

SOFTWARE VERIFICATION RESEARCH CENTRE
SCHOOL OF INFORMATION TECHNOLOGY
THE UNIVERSITY OF QUEENSLAND

Queensland 4072
Australia

TECHNICAL REPORT

No. 98-21

A revised deductive system for Z

Andrew Martin

February 1998

Phone: +61 7 3365 1003

Fax: +61 7 3365 1533

<http://svrc.it.uq.edu.au>

Note: Most SVRC technical reports are available via anonymous ftp, from `svrc.it.uq.edu.au` in the directory `/pub/techreports`. Abstracts and compressed postscript files are available via `http://svrc.it.uq.edu.au`

A revised deductive system for Z

Andrew Martin*

Software Verification Research Centre
The University of Queensland

Abstract

We present a deductive system for Z. The first part of this report describes the background to the presentation, and seeks to clarify some common misunderstandings. The greater part of the report is taken up with a proposed revision of the section of the Z Standard which describes a deductive system.

Contents

1	Introduction	1
2	Deductions in Z	2
3	Objections raised against previous versions	2
4	Reasoning with schemas	3
5	Conclusion	3
A	A deductive system for Z	5
A.1	Introduction	5
A.2	Symbols used in the deductive system	6
A.3	Alphabets	10
A.4	Free Variables	10
A.5	Substitution	11
A.6	Logical inference rules	17
A.7	Using the deductive system	23

1 Introduction

This technical report presents a variant of the deductive system described in version 1.2 (the Committee Draft) of the Z Standard (Nicholls 1995). In most regards it represents only a small variation on the previous version. It has tried to cover all the comments and issues raised.

Whilst largely dealing with the particular requirements of Z standardisation, this work may also be relevant to other projects considering machine assistance for Z reasoning. In particular, a dedicated proof tool for the *Cogito* (Traynor, Hazel, Kearney,

* Author's address (from April '98): Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK. apm@ecs.soton.ac.uk

Martin, Nickson, & Wildman 1997) specification language *Sum* is under consideration, and this deductive system, with additional constructs to support modularity, might be useful there.

The document is structured as follows. The next section introduces the concept of a dedicated deductive system for *Z*, and explains why it might be useful. The following section outlines some of the problems with the version presented in the current version of the Standard, and explains how they are addressed here. Another section addresses reasoning with schemas, and how this is covered in the system presented here. Following this, a section describes how the soundness of the system might be presented.

The deductive system itself is conjoined as an annex, following very largely the form it might take in the Standard.

Notation The deductive system is presented using the current version of the abstract syntax (Nicholls 1997).

2 Deductions in *Z*

The introduction to the annex explains the rationale for constructing a deductive system for *Z*. Many have proposed approaches to constructing reasoning environments for *Z* (Martin 1997); some more successfully than others. Whilst *Z* is superficially a typed version of elementary discrete mathematical ideas, supporting its full range of expression, and in particular schemas, is somewhat problematic.

In essence, there are two approaches which may be followed. One is to find ways to express *Z* using more familiar constructs from logic and set theory; the other is to describe a reasoning system in which all deductions stay within the *Z* language. The latter is the approach used here.

In taking this approach, we ensure that all the terms arising in the proof are part of *Z*. In particular, they are well-formed with respect to *Z*'s grammar, and well-typedness is preserved by the inference rules. The inference rules presented are thus intended to be axiomatic: it should not be necessary to decompose any *Z* terms, or to step outside the standard grammar in order to accomplish a proof. A lengthy account of proof in this style is published as a tutorial on proof in Standard *Z* (Brien & Martin 1995).

This property is akin to completeness, but we do not expect a logic for *Z* to be complete in the usual sense. The soundness of the axioms may be verified using the semantics presented in the standard. Indeed, one reason for the particular style of presentation used there is to facilitate relatively easy proof of such properties.

3 Objections raised against previous versions

A number of objections may be raised against the presentation of a deductive system for *Z* in the Committee Draft of the Standard (Nicholls 1995). Some came through the formal ballot procedure, a very few others came from different sources.

There are a number of typos in that presentation, some missing provisos, and a few incomplete rules. Some rules were incorrectly marked as reversible. Some notation was inconsistent with the rest of the Standard. The deductive system was intended to be presented using the abstract syntax, but did not adhere to it tightly. The scope of provisos on various calculational parts (for α , ϕ , and substitution) was unclear. Most of those infelicities should be rectified here.

The presentation of ‘provisos-as-judgements’ is flawed. If the ‘tree-squashing’ theorem is invoked, the context in which each proviso is to be evaluated is lost. In the new presentation, the context for each proviso is made explicit.

The deductive system takes a particular view with respect to non-denoting terms. In this regard, it is a valid interpretation of the standard semantics, which has been constructed to allow several different interpretations. This was not made clear, and so the presentation has been called unsound because it identified theorems which cannot be deduced from the semantics. (For example, for any expression e , the theorem $\vdash e = e$ is provable in the deductive system.)

4 Reasoning with schemas

Given our concern to stay entirely within the Z language and use this deductive system to prove all useful theorems, the number and scope of the rules for the schema calculus may seem to be unduly limited. In fact, no further rules are needed.

The key to this observation is the use of *bindings* (i.e. members of schema types) as substitutions in this deductive system. Instead of substitution being a metalogical notion, it is actually defined within the logic. Since substitution is highly context-sensitive, this turns out to be very appropriate.

A statement such as

$$b \in S$$

can be transformed (subject to certain well-formedness conditions) into the predicate

$$b \odot S$$

and the rules of substitution used to simplify the predicate so that no schema calculus operators remain. This has been the approach ever since \mathcal{W} (Woodcock & Brien 1992) demonstrated that bindings can be used for substitution.

The standard now replaces schema predicates by membership statements. The predicate

$$S$$

is replaced by

$$\theta S \in S$$

This presents a problem, because it means that $b \odot S$ is not well-formed.

The simplest solution is to make an extension to the abstract syntax in the logic, saying that S is a shorthand for $\theta S \in S$, and then also giving axioms for $b \odot S$.

5 Conclusion

This presentation offers the present best attempt at a deductive system which stays within Standard Z. It is intended for consideration for inclusion in the next Committee Draft of the Z Standards Panel.

We have not addressed the issue of soundness here. An outline proof of soundness is included in (Brien 1998). The relational presentation of the Z Standard’s semantics is intended to make such a proof relatively simple. We would expect to publish a soundness proof to coincide with publication of the Standard.

Acknowledgements

Most of the good ideas here are due to Stephen Brien, with earlier input from Jim Woodcock. I would like to thank Jon Hall for many detailed comments and discussions in preparation of this draft. The introduction to the annex was written by Peter Lupton. Other members of the Z Standards panel have provided useful feedback.

References

- Brien, S. M. (1998). *A Logic and Model for the Z Standard*, Draft D.Phil. thesis, University of Oxford.
- Brien, S. M. & Martin, A. P. (1995). A tutorial on proof in Standard Z, *Technical Monograph PRG-120*, Programming Research Group, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK. Presented at ZUM'95.
- Hall, J. & Martin, A. (1997). W reconstructed, in J. P. Bowen, M. G. Hinchey, & D. Till (eds), *ZUM'97: The Z Formal Specification Notation*, Vol. 1212 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin Heidelberg.
- Martin, A. (1997). Why effective proof tool support for Z is hard, *Technical report 97-34*, Software Verification Research Centre, School of Information Technology, The University of Queensland, Brisbane 4072. Australia.
URL: <http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?97-34>
- Nicholls, J. (1997). Abstract syntax. 'Preliminary working draft': was on Oxford FTP server.
URL: <http://www.it.uq.edu.au/personal/apm/zstan/absyn.ps>
- Nicholls, J. (ed.) (1995). *Z Notation*, Z Standards Panel, ISO Panel JTC1/SC22/WG19 (Rapporteur Group for Z). Version 1.2, ISO Committee Draft; CD 13568.
URL: <ftp://ftp.comlab.ox.ac.uk/pub/Zforum/ZSTAN/drafts>
- Traynor, O., Hazel, D., Kearney, P., Martin, A., Nickson, R., & Wildman, L. (1997). The Cogito development system, in M. Johnson (ed.), *Algebraic Methodology and Software Technology*, Vol. 1349 of *LNCS*, Springer-Verlag, Berlin, pp. 586–591. 6th International conference, AMAST'97, Sydney, Australia.
- Woodcock, J. C. P. & Brien, S. M. (1992). *W: A Logic for Z*, *Proceedings 6th Z User Meeting*, Springer-Verlag.

A A deductive system for Z

A.1 Introduction

The definition of the Z notation defines a way in which any specification written in Z can be translated into the language of set theory. This defines, indirectly, a logic for Z: a specification entails a predicate when the translation of the specification entails the translation of the predicate. The purpose of this informative annex is to give an example of a deductive system for Z that is more direct than such a double translation scheme. It is, further, a deductive system for Z in which the language used for deduction is closely related to Z. The purpose of such a deductive system should be clear to anyone that has written or attempted to understand a specification. Reasoning about a specification written in Z is much easier for human to do without such indirect translations. The value of the following deductive system is that any deductive system for a notation such as Z must answer such questions as 'what are the side conditions which should constrain specific inference laws' and these questions have been answered in detail. It is to be stressed that the following inference system is intended to be for human consumption: the issues and concerns raised in the case of mechanical theorem provers are in some respects different. There is no presumption that the inference system which follows addresses those issues and concerns.

Z users have long been writing conjectures (or assertions) in their documents in the form

$$\vdash \text{PRED}$$

or

$$S \vdash \text{PRED}$$

(where S is some schema). In this annex, we take these to be elliptical forms of the general Z assertion. The general form of an assertion in the Z notation will be

$$\text{SPEC} \vdash \text{PRED}$$

It means that PRED holds in every environment which satisfies SPEC, that is every environment related to the empty environment via the meaning of SPEC. The theorems of Z are the members of the set of all such predicates — the semantics of Z defines the set of all such theorems.

Such a conjecture is not given a semantic definition since it is not part of what a specification means. Rather, a conjecture is a formal statement made about the specification: the combination of this specification and the conjectured predicate should appear among the theorems of Z. Conjectures are therefore normative statements made by a specifier about a specification.

An inference system for Z may define an alternative method for generating a set of such assertions. As such, it must satisfy the requirement of soundness and could satisfy the property of completeness. The following inference system is not intended to be complete. This is largely for technical reasons of no interest to the user of Z. The semantics of Z translates Z into a variant of set theory in which certain axioms hold, regardless of whether they are consequences of the specification being translated. As a consequence, the semantics of Z will be more restrictive than the following inference system leaving the inference system incomplete in ways that will be entirely irrelevant to the practitioner. To reiterate, this inference system is not intended to be

(necessarily) *the* inference system for Z , rather one against which other systems (such as an equational reasoning framework) might be proven sound.

To express this inference system, we require some additional metalanguage. This is presented in the following section. Thereafter, we give inference rules which cover reasoning about every term of the abstract syntax.

A.2 Symbols used in the deductive system

In adhering strictly to the abstract syntax, some of the inference rules will look rather peculiar, and/or unlike Z .

Naming Conventions Z phrases are written in a canonical form, with pre-determined letters standing for elements of Z , as listed in the following table.

Table 1: Naming conventions

Symbol(s)	Description	Abstract syntactic category
x, y	names	NAME
q	decoration	
i	number	
b, e, f, s, u, v	expressions	EXP
P, Q, R	predicates	PRED
S, T	schemas	EXP
Π	paragraph	PAR
Γ	specification	PAR ... PAR

In addition, it will sometimes be necessary to refer directly to the *names* of Z variables (as names, not Z values). For example, the provisos on certain rules will require $x \neq y$, by which we mean that x and y are distinct identifiers; we make no assertion about their respective values. In this case, we shall enclose the names in quotes, writing for the above ' $x \neq y$ '. We use ψ (decorated as necessary) to denote sets of quoted identifiers.

Extra Symbols This deductive system uses a number of symbols not found elsewhere in the Standard. They are listed in Table 2, and explained in the following pages.

The extra symbols divide into two groups. Substitution symbols are a minor extension to the Z language (or possibly some syntactic sugar); the other symbols are metalogical and are used in describing the logical system. In the sequel, the substitution symbols will be assumed to be part of the Z notation. The turnstile symbol (\vdash) is given a meaning which extends its use in signalling a conjecture in the Z syntax. (if conjectures are in the language this week)

Table 2: Extra Symbols used in deductive system

Symbol(s)	Name	Syntax	Description
\odot	Predicate substitution	$EXPR \odot PRED$	The binding on the left is used as a substitution into the predicate on the right. See Section A.2.5.
\circ	Expression substitution	$EXPR \circ EXPR$	The binding on the left is used as a substitution into the expression on the right. See Section A.2.5.
\vdash	Sequent	$SPEC \vdash PRED$	The fundamental <i>judgement</i> of the deductive system. Expresses the conjecture that the predicate on the right can be proved in the context of the specification on the left. See Section A.2.1.
—	Inference rule	$\frac{Sequents}{Sequent}$	Says that the sequents above the line are sufficient to prove the sequent below the line. See Section A.2.2
==	Reversible rule	$\frac{Sequent}{Sequent}$	Inference rule which says that the sequent above the line is sufficient to prove the sequent below the line, and <i>vice versa</i> . See Section A.2.2.
α	Alphabet	$\alpha EXPR$	Alphabet of expression, i.e. the set of variables defined by that expression. See Section A.2.3.
ϕ	Free variables	$\phi EXPR$	The set of variables appearing in the given expression which are not bound in that expression. See Section A.2.4.
Φ	Free variables	$\Phi PRED$	The set of variables appearing in the given predicate which are not bound in that predicate. See Section A.2.4.

A.2.1 Sequent

The deductive system is presented using a language of sequents. The antecedent (left hand side of the turnstile) is a flat specification (list of Z paragraphs); the consequent is a Z predicate.

$$Sequent == \Pi_1 \cdots \Pi_n \vdash P$$

A sequent is *true* if the environments defined by the specification satisfy the predicate.

$$\{\{ SPEC \} \} \subseteq \{\{ PRED \} \}$$

The deductive system defines which sequents may be deduced from which other sequents.

The intention is that a sequent may appear in a Z specification, as has customarily been the case. Any explicit antecedents are added to the specification to provide a

scope in which the consequent is to be proved. Therefore, in the rules which follow, the mathematical toolkit is, for example, implicitly present in the antecedent of every sequent.

Observe that the interpretation of the antecedent in these rules does *not* require the ‘no redefinition’ principle which normally applies to Z paragraphs. This style of sequent (with a single predicate on as the consequent), is often associated with intuitionistic logics. The account here is classical, as demonstrated by (Hall & Martin 1997).

A.2.2 Rules

The deductive system consists of a number of rules for manipulating sequents. Inference rules will be written as instances of *Rule*:

$$Rule == \frac{Premises}{Conclusion} Name (Proviso)$$

The premises are a (possibly empty) list of sequents:

$$Premises == Sequent \dots Sequent$$

The conclusion is always a single sequent:

$$Conclusion == Sequent$$

The Proviso is a decidable condition on the free variables of the expressions and predicates in the rule. If the central rule is doubled, this indicates that the rule can be applied in both directions—that is, the rule:

$$\frac{\Gamma \vdash Q}{\Gamma' \vdash P}$$

denotes both of the following inference rules

$$\frac{\Gamma \vdash Q}{\Gamma' \vdash P} \quad \text{and} \quad \frac{\Gamma' \vdash P}{\Gamma \vdash Q}$$

The rule names are intended to be systematic, descriptive, and short. For the logical operators, this means that the name *AndI* denotes ‘and introduction’, whilst *AndEr* denotes ‘and elimination on the right-hand term’.

A rule is *sound* if whenever it is applied to valid premisses, a valid conclusion results. This is defined in the semantics by saying that the set of environments supporting the premisses is a subset of those supporting the conclusion. The rule

$$\frac{S_1 \quad \dots \quad S_m}{Seq} N(C)$$

is sound if and only if

$$C \Rightarrow \{ \{ S_1 \} \} \cap \dots \cap \{ \{ S_m \} \} \subseteq \{ \{ Seq \} \}$$

Provisos The provisos on the inference rules either determine the permitted values for indexes used in the rules, or constrain the interaction of free and bound variables in some way. The latter are expressed using the meta-functions α and ϕ , defined in Sections A.2.3 and A.2.4. Furthermore, we use βS to denote that subset of αS whose members appears therein in both primed and unprimed forms.

A.2.3 Alphabets

A Z schema declares a collection of variables and may include predicates over those (and other) variables. The *alphabet* of a schema is the set consisting of those variable *names*, and for a schema S , is denoted αS .

For example, (using the concrete syntax for ease of reading, here and in other examples):

$$\alpha[x, y : X; z : \mathbb{N} \mid x \neq y \wedge z \in C] = \{ 'x', 'y', 'z' \} .$$

One way to compute the alphabet of an expression is by reference to the type system of Z . This approach is explained in Section A.3. More generally, observe that the calculation of alphabets is dependent upon the context in which the expression appears.

A.2.4 Free Variables

The free variables of an expression or predicate are those which may be captured if it is placed in the context of a particular definition (typically, a quantifier). We use ϕe to denote the free variables of expression e , and ΦP to denote the free variables of predicate P . Observe that the definitions of these may differ from classical expectations.

Note, in particular, that in the context of a schema definition $S ::= [x : \mathbb{N} \mid x \in C]$, we have

$$\begin{aligned} \phi S &= \{ 'S' \} \\ \phi[x : \mathbb{N} \mid x \in C] &= \{ 'C' \} \\ \Phi S &= \{ 'S', 'x' \} \\ \Phi[x : \mathbb{N} \mid x \in C] &= \{ 'x', 'C' \} . \end{aligned}$$

As with alphabets, the computation of free variables is dependent upon the context in which the predicate or expression appears. Rules for the calculation of free variables are given at Section A.4.

A.2.5 Substitution

In many logics, substitution is a metalogical notion. The way names and variables are handled in Z allows us to define substitution *within* Z . A binding is used to describe a substitution. Because the effect of substitution into schemas used as expressions is different from that of substitution into schemas used as predicates, separate symbols are used as an aid to the reader, though the correct interpretation is invariably clear from the context.

The predicate $b \odot P$ is true if and only if the predicate P is true in the environment enriched by the binding b .

$$\{ \{ b \odot P \} \} = \text{dom}(\langle I, \llbracket b \rrbracket \circ \diamond \rangle \circ (\oplus) \triangleright \{ \{ P \} \})$$

The expression $b \circ e$ denotes an expression equal to x in the environment enriched by the binding b .

$$\llbracket b \circ e \rrbracket = \langle I, \llbracket b \rrbracket \circ \diamond \rangle \circ (\oplus) \circ \llbracket e \rrbracket$$

These notations may also (apparently) be defined using rewrite rules:

$$\begin{aligned} b \odot P &\xrightarrow{R} (\exists\{b\} \bullet P) \\ b \odot e &\xrightarrow{R} (\mu\{b\} \bullet e) . \end{aligned}$$

Substitutions may generally be distributed through the structure of terms (always having regard to the context, for the calculation of free variables, etc.). The tables at Section A.5 give simplification rules which may be used for this purpose.

This notation of substitution is a slight generalisation of the ‘**let**’ notation, in that in place of an explicit declaration of the bound variable, it permits an arbitrary binding. Thus the expression **let** $x == e \bullet s$ is equivalent to $\langle x == e \rangle_{\odot s}$, but $b \odot e$ cannot be expressed simply using **let**.

A.2.6 Decoration

The distribution of decorations through expressions will be covered elsewhere in the Standard.

A.3 Alphabets

Alphabets may be computed using type inference.

If e has a schema type, αe is available directly:

$$\frac{\Gamma \vdash e \text{ : } \mathcal{P}\Sigma(i_1 : \tau_1, \dots, i_n : \tau_n)}{\Gamma \vdash \alpha e = \{‘i_1’, \dots, ‘i_n’\}}$$

For e of any other type, the alphabet of e is empty.

$$\frac{\Gamma \vdash e \text{ : } \tau}{\Gamma \vdash \alpha e = \{ \}} \tau \neq \mathcal{P}\Sigma(\dots)$$

An alternative formulation is to use the computed types of expressions, putting

$$\alpha(e) = \alpha(e \text{ : } \tau)$$

$$\begin{aligned} \alpha(e \text{ : } \mathcal{P}\Sigma(i_1 : \tau_1, \dots, i_n : \tau_n)) &= \{‘i_1’, \dots, ‘i_n’\} \\ \alpha(e \text{ : } \Sigma(i_1 : \tau_1, \dots, i_n : \tau_n)) &= \{‘i_1’, \dots, ‘i_n’\} \\ \alpha(e \text{ : } \chi\tau) &= \{ \} \\ \alpha(e \text{ : } \mathcal{P}\dot{\cup}\tau) &= \{ \} \\ \alpha(e \text{ : } \mathcal{P}\chi\tau) &= \{ \} \\ \alpha(e \text{ : } \mathcal{P}\mathcal{P}\tau) &= \{ \} \\ \alpha(e \text{ : } \dot{\cup}\tau) &= \{ \} \end{aligned}$$

A.4 Free Variables

Free variable sets may be computed using inference. For example:

$$\frac{\Gamma \vdash \Phi P = \psi_1 \quad \Gamma \vdash \Phi Q = \psi_2}{\Gamma \vdash \Phi(P \wedge Q) = \psi_3} \quad \psi_3 = \psi_1 \cup \psi_2$$

$$\overline{\Gamma \vdash \Phi \mathbf{true} = \emptyset}$$

$$\frac{\Gamma \vdash \phi S = \psi_1 \quad \Gamma \text{ AX S END } \vdash \Phi P = \psi_2 \quad \Gamma \vdash \alpha S = \psi_3}{\Gamma \vdash \Phi(\forall S \bullet P) = \psi_4} \quad \psi_4 = \psi_1 \cup (\psi_2 \setminus \psi_3)$$

However, for compactness, most of these rules will be presented in tabular form: see Table 3 for predicates and Table 4 for expressions.

Table 3: Free variable calculations for predicates

Conclusion	Premisses	Condition
$\Gamma \vdash \Phi(e \in s) = \psi_1$	$\Gamma \vdash \phi e = \psi_2$ $\Gamma \vdash \phi s = \psi_1$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \Phi(e = v) = \psi_1$	$\Gamma \vdash \phi e = \psi_2$ $\Gamma \vdash \phi v = \psi_1$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \Phi(\neg P) = \psi_1$	$\Gamma \vdash \Phi P = \psi_2$	$\psi_1 = \psi_2$
$\Gamma \vdash \Phi \mathbf{true} = \emptyset$		
$\Gamma \vdash \Phi \mathbf{false} = \emptyset$		
$\Gamma \vdash \Phi(P \wedge Q) = \psi_1$	$\Gamma \vdash \Phi P = \psi_2$ $\Gamma \vdash \Phi Q = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \Phi(\forall S \bullet P) = \psi_4$	$\Gamma \vdash \phi S = \psi_1$ $\Gamma S \text{ END} \vdash \Phi P = \psi_2$	$\psi_4 = \psi_1 \cup (\psi_2 \setminus \psi_3)$
$\Gamma \vdash \Phi(\exists_1 S \bullet P) = \psi_4$	$\Gamma \vdash \alpha S = \psi_3$ $\Gamma \vdash \phi S = \psi_1$ $\Gamma S \text{ END} \vdash \Phi P = \psi_2$	$\psi_4 = \psi_1 \cup (\psi_2 \setminus \psi_3)$
$\Gamma \vdash \Phi x = \psi_1$	$\Gamma \vdash \alpha x = \psi_2$ $\Gamma \vdash \phi x = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$

A.5 Substitution

The equivalences in the first column of Table 5 are sound in the context of the provisos and premisses in the second. This is equivalent to saying that the following table may be read as a presentation of inference rules. For each rule, the first column denotes the conclusion, and the second, the premiss.

In addition, we have simple rewrite rules for substitutions which do not require any context:

$$\begin{aligned}
 e &\xrightarrow{R} \langle x == e \rangle_{\circ x} \\
 \langle x == e \rangle_{\circ x} &\xrightarrow{R} e
 \end{aligned}$$

Table 4: Free variable calculations for expressions

Conclusion	Premisses	Condition
$\Gamma \vdash \phi(x) = \{x\}$		
$\Gamma \vdash \phi(x[e]) = \psi_1$	$\Gamma \vdash \phi e = \psi_2$	$\psi_1 = \{x\} \cup \psi_2$
$\Gamma \vdash \phi\{e_1, \dots, e_n\} = \psi$	$\Gamma \vdash \phi(e_1) = \psi_1$ \vdots $\Gamma \vdash \phi(e_n) = \psi_n$	$\psi = \psi_1 \cup \dots \cup \psi_n$
$\Gamma \vdash \phi\{S \bullet e\} = \psi_1$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi e = \psi_3$ $\Gamma \vdash \alpha S = \psi_4$	$\psi_1 = \psi_2 \cup (\psi_3 \setminus \psi_4)$
$\Gamma \vdash \phi(\mathbb{P} s) = \psi$	$\Gamma \vdash \phi(s) = \psi$	
$\Gamma \vdash \phi(e_1, \dots, e_n) = \psi$	$\Gamma \vdash \phi(e_1) = \psi_1$ \vdots $\Gamma \vdash \phi(e_n) = \psi_n$	$\psi = \psi_1 \cup \dots \cup \psi_n$
$\Gamma \vdash \phi(s_1 \times \dots \times s_n) = \psi$	$\Gamma \vdash \phi(s_1) = \psi_1$ \vdots $\Gamma \vdash \phi(s_n) = \psi_n$	$\psi = \psi_1 \cup \dots \cup \psi_n$
$\Gamma \vdash \phi(e.i) = \psi$	$\Gamma \vdash \phi(e) = \psi$	
$\Gamma \vdash \phi\langle x_1 == e_1, \dots, x_n == e_n \rangle = \psi$	$\Gamma \vdash \phi(e_1) = \psi_1$ \vdots $\Gamma \vdash \phi(e_n) = \psi_n$	$\psi = \psi_1 \cup \dots \cup \psi_n$
$\Gamma \vdash \phi(\theta S) = \psi$	$\Gamma \vdash \Phi S = \psi$	
$\Gamma \vdash \phi(b.x) = \psi$	$\Gamma \vdash \phi(b) = \psi$	
$\Gamma \vdash \phi(fe) = \psi$	$\Gamma \vdash \phi f = \psi_1$ $\Gamma \vdash \phi e = \psi_2$	$\psi = \psi_1 \cup \psi_2$
$\Gamma \vdash \phi(\mu S \bullet e) = \psi$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi e = \psi_3$ $\Gamma \vdash \alpha S = \psi_4$	$\psi_1 = \psi_2 \cup (\psi_3 \setminus \psi_4)$
$\Gamma \vdash \phi(b \circ e) = \psi_1$	$\Gamma \vdash \phi b = \psi_2$ $\Gamma \text{ AX } b \text{ END } \vdash \phi e = \psi_3$ $\Gamma \vdash \alpha b = \psi_4$	$\psi_1 = \psi_2 \cup (\psi_3 \setminus \psi_4)$

Conclusion	Premises	Proviso
$\Gamma \vdash \phi[x : s] = \psi$	$\Gamma \vdash \phi(s_1) = \psi$	
$\Gamma \vdash \phi[x == s] = \psi$	$\Gamma \vdash \phi(s_1) = \psi$	
$\Gamma \vdash \phi[S \mid P] = \psi_1$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \Phi P = \psi_3$ $\Gamma \vdash \alpha S = \psi_4$	$\psi_1 = \psi_2 \cup (\psi_3 \setminus \psi_4)$
$\Gamma \vdash \phi(\neg S) = \psi$	$\Gamma \vdash \phi S = \psi$	
$\Gamma \vdash \phi(S \wedge T) = \psi_1$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi T = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \phi(\text{pre } S) = \psi$	$\Gamma \vdash \phi(S) = \psi$	
$\Gamma \vdash \phi(S[x_1, \dots, x_n]) = \psi$	$\Gamma \vdash \phi S = \psi$	
$\Gamma \vdash \phi(\forall S \bullet T) = \psi_1$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi T = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \phi(\exists_1 S \bullet T) = \psi$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi T = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \phi(S[x_1/y_1, \dots, x_n/y_n]) = \psi$	$\Gamma \vdash \phi S = \psi$	
$\Gamma \vdash \phi(S \S T) = \psi_1$	$\Gamma \vdash \phi S = \psi_2$ $\Gamma \vdash \phi T = \psi_3$	$\psi_1 = \psi_2 \cup \psi_3$
$\Gamma \vdash \phi(S^g) = \psi$	$\Gamma \vdash \phi S = \psi$	

Table 5: Predicate Substitutions

Conclusion	Premiss	Proviso
$\Gamma \vdash \langle x == x \rangle \odot P \Leftrightarrow P$ $\Gamma \vdash \langle x == e \rangle \odot P \Leftrightarrow P$ $\Gamma \vdash \langle y == e \rangle \odot (\langle x == y \rangle \odot P) \Leftrightarrow \langle x == e \rangle \odot P$	$\Gamma \vdash \Phi P = \psi$ $\Gamma \text{ AX } x == e \text{ END}$ $\vdash \Phi P = \psi$	$'x' \notin \psi$ $'y' \notin \psi$
$\Gamma \vdash \langle y == u \rangle \odot (\langle x == e \rangle \odot P) \Leftrightarrow$ $\langle x == \langle y == u \rangle \odot e \rangle \odot (\langle y == u \rangle \odot P)$ $\Gamma \vdash \langle x_1 == e_1, \dots, x_{n-1} == e_{n-1}, x_n = e_n \rangle \odot P \Leftrightarrow$ $\langle x_1 == e_1, \dots, x_{n-1} == e_{n-1} \rangle \odot (\langle x_n = e_n \rangle \odot P)$	$\Gamma \vdash \phi u = \psi$ $\Gamma \vdash \phi e_n = \psi$	$'x' \neq 'y' \wedge 'x' \notin \psi$ $\psi \cap \{'x_1', \dots, 'x_n'\} = \emptyset$
$\Gamma \vdash \langle x == u \rangle \odot (e_1 = e_2) \Leftrightarrow (\langle x == u \rangle \odot e_1) = e_2$ $\Gamma \vdash \langle x == u \rangle \odot (e \in s) \Leftrightarrow (\langle x == u \rangle \odot e) \in s$ $\Gamma \vdash \langle x == u \rangle \odot (e \in s) \Leftrightarrow e \in \langle x == u \rangle \odot s$ $\Gamma \vdash b \odot \mathbf{true} \Leftrightarrow \mathbf{true}$ $\Gamma \vdash b \odot \neg P \Leftrightarrow \neg (b \odot P)$ $\Gamma \vdash b \odot (P \wedge Q) \Leftrightarrow (b \odot P) \wedge (b \odot Q)$ $\Gamma \vdash \langle x == e \rangle \odot \forall x: s \bullet P \Leftrightarrow \forall x: \langle x == e \rangle \odot s \bullet P$ $\Gamma \vdash \langle x == e \rangle \odot \exists y: s \bullet P \Leftrightarrow$ $\exists y: \langle x == e \rangle \odot s \bullet \langle x == e \rangle \odot P$ $\Gamma \vdash \langle x == e \rangle \odot \exists_1 x: s \bullet P \Leftrightarrow \exists_1 x: \langle x == e \rangle \odot s \bullet P$ $\Gamma \vdash \langle x == e \rangle \odot \exists_1 y: s \bullet P \Leftrightarrow$ $\exists_1 y: \langle x == e \rangle \odot s \bullet \langle x == e \rangle \odot P$	$\Gamma \vdash \phi e_2 = \psi$ $\Gamma \vdash \phi s = \psi$ $\Gamma \vdash \phi e = \psi$ $\Gamma \vdash \phi e = \psi$ $\Gamma \vdash \phi e = \psi$	$'x' \notin \psi$ $'x' \notin \psi$ $'x' \notin \psi$ $'y' \neq 'x' \wedge 'y' \notin \psi$ $'y' \neq 'x' \wedge 'y' \notin \psi$

Table 6: Expression Substitutions

Conclusion	Premisses	Proviso
$\Gamma \vdash e = \langle x == x \rangle \circ e$ $\Gamma \vdash e_1 = \langle x == e_2 \rangle \circ e_1$ $\Gamma \vdash (\langle y == e \rangle \circ \langle x == y \rangle \circ u) = \langle x == e \rangle \circ u$ $\Gamma \vdash \langle y == u \rangle \circ (\langle x == e \rangle \circ s) \Leftrightarrow$ $\langle x == \langle y == u \rangle \circ e \rangle \circ (\langle y == u \rangle \circ s)$ $\Gamma \vdash \langle x_1 == e_1, \dots, x_{n-1} == e_{n-1}, x_n = e_n \rangle \circ u \Leftrightarrow$ $\langle x_1 == e_1, \dots, x_{n-1} == e_{n-1} \rangle \circ (\langle x_n = e_n \rangle \circ u)$	$\Gamma \vdash \phi e_1 = \psi$ $\Gamma \text{ AX } x == e \text{ END}$ $\vdash \phi u = \psi$ $\Gamma \vdash \phi u = \psi$ $\Gamma \vdash \phi e_n = \psi$	$'x' \notin \psi$ $'y' \notin \psi$ $'x' \neq 'y' \wedge 'x' \notin \psi$ $\psi \cap \{'x_1', \dots, 'x_n'\} = \emptyset$
$\Gamma \vdash b \circ x = x$ $\Gamma \vdash b \circ x[y] = x[b \circ y]$ $\Gamma \vdash b \circ \{e_1, \dots, e_n\} = \{b \circ e_1, \dots, b \circ e_n\}$ $\Gamma \vdash b \circ \{S \bullet e\} = \{b \circ S \bullet b \circ e\}$ $\Gamma \vdash b \circ \mathbb{P} s = \mathbb{P} b \circ s$ $\Gamma \vdash b \circ (e_1, \dots, e_n) = (b \circ e_1, \dots, b \circ e_n)$ $\Gamma \vdash b \circ (s_1 \times \dots \times s_n) = b \circ s_1 \times \dots \times b \circ s_n$ $\Gamma \vdash b \circ (e.i) = (b \circ e).i$ $\Gamma \vdash b \circ \langle x_1 == e_1, \dots, x_n == e_n \rangle =$ $\langle x_1 == b \circ e_1, \dots, x_n == b \circ e_n \rangle$ $\Gamma \vdash b \circ \theta S = \theta b \circ S$ $\Gamma \vdash b \circ \theta S = b$ $\Gamma \vdash b_1 \circ b.x = (b_1 \circ b).x$ $\Gamma \vdash b \circ (f e) = (b \circ f)(b \circ e)$ $\Gamma \vdash b \circ (\mu S \bullet e) = \mu b \circ S \bullet b \circ e$	$\Gamma \vdash \alpha b = \psi$ $\Gamma \vdash \alpha b = \psi$ $\Gamma \text{ AX } \{b\} \text{ END } \vdash \alpha S = \psi_1$ $\Gamma \vdash \phi b = \psi_2$ $\Gamma \text{ AX } \{b\} \text{ END } \vdash \phi e = \psi_3$ $\Gamma \vdash \alpha b = \psi_1$ $\Gamma \vdash \alpha S = \psi_2$ $\Gamma \vdash \alpha b = \psi_1$ $\Gamma \vdash \alpha S = \psi_2$ $\Gamma \text{ AX } \{b\} \text{ END } \vdash \alpha S = \psi_1$ $\Gamma \vdash \phi b = \psi_2$ $\Gamma \text{ AX } \{b\} \text{ END } \vdash \phi e = \psi_3$	$'x' \notin \psi$ $'x' \notin \psi$ $\psi_1 \cap \psi_2 = \emptyset \wedge \psi_1 \cap \psi_3 = \emptyset$ $\psi_1 \cap \psi_2 = \emptyset$ $\psi_1 = \psi_2$ $\psi_1 \cap \psi_2 = \emptyset \wedge \psi_1 \cap \psi_3 = \emptyset$

Conclusion	Premisses	Proviso
$\Gamma \vdash b \circ [x : s] = [x : b \circ s]$		
$\Gamma \vdash b \circ [x == e] = [x == b \circ e]$		
$\Gamma \vdash \Gamma \vdash b \circ [S \mid P] = [b \circ S \mid b \circ P]$	$\Gamma \vdash \alpha b = \psi_1$	$\psi_1 \cap \psi_2 = \emptyset$
	$\Gamma \vdash \alpha S = \psi_2$	
$\Gamma \vdash b \circ [S \mid P] = [b \circ S \mid P]$	$\Gamma \vdash \alpha b = \psi_1$	$\psi_1 \cap \psi_2 \subseteq \psi_3$
	$\Gamma \vdash \Phi P = \psi_2$	
	$\Gamma \vdash \alpha S = \psi_3$	
$\Gamma \vdash b \circ [\neg S] = [\neg b \circ S]$		
$\Gamma \vdash b \circ [S \wedge T] = [b \circ S \wedge b \circ T]$		
$\Gamma \vdash b \circ \text{pre } S = \text{pre}(b \circ S)$		
$\Gamma \vdash b \circ S \setminus [x_1, \dots, x_n] = (b \circ S) \setminus [x_1, \dots, x_n]$		
$\Gamma \vdash b \circ [\forall S \bullet T] = [\forall b \circ S \bullet b \circ T]$		
$\Gamma \vdash b \circ [\exists_1 S \bullet T] = [\exists_1 b \circ S \bullet b \circ T]$		
$\Gamma \vdash b \circ [S[x_1/y_1, \dots, x_n/y_n]] = [b \circ S[x_1/y_1, \dots, x_n/y_n]]$		
$\Gamma \vdash b \circ [S \circledast T] = [b \circ S \circledast b \circ T]$		
$\Gamma \vdash b \circ [S^q] = [(b \circ S)^q]$		

A.6 Logical inference rules

A.6.1 Structural rules

Assumption rules

$$\frac{}{\Gamma \text{AX}[[P] \text{END}] \vdash P} \text{AssumPred}$$

$$\frac{\Gamma \vdash \phi e = \psi}{\Gamma \text{AX}[x == e] \text{END} \vdash x = e} \text{AssumDef}('x' \notin \psi)$$

$$\frac{\Gamma \vdash \phi s = \psi}{\Gamma \text{AX}[x : s] \text{END} \vdash x \in s} \text{AssumDecl}('x' \notin \psi)$$

$$\frac{\Gamma \vdash \alpha S = \psi_1 \quad \Gamma \vdash \phi S = \psi_2}{\Gamma \text{AX} S \text{END} \vdash \theta S \in S} \text{AssumSch}(\psi_1 \cap \psi_2 = \emptyset)$$

Paragraph and thinning rules

$$\frac{\Gamma \text{AX}[[P \wedge R] \text{END}] \vdash Q}{\Gamma \text{AX}[[P] \text{END}] \text{AX}[[R] \text{END}] \vdash Q} \text{PredConj}$$

$$\frac{\Gamma \text{AX} S \text{END} \text{AX}[[P] \text{END}] \vdash Q}{\Gamma \text{AX}[S | P] \text{END} \vdash Q} \text{SchPred}$$

$$\frac{\Gamma \vdash P}{\Pi \Gamma \vdash P} \text{Thinl}$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash \alpha \Pi = \psi_1 \quad \Gamma \Pi \vdash \Phi P = \psi_2}{\Gamma \Pi \vdash P} \text{Thinr}(\psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\begin{array}{l} \Gamma_1 \Gamma_2 \Pi \vdash P \\ \Gamma_1 \vdash \alpha \Gamma_2 = \psi_1 \\ \Gamma_1 \vdash \alpha \Pi = \psi_2 \\ \Gamma_1 \Gamma_2 \vdash \phi \Pi = \psi_3 \\ \Gamma_1 \Pi \vdash \phi \Gamma_2 = \psi_4 \end{array}}{\Gamma_1 \Pi \Gamma_2 \vdash P} \text{Shift} \left(\begin{array}{l} \psi_1 \cap \psi_3 = \emptyset \\ \psi_2 \cap \psi_4 = \emptyset \\ \psi_1 \cap \psi_2 = \emptyset \end{array} \right)$$

A.6.2 Equality and substitution

$$\frac{}{\Gamma \vdash e = e} \text{Refl}$$

$$\frac{\Gamma \vdash u = e}{\Gamma \vdash e = u} \text{Symm}$$

$$\frac{\Gamma \text{AX}[[u = e] \text{END}] \vdash v = e}{\Gamma \text{AX}[[u = e] \text{END}] \vdash v = u} \textit{Trans}$$

$$\frac{\Gamma \text{AX}\{b\} \text{END} \vdash P}{\Gamma \vdash b \odot P} \textit{UseBind}$$

$$\frac{\Gamma \text{AX}\{b\} \text{END} \vdash u = e \quad \Gamma \vdash \alpha b = \psi_1 \quad \Gamma \vdash \phi u = \psi_2}{\Gamma \vdash u = b \odot e} \textit{EquBind} (\psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\Gamma \vdash u = b \odot e \quad \Gamma \vdash \alpha b = \psi_1 \quad \Gamma \vdash \phi u = \psi_2}{\Gamma \text{AX}\{b\} \text{END} \vdash u = e} \textit{EquBind}' (\psi_1 \cap \psi_2 = \emptyset)$$

A.6.3 Propositional calculus

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q} \textit{AndI}$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P} \textit{AndEr}$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q} \textit{AndEl}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q} \textit{OrIr}$$

$$\frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q} \textit{OrIl}$$

$$\frac{\Gamma \vdash P \vee Q \quad \Gamma \text{AX}[[P] \text{END}] \vdash R \quad \Gamma \text{AX}[[Q] \text{END}] \vdash R}{\Gamma \vdash R} \textit{OrE}$$

$$\frac{\Gamma \text{AX}[[P] \text{END}] \vdash Q}{\Gamma \vdash P \Rightarrow Q} \textit{ImpI}$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash P \Rightarrow Q}{\Gamma \vdash Q} \textit{ImpE}$$

$$\frac{\Gamma \vdash P \Rightarrow \mathbf{false}}{\Gamma \vdash \neg P} \textit{NotDef}$$

$$\frac{}{\Gamma \vdash \mathbf{false} \Rightarrow P} \text{FalseDef}$$

$$\frac{\Gamma \vdash \mathbf{false}}{\Gamma \vdash P} \text{FalseE}$$

$$\frac{\Gamma \text{ AX}[[\neg P] \text{ END}] \vdash \mathbf{false}}{\Gamma \vdash P} \text{NotE}$$

$$\frac{}{\Gamma \vdash P \Rightarrow \mathbf{true}} \text{TrueDef}$$

$$\frac{\Gamma \vdash P \Rightarrow Q \wedge Q \Rightarrow P}{\Gamma \vdash P \Leftrightarrow Q} \text{IffDef}$$

A.6.4 Quantifier rules

$$\frac{\Gamma \text{ AX } S \text{ END} \vdash P}{\Gamma \vdash \forall S \bullet P} \text{AllI}$$

$$\frac{\Gamma \vdash \forall S \bullet P \quad \Gamma \vdash b \in S}{\Gamma \vdash b \odot P} \text{AllE}$$

$$\frac{\Gamma \vdash b \odot P \quad \Gamma \vdash b \in S}{\Gamma \vdash \exists S \bullet P} \text{ExistsI}$$

$$\frac{\begin{array}{l} \Gamma \vdash \exists S \bullet P \\ \Gamma \text{ AX } S \text{ END AX}[[P] \text{ END}] \vdash Q \\ \Gamma \vdash \alpha S = \psi_1 \\ \Gamma \vdash \Phi Q = \psi_2 \end{array}}{\Gamma \vdash Q} \text{ExistsE} (\psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\begin{array}{l} \Gamma \vdash \exists S \bullet P \wedge \\ \forall S' \bullet \langle x_1 == x'_1, \dots, x_n == x'_n \rangle \odot P \Rightarrow \theta S = \theta S' \\ \Gamma \vdash \alpha S = \{x'_1, \dots, x'_n\} \end{array}}{\Gamma \vdash \exists_1 S \bullet P} \text{UniqExists}$$

$$\frac{\begin{array}{l} \Gamma \vdash \exists_1 S \bullet P \\ \Gamma \vdash \alpha S = \{x'_1, \dots, x'_n\} \end{array}}{\Gamma \vdash \exists S \bullet P \wedge \\ \forall S' \bullet \langle x_1 == x'_1, \dots, x_n == x'_n \rangle \odot P \Rightarrow \theta S = \theta S'} \text{UniqExists'}$$

A.6.5 Expression rules

Sets

$$\frac{\begin{array}{c} \Gamma \text{GENAX}[x]T \text{END} \vdash \{ \langle x == s \rangle \circ T \bullet y \} \in \mathbb{P}e \\ \Gamma \vdash \alpha T = \psi_1 \\ \Gamma \vdash \Phi T = \psi_2 \end{array}}{\Gamma \text{GENAX}[x]T \text{END} \vdash y_{[s]} \in e} \text{GenMem} ('y' \in \psi_1 \wedge \psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\begin{array}{c} \Gamma \text{GENAX}[x]T \text{END} \vdash y_{[s]} \in e \\ \Gamma \vdash \alpha T = \psi_1 \\ \Gamma \vdash \Phi T = \psi_2 \end{array}}{\Gamma \text{GENAX}[x]T \text{END} \vdash \{ \langle x == s \rangle \circ T \bullet y \} \in \mathbb{P}e} \text{GenMem}' ('y' \in \psi_1 \wedge \psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\begin{array}{c} \Gamma \vdash (\forall x : s \bullet x \in t) \wedge (\forall x : t \bullet x \in s) \\ \Gamma \vdash \phi s = \psi_1 \\ \Gamma \vdash \phi t = \psi_2 \end{array}}{\Gamma \vdash s = t} \text{Seteq} ('x' \notin \psi_1 \cup \psi_2)$$

$$\frac{\begin{array}{c} \Gamma \vdash s = t \\ \Gamma \vdash \phi s = \psi_1 \\ \Gamma \vdash \phi t = \psi_2 \end{array}}{\Gamma \vdash (\forall x : s \bullet x \in t) \wedge (\forall x : t \bullet x \in s)} \text{Seteq}' ('x' \notin \psi_1 \cup \psi_2)$$

$$\frac{\Gamma \vdash v = e_1 \vee \dots \vee v = e_n}{\Gamma \vdash v \in \{e_1, \dots, e_n\}} \text{Extmem}$$

$$\frac{\Gamma \vdash \exists S \bullet e = u \quad \Gamma \vdash \phi e = \psi_1 \quad \Gamma \vdash \alpha S = \psi_2}{\Gamma \vdash e \in \{S \bullet u\}} \text{Setcomp} (\psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\Gamma \vdash e \in \{S \bullet u\} \quad \Gamma \vdash \phi e = \psi_1 \quad \Gamma \vdash \alpha S = \psi_2}{\Gamma \vdash \exists S \bullet e = u} \text{Setcomp}' (\psi_1 \cap \psi_2 = \emptyset)$$

$$\frac{\Gamma \vdash \forall x : t \bullet x \in s \quad \Gamma \vdash \phi s = \psi_1}{\Gamma \vdash t \in \mathbb{P}s} \text{Powerset} ('x' \notin \psi_1)$$

$$\frac{\Gamma \vdash t \in \mathbb{P}s \quad \Gamma \vdash \phi s = \psi_1}{\Gamma \vdash \forall x : t \bullet x \in s} \text{Powerset}' ('x' \notin \psi_1)$$

Cartesian products

$$\frac{\Gamma \vdash v = e_i}{\Gamma \vdash v = (e_1, \dots, e_n).i} \text{Tupleequ} (1 \leq i \leq n)$$

$$\frac{\Gamma \vdash u.1 \in s_1 \wedge \dots \wedge u.n \in s_n}{\Gamma \vdash u \in s_1 \times \dots \times s_n} \text{Prodmem}$$

$$\frac{\Gamma \vdash u = (e_1, \dots, e_n)}{\Gamma \vdash u.1 = e_1 \wedge \dots \wedge u.n = e_n} \text{Tuplesel}$$

Labelled products

$$\frac{}{\Gamma \vdash \langle x_1 == e_1, \dots, x_n == e_n \rangle . x_i = e_i} \text{BindEqu } (1 \leq i \leq n)$$

$$\frac{\Gamma \vdash y . x_1 = e_1 \wedge \dots \wedge u . x_n = e_n}{\Gamma \vdash y = \langle x_1 == e_1, \dots, x_n == e_n \rangle} \text{BindSel}$$

$$\frac{\Gamma \vdash \alpha b = \psi_1 \quad \Gamma \vdash \phi b = \psi_2}{\Gamma \text{ AX}\{b\} \text{ END} \vdash x = b . x} \text{BindMem } ('x' \in \psi_1 \wedge \psi_1 \cap \psi_2 = \emptyset)$$

Schemas

$$\frac{\Gamma \vdash e . x_1 = x_1 \wedge \dots \wedge e . x_n = x_n \quad \Gamma \vdash S}{\Gamma \vdash e = \theta S} \text{ThetaEqu}$$

$$\frac{\Gamma \vdash \langle x_1 == x_1, \dots, x_n == x_n \rangle \in S \quad \Gamma \vdash \alpha S = \{ 'x_1', \dots, 'x_n' \}}{\Gamma \vdash S} \text{BindSch}$$

$$\frac{\Gamma \vdash S \quad \Gamma \vdash \alpha S = \{ 'x_1', \dots, 'x_n' \}}{\Gamma \vdash \langle x_1 == x_1, \dots, x_n == x_n \rangle \in S} \text{BindSch}'$$

Description

$$\frac{\Gamma \vdash (e, u) \in f \quad \Gamma \vdash \forall y : f \bullet (y.1 = e) \Rightarrow (y.2 = u) \quad \Gamma \vdash \phi e = \psi_1 \quad \Gamma \vdash \phi u = \psi_2}{\Gamma \vdash u = f e} \text{FunctApp } ('y' \notin \psi_1 \cup \psi_2)$$

$$\frac{\Gamma \vdash e \in s \quad \Gamma \vdash \langle x == e \rangle \odot P \quad \Gamma \text{ AX}[y : s] \text{ END} \vdash \langle x == y \rangle \odot P \Rightarrow y = e}{\Gamma \vdash e = \mu x : s \mid P} \text{DefnDescr}$$

$$\frac{\Gamma \vdash v = \langle x == u \rangle \odot e \quad \Gamma \vdash \phi u = \psi_1}{\Gamma \text{ AX}[x == u] \text{ END} \vdash v = e} \text{Usedef } ('x' \notin \psi_1)$$

$$\frac{\Gamma \text{ AX}[x == u] \text{ END} \vdash v = e \quad \Gamma \vdash \phi u = \psi_1}{\Gamma \vdash v = \langle x == u \rangle \odot e} \text{Usedef}' ('x' \notin \psi_1)$$

A.6.6 Schema calculus

$$\frac{\Gamma \vdash u \in [x_1 : s_1]}{\Gamma \vdash u.x_1 \in s_1} \text{BindProd}$$

$$\frac{\Gamma \vdash b \in S \wedge b \odot P}{\Gamma \vdash b \in [S \mid P]} \text{SchemaMem}$$

$$\frac{\begin{array}{l} \Gamma \vdash b \odot S \\ \Gamma \vdash \phi S = \psi_1 \\ \Gamma \vdash \alpha S = \psi_2 \\ \Gamma \vdash \alpha b = \psi_3 \end{array}}{\Gamma \vdash b \in S} \text{SchBindMem} (\psi_1 \cap \psi_2 = \emptyset \wedge \psi_2 = \psi_3)$$

$$\frac{\begin{array}{l} \Gamma \vdash b \in S \\ \Gamma \vdash \phi S = \psi_1 \\ \Gamma \vdash \alpha S = \psi_2 \\ \Gamma \vdash \alpha b = \psi_3 \end{array}}{\Gamma \vdash b \odot S} \text{SchBindMem}'' (\psi_1 \cap \psi_2 = \emptyset \wedge \psi_2 = \psi_3)$$

$$\frac{\begin{array}{l} \Gamma \vdash b \odot [b \circ S] \\ \Gamma \vdash \phi b = \psi_1 \\ \Gamma \vdash \alpha b = \psi_2 \\ \Gamma \vdash \alpha S = \psi_3 \end{array}}{\Gamma \vdash b \in S} \text{SchBindMem}' (\psi_1 \cap \psi_2 = \emptyset \wedge \psi_2 = \psi_3)$$

$$\frac{\begin{array}{l} \Gamma \vdash b \in S \\ \Gamma \vdash \phi b = \psi_1 \\ \Gamma \vdash \alpha b = \psi_2 \\ \Gamma \vdash \alpha S = \psi_3 \end{array}}{\Gamma \vdash b \odot [b \circ S]} \text{SchBindMem}''' (\psi_1 \cap \psi_2 = \emptyset \wedge \psi_2 = \psi_3)$$

A.7 Using the deductive system

This deductive system permits proofs to be carried out entirely within the Z notation (with some additional syntax, if binding substitution is to be omitted from the language). It avoids the need to map Z terms into an underlying representation.

Proofs in the deductive system proceed in the way that is usual for the sequent calculus: proofs are developed *backwards*, starting from the sequent to be proved. A rule is applied, resulting in fresh sequents which must be proved. This process continues until there are no more sequents requiring proof, in which case the original sequent is proved.

A completed proof may thus be represented as a tree, with the proved sequent as the root node, and every leaf node containing an empty list of sequents. However, if some of these lists in the leaves are non-empty, then the derivation tree is still useful, although it does not represent a proof, it represents a partial proof—or a proof of a derived rule of inference.

Given the derivation tree

$$\frac{S_1 \quad \dots \quad \frac{S_{i1} \quad \dots \quad S_{im}}{S_i} R_i(P_i) \quad \dots \quad S_n}{Seq} R(P)$$

we may use as a derived rule:

$$\frac{S_1 \quad \dots \quad S_{i1} \quad \dots \quad S_{im} \quad \dots \quad S_n}{Seq} R'(P, P_i)$$

The deductive system is defined over the abstract syntax only. However, the transformation rules found elsewhere in the Standard may be used to enrich the deductive system: they may either be used as additional inference rules, as below, or in the derivation of derived inference rules which may involve terms of the concrete syntax which are not present in the abstract syntax.

Given the rewrite rule for any predicates

$$P \xrightarrow{R} Q$$

the inference rule which replaces occurrences of P with occurrences of Q is sound. That is, if $\Delta(P)$ is some sequent containing instances of predicate P , then the inference rule

$$\frac{\Delta(P)}{\Delta(Q)}$$

is sound. Similar remarks apply to expressions.

A fuller explanation of the use of the deductive system may be found in the reference (Brien & Martin 1995). In particular, those notes show how the rules for the schema calculus may be derived from those presented here.

□