A Typed Formulation of the Semantics of Z

R.D. Arthan
rda@lemma-one.com

3rd August 2005

Abstract

This document is a companion to Ian Toyn's presentation of the semantics of Z that is now in the ISO Z standard. It contains a reformulation of the semantics within the ProofPower-Z dialect of Z that has been type-checked using ProofPower. The purpose of the document is to act as a check on the definition of the semantics. It also highlights certain points of interest in the required structure of the semantic universe $\mathbb{U}_Z$.

Change History

Version 1	Issued as Z document reference D-235, corresponding to the first complete version of the semantics.	20/05/1998
Version 2	Issued as Z document reference D-256, addressing comments received from various members of the standards committe and corresponding to the semantics as they appeared in a contemporary draft of the standard.	10/05/2000
Version 3	Accommodating a small change to the ProofPower Z toolkit (U has been rechristened as $\mathbb{U}$).	18/02/2005
Version 4	Bringing the specification into line with the 2002 ISO standard, and including a proposed change to the semantics of schema universal quantification.	03/08/2005

1 INTRODUCTION

In 1998, Ian Toyn and I developed a definition of the semantics of the kernel sublanguage of Z. With some modifications as a result of review, this work is now captured in the ISO Z standard. The definition is written in first-order set theory using a surface syntax borrowed from Z itself.

An advantage of using the Z-like surface syntax is that with only small adjustments, and without having to define a large number of auxiliary functions, the definition can be viewed as a Z specification and so can be analysed using tools that support Z. As a step in this direction, this document presents a reformulation of the semantic definitions in the ProofPower-Z dialect.

Processing the specification with ProofPower reveals that the semantic definitions are now well-typed in some sense. It is, presumably, useful to know this, since preparing this Z version revealed several minor errors and highlighted some issues. The Z formulation also points up the places at which the type constraints of Z kick in to ensure that the semantic equations are well-defined.

This document is subject to the following cautionary remarks:

- No attempt has been made to make the formal material in this document intelligible independently of the narrative in the ISO Z standard: please do *not* start here!
- In several places, I have simply said that things are done in the same order as the original. The intention is that you can put the two documents side by side and compare, and you may well need to do that in order to understand what has been done here.
- Meaning has been sacrificed in favour of surface syntax in this document. A reasonably close syntactic correspondence between the original and the Z version has been maintained by trickery that would detract from semantic analysis of the semantics.
- The transcription into Z has not been done by automated means. Errors of transcription that are not detected by Z type-checking may well persist.
- The current version of this document corresponds to the semantics given in the 2002 ISO Standard for Z (ISO/IEC 13568:2002).
- Some observations concerning on the 2002 ISO Standard have been included. See sections 3.2, 4.6 and 4.6.13 below.

The remainder of this document is organised as follows:

Section 2 describes our version of the semantic universe. Because we have to obey the Z type discipline, several transfer functions are needed to stand in for the untyped constructions of the original.

Section 3 gives a Z definition of the kernel abstract syntax sufficient for present needs.

Section 4 gives a few preliminaries that enable us to give the equations in a syntax like that of the original and then gives the Z axioms that model the semantic equations.

Section 5 lists the type assignment generated by ProofPower for the specification.

Section 6 gives an index to the global variables declared in the specification. Global variables are generally shown in bold face at the point of their declaration.

2 SEMANTIC UNIVERSE

Our setup for the semantic universe is necessarily slightly different from the untyped original. First we introduce $\mathbb{U}_Z$ as a given set and then introduce its subsets *NAME* and $\mathbb{W}$:

$[\ \mathbb{U}_{\boldsymbol{Z}}\ ]$

$$\boldsymbol{NAME}, \mathbb{W} : \mathbb{P}\ \mathbb{U}_Z$$

$$NAME \subseteq \mathbb{W}$$

$\mathbb{W}$ is a model of a world of pure sets — each member of $\mathbb{W}$ itself represents a set of elements of $\mathbb{W}$. To describe this situation in Z, one possibility would be to introduce a binary relation on $\mathbb{W}$ corresponding to the membership relation in the world of sets. In the present context, it is technically more convenient to introduce a function, η, that maps an element $A \in \mathbb{W}$ to the subset of $\mathbb{W}$ that A represents. We call $\eta(A)$ the *extent* of A. Since we expect the world of sets to be extensional, i.e., we expect two sets to be equal if and only if they have the same elements, η will be an injective function. We also expect that, for any $A \in \mathbb{W}$, there will be a unique element of $\mathbb{W}$ representing the power set of A. Accordingly, we introduce a function, $\mathbb{P}_W$, that sends a set $A \in \mathbb{W}$ to the member of $\mathbb{W}$ that represents its power set. The expected interrelationship of the extent and power set functions gives rise to the defining property in the following:

$$\boldsymbol{\eta} : \mathbb{W} \rightarrowtail \mathbb{P}\ \mathbb{W};$$
$$\mathbb{P}_{\boldsymbol{W}} : \mathbb{W} \rightarrowtail \mathbb{W}$$

$$\forall\ w : \mathbb{W} \bullet\ \eta(\!|\eta(\mathbb{P}_W\ w)|\!) = \mathbb{P}(\eta\ w)$$

Z having reserved superscription for something else, instead of A^k, we use the notation $A \uparrow k$ for the k-fold cartesian product of a set A:

fun 10 $\_ \uparrow \_$

$[X]$

$$\_\uparrow\_ : (\mathbb{P}\ X \times \mathbb{N}) \rightarrow \mathbb{P}\ (seq\ X)$$

$$\forall A : \mathbb{P}X;\ k : \mathbb{N} \bullet\ A \uparrow k = (1..k) \rightarrow A$$

Now we define functions that transfer between various typed constructions on $\mathbb{W}$ and the untyped universe $\mathbb{U}_Z$. There are three such constructions corresponding to the semantic values of bindings (β), finite sets (ϕ), tuples (χ), and generics (γ). All but the last of these constructions can be carried out within $\mathbb{W}$ itself. We give axioms relating the first four to one another and to the extent operator, η. No property is required of γ other than that it be an injection.

$$
\begin{array}{l}
\boldsymbol{\beta} : (NAME \mathrel{\text{⇻}} \mathbb{W}) \rightarrowtail \mathbb{W};\\
\boldsymbol{\phi} : \mathbb{F}\ \mathbb{W} \rightarrowtail \mathbb{W};\\
\boldsymbol{\chi} : \bigcup\{k : \mathbb{N}_1 \bullet \mathbb{W} \uparrow k\} \rightarrowtail \mathbb{W};\\
\boldsymbol{\gamma} : \bigcup\{k : \mathbb{N}_1 \bullet (\mathbb{W} \uparrow k) \rightarrow \mathbb{W}\} \rightarrowtail \mathbb{U}_Z\\
\hline
\forall i : NAME;\ w : \mathbb{W};\ t : NAME \mathrel{\text{⇻}} \mathbb{W} \bullet\\
\qquad i \mapsto w \in t \Leftrightarrow \chi\ \langle i,\ w\rangle \in \eta(\beta\ t);\\
\forall a : \mathbb{F}\ \mathbb{W};\ w : \mathbb{W} \bullet\ w \in a \Leftrightarrow w \in \eta(\phi\ a);\\
\exists\ \nu : \mathbb{N} \rightarrowtail \mathbb{W} \bullet \forall s : \bigcup\{k : \mathbb{N}_1 \bullet \mathbb{W} \uparrow k\};\ i : \mathbb{N};\ w : \mathbb{W} \bullet\\
\qquad i \mapsto w \in s \Leftrightarrow \chi\ \langle \nu\ i,\ w\rangle \in \eta(\chi\ \ s)
\end{array}
$$

Small names (lower-case Greek letters) have been deliberately chosen for the various transfer functions above. All such functions would be represented by the identity function in an untyped universe. So when comparing this document with the original, if you see a lower-case Greek letter or a composite of Greek letters and possibly their inverses, you can simply ignore it. The inverses crop up at precisely those points where the type rules imply that an object will have a certain form (say a binding) and so will be in the domain of the appropriate inverse function ($\beta^{\sim}$).

3 SYNTAX

We need to develop a Z model of the abstract syntax which forms the domain of the semantic bracket functions. The treatment below is completely customised for the task at hand. E.g., type annotations are only inserted where they are actually used.

We make use of fixity declarations and exploit the fact that ProofPower does not require the chevrons in a free type definition. Within each category, the various alternatives are listed in the same order as the corresponding semantic equations are given in the original, q.v. Occasionally, we introduce additional syntactic categories (e.g., for an individual declaration) just to gain a surface syntax effect. Dependencies between the categories mean that the categories are treated in the opposite order to the original (bottom-up rather than top-down).

3.1 Type

$$
\begin{array}{ll}
\mathit{fun} & \_ :_t \_,\\
& [_t \ldots]_t,\\
& (_\times \ldots)_\times,\\
& \lambda_t \ldots \bullet_t \_
\end{array}
$$

$$
\begin{array}{lll}
\boldsymbol{TYPE} & ::= & \boldsymbol{given}\ NAME \\
 & | & \boldsymbol{generic}\ NAME \\
 & | & \mathbb{P}_{\boldsymbol{t}}\ TYPE \\
 & | & (_\times\ seq\ TYPE\ )_\times \\
 & | & [_{\boldsymbol{t}}\ seq\ DECL\ ]_{\boldsymbol{t}} \\
 & | & \boldsymbol{\lambda_t}\ seq\ NAME\ \bullet_{\boldsymbol{t}}\ TYPE \\
\&\ \boldsymbol{DECL} & ::= & NAME\ \boldsymbol{:_t}\ TYPE
\end{array}
$$

3.2 Expression and Predicate

$$
\begin{array}{ll}
fun & geninst\ \_\ [_g\ \cdots\ ]_g, \\
 & \{_e\ \cdots\ \}_e, \\
 & \{_c\ \_\ \bullet_c\ \_\ \}_c, \\
 & (_e\ \cdots\ )_e, \\
 & (_t\ \cdots\ )_t, \\
 & (_b\ \cdots\ )_b, \\
 & \mu_d\ \_\ \bullet_d\ \_, \\
 & [_v\ \_\ ]_v, \\
 & [_s\ \_\ |_s\ \_\ ]_s, \\
 & \_\ {}_o^o{}_e\ \_, \\
 & \neg_e\ \_, \\
 & \_\ \wedge_e\ \_, \\
 & \forall_s\ \_\ \bullet_s\ \_, \\
 & \_\ [_r\ \cdots\ ]_r, \\
 & \_\ ==_b\ \_, \\
 & \_\ {}_o^o{}_d\ \_, \\
 & \_\ /_r\ \_, \\
 & \_\ \in_p\ \_, \\
 & \forall_p\ \_\ \bullet_p\ \_, \\
 & \_\ \wedge_p\ \_
\end{array}
$$

$$
\begin{array}{lll}
\boldsymbol{EXPRESSION} & ::= & \boldsymbol{var}\ NAME \\
& | & \boldsymbol{geninst}\ NAME\ [_{\boldsymbol{g}}\ seq\ EXPRESSION\ ]_{\boldsymbol{g}} \\
& | & \{_{\boldsymbol{e}}\ seq\ EXPRESSION\ \}_{\boldsymbol{e}} \\
& | & \{_{\boldsymbol{c}}\ EXPRESSION\ \bullet_{\boldsymbol{c}}\ EXPRESSION\ \}_{\boldsymbol{c}} \\
& | & \mathbb{P}_{\boldsymbol{p}}\ EXPRESSION \\
& | & (_{\boldsymbol{t}}\ seq\ EXPRESSION\ )_{\boldsymbol{t}} \\
& | & (_{\boldsymbol{b}}\ seq\ BIND\ )_{\boldsymbol{b}} \\
& | & \boldsymbol{\mu}_{\boldsymbol{d}}\ EXPRESSION\ \bullet_{\boldsymbol{d}}\ EXPRESSION \\
& | & [_{\boldsymbol{v}}\ DEC\ ]_{\boldsymbol{v}} \\
& | & [_{\boldsymbol{s}}\ EXPRESSION\ |_{\boldsymbol{s}}\ PREDICATE\ ]_{\boldsymbol{s}} \\
& | & \neg_{\boldsymbol{e}}\ EXPRESSION \\
& | & EXPRESSION\ \wedge_{\boldsymbol{e}}\ EXPRESSION \\
& | & \forall_{\boldsymbol{s}}\ EXPRESSION\ \bullet_{\boldsymbol{s}}\ EXPRESSION \\
& | & EXPRESSION\ [_{\boldsymbol{r}}\ seq\ RENAME\ ]_{\boldsymbol{r}} \\
& | & EXPRESSION\ \circ\!\!\!\circ_{\boldsymbol{e}}\ TYPE \\
\&\ \boldsymbol{BIND} & ::= & NAME\ ==_{\boldsymbol{b}}\ EXPRESSION \\
\&\ \boldsymbol{DEC} & ::= & NAME\ \circ\!\!\!\circ_{\boldsymbol{d}}\ EXPRESSION \\
\&\ \boldsymbol{RENAME} & ::= & NAME\ /_{\boldsymbol{r}}\ NAME \\
\&\ \boldsymbol{PREDICATE} & ::= & EXPRESSION\ \in_{\boldsymbol{p}}\ EXPRESSION \\
& | & \boldsymbol{true_p} \\
& | & \neg_{\boldsymbol{p}}\ PREDICATE \\
& | & PREDICATE\ \wedge_{\boldsymbol{p}}\ PREDICATE \\
& | & \forall_{\boldsymbol{p}}\ EXPRESSION\ \bullet_{\boldsymbol{p}}\ PREDICATE
\end{array}
$$

Observation A: In earlier version of this document, the type ascriptions were only given on the specific constructs that needed them (i.e., schema negation, schema conjunction and schema universal conjunction). However, the proposed amendment to the semantic equation for schema universal conjunction (see section 4.6.13 below) requires a type ascription on the second operand as well as on the expression as a whole. The above now reflects the ISO Standard in allowing a type ascription to be attached to any form of expression.

3.3 Paragraph

$$
\begin{array}{ll}
fun & [_d\ \ldots\ ]_d, \\
& GENAX\ \ \ldots\ (_g\ \_\ \circ\!\!\!\circ_g\ \_\ )_g, \\
& [_\vdash\ \ldots\ ]_\vdash\ \_
\end{array}
$$

$$
\begin{array}{lll}
\boldsymbol{PARAGRAPH} & ::= & [_{\boldsymbol{d}}\ seq\ NAME\ ]_{\boldsymbol{d}} \\
& | & \boldsymbol{AX}\ EXPRESSION \\
& | & \boldsymbol{GENAX}\ seq\ NAME\ (_{\boldsymbol{g}}\ EXPRESSION\ \circ\!\!\!\circ_{\boldsymbol{g}}\ TYPE\ )_{\boldsymbol{g}} \\
& | & \vdash_{\boldsymbol{d}}\ PREDICATE \\
& | & [_\vdash\ seq\ NAME\ ]_\vdash\ PREDICATE
\end{array}
$$

3.4 Section

fun *section* _ *parents* ... *end* ... *END*

$\boldsymbol{SECTION} ::= \boldsymbol{section}\ NAME\ \boldsymbol{parents}\ seq\ NAME\ \boldsymbol{end}\ seq\ PARAGRAPH\ \boldsymbol{END}$

3.5 Specification

$\boldsymbol{SPECIFICATION} ::= \boldsymbol{spec}\ (seq\ SECTION)$

3.6 Decoration

The semantics use the reserved strokes ♡ and ♠ to distinguish given type names from generic formal parameter names. The function *decor* is used to apply ordinary decorations (strokes) and these special decorations to names. To declare this function we define a given type of strokes (corresponding to the syntactic category `STROKE` in the lexis) and extend it with the two reserved strokes to give a free type, *DECORATOR* representing a set which does not have an official name in the standard.

$[\ \boldsymbol{STROKE}\ ]$

$\boldsymbol{DECORATOR} ::= \boldsymbol{stroke}\ STROKE \mid \heartsuit \mid \spadesuit$

We can now declare *decor*. Note that both the arrows are injections: each *DECORATOR* determines its own unique injection of the set of names into itself.

$\boldsymbol{decor} : DECORATOR \rightarrowtail NAME \rightarrowtail NAME$

4 SEMANTICS

Now after just a few more preliminaries, we can get down to business. Section 4.1 gives the preliminaries and then sections 4.2 to 4.7 give the semantic equations in exactly the same order as the original.

4.1 Preliminaries

We define *Model* and *SectionModels* just as in the original and introduce the name of the prelude

$\boldsymbol{Model} \mathrel{\widehat=} NAME \twoheadrightarrow\!\!\!\!\!\!\;\shortmid\;\; \mathbb{U}_Z$

$\boldsymbol{SectionModels} \mathrel{\widehat=} NAME \twoheadrightarrow\!\!\!\!\!\!\;\shortmid\;\; \mathbb{P} Model$

$\boldsymbol{prelude} : NAME$

Now we introduce the semantic brackets. In principle, we could give just one or more large axiomatic descriptions containing all the equations. That would be appropriate if one wished to model the semantics of the semantics more accurately than we do. Our goal is to stay near the surface syntax of the original, so we will introduce the semantic equations as individual axioms.

fun $[\![_Z \ \_ \ ]\!]_Z$,
$[\![_S \ \_ \ ]\!]_S$,
$[\![_D \ \_ \ ]\!]_D$,
$[\![_P \ \_ \ ]\!]_P$,
$[\![_E \ \_ \ ]\!]_E$,
$[\![_T \ \_ \ ]\!]_T$

$$
\begin{array}{|l}
[\![_{\boldsymbol{Z}} \ \_ \ ]\!]_{\boldsymbol{Z}} : \mathit{SPECIFICATION} \to \mathit{SectionModels};\\
[\![_{\boldsymbol{S}} \ \_ \ ]\!]_{\boldsymbol{S}} : \mathit{SECTION} \to \mathit{SectionModels} \to \mathit{SectionModels};\\
[\![_{\boldsymbol{D}} \ \_ \ ]\!]_{\boldsymbol{D}} : \mathit{PARAGRAPH} \to \mathit{Model} \leftrightarrow \mathit{Model};\\
[\![_{\boldsymbol{P}} \ \_ \ ]\!]_{\boldsymbol{P}} : \mathit{PREDICATE} \to \mathbb{P}\ \mathit{Model};\\
[\![_{\boldsymbol{E}} \ \_ \ ]\!]_{\boldsymbol{E}} : \mathit{EXPRESSION} \to \mathit{Model} \to \mathbb{W};\\
[\![_{\boldsymbol{T}} \ \_ \ ]\!]_{\boldsymbol{T}} : \mathit{TYPE} \to \mathit{Model} \to \mathbb{P}\mathbb{U}_Z
\end{array}
$$

In principle, in Z, which doesn't allow free variables in axioms, each equation should be individually universally quantified over its free variables. This would clutter our presentation. Instead we declare all the syntactic jokers as global variables. This makes our description logically too weak, but suffices for type-checking purposes.

$$
\begin{array}{|l}
\boldsymbol{m},\ \boldsymbol{n} : \mathbb{N};\\
\boldsymbol{i},\ \boldsymbol{i_1},\ \boldsymbol{i_m},\ \boldsymbol{i_n},\ \boldsymbol{j_1},\ \boldsymbol{j_m},\ \boldsymbol{j_n} : \mathit{NAME};\\
\boldsymbol{\tau},\ \boldsymbol{\tau_1},\ \boldsymbol{\tau_m},\ \boldsymbol{\tau_n} : \mathit{TYPE};\\
\boldsymbol{s_1},\ \boldsymbol{s_n} : \mathit{SECTION};\\
\boldsymbol{d_1},\ \boldsymbol{d_n} : \mathit{PARAGRAPH};\\
\boldsymbol{e},\ \boldsymbol{e_1},\ \boldsymbol{e_2},\ \boldsymbol{e_n} : \mathit{EXPRESSION};\\
\boldsymbol{p},\ \boldsymbol{p_1},\ \boldsymbol{p_2} : \mathit{PREDICATE}
\end{array}
$$

Finally, we need some help with the equations that involve sequences of syntactic constructs. Z does not support the elliptical notation that is used in the original. However, we can pretend that it does. To do this, we will weaken the specification but preserve the surface syntax by taking $n = m = 3$. We then use the notations "...", "⋯" and "⋯", which by a little sleight of hand we have made into ProofPower-Z names, in place of the ellipses of the original.

"⋯" and "⋯" are declared below as a generic value and a generic schema respectively. "..." is used as a bound variable and so needs no declaration here (although, for want of a better name, we use it for the single component of "⋯"). Given these declarations, we can use "⋯" to represent an ellipsis used in a syntactic phrase inside semantic brackets, "⋯" to represent an ellipsis standing for zero or more declarations, and "..." as a local variable for all the other uses of ellipses.

$$\begin{array}{|l}\hline [X] \\ \cdots : X \\ \hline\end{array}$$

$$\begin{array}{|l}\hline \cdots[X] \\ \ldots : X \\ \hline\end{array}$$

4.2 Specification

$$\begin{gathered}[\![_Z\ spec\ \langle s_1,\ \cdots,\ s_n\rangle\ ]\!]_Z \\ = \\ ([\![_S\ section\ prelude\ parents\ \cdots\ end\ \cdots\ END\ ]\!]_S \mathbin{⨟} [\![_S\ s_1\ ]\!]_S \mathbin{⨟} \cdots \mathbin{⨟} [\![_S\ s_n\ ]\!]_S)\ \varnothing\end{gathered}$$

4.3 Section

$$[\![_S\ section\ prelude\ parents\ \ end\ d_1,\ \cdots,\ d_n\ END\ ]\!]_S$$
$$=$$
$$(\lambda T : SectionModels \bullet \{prelude \mapsto ([\![_D\ d_1\ ]\!]_D \mathbin{⨟} \cdots \mathbin{⨟} [\![_D\ d_n\ ]\!]_D)(\!|\ \varnothing\ |\!)\}\ )$$

$$[\![_S\ section\ i\ parents\ i_1,\ \cdots,\ i_m\ end\ d_1,\ \cdots,\ d_n\ END\ ]\!]_S$$
$$=$$
$$\begin{array}{l}(\lambda T : SectionModels \bullet T \cup \{i \mapsto \\ ([\![_D\ d_1\ ]\!]_D \mathbin{⨟} \cdots \mathbin{⨟} [\![_D\ d_n\ ]\!]_D) \\ \quad (\!|\ \{M_0 : T\ prelude;\ M_1 \colon T\ i_1;\ \cdots;\ M_m : T\ i_m;\ M : Model \\ \qquad \mid M = M_0 \cup \cdots \cup M_m \bullet M\} |\!)\})\end{array}$$

4.4 Paragraph

4.4.1 Given types paragraph

$$\begin{array}{l}[\![_D\ [_d\ i_1,\ \cdots,\ i_n\ ]_d\ ]\!]_D = \\ \quad \{M : Model;\ w_1,\ \ldots,\ w_n : \mathbb{W} \bullet \\ \qquad (M,\ M \cup \{i_1 \mapsto w_1,\ \cdots,\ i_n \mapsto w_n\} \cup \\ \qquad\quad \{decor\ \heartsuit\ i_1 \mapsto w_1,\ \cdots,\ decor\ \heartsuit\ i_n \mapsto w_n\})\}\end{array}$$

4.4.2 Axiomatic description paragraph

$$[\![_D\ AX\ \ e\ ]\!]_D = \{M : Model;\ t : \mathbb{W} \mid t \in \eta([\![_E\ e\ ]\!]_E\ M) \bullet (M,\ M \cup (\beta^{\sim})\ t)\}$$

4.4.3 Generic axiomatic description paragraph

$$
\begin{aligned}
&[\![_D\ GENAX\ i_1,\ \cdots,\ i_n\ (_g\ e\ \text{⨾}_g\ \mathbb{P}_t\ [_t\ j_1\ :_t\ \tau_1,\ \cdots,\ j_m\ :_t\ \tau_m\ ]_t)_g\ ]\!]_D = \\
&\quad \{M : Model;\ u : \mathbb{W} \uparrow n \to \mathbb{W} \\
&\qquad \mid \forall\ w_1,\ \ldots,\ w_n : \mathbb{W} \bullet \exists\ w : \mathbb{W} \bullet \\
&\qquad\qquad u\langle w_1,\ \ldots,\ w_n\rangle \in \eta\ w \\
&\qquad\qquad \wedge ((M \oplus \{i_1 \mapsto w_1,\ \cdots,\ i_n \mapsto w_n\} \cup \\
&\qquad\qquad\qquad \{decor\spadesuit i_1 \mapsto w_1,\ \cdots,\ decor\spadesuit i_n \mapsto w_n\}) \mapsto w) \in [\![_E\ e\ ]\!]_E \\
&\quad \bullet (M,\ M \cup (\lambda\ y : \{j_1,\ \cdots,\ j_m\} \bullet \gamma(\lambda x : \mathbb{W} \uparrow n \bullet (\beta^{\sim})(u\ x)\ y)))\}
\end{aligned}
$$

4.4.4 Conjecture paragraph

$$[\![_D \vdash_d p\ ]\!]_D = id\ Model$$

4.4.5 Generic conjecture paragraph

$$[\![_D\ [_\vdash\ i_1,\ \cdots,\ i_n\ ]_\vdash\ p\ ]\!]_D = id\ Model$$

4.5 Predicate

4.5.1 Membership predicate

$$[\![_P\ e_1 \in_p e_2\ ]\!]_P = \{M : Model \mid [\![_E\ e_1\ ]\!]_E\ M \in \eta([\![_E\ e_2\ ]\!]_E\ M) \bullet M\}$$

4.5.2 Truth predicate

$$[\![_P\ true_p\ ]\!]_P = Model$$

4.5.3 Negation predicate

$$[\![_P\ \neg_p\ p\ ]\!]_P = Model \setminus [\![_P\ p\ ]\!]_P$$

4.5.4 Conjunction predicate

$$[\![_P\ p_1 \wedge_p p_2\ ]\!]_P = [\![_P\ p_1\ ]\!]_P \cap [\![_P\ p_2\ ]\!]_P$$

4.5.5 Universal quantification predicate

$$[\![_P\ \forall_p\ e \bullet_p p\ ]\!]_P = \{M : Model \mid \forall\ t : \eta([\![_E\ e\ ]\!]_E\ M) \bullet M \oplus (\beta^{\sim})t \in [\![_P\ p\ ]\!]_P \bullet M\}$$

4.6 Expression

Observation B: As remarked in section 3.2 above, the abstract syntax of expressions permits a type ascription on any form of expression. However, the type ascriptions are simply to be ignored on expressions other than schema negations, schema conjunctions and schema quantifications. The following supplementary semantic equation captures this.

$$\forall e{:}EXPRESSION \setminus (ran\ (\neg_e\ \_) \cup ran\ (\_\ \wedge_e\ \_) \cup ran\ (\forall_s\ \_\ \bullet_s\ \_)) \bullet$$
$$[\![_E\ e\ ⨟_e\ \mathbb{P}_t\ \tau\ ]\!]_E = [\![_E\ e\ ]\!]_E$$

4.6.1 Reference expression

$$[\![_E\ var\ i\ ]\!]_E = (\lambda M : Model \bullet\ M\ i)$$

4.6.2 Generic instantiation expression

$$[\![_E\ geninst\ i\ [_g\ e_1, \cdots, e_n\ ]_g\ ]\!]_E =$$
$$(\lambda M : Model \bullet (\gamma^\sim)(M\ i)\ \langle [\![_E\ e_1]\!]_E\ M, \cdots, [\![_E\ e_n]\!]_E\ M\rangle)$$

4.6.3 Set extension expression

$$[\![_E\ \{_e\ e_1, \cdots, e_n\ \}_e\ ]\!]_E =$$
$$(\lambda M : Model \bullet\ \phi\{[\![_E\ e_1]\!]_E\ M, \cdots, [\![_E\ e_n]\!]_E\ M\})$$

4.6.4 Set comprehension expression

$$[\![_E\ \{_c\ e_1\ \bullet_c\ e_2\ \}_c\ ]\!]_E =$$
$$(\lambda M : Model \mid \forall\ t : \eta([\![_E\ e_1\ ]\!]_E\ M) \bullet\ (M \oplus (\beta^\sim)\ t) \in dom[\![_E\ e_2]\!]_E$$
$$\bullet (\eta^\sim)\{t_1 : \eta([\![_E\ e_1\ ]\!]_E\ M) \bullet\ [\![_E\ e_2]\!]_E\ (M \oplus (\beta^\sim)\ t_1)\})$$

4.6.5 Powerset expression

$$[\![_E\ \mathbb{P}_p\ e\ ]\!]_E = (\lambda M : Model \bullet\ \mathbb{P}_W([\![_E\ e]\!]_E\ M))$$

4.6.6 Tuple extension expression

$$[\![_E\ (_t\ e_1, \cdots, e_n\ )_t\ ]\!]_E =$$
$$(\lambda M : Model \bullet\ \chi\langle [\![_E\ e_1]\!]_E\ M, \cdots, [\![_E\ e_n]\!]_E\ M\rangle)$$

4.6.7 Binding extension expression

$$[\![_E\ (_b\ i_1 ==_b\ e_1, \cdots, i_n ==_b\ e_n\ )_b\ ]\!]_E =$$
$$(\lambda M : Model \bullet\ \beta\{i_1 \mapsto [\![_E\ e_1]\!]_E\ M, \cdots, i_n \mapsto [\![_E\ e_n]\!]_E\ M\})$$

4.6.8 Definite description expression

$$\begin{array}{l}\{M : Model;\ t_1 : \mathbb{W} \\ \qquad \mid t_1 \in \eta(\llbracket_E\ e_1\ \rrbracket_E\ M) \\ \qquad \wedge\ (\forall t_3 : \eta(\llbracket_E\ e_1\ \rrbracket_E\ M) \\ \qquad\qquad \bullet \llbracket_E\ e_2\ \rrbracket_E\ (M \oplus (\beta^{\sim})\ t_3) = \llbracket_E\ e_2\ \rrbracket_E\ (M \oplus (\beta^{\sim})\ t_1)) \\ \qquad \bullet (M,\ \llbracket_E\ e_2\ \rrbracket_E\ (M \oplus (\beta^{\sim})\ t_1))\} \qquad \subseteq \llbracket_E\ \mu_d\ e_1 \bullet_d\ e_2\ \rrbracket_E\end{array}$$

4.6.9 Variable construction expression

$$\llbracket_E\ [_v\ i\ ⨾_d\ e\ ]_v\ \rrbracket_E = (\lambda M : Model \bullet (\eta^{\sim})\{w : \eta(\llbracket_E\ e\ \rrbracket_E\ M) \bullet\ \beta\{i \mapsto w\}\})$$

4.6.10 Schema construction expression

$$\begin{array}{l}\llbracket_E\ [_s\ e\ |_s\ p\ ]_s\ \rrbracket_E = (\lambda M : Model \bullet \\ \qquad (\eta^{\sim})\{t : \eta(\llbracket_E\ e\ \rrbracket_E\ M) \mid M \oplus (\beta^{\sim})\ t \in \llbracket_P\ p\ \rrbracket_P \bullet\ t\})\end{array}$$

4.6.11 Schema negation expression

$$\begin{array}{l}\llbracket_E\ (\neg_e\ e)\ ⨾_e\ \mathbb{P}_t\ \tau\ \rrbracket_E = (\lambda M : Model \bullet \\ \qquad (\eta^{\sim})\{t : \llbracket_T\ \tau\ \rrbracket_T\ M \mid \neg t \in \eta(\llbracket_E\ e\ \rrbracket_E\ M) \bullet\ t\})\end{array}$$

4.6.12 Schema conjunction expression

$$\begin{array}{l}\llbracket_E\ (e_1 \wedge_e\ e_2)\ ⨾_e\ \mathbb{P}_t\ \tau\ \rrbracket_E = \\ \qquad (\lambda M : Model \\ \qquad\qquad \bullet (\eta^{\sim})(\{t : \llbracket_T\ \tau\ \rrbracket_T\ M;\ t_1 : \eta(\llbracket_E\ e_1 \rrbracket_E\ M);\ t_2 : \eta(\llbracket_E\ e_2 \rrbracket_E\ M) \\ \qquad\qquad\qquad \mid \eta\ t_1 \cup \eta\ t_2 = \eta\ t \bullet\ t\}))\end{array}$$

4.6.13 Schema universal quantification expression

$$\begin{array}{l}\llbracket_E\ (\forall_s\ e_1 \bullet_s\ e_2)\ ⨾_e\ \mathbb{P}_t\ \tau\ \rrbracket_E = \\ \qquad (\lambda M : Model \\ \qquad\qquad \bullet (\eta^{\sim})\{t_2 : \llbracket_T\ \tau\ \rrbracket_T\ M \mid \\ \qquad\qquad\ \forall\ t_1 : \eta(\llbracket_E\ e_1 \rrbracket_E\ M) \bullet \\ \qquad\qquad\qquad \beta((\beta^{\sim})t_1 \cup (\beta^{\sim})t_2) \in \eta(\llbracket_E\ e_2 \rrbracket_E\ (M \oplus (\beta^{\sim})t_1)) \bullet\ t_2\})\end{array}$$

Observation C: It has been objected that the above requires the signature of e_1 to be contained in that of e_2 not just compatible with it. Spivey made this extra restriction but the syntactic rules in the Z Standard do not. The following variant appears to give the desired semantics.

$$\begin{array}{l}\llbracket_E\ (\forall_s\ e_1 \bullet_s\ e_2\ ⨾_e\ \mathbb{P}_t([_t\ i_1 :_t \tau_1, \cdots, i_n :_t \tau_n\ ]_t))\ ⨾_e\ \mathbb{P}_t\ \tau\ \rrbracket_E = \\ \qquad (\lambda M : Model \\ \qquad\qquad \bullet (\eta^{\sim})\{t : \llbracket_T\ \tau\ \rrbracket_T\ M \mid \\ \qquad\qquad\ \forall\ t_1 : \eta(\llbracket_E\ e_1 \rrbracket_E\ M) \bullet \\ \qquad\qquad\qquad \beta(\{i_1, \cdots, i_n\} \lhd (\beta^{\sim})t_1 \cup (\beta^{\sim})t) \in \eta(\llbracket_E\ e_2 \rrbracket_E\ (M \oplus (\beta^{\sim})t_1)) \bullet\ t\})\end{array}$$

4.6.14 Schema renaming expression

$$
\begin{array}{l}
[\![_E\ e\ [_r\ j_1\ /_r\ i_1,\ \cdots,\ j_n\ /_r\ i_n\ ]_r\ ]\!]_E = \\
\quad (\lambda M : Model \\
\qquad \bullet (\eta^\sim)\{t_1 : \eta([\![_E\ e\ ]\!]_E\ M);\ t_2 : \mathbb{W}\ | \\
\qquad\quad (\beta^\sim)t_2 = \{j_1 \mapsto i_1,\ \cdots,\ j_n \mapsto i_n\} \mathbin{\text{⨾}} (\beta^\sim)t_1 \cup \{i_1,\ \cdots,\ i_n\} \lhd (\beta^\sim)t_1 \\
\qquad\quad \wedge\ (\beta^\sim)\ t_2 \in (\_ \mathrel{\ooalign{$\rightarrow$\cr\hidewidth$\shortmid$\hidewidth}}\ \_) \\
\qquad\quad \bullet\ t_2\})
\end{array}
$$

4.7 Type

4.7.1 Given type

$$[\![_T\ given\ i\ ]\!]_T = (\lambda M : Model \bullet\ \eta(M\ i))$$

4.7.2 Generic parameter type

$$[\![_T\ generic\ i\ ]\!]_T = (\lambda M : Model \bullet\ \eta(M\ (decor\ \spadesuit\ i)))$$

4.7.3 Set type

$$[\![_T\ \mathbb{P}_t\ \tau\ ]\!]_T = (\lambda M : Model \bullet\ (\eta^\sim)(\!|\mathbb{P}([\![_T\ \tau\ ]\!]_T\ M)|\!))$$

4.7.4 Cartesian product type

There seems to be no particularly good way of preserving the syntax used in the original for the cartesian product operator in $\mathbb{W}$. Consequently we have changed the right-hand side of the next equation to use a description of the cartesian product along the lines of that used below for schema types.

$$
\begin{array}{l}
[\![_T\ (_\times\ \tau_1,\ \cdots,\ \tau_n\ )_\times\ ]\!]_T = (\lambda M : Model \bullet \\
\quad \{\ f\!: \{1,\ \cdots,\ n\} \rightarrow \mathbb{W} \\
\qquad |\ f\ 1 \in ([\![_T\ \tau_1\ ]\!]_T\ M) \wedge \cdots \wedge f\ n \in ([\![_T\ \tau_n\ ]\!]_T\ M) \bullet \chi\ f\})
\end{array}
$$

4.7.5 Schema type

$$
\begin{array}{l}
[\![_T\ [_t\ i_1\ :_t\ \tau_1,\ \cdots,\ i_n\ :_t\ \tau_n\ ]_t\ ]\!]_T = (\lambda M : Model \bullet \\
\quad \{\ t\!: \{i_1,\ \cdots,\ i_n\} \rightarrow \mathbb{W} \\
\qquad |\ t\ i_1 \in ([\![_T\ \tau_1\ ]\!]_T\ M) \wedge \cdots \wedge t\ i_n \in ([\![_T\ \tau_n\ ]\!]_T\ M) \bullet \beta\ t\})
\end{array}
$$

5 TYPES INFERRED

The following table shows the types inferred by ProofPower for the global variables of the specification:

AX	$EXPRESSION \leftrightarrow PARAGRAPH$
$BIND$	$\mathbb{P}\ BIND$
DEC	$\mathbb{P}\ DEC$
$DECL$	$\mathbb{P}\ DECL$
$decor$	$DECORATOR \leftrightarrow \mathbb{U} \leftrightarrow \mathbb{U}$
$DECORATOR$	$\mathbb{P}\ DECORATOR$
dn	$PARAGRAPH$
$d1$	$PARAGRAPH$
e	$EXPRESSION$
$EXPRESSION$	$\mathbb{P}\ EXPRESSION$
en	$EXPRESSION$
$e1$	$EXPRESSION$
$e2$	$EXPRESSION$
$generic$	$\mathbb{U} \leftrightarrow TYPE$
$given$	$\mathbb{U} \leftrightarrow TYPE$
i	$\mathbb{U}$
im	$\mathbb{U}$
in	$\mathbb{U}$
$i1$	$\mathbb{U}$
jm	$\mathbb{U}$
jn	$\mathbb{U}$
$j1$	$\mathbb{U}$
m	$\mathbb{Z}$
$Model$	$\mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U})$
n	$\mathbb{Z}$
$NAME$	$\mathbb{P}\ \mathbb{U}$
p	$PREDICATE$
$PARAGRAPH$	$\mathbb{P}\ PARAGRAPH$
$PREDICATE$	$\mathbb{P}\ PREDICATE$
$prelude$	$\mathbb{U}$
$p1$	$PREDICATE$
$p2$	$PREDICATE$
$RENAME$	$\mathbb{P}\ RENAME$
$SECTION$	$\mathbb{P}\ SECTION$
$spec$	$(\mathbb{Z} \leftrightarrow SECTION) \leftrightarrow SPECIFICATION$
$SPECIFICATION$	$\mathbb{P}\ SPECIFICATION$
$STROKE$	$\mathbb{P}\ STROKE$
$stroke$	$STROKE \leftrightarrow DECORATOR$
sn	$SECTION$
$s1$	$SECTION$
$Theory$	$\mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U}))$
$truep$	$PREDICATE$

$TYPE$	$\mathbb{P}\ TYPE$
var	$\mathbb{U} \leftrightarrow EXPRESSION$
$\mathbb{U}$	$\mathbb{P}\ \mathbb{U}$
$\mathbb{W}$	$\mathbb{P}\ \mathbb{U}$
$\heartsuit$	
	$DECORATOR$
$\spadesuit$	
	$DECORATOR$
$(GENAX\ ...\ (g\ \_\ {}_{9}^{o}g\ \_\ )g)$	
	$(\mathbb{Z} \leftrightarrow \mathbb{U}) \times EXPRESSION \times TYPE \leftrightarrow PARAGRAPH$
$(geninst\ \_\ [g\ ...\ ]g)$	
	$\mathbb{U} \times (\mathbb{Z} \leftrightarrow EXPRESSION) \leftrightarrow EXPRESSION$
$(section\ \_\ parents\ ...\ end\ ...\ END)$	
	$\mathbb{U} \times (\mathbb{Z} \leftrightarrow \mathbb{U}) \times (\mathbb{Z} \leftrightarrow PARAGRAPH) \leftrightarrow SECTION$
$((b\ ...\ )b)$	
	$(\mathbb{Z} \leftrightarrow BIND) \leftrightarrow EXPRESSION$
$((t\ ...\ )t)$	
	$(\mathbb{Z} \leftrightarrow EXPRESSION) \leftrightarrow EXPRESSION$
$((\times\ ...\ )\times)$	
	$(\mathbb{Z} \leftrightarrow TYPE) \leftrightarrow TYPE$
$([d\ ...\ ]d)$	
	$(\mathbb{Z} \leftrightarrow \mathbb{U}) \leftrightarrow PARAGRAPH$
$([s\ \_\ \|s\ \_\ ]s)$	
	$EXPRESSION \times PREDICATE \leftrightarrow EXPRESSION$
$([t\ ...\ ]t)$	
	$(\mathbb{Z} \leftrightarrow DECL) \leftrightarrow TYPE$
$([v\ \_\ ]v)$	
	$DEC \leftrightarrow EXPRESSION$
$([\vdash\ ...\ ]\vdash\ \_)$	
	$(\mathbb{Z} \leftrightarrow \mathbb{U}) \times PREDICATE \leftrightarrow PARAGRAPH$
$(\_\ \uparrow\ \_)[X]$	
	$\mathbb{P}\ X \times \mathbb{Z} \leftrightarrow \mathbb{P}\ (\mathbb{Z} \leftrightarrow X)$
$(\_\ /r\ \_)$	$\mathbb{U} \times \mathbb{U} \leftrightarrow RENAME$
$(\_\ :t\ \_)$	$\mathbb{U} \times TYPE \leftrightarrow DECL$
$(\_\ ==b\ \_)$	$\mathbb{U} \times EXPRESSION \leftrightarrow BIND$
$(\_\ [r\ ...\ ]r)$	
	$EXPRESSION \times (\mathbb{Z} \leftrightarrow RENAME) \leftrightarrow EXPRESSION$
$(\_\ \in p\ \_)$	$EXPRESSION \times EXPRESSION \leftrightarrow PREDICATE$
$(\_\ \wedge e\ \_\ {}_{9}^{o}c\ \_)$	
	$EXPRESSION \times EXPRESSION \times TYPE \leftrightarrow EXPRESSION$
$(\_\ \wedge p\ \_)$	$PREDICATE \times PREDICATE \leftrightarrow PREDICATE$
$(\_\ {}_{9}^{o}d\ \_)$	$\mathbb{U} \times EXPRESSION \leftrightarrow DEC$
$(\{c\ \_\ \bullet c\ \_\ \}c)$	
	$EXPRESSION \times EXPRESSION \leftrightarrow EXPRESSION$
$(\{e\ ...\ \}e)$	
	$(\mathbb{Z} \leftrightarrow EXPRESSION) \leftrightarrow EXPRESSION$
$(\neg e\ \_\ {}_{9}^{o}e\ \_)$	
	$EXPRESSION \times TYPE \leftrightarrow EXPRESSION$

$(\forall p\ \_\ \bullet p\ \_)$	
	$\mathit{EXPRESSION} \times \mathit{PREDICATE} \leftrightarrow \mathit{PREDICATE}$
$(\forall s\ \_\ \bullet s\ \_\ \fatsemi s\ \_)$	
	$\mathit{EXPRESSION} \times \mathit{EXPRESSION} \times \mathit{TYPE} \leftrightarrow \mathit{EXPRESSION}$
$(\lambda t\ \ldots\ \bullet t\ \_)$	
	$(\mathbb{Z} \leftrightarrow \mathbb{U}) \times \mathit{TYPE} \leftrightarrow \mathit{TYPE}$
$(\mu d\ \_\ \bullet d\ \_)$	
	$\mathit{EXPRESSION} \times \mathit{EXPRESSION} \leftrightarrow \mathit{EXPRESSION}$
$([\![D\ \_\ ]\!]D)$	
	$\mathit{PARAGRAPH} \leftrightarrow (\mathbb{U} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U} \leftrightarrow \mathbb{U}$
$([\![E\ \_\ ]\!]E)$	
	$\mathit{EXPRESSION} \leftrightarrow (\mathbb{U} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U}$
$([\![P\ \_\ ]\!]P)$	
	$\mathit{PREDICATE} \leftrightarrow \mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U})$
$([\![S\ \_\ ]\!]S)$	
	$\mathit{SECTION}$ $\leftrightarrow (\mathbb{U} \leftrightarrow \mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U}))$ $\leftrightarrow \mathbb{U} \leftrightarrow \mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U})$
$([\![T\ \_\ ]\!]T)$	
	$\mathit{TYPE} \leftrightarrow (\mathbb{U} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{P}\ \mathbb{U}$
$([\![Z\ \_\ ]\!]Z)$	
	$\mathit{SPECIFICATION} \leftrightarrow \mathbb{U} \leftrightarrow \mathbb{P}\ (\mathbb{U} \leftrightarrow \mathbb{U})$
$\cdots[X]$	X
$\cdots[X]$	$\mathbb{P}\ [\ldots : X]$
$\neg p$	$\mathit{PREDICATE} \leftrightarrow \mathit{PREDICATE}$
β	$(\mathbb{U} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U}$
ϕ	$\mathbb{P}\ \mathbb{U} \leftrightarrow \mathbb{U}$
γ	$((\mathbb{Z} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U}$
η	$\mathbb{U} \leftrightarrow \mathbb{P}\ \mathbb{U}$
χ	$(\mathbb{Z} \leftrightarrow \mathbb{U}) \leftrightarrow \mathbb{U}$
τ	TYPE
τm	TYPE
τn	TYPE
$\tau 1$	TYPE
$\mathbb{P}p$	$\mathit{EXPRESSION} \leftrightarrow \mathit{EXPRESSION}$
$\mathbb{P}t$	$\mathit{TYPE} \leftrightarrow \mathit{TYPE}$
$\mathbb{P}W$	$\mathbb{U} \leftrightarrow \mathbb{U}$
$\vdash d$	$\mathit{PREDICATE} \leftrightarrow \mathit{PARAGRAPH}$

6 INDEX

AX 6
β 4
$BIND$ 6
$\bullet_c$ 6
$\bullet_d$ 6
$\bullet_p$ 6
$\bullet_s$ 6
$\bullet_t$ 5
χ 4
DEC 6
$DECL$ 5
$decor$ 7
$DECORATOR$ 7
$\{_c$ 6
$\{_e$ 6
$\}_c$ 6
$\}_e$ 6
d_1 8
d_n 8
e 8
end 7
END 7
e_1 8
e_2 8
e_n 8
η 3
$EXPRESSION$ 6
$\forall_p$ 6
$\forall_s$ 6
γ 4
$GENAX$ 6
$generic$ 5
$geninst$ 6
$given$ 5
i 8
$\in_p$ 6
i_1 8
i_m 8
i_n 8
j_1 8
j_m 8
j_n 8
$\_ \uparrow \_$ 3
λ_t 5
$\wedge_e$ 6
$\wedge_p$ 6
$\neg_e$ 6
$\neg_p$ 6
m 8
$\fatsemi_d$ 6
$\fatsemi_e$ 6
$\fatsemi_g$ 6
$Model$ 7
$[\![_D$ 8
$]\!]_D$ 8
$[\![_E$ 8
$]\!]_E$ 8
$[\![_P$ 8
$]\!]_P$ 8
$[\![_S$ 8
$]\!]_S$ 8
$[\![_T$ 8
$]\!]_T$ 8
$[\![_Z$ 8
$]\!]_Z$ 8
μ_d 6
n 8
$NAME$ 3
p 8
$PARAGRAPH$ 6
$parents$ 7
ϕ 4
$PREDICATE$ 6
$prelude$ 7
$\cdots$ 8
$\cdots$ 8
$\mathbb{P}_p$ 6
$\mathbb{P}_t$ 5
$\mathbb{P}_W$ 3
$\mathbb{U}_Z$ 3
$\mathbb{W}$ 3
p_1 8
p_2 8
$RENAME$ 6
$==_b$ 6
$(_b$ 6
$)_b$ 6
$[_d$ 6
$]_d$ 6
$(_g$ 6

$)_g$ 6
$[_g$ 6
$]_g$ 6
$/_r$ 6
$[_r$ 6
$]_r$ 6
$|_s$ 6
$[_s$ 6
$]_s$ 6
$:_t$ 5
$[_t$ 5
$]_t$ 5
$(_t$ 6
$)_t$ 6
$(_\times$ 5
$)_\times$ 5
$[_v$ 6
$]_v$ 6
$[_\vdash$ 6
$]_\vdash$ 6
$section$ 7
$SECTION$ 7
$SectionModels$ 7
$spec$ 7
$SPECIFICATION$ 7
s_1 8
s_n 8
$stroke$ 7
$STROKE$ 7
τ 8
τ_1 8
τ_m 8
τ_n 8
$true_p$ 6
$TYPE$ 5
var 6
$\vdash_d$ 6