4

Logic and Proof Beyond Equality

4.1 Formulas

The core syntax of formulas (form below) provides the standard logical constructs, in decreasing order of precedence:

\[\text{form ::= (form) | True | False | term = term} \]
\[\text{ | ¬ form | form \land form | form \lor form | form \to form} \]
\[\text{ | \forall x. form | \exists x. form} \]

Terms are the ones we have seen all along, built from constants, variables, function application and \(\lambda\)-abstraction, including all the syntactic sugar like infix symbols, if, case, etc.

\[\begin{align*}
\text{Remember that formulas are simply terms of type } \text{bool. Hence } \text{=} \text{ also works for formulas. Beware that } \text{=} \text{ has a higher precedence than the other logical operators. Hence } s = t \land A \text{ means } (s = t) \land A, \text{ and } A \land B = B \land A \text{ means } A \land (B = B) \land A. \text{ Logical equivalence can also be written with } \leftrightarrow \text{ instead of } =, \text{ where } \leftrightarrow \text{ has the same low precedence as } \to. \text{ Hence } A \land B \leftrightarrow B \land A \text{ really means } (A \land B) \leftrightarrow (B \land A). \\
\text{Quantifiers need to be enclosed in parentheses if they are nested within other constructs (just like if, case and let).}
\end{align*}\]

The most frequent logical symbols and their ASCII representations are shown in Fig. 4.1. The first column shows the symbols, the other columns ASCII representations. The \(<\ldots\text{> form is always converted into the symbolic form by the Isabelle interfaces, the treatment of the other ASCII forms depends on the interface. The ASCII forms } \text{:\textbackslash {\textbackslash} and } \text{:\textbackslash / are special in that they are merely keyboard shortcuts for the interface and not logical symbols by themselves.} \]
Fig. 4.1. Logical symbols and their ASCII forms

The implication \( \rightarrow \) is part of the Isabelle framework. It structures theorems and proof states, separating assumptions from conclusions. The implication \( \to \) is part of the logic HOL and can occur inside the formulas that make up the assumptions and conclusion. Theorems should be of the form \[ [ A_1; \ldots; A_n ] \to A \], not \( A_1 \land \ldots \land A_n \to A \). Both are logically equivalent but the first one works better when using the theorem in further proofs.

4.2 Sets

Sets of elements of type \( 'a \) have type \( 'a \) set. They can be finite or infinite. Sets come with the usual notation:

- \( \{ \}, \{ e_1, \ldots, e_n \} \)
- \( e \in A, \quad A \subseteq B \)
- \( A \cup B, \quad A \cap B, \quad A - B, \quad - A \)

(where \( A - B \) and \( -A \) are set difference and complement) and much more. \( UNIV \) is the set of all elements of some type. Set comprehension is written \( \{ x. \ P \} \) rather than \( \{ x \mid P \} \).

In \( \{ x. \ P \} \) the \( x \) must be a variable. Set comprehension involving a proper term \( t \) must be written \( \{ t \mid x. \ P \} \), where \( x, y \) are those free variables in \( t \) that occur in \( P \). This is just a shorthand for \( \{ v. \ \exists x. \ y. \ v = t \land P \} \), where \( v \) is a new variable.

For example, \( \{ x + y \mid x. \ x \in A \} \) is short for \( \{ v. \ \exists x. \ v = x+y \land x \in A \} \).

Here are the ASCII representations of the mathematical symbols:

- \( \in \) : \( \in \)
- \( \subseteq \) : \( \subseteq \)
- \( \cup \) : \( \cup \)
- \( \cap \) : \( \cap \)

Sets also allow bounded quantifications \( \forall x \in A. \ P \) and \( \exists x \in A. \ P \).

For the more ambitious, there are also \( \bigcup \) and \( \bigcap \):