

1 Basic Theories

1.1 Boolean Theory

Operators Some boolean operators are supported by \LaTeX , but they have names suggesting shape rather than content, e.g., \a \vee b . It would be nice if they were given informative, short names (because you don't want to enter \conjunction all the time) without clashing with existing \LaTeX commands (e.g., \and is already taken).

$a \wedge b$	\a \et b
$a \vee b$	\a \vel b
$a \Rightarrow b$	\a \imp b
$a \Rightarrow b$	\a \Imp b
$a \Leftarrow b$	\a \pmi b or \a \impby b
$a \Leftarrow b$	\a \Pmi b or \a \Impby b
$a = b$	\a \Eq b
$\text{\if b then x else y}$	$\text{\cond\{b\}\{x\}\{y\}}$

The first two come from Latin. The rest should be obvious. Other boolean symbols (\perp , $=$, etc.) have reasonable names in \LaTeX , and I will not show them.

Proof format Using the \align* environment provided by \AMS-L\TeX , a calculational proof with hints can be typeset easily. The first proof in the textbook:

$a \wedge b \Rightarrow c$	Material Implication
$= \neg(a \wedge b) \vee c$	Duality
$= \neg a \vee \neg b \vee c$	Material Implication
$= a \Rightarrow \neg b \vee c$	Material Implication
$= a \Rightarrow (b \Rightarrow c)$	

Its code:

```
\begin{align*}
&\text{\Blank a \et b \imp c} && \text{\&\& \text{\text{Material Implication}} \text{\&\&} \\
&\text{\Eq \neg(a \et b) \vel c} && \text{\&\& \text{\text{Duality}} \text{\&\&} \\
&\text{\Eq \neg a \vel \neg b \vel c} && \text{\&\& \text{\text{Material Implication}} \text{\&\&} \\
&\text{\Eq a \imp \neg b \vel c} && \text{\&\& \text{\text{Material Implication}} \text{\&\&} \\
&\text{\Eq a \imp (b \imp c)} && \\
\end{align*}
```

The command \Blank is a blank relation symbol I invented; it is necessary in that position to keep \align* happy. Its definition is simply: $\text{\mathrel\{\}}$.

2 Basic Data Structures

Bunch Theory

A, B	A,B
$A \dot{B}$	A'B
$null$	\nul
$\#A$	\card A
$0, \dots, 10$	0 \bto 10
$nat, xnat$	\nat, \xnat
$int, xint$	\int, \xint
$rat, xrat$	\rat, \xrat

String Theory

nil	\nil
n^*S	n^*S
$*S$	{ }^*S
$0; \dots, 10$	0 \sto 10

List Theory

L^+M	L^+M
$n \rightarrow i \mid L$	n \to i \ow L
Ln	L \ap n

3 Function Theory

The `\fun` and `\fn` commands produce functions; `\fun` requires a domain and `\fn` omits the domain.

$\langle x: nat \rightarrow x + 1 \rangle$	<code>\fun{x}{\nat}{x+1}</code>
$\langle x \rightarrow x + 1 \rangle$	<code>\fn{x}{x+1}</code>

The `\bind` and `\bnd` commands help you produce quantified expressions. They just have the quantifier missing, and you just put it back. `\bind` requires a domain and `\bnd` omits the domain. Some examples:

$\forall x \cdot x = x$	<code>\forall\forall\bnd{x}{x=x}</code>
$\Sigma i: 0, \dots, 10 \cdot i^2$	<code>\Sigma\Sigma\bind{i}{0 \bto 10}{i^2}</code>
$\S x: nat \cdot x/2 : nat$	<code>\S\S\bnd{x}{\nat}{x/2:\nat}</code>

Two quantifiers are not already available in \LaTeX : *MAX* and *MIN*. I have defined them as `\MAX` and `\MIN`, respectively.

Both application and composition are `\ap`. You can think of it as standing for “apposition”. Selective union is `\ow`, standing for “otherwise”. You have seen them in List Theory. More examples:

$MAX v: x \cdot n$	$\backslash MAX \backslash bind \{v\} \{x\} \{n\}$
$MIN v: x \cdot n$	$\backslash MIN \backslash bind \{v\} \{x\} \{n\}$
$f g$	$f \backslash ow \ g$
$h f x g y$	$h \backslash ap \ f \backslash ap \ x \backslash ap \ g \backslash ap \ y$

4 Program Theory

ok	$\backslash ok$
$S . R$	$S \backslash dc \ R$
$x := e$	$x \backslash get \ e$