Z Reference Card

Mike Spivey

Specifications

Schema box

Name[*Params*]
Declarations
———
Predicates
	`\begin{schema}{Name}[Params]`
	`  Declarations`
	`\where`
	`  Predicates`
	`\end{schema}`

Axiomatic description

Declarations
———
Predicates
	`\begin{axdef}`
	`  Declarations`
	`\where`
	`  Predicates`
	`\end{axdef}`

Generic definition

[*Params*]
Declarations
———
Predicates
	`\begin{gendef}[Params]`
	`  Declarations`
	`\where`
	`  Predicates`
	`\end{gendef}`

`\begin{zed} ...`

Basic type definition	
$[NAME, DATE]$	`[NAME, DATE]`
Abbreviation definition	
$DOC == \mathrm{seq}\, CHAR$	`DOC == \seq CHAR`
Constraint	
$n\_disks < 5$	`n\_disks < 5`
Schema definition	
$Point \mathrel{\widehat=} [\, x, y : \mathbb{Z}\,]$	`Point \defs [~x, y: \num~]`
Free type definition	
$Ans ::= ok \langle\!\langle \mathbb{Z} \rangle\!\rangle \mid error$	`Ans ::= ok \ldata\num\rdata | error`

`... \end{zed}`

Logic and schema calculus

$true, false$	`true, false`	Logical constants
$\neg P$	`\lnot P`	Negation
$P \land Q$	`P \land Q`	Conjunction
$P \lor Q$	`P \lor Q`	Disjunction
$P \Rightarrow Q$	`P \implies Q`	Implication
$P \Leftrightarrow Q$	`P \iff Q`	Equivalence
$\forall x : T \mid P \bullet Q$	`\forall ...`	Universal quantifier
$\exists x : T \mid P \bullet Q$	`\exists ...`	Existential quantifier
$\exists_1 x : T \mid P \bullet Q$	`\exists_1 ...`	Unique quantifier

Special schema operators

$S[y_1/x_1, y_2/x_2]$	`S[y_1/x_1, y_2/x_2]`	Renaming
$S \setminus (x_1, x_2)$	`S \hide (x_1, x_2)`	Hiding
$S1 \upharpoonright S2$	`S1 \project S2`	Projection
$\mathrm{pre}\ Op$	`\pre Op`	Pre-condition
$Op1 \fatsemi Op2$	`Op1 \semi Op2`	Sequential composition
$Op1 \gg Op2$	`Op1 \pipe Op2`	Piping

Basic expressions

$x = y$	`x = y`	Equality
$x \neq y$	`x \neq y`	Inequality
if P **then** E_1	`\IF P \THEN E_1`	Conditional
else E_2	`\ELSE E_2`	Expression
θS	`\theta S`	Theta-expression
$E.x$	`E.x`	Selection
$(\mu\, x : T \mid P \bullet E)$	`(\mu x: T | P @ E)`	Mu-expression
$(\mathbf{let}\ x{=}{=}E1 \bullet E2)$	`(\LET x == E1 @ E2)`	Let-expression

Sets

$x \in S$	`x \in S`	Membership
$x \notin S$	`x \notin S`	Non-membership
$\{x_1, \ldots, x_n\}$	`\{x_1, ..., x_n\}`	Set display
$\{\, x : T \mid P \bullet E \,\}$	`\{~x: T | P @ E~\}`	Set comprehension
$\varnothing$	`\emptyset`	Empty set
$S \subseteq T$	`S \subseteq T`	Subset relation
$S \subset T$	`S \subset T`	Proper subset relation
$\mathbb{P}\, S$	`\power S`	Power set
$\mathbb{P}_1\, S$	`\power_1 S`	Non-empty subsets
$S \times T$	`S \cross T`	Cartesian product
(x, y, z)	`(x, y, z)`	Tuple
$\mathit{first}\ p$	`first~p`	First of pair
$\mathit{second}\ p$	`second~p`	Second of pair
$S \cup T$	`S \cup T`	Set union
$S \cap T$	`S \cap T`	Set intersection
$S \setminus T$	`S \setminus T`	Set difference
$\bigcup A$	`\bigcup A`	Generalized union
$\bigcap A$	`\bigcap A`	Generalized intersection
$\mathbb{F}\, X$	`\finset X`	Finite sets
$\mathbb{F}_1\, X$	`\finset_1 X`	Non-empty finite sets

Relations

$X \leftrightarrow Y$	`X \rel Y`	Binary relations
$x \mapsto y$	`x \mapsto y`	Maplet
$\mathrm{dom}\, R$	`\dom R`	Domain
$\mathrm{ran}\, R$	`\ran R`	Range
$\mathrm{id}\, X$	`\id X`	Identity relation
$Q \mathbin{⨾} R$	`Q \comp R`	Composition
$Q \circ R$	`Q \circ R`	Backwards composition
$S \lhd R$	`S \dres R`	Domain restriction
$R \rhd S$	`R \rres S`	Range restriction
$S \mathbin{⩤} R$	`S \ndres R`	Domain anti-restriction
$R \mathbin{⩥} S$	`R \nrres S`	Range anti-restriction
$R^{\sim}$	`R \inv`	Relational inverse
$R(\!\vert S\vert\!)$	`R \limg S\rimg`	Relational image
$Q \oplus R$	`Q \oplus R`	Overriding
R^k	`R^{k}`	Iteration
R^+	`R \plus`	Transitive closure
R^*	`R \star`	Reflexive–trans. closure

Functions

$f(x)$	`f(x)`	Function application
$(\lambda\, x : T \mid P \bullet E)$	`(\lambda ...)`	Lambda-expression
$X \mathbin{⇸} Y$	`X \pfun Y`	Partial functions
$X \longrightarrow Y$	`X \fun Y`	Total functions
$X \mathbin{⤔} Y$	`X \pinj Y`	Partial injections
$X \rightarrowtail Y$	`X \inj Y`	Total injections
$X \mathbin{⤀} Y$	`X \psurj Y`	Partial surjections
$X \twoheadrightarrow Y$	`X \surj Y`	Total surjections
$X \mathbin{⤖} Y$	`X \bij Y`	Bijections
$X \mathbin{⇻} Y$	`X \ffun Y`	Finite partial functions
$X \mathbin{⤕} Y$	`X \finj Y`	Finite partial injections

Numbers and arithmetic

$\mathbb{N}$	`\nat`	Natural numbers
$\mathbb{Z}$	`\num`	Integers
$+ - *$ div mod	`+ - * \div \mod`	Arithmetic operations
$< \leq \geq >$	`< \leq \geq >`	Arithmetic comparisons
$\mathbb{N}_1$	`\nat_1`	Strictly positive integers
$succ$	`succ`	Successor function
$m \mathrel{..} n$	`m \upto n`	Number range
$\#S$	`\# S`	Size of a set
$min\ S$	`min~S`	Minimum of a set
$max\ S$	`max~S`	Maximum of a set

Sequences

$\mathrm{seq}\ X$	`\seq X`	Finite sequences
$\mathrm{seq}_1\ X$	`\seq_1 X`	Non-empty sequences
$\mathrm{iseq}\ X$	`\iseq X`	Injective sequences
$\langle x_1, \ldots, x_n \rangle$	`\langle ... \rangle`	Sequence display
$s \frown t$	`s \cat t`	Concatenation
$rev\ s$	`rev~s`	Reverse
$head\ s$	`head~s`	Head of sequence
$last\ s$	`last~s`	Last element of sequence
$tail\ s$	`tail~s`	Tail of sequence
$front\ s$	`front~s`	All but last element
$U \upharpoonleft s$	`U \extract S`	Extraction
$s \upharpoonright V$	`s \filter V`	Filtering
$squash\ f$	`squash~f`	Compaction
s prefix t	`s \prefix t`	Prefix relation
s suffix t	`s \suffix t`	Suffix relation
s in t	`s \inseq t`	Segment relation
$\frown/ss$	`\dcat ss`	Distributed concat.
$\mathsf{disjoint}$ SS	`\disjoint SS`	Disjointness
SS $\mathsf{partition}$ T	`SS \partition T`	Partition relation

Bags

$\mathrm{bag}\ X$	`\bag X`	Bags
$[\![x_1, \ldots, x_n]\!]$	`\lbag ... \rbag`	Bag display
$count\ B\ x$	`count~B~x`	Count of an element
$B \sharp x$	`B \bcount x`	Infix count operator
$n \otimes B$	`n \otimes B`	Bag scaling
$x \sqsubset B$	`x \inbag B`	Bag membership
$B \sqsubseteq C$	`B \subbageq C`	Sub-bag relation
$B \uplus C$	`B \uplus C`	Bag union
$B \uminus C$	`B \uminus C`	Bag difference
$items\ s$	`items~s`	Items in a sequence

*f*UZZ flags

Usage: `fuzz` *[*`-aqstv`*]* *[*`-p` *prelude]* *[file ...]*

`-a`	Don't use type abbreviations
`-p` *predude*	Use *prelude* in place of the standard one
`-q`	Assume implicit quantifiers for undeclared variables
`-d`	Dependency analysis
`-s`	Syntax check only
`-t`	Report types of global definitions
`-v`	Echo formal text as it is parsed