NICTA Advanced Course

Theorem Proving
Principles, Techniques, Applications

locales
CONTENT

→ Intro & motivation, getting started with Isabelle

→ Foundations & Principles
  ● Lambda Calculus
  ● Higher Order Logic, natural deduction
  ● Term rewriting

→ Proof & Specification Techniques
  ● Inductively defined sets, rule induction
  ● Datatypes, recursion, induction
  ● More recursion, Calculational reasoning
  ● Hoare logic, proofs about programs
  ● Locales, Presentation
LAST TIME

→ Syntax and semantics of IMP
→ Hoare logic rules
→ Soundness of Hoare logic
→ Verification conditions
→ Example program proofs
Theorem $\bigwedge x. A \implies C$

Proof -

fix $x$

assume $Ass: A$

\vdots

from $Ass$ show $C$ \ldots

qed
**Theorem**  \( \bigwedge x. A \Rightarrow C \)

**Proof**

- fix \( x \)
  - assume \( \text{Ass}: A \)
  - \( \vdash \) \( x \) and \( \text{Ass} \) are visible
  - from \( \text{Ass} \) show \( C \) …
  - inside this context

**Qed**
Locales are extended contexts
Locales are extended contexts

→ Locales are named
Locales are extended contexts

- Locales are **named**
- Fixed variables may have **syntax**
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
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 **context elements**.

- **fixes**: Parameter, with syntax
- **assumes**: Assumption
- **defines**: Definition
Locales consist of **context elements**.

- **fixes** Parameter, with syntax
- **assumes** Assumption
- **defines** Definition
- **notes** Record a theorem
Locales consist of **context elements**.

- **fixes** Parameter, with syntax
- **assumes** Assumption
- **defines** Definition
- **notes** Record a theorem
- **includes** Import other locales (locale expressions)
Declaring locales

Declaring `locale` (named context) `loc`:

\[
\text{locale } \textit{loc} = \\
\]
Declaring locale (named context) loc:

```
locale loc =
  loc1 + Import
```
Declaring **locale** (named context) $loc$:

```plaintext
locale $loc =$

$loc1 +$  Import

fixes ...  Context elements

assumes ...```

---

**DECLARING LOCALES**

---

**DECLARING LOCALES**
Theorems may be stated relative to a named locale.

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

\textit{proof}
Theorems may be stated relative to a named locale.

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

\textit{proof}

→ Adds theorem \( P \) to context \textit{loc}. 
Theorems may be stated relative to a named locale.

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

**proof**

- Adds theorem $P$ to context $loc$.
- Theorem $P$ is in the simpset in context $loc$. 
DECLARING LOCALES

Theorems may be stated relative to a named locale.

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

\textit{proof}

\rightarrow \text{Adds theorem } P \text{ to context } \textit{loc}.

\rightarrow \text{Theorem } P \text{ is in the simpset in context } \textit{loc}.

\rightarrow \text{Exported theorem } \textit{loc}.P \text{ visible in the entire theory.}
DEMO: LOCALES 1
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 cannot occur in preceding `assumes` nor `defines`. 
Locale name: $n$
Locale name: \( n \)

Rename: \( e \ q_1 \ldots q_n \)

Change names of parameters in \( e \).
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
Locale expressions are converted to flattened lists of locale names.
Locale expressions are converted to flattened lists of locale names.

➢ With full parameter lists
Locale expressions are converted to flattened lists of locale names.

- With full parameter lists
- Duplicates removed
Locale expressions are converted to flattened lists of locale names.

- With full parameter lists
- Duplicates removed

Allows for **multiple inheritance**!
Move from **abstract** to **concrete**.
Instantiation

Move from abstract to concrete.

\[
\text{instantiate} \quad \text{label: loc}
\]
Move from **abstract** to **concrete**.

**Instantiate** \( \text{label: loc} \)

⇒ From chained fact \( \text{loc } t_1 \ldots t_n \) instantiate locale \( \text{loc} \).
Move from **abstract** to **concrete**.

**instantiate**  \( \text{label: } \text{loc} \)

- From chained fact \( \text{loc} \, t_1 \ldots t_n \) instantiate locale \( \text{loc} \).
- Imports all theorems of \( \text{loc} \) into current context.
Move from **abstract** to **concrete**.

**Instantiate** *label: loc*

- From chained fact *loc t₁ ... tₙ* instantiate locale *loc*.
- Imports all theorems of *loc* into current context.
  - Instantiates the parameters with *t₁ ... tₙ*.
  - Interprets attributes of theorems.
  - Prefixes theorem names with *label*
Move from **abstract** to **concrete**.

**instantiate** *label: loc*

→ From chained fact $\text{loc } t_1 \ldots t_n$ instantiate locale $\text{loc}$.

→ Imports all theorems of $\text{loc}$ into current context.
  
  • Instantiates the parameters with $t_1 \ldots t_n$.
  
  • Interprets attributes of theorems.
  
  • Prefixes theorem names with *label*

→ **Currently only works inside Isar contexts.**
DEMO: LOCALES 3
PRESENTATION
Isabelle’s Batch Mode

- used to process and check larger number of theories
ISABELLE’S BATCH MODE

➔ used to process and check larger number of theories

➔ no interactive niceties (no sorry, no quick_and_dirty)
Isabelle’s Batch Mode

→ used to process and check larger number of theories
→ no interactive niceties (no sorry, no quick_and_dirty)
→ controlled by file ROOT.ML and script set isatool
ISABELLE’S BATCH MODE

- used to process and check larger number of theories
- no interactive niceties (no sorry, no quick_and_dirty)
- controlled by file `ROOT.ML` and script set `isatool`
- can save state for later use (images)
ISABELLE’S BATCH MODE

- used to process and check larger number of theories
- no interactive niceties (no sorry, no quick_and_dirty)
- controlled by file ROOT.ML and script set isatool
- can save state for later use (images)
- can generate HTML and \LaTeX documentation
isatool <tool> <options>
ISATOOOL

isatool <tool> <options>

Get help with:

isatool                     shows available tools
isatool <tool> -?          shows options for <tool>
isatool <tool> <options>

Get help with:

- `isatool` shows available tools
- `isatool <tool> -?` shows options for `<tool>`

Interesting tools:

- `isatool mkdir` create session directory
- `make/makeall` run make for directory/all logics
- `usedir` batch session
  (documents, HTML, session graph)
- `document/latex` run \LaTeX\ for generated sources
<..>/isatool usedir -d pdf HOL <session>
<..>/<session>/ROOT.ML
<..>/<session>/MyTheory.thy
<..>/<session>/document/root.tex
Generating \LaTeX\ from Isabelle

```plaintext
<..>/isatool usedir -d pdf HOL <session>

<..>/<session>/ROOT.ML
<..>/<session>/MyTheory.thy
<..>/<session>/document/root.tex

⇒ In ROOT.ML:
no\_document use_thy "MyLibrary";
use_thy "MyTheory";
```
GENERATING LaTeX FROM ISABELLE

```latex
<..>/isatool usedir -d pdf HOL <session>
<..>/<session>/ROOT.ML
<..>/<session>/MyTheory.thy
<..>/<session>/document/root.tex
```

→ In `ROOT.ML`:
  ```plaintext
  no\_document use_thy "MyLibrary";
  use_thy "MyTheory";
  ```

→ In `document/root.tex`:
  - include Isabelle style packages (`isabelle.sty`, `isabellesym.sty`)
  - include generated files
    ```latex
    session.tex (for all theories) or
    MyTheory.tex
    ```
DEMO: EXAMPLE
Creating Images:

<..>/<session>/isatool usedir -b HOL <session>
<..>/<session>/ROOT.ML
<..>/<session>/MyLibrary.thy
Creating Images:

```sh
<..>/<session>/isatool usedir -b HOL <session>

<..>/<session>/ROOT.ML
<..>/<session>/MyLibrary.thy
```

→ Processes `ROOT.ML`
Creating Images:

<..>/<session>/isatool usedir -b HOL <session>

<..>/<session>/ROOT.ML
<..>/<session>/MyLibrary.thy

→ Processes ROOT.ML

→ Saves state after processing in
  ~/isabelle/heaps/<ML-system>/HOL-<session>
Creating Images:

```
<..>/<session>/isatool usedir -b HOL <session>
<..>/<session>/ROOT.ML
<..>/<session>/MyLibrary.thy
```

- Processes `ROOT.ML`
- Saves state after processing in `~/isabelle/heaps/<ML-system>/HOL-<session>`
- Makes `HOL-<session>` available as logic in menu Isabelle→Logics
Creating Images:

```
<..>/<session>/isatool usedir -b HOL <session>
<..>/<session>/ROOT.ML
<..>/<session>/MyLibrary.thy
```

- **Processes** `ROOT.ML`
- **Saves state after processing in** `~/isabelle/heaps/<ML-system>/HOL-<session>`
- **Makes** `HOL-<session>` **available as logic in menu Isabelle→Logics**
- **Direct start of Isabelle with new logic:**
  ```
  Isabelle -l HOL-<session>
  ```
document structure commands:
document structure commands:

header section subsection subsubsection
(meaning defined in isabelle.sty)
document structure commands:

- header
- section
- subsection
- subsubsection

(meaning defined in isabelle.sty)

normal text

- text {*...*}
- text_raw {*...*}
MARKUP COMMANDS

→ document structure commands:
  
  header  section  subsection  subsubsection
  (meaning defined in isabelle.sty)

→ normal text
  
  text  {*...*}  text_raw  {*...*}

→ text inside proofs
  
  txt  {*...*}  txt_raw  {*...*}
MARKUP COMMANDS

→ document structure commands:

    header section subsection subsubsection

(meaning defined in isabelle.sty)

→ normal text

    text {*...*} text_raw {*...*}

→ text inside proofs

    txt {*...*} txt_raw {*...*}

→ formal comments

    -- {*...*}
MARKUP COMMANDS

→ document structure commands:

header section subsection subsubsection
(meaning defined in isabelle.sty)

→ normal text

\text{\{\ldots\}} \hspace{1cm} \text{\texttt{text\_raw}} \{\ldots\}

→ text inside proofs

\texttt{txt} \{\ldots\} \hspace{1cm} \text{\texttt{txt\_raw}} \{\ldots\}

→ formal comments

\texttt{---} \{\ldots\}

→ make text invisible:

\(* < *) \ldots (\ast > \ast)\)
ANTQUOTATIONS

Inside \LaTeX you can go back to Isabelle commands and syntax.

Useful Antiquotations:
Inside \LaTeX you can go back to Isabelle commands and syntax.

Useful Antiquotations:

\begin{itemize}
\item \texttt{@\{typ \tau\}} \hspace{2cm} \text{print type } \tau
\end{itemize}
Inside \LaTeX you can go back to Isabelle commands and syntax.

Useful Antiquotations:

\begin{itemize}
  \item @{\texttt{typ } \tau} \quad \text{print type } \tau
  \item @{\texttt{term } t} \quad \text{print term } t
\end{itemize}
**ANTIQUOTATIONS**

Inside \LaTeX you can go back to Isabelle commands and syntax.

**Useful Antiquotations:**

- `@\{typ \tau\}`: print type \(\tau\)
- `@\{term \ t\}`: print term \(\!t\)
- `@\{prop \ \phi\}`: print proposition \(\phi\)
- `@\{prop [display] \ \phi\}`: print proposition \(\phi\) with linebreaks
- `@\{prop [source] \ \phi\}`: check proposition \(\phi\), print its input
Inside \LaTeX you can go back to Isabelle commands and syntax.

Useful Antiquotations:

@\{typ \, \tau\} \quad \text{print type } \tau

@\{term \, t\} \quad \text{print term } t

@\{prop \, \phi\} \quad \text{print proposition } \phi

@\{prop \,[\text{display}] \, \phi\} \quad \text{print proposition } \phi \text{ with linebreaks}

@\{prop \,[\text{source}] \, \phi\} \quad \text{check proposition } \phi, \text{ print its input}

@\{thm \, a\} \quad \text{print fact } a

@\{thm \, a \,[\text{no_vars}]\} \quad \text{print fact } a, \text{ fixing schematic variables}

@\{thm \,[\text{source}] \, a\} \quad \text{check availability of } a, \text{ print its name}
Inside \LaTeX you can go back to Isabelle commands and syntax.

Useful Antiquotations:

@{typ \tau} \quad \text{print type } \tau

@{term t} \quad \text{print term } t

@{prop \phi} \quad \text{print proposition } \phi

@{prop [display] \phi} \quad \text{print proposition } \phi \text{ with linebreaks}

@{prop [source] \phi} \quad \text{check proposition } \phi, \text{ print its input}

@{thm a} \quad \text{print fact } a

@{thm a [no vars]} \quad \text{print fact } a, \text{ fixing schematic variables}

@{thm [source] a} \quad \text{check availability of } a, \text{ print its name}

@{text s} \quad \text{print uninterpreted text } s
To document definitions and proofs:

- put comments explanations directly in original theory
- keep explanations short and to the point
- formal definitions, lemmas, syntax should speak for themself
To document definitions and proofs:

- put comments explanations directly in original theory
- keep explanations short and to the point
- formal definitions, lemmas, syntax should speak for themselves

To write a paper/thesis about a formal development

- use a separate theory/document on top of the development
WRITING ABOUT ISABELLE THEORIES

To document definitions and proofs:

➤ put comments explanations directly in original theory
➤ keep explanations short and to the point
➤ formal definitions, lemmas, syntax should speak for themself

To write a paper/thesis about a formal development

➤ use a separate theory/document on top of the development
➤ only talk about the interesting parts
WRITING ABOUT ISABELLE THEORIES

To document definitions and proofs:

- put comments explanations directly in original theory
- keep explanations short and to the point
- formal definitions, lemmas, syntax should speak for themself

To write a paper/thesis about a formal development

- use a separate theory/document on top of the development
- only talk about the interesting parts
- use antiquoations for theorems and definitions
WRITING ABOUT ISABELLE THEORIES

To document definitions and proofs:

➡ put comments explanations directly in original theory
➡ keep explanations short and to the point
➡ formal definitions, lemmas, syntax should speak for themself

To write a paper/thesis about a formal development

➡ use a separate theory/document on top of the development
➡ only talk about the interesting parts
➡ use antiquoations for theorems and definitions
➡ use extra locales, definitions, syntax for polish
To document definitions and proofs:

- put comments explanations directly in original theory
- keep explanations short and to the point
- formal definitions, lemmas, syntax should speak for themself

To write a paper/thesis about a formal development

- use a separate theory/document on top of the development
- only talk about the interesting parts
- use antiquoations for theorems and definitions
- use extra locales, definitions, syntax for polish
- make full proof document available separately
Know your audience. Use the right notation.
Polish

Know your audience. Use the right notation.

→ Change \LaTeX symbol interpretations

\renewcommand{\isasymLongrightarrow}{\isamath{\longrightarrow}}

→ Declare special \LaTeX outputsyntax:
syntax (latex)Cons::"'a 'alist 'alist"

→ Use translations to change outputsyntax:
syntax (latex)notEx::"('a bool) (bool)" (binder "\notex")
translations "\notex" x.P "\notex" x.P =: (9 x.P)

\newcommand{\isasymnotex}{\isamath{\neg\exists}}
Know your audience. Use the right notation.

➜ Change \LaTeX\ symbol interpretations
\renewcommand{\isasymLongrightarrow}{\isamath{\longrightarrow}}

➜ Declare special \LaTeX\ output syntax:
\texttt{syntax (latex)} \ Cons :: \”a ⇒ ’a list ⇒ ’a list” (”_·_/” [66,65] 65)
Know your audience. Use the right notation.

- Change LaTeX symbol interpretations
  \renewcommand{\isasymLongrightarrow}{\isamath{\longrightarrow}}

- Declare special LaTeX output syntax:
  \texttt{syntax (latex)  Cons :: "'a \Rightarrow 'a list \Rightarrow 'a list" ("\cdot/\cdot" [66,65] 65)}

- Use translations to change output syntax:
  \texttt{syntax (latex)  notEx :: "'(a \Rightarrow bool) \Rightarrow bool" (binder "\<notex>" 10)}
  \texttt{translations  "\<notex>x. P"  \iff  "\neg(\exists x. P)"}

  in document/root.tex:
  \newcommand{\isasymnotex}{\isamath{\neg\exists}}
USING LOCALES

making large developments more accessible
USING LOCALES

making large developments more accessible

Math textbook:

Let \((A, \cdot, 0)\) in the following be a group with \(x \cdot y = y \cdot x\)
USING LOCALES

making large developments more accessible

Math textbook:

Let \((A, \cdot, 0)\) in the following be a group with \(x \cdot y = y \cdot x\)

Isabelle:

→ Use locales to formalize contexts
USING LOCALES

making large developments more accessible

Math textbook:

Let \((A, \cdot, 0)\) in the following be a group with \(x \cdot y = y \cdot x\)

Isabelle:

- Use locales to formalize contexts
- Antiquotations are sensitive to current locale context
USING LOCALES

making large developments more accessible

Math textbook:

Let \((A, \cdot, 0)\) in the following be a group with \(x \cdot y = y \cdot x\)

Isabelle:

→ Use locales to formalize contexts

→ Antiquotations are sensitive to current locale context

→ Example:
   locale agroup = group + assumes com: "\(x \cdot y = y \cdot x\)"
   
   \(\ldots\)
   
   \((\ast < \ast)\) lemma (in agroup) True (\(\ast > \ast\))
   
   txt \(\{\ast \ldots \ast\}\)
   
   \((\ast < \ast)\) oops (\(\ast > \ast\))
DEMO
We have seen today ...

- Locale Declarations + Theorems in Locales
- Locale Expressions + Inheritance
- Locale Instantiation
- Generating \LaTeX
- Writing a thesis/paper in Isabelle
No Exercise Today
No Exercise Today

Theorem Proving
Principles, Techniques, Applications

The End