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

Isar Is Based On Contexts

theorem $\bigwedge x.\ A \Longrightarrow C$

proof -

fix x

assume Ass: A

$\vdots$

from Ass **show** C . . .

qed

Isar Is Based On Contexts

theorem $\bigwedge x.\ A \Longrightarrow C$

proof -

fix x	
assume Ass: A	
$\vdots$	x and Ass are visible
from Ass **show** C ...	inside this context

qed

Beyond Isar Contexts

Locales are extended contexts

Beyond Isar Contexts

Locales are extended contexts

➜ Locales are **named**

Beyond Isar Contexts

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

Beyond Isar Contexts

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**.

Context Elements

Locales consist of **context elements**.

fixes Parameter, with syntax

Context Elements

Locales consist of **context elements**.

fixes	Parameter, with syntax
assumes	Assumption

Context Elements

Locales consist of **context elements**.

fixes	Parameter, with syntax
assumes	Assumption
defines	Definition

Context Elements

Locales consist of **context elements**.

fixes	Parameter, with syntax
assumes	Assumption
defines	Definition
notes	Record a theorem

Context Elements

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:

locale loc =

Declaring Locales

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

locale loc =

$loc1$ + Import

Declaring Locales

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

locale loc =

$loc1$ + Import

fixes . . . Context elements

assumes . . .

Declaring Locales

Theorems may be stated relative to a named locale.

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

$proof$

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.

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.

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

$proof$

- ➜ Adds theorem P to context loc.
- ➜ Theorem P is in the simpset in context loc.
- ➜ Exported theorem $loc.P$ 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 cannot occur in preceding **assumes** nor **defines**.

Locale Expressions

Locale name: n

Locale Expressions

Locale name: n

Rename: $e\ q_1 \ldots q_n$

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**!

Instantiation

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

INSTANTIATION

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

instantiate $label$**:** loc

INSTANTIATION

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

instantiate $label$**:** loc

➜ From chained fact $loc\, t_1 \ldots t_n$ instantiate locale loc.

INSTANTIATION

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

instantiate $label$**:** loc

➜ From chained fact $loc\, t_1 \ldots t_n$ instantiate locale loc.

➜ Imports all theorems of loc into current context.

INSTANTIATION

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

instantiate $label$**:** loc

➜ From chained fact $loc\, t_1 \ldots t_n$ instantiate locale loc.

➜ Imports all theorems of loc into current context.

- Instantiates the parameters with $t_1 \ldots t_n$.
- Interprets attributes of theorems.
- Prefixes theorem names with $label$

INSTANTIATION

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

instantiate $label$**:** loc

➜ From chained fact $loc\, t_1 \ldots t_n$ instantiate locale loc.

➜ Imports all theorems of 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

```
isatool <tool> <options>
```

ISATOOL

```
isatool <tool> <options>
```

Get help with:

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

ISATOOL

```
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

Generating LaTeX from Isabelle

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

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

Generating LaTeX from Isabelle

```
<..>/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

```
<..>/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";
```

➜ In `document/root.tex`:

- include Isabelle style packages (isabelle.sty, isabellesym.sty)
- include generated files
  `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:

```
<..>/<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>`

MARKUP COMMANDS

➜ document structure commands:

MARKUP COMMANDS

➜ document structure commands:

header section subsection subsubsection

(meaning defined in `isabelle.sty`)

MARKUP COMMANDS

➜ 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 {*...*} **text_raw** {*...*}

➜ text inside proofs

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

➜ formal comments

-- {*...*}

➜ make text invisible:

(* < *) ... (* > *)

ANTIQUOTATIONS

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

Useful Antiquotations:

ANTIQUOTATIONS

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

Useful Antiquotations:

`@{typ` τ`}`	print type τ

ANTIQUOTATIONS

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

Useful Antiquotations:

`@{typ` τ`}`	print type τ
`@{term` t`}`	print term t

ANTIQUOTATIONS

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

Useful Antiquotations:

`@{typ` τ`}`	print type τ
`@{term` t`}`	print term t
`@{prop` ϕ`}`	print proposition ϕ
`@{prop [display]` ϕ`}`	print proposition ϕ with linebreaks
`@{prop [source]` ϕ`}`	check proposition ϕ, print its input

ANTIQUOTATIONS

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

Useful Antiquotations:

`@{typ` τ`}`	print type τ
`@{term` t`}`	print term t
`@{prop` ϕ`}`	print proposition ϕ
`@{prop [display]` ϕ`}`	print proposition ϕ with linebreaks
`@{prop [source]` ϕ`}`	check proposition ϕ, print its input
`@{thm` a`}`	print fact a
`@{thm` a `[no_vars]}`	print fact a, fixing schematic variables
`@{thm [source]` a`}`	check availability of a, print its name

ANTIQUOTATIONS

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

Useful Antiquotations:

`@{typ` τ`}`	print type τ
`@{term` t`}`	print term t
`@{prop` ϕ`}`	print proposition ϕ
`@{prop [display]` ϕ`}`	print proposition ϕ with linebreaks
`@{prop [source]` ϕ`}`	check proposition ϕ, print its input
`@{thm` a`}`	print fact a
`@{thm` a `[no_vars]}`	print fact a, fixing schematic variables
`@{thm [source]` a`}`	check availability of a, print its name
`@{text` s`}`	print uninterpreted text s

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

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

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

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
- ➜ 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}}
```

Know your audience. Use the right notation.

- Change LaTeX symbol interpretations

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

- Declare special LaTeX output syntax:

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:

  syntax (latex) Cons :: "'a ⇒ 'a list ⇒ 'a list" ("_ ·/ _" [66,65] 65)

- Use translations to change output syntax:

  syntax (latex) notEx :: "('a ⇒ bool) ⇒ bool" (binder "\<notex>" 10)

  translations "\<notex>x. P" <= "¬(∃x. P)"

  in document/root.tex:

```
\newcommand{\isasymnotex}{\isamath{\neg\exists}}
```

USING LOCALES

making large developments more accessible

making large developments more accessible

Math textbook:

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

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

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

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: "$x \cdot y = y \cdot x$"
  ...
  $(* < *)$ **lemma** (in agroup) True $(* > *)$
  txt $\{* \ldots *\}$
  $(* < *)$ **oops** $(* > *)$

DEMO

We have seen today ...

- ➜ Locale Declarations + Theorems in Locales
- ➜ Locale Expressions + Inheritance
- ➜ Locale Instantiation
- ➜ Generating LaTeX
- ➜ Writing a thesis/paper in Isabelle

EXERCISES

➜ No Exercise Today

EXERCISES

➜ No Exercise Today

Theorem Proving
Principles, Techniques, Applications

The End