Session III

More about Isar
Overview

- Abbreviations
- Predicate Logic
- Accumulating facts
- Reasoning with chains of equations
- Locales: the module system
Abbreviations

\[
\begin{align*}
this & = \text{the previous proposition proved or assumed} \\
then & = \text{from this} \\
\text{with } \vec{a} & = \text{from } \vec{a} \text{ this} \\
?thesis & = \text{the last enclosing show formula}
\end{align*}
\]
Mixing proof styles

from . . . have . . .

apply - make incoming facts assumptions
apply ( . . )
:
apply ( . . )
done
Demo: Abbreviations
Predicate Calculus
Syntax:

\texttt{fix variables}

Introduces new arbitrary but fixed variables \((\sim \text{parameters})\)
obtain

Syntax:

\texttt{obtain \ variables \ where \ proposition \ proof}

Introduces new variables together with property
Demo: predicate calculus
moreover/ultimately

have \( f_{\text{ormula}_1} \ldots \)

have \( f_{\text{ormula}_2} \ldots \)

\vdots

have \( f_{\text{ormula}_n} \ldots \)

show \ldots

proof \ldots
moreover/ultimately

have $\text{formula}_1$ . . .
moresover
have $\text{formula}_2$ . . .
moresover
 . . .
moresover
have $\text{formula}_n$ . . .

show . . .

proof . . .
moreover
moreover
moreover
moreover
moreover
ultimately
ultimately
ultimately
ultimately
have $formula_1$ . . .
morerover
have $formula_2$ . . .
morerover
::
morerover
have $formula_n$ . . .
ultimately
show . . .
— pipes facts $formula_1$ . . . $formula_n$ into the proof
proof . . .
Demo: moreover/ultimately
General case distinctions

show \textit{formula}

proof -

have $P_1 \lor P_2 \lor P_3 \ldots$
show \textit{formula}

proof -

have \( P_1 \lor P_2 \lor P_3 \ldots \)

moreover

\{ assume \( P_1 \ldots \) have \( \textit{thesis} \ldots \} \)
General case distinctions

show \( f_{ormula} \)

proof -

have \( P_1 \lor P_2 \lor P_3 \ldots \)

moreover

\{ assume \( P_1 \ldots \) have \( \text{thesis} \ldots \} \)

moreover

\{ assume \( P_2 \ldots \) have \( \text{thesis} \ldots \} \)
show \textit{formula}

proof -

have $P_1 \lor P_2 \lor P_3 \ldots$

moreover

\{ assume $P_1 \ldots$ have \textit{thesis} \ldots\}\n
moreover

\{ assume $P_2 \ldots$ have \textit{thesis} \ldots\}\n
moreover

\{ assume $P_3 \ldots$ have \textit{thesis} \ldots\}
General case distinctions

\[\text{show } f_{\text{ormula}}\]

proof -

have \(P_1 \lor P_2 \lor P_3 \ldots\)

moreover

\{ \text{assume } P_1 \ldots \text{have } ?\text{thesis } \ldots \}\n
moreover

\{ \text{assume } P_2 \ldots \text{have } ?\text{thesis } \ldots \}\n
moreover

\{ \text{assume } P_3 \ldots \text{have } ?\text{thesis } \ldots \}\n
ultimately show \ ?\text{thesis by blast}\n
qed
Chains of equations

- Keywords also and finally.
Chains of equations

- Keywords *also* and *finally*.
- ...: predefined schematic term variable, refers to the right hand side of the last expression.
Chains of equations

- Keywords also and finally.
- ...: predefined schematic term variable, refers to the right hand side of the last expression.
- Uses transitivity rule.
also/finally

have \( t_0 = t_1 \) \ldots 
also
have \( \ldots = t_2 \) \ldots 
also
\vdots
also
have \( \ldots = t_n \) \ldots
also/finally

have "t_0 = t_1" \ldots
also
have "\ldots = t_2" \ldots
also
\ldots
also
have "\ldots = t_n" \ldots
have \( t_0 = t_1 \) \ldots
also
have \( \ldots = t_2 \) \ldots
also
\vdots
also
have \( \ldots = t_n \) \ldots
also/finally

have "\( t_0 = t_1 \)" ... 
also have "\( ... = t_2 \)" ... 
also ... 
also have "\( ... = t_n \)" ...

\[
\begin{align*}
t_0 &= t_1 \\
t_0 &= t_2 \\
&\vdots \\
t_0 &= t_{n-1}
\end{align*}
\]
also/finally

have "\(t_0 = t_1\)" . . .
also
have "\(\ldots = t_2\)" . . .
also
\[ t_0 = t_1 \]
also
\[ t_0 = t_2 \]
also
\[ \vdots \]
also
\[ t_0 = t_{n-1} \]

finally show . . .
— pipes fact \(t_0 = t_n\) into the proof
proof
\[ \vdots \]
More about also

- Works for all combinations of $=,$ $\leq$ and $<$. 
More about also

- Works for all combinations of $=, \leq$ and $<$.  
- Uses rules declared as $\text{[trans]}$. 
More about also

- Works for all combinations of $=\,$, $\leq\,$ and $<\,$.
- Uses rules declared as $[\text{trans}]$.
- To view all combinations in Proof General: Isabelle/Isar $\rightarrow$ Show me $\rightarrow$ Transitivity rules
Demo: also/finally
Locales

Isabelle’s Module System
Isar is based on contexts

\[ \forall x. A \Rightarrow C \]

proof -

fix \( x \)

assume \( \text{Ass}: A \)

from \( \text{Ass} \) show \( C \ldots \)

qed
Isar is based on contexts

\[ \forall x. \ A \Rightarrow C \]

proof -
  fix \( x \)
  assume \( \text{Ass: } A \)
  from \( \text{Ass show } C \ldots \)
qed

x and Ass are visible inside this context
Beyond Isar contexts

Locales are extended contexts
Locales are extended contexts

- Locales are named
Locales are extended contexts

- Locales are named
- Fixed variables may have syntax
Beyond Isar contexts

Locales are extended contexts

- Locales are named
- Fixed variables may have syntax
- It is possible to add and export theorems
Locales are extended contexts

- Locales are named
- Fixed variables may have syntax
- It is possible to add and export theorems
- Locale expression: combine and modify locales
Context elements

Locales consist of context elements.
Locales consist of context elements.

fixes Parameter, with syntax
Locales consist of **context elements**.

- **fixes** Parameter, with syntax
- **assumes** Assumption
Locales consist of \textbf{context elements}.

- \textbf{fixes} Parameter, with syntax
- \textbf{assumes} Assumption
- \textbf{defines} Definition
Locales consist of context elements.

- **fixes**: Parameter, with syntax
- **assumes**: Assumption
- **defines**: Definition
- **notes**: Record a theorem
Declaring locales

locale $loc =$
  $loc1 +$
  fixes . . .
  assumes . . .
Declaring locales

locale \( loc = \)

\( loc1 + \)

fixes . . .

assumes . . .

Declares named locale \( loc \).
Declaring locales

locale \textit{loc} = \\
\textit{loc1} + \\
\text{Import} \\
\text{fixes} . . . \\
\text{assumes} . . .

Declares named locale \textit{loc}. 
Declaring locales

locale \( \textit{loc} = \textit{loc1} + \)

\hspace{1cm} \textit{fixes} \ldots \hspace{1cm} \text{Context elements}

\hspace{1cm} \textit{assumes} \ldots \hspace{1cm}

Declares \textit{named locale} \( \textit{loc}. \)
Declaring locales

Theorems may be stated relative to a named locale.

lemma (in \textit{loc}) \( P \) [simp]: \textit{proposition}

\textit{proof}
Declaring locales

Theorems may be stated relative to a named locale.

lemma (in loc) P [simp]: proposition
proof

- Adds theorem $P$ to context $loc$. 
Declaring locales

Theorems may be stated relative to a named locale.

\textbf{lemma (in} \textit{loc}) \textit{P} [simp]: \textit{proposition}

\textit{proof}

\begin{itemize}
  \item Adds theorem \textit{P} to context \textit{loc}.
  \item Theorem \textit{P} is in the simpset in context \textit{loc}.
\end{itemize}
Declaring locales

Theorems may be stated relative to a named locale.

\textbf{lemma} (in }\textit{loc} \text{) } P \text{ [simp]: proposition}

\textit{proof}

- Adds theorem }\textit{P}\text{ to context }\textit{loc}.
- Theorem }\textit{P}\text{ is in the simpset in context }\textit{loc}.
- Exported theorem }\textit{loc}.\textit{P}\text{ visible in the entire theory.
Demo: locales 1
Parameters must be consistent!

- Parameters in fixes are distinct.
Parameters must be consistent!

- Parameters in `fixes` are distinct.
- Free variables in `assumes` and `defines` occur in preceding `fixes`.
Parameters must be consistent!

- Parameters in `fixes` are distinct.
- Free variables in `assumes` and `defines` occur in preceding `fixes`.
- Defined parameters must neither occur in preceding `assumes` nor `defines`.
Locale expressions

Locale name: \( n \)
Locale expressions

Locale name: \( n \)
Rename: \( e \quad q_1 \ldots \quad q_n \)
Change names of parameters in \( e \).
Locale expressions

Locale name: \( n \)
Rename: \( e q_1 \ldots q_n \)
Change names of parameters in \( e \).

Merge: \( e_1 + e_2 \)
Context elements of \( e_1 \), then \( e_2 \).
Locale expressions

Locale name: \( n \)
Rename: \( e q_1 \ldots q_n \)
Change names of parameters in \( e \).
Merge: \( e_1 + e_2 \)
Context elements of \( e_1 \), then \( e_2 \).

▶ Syntax is lost after rename (currently).
Demo: locales 2
Normal form of locale expressions

Locale expressions are converted to flattened lists of locale names.
Normal form of locale expressions

Locale expressions are converted to flattened lists of locale names.

- With full parameter lists
Normal form of locale expressions

Locale expressions are converted to flattened lists of locale names.

- With full parameter lists
- Duplicates removed
Normal form of locale expressions

Locale expressions are converted to flattened lists of locale names.

- With full parameter lists
- Duplicates removed

Allows for multiple inheritance!
Interpretation

Move from abstract to concrete.
Interpretation

Move from abstract to concrete.

\[ \text{interpret } \text{label} : \text{loc} [t_1 \ldots t_n] \text{ proof} \]
Interpretation

Move from abstract to concrete.

\textbf{interpret} \textit{label} : \textit{loc} \left[ t_1 \ldots t_n \right] \textit{proof}

- Interpret \textit{loc} with parameters $t_1 \ldots t_n$
Interpretation

Move from abstract to concrete.

\[
\text{interpret} \ label : \ loc \ [t_1 \ldots t_n] \ proof
\]

- Interpret \( loc \) with parameters \( t_1 \ldots t_n \)
- Generates proof obligation.
**Interpretation**

Move from **abstract** to **concrete**.

\[
\text{interpret } \textit{label} : \textit{loc} \ [t_1 \ldots t_n] \ \text{proof}
\]

- Interpret \textit{loc} with parameters \( t_1 \ldots t_n \)
- Generates proof obligation.
- Imports all theorems of \textit{loc} into current context.
Interpretation

Move from abstract to concrete.

\textbf{interpret} \textit{label} : \textit{loc} \left[ t_1 \ldots t_n \right] \textit{proof}

- Interpret \textit{loc} with parameters \( t_1 \ldots t_n \).
- Generates proof obligation.
- Imports all theorems of \textit{loc} into current context.
  - Instantiates the parameters with \( t_1 \ldots t_n \).
  - Interprets attributes of theorems.
  - Prefixes theorem names with \textit{label}.
Interpretation

Move from abstract to concrete.

interpret label : loc [t₁ … tₙ] proof

- Interpret loc with parameters t₁ … tₙ
- Generates proof obligation.
- Imports all theorems of loc into current context.
  - Instantiates the parameters with t₁ … tₙ.
  - Interprets attributes of theorems.
  - Prefixes theorem names with label
- Currently only works inside Isar contexts.
Demo: locales 3
Practical Session III

The sun spills darkness
A dog howls after midnight
Goals remain unsolved.

— Chris Owens