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., \( a \Rightarrow b \). It would be nice if they were given informative, short names without clashing with existing \LaTeX commands.

\[
\begin{align*}
  a \Rightarrow b & \quad a \imp b \\
  a \Rightarrow b & \quad a \Imp b \\
  a \Leftarrow b & \quad a \mif b \text{ or } a \Impby b \\
  a \Leftarrow b & \quad a \Pmi b \text{ or } a \Impby b \\
  a = b & \quad a \Eq b \\
  \text{if } c \text{ then } x \text{ else } y \text{ fi} & \quad \cond{c}{x}{y}
\end{align*}
\]

Other boolean symbols (=, \bot for \( \bot \), \lor for \( \lor \), etc.) have reasonable names in \LaTeX, and I will not show them.

Two more variants of if-then-else-fi:

- \condb{c}{x}{y}:
  \[
  \begin{align*}
    & \text{if } c \text{ then } x \\
    & \text{else } y \text{ fi}
  \end{align*}
  \]

- \condbb{c}{x}{y}:
  \[
  \begin{align*}
    & \text{if } c \text{ then } x \\
    & \text{else } y \text{ fi}
  \end{align*}
  \]

Proof format  Using the align* environment provided by \texttt{AMS-LaTeX}(package name amsmath), a calculational proof with hints can be typeset easily. The first proof in the textbook:

\[
\begin{align*}
  a \wedge b \Rightarrow c & \quad \text{Material Implication} \\
  = \neg(a \wedge b) \vee c & \quad \text{Duality} \\
  = \neg a \vee \neg b \vee c & \quad \text{Material Implication} \\
  = a \Rightarrow \neg b \vee c & \quad \text{Material Implication} \\
  = a \Rightarrow (b \Rightarrow c)
\end{align*}
\]

Its code:

\begin{verbatim}
\begin{align*}
  \&\text{Blank a \et b \imp c} & \& \text{Material Implication} \\
  \&\text{Neg (a \et b) \vel c} & \& \text{Duality} \\
  \&\text{Neg a \vel neg b \vel c} & \& \text{Material Implication}
\end{align*}
\end{verbatim}
The command \Blank is a blank relation symbol I invented; it is necessary in that position to keep align* happy. Its definition is simply: \mathrel{\phantom{\Eq}}.

2 Basic Data Structures

Bunch Theory

\begin{align*}
A, B & \quad A, B \\
A' B & \quad A'B \\
null & \quad \null \\
\mathrel{\phantom{\Eq}}. A & \quad \card A \\
0 \ldots 10 & \quad 0 \ \mathbf{bto} \ 10 \\
\mathrel{\phantom{\Eq}}. \ nat, xnat & \quad \nat, \ \xnat \\
\mathrel{\phantom{\Eq}}. \ int, xint & \quad \int, \ \xint \\
\mathrel{\phantom{\Eq}}. \ rat, xrat & \quad \rat, \ \xrat
\end{align*}

String Theory

\begin{align*}
\mathrel{\phantom{\Eq}}. \ nil & \quad \nil \\
\mathrel{\phantom{\Eq}}. \ n^* S & \quad n^*S \\
\mathrel{\phantom{\Eq}}. \ {}^* S & \quad {}^*S \\
0 \ldots 10 & \quad 0 \ \mathbf{sto} \ 10
\end{align*}

List Theory

\begin{align*}
L + M & \quad L^+M \\
L n & \quad L \ \mathbf{ap} \ n
\end{align*}

3 Function Theory

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

\begin{align*}
\langle x : nat \to x + 1 \rangle & \quad \fun{x}{\nat}{x+1} \\
\langle x \to x + 1 \rangle & \quad \fn{x}{x+1}
\end{align*}

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:

\begin{align*}
\forall x : x = x & \quad \forall x \bnd{x}{x=x} \\
\Sigma i : 0 \ldots 10 \cdot i^2 & \quad \Sigma i \bnd{i}{0 \ \mathbf{bto} \ 10}{i^2} \\
\mathrel{\phantom{\Eq}}. x : nat \cdot x/2 : nat & \quad \S x \bnd{x}{\nat}{x/2 : \nat}
\end{align*}
Two quantifiers are not already available in \LaTeX: \textit{MAX} and \textit{MIN}. I have defined them as \texttt{\textbackslash{MAX}} and \texttt{\textbackslash{MIN}}, respectively.

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

\begin{align*}
\text{MAX} \: v \cdot x \cdot n & \quad \texttt{\textbackslash{MAX}\textbackslash{bind}(v){x}{n}} \\
\text{MIN} \: v \cdot x \cdot n & \quad \texttt{\textbackslash{MIN}\textbackslash{bind}(v){x}{n}} \\
 f \mid g & \quad f \texttt{\textbackslash{ow}} g \\
h \, f \, x \, g \, y & \quad h \texttt{\textbackslash{ap}} f \texttt{\textbackslash{ap}} x \texttt{\textbackslash{ap}} g \texttt{\textbackslash{ap}} y
\end{align*}

4 Program Theory

\begin{align*}
\texttt{\textbackslash{ok}} & \quad \texttt{\textbackslash{ok}} \\
S \cdot R & \quad S \texttt{\textbackslash{dc}} R \\
x := e & \quad x \texttt{\textbackslash{get}} e
\end{align*}

5 Programming Language

Two forms of while-do-od:

- \texttt{\textbackslash{while}(c){P}}: \quad \texttt{while } c \texttt{ do } P \texttt{ od}

- \texttt{\textbackslash{whileb}(c){P}}: \quad \texttt{while } c \texttt{ do } P \texttt{ od}