Getting Started With Isabelle

Lecture II: Theory Files

Lawrence C. Paulson
Computer Laboratory

UNIVERSITY OF CAMBRIDGE
Syntax Fundamentals

**sorts** to classify types for overloading*

**types** to classify terms (including polymorphism)

**terms** and formulas (which are just Boolean terms)

**inference rules** as assertions of the meta-logic

**theory files** to declare types, constants, etc.

**proof files** containing Goal, by, qed commands

**new-style theories** by Markus Wenzel (Isar)*

*not in this course
Types in Isabelle/HOL

\[ \sigma \Rightarrow \tau \] function types

'\(a\), '\(b\), ... type variables (like in ML)

bool, nat, ... base types

'\(a\) list, ... type constructors

(bool*nat)list instance of a type constructor

\(x :: \tau\) means “\(x\) has type \(\tau\)"
Type $\texttt{bool}$: Formulas of Higher-Order Logic

$\neg P$  negation of $P$

$P \land Q$  conjunction of $P$ and $Q$

$P \lor Q$  disjunction of $P$ and $Q$

$P \rightarrow Q$  implication between $P$ and $Q$

$(P) = (Q)$  logical equivalence of $P$ and $Q$

$\forall x. \ P$ or $\exists x. \ P$  for all (universal quantifier)

$\exists x. \ P$ or $\exists x. \ P$  for some (existential quantifier)

Also conditional expressions: if $P$ then $t$ else $u$
**Numeric Types** $\text{nat, int, real, ...}$

- $-x$ unary minus of $x$ all numerics
- $+ - *$ sum, difference, product all numerics
- $\text{#ddd}$ binary numerals all numerics
- $\text{div mod}$ quotient, remainder types $\text{nat, int}$
- $\text{Suc } n$ successor $n + 1$ type $\text{nat}$
- $0 1 2$ unary numerals type $\text{nat}$
- $< <=$ orderings overloaded
- $= \sim =$ equality, non-equality overloaded

Automatic simplification, including linear arithmetic
**Lists: the Type Constructor 'a list**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nil</td>
<td>the empty list</td>
</tr>
<tr>
<td>Cons $x ; l$</td>
<td>list with head $x$, tail $l$</td>
</tr>
<tr>
<td>$xs ; @ ; ys$</td>
<td>append of $xs$, $ys$</td>
</tr>
<tr>
<td>$hd ; tl ; rev...$</td>
<td>common list functions</td>
</tr>
<tr>
<td>map filter...</td>
<td>common list functionals</td>
</tr>
<tr>
<td>$[x_1, \ldots, x_n]$</td>
<td>list notation</td>
</tr>
<tr>
<td>$[x:l. ; P]$</td>
<td>nice syntax for filter</td>
</tr>
</tbody>
</table>
Sets: the Type Constructor ‘a set

\[
\begin{align*}
\text{x : A} & \quad \text{membership, } x \in A \\
\text{x \sim: A} & \quad \text{non-membership, } x \notin A \\
\text{A <= B} & \quad \text{subset, } A \subseteq B \\
\text{-A} & \quad \text{complement of } A \\
\text{A Un B} & \quad \text{union of } A \text{ and } B \\
\text{A Int B} & \quad \text{intersection of } A \text{ and } B \\
\text{ALL x:A. P} & \quad \text{bounded quantifier (also EX)} \\
\text{UN x:A. P} & \quad \text{union of a family of sets (also INT)}
\end{align*}
\]
Tupled and Curried Functions

\[ [\sigma_1, \ldots, \sigma_n] \Rightarrow \tau \] curried function type

\( \% x_1 \ldots x_n. \, t \) curried \( \lambda \)-abstraction

\( f \, t_1 \ldots t_n \) curried function application

\[ \sigma_1 * \ldots * \sigma_n \Rightarrow \tau \] tupled function type

\( \% (x_1, \ldots, x_n). \, t \) tupled \( \lambda \)-abstraction

\( f \, (t_1, \ldots, t_n) \) tupled function application

Tupled abstraction allowed elsewhere:

\[ \text{ALL} \, (x, y) : \text{edges}. \, x \sim= y \]
**Constants and Variables**

Name spaces resolve duplicate constant declarations.

Identifiers not declared as constants can be variables.

Unknowns are instantiated automatically.

\[ T.c \] constant \( c \) declared in theory \( T \)

\[ c \] constant declared most recently

\[ x \] free variable (if not declared as a constant)

\[ ?x \] schematic variable (unknown)
Format of a Theory File

\[ T = T_1 + \cdots + T_n + \]

\textbf{consts} \ uList :: "'a => 'a list"

\textbf{defs} \ uList_def "uList x == [x]"
\hspace{1cm} (*note the == symbol!*)

\textbf{rules} \ f_axiom "f(f n) < f (Suc n)"

\textbf{record} ...

\textbf{inductive} ...

\textbf{end}

Extend theories \( T_1, \ldots , T_n \) with constants, axioms, record declarations, etc., etc.
Further Material Provided by Isabelle/HOL

Relations — their properties and operations on them
Equivalence classes — quotients and congruences
Well-foundedness of many orderings including multisets
Cardinality including binomials and powersets
Non-standard analysis (thanks to Jacques Fleuriot)
Prime numbers — GCDs, unique factorization

Browse the Isabelle theory library on the WWW