Logical and Mathematical Notation

William M. Farmer

30 August 2004

	BESTT	**PVS**	**Oberon**	**Huth/Ryan**	**Dr. Khedri**
truth values type	$*$	`bool`	`BOOLEAN`	—	**B**
truth	T	`TRUE`	`TRUE`	T, $\top$	true
falsehood	F	`FALSE`	`FALSE`	F, $\bot$	false
conjunction	$A \wedge B$	A `AND` B, A `&` B	A `&` B	$A \wedge B$	$A \wedge B$
disjunction	$A \vee B$	A `OR` B	A `OR` B	$A \vee B$	$A \vee B$
negation	$\neg A$	`NOT` A	`~`A	$\neg A$	$\neg A$
implication	$A \Rightarrow B$	A `IMPLIES` B, A `=>` B	—	$A \rightarrow B$	$A \Rightarrow B$
iff (biimplication)	$A \Leftrightarrow B$	A `IFF` B, A `<=>` B	A `=` B	$A \leftrightarrow B$	$A \Leftrightarrow B$
universal quantification	$\forall x : \alpha \,.\, A$	`(FORALL(`x`:`α`):`A`)`	—	$\forall x A$	$\forall(x \mid \alpha : A)$
existential quantification	$\exists x : \alpha \,.\, A$	`(EXISTS(`x`:`α`):`A`)`	—	$\exists x A$	$\exists(x \mid \alpha : A)$
equality	$s = t$	s `=` t	s `=` t	$s = t$	$s = t$
inequality	$s \neq t$	s `/=` t	s `#` t	$s \neq t$	$s \neq t$
if-then-else	if(A, s, t)	`IF` A `THEN` s `ELSE` t	—	—	—
functions type	$\alpha \rightarrow \beta$	α `->` β	—	—	$\alpha \rightarrow \beta$
function application	$f(a_1, \ldots, a_n)$	f`(`$a_1, \ldots, a_n$`)`	f`(`$a_1, \ldots, a_n$`)`	$f(a_1, \ldots, a_n)$	$f(a_1, \ldots, a_n)$
function abstraction	$\lambda x : \alpha \,.\, t$	`(LAMBDA(`x`:`α`):`t`)`	—	—	—
definedness	$t\downarrow$	—	—	—	—
undefinedness	$t\uparrow$	—	—	—	—
equivalence $(s\downarrow \vee t\downarrow \supset s = t)$	$s \simeq t$	—	—	—	$s \stackrel{\triangle}{=} t$
definite description	$\mathrm{I}\, x : \alpha \,.\, A$	—	—	—	—
undefined expression	$\bot$, $\bot_\alpha$	—	—	—	—
tuples type	$\alpha_1 \times \cdots \times \alpha_n$	`[`$\alpha_1, \ldots, \alpha_n$`]`	—	—	$\alpha_1 \times \cdots \times \alpha_n$
tuple	$(a_1, \ldots, a_n)$	$(a_1, \ldots, a_n)$	—	—	$(a_1, \ldots, a_n)$
tuple projection	#$m(a_1, \ldots, a_n)$	`proj_`m`(`$a_1, \ldots, a_n$`)`	—	—	—
list type	list$[\alpha]$	`list[`α`]`	—	—	—
list	$\langle a_1, \ldots, a_n \rangle$	`(:`$a_1, \ldots, a_n$`:)`	—	—	—
empty list	nil	`null`	—	—	—
list construction	mlist(a, L)	`cons(`a, L`)`	—	—	—
list head	hd(L)	—	—	—	—
list tail	tl(L)	—	—	—	—
list projection	$L[m]$	—	—	—	—
set type	set$[\alpha]$	`set[`α`]`, `setof[`α`]`	—	—	$\mathcal{P}(\alpha)$
set	$\{a_1, \ldots, a_n\}$	$\{a_1, \ldots, a_n\}$	—	$\{a_1, \ldots, a_n\}$	$\{a_1, \ldots, a_n\}$
set membership	$a \in S$	`member(`a, S`)`, $S(a)$	s `IN` t	$a \in S$	$a \in S$
set abstraction	$\{x : \alpha \mid A\}$	$\{x$`:`α `|` $A\}$, (A)	—	—	$\{x : \alpha \mid A\}$