Automatic Deduction — LVA 703522

Introduction to Isabelle

Clemens Ballarin

Universität Innsbruck

{}

Contents

- Intro & motivation, getting started with Isabelle
- Foundations & Principles
  - Lambda Calculus
  - Types & Classes
  - Natural Deduction
  - Term rewriting
- Proof & Specification Techniques
  - Isar: mathematics style proofs
  - Inductively defined sets, rule induction
  - Datatypes, structural induction
  - Recursive functions & code generation

Sets

Sets in Isabelle

Type 'a set: sets over type 'a

- $\{\}$, $\{e_1, \ldots, e_n\}$, $\{x.\ P\ x\}$
- $e \in A$, $A \subseteq B$
- $A \cup B$, $A \cap B$, $A - B$, UNIV, $-A$
- $\bigcup x \in A.\ B\ x$, $\bigcap x \in A.\ B\ x$, $\bigcup\ A$, $\bigcap A$
- $\{i..j\}$
- insert :: $\alpha \Rightarrow \alpha$ set $\Rightarrow \alpha$ set
- $f`A \equiv \{y.\ \exists x \in A.\ y = f\ x\}$
- ...

Proofs about Sets

Natural deduction proofs:

- equalityI: $[\![A \subseteq B;\ B \subseteq A]\!] \Longrightarrow A = B$
- subsetI: $(\bigwedge x.\ x \in A \Longrightarrow x \in B) \Longrightarrow A \subseteq B$
- IntI: $[\![x \in A; x \in B]\!] \Longrightarrow x \in A \cap B$
- IntE: $[\![x \in A \cap B; [\![x \in A; x \in B]\!] \Longrightarrow P]\!] \Longrightarrow P$
- ... (see LNCS 2283)

Bounded Quantifiers

- $\forall x \in A.\ P\ x \equiv \forall x.\ x \in A \longrightarrow P\ x$
- $\exists x \in A.\ P\ x \equiv \exists x.\ x \in A \wedge P\ x$
- ballI: $(\bigwedge x.\ x \in A \Longrightarrow P\ x) \Longrightarrow \forall x \in A.\ P\ x$
- bspec: $[\![\forall x \in A.\ P\ x; x \in A]\!] \Longrightarrow P\ x$
- bexI: $[\![P\ x; x \in A]\!] \Longrightarrow \exists x \in A.\ P\ x$
- bexE: $[\![\exists x \in A.\ P\ x; \bigwedge x.\ [\![x \in A; P\ x]\!] \Longrightarrow Q]\!] \Longrightarrow Q$

Demo: Sets

Inductive Definitions

Example

$$\frac{}{\langle \mathsf{skip}, \sigma\rangle \longrightarrow \sigma} \qquad \frac{[\![e]\!]\sigma = v}{\langle \mathsf{x} := \mathsf{e}, \sigma\rangle \longrightarrow \sigma[x \mapsto v]}$$

$$\frac{\langle c_1, \sigma\rangle \longrightarrow \sigma' \qquad \langle c_2, \sigma'\rangle \longrightarrow \sigma''}{\langle c_1; c_2, \sigma\rangle \longrightarrow \sigma''}$$

$$\frac{[\![b]\!]\sigma = \mathsf{False}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \longrightarrow \sigma}$$

$$\frac{[\![b]\!]\sigma = \mathsf{True} \qquad \langle c, \sigma\rangle \longrightarrow \sigma' \qquad \langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma'\rangle \longrightarrow \sigma''}{\langle \mathsf{while}\ b\ \mathsf{do}\ c, \sigma\rangle \longrightarrow \sigma''}$$

What Does This Mean?

- $\langle c, \sigma \rangle \longrightarrow \sigma'$ fancy syntax for a relation $(c, \sigma, \sigma') \in E$
- Relations are sets: E :: (com $\times$ state $\times$ state) set
- The rules define a set inductively.

But which set?

Simpler Example

$$\frac{}{0 \in N} \qquad \frac{n \in N}{n + 1 \in N}$$

- N is the set of natural numbers $\mathbb{N}$.
- But why not the set of real numbers?
  $0 \in \mathbb{R}$, $n \in \mathbb{R} \Longrightarrow n + 1 \in \mathbb{R}$
- $\mathbb{N}$ is the smallest set that is consistent with the rules.

Why the smallest set?

- No junk. Only what must be in N shall be in N.
- Gives rise to a nice proof principle: rule induction.
- Greatest set occasionally also useful: coinduction.

Formally

Set of rules $R \subseteq \mathcal{P}(A) \times A$

$$\frac{a_1 \in X \qquad \cdots \qquad a_n \in X}{a \in X}$$

with $a_1, \ldots, a_n, a \in A$ defines set $X \subseteq A$.

Applying rules R

$$\hat{R}\ B \equiv \{x.\ \exists H.\ (H, x) \in R \wedge H \subseteq B\}$$

Example

$$\begin{array}{lcl} R & \equiv & \{(\{\}, 0)\} \cup \{(\{n\}, n+1).\ n \in \mathbb{R}\} \\ \hat{R}\ \{3, 6, 10\} & = & \{0, 4, 7, 11\} \end{array}$$

The Set

Definition B is R-closed iff $\hat{R}\ B \subseteq B$

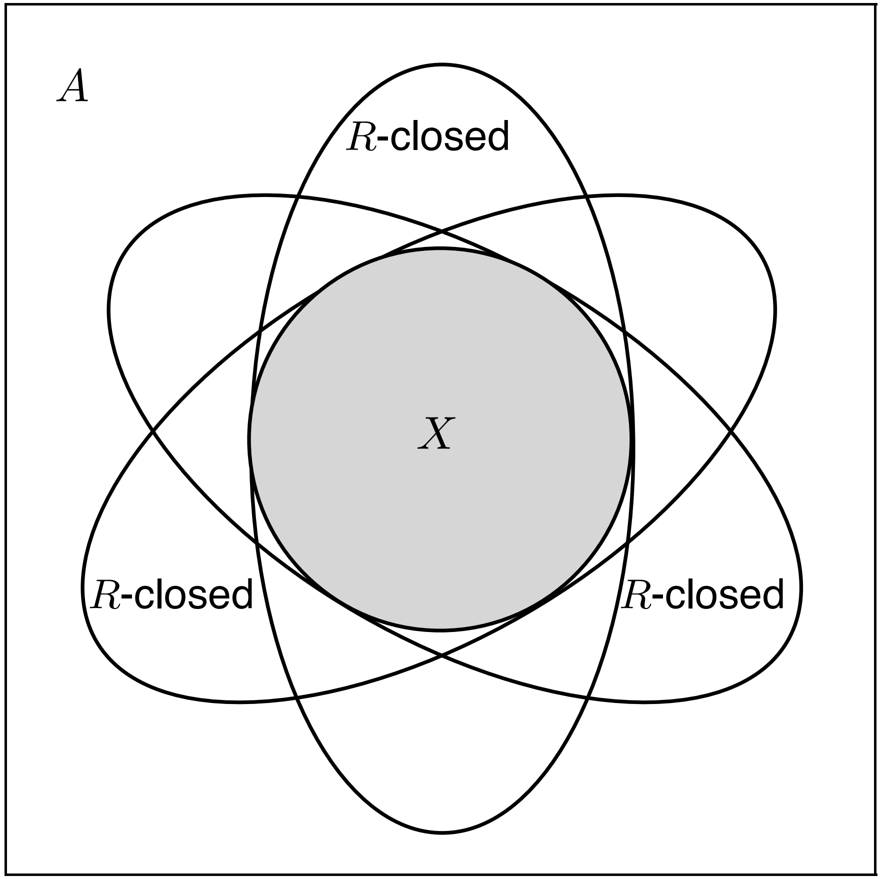
Definition X is the least R-closed subset of A

This does always exist:

Fact B_1 R-closed $\wedge$ B_2 R-closed $\Longrightarrow$ $B_1 \cap B_2$ R-closed

Hence $X = \bigcap\{B \subseteq A.\ B\ R\text{-closed}\}$

Generation from Above



Rule Induction

The rules

$$\frac{}{0 \in N} \qquad \frac{n \in N}{n + 1 \in N}$$

induce the induction principle

$$[\![P\ 0;\ \bigwedge n.\ P\ n \Longrightarrow P\ (n+1)]\!] \Longrightarrow \forall x \in X.\ P\ x$$

In general

$$\frac{\forall (\{a_1, \ldots a_n\}, a) \in R.\ P\, a_1 \wedge \ldots \wedge P\, a_n \Longrightarrow P\, a}{\forall x \in X.\ P\, x}$$

Why Does This Work?

$$\frac{\forall(\{a_1,\ldots a_n\},a) \in R.\; P\,a_1 \wedge \ldots \wedge P\,a_n \Longrightarrow P\,a}{\forall x \in X.\; P\,x}$$

Proof

$$\forall(\{a_1,\ldots a_n\},a) \in R.\; P\,a_1 \wedge \ldots \wedge P\,a_n \Longrightarrow P\,a$$

says $\{x.\; P\; x\}$ is R-closed

but	X is the least R-closed set
hence	$X \subseteq \{x.\; P\; x\}$
which means	$\forall x \in X.\; P\; x$ qed.

Rules with Side Conditions

$$\frac{a_1 \in X \qquad \ldots \qquad a_n \in X \qquad\qquad C_1 \qquad \ldots \qquad C_m}{a \in X}$$

Induction scheme

$$\begin{aligned}(\forall (\{a_1, \ldots a_n\}, a) \in R.\ & P\ a_1 \wedge \ldots \wedge P\ a_n \wedge \\ & C_1 \wedge \ldots \wedge C_m \wedge \\ & \{a_1, \ldots, a_n\} \subseteq X \Longrightarrow P\ a)\end{aligned}$$

$$\Longrightarrow$$

$$\forall x \in X.\ P\ x$$

X as Fixed Point

How to compute X?

$X = \bigcap\{B \subseteq A.\ B\ R\text{-closed}\}$ hard to work with.

Instead

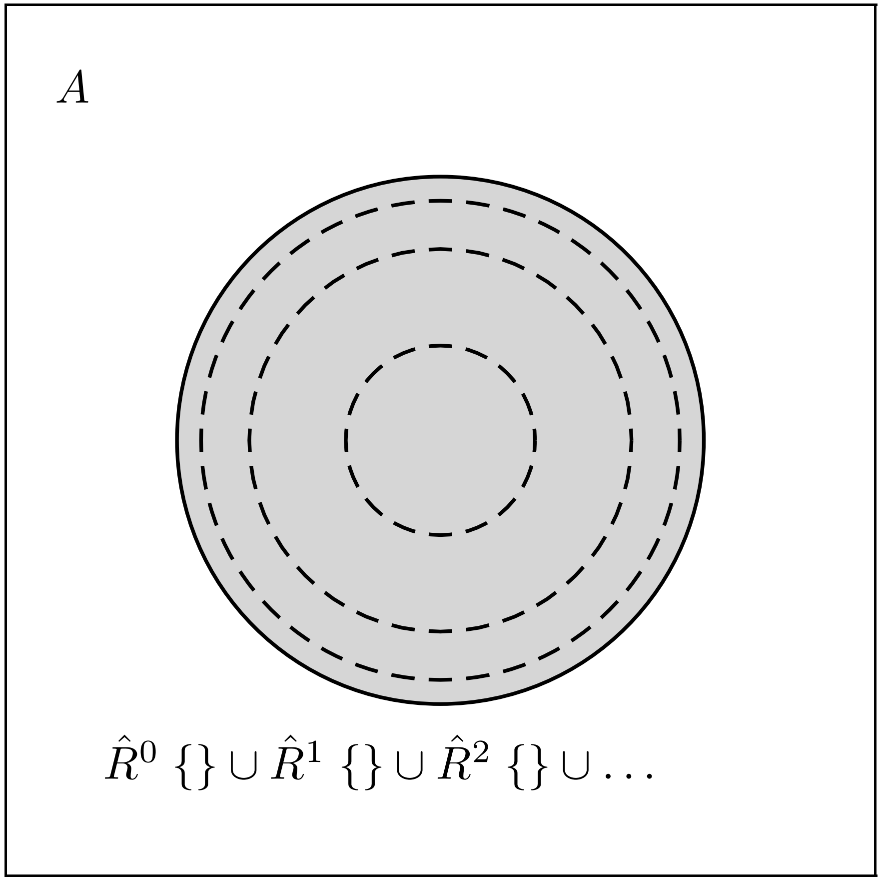
View X as least fixed point, X least set with $\hat{R}\ X = X$.

Fixed points can be approximated by iteration

$$
\begin{aligned}
X_0 &= \hat{R}^0\ \{\} = \{\} \\
X_1 &= \hat{R}^1\ \{\} = \text{rules without hypotheses} \\
&\vdots \\
X_n &= \hat{R}^n\ \{\}
\end{aligned}
$$

$$X_\omega = \bigcup_{n \in \mathbb{N}} (R^n\ \{\}) = X$$

Generation from Below



Demo: Inductive Definitions

Inductive Definitions in Isabelle

Inductive Set

inductive_set S [:: "τ"]
where

rule$_1$: "$[\![x_1 \in S; \ldots ; x_k \in S; C_1; \ldots ; C_m]\!] \Longrightarrow t \in S$"
| rule$_2$: $\ldots$
$\vdots$
| rule$_n$: $\ldots$

Inductive Predicate

inductive P [:: "τ"]
where

rule$_1$: "$[\![P\,x_1; \ldots ; P\,x_k; C_1; \ldots ; C_m]\!] \Longrightarrow P\,t$" $\ldots$

Inductive Definitions in Isabelle

inductive_set S
where rule$_1$: ... $\cdots$ | rule$_n$: ...

Proved Theorems

- Introduction rules
  S.rule$_1$ ... S.rule$_n$ S.intros
- Elimination rule (case analysis)
  S.cases
- Induction rules
  S.induct (default) S.inducts (for mutual induction)

Case Analysis

inductive_set S
where emptyI: "$\{\} \in S$" | insertI: "$A \in S \Longrightarrow \text{insert}\, a\, A \in S$"

Proof Schema

fix x **assume** "$x \in S$" **then show** "$P\,x$"

```
proof cases
  assume "x = {}"
  show "P x" ⟨proof⟩
next
  fix a and A
  assume "x = insert a A"
    and "A ∈ S"
  show "P x" ⟨proof⟩
qed
```

```
proof cases
  case emptyI
  show ?thesis ⟨proof⟩
next
  case (insertI A a)
  show ?thesis ⟨proof⟩
qed
```

Rule Induction

inductive_set S
where emptyI: "$\{\} \in S$" | insertI: "$A \in S \Longrightarrow \text{insert}\, a\, A \in S$"

Proof Schema

fix x **assume** "$x \in S$" **then show** "$P\, x$"

```
proof induct
  show "P {}" ⟨proof⟩
next
  fix a and A
  assume "A ∈ S"
    and "P A"
  show "P (insert a A)" ⟨proof⟩
qed
```

```
proof induct
  case emptyI
  show ?case ⟨proof⟩
next
  case (insertI A a)
  show ?case ⟨proof⟩
qed
```

State variables A a in order of occurrence in the rule.

How the Rule is Selected

Cases

- no information — classical case split $P \vee \neg P$
- cases "t" — datatype exhaustion (type of t)
- chained fact $t \in S$ — elimination of inductive set S
- cases rule: R — explicit selection of rule

Induct

- induct x — datatype induction (type of induction variable)
  Multiple induction variables for mutual induction.
- chained fact $x \in S$ — rule induction for S
- induct rule: R — explicit selection of rule

A Remark on Style

- **case** (rule$_i$ x y) . . . **show** ?case
  is easy to write and maintain
- **fix** x y **assume** ⟨*formula*⟩ . . . **show** ⟨*formula*⟩
  is easier to read:
  - All information is shown locally
  - No contextual references (e.g. ?case)

Demo: Rule Induction in Isar

Induction on Structured Statements

Generalising Variables

To generalise variables $x_1, \ldots, x_n$ in the induction hypothesis:

(induct arbitrary: $x_1, \ldots, x_n$)

Pushing Assumptions

To push assumptions A$_1$, . . . , A$_m$ into the induction use chaining:

from R **and** A$_1$ . . . A$_m$ **show** P **proof** induct ⟨*proof*⟩

or

show P **using** R **and** A$_1$ . . . A$_m$ **proof** induct ⟨*proof*⟩

We Have Seen so far . . .

- Sets in Isabelle
- Inductive Definitions
- Rule induction
- Fixed points
- Case analysis and induction in Isar