frac{\Gamma \triangleright S : \mathbb{P}\, T \quad \Gamma \triangleright S' : \mathbb{P}\, T'}{\Gamma \triangleright S \Rightarrow S' : \mathbb{P}(T \vee T')}$$

Translation into Z_C is, then, given by:

$$[\![S_T \Rightarrow S'_{T'}]\!] =_{df} \neg_{\ T} [\![S]\!] \vee_{(T,T')} [\![S']\!]$$

Simplifying the right-hand side, we obtain:

$$\begin{aligned} [\![S_T \Rightarrow S'_{T'}]\!] &=_{df} \neg_{\ T} [\![S]\!] \vee_{(T,T')} [\![S']\!] \\ &= \{z : T \vee T' \mid z \restriction T \in \neg_{\ T} [\![S]\!] \vee z \restriction T' \in [\![S']\!]\} \\ &= \{z : T \vee T' \mid z \restriction T \notin [\![S]\!] \vee z \restriction T' \in [\![S']\!]\} \end{aligned}$$

The introduction rule that follows most directly from this would be

$$\frac{\Gamma \vdash t \restriction T \notin S \quad \Gamma \vdash t \restriction T' \in S' \quad \Gamma^- \triangleright t : T \vee T'}{\Gamma \vdash t \in S \Rightarrow S'}$$

however, the following derivation (which has some type information suppressed and assumes the theorem $\vdash P \vee \neg P$ for all propositions P) shows clearly that we can have a derived rule which is much more useful, and like the usual logical rule:

$$\dfrac{\dfrac{}{t \restriction T \in S \vee t \restriction T \notin S} \quad \dfrac{t \restriction T \in S \vdash t \restriction T' \in S' \quad \overline{t \restriction T \notin S \vdash t \restriction T \notin S \vee t \restriction T' \in S'} \quad \overline{t \restriction T \notin S \vdash t \restriction T \notin S}}{t \restriction T \in S \vdash t \restriction T \notin S \vee t \restriction T' \in S'}}{t \restriction T \notin S \vee t \restriction T' \in S'}$$

This derivation depends upon the assumption $t \upharpoonright T \in S \vdash t \upharpoonright T' \in S'$ and its conclusion is, by definition, $t \in S \Rightarrow S'$, so we have the derived rule:

$$\frac{\Gamma, t \upharpoonright T \in S \vdash t \upharpoonright T' \in S' \quad \Gamma^- \rhd t : T \vee T'}{\Gamma \vdash t \in S \Rightarrow S'} \ (S^+_\Rightarrow)$$

The obvious elimination rule follows directly from the semantics:

$$\frac{\Gamma \vdash t \in S \Rightarrow S' \quad \Gamma \vdash t \upharpoonright T \in S \quad \Gamma^- \rhd t : T \vee T'}{\Gamma \vdash t \upharpoonright T' \in S'} \ (S^-_\Rightarrow)$$

The expected relationship holds:

Lemma 38.

$$\frac{\Gamma^- \rhd [D_0^* \mid P_0] : \mathbb{P}\, T_0 \quad \Gamma^- \rhd [D_1^* \mid P_1] : \mathbb{P}\, T_1}{\Gamma \vdash [D_0^* \mid P_0] \Rightarrow [D_1^* \mid P_1] = [D_0^* \vee D_1^* \mid P_0 \Rightarrow P_1]}$$

Proof. Using the equational logic: rules $(\neg^=)$ and $(\vee^=)$. □

Finally we have the expected substitution rules:

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_0 \Rightarrow S_2 = S_1 \Rightarrow S_2} \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_2 \Rightarrow S_0 = S_2 \Rightarrow S_1}$$

6.3 Schema inclusion

Schema inclusion can be defined in terms of schema conjunction[11].

$$[D_0;\ [D_1 \mid P_1] \mid P_0] =_{df} [D_0 \vee D_1 \mid P_0] \wedge [D_1 \mid P_1]$$

It should be noted that this, unlike the other operators we consider, is a non-compositional definition which involves a generalisation, on the left-hand side, of the language of declarations to include schema references. But, since the definiens is meta-notation, this presents no further technical problems.

The rules are then easily calculated as special cases of those for schema conjunction. First the typing rules:

$$\frac{\Gamma \rhd [D_0;\ D_1 \mid P_0] : \mathbb{P}(T_0 \vee T_1) \quad \Gamma \rhd [D_1 \mid P_1] : \mathbb{P}\, T_1}{\Gamma \rhd [D_0;\ [D_1 \mid P_1] \mid P_1] : \mathbb{P}\, T_1} \ (Z_{inc})$$

The introduction rule is:

$$\frac{\Gamma^- \rhd t : T_0 \vee T_1 \quad \Gamma \vdash t \in [D_0 \vee D_1 \mid P_0] \quad \Gamma \vdash t \upharpoonright T_1 \in [D_1 \mid P_1]}{\Gamma \vdash t \in [D_0;\ [D_1 \mid P_1] \mid P_0]}$$

The elimination rules are:

$$\frac{\Gamma^- \rhd t : T_0 \vee T_1 \quad \Gamma \vdash t \in [D_0;\ [D_1 \mid P_1] \mid P_0]}{\Gamma \vdash t \in [D_0 \vee D_1 \mid P_0]} \qquad \frac{\Gamma^- \rhd t : T_0 \vee T_1 \quad \Gamma \vdash t \in [D_0;\ [D_1 \mid P_1] \mid P_1]}{\Gamma \vdash t \upharpoonright T_1 \in [D_1 \mid P_1]}$$

Finally, we have the expected equational law:

$$\frac{\Gamma^- \rhd [D_0^* \mid P_0] : \mathbb{P}\, T_0 \quad \Gamma^- \rhd [D_1^* \mid P_1] : \mathbb{P}\, T_1}{\Gamma \vdash [D_0^*;\ [D_1^* \mid P_1] \mid P_0] = [D_0^* \vee D_1^* \mid P_0 \wedge P_1]} \ (inc^=)$$

Although the Δ and Ξ schemas of Z are intimately linked, in the standard accounts, with schema inclusion, we shall delay their consideration until we discuss schema priming and the theta operator (section 7 below).

[11] Note that a similar strategy of eliminating schema inclusion which can be found in the existing literature, does not work, despite what is often claimed (*e.g.* [She95] p. 207, [Rat94], p. 206): in many cases including S in T, according to the standard account above, will type check when $S \wedge T$ does not. This is not a peculiar feature of our formalisation as reference to rule (T_{55}) ([Nic95] p. 203) will testify. [BJ95] (p. 173) makes a similar observation though expressed in terms of variable capture. These comments assume that expressions such as $S \wedge T$ are part of the object language, which is true of the two normative references [Spi92] and [Nic95]. Many of the text books tend to suggest that $S \wedge T$ is *meta*-notation and can be removed by syntactic translation. In such a circumstance the simple approach is satisfactory. As we have argued strongly, it is important that schema expressions reside in the object language so that the syntactic translations become derivable equalities. In this regime the definition we provide is essential.

6.4 Schema composition and piping

We proceed by adopting a standard definition of schema composition, in terms of renaming, conjunction and hiding (see *e.g.* [Dil94]). For simplicity we will restrict the composition along a single pair of complementary labels. It will then be easy to see how this might be generalised to a set of such pairs.

We shall need to index the composition operator with the pair of labels along which the composition is taken. That is:

$$S ;_{(l',l)} S' =_{df} (S[l' \leftarrow v] \wedge S'[l \leftarrow v]) \setminus (v)$$

For notational simplicity it is sensible to define a derived operator on types:

$$T ;_{(l',l)} T' =_{df} (T[l'/v] \vee T'[l/v]) \setminus (v)$$

First we have the typing rule, which is easily derived:

$$\frac{S : \mathbb{P}\, T \quad S' : \mathbb{P}\, T'}{S ;_{(l,l')} S' : \mathbb{P}(T ;_{(l,l')} T')}$$

Then the introduction rule is:

$$\frac{\Gamma \vdash (t \upharpoonright T[l'/v])[v/l'] \in S \quad \Gamma \vdash (t \upharpoonright T'[l/v])[v/l] \in S' \quad \Gamma^- \triangleright t : T[l'/v] \vee T'[l/v]}{\Gamma \vdash t \upharpoonright T ;_{(l',l)} T' \in S ;_{(l',l)} S'} \; (S^+_;)$$

This is calculated using the rules (S^+_h), $(S^+_\wedge)$ and $(S^+_\leftarrow)$ twice.

We also obtain derived elimination rules for composition. There are two rules:

$$\frac{t : T ;_{(s_1,s_2)} T' \quad t \in S ;_{(s_1,s_2)} S'}{t \upharpoonright T \setminus (s_1) \in S \setminus (s_1)} \; (S^-_{;\,o})$$

$$\frac{t : T ;_{(s_1,s_2)} T' \quad t \in S ;_{(s_1,s_2)} S'}{t \upharpoonright T \setminus (s_2) \in S' \setminus (s_2)} \; (S^-_{;\,i})$$

The substitution rules are as expected:

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \triangleright S_2 : \mathbb{P}\, T}{\Gamma \vdash S_0 ;_{(l,l')} S_2 = S_1 ;_{(l,l')} S_2} \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \triangleright S_2 : \mathbb{P}\, T}{\Gamma \vdash S_2 ;_{(l,l')} S_0 = S_2 ;_{(l,l')} S_1}$$

Example 1. Consider the following operation:

$$Inc =_{df} [z, z' \in \mathbb{N} \mid z' = z + 1]$$

We should be able to prove that

$$\langle\!| \, z \Rrightarrow 0, z' \Rrightarrow 2 \, |\!\rangle \in Inc ;_{(z',z)} Inc$$

First we can easily show in the logic that

$$\langle\!| \, z \Rrightarrow 0, z' \Rrightarrow 1 \, |\!\rangle \in Inc$$

and

$$\langle\!| \, z \Rrightarrow 1, z' \Rrightarrow 2 \, |\!\rangle \in Inc$$

Taking t to be the binding: $\langle\!| \, z \Rrightarrow 0, v \Rrightarrow 1, z' \Rrightarrow 2 \, |\!\rangle$ it only remains, by rule $(S^+_;)$ above, to show three equalities. In this case, $[z, z' : \mathbb{N}] ;_{(z,z')} [z, z' : \mathbb{N}]$ is just $[z, z' : \mathbb{N}]$. So we have to show that:

$$t \upharpoonright [z, z' : \mathbb{N}] = \langle\!| \, z \Rrightarrow 0, z' \Rrightarrow 2 \, |\!\rangle$$

$$(t \upharpoonright [z, v : \mathbb{N}])[v/z'] = \langle\!| \, z \Rrightarrow 0, z' \Rrightarrow 1 \, |\!\rangle$$

and

$$(t \upharpoonright [v, z' : \mathbb{N}])[v/z] = \langle\!| \, z \Rrightarrow 1, z' \Rrightarrow 2 \, |\!\rangle$$

but these are also easily proved.

Turning now to piping, we immediately see that the definition, and therefore the rules, are much the same. All we require is to select our complementary labels to be distinguished by the diacritical marks which indicate input and output. For example, the composition operator we have introduced above, indexed by a pair of labels $(l!, l?)$ implements the piping operator over a single input/output channel. Again, the generalisation to permit an arbitrary number of input/output channels is easy once the composition operator is similarly extended.

6.5 Schema restriction

In view of our filtering operation on terms which we have extended to sets, we can give a pleasant definition:

$$S_{\mathbb{P}\,T} \upharpoonright S'_{\mathbb{P}\,T'} =_{df} S \upharpoonright \mathbb{P}\,T' \wedge S'$$

when $T \preceq T'$ It is, then, easy to see that this collapses to our extension of filtered terms to sets when the schema S' is just a schema *type*.

This is a little less general then the standard definition (*e.g.* [Spi92] p. 34), though arguably what is required, since the standard definition permits T' to introduce new components and this is, perhaps, slightly odd for a *restriction* operation. In order to work with the standard definition we could use the general hiding operation over schema types in a manner similar to that employed below in section 6.6, but we shall not give the details here.

The rules are then just a special case of those for conjunction. Using rules $(Z_\wedge)$ and $(Z_{\mathbb{P}\upharpoonright})$ we obtain the type rule:

$$\frac{\Gamma \rhd S : \mathbb{P}\,T \quad \Gamma \rhd S' : \mathbb{P}\,T' \quad T' \preceq T}{\Gamma \rhd S \upharpoonright S' : \mathbb{P}\,T'}$$

The introduction and elimination rules are then as follows:

$$\frac{\Gamma \vdash t \in S \quad \Gamma \vdash t \upharpoonright T' \in S' \quad T' \preceq T}{\Gamma \vdash t \upharpoonright T' \in S \upharpoonright S'}\ (S^+_\upharpoonright)$$

This follows by rules $(S^+_\wedge)$ and $(\in_\upharpoonright)$, noting that $T' = T' \vee T'$ and $T = T \vee T'$.

$$\frac{\Gamma \vdash t \in S \upharpoonright S'}{\Gamma \vdash t \in S \upharpoonright T'}\ (S^-_{\upharpoonright_0}) \qquad \frac{\Gamma \vdash t \in S \upharpoonright S'}{\Gamma \vdash t \in S'}\ (S^-_{\upharpoonright_1})$$

These follow directly from the rules $(S^+_{\wedge_0})$ and $(S_{\wedge_1})$ noting that $t_T = t \upharpoonright T$.

The substitution rules are:

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\,T}{\Gamma \vdash S_0 \upharpoonright S_2 = S_1 \upharpoonright S_2} \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\,T}{\Gamma \vdash S_2 \upharpoonright S_0 = S_2 \upharpoonright S_1}$$

6.6 Schema level hiding

Our basic hiding operation takes a single label as an argument and, as we explained earlier, does duty for what, in other accounts, is a simple form of schema existential quantification. In those accounts one also finds quantification over schema in the category of schema expressions, for example [Spi92] p. 76. We should provide, within our framework, a form of schema level hiding to correspond to this. This kind of operation has turned out to be of considerable value in the structuring of Z specifications. [WD96] provides some excellent examples of operation *promotion* (*ibid.* chapter 13, see *e.g.* p. 187) which utilise this operation in order to promote an operation on a simple state to an operator on a global state of which it is a component. We shall return to this application in section 8.

The definition is quite simple. In view of earlier infrastructure we can define this easily using schema conjunction and restriction:

$$S_{\mathbb{P}\,T} \setminus S'_{\mathbb{P}\,T'} =_{df} (S \wedge S') \upharpoonright \mathbb{P}(T \setminus T')$$

Using rules $(Z_\wedge)$, $(Z_{\mathbb{P}\upharpoonright})$, together with the fact that $T' \preceq T$, we obtain the following type rule:

$$\frac{\Gamma \rhd S : \mathbb{P}\,T \quad \Gamma \rhd S' : \mathbb{P}\,T' \quad T' \preceq T}{\Gamma \rhd S \setminus S' : \mathbb{P}(T \setminus T')}$$

The introduction rule is calculated using rules $(S^+_\upharpoonright)$, $(S^+_\wedge)$ and the fact that $T \setminus T' \preceq T$.

$$\frac{\Gamma \vdash t \in S \quad \Gamma \vdash t \upharpoonright T' \in S' \quad T' \preceq T}{\Gamma \vdash t \upharpoonright (T \setminus T') \in S \setminus S'}$$

The elimination rule is obtained using rule $(\in^-_\upharpoonright)$:

$$\frac{\Gamma \vdash t \in S \setminus S' \quad \Gamma^-, x : T;\ \Gamma^+, x \in S, x \upharpoonright T' \in S', x \upharpoonright (T \setminus T') = t \vdash P \quad \Gamma^- \rhd t : T \setminus T'}{\Gamma \vdash P}$$

There is a useful equational rule for schema level hiding. This may be compared with the syntactic characterisation of (a simpler form of) schema existential quantification which is given in [WD96] (p. 178). Because we cannot quantify over constants we have some notational complications when describing this equation at the level of schema.

Let $D' = \cdots l_i \cdots$ and $\sigma = [\cdots l_i \cdots / \cdots z_i \cdots]$ where the z_i are fresh variables.

$$\frac{\Gamma^- \rhd [D \mid P] : \mathbb{P}\, T \quad \Gamma^- \rhd [D' \mid P'] : \mathbb{P}\, T'}{\Gamma \vdash [D \mid P] \setminus [D' \mid P'] = [D \setminus D' \mid \exists\, D'\sigma \bullet (P \wedge P')\sigma]}$$

The substitution rules are:

$$\frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_0 \setminus S_2 = S_1 \setminus S_2} \qquad \frac{\Gamma \vdash S_0 = S_1 \quad \Gamma^- \rhd S_2 : \mathbb{P}\, T}{\Gamma \vdash S_2 \setminus S_0 = S_2 \setminus S_1}$$

6.7 Definite description

Although definite descriptions nominally appear in Z as *terms* it is clear, from those sources which provide a logic for Z, that these terms can be understood to appear syncategorematically: the rules in [WD96] and [Nic95] are expressed in terms of equality propositions of the form:

$$\mu\, x : T \bullet P = t$$

Further evidence for this being the correct approach comes from [Spi88] in which the author remarks that (the meta-theory of) Z can be modelled within ZF set-theory *without* the axiom of choice. The salient point being that ZF with an *explicit* operator for definite descriptions (such as Hilbert's epsilon or Russell's iota) would *imply* the axiom of choice (see *e.g.* [Lei69]).

The characteristic formula for definite descriptions, $\mu\, x \in C \bullet P = t$, is translated into Z by means of:

$$\mu\, x \in C \bullet P = t =_{df} (\exists_1\, x \in C \bullet P) \wedge P[x/t]$$

and then all references to such terms may be removed by the following contextual definition into Z:

$$P_0[z/\mu\, x \in C \bullet P_1] =_{df} \exists\, z \in C \bullet \mu\, x \in C \bullet P_1 = z \wedge P_0$$

which simplifies to:

$$P_0[z/\mu\, x \in C \bullet P_1] =_{df} \exists_1\, z \in C \bullet P_1[x/z] \wedge P_0$$

Given the definition we easily obtain the following derived rules of typing and inference:

$$\frac{C : \mathbb{P}\, T \quad z : T \rhd P \ prop}{\mu\, z \in C \bullet P : T}\ (Z_\mu)$$

$$\frac{\exists_1\, z \in C \bullet P \quad t \in C \quad P[z/t]}{\mu\, z \in C \bullet P = t}\ (\mu^+) \qquad \frac{\mu\, z \in C \bullet P = t}{t \in C}\ (\mu_0^-) \qquad \frac{\mu\, z \in C \bullet P = t}{P[z/t]}\ (\mu_1^-)$$

6.8 Conditional terms

With definite descriptions in hand we have a method for interpreting the *conditional terms* often employed in example Z specifications.

$$if\ P\ then\ t_T\ else\ t'_T =_{df} \mu\, x \in T \bullet (P \Rightarrow x = t) \wedge (\neg P \Rightarrow x = t')$$

The following type assignment rule is then derivable using rules (Z_μ), $(Z_\wedge)$, $(Z_\Rightarrow)$, $(Z_=)$, $(Z_\neg)$, lemmata 16(i) and 19:

$$\frac{\Gamma \rhd P \ prop \quad \Gamma \rhd t_0 : T \quad \Gamma \rhd t_1 : T}{\Gamma \rhd if\ P\ then\ t_0\ else\ t_1 : T}$$

The following introduction rule can be derived, using the law of the excluded middle for the proposition P:

$$\frac{P \vdash t_0 = t_2 \quad \neg P \vdash t_1 = t_2}{if\ P\ then\ t_0\ else\ t_1 = t_2}$$

6.9 Generic schema

A generic schema is parameterised over one or more types. These are very easily accommodated within our regime. We will permit an extension of our language of types to include type variables:

$$T ::= \cdots \mid X$$

Then we may introduce a new category of generic schema:

$$GS ::= S[X]$$

Finally we extend the language of schema expressions to include instantiated generic schema:

$$S ::= \cdots \mid S[X := T]$$

where T is a closed type.

Such instantiated schema are interpreted into Z by means of:

$$S[X := T] =_{df} S[X/T]$$

The following rules are then immediate:

$$\frac{P[\alpha[D]/t.\alpha[D]] \quad t \in [D][X/T]}{t \in [D \mid P][X := T]} \qquad \frac{t \in [D \mid P][X := T]}{P[\alpha[D]/t.\alpha[D]]} \qquad \frac{t \in [D \mid P][X := T]}{t \in [D][X/T]}$$

6.10 Alternative forms of quantification

There are several different forms of quantification which are adopted in the literature on Z. We shall, in this short section, only attempt to develop those alternatives presented in one of the normative sources: [Spi92].

The basic form of existential quantification ([Spi92] p. 70) is:

$$\exists S \bullet P$$

We shall interpret this by means of the following definitional extension[12]:

$$\exists S \bullet P =_{df} \exists z \in S \bullet P[\alpha S/z.\alpha S]$$

As a consequence we would then induce the following rules:

$$\frac{S : \mathbb{P}\, T \quad z : T \triangleright P[\alpha S/z.\alpha S]\ prop}{\exists S \bullet P\ prop}$$

$$\frac{P[\alpha S/t.\alpha S] \quad t \in S}{\exists S \bullet P} \qquad \frac{\Gamma \vdash \exists S \bullet P \quad \Gamma^- \ S : \mathbb{P}\, T \quad \Gamma^-, y : T;\ \Gamma^+, P[\alpha S/y.\alpha S] \vdash P'}{P'}$$

The basic forms for λ-expressions and definite descriptions in [Spi92] (p. 58) are:

$$\lambda S \bullet t$$

[12] What we develop, in this section, are forms of expression which are *analogous* to those one can find in the literature. They do not, however, constitute (even part of) an interpretation of the Z one can find there, into our version. For example, in standard Z one has the θ-operator; an operation which we have, until now, studiously ignored. If we assume that such expressions *can* occur in the propositions under consideration we would, perhaps adopt a more general definition, for example:

$$\exists S \bullet P =_{df} \exists z \in S \bullet P[\alpha S/z.\alpha S][\theta S/z]$$

However, even this would not constitute a full translation because of the interaction between the θ-operator and schema *priming*. We have no interest in interpreting, in its entirety, the whole of standard Z since our contribution here is both technical exposition *and critique*. We discuss these issues in fuller detail below, in section 7.

and

$$\mu S \bullet P$$

We shall omit the translation of these (and their induced rules) since they follow the pattern we have just presented for the existential quantifier and are easily calculated by analogy.

The primitive form for set comprehension in [Spi92] (p. 57) is:

$$\{S \bullet t\}$$

We interpret this by means of the following definition:

$$\{S \bullet t_T\} =_{df} \{z \in T \mid \exists S \bullet z = t\}$$

The type assignment rule:

$$\frac{S : \mathbb{P}\, T' \quad x : T' \rhd t[\alpha S / x.\alpha S] : T}{\{S \bullet t\} : \mathbb{P}\, T}$$

The logical rules for this form of set comprehension are:

$$\frac{S \in \mathbb{P}\, T' \quad z : T \vdash t = t'[\alpha S / z.\alpha S]}{t \in \{S \bullet t'\}}$$

$$\frac{\Gamma \vdash t \in \{S \bullet t'\} \quad \Gamma^- \rhd S : \mathbb{P}\, T' \quad \Gamma^-, y : T';\ \Gamma^+, t = t'[\alpha S / y.\alpha S] \vdash P'}{P'}$$

6.11 The mathematical toolkit

We have said nothing about functions, sequences, bags *etc.* and the notation and operations which correspond to them. The reason for this is that these remaining features of Z are already understood to be infrastructure and are provided in Z by definition. As an example recall the standard definition of sequences. In our notation this would be:

$$\mathrm{seq}\, T =_{df} \{f \in \mathbb{N} \pfun T \mid \mathrm{dom}\, f = 1 \cdots \# f\}$$

which itself requires the infrastructure for finite partial functions; which in turn is defined (*e.g.* [Spi92] (p. 112)) in terms of partial functions *etc.* The corresponding display form $< \cdots >$ is of type $\mathbb{P}(\mathbb{N} \times T)$ and so is interpreted in terms of the display form for tuples. These details, and all the others, are completely covered in *e.g.* [Spi92].

6.12 Organisation of specifications

Z provides mechanisms for the overall organisation of specificaitons into paragraphs and sections. Where rules for these have been provided they have turned out to be among the most complex required, and are clearly the result of much ingenuity. In [Toy97] (p. 43) there are typechecking rules for sections and paragraphs which are enormously complex and which require extremely baroque side-conditions. At the very least these rules establish a useful basis for further work. The complications occurring in the rule for sections arise because each component induces a context which subsequent components inherit. It remains to be seen to what extent these complications are tamed by treating schema components as constants rather than variables. But it is a topic we shall have to leave for future research.

7 Priming and the theta operator

There are two operations, very commonly utilised in Z specifications, which we have, until now, avoided entirely. In this section we shall explain why and demonstrate the mathematical problems which they, jointly, cause. Following this we will describe an alternative means by which the services they are meant to provide can be presented, with the added advantage that the formalisation is relatively simple, comprehensible and, consequently, usable[13].

There are two competing perspectives on schema in Z, as it is currently understood, which are mutually incompatible. The older view is that a schema is a "piece of mathematical text" ([WD96] p. 148) or the description of a state (*e.g.* [She95] p. 202). The more recent, dating roughly from the time when schema became routinely used as sets and the theta operator was introduced, is that a schema is a "set of bindings" [WD96] (p. 156) or "collection of possible values" [She95] (p. 199)[14]. The most striking example of this appears in [Dil94] (pp. 46-7), where, within two paragraphs, the author gives *both* accounts of schema.

> "$\cdots$ Schemas are used $\cdots$ to make precise what the *state space* of a schema is. The state space is defined by means of a state schema." ([Dil94], section 4.3.4, p. 46. Our emphasis.) "*PhoneDB* is the name of a schema which represents a before *state*. Decorating the name with a prime, for example *PhoneDB'*, represents the after *state*." ([Dil94], section 4.3.4, p. 47. Our emphasis.)

The older perspective accounts for the use of schema priming: if S is a schema representing the before state (singular), then S' represents the after state. The notion of the Δ-schema is paradigmatic of this view. It is somewhat surprising to discover that the Ξ-schema is paradigmatic of the alternative perspective. To see this we must first see what goes wrong when we attempt understand such schema from the older perspective. Consider the schema $\Xi S =_{df} [\Delta S \mid \theta S = \theta S']$. It is very well-known that in the context of the definition $T =_{df} S'$ the schema $[S;\ T \mid \theta S = \theta T]$ is not even well-typed *a fortiori* not equal to ΞS. But instead of tracing this unfortunate observation back to the root cause (the clash of perspectives we have introduced) a range of mathematically unpleasant manoeuvres have taken place in order to accommodate the situation[15]. The problem is, of course, that the type of S' is not the type required: we need the type of S here. Indeed we do: the Ξ-schema is intended to link the initial state and the final state and these, under the *second* perspective, are both elements of S. It appears that the θ operation is inextricably linked with this second perspective. But from this viewpoint the Δ-schema is incomprehensible, for it appears to suggest that operations change specifications of states (state spaces) rather than states. The solution to all this must begin by reconciling these pre-theoretic contradictions.

The older perspective, that schema are states, is highly syntactic and it is linked with interpretations of the notation which are essentially based on macro-expansion. These have no, or very limited, mathematical properties. Moreover, this view is incompatible with almost all of the innovative work on Z which has taken place during the last seven or eight years, much of which has been introduced as a result of applying Z to realistic examples. In particular, the greater role for schema, as first-class entities, presupposes that

[13] It should not be underestimated how important simplicity is to the enterprise we are considering. That the non-existence of complete formal apparatus is unacceptable requires no argument. However, complex formalisation is only marginally to be preferred. For example, although a system of rules with inscrutable side conditions is, perhaps, implementable, it is most difficult to comprehend. The purpose of formalisation is only in part to provide automatic or semi-automatic means for reasoning; it is also the means by which we may understand, jointly, and indeed communicate to others, the meaning of the system. It is vital that the notions can be expressed in the simplest possible manner. In this regard the algebraic principles of compositionality and referential transparency are to be prized. In contrast, syntactic intensionality, the failure of substitutivity, non-compositional analyses (all of which are commonly used to describe Z) are to be avoided at all cost.

[14] Surprisingly, many authors barely describe a schema beyond describing the features of its concrete syntax, although some (*e.g.* [MP93] p. 80) hint at a similarity to a structure definition in a programming language. Although informal, this hint is consistent with the second perspective: a schema describes a collection of values of a particular kind.

[15] In order to prevent Leibniz's principle from failing one must ensure that the expression $\theta S'$ is not the application of θ to the schema S' which ensures that the substitution is invalid. But this may not be enough: generally, θ may only be applied to schema *names*. This has the effect of making θ a highly intensional operation. These devices may prevent ambiguity and avoid the incoherence of a failure of Leibniz's principle, but they do so by technical means which are complex and unwieldy, making formalisation extremely difficult, and even if achieved, of limited value.

they represent specifications of collections and not specifications of individuals[16]. From this perspective it is easy to render the Ξ-schema by means of $\Xi S =_{df} [z, z' \in S \mid z = z']$. Note that it is now quite clear that S describes the *set* of states over which the operation computes, and the before and after states both conform to that specification. As a result the type of the equality is preserved naturally, without resort to dubious technical tricks. The Δ-schema is now best thought of as a declaration and not as a schema at all: $\Delta S =_{df} z, z' \in S$.

So far as the theta operation is concerned, we have not needed to employ it in the definition of the Ξ-schema because, instead of *including* a schema, we have introduced a declaration over the schema as a set. But this approach can be taken whenever the theta operation is normally required. It is a natural corollary of adding schema as sets to Z in the systematic fashion we are advocating: the operation θ has no role to play.

7.1 Latent declarations

We have argued that we should remove the concepts of schema priming and the theta operator on conceptual and mathematical grounds. We must then investigate whether or not the language remains expressive enough for its purposes. Certainly there is a change of style. Adapting existing Z specifications to our revised framework requires some care: when the theta operator is useful in standard Z we would introduce a declaration of schema type, where, most often, the standard Z would invoke a schema inclusion.

This approach, though, can require more explicit use of binding projection in specifications written in our system. Compare, for example, the following pair in standard Z ([WD96], p. 175) and then our revised language.

$Return_0$

$\Delta BoxOffice$
$s? : Seat$
$c? : Customer$

$s? \mapsto c? \in sold$
$sold' = sold \setminus \{s? \mapsto c?\}$
$seating' = seating$

$Return_0$

$b, b' \in BoxOffice$
$s? \in Seat$
$c? \in Customer$

$s? \mapsto c? \in b.sold$
$b'.sold = b.sold \setminus \{s? \mapsto c?\}$
$b'.seating = b.seating$

The notational burden is rather similar to that one can encounter in programs which manipulate structured data. In Pascal, for example, one has the "with" idiom to aid presentation. A generalisation of this seems called for here.

We shall permit, as prime declarations, a new form which we will call *latent declarations*. These are written:

$$(l\xi \in)S$$

where ξ is a, possibly absent, diacritical mark (prime, subscript *etc.*). Notice that we restrict the use of this idiom to *schemas* only: its purpose is to ameliorate the inexpressivity of our revision of Z which accrues because of the occasional replacement of schema inclusion by a declaration, and there is nothing to be gained by making it more general than absolutely necessary.

[16] However, the intended model is classical, *extensional* set theory and this means that many, deterministic, operations denote collections of cardinality 1. In other words, in certain cases, there is a one-one correspondence between a schema as a set and its elements. This opens up an enormous area for debate which calls into question the intended model and its suitability. It is another story we aim to tell in the future.

The idea is that one may, in the context of this declaration, refer to the components of S directly. On the other hand l is available, if necessary, when one would conventionally require the θ-operator.

We can translate such a novelty into Z by means of:

$$[\cdots (l\xi \in) S_{\mathbb{P}\,T} \cdots \mid P] =_{df} [\cdots l\xi \in S \cdots \mid P[\alpha T\xi / l.\alpha T]]$$

The diacritical mark ξ plays a crucial role. It is perfectly possible (indeed highly likely in view of the inclusion of Δ-schemas in operations) that a schema is effectively included twice in our version of Z. Consequently, these marks, which in standard Z refer to distinct components in distinct schema, allow us to determine to which declaration the component belongs.

In the presence of this syntactic device we can write the schema above as:

Return$_0$

$(b, b' \in) BoxOffice$
$s? \in Seat$
$c? \in Customer$

$s? \mapsto c? \in sold$
$sold' = sold \setminus \{s? \mapsto c?\}$
$seating' = seating$

This is not significantly different from the standard presentation.

Additionally, we make use of the latently declared components at the same time as suppressing their appearance elsewhere. For example, in standard Z we might have ([WD96] p. 193):

Promote

$\Delta Array$
$\Delta Data$
$index? : \mathbb{N}$

$index? \in \mathrm{dom}\, array$
$\{index?\} \lhd array = \{index?\} \lhd array'$
$array\ index? = \theta Data$
$array'\ index? = \theta Data'$

In our revised language this could now appear as:

Promote

$(a, a' \in) Array$
$(d, d' \in) Data$
$index? \in \mathbb{N}$

$index? \in \mathrm{dom}\, array$
$\{index?\} \lhd array = \{index?\} \lhd array'$
$array\ index? = d$
$array'\ index? = d'$

It is, perhaps, important to reinforce the point that our framework is likely to impose some differences in the *style* of specification. In particular, in evaluating our proposals with standard Z one must guard against assuming that simply transliterating existing specifications is the correct point of comparison. The following example demonstrates that one might approach a problem in quite a different way. The technique we shall illustrate is described in [Bow96] from which the example is adapted.

Example 2. The objective is to define a form of Ξ-schema which ensures that only some of the state components are invariant across a state change. Consider:

S

$a, b, c : \mathbb{N}$

Taking ΔS and ΞS as usual we define:

$$\Phi(z) =_{df} \Delta S \wedge (\Xi S \setminus (z))$$

This may seem somewhat inscrutable. However, calculation reveals that $\Phi(a) =_{df}$

ΔS
$b' = b$
$c' = c$

In other words $\Phi(a)$ is the same as ΞS except that one component of S (the component a) is *not* held invariant. Whereas we could represent this directly in our version of Z we might observe that the following is possible: $\Phi[X] =_{df}$

$s, s' \in S$
$s \upharpoonright X = s' \upharpoonright X$

Then the schema $\Phi(a)$ above would be written as $\Phi[b, c : \mathbb{N}]$.

Although we might wish to argue that this is much clearer, this is not our purpose here. The point at issue is that it is a complex matter to determine the relative expressive merits of standard Z and our revision, because each language determines its own natural styles. This is well worth exploring in much more detail in the future. We shall make some further comments in section 9.

8 Example

We shall not try to be over ambitious and will, by no means, attempt encyclopaedic coverage of Z specification techniques in this section. It will certainly remain to be seen whether or not what we have established as a revised Z meets the demands of practice. We would hope, at the very least, that the existence of a complete mathematical framework will encourage others to experiment.

Let us, at least, consider a reasonable example from the literature. This concerns the technique of *promotion* (see [WD96] chapter 13). The example taken from this chapter (pp. 186-7) concerns the promotion of an operation over a local state to an operation over a global state. This is Z at its very best: providing a general organising strategy which structures a specification. First we present the example as it stands in the book.

LocalScore

$s : \mathbb{P}\ Colour$

GlobalScore

$score : Players \pfun LocalScore$

AnswerLocal

$\Delta LocalScore$ $c? : Colour$
$s' = s \cup \{c?\}$

Promote

$\Delta GlobalScore$
$\Delta LocalScore$
$p? : Player$

$p? \in \mathrm{dom}\ score$
$\theta LocalScore = scorep?$
$score' = score \oplus \{p? \mapsto \theta LocalScore'\}$

Then the specification of *AnswerGlobal*, the operation over the global state, is given by:

$$\exists \Delta LocalScore \bullet AnswerLocal \wedge Promote$$

This last definition is an instance of a schema for promoting operations in this manner.
In our presentation this would be rewritten as follows:

LocalScore

$s \in \mathbb{P}\ Colour$

GlobalScore

$score \in Players \nrightarrow LocalScore$

AnswerLocal

$(l, l' \in) LocalScore$
$c? \in Colour$

$s' = s \cup \{c?\}$

Promote

$(g, g' \in) GlobalScore$
$l, l' \in LocalScore$
$p? : Player$

$p? \in \mathrm{dom}\ score$
$l = score\ p?$
$score' = score \oplus \{p? \mapsto l'\}$

Then the specification of *AnswerGlobal* the operation over the global state is then given by:

$$(AnswerLocal \wedge Promote) \setminus [l, l' \in LocalScore]$$

What confidence can we have that the schemas we have defined are the intended interpretation? Since our operators are not defined by syntactic transformation we cannot undertake the simplification of [WD96] p. 188 which demonstrates that $\exists \Delta LocalScore \bullet AnswerLocal \wedge Promote$ is equivalent to:

AnswerGlobal

$\Delta GlobalScore$
$p? : Player$
$c? : colour$

$p? \in \mathrm{dom}\ score$
$\{p?\} \lhd score' = \{p?\} \lhd score$
$(score'\ p?).s = (score\ p?).s \cup \{c?\}$

However, we have more or less the same apparatus in another guise: each of the syntactic transformations in the text-book have become instances of provable equalities in our Z logic. Putting together the

various lemmata for the schema expressions from the technical development has established an equational logic for reasoning about schemas.

The first stage is to remove the latent declarations.

AnswerLocal

$$l, l' \in LocalScore$$
$$c? \in Colour$$
$$\hline$$
$$l'.s = l.s \cup \{c?\}$$

Promote

$$g, g' \in GlobalScore$$
$$l, l' \in LocalScore$$
$$p? : Player$$
$$\hline$$
$$p? \in \mathrm{dom}\, score$$
$$l = g.score\ p?$$
$$g'.score = g.score \oplus \{p? \mapsto l'\}$$

Next, since our equations always require the D^* form of declarations, we clearly have to use the rule $(\in^=)$ on *GlobalScore* since its declaration part is not of the right form.

$GlobalScore =_{(\in^=)}$

GlobalScore$_0$

$$score \in \mathbb{P}(Players \times [s \in \mathbb{P}\, Colour])$$
$$\hline$$
$$score \in partial(Players, [s \in \mathbb{P}\, Colour])$$

where $partial(A, B) =_{df} \{f \in \mathbb{P}(A \times B) \mid \forall x \in A \bullet \forall a, b \in B \bullet (x \mapsto a \in F \wedge x \mapsto b \in f) \Rightarrow a = b\}$.

We can now substitute this for *GlobalScore* in *Promote*, and then, in turn, we can equate *Promote* with a schema whose declaration part is in the D^* form.

$Promote =_{(sub,\in^=)}$

Promote$_0$

$$g, g' \in [score \in \mathbb{P}(Players \times [s \in \mathbb{P}\, Colour])]$$
$$l, l' \in LocalScore$$
$$p? \in Player$$
$$\hline$$
$$p? \in \mathrm{dom}\, score$$
$$l = g.score\ p?$$
$$g'.score = g.score \oplus \{p? \mapsto l'\}$$
$$g, g' \in GlobalScore_0$$

We now proceed to the conjunction:

$$AnswerLocal \wedge Promote =_{(sub)} AnswerLocal \wedge Promote_0 =_{(\wedge^=)}$$

AG

$$g, g' \in [score \in \mathbb{P}(Players \times [s \in \mathbb{P}\, Colour])]$$
$$l, l' \in LocalScore$$
$$c? \in Colour$$
$$p? \in Player$$
$$\hline$$
$$l'.s = l.s \cup \{c?\}$$
$$p? \in \mathrm{dom}\, g.score$$
$$l = g.score\ p?$$
$$g'.score = g.score \oplus \{p? \mapsto l'\}$$
$$g, g' \in GlobalScore_0$$

Then, by substitution, we have $AnswerLocal \land Promote_0 \setminus [l, l' \in LocalScore] =_{(sub)} AG \setminus [l, l' \in LocalScore]$, and then, by the equality rule for hiding, $AG \setminus [l, l' \in LocalScore] =_{(\setminus^=)}$

AnswerGlobal$_0$

$g, g' \in [score \in \mathbb{P}(Players \times [s \in \mathbb{P}\, Colour])]$
$c? \in Colour$
$p? \in Player$

$\exists z, z' \in LocalScore \bullet$
$(z'.s = z.s \cup \{c?\}$
$p? \in \mathrm{dom}\, g.score$
$z = g.score\ p?$
$g'.score = g.score \oplus \{p? \mapsto z'\}$
$g, g' \in GlobalScore_0)$

Note that $z'.s = z.s \cup \{c?\} \Leftrightarrow z' = (\!|\ s \Rrightarrow z.s \cup \{c?\}\ |\!)$ is easily proved in the logic.
So the predicate part of $AnswerGlobal_0$ is:

$\exists z, z' \in LocalScore \bullet$
$(z' = (\!|\ s \Rrightarrow z.s \cup \{c?\}\ |\!)$
$p? \in \mathrm{dom}\, g.score$
$z = g.score\ p?$
$g'.score = g.score \oplus \{p? \mapsto z'\}$
$g, g' \in GlobalScore_0)$

By the one-point rule, on the first equation, we have:

$\exists z \in LocalScore \bullet$
$p? \in \mathrm{dom}\, g.score$
$z = g.score\ p?$
$g'.score = g.score \oplus \{p? \mapsto (\!|\ s \Rrightarrow z.s \cup \{c?\}\ |\!)\}$
$g, g' \in GlobalScore_0)$

and again on the second equation gives:

$p? \in \mathrm{dom}\, g.score$
$g'.score = g.score \oplus \{p? \mapsto (\!|\ s \Rrightarrow (g.score\ p?).s \cup \{c?\}\ |\!)\}$
$g, g' \in GlobalScore_0)$

This then yields, by substitution:

AnswerGlobal$_2$

$g, g' \in [score \in \mathbb{P}(Players \times [s \in \mathbb{P}\, Colour])]$
$c? \in Colour$
$p? \in Player$

$p? \in \mathrm{dom}\, g.score$
$g'.score = g.score \oplus \{p? \mapsto (\!|\ s \Rrightarrow (g.score\ p?).s \cup \{c?\}\ |\!)\}$
$g, g' \in GlobalScore_0)$

Now, using $(\in^=)$ again (right to left) we can undo the manipulations on *GlobalScore* we began with:

AnswerGlobal$_3$

$g, g' \in GlobalScore$
$c? \in Colour$
$p? \in Player$

$p? \in \mathrm{dom}\, g.score$
$g'.score = g.score \oplus \{p? \mapsto (\!|\ s \Rrightarrow (g.score\ p?).s \cup \{c?\}\ |\!)\}$

Rewriting the second equality using the same argument as [WD96][17] we then have:

AnswerGlobal$_4$

$g, g' \in GlobalScore$
$c? \in Colour$
$p? \in Player$

$p? \in \operatorname{dom} g.score$
$\{p?\} \lhd g'.score = \{p?\} \lhd g.score$
$(g'.score\ p?).s = (g.score\ p?).s \cup \{c?\}$

Then re-introducing latent declarations, we finally obtain:

AnswerGlobal

$(g, g' \in) GlobalScore$
$c? \in Colour$
$p? \in Player$

$p? \in \operatorname{dom} score$
$\{p?\} \lhd score' = \{p?\} \lhd score$
$(score'\ p?).s = (score\ p?).s \cup \{c?\}$

This is precisely the natural transliteration of the *AnswerGlobal* schema which is given in [WD96] into our version of Z.

9 Conclusions and future work

The purpose of this paper was twofold. Most crucially, we wished to provide the language Z within the context of a useful mathematical framework, thus establishing Z as a specification logic. A secondary aim has been a critique of the Z language which has become established in the literature. These two trajectories are linked. Whilst it would have been entirely possible to outline many of the conceptual conundrums which Z poses in a discursive style (and it must be said that almost everything we have said is known and shared by various workers in the Z research community) we have been determined to allow the mathematics to take the lead. As is very often the case, a mathematical approach does more than formalise; it additionally highlights areas of confusion and complexity. Consequently, we have used mathematical criteria to produce, not only a formal account but, a simple and (ultimately, we hope) useable account which retains the major benefits which Z offers: expressibility and scalability.

We have attempted to be reasonably comprehensive and have addressed, if in places in only in outline, most of the major areas of the Z language. However, much remains to be done. We should like, in future publications, to develop and extend the work we have begun here on the schema calculus, and as we have mentioned, explore the organisation of specifications at the level of sections. In addition we wish to pursue program development in the context of the specification logic we have established. In particular, we are very interested in exploring other semantic foundations for Z based on a constructive intensional set theory and to compare this with the traditional model based as it is on classical extensional set theory. It would not be appropriate to outline the reasons for this here although we hint at the issues in section 7 particularly in its final footnote.

Finally, as we acknowledge in section 7.1, our revised framework requires a significant change in *style* and a significant investigation in which existing strategies are re-expressed must be undertaken. The results of such an investigation must then be used to evaluate and modify our approach. Such an interplay between theory and practice is vital. It is also not clear how the revised language interacts with work on program development. From our point of view this is not a concern for, as we indicated in the previous paragraph, we aim to address this topic by replacing the standard classical, extensional model with an intensional and constructive model. However, there are clearly interesting avenues to explore which utilise more conventional mechanisms. In order to investigate any of the these topics deeply, it would be very useful to use the systems provided here as the basis for a proof development tool. Work on this has already begun [Völ98], though much remains to be achieved.

[17] We have not developed the mathematical toolkit explicitly (see section 6.11) as it is parasitic on what we have presented. The features of this obviously obey the usual rules.

10 Acknowledgements

We would like to thank the Department of Computer Science at the University of Waikato, New Zealand, the Centre for Discrete Mathematics and Theoretical Computer Science, New Zealand, the Royal Society of Great Britain, and the EPSRC (grant number GR/L57913) for financial assistance which has supported the development of this research. We are most grateful to Lindsay Groves, Ray Turner and Mark Utting for many useful discussions.

References

[BJ95] L. Bottaci and J. Jones. *Formal specification using Z: a modelling approach.* Thompson, 1995.

[Bow96] J. Bowen. *Formal specification and documentation using Z.* International Thompson Computer Press, 1996.

[Bri95] S. Brien. A model and logic for generically typed set theory (z). Technical report, (draft) D. Phil. thesis, University of Oxford, 1995.

[Dil90] A. Diller. *Z: An introduction to formal methods.* J. Wiley and Sons, 1990.

[Dil94] A. Diller. *Z: An introduction to formal methods (2^{nd} ed.).* J. Wiley and Sons, 1994.

[Har96] A. Harry. *Formal Methods fact file: VDM and Z.* Wiley, 1996.

[Hay87] I. Hayes. *Specification case studies.* Prentice Hall, 1987.

[HM97] J. Hall and A. Martin. $\mathcal{W}$ reconstructed. In *Proceedings ZUM '97, LNCS 1212*, pages 115–134. Springer, 1997.

[Jac97] J. Jacky. *The way of Z: Practical programming with formal methods.* Cambridge University Press, 1997.

[KSW96] Kolyang, B. Santen, and B. Wolff. A structure preserving encoding of Z in isabelle/hol. In *Proceedings Formal Methods Europe.* LNCS Vol. 1051, Springer Verlag, 1996.

[Lei69] A. C. Leisenring. *Mathematical logic and Hilbert's ϵ-symbol.* McDonald Technical and Scientific, 1969.

[Mar97] A. Martin. Approaches to proof in z. Technical report, Technical Report 97-34, University of Queensland, 1997.

[Mar98] A. Martin. Private communication, 1998.

[MP93] M. McMorran and S. Powell. *Z: Guide for beginners.* Blackwell Scientific, 1993.

[Nic95] J. Nicholls. *Z Notation: Version 1.2.* Z Standards Panel, 1995.

[PST96] B. Potter, J. Sinclair, and D. Till. *An introduction to formal specification and Z ($2^n d$ ed.).* Prentice Hall, 1996.

[Rat94] B. Ratcliff. *Introducing specification using Z: a practical case study approach.* McGraw-Hill International, 1994.

[She95] D. Sheppard. *An introduction to formal specification with Z and VDM.* McGraw-Hill International, 1995.

[Spi88] J. M. Spivey. *Understanding Z: A specification language and its formal semantics.* C.U.P., 1988.

[Spi92] J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall, 1992.

[Toy97] I. Toyn. *Z Notation: Draft 0.8.* Unpublished draft, 1997.

[Völ98] N. Völker. Private communication, 1998.

[WB92] J. Woodcock and S. Brien. W: A logic for z. Springer Verlag, 1992.

[WD96] J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof.* Prentice Hall, 1996.