Working Paper Series
ISSN 1170-487X

A Logic for the Schema Calculus

by Martin C Henson and Steve Reeves

Working Paper 98/5
March 1998

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

A Logic for the Schema Calculus

Martin C. Henson[1] and Steve Reeves[2]

[1]Department of Computer Science, University of Essex, U.K.; hensm@essex.ac.uk
[2]Department of Computer Science, University of Waikato, New Zealand; stever@cs.waikato.ac.nz

Abstract. In this paper we introduce and investigate a logic for the schema calculus of Z. The schema calculus is arguably the reason for Z's popularity but so far no true calculus (a sound system of rules for reasoning about schema expressions) has been given. Presentations thus far have either failed to provide a calculus (*e.g.* the draft standard [3]) or have fallen back on informal descriptions at a syntactic level (most text books *e.g.* [7]). Once the calculus is established we introduce a derived equational logic which enables us to formalise properly the informal notions of schema expression equality to be found in the literature.

1 Introduction

1.1 Background

The specification language Z has been developed over the last decade or so mainly in response to the needs of the practioners who have used it. This is the source of its undoubted strength and popularity.

Z's main technical innovation (which sets it apart from *VDM*, for example) is the idea of a *schema*. Given this structuring device, a need naturally arose for means to manipulate it: we wanted to combine schemas in various ways so as to build larger and more complicated systems from smaller and simpler ones. That is to say, we naturally wanted our specifications to follow the tried and tested (and successful) techniques of separation of concerns and divide-and-conquer.

There emerged from this pragmatic pressure what has become known as the *schema calculus*.

For example:

$$[D_0 \mid P_0] \wedge [D_1 \mid P_1] = [D_0 \oplus D_1 \mid P_0 \wedge P_1]$$

expresses (where $\oplus$—defined later—indicates a merging of the components D_0 and D_1) schema conjunction. In most text book accounts this amounts to a *definition*, which simply makes the schema expressions into meta-notational conventions for ordinary schemas. This is in conflict with the standard [3] in which schema expressions appear in the *object* language, and in which these text book definitions must be established as *provable equalities*. But, since the standard does not provide a sufficiently complete logic for Z, this proof obligation cannot be discharged and the text book approach is the best that can be achieved. It, at least, provides an opportunity to say *something* about complex

schema expressions (*e.g.* [7], pp. 194-195, of which more below in section 5), if only informally.

The rules which exist in the draft standard ([3], section F.6.6, pp. 207-208) are incomplete, too restrictive, or incorrect. The lacunae are easy to spot. Less easy is the observation that of the following rules (which are derivable from rules ($SchBindMem$) and ($SAnd$)) the first is too limited and the conclusion of the second and third are not in general even well-formed.

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

The problem (with rule ($SchBindMem$)) has been acknowledged by the community, but the suggested repair (adding a proviso to that rule) only ensures that the derived elimination rules make *sense*. They remain, like the introduction rule, too restrictive and, most importantly, cannot be used to establish the equation for conjunction with which we began. In this paper we seek to show how this state-of-affairs can be remedied.

We do this by first introducing a 'core' version of Z. This language can be extended by definition to Z; it will also be simple enough that its logic can be straightforwardly and clearly given. We call this language Z_C. In this paper we shall concentrate exclusively on the extension of Z_C to the schema calculus.

In all that follows we shall rely on the reader's informal knowledge of Z: we do not have the space to provide details of several important meta-mathematical properties (though [2] does include all of these).

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.

2.1 The language of Z_C

We begin with type names:

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

where

$$D_T ::= \cdots l_i : T_i \cdots$$

Types of the form $[D_T]$ are called *schema types*. Each type name denotes a type carrier, *i.e.* a set. The *proto-syntax*[1] of sets, in general, are given by:

$$C ::= \{z \in C \mid P\} \mid \mathbb{P}\, C \mid C \times C \mid \mathbb{N}^* \mid \cdots l_i \in C_i \cdots] \mid S$$

[1] The categories of sets, propositions and terms, which we give in this section, overgenerate. The syntax of these is finally determined by the type assignment and propositionhood system which we give in section 2.2.

where

$$D ::= \cdots l_i \in C_i \cdots$$

The carriers are picked out as T^* by:

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

where

$$D^* ::= \cdots l_i \in T_i^* \cdots$$

Sets of the form $[D^*]$ are called *schema carriers*. Declarations of the form $l \in T$ are called *prime* declarations. The labels l are *constants*. We shall write $[D_0\, T] \preceq [D_1\, T]$ when the set of prime declarations of $D_0\, T$ is a subset of that of $D_1\, T$. Other meta-operations we shall need over schema types: $[D_0\, T] \setminus [D_1\, T]$ is the schema type comprising all prime declarations of $[D_0\, T]$ which do not occur in $[D_1\, T]$. When we are interested only in a single label we will write this as $[D_T] \setminus (l : T)$. The schema type $[D_0\, T] \oplus [D_1\, T]$ is the schema type comprising the union of the prime declarations in $D_0\, T$ and $D_1\, T$. It is not defined when this union contains prime declarations $l \in T_0$ and $l \in T_1$ for which $T_0 \neq T_1$. Finally, we shall introduce meta-notational conventions which require substitution for labels. For this we need the *alphabet* operator, defined as follows: $\alpha[\cdots l_i \in T_i \cdots] =_{df} \{\cdots l_i \cdots\}$ and then we shall write $[\alpha[D_T]/t(\alpha[D_T])]$ to represent the family of substitutions: $[\cdots][l_i/t(l_i)][\cdots]$. All of these operations can clearly be extended to T^* and D^* too (remembering that : will change to $\in$ in the those cases). We will also extend α to members of D in the obvious way.

The proto-syntax of formulæ is given by:

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

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 is as follows:

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

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 $\restriction$ "filter" and its purpose is to permit the restriction of bindings to a given schema type. We shall see later how important these filtered terms are for constructing compositional interpretations for schema expressions.

The subcategory of numerals is as expected:

$$n ::= 0 \mid succ\ n$$

Finally, among the set comprehensions we shall isolate the *schema* as a special case and introduce special meta-notation for them:

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

We then have schema *expressions*:

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

Note that all such expressions are included as sets in Z_C. 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.* [7] 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.

The reader will have detected a notational shift in our presentation: we use a set membership relation and not a type assignment judgement in declarations. This is a consequence of thinking through the implications of the *types-as-sets* doctrine of Z. Since declarations in Z can utilise sets, and not just types, 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. The colon and type names only occur in judgements of the type assignment and propositionhood system for Z_C, and consequently never appear in the object language. Similarly, type carriers only appear in the object language (as sets). This is a formalisation of Spivey's analysis ([5] p. 24) and, as he also does, we will no longer need to burden the presentation by distinguishing (with the *) type names from carriers, precisely *because* the context will always determine which is intended.

Of course, the reader who is not convinced by this argument, or concerned with the details, can always remain with the traditional syntax and not lose anything of what follows.

2.2 Type assignment and propositionhood in Z_C

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

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

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

$$\frac{}{0 : \mathbb{N}}\ (C_o) \qquad \frac{n : \mathbb{N}}{succ\ n : \mathbb{N}}\ (C_s) \qquad \frac{S_0 : \mathbb{P}\, T_0 \quad S_1 : \mathbb{P}\, T_1}{S_0 \vee S_1 : \mathbb{P}(T_0 \oplus T_1)}\ (C_{\vee S}) \qquad \frac{S : T}{\neg S : T}\ (C_{\neg S})$$

$$\frac{S : \mathbb{P}\, T_0 \quad C : \mathbb{P}\, T_1}{S \setminus (l \in C) : \mathbb{P}\, T_0 \setminus (l : T_1)}\ (C_h) \qquad \frac{S : T}{S[l_0 \leftarrow l_1] : T[l_0/l_1]}\ (C_r) \qquad \frac{C : \mathbb{P}\, T}{\mathbb{P}\, C : \mathbb{P}\mathbb{P}\, T}\ (C_{\mathbb{P}})$$

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

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

$$\frac{\cdots \quad t_i : T_i \quad \cdots}{\langle\!| \cdots l_i \Rrightarrow t_i \cdots |\!\rangle : [\cdots l_i : T_i \cdots]}\ (C_{\Rrightarrow}) \qquad \frac{t : T_0 \times T_1}{t.1 : T_0}\ (C_1) \qquad \frac{t : T_0 \times T_1}{t.2 : T_1}\ (C_2)$$

$$\frac{t_0 : T_0 \quad t_1 : T_1}{(t_0, t_1) : T_0 \times T_1}\ (C_{()}) \qquad \frac{t : T_1 \quad T_0 \preceq T_1}{t \restriction T_0 : T_0}\ (C_\restriction)$$

We will occasionally write $S^{\mathbb{P}\, T}$ when $\triangleright\ S : \mathbb{P}\, T$ (*etc.*) for convenience.

2.3 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.

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

As before, we omit all data which remains unchanged by a rule.

$$\frac{\Gamma \vdash P_0 \quad \Gamma^- \triangleright P_1\ prop}{\Gamma \vdash P_0 \vee P_1}\ (\vee_o^+) \qquad \frac{\Gamma \vdash P_1 \quad \Gamma^- \triangleright P_0\ prop}{\Gamma \vdash P_0 \vee P_1}\ (\vee_i^+)$$

$$\frac{P_0 \vee P_1 \quad P_0 \vdash P_2 \quad P_1 \vdash P_2}{P_2}\ (\vee^-) \qquad \frac{\Gamma, P \vdash \bot \quad \Gamma^- \triangleright P\ prop}{\Gamma \vdash \neg P}\ (\neg^+)$$

$$\frac{\neg\neg P}{P}\ (\neg^-) \qquad \frac{P \quad \neg P}{\bot}\ (\bot^+)$$

$$\frac{\Gamma \vdash \bot \quad \Gamma^- \triangleright P\ prop}{\Gamma \vdash P}\ (\bot^-) \qquad \frac{P[z/t] \quad t \in C}{\exists\, z \in C \bullet P}\ (\exists^+)$$

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

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

$$\frac{\Gamma^- \triangleright (\!|\, \cdots l_i \Rrightarrow t_i \cdots |\!) : T}{\Gamma \vdash (\!|\, \cdots l_i \Rrightarrow t_i \cdots |\!).l_i = t_i}\ (\Rrightarrow^{=}_{0}) \qquad \frac{\Gamma^- \triangleright t : [\cdots l_i : T_i \cdots]}{\Gamma \vdash (\!|\, \cdots l_i \Rrightarrow t.l_i \cdots |\!) = t}\ (\Rrightarrow^{=}_{1})$$

$$\frac{\Gamma^- \triangleright (t, t') : T}{\Gamma \vdash (t, t').1 = t}\ ((,)^{=}_{0}) \qquad \frac{\Gamma^- \triangleright (t, t') : T}{\Gamma \vdash (t, t').2 = t'}\ ((,)^{=}_{1}) \qquad \frac{\Gamma^- \vdash t : T_0 \times T_1}{\Gamma \vdash (t.1, t.2) = t}\ ((,)^{=}_{2})$$

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

$$\frac{t \notin S}{t \in \neg S}\ (\neg S^+) \qquad \frac{t \in \neg S}{t \notin S}\ (\neg S^-)$$

$$\frac{t \in S}{t[l_0/l_1] \in S[l_0 \leftarrow l_1]}\ (S^+_{\leftarrow}) \qquad \frac{t \in S[l_0 \leftarrow l_1]}{t[l_1/l_0] \in S}\ (S^-_{\leftarrow})$$

$$\frac{\Gamma \vdash t \upharpoonright T_0 \in S_0 \quad \Gamma^- \triangleright S_1 : \mathbb{P}\, T_1 \quad \Gamma^- \triangleright t : T_0 \oplus T_1}{\Gamma \vdash t \in S_0 \vee S_1}\ (S^+_{\vee_0})$$

$$\frac{\Gamma \vdash t \upharpoonright T_1 \in S_1 \quad \Gamma^- \triangleright S_0 : \mathbb{P}\, T_0 \quad \Gamma^- \triangleright t : T_0 \oplus T_1}{\Gamma \vdash t \in S_0 \vee S_1}\ (S^+_{\vee_1})$$

$$\frac{t \in S_0 \vee S_1 \quad t \upharpoonright T_0 \in S_0 \vdash P \quad t \upharpoonright T_1 \in S_1 \vdash P \quad t : T_0 \oplus T_1}{P}\ (S^-_{\vee})$$

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

$$\frac{t_0 \in C_0 \quad t_1 \in C_1}{(t_0, t_1) \in C_0 \times C_1}\ (\times^+) \qquad \frac{t \in C_0 \times C_1}{t.1 \in C_0}\ (\times_0^-) \qquad \frac{t \in C_0 \times C_1}{t.2 \in C_1}\ (\times_1^-)$$

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

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

$$\frac{z : T \vdash P_0 \Leftrightarrow P_1 \quad C : \mathbb{P}\, T}{\{z \in C \mid P_0\} = \{z \in C \mid P_1\}}\ (\{\}^=)$$

$$\frac{\Gamma \vdash t.l_i = t_i \quad \Gamma^- \triangleright t : T \quad [\cdots l_i : T_i \cdots] \preceq T}{\Gamma \vdash (t \restriction [\cdots l_i \in T_i \cdots]).l_i = t_i}\ (\restriction^=)$$

$$\frac{\Gamma \vdash t \in S \quad C : \mathbb{P}\, T_1 \quad \Gamma^- \triangleright t : T_0}{\Gamma \vdash t \restriction T_0 \setminus (l \in T_1) \in S \setminus (l \in C)}\ (S_h^+)$$

$$\frac{\Gamma \vdash t \in S \setminus (l \in C) \quad \Gamma^- \triangleright C : \mathbb{P}\, T_1 \quad \Gamma^- \triangleright t : T_0 \setminus (l : T_1) \quad \Gamma' \vdash P}{\Gamma \vdash P}\ (S_h^-)$$

In the last rule $\Gamma' =_{df} \Gamma^-, y : T_0;\ \Gamma^+, y \in S, y \restriction T_0 \setminus (l \in T_1) = t$.

As usual, we can derive rules for $\wedge$ and $\Rightarrow$ from the above and we will use such rules as necessary in what follows. Many other rules are derivable too, for example:

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

This follows by rules $(\{\}_1^-)$ and $(\exists^-)$.

The convention whereby schema are particular forms of set induces immediately the following rules:

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

Note that sets are extensional as a consequence of $(\{\}^{=})$. We also have the following admissible rule

$$\frac{\Gamma^{-} \triangleright t : T}{\Gamma \vdash t \in T}\ (T^{+})$$

This is the formal counterpart to our intuitions that types (*qua* type carriers) are sets, and so it is hardly surprising to see it.

There are many congruence rules for equality which are derivable in, or admissible for, this system, *e.g.*

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

That these are admissible rules follows by induction on the structure of schema expressions, from the fact that equality is an equivlence relation and by the rule $(\{\}^{=})$.

Most importantly we have a syntactic consistency result for this system, which we state without proof (see [2] for the details).

Proposition 1. *If $\Gamma \vdash_C P$ when Γ context then $\Gamma^{-} \triangleright_C P$ prop* □

3 The schema calculus

All of the following rules, which allow equational reasoning on schemas, are either *derived rules* or *admissible rules* of the logic for Z_C. Complete derivations of these rules from the logic for Z_C can be found in [2].

3.1 Schema negation

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

3.2 Schema disjunction

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

Providing, of course, that $T_0 \oplus T_1$ is defined.

3.3 Schema hiding

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

$$\frac{\Gamma^{-} \triangleright [D^* \mid P] : \mathbb{P}\, T_1 \quad C : \mathbb{P}\, T_0}{\Gamma \vdash [D^* \mid P] \setminus (l \in C) = [D^* \setminus (l \in T_0) \mid \exists z \in C \bullet P[l/z]]}\ (\setminus^{=})$$

3.4 Relating sets and types in schemas

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

$$\frac{\Gamma^- \rhd [D;\ l \in C \mid P] : \mathbb{P}\, T_1 \quad \Gamma^- \rhd C : \mathbb{P}\, T_0}{\Gamma \vdash [D;\ l \in C \mid P] = [D;\ l \in T_0 \mid l \in C \wedge P]}\ (\in^{=})$$

3.5 Schema conjunction

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

$$S_0 \wedge S_1 =_{df} \neg(\neg S_0 \vee \neg S_1)$$

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

$$\frac{\Gamma \rhd S_0 : \mathbb{P}\, T_0 \quad \Gamma \rhd S_1 : \mathbb{P}\, T_1}{\Gamma \rhd S_0 \wedge S_1 : \mathbb{P}(T_0 \oplus T_1)}\ (C_\wedge)$$

The right-hand side can be shown to be equal to the following set:

$$\{y \in T_0 \oplus T_1 \mid y \restriction T_0 \in S \wedge y \restriction T_1 \in S'\}$$

and, given this, the following rule follows by rules $(\wedge^+)$ (derived) and $(\{\}^+)$:

$$\frac{\Gamma \vdash t \restriction T_0 \in S_0 \quad \Gamma \vdash t \restriction T_1 \in S_1 \quad \Gamma^- \rhd t : T_0 \oplus T_1}{\Gamma \vdash t \in S_0 \wedge S_1}\ (S^+_\wedge)$$

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

$$\frac{\Gamma \vdash t \in S_0 \wedge S_1 \quad \Gamma^- \rhd t : T_0 \oplus T_1}{\Gamma \vdash t \restriction T_0 \in S_0}\ (S^-_{\wedge_0})$$

$$\frac{\Gamma \vdash t \in S_0 \wedge S_1 \quad \Gamma^- \rhd t : T_0 \oplus T_1}{\Gamma \vdash t \restriction T_1 \in S_1}\ (S^-_{\wedge_1})$$

With these in place we can prove the expected relationship (see [7] p. 165-6):

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

Finally we have substitution rules:

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

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

3.6 Schema implication

Following the pattern given above for conjunction, we have the definition:

$$S_0 \Rightarrow S_1 =_{df} \neg S_0 \vee S_1$$

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

$$\frac{\Gamma \triangleright S_0 : \mathbb{P}\, T_0 \quad \Gamma \triangleright S_1 : \mathbb{P}\, T_1}{\Gamma \triangleright S_0 \Rightarrow S_1 : \mathbb{P}(T_0 \oplus T_1)}$$

The right-hand side of the definition can be shown to be equal to the set

$$\{z \in T_0 \oplus T_1 \mid z \restriction T_0 \notin S_0 \vee z \restriction T_1 \in S_1\}$$

The introduction rule that follows most directly from this would be

$$\frac{\Gamma \vdash t \restriction T_0 \notin S_0 \quad \Gamma \vdash t \restriction T_1 \in S_1 \quad \Gamma^- \triangleright t : T_0 \oplus T_1}{\Gamma \vdash t \in S_0 \Rightarrow S_1}$$

however, we can show that the following more useful rule is derivable:

$$\frac{\Gamma, t \restriction T_0 \in S_0 \vdash t \restriction T_1 \in S_1 \quad \Gamma^- \triangleright t : T_0 \oplus T_1}{\Gamma \vdash t \in S_0 \Rightarrow S_1}\ (S^+_\Rightarrow)$$

The obvious elimination rule follows directly from the definition:

$$\frac{\Gamma \vdash t \in S_0 \Rightarrow S_1 \quad \Gamma \vdash t \restriction T_0 \in S_0 \quad \Gamma^- \triangleright t : T_0 \oplus T_1}{\Gamma \vdash t \restriction T_1 \in S_1}\ (S^-_\Rightarrow)$$

The expected relationship holds:

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

Finally we have the substitution rules:

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

3.7 Schema inclusion

Schema inclusion can be defined in terms of schema conjunction.

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

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

$$\frac{\Gamma\triangleright[D_0^*\oplus D_1^*\mid P_0]:\mathbb{P}(T_0\oplus T_1)\quad \Gamma\triangleright[D_1^*\mid P_1]:\mathbb{P}\,T_1}{\Gamma\triangleright[D_0^*;\ [D_1^*\mid P_1]\mid P_0]:\mathbb{P}(T_0\oplus T_1)}\ (C_{inc})$$

The introduction rule is:

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

The elimination rules are:

$$\frac{\Gamma^-\triangleright t:T_0\oplus T_1\quad \Gamma\vdash t\in[D_0^*;\ [D_1^*\mid P_1]\mid P_0]}{\Gamma\vdash t\in[D_0^*\oplus D_1^*\mid P_0]}$$

$$\frac{\Gamma^-\triangleright t:T_0\oplus T_1\quad \Gamma\vdash t\in[D_0^*;\ [D_1^*\mid P_1]\mid P_1]}{\Gamma\vdash t\upharpoonright T_1\in[D_1^*\mid P_1]}$$

Finally, we have the expected equational law:

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

3.8 Schema restriction

In view of our filtering operation on terms which we have extended to sets, we can give a pleasant definition[2]:

$$S_0^{\mathbb{P}\,T_0}\upharpoonright S_1^{\mathbb{P}\,T_1} =_{df} S_0\upharpoonright\mathbb{P}\,T_1\wedge S_1$$

when $T_1\preceq T_0$ It is, then, easy to see that this collapses to our extension of filtered terms to sets when the schema S_1 is just a schema *type*.

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

$$\frac{\Gamma\triangleright S_0:\mathbb{P}\,T_0\quad \Gamma\triangleright S_1:\mathbb{P}\,T_1\quad T_1\preceq T_0}{\Gamma\triangleright S_0\upharpoonright S_1:\mathbb{P}\,T_1}$$

[2] This is deliberately weaker than the standard definition (see [5] p. 34) which permits T_1 to introduce new components. The interested reader will have no difficulty in extending our definition to the standard, if that is considered necessary.

The introduction and elimination rules are then as follows:

$$\frac{\Gamma \vdash t \in S_0 \quad \Gamma \vdash t \upharpoonright T_1 \in S_1 \quad T_1 \preceq T_0}{\Gamma \vdash t \upharpoonright T_1 \in S_0 \upharpoonright S_1} \ (S^+_{\upharpoonright})$$

This follows by rules $(S^+_{\wedge})$ and $(\in_{\upharpoonright})$, noting that $T_1 = T_1 \oplus T_1$ and $T_0 = T_0 \oplus T_1$.

$$\frac{\Gamma \vdash t \in S_0 \upharpoonright S_1}{\Gamma \vdash t \in S_0 \upharpoonright T_1} \ (S^-_{\upharpoonright_0}) \qquad \frac{\Gamma \vdash t \in S_0 \upharpoonright S_1}{\Gamma \vdash t \in S_1} \ (S^-_{\upharpoonright_1})$$

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

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

3.9 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 [5] p. 76. We should provide, within our framework, a form of schema level hiding to correspond to this.

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

$$S_0^{\mathbb{P}\, T_0} \setminus S_1^{\mathbb{P}\, T_1} =_{df} (S_0 \wedge S_1) \upharpoonright \mathbb{P}(T_0 \setminus T_1)$$

Using rules $(C_{\wedge})$, $(C_{\mathbb{P}\upharpoonright})$, together with the fact that $T_1 \preceq T_0$, we obtain the following type rule:

$$\frac{\Gamma \rhd S_0 : \mathbb{P}\, T_0 \quad \Gamma \rhd S_1 : \mathbb{P}\, T_1 \quad T_1 \preceq T_0}{\Gamma \rhd S_0 \setminus S_1 : \mathbb{P}(T_0 \setminus T_1)}$$

The introduction rule is calculated using rules $(S^+_{\upharpoonright})$, $(S^+_{\wedge})$ and the fact that $T_0 \setminus T_1 \preceq T_0$.

$$\frac{\Gamma \vdash t \in S_0 \quad \Gamma \vdash t \upharpoonright T_1 \in S_1 \quad T_1 \preceq T_0}{\Gamma \vdash t \upharpoonright (T_0 \setminus T_1) \in S_0 \setminus S_1}$$

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

$$\frac{\Gamma \vdash t \in S_0 \setminus S_1 \quad \Gamma' \vdash P \quad \Gamma^- \rhd t : T_0 \setminus T_1}{\Gamma \vdash P}$$

where $\Gamma' =_{df} \Gamma^-, x : T_0;\ \Gamma^+, x \in S_0, x \upharpoonright T_1 \in S_1, x \upharpoonright (T_0 \setminus T_1) = t$ 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 [7] (p. 178). Since it makes no sense to quantify over constants, this equation requires an additional substitution.

Let $D_2^* = \cdots l_i \in T_i \cdots$ and $\sigma = [\cdots l_i \cdots / \cdots z_i \cdots]$ where the z_i are fresh variables.

$$\frac{\Gamma^- \rhd [D_1^* \mid P] : \mathbb{P}\, T \quad \Gamma^- \rhd [D_2^* \mid P'] : \mathbb{P}\, T'}{\Gamma \vdash [D_1^* \mid P] \setminus [D_2^* \mid P'] = [D_1^* \setminus D_2^* \mid \exists\, D_2^* \sigma \bullet (P \wedge P')\sigma]}$$

The substitution rules are:

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

3.10 Some useful generalisations

It turns out then when it comes to *using* the schema calculus, rather more general rules, which package-up common strategies for combining several of the above rules, are needed. In this section we review a few that will be useful in our extended example at the end of the paper.

The following two rules are used together for reasoning about general conjunction schemas.

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

This is easily proved using (*sub*), the rule $(\in^=)$ for moving sets out of the declaration in favour of types and the standard version of the conjunction rule $(\wedge^=)$.

In fact it is *admissible* rather than derivable in this general form since we need to do an induction on the *length* of the declarations. All instances for fixed length declarations are, however, derivable.

A more general form of the rule for conjunction is:

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

subject to the condition $\alpha D_0 \cap \alpha D_1 = \{\}$. The proof of this uses the rules $(\in^=)$, (*sub*) and $(\wedge^=)$.

There is a version of this pair for disjunction too, proved in similar ways:

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

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

A final generalisation, for hiding, that we shall need for our example is:

$$\frac{\Gamma^{-} \vdash [D;\ l \in C \mid P] : \mathbb{P}\, T_1 \quad C : \mathbb{P}\, T_0}{\Gamma \vdash [D;\ l \in C \mid P] \setminus (l \in C) = [D \mid \exists z \in C \bullet P[l/z]]}\ (\setminus^{=}_{gen})$$

which is proved using (*sub*), $(\in^{=})$ and $(\setminus^{=})$.

4 The language of Z

We do not have the space at our disposal in this paper to outline the interpretation of Z into Z_C in full generality. In this section we will simply develop just enough extra infrastructure to enable us to demonstrate the value of our logic for the schema calculus when we turn to the examination of a non-trivial example in section 5.

First we consider the use in Z of Δ-schemas. These appear in the context of operation schema to indicate initial and final states. We note that, in recent years, schema have come to play a far more active role than their original structuring one. This greater role presupposes that they represent specifications of collections rather than of individuals. This perspective offers us the opportunity to interpret the Z idiom ΔS by the declaration $z, z' \in S$ in our core logic Z_C. This turns out to have a desirable technical side effect: we can represent the Z idiom ΞS by the schema $[z, z \in S \mid z = z']$. As a result the *type* of the equality is preserved naturally, without the complications which, in some accounts, accompany discussion of the θ operator, in particular those concerning the equation $\theta S = \theta S'$ (in which S' does *not* denote the schema expression S'). Indeed this approach can be employed whenever the θ operation is normally required. It is a natural corollary of adding schema as sets in Z that the θ operation is not required in the core logic Z_C and that its use in Z can always be interpreted in the manner illustrated.

The following schematic equations are sufficient for our requirements in this paper.

$$[\cdots \Delta S \cdots \mid P] =_{df} [\cdots z, z' \in S \cdots \mid P\sigma]$$

where the substitution is:

$$\sigma =_{df} [\alpha S/z.\alpha S][\alpha S'/z'.\alpha S][\theta S/z][\theta S'/z']$$

Then:

$$[\cdots \Xi S \cdots \mid P] =_{df} [\cdots \Delta S \cdots \mid P \wedge \theta S = \theta S']$$

as expected.

5 Example

In order to show our schema calculus in use we consider a reasonably complex example from the literature. This uses the technique of *promotion* (see [7] chapter 13). The example is taken from this chapter (*ibid.* pp. 194-195) and the earlier chapter which introduces the schema operators (*ibid.* chapter 12, pp. 170-175) and 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[3].

First we have the Box Office itself:

BoxOffice

$$seating \in \mathbb{P}\, Seat$$
$$sold \in Seat \pfun Customer$$
$$\dom sold \subseteq seating$$

Purchasing a ticket can succeed or fail. The prototype for a successful purchase is given by:

$Purchase_0$

$$\Delta BoxOffice$$
$$s? \in Seat$$
$$c? \in Customer$$
$$s? \in seating \setminus \dom sold$$
$$sold' = sold \cup \{s? \mapsto c?\}$$
$$seating = seating'$$

and success is specified to be:

Success

$$r! \in Response$$
$$r! = \text{okay}$$

If, on the other hand, the seat requested by the customer is not available, we have:

NotAvailable

$$\Xi BoxOffice$$
$$s? \notin seating \setminus sold$$

and failure is captured by means of:

[3] Apart from our use of $\in$ instead of :, as explained in section two.

Failure

$r! \in Response$

$r! = \text{sorry}$

The specification for purchasing a ticket is now composed from these individual specifications:

$$Purchase =_{df} (Purchase_0 \wedge Success) \vee (NotAvailable \wedge Failure)$$

The equational logic can now be deployed in order to investigate this new composite specification:

By rule $(\wedge^{=}_{gen})$, we have $Purchase_0 \wedge Success =$

PandS

$\Delta BoxOffice$
$s? \in Seat$
$c? \in Customer$
$r! \in Response$

$s? \in seating \setminus \text{dom}\, sold$
$sold' = sold \cup \{s? \mapsto c?\}$
$seating' = seating$
$r! = \text{okay}$

Similarly, using rule $(\wedge^{=}_{gen})$, we have $NotAvailable \wedge Failure =$

NandF

$\Xi BoxOffice$
$s? \in Seat$
$r! \in Response$

$s? \notin seating \setminus \text{dom}\, sold$
$r! = \text{sorry}$

By rules $(sub_{\vee_0})$ and $(sub_{\vee_1})$ we now have:

$$Purchase = PandS \vee NandF$$

NandF is, by the definition of *Xi*-schemas, equivalently:

NandF

$\Delta BoxOffice$
$s? \in Seat$
$r! \in Response$

$s? \notin seating \setminus \text{dom}\, sold$
$r! = \text{sorry}$
$\theta BoxOffice = \theta BoxOffice'$

Finally, we use the general rule for disjunction to obtain $Purchase =$

$$
\begin{array}{|l}
\Delta BoxOffice \\
s? \in Seat \\
c? \in Customer \\
r! \in Response \\
\hline
(s? \in seating \setminus \mathrm{dom}\ sold \land \\
\quad sold' = sold \cup \{s? \mapsto c?\} \land \\
\quad seating' = seating \land \\
\quad r! = \mathrm{okay}) \\
\quad \lor \\
(s? \notin seating \setminus \mathrm{dom}\ sold \land r! = \mathrm{sorry} \land \\
\quad \theta BoxOffice = \theta BoxOffice') \\
\hline
\end{array}
$$

We promote this local operation of purchasing a ticket to be a global one by conjoining it with the promotion schema:

$$
\begin{array}{|l}
Promote \\
\hline
\Delta GlobalBoxOffice \\
p? \in Performance \\
\hline
p? \in \mathrm{dom}\ booking \\
\theta BoxOffice = booking\ p? \\
\theta BoxOffice' = booking\ p? \\
\{p?\} \lhd booking = \{p?\} \lhd booking \\
announced' = announced \\
\hline
\end{array}
$$

where we have:

$$
\begin{array}{|l}
GlobalBoxOffice \\
\hline
announced \in \mathbb{P}\ Performance \\
booking \in Performance \pfun BoxOffice \\
\hline
\mathrm{dom}\ booking \subseteq announced \\
\hline
\end{array}
$$

We then hide the components of the local state (we use hiding rather than the existential quantification used in the textbook, though these amount to the same thing) to get:

$$GlobalPurchase_0 =_{df} (Purchase \land Promote) \setminus (\Delta BoxOffice)$$

We then have, by rule $(\land^{=}_{gen})$ $Purchase \land Promote =$

PandP

$\Delta GlobalBoxOffice$
$\Delta BoxOffice$
$s? \in Seat$
$c? \in Customer$
$p? \in Performance$
$r! \in Response$

$p? \in \mathrm{dom}\ booking$
$\theta BoxOffice = booking\ p?$
$\theta BoxOffice' = booking'\ p?$
$\{p?\} \lhd booking' = \{p?\} \lhd booking$
$announced' = announced$
$((s? \in seating \setminus \mathrm{dom}\ sold \land$
$sold' = sold \cup \{s? \mapsto c?\} \land$
$seating' = seating \land$
$r! = \mathrm{okay})$
$\lor$
$(s? \notin seating \setminus \mathrm{dom}\ sold \land r! = \mathrm{sorry} \land$
$\theta BoxOffice = \theta BoxOffice'))$

Then we have: $(Purchase \land Promote) \setminus (\Delta BoxOffice) = (rule(sub_{\setminus}))$
$PandP \setminus (\Delta BoxOffice) = ((\setminus^{=}_{gen}))$

GlobalPurchase$_1$

$\Delta GlobalBoxOffice$
$s? \in Seat$
$c? \in Customer$
$p? \in Performance$
$r! \in Response$

$\exists z, z' \in BoxOffice \bullet$
$(p? \in \mathrm{dom}\ g.booking$
$z = booking\ p?$
$z' = booking'\ p?$
$\{p?\} \lhd booking' = \{p?\} \lhd booking$
$announced' = announced$
$((s? \in z.seating \setminus \mathrm{dom}\ z.sold \land$
$z'.sold = z.sold \cup \{s? \mapsto c?\} \land$
$z'.seating = z.seating \land$
$r! = \mathrm{okay})$
$\lor$
$(s? \notin z.seating \setminus \mathrm{dom}\ sold \land r! = \mathrm{sorry} \land$
$z = z'))$

Use of the one-point rule on both z and z' in the predicate part of this schema gives, by substitution:

GlobalPurchase

$\Delta GlobalBoxOffice$
$s? \in Seat$
$c? \in Customer$
$p? \in Performance$
$r! \in Response$

$p? \in \mathrm{dom}\ booking$
$\{p?\} \lhd booking' = \{p?\} \lhd booking$
$announced' = announced$
$((s? \in booking\ p?.seating \setminus \mathrm{dom}\ booking\ p?.sold \land$
$booking'\ p?.sold = booking\ p?.sold \cup \{s? \mapsto c?\} \land$
$booking'\ p?.seating = booking\ p?.seating \land$
$r! = \mathrm{okay})$
$\lor$
$(s? \notin booking\ p?.seating \setminus \mathrm{dom}\ booking\ p?.sold \land r! = \mathrm{sorry} \land$
$booking\ p? = booking'\ p?))$

Note that this is not quite the same as the corresponding result on p.195 of [7] since that version contains several errors, the worst of which makes that schema refer to the label *sold'* when no such label has been declared. But such errors are only to be expected when only informal reasoning is available, and their detection serves to remind us why a formal method requires precise formalisation. So, once the errors are accounted for, we can see that, as expected and by design, we now do have a schema *calculus*.

6 Future Work

We have been able to present the outlines of a powerful equational logic which establish a schema calculus for Z. This was possible only because we have also provided, for the first time, a logic for schema expressions. Now that this much is in place, it is possible to contemplate the development of more complex modes of reasoning which will permit reasoning at a usefully high level. The few rules we introduce in section 3.10 begin this process. Additionally, we must cover, in detail, those schema operations designed particularly to encourage the modular design of operations, for example, schema composition and piping. These topics will be explored in detail in the full version of this paper. Here we can simply sketch the trajectory: If we restrict ourselves to composition along a single pair of complementary labels (for expository purposes) we can follow the approach of, for example, [1] and define:

$$S_0 ;_{(l',l):T} S_1 =_{df} (S_0[l' \leftarrow v] \land S_1[l \leftarrow v]) \setminus (v : T)$$

In general the data indexing the operator, that is the set of complementary pairs of labels and their types, can be calculated from the *type* of the component

schema. We can define a similar operation over types, as follows:

$$T_0 ;_{(l,l'):T} T_1 =_{df} (T_0[l'/v] \oplus T_1[l/v]) \setminus (v : T)$$

and then, for example, we can easily derive the following type assignment rule:

$$\frac{S_0 : \mathbb{P}\, T_0 \quad S_1 : \mathbb{P}\, T_1}{S_0 ;_{(l',l):T} S_1 : \mathbb{P}(T_0 ;_{(l,l'):T} T_1)}$$

and the corresponding introduction and elimination rules. Naturally, our system lends itself to mechanisation and an implementation of Z_C extended to the schema calculus we have presented here is currently under construction in Isabelle ([4], [6]).

We are also involved in a wider project which is looking at alternative foundations for Z, specifically based on intensional set theory and constructive logic. The purpose of this is to investigate alternative means for integrating program development with specification. In the context of this project, the schema calculus we have introduced here is of particular significance: associated with the various rules are mechanisms for combining programs, and this allows the schema calculus to play the dual role of organising specifications and, additionally, methodically constructing implementations.

Acknowledgments

We would like to thank the University of Waikato, the Royal Society of Great Britain and the EPSRC (grant number GR/L57913) for supporting the work reported here.

References

1. A. Diller. *Z: An introduction to formal methods (2^{nd} ed.).* J. Wiley and Sons, 1994.
2. M. C. Henson and S. V. Reeves. Revising Z: semantics and logic. *Formal Aspects of Computing (submitted)*, 0:000–000, 1998.
3. J. Nicholls. *Z Notation: Version 1.2.* Z Standards Panel, 1995.
4. L. Paulson. *Isabelle: A generic theorem prover.* Springer Verlag, LNCS Vol. 828, 1994.
5. J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall, 1992.
6. N. Völker. Private communication, 1998.
7. J. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof.* Prentice Hall, 1996.