COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Toby Murray, June Andronick, Gerwin Klein

{ }

Content

➜ Intro & motivation, getting started [1]

➜ Foundations & Principles

- Lambda Calculus, natural deduction [1,2]
- Higher Order Logic [3[a]]
- Term rewriting [4]

➜ Proof & Specification Techniques

- Inductively defined sets, rule induction [5]
- Datatypes, recursion, induction [6, 7]
- Hoare logic, proofs about programs, C verification [8[b],9]
- (mid-semester break)
- Writing Automated Proof Methods [10]
- Isar, codegen, typeclasses, locales [11[c],12]

[a] a1 due; [b] a2 due; [c] a3 due

Last Time

- ➜ Conditional term rewriting
- ➜ Case Splitting with the simplifier
- ➜ Congruence rules
- ➜ AC Rules
- ➜ Knuth-Bendix Completion (Waldmeister)
- ➜ Orthogonal Rewrite Systems

Specification Techniques: Sets

Sets in Isabelle

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

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

Proofs about Sets

Natural deduction proofs:

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

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: **axioms** 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

NICTA

- **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

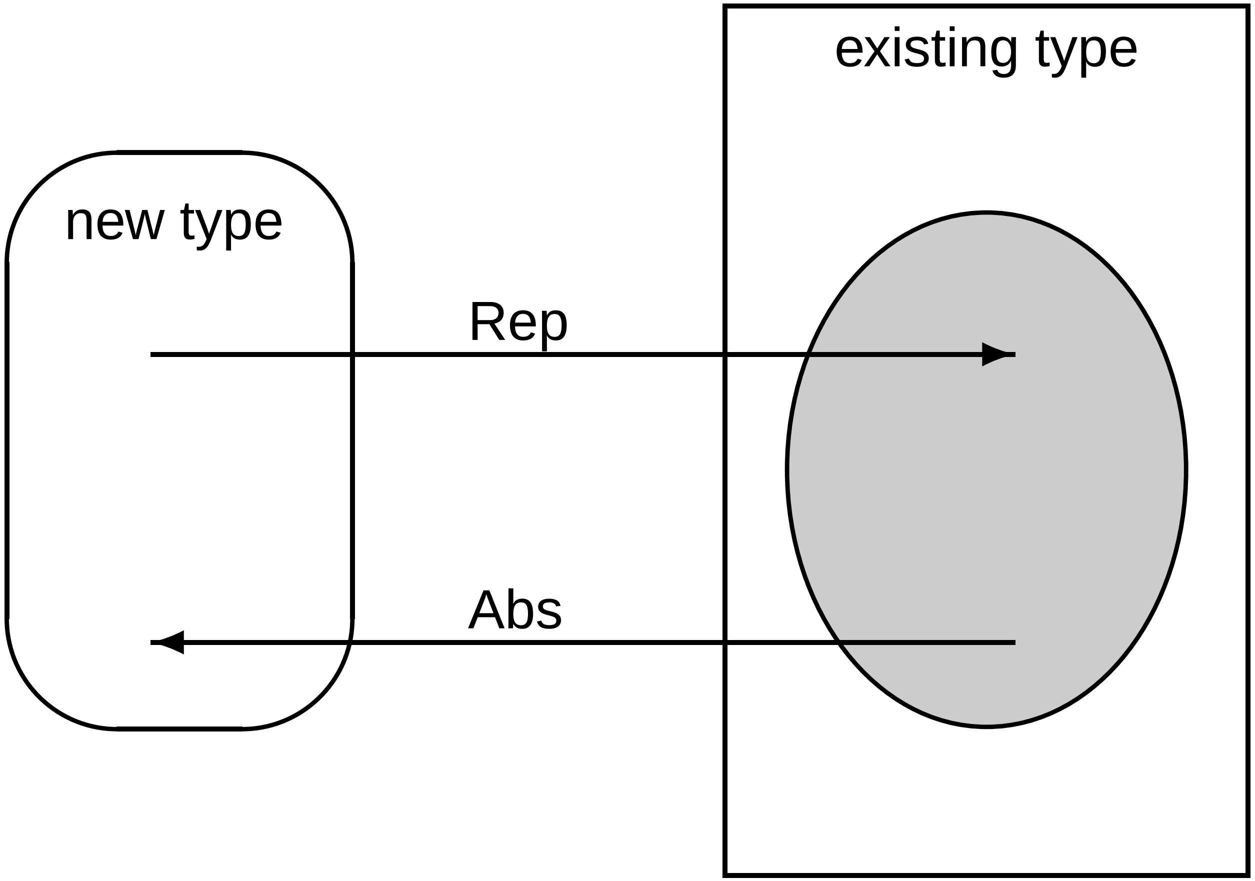
- **typedef**: by definiton as a set

  Example: **typedef** new_type = "$\{$some set$\}$" $<$proof$>$

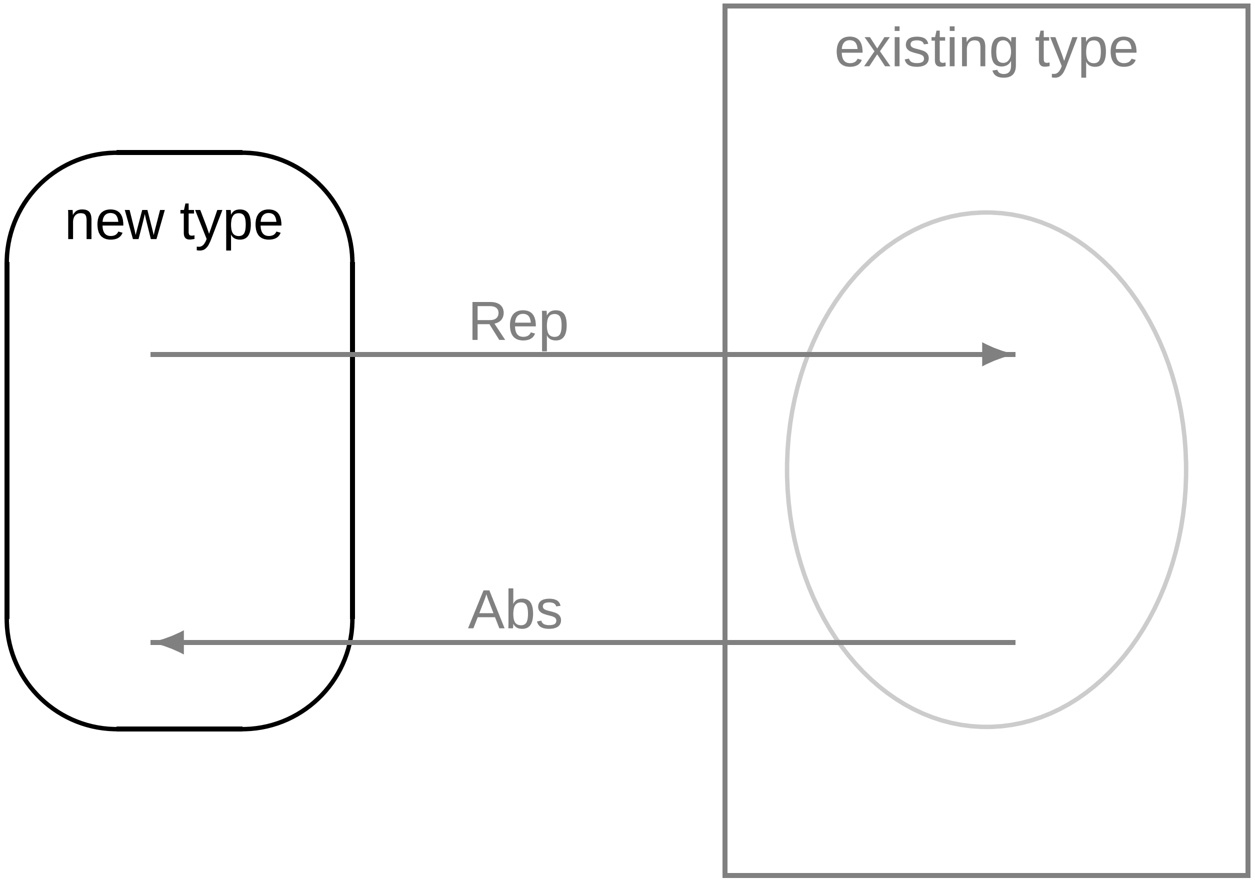
  Introduces a new type as a subset of an existing type.

  The proof shows that the set on the rhs in non-empty.

How typedef works



How typedef works



Example: Pairs

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

① Pick existing type: $\alpha \Rightarrow \beta \Rightarrow \text{bool}$

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 :: (com × state × state) set

➜ the rules define a set inductively

But which set?

Simpler Example

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

➜ 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?

➜ 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 DEFINITONS

We have learned today ...

- ➜ Sets
- ➜ Type Definitions
- ➜ Inductive Definitions