Formal Specification and Verification of Software

The Z Specification Language

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

The Z Specification Language

Based on

- **Typed first-order predicate logic**
- **Zermelo-Fraenkel set theory**
- **Rich notation**

The Z Specification Language

Based on

- Typed first-order predicate logic
- Zermelo-Fraenkel set theory
- Rich notation

Invented/developed by

J.-R. Abrial, Oxford University Computing Laboratory

International standard

ISO/IEC JTC1/SC22

The Z Specification Language

Tools

- LaTeX style
- Type checker
- Z/Eves deduction system

But

No tools for simulation/execution/testing

Built-in Operators

Logical operators

- $\neg$ **negation**
- $\wedge$ **conjunction**
- $\vee$ **disjunction**
- $\Rightarrow$ **implication** **(note: not $\rightarrow$)**
- $\Leftrightarrow$ **equivalence** **(note: not $\leftrightarrow$)**

Equality

- $=$ **equality**

On all types (but not predicates)

Built-in Operators

Quantification

$$\mathcal{Q}\ x_1 : S_1;\ \ldots;\ x_n : S_n \mid p \bullet q$$

where $\mathcal{Q}$ is one of $\forall \quad \exists \quad \exists_1$

Meaning

$$\forall x_1 : S_1;\ \ldots;\ x_n : S_n (p \Rightarrow q) \qquad \textbf{resp.}$$
$$\exists x_1 : S_1;\ \ldots;\ x_n : S_n (p \wedge q)$$

Abbreviation

$\forall x : T \bullet q$ **for** $\forall x : T \mid true \bullet q$

Notation for Sets

Enumeration

$$\{e_1, \ldots, e_n\}$$

The set of type-compatible elements $e_1, \ldots, e_n$

Example

$$\{3, 5, 8, 4\}$$

Notation for Sets

Set comprehension

$$\{x : T \mid pred(x) \bullet expr(x)\}$$

The set of all elements that result from evaluating $expr(x)$ for all x of type T for which $pred(x)$ holds

Example

$$\{x : \mathbb{Z} \mid prime(x) \bullet x * x\}$$

The set of all squares of prime numbers

Notation for Sets

Abbreviation

$\{x : T \mid pred(x)\}$ **for** $\{x : T \mid pred(x) \bullet x\}$

Example

$$\mathbb{N} = \{x : \mathbb{Z} \mid x \geq 0\}$$

The empty set

$$\varnothing = \{x : T \mid \mathbf{false}\}$$

Note:

$\varnothing = \varnothing[T]$ **is typed**

Set Operations

$\in$ **element-of relation**

$\subseteq$ **subset relation**

S_1 **and** S_2 **must have the same type**

$$S_1 \subseteq S_2 \Leftrightarrow (\forall x : S_1 \bullet x \in S_2)$$

$\mathbb{P}$ **power set operator**

$$S' \in \mathbb{P}\, S \Leftrightarrow S' \subseteq S$$

$\times$ **cartesian product**

$$(x_1, \ldots, x_n) \in S_1 \times \ldots \times S_n \Leftrightarrow (x_1 \in S_1 \wedge \ldots \wedge x_n \in S_n)$$

Set Operations

$\cup, \bigcup$ union

Involved sets must have the same type T

$$x \in S_1 \cup S_2 \Leftrightarrow (x \in S_1 \vee x \in S_2)$$

$$x \in \bigcup S \Leftrightarrow (\exists s' : T \bullet s' \in S \wedge x \in s')$$

$\cap, \bigcap$ intersection

$\setminus$ set difference

Types

Pre-defined types

$\mathbb{Z}$ **with constants:** $0, 1, 2, 3, 4, \ldots$

functions: $+, -, *, /$

predicates: $<, \leq, >, \geq$

Sets

Every set can be used as a type

Basic types (given sets)

Example

$[Person]$

Free Type Definitions

Example

$$weekDay ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$$

Example

$$Tree ::= leaf\langle\langle \mathbb{Z} \rangle\rangle \mid node\langle\langle Tree \times Tree \rangle\rangle$$

Meaning

$[Tree]$ **generated by** $leaf, node$

$$\begin{array}{l} \forall\, x_1, y_1, x_2, y_2 : Tree \mid \\ \quad node(x_1, y_1) = node(x_2, y_2) \bullet (x_1 = x_2 \land y_1 = y_2) \\ \forall\, x_1, x_2 : \mathbb{Z} \mid leaf(x_1) = leaf(x_2) \bullet x_1 = x_2 \\ \forall\, x : \mathbb{Z};\ y, z : Tree \bullet leaf(x) \neq node(y, z) \end{array}$$

Note: Generatedness is not expressible in first-order logic

Compound Types

Set type: $\mathbb{P}\,T$

The type of sets of elements of type T

Cartesian product type: $T_1 \times \cdots \times T_n$

The type of tuples $(t_1, \ldots, t_n)$ **with** $t_i \in T_i$

Types: Overview

Possible type definitions

- $T = \mathbb{Z}$
- $T = [Type]$
- $T ::= \ldots$ **(free type)**
- $T = \mathbb{P}\, T'$
- $T = T_1 \times \cdots \times T_n$

Types: Overview

Possible type definitions

- $T = \mathbb{Z}$
- $T = [Type]$
- $T ::= \ldots$ **(free type)**
- $T = \mathbb{P}\, T'$
- $T = T_1 \times \cdots \times T_n$

Note

All types are disjoint (not for sets that are used as types)

All terms have a unique type

Variables

Variable declarations

Example

$$x : \mathbb{Z}$$
$$sold : \mathbb{P}\, Seat$$

Variables can range over types and over sets

Syntactical Abbreviations

Abbreviations

- **must not be recursive**
- **can be generic**

Examples

$$numberPairs == \mathbb{Z} \times \mathbb{Z}$$

$$pairWithNumber[S] == \mathbb{Z} \times S$$

Note

Type variables are “meta-variables” (cannot be quantified)

Abbreviations vs. Generated Types

$$weekDay1 == \{mon, tue, wed, thu, fri, sat, sun\}$$

vs.

$$WeekDay2 ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$$

Abbreviations vs. Generated Types

$$weekDay1 == \{mon, tue, wed, thu, fri, sat, sun\}$$

vs.

$$WeekDay2 ::= mon \mid tue \mid wed \mid thu \mid fri \mid sat \mid sun$$

Not the same

Type definition implies elements to be different

Axiomatic Definitions

Form of an axiomatic definition

$$\begin{array}{|l} SymbolDeclarations \\ \hline ConstrainingPredicates \end{array}$$

Example

$$\begin{array}{|l} \mathbb{N}_1 : \mathbb{P}\,\mathbb{Z} \\ \hline \forall z : \mathbb{Z} \bullet (z \in \mathbb{N}_1 \leftrightarrow z \geq 1) \end{array}$$

Relations

Relation types/sets

$S \leftrightarrow T$ **is the type/set of relations between types/sets** S **and** T

$$S \leftrightarrow T = \mathbb{P}(S \times T)$$

Notation

$a \mapsto b$ **for** (a, b) **if** $(a, b) \in S \times T$

Operations on Relations

Domain $\mathbf{dom}\, R$

$$\mathbf{dom}\, R = \{a : S, b : T \mid a \mapsto b \in R \bullet a\}$$

Range $\mathbf{ran}\, R$

$$\mathbf{ran}\, R = \{a : S;\ b : T \mid a \mapsto b \in R \bullet b\}$$

Operations on Relations

Domain $\mathbf{dom}\, R$

$$\mathbf{dom}\, R = \{a : S, b : T \mid a \mapsto b \in R \bullet a\}$$

Range $\mathbf{ran}\, R$

$$\mathbf{ran}\, R = \{a : S;\ b : T \mid a \mapsto b \in R \bullet b\}$$

Restrictions of relations

$$S' \lhd R = \{a : S;\ b : T \mid a \mapsto b \in R \land a \in S' \bullet a \mapsto b\}$$

$$R \rhd T' = \{a : S;\ b : T \mid a \mapsto b \in R \land b \in T' \bullet a \mapsto b\}$$

$$S' \lhd\!\!\!- R = \{a : S;\ b : T \mid a \mapsto b \in R \land a \notin S' \bullet a \mapsto b\}$$

$$R \rhd\!\!\!- T' = \{a : S;\ b : T \mid a \mapsto b \in R \land b \notin T' \bullet a \mapsto b\}$$

Operations on Relations

Inverse relation R^{-1}

$$R^{-1} = \{a : S;\ b : T \mid a \mapsto b \in R \bullet b \mapsto a\}$$

Operations on Relations

Inverse relation R^{-1}

$$R^{-1} = \{a : S;\ b : T \mid a \mapsto b \in R \bullet b \mapsto a\}$$

Composition $R \fatsemi R'$ $\qquad R : S \leftrightarrow T$ **and** $R' : T \leftrightarrow U$

$$R \fatsemi R' = \{a : S;\ b : T;\ c : U \mid a \mapsto b \in R \land b \mapsto c \in R' \bullet a \mapsto c\}$$

Operations on Relations

Inverse relation R^{-1}

$$R^{-1} = \{a : S;\ b : T \mid a \mapsto b \in R \bullet b \mapsto a\}$$

Composition $R \fatsemi R'$ $\quad R : S \leftrightarrow T$ **and** $R' : T \leftrightarrow U$

$$R \fatsemi R' = \{a : S;\ b : T;\ c : U \mid a \mapsto b \in R \land b \mapsto c \in R' \bullet a \mapsto c\}$$

Closures $\quad R : S \leftrightarrow S$

iteration	$R^n = R \fatsemi R^{n-1}$
identity	$R^0 = \{a : S \mid \textbf{true} \bullet a \mapsto a\}$
refl./trans.	$R^* = \bigcup \{n : \mathbb{N} \mid \textbf{true} \bullet R^n\}$
transitive	$R^+ = \bigcup \{n : \mathbb{N} \mid n \geq 1 \bullet R^n\}$
symetric	$R^s = R \cup R^{-1}$
reflexive	$R^r = R \cup R^0$

Functions

Special relations

Functions are special relations

Notation

Instead of $\leftrightarrow$

- $\rightarrow$ **total function**
- $\nrightarrow$ **partial function**

Functions

Partial functions

$$f \in S \nrightarrow T \;\; \Leftrightarrow$$
$$f \in S \leftrightarrow T \;\; \wedge$$
$$\forall a : S, b : T, b' : T \mid (a \mapsto b \in f \wedge a \mapsto b' \in f) \bullet b = b'$$

Functions

Partial functions

$$f \in S \pfun T \;\Leftrightarrow$$
$$f \in S \rel T \;\wedge$$
$$\forall a : S, b : T, b' : T \mid (a \mapsto b \in f \wedge a \mapsto b' \in f) \bullet b = b'$$

Total functions

$$f \in S \fun T \;\Leftrightarrow$$
$$f \in S \pfun T \;\wedge$$
$$\forall a : S \bullet \exists b : T \bullet a \mapsto b \in f$$

λ Notation for Functions

General form

$$\lambda a : S \mid p \bullet e$$

Example

$$\begin{array}{|l} double : \mathbb{Z} \pfun \mathbb{Z} \\ \hline double = \lambda n : \mathbb{Z} \mid n \geq 0 \bullet n + n \end{array}$$

Equivalent to

$$\begin{array}{|l} double : \mathbb{Z} \pfun \mathbb{Z} \\ \hline double = \{n : \mathbb{N} \mid \mathbf{true} \bullet n \mapsto n + n\} \end{array}$$

Prefix and Infix Notation

Notation

Relations and functions can be declared prefix and infix

Parameter positions are indicated with "_"

Example

$$\begin{array}{|l} even\ \_ : \mathbb{Z} \rightarrow \mathbb{B} \\ \hline \forall x : \mathbb{Z} \bullet (even\ x \Leftrightarrow (\exists y : \mathbb{Z} \bullet x = y + y)) \end{array}$$

Equivalent to

$$\begin{array}{|l} even\ \_ : \mathbb{Z} \rightarrow \mathbb{B} \\ \hline even = \{x : \mathbb{Z} \mid (\exists y : \mathbb{Z} \bullet x = y + y)\} \end{array}$$

More Notation for Functions

Notation

- $\rightarrowtail\!\!\!\!\!\!\!\!+\;\;$ **partial injective function**
- $\rightarrowtail$ **total injective function**
- $\twoheadrightarrow\!\!\!\!\!\!\!\!+\;\;$ **partial surjective function**
- $\twoheadrightarrow$ **total surjective function**
- $\rightarrowtail\!\!\!\!\!\!\!\!\twoheadrightarrow$ **total bijective function**

Three Definitions of abs

Relation (in infix notation)

$$\_\ abs\ \_ : \mathbb{Z} \leftrightarrow \mathbb{N}$$

$$\forall m : \mathbb{Z}, n : \mathbb{N} \bullet (m\ abs\ n) \leftrightarrow$$
$$((m = n \land m \geq 0) \lor (-m = n \land m \leq 0))$$

Three Definitions of abs

Relation (in infix notation)

$$\_\ abs\ \_ : \mathbb{Z} \leftrightarrow \mathbb{N}$$

$$\forall m : \mathbb{Z}, n : \mathbb{N} \bullet (m\ abs\ n) \leftrightarrow ((m = n \land m \geq 0) \lor (-m = n \land m \leq 0))$$

Function

$$abs : \mathbb{Z} \rightarrow \mathbb{N}$$

$$abs = (\lambda m : \mathbb{Z} \mid m \leq 0 \bullet -m) \cup (\lambda m : \mathbb{Z} \mid m \geq 0 \bullet m)$$

Three Definitions of abs

Relation (in infix notation)

$$\_\ abs\ \_ : \mathbb{Z} \leftrightarrow \mathbb{N}$$

$$\forall m : \mathbb{Z}, n : \mathbb{N} \bullet (m\ abs\ n) \leftrightarrow ((m = n \land m \geq 0) \lor (-m = n \land m \leq 0))$$

Function

$$abs : \mathbb{Z} \rightarrow \mathbb{N}$$

$$abs = (\lambda m : \mathbb{Z} \mid m \leq 0 \bullet -m) \cup (\lambda m : \mathbb{Z} \mid m \geq 0 \bullet m)$$

Function (in prefix notation)

$$abs\ \_ : \mathbb{Z} \pfun \mathbb{N}$$

$$\forall x : \mathbb{Z} \mid x \leq 0 \bullet x = -(abs\ x)$$
$$\forall x : \mathbb{Z} \mid x \geq 0 \bullet x = abs\ x$$

Finite Constructs

Finite subsets of $\mathbb{Z}$

$$m..n = \{n' : \mathbb{N} \mid m \leq n' \land n' \leq n\}$$

Finite Constructs

Finite subsets of $\mathbb{Z}$

$$m..n = \{n' : \mathbb{N} \mid m \leq n' \land n' \leq n\}$$

Finite sets

$\mathbb{F}\, T$ **consists of the finite sets in** $\mathbb{P}\, T$

$$[S]$$

$$\mathbb{F} : \mathbb{P}(\mathbb{P}\, S)$$

$$\mathbb{F} = \{s : \mathbb{P}\, S \mid (\exists n : \mathbb{N} \bullet (\exists f : 1..n \rightarrowtail\!\!\!\!\!\rightarrow s \bullet \mathbf{true}))\}$$

Finite Sets: Cardinality

Cardinality operator

$$[S]$$

$$\#: \mathbb{F}\, S \to \mathbb{N}$$

$$\forall s : \mathbb{F}\, S;\ n : \mathbb{N} \bullet (n = \#s \leftrightarrow (\exists f : 1..n \rightarrowtail\!\!\!\!\!\rightarrow s \bullet \mathbf{true}))$$

Finite Functions

Notation

$\pfun$ **finite (partial) functions** (e.g. arrays)

$$S \pfun T = \{f : S \pfun T \mid \mathbf{dom} f \in \mathbb{F}\, S\}$$

$\pinj$ **finite (partial) injective functions** (e.g. duplicate-free arrays)

$$S \pinj T = \{f : S \pinj T \mid \mathbf{dom} f \in \mathbb{F}\, S\}$$

Sequences

Definition

$$\mathbf{seq}\, T \;==\; \{s : \mathbb{Z} \pfun T \mid \mathbf{dom}\, s = 1..\#s\}$$

Note

- **sequences are functions, which are relations, which are sets**
- **the length of s is $\#s$**

Sequences

Definition

$$\mathbf{seq}\, T \;==\; \{s : \mathbb{Z} \pfun T \mid \mathbf{dom}\, s = 1..\#s\}$$

Note

- **sequences are functions, which are relations, which are sets**
- **the length of s is $\#s$**

Notation

The sequence $\{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$

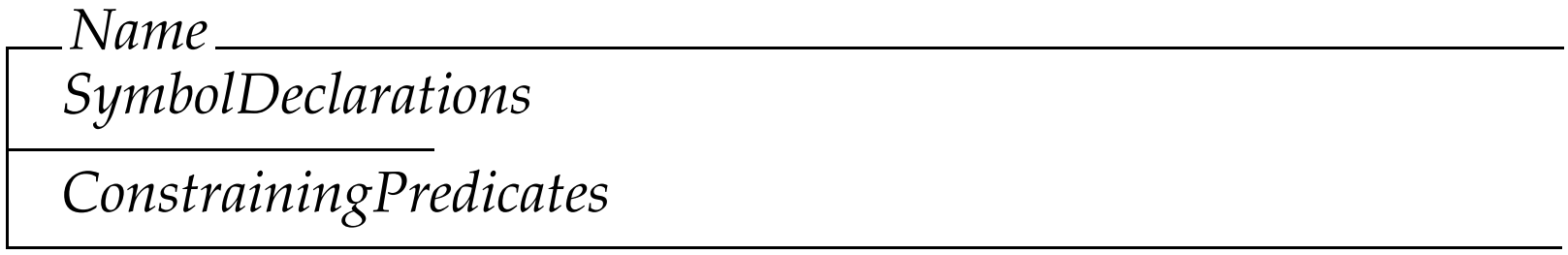
is written as $\langle x_1, x_2, \ldots, x_n \rangle$

Example: Concatenation of Sequences

$$
\begin{array}{l}
s \frown t \ == \\
\qquad s \cup \\
\qquad (\lambda\, n : \mathbb{Z} \mid n \in \#s + 1 .. \#s + \#t \bullet n - \#s) \fatsemi t
\end{array}
$$

Schemata

General form



Linear notation

$$Name \mathrel{\widehat=} [SymbolDeclarations \mid ConstrainingPredicates]$$

Schemata

With empty predicate part

$$\begin{array}{|l} Name \\ \hline SymbolDeclarations \\ \hline \end{array}$$

Linear notation

$$Name \mathrel{\widehat=} [SymbolDeclarations]$$

Schemata: Example

Theater tickets

$[Seat]$
$[Person]$

*TicketsForPerformance*0

$$seating : \mathbb{P}\, Seat$$
$$sold : Seat \pfun Person$$

$$\mathbf{dom}\, sold \subseteq seating$$

Schemata as Sets/Types

Schema

$$
\begin{array}{|l}
\hline
Name \\
x_1 : T_1 \\
\ldots \\
x_n : T_n \\
\hline
ConstrainingPredicates \\
\hline
\end{array}
$$

can be seen as the following set (type) of tuples:

$$
\begin{array}{l}
Name \ = \\
\{x_1 : T_1; \ \ldots; \ x_n : T_n \mid ConstrainingPredicates \bullet (x_1, \ldots, x_n)\}
\end{array}
$$

Schema Inclusion

Inclusion

Schemata can be used (included) in

- **schema**
- **set comprehension**
- **quantification**

by adding the schema name to the declaration part

Meaning

- **declarations**
- **constraining predicates**

are added to the corresponding parts of the including schema / set comprehension / quantification

<u>Note:</u> Matching names merge and must be type compatible

Schema Inclusion

Example

$$
\begin{array}{|l}
\hline
\textit{NumberInSet} \\
a : \mathbb{Z} \\
c : \mathbb{P}\,\mathbb{Z} \\
\hline
a \in c \\
\hline
\end{array}
$$

$$\{\textit{NumberInSet} \mid a = 0 \bullet c\}$$

is the same as

$$\{a : \mathbb{Z}, c : \mathbb{P}\,\mathbb{Z} \mid a \in c \land a = 0 \bullet c\}$$

(the set of all integer sets containing 0)

Schemata as Predicates

Schemata can be used as predicates in

- **schema**
- **set comprehension**
- **quantification**

by adding the schema name to the predicate part (occurring variables must already be declared)

Meaning

The constraining predicates (not: the declaration part) are added to the corresponding part of the schema / set comprehension / quantification

Schemata as Predicates

Example

$$
\begin{array}{|l}
\hline
\textit{NumberIn}01 \\
a : \mathbb{Z} \\
c : \mathbb{P}\,\mathbb{Z} \\
\hline
a \in c \\
c \subseteq \{0, 1\} \\
\hline
\end{array}
$$

$$\forall a : \mathbb{Z};\ c : \mathbb{P}\,\mathbb{Z} \mid \textit{NumberIn}01 \bullet \textit{NumberInSet}$$

is the same as

$$\forall a : \mathbb{Z};\ c : \mathbb{P}\,\mathbb{Z} \mid a \in c \land c \subseteq \{0, 1\} \bullet a \in c$$

Generic Schemata

Type/set variables can be used in schema definitions

Example

$$
\begin{array}{|l}
\hline
NumberInSetGeneric[X] \\
a : X \\
c : \mathbb{P}\, X \\
\hline
a \in c \\
\hline
\end{array}
$$

Then

$$NumberInSetGeneric[\mathbb{Z}] \;=\; NumberInSet$$

Variable Renaming in Schemata

Variables in schemata can be renamed

Example

$$NumberInSet[a/q, c/s]$$

is equal to

$$\begin{array}{|l} q : \mathbb{Z} \\ s : \mathbb{P}\,\mathbb{Z} \\ \hline q \in s \\ \hline \end{array}$$

Conjunctions of Schemata

Schemata can be composed conjunctively

Example

Given

$ConDis1$

$a : A;\ b : B$
P

$ConDis2$

$b : B;\ c : C$
Q

Then the following are equivalent

$$ConDis1 \land ConDis2$$

$a : A;\ b : B;\ c : C$
P
Q

Disjunctions of Schemata

Schemata can be composed disjunctively

Example

Given

$ConDis1$

$a : A;\ b : B$

P

$ConDis2$

$b : B;\ c : C$

Q

Then the following are equivalent

$ConDis1 \lor ConDis2$

$a : A;\ b : B;\ c : C$

$P \lor Q$

Example

Informal specification

Theater: Tickets for first night are only sold to friends

Specification in Z

$$Status ::= standard \mid firstNight$$

$$\begin{array}{|l} \textit{Friends} \\ \hline friends : \mathbb{P}\, Person \\ status : Status \\ sold : Seat \pfun Person \\ \hline status = firstNight \Rightarrow \mathbf{ran}\, sold \subseteq friends \\ \hline \end{array}$$

Example

$$TicketsForPerformance1 \mathrel{\widehat{=}} TicketsForPerformance0 \land Friends$$

and

$$\begin{array}{|l} TicketsForPerformance1 \\ \hline Friends \\ TicketsForPerformance0 \\ \hline \end{array}$$

Example

$$TicketsForPerformance1 \mathrel{\widehat=} TicketsForPerformance0 \land Friends$$

and

$$\begin{array}{|l} \hline TicketsForPerformance1 \\ \hline Friends \\ TicketsForPerformance0 \\ \hline \end{array}$$

are the same as

$$\begin{array}{|l} \hline TicketsForPerformance1 \\ \hline friends : \mathbb{P}\, Person;\ status : Status \\ sold : Seat \pfun Person;\ seating : \mathbb{P}\, Seat \\ \hline status = firstNight \Rightarrow \mathbf{ran}\, sold \subseteq friends \\ \mathbf{dom}\, sold \subseteq seating \\ \hline \end{array}$$

Normalisation of Schemata

Normalisation

A schema is normalised if in the declaration part

- **Variables are typed**
- **but not restricted to subsets of types**

Normalisation of Schemata

Normalisation

A schema is normalised if in the declaration part

- **Variables are typed**
- **but not restricted to subsets of types**

Example

The normalisation of $x : \mathbb{N} \mid P$ **is** $x : \mathbb{Z} \mid x \geq 0 \; P$

Negation of Schemata

A schema is negated by negating the predicate part in its normalised form

Example

The negation of

$$x : \mathbb{N} \mid P$$

is the negation of

$$x : \mathbb{Z} \mid x \in \mathbb{N} \; P$$

which is

$$x : \mathbb{Z} \mid \neg\,(x \in \mathbb{N} \land P)$$

Schemata as Operations

States

A state is a variable assignment

A schema describes a set of states

Operations

To describe an operation,
a schema must describe pairs of states (pre/post)

Schemata as Operations

States

A state is a variable assignment

A schema describes a set of states

Operations

To describe an operation,
a schema must describe pairs of states (pre/post)

Notation

Variables are decorated with $'$ to refer to their value in the post state

Whole schemata can be decorated

Schemata as Operations

Example

$$NumberInSet'$$

is the same as

$$\begin{array}{|l} \hline NumberInSet' \\ a' : \mathbb{Z} \\ c' : \mathbb{P}\,\mathbb{Z} \\ \hline a' \in c' \\ \hline \end{array}$$

Schemata as Operations

Example

$$NumberInSet'$$

is the same as

$$\begin{array}{|l}
\hline
NumberInSet' \\
a' : \mathbb{Z} \\
c' : \mathbb{P}\,\mathbb{Z} \\
\hline
a' \in c' \\
\hline
\end{array}$$

Further decorations

- **input variables are decorated with “?”**
- **output variables are decorated with “!”**

Example

Theater: Selling tickets

$$
\begin{array}{|l}
\hline
Purchase0 \\
\hline
TicketsForPerformance0 \\
TicketsForPerformance0' \\
s? : Seat \\
p? : Person \\
\hline
s? \in seating \setminus \mathbf{dom}\, sold \\
sold' = sold \cup \{s? \mapsto p?\} \\
seating' = seating \\
\hline
\end{array}
$$

(no output variables in this schema)

Example

$$Response ::= okay \mid sorry$$

Success

$$r! : Response$$

$$r! = okay$$

Then

$$Purchase0 \land Success$$

is a schema that reports successful ticket sale

Schemata as Operations

General Form

StateSpace

$x_1 : T_1; \ \ldots; \ x_n : T_n$

$inv(x_1, \ldots, x_n)$

Operation

StateSpace

StateSpcae′

$i_1? : U_1; \ \ldots; \ i_m? : U_m$

$o_1! : V_1; \ \ldots; \ o_p! : V_p$

$pre(i_1?, \ldots, i_m?, \ x_1, \ldots, x_n)$

$op(i_1?, \ldots, i_m?, \ x_1, \ldots, x_n, \ x'_1, \ldots, x'_n, \ o_1!, \ldots, o_p!)$

The Δ Operator

Definition

$\Delta Schema$ **abbreviates** $Schema \wedge Schema'$

General form of operation schema using Δ

Operation

$\Delta StateSpace$

$i_1? : U_1;\ \ldots;\ i_m? : U_m$

$o_1! : V_1;\ \ldots;\ o_p! : V_p$

$pre(i_1?, \ldots, i_m?,\ x_1, \ldots, x_n)$

$op(i_1?, \ldots, i_m?,\ x_1, \ldots, x_n,\ x'_1, \ldots, x'_n,\ o_1!, \ldots, o_p!)$

The Ξ Operator

Definition

$\Xi Schema$ **abbreviates** $\Delta Schema \land (x_1 = x_1' \land \ldots \land x_n = x_n')$

where $x_1, \ldots x_n$ **are the variables declared in** $Schema$

General form of operation schema using Ξ

Operation

$\Xi StateSpace$

$i_1? : U_1;\ \ldots;\ i_m? : U_m$

$o_1! : V_1;\ \ldots;\ o_p! : V_p$

$pre(i_1?, \ldots, i_m?,\ x_1, \ldots, x_n)$

$op(i_1?, \ldots, i_m?,\ x_1, \ldots, x_n,\ o_1!, \ldots, o_p!)$

Using Ξ indicates that the operation does not change the state

The Operators Δ and Ξ: Example

The following schemata are equivalent

$$\Xi NumberInSet$$

$\Delta NumberInSet$
$a = a'$
$c = c'$

$NumberInSet$ $NumberInSet'$
$a = a'$
$c = c'$

Example

Theater: Selling tickets, but only to friends if first night performance

$$
\begin{array}{|l}
\hline
Purchase1 \\
\Delta TicketsForPerformance1 \\
s? : Seat \\
p? : Person \\
\hline
s? \in seating \setminus \mathbf{dom}\, sold \\
status = firstNight \Rightarrow (p? \in friends) \\
sold' = sold \cup \{s? \mapsto p?\} \\
seating' = seating \\
status' = status \\
friends' = friends \\
\hline
\end{array}
$$

Example

NotAvailable

$\Xi TicketsForPerformance1$
$s? : Seat$
$p? : Person$

$s? \in \mathbf{dom}\, sold \lor (status = firstNight \land \neg\, p? \in friends)$

Failure

$r! : Response$

$r! = sorry$

$$TicketServiceForPerformance \mathrel{\widehat{=}} (Purchase1 \land Success) \lor (NotAvailable \land Failure)$$

Quantifying Variables in Schemata

Schema quantification

$\forall x : S \bullet Schema$ **resp.**
$\exists x : S \bullet Schema$

(existential quantification is also called “variable hiding”)

Quantifying Variables in Schemata

Schema quantification

$$\forall x : S \bullet Schema \qquad \textbf{resp.}$$
$$\exists x : S \bullet Schema$$

(existential quantification is also called “variable hiding”)

Example

$$\exists a : \mathbb{Z} \bullet NumberInSet$$

is the same as

$$c : \mathbb{P}\,\mathbb{Z}$$

$$\exists a : \mathbb{Z} \bullet a \in c$$

Composition of Operation Schemata

Definition

Operation schemata can be composed using $\fatsemi$, where

- **every variable with $'$ in the first schema must occur without $'$ in the second schema**
- **these variables are identified and**
- **hidden from the outside**

Composition: General form

$Op1$

$x_1 : T_1; \ldots; x_p : T_p$
$z_1 : V_1; \ldots; z_n : V_n$
$z'_1 : V_1; \ldots; z'_n : V_n$

$op1(x_1, \ldots, x_p, z_1, \ldots, z_n, z'_1, \ldots, z'_n)$

$Op2$

$y_1 : U_1; \ldots; y_q : U_q$
$z_1 : V_1; \ldots; z_n : V_n$
$z'_1 : V_1; \ldots; z'_n : V_n$

$op2(y_1, \ldots, y_q, z_1, \ldots, z_n, z'_1, \ldots, z'_n)$

$Op1 \fatsemi Op2$

$x_1 : T_1; \ldots; x_p : T_p$
$y_1 : U_1; \ldots; y_q : U_q$
$z_1 : V_1; \ldots; z_n : V_n$
$z'_1 : V_1; \ldots; z'_n : V_n$

$\exists z''_1 : V_1; \ldots; z''_n : V_n \bullet$
$op1(x_1, \ldots, x_p, z_1, \ldots, z_n, z''_1, \ldots, z''_n)$
$op2(y_1, \ldots, y_q, z''_1, \ldots, z_n, z'_1, \ldots, z'_n)$

Example

$$Purchase1 \fatsemi Purchase1[s?/s2?]$$

is equivalent to

$$\Delta TicketsForPerformance1$$
$$s? : Seat;\ s2? : Seat;\ p? : Person$$

$$s? \in seating \setminus \mathbf{dom}\, sold$$
$$s2? \in seating \setminus \mathbf{dom}(sold \cup \{s? \mapsto p?\})$$
$$status = firstNight \Rightarrow (p? \in friends)$$
$$sold' = sold \cup \{s? \mapsto p?, s2? \mapsto p?\}$$
$$seating' = seating$$
$$status' = status$$
$$friends' = friends$$