COMP4011/8011
Advanced Topics in
Formal Methods and Programming Languages

– Software Verification with Isabelle/HOL –

Peter Höfner

August 26, 2024

Specification Techniques

Section 10

Sets, Types & Rule Induction

Sets in Isabelle

Type **'a set**: sets over type 'a

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

Proofs about Sets

Natural deduction proofs:

- equalityI: $[\![A \subseteq B;\ B \subseteq A]\!] \Longrightarrow A = B$
- subsetI: $(\bigwedge x.\ x \in A \Longrightarrow x \in B) \Longrightarrow A \subseteq B$
- … **find_theorems**

Australian National University

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

The Three Basic Ways of Introducing Theorems

- **Axioms**:

  Example: **axiomatization where** refl: "$t = t$"

  Do not use. Evil. Can make your logic inconsistent.

- **Definitions:**

  Example: **definition** inj **where**
  "inj $f \equiv \forall x\ y.\ f\ x = f\ y \longrightarrow x = y$"
  Introduces a new lemma called inj_def.

- **Proofs:**

  Example: **lemma** "inj $(\lambda x.\ x + 1)$"

  The harder, but safe choice.

The Three Basic Ways of Introducing Types

- **typedecl**: by name only

  Example: **typedecl** names
  Introduces new type *names* without any further assumptions

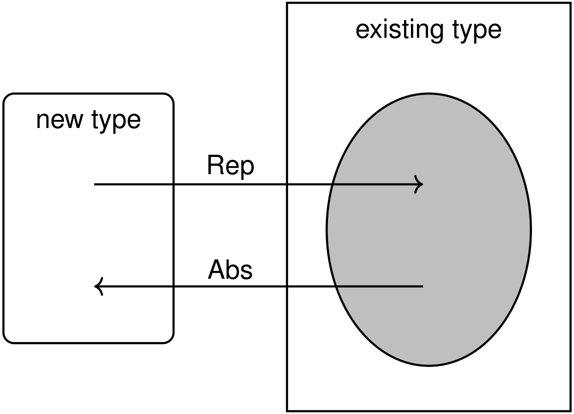
- **type_synonym**: by abbreviation

  Example: **type_synonym** α rel = "$\alpha \Rightarrow \alpha \Rightarrow$ *bool*"
  Introduces abbreviation *rel* for existing type $\alpha \Rightarrow \alpha \Rightarrow$ *bool*
  Type abbreviations are immediately expanded internally

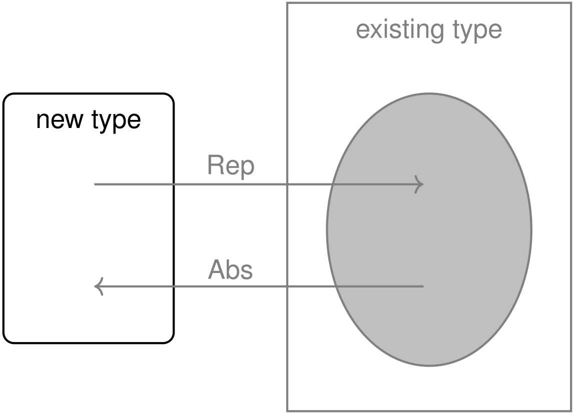
- **typedef**: by definiton as a set

  Example: **typedef** new_type = "{some set}" <proof>
  Introduces a new type as a subset of an existing type.
  The proof shows that the set on the rhs in non-empty.

How typedef works



How typedef works



Example: Pairs

$$(\alpha, \beta)\ \mathsf{Prod}$$

1. Pick existing type: $\alpha \Rightarrow \beta \Rightarrow \mathsf{bool}$
2. Identify subset:
   (α, β) Prod $= \{f.\ \exists a\ b.\ f = \lambda(x :: \alpha)\ (y :: \beta).\ x = a \wedge y = b\}$
3. We get from Isabelle:
   - functions Abs_Prod, Rep_Prod
   - both injective
   - Abs_Prod (Rep_Prod x) $= x$
4. We now can:
   - define constants Pair, fst, snd in terms of Abs_Prod and Rep_Prod
   - derive all characteristic theorems
   - forget about Rep/Abs, use characteristic theorems instead

Demo: Introducing new Types

Inductive Definitions

Example

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

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

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

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

What does this mean?

- $\langle c, \sigma \rangle \longrightarrow \sigma'$ fancy syntax for a relation $(c, \sigma, \sigma') \in E$
- relations are sets: $E :: (\mathsf{com} \times \mathsf{state} \times \mathsf{state})\ \mathsf{set}$
- 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? $0 \in \mathbb{R}$, $n \in \mathbb{R} \implies n + 1 \in \mathbb{R}$
- $\mathbb{N}$ is the smallest set that is consistent with the rules.

Why the smallest set?

- Objective: no junk. Only what must be in X shall be in X.
- Gives rise to a nice proof principle (rule induction)
- Alternative (greatest set) occasionally also useful: coinduction

Rule Induction

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

induces induction principle

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

Demo: Inductive Definitions

Formally

$$\text{Rules } \frac{a_1 \in X \quad \dots \quad a_n \in X}{a \in X} \text{ with } a_1, \dots, a_n, a \in A$$

define set $X \subseteq A$

Formally: set of rules $R \subseteq A \text{ set} \times A$ (R, X possibly infinite)

Applying rules R to a set B: $\hat{R}\ B \equiv \{x.\ \exists H.\ (H, x) \in R \wedge H \subseteq B\}$

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

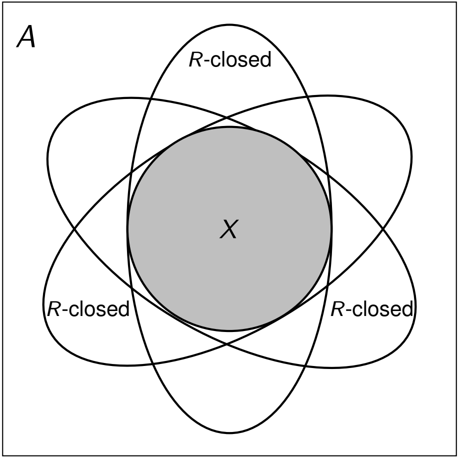
Definition: B is R-closed iff $\hat{R}\ B \subseteq B$

Definition: X is the least R-closed subset of A

This does always exist:

Fact: $X = \bigcap\{B \subseteq A.\ B\ R\text{−closed}\}$

Generation from Above



Rule Induction

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

induces induction principle

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

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?

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

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

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

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

induction scheme:

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

$$\Longrightarrow$$

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

X as Fixpoint

How to compute X?

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

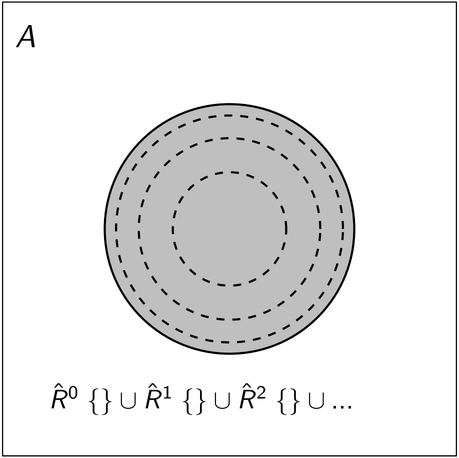
Instead: view X as least fixpoint, X least set with $\hat{R}\ X = X$.

Fixpoints can be approximated by iteration:

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

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

Generation from Below



Does this always work?

Knaster-Tarski Fixpoint Theorem:
Let $(A, \leq)$ be a complete lattice, and $f :: A \Rightarrow A$ a monotone function. Then the fixpoints of f again form a complete lattice.

Lattice:
Finite subsets have a greatest lower bound (meet) and least upper bound (join).

Complete Lattice:
All subsets have a greatest lower bound and least upper bound.

Implications:

- least and greatest fixpoints exist (complete lattice always non-empty).
- can be reached by (possibly infinite) iteration. (Why?)

Exercise

Formalise this lecture in Isabelle:

- Define **closed** $f\ A :: (\alpha\ \text{set} \Rightarrow \alpha\ \text{set}) \Rightarrow \alpha\ \text{set} \Rightarrow \text{bool}$
- Show closed $f\ A \wedge$ closed $f\ B \Longrightarrow$ closed $f\ (A \cap B)$ if f is monotone (**mono** is predefined)
- Define **lfpt** f as the intersection of all f-closed sets
- Show that lfpt f is a fixpoint of f if f is monotone
- Show that lfpt f is the least fixpoint of f
- Declare a constant $R :: (\alpha\ \text{set} \times \alpha)\ \text{set}$
- Define $\hat{R} :: \alpha\ \text{set} \Rightarrow \alpha\ \text{set}$ in terms of R
- Show soundness of rule induction using R and lfpt $\hat{R}$

We have learned ...

- Formal background of inductive definitions
- Definition by intersection
- Computation by iteration
- Formalisation in Isabelle