COMP 4161
NICTA Advanced Course

Advanced Topics in Software Verification

Gerwin Klein, June Andronick, Toby Murray
Content

→ Intro & motivation, getting started

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

→ Proof & Specification Techniques
  • Isar
  • Inductively defined sets, rule induction
  • Datatypes, recursion, induction
  • Calculational reasoning, mathematics style proofs
  • Hoare logic, proofs about programs

---

[1]

[1] Rough timeline

\[\text{a} \quad \text{a1 out; b a1 due; c a2 out; d a2 due; e session break; f a3 out; g a3 due}\]
Last Time

➡️ Sets
➡️ Type Definitions
➡️ Inductive Definitions
HOW INDUCTIVE DEFINITIONS WORK
The Nat Example

\[\begin{align*}
0 & \in \mathbb{N} \\
n & \in \mathbb{N} \quad \Rightarrow \quad n + 1 & \in \mathbb{N}
\end{align*}\]

→ \(\mathbb{N}\) 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)
Formally

Rules \( a_1 \in X \ldots a_n \in X \) with \( a_1, \ldots, a_n, a \in A \)

define set \( X \subseteq A \)

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

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

Example:

\[
R \equiv \{(\{} \}, 0\} \cup \{(\{} n \}, n + 1\). n \in \mathbb{R}\}
\]

\( \hat{R} \{3, 6, 10\} = \{0, 4, 7, 11\} \)
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:** \( X = \bigcap\{ B \subseteq A. B \text{ } R-\text{closed}\} \)
Generation from Above

\[ A \]

\[ R\text{-closed} \]

\[ X \]

\[ R\text{-closed} \]

\[ R\text{-closed} \]
Rule Induction

\[
\begin{align*}
0 & \in N \\
n & \in N \\
n + 1 & \in N
\end{align*}
\]

induces induction principle

\[\left[ P 0; \ \land n. \ P n \implies P (n + 1) \right] \implies \forall x \in X. \ P x\]

In general:

\[
\forall (\{a_1, \ldots a_n\}, a) \in R. \ P a_1 \land \ldots \land P a_n \implies P a
\]

\[
\forall x \in X. \ P x
\]
Why does this work?

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

$\forall x \in X. P x$

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

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

\[ a_1 \in X \quad \ldots \quad a_n \in X \quad C_1 \quad \ldots \quad C_m \]
\[ a \in X \]

induction scheme:

\[
(\forall \{a_1, \ldots, a_n\}, a) \in R. \quad P \ a_1 \land \ldots \land P \ a_n \land \\
C_1 \land \ldots \land C_m \land \\
\{a_1, \ldots, a_n\} \subseteq X \implies P \ a)
\]

\[
\implies \\
\forall x \in X. \ P \ x
\]
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:

\[
X_0 = \hat{R}^0 \{\} = \{\}
\]
\[
X_1 = \hat{R}^1 \{\} = \text{rules without hypotheses}
\]
\[
\vdots
\]
\[
X_n = \hat{R}^n \{\}
\]

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

$\hat{R}^0 \{ \} \cup \hat{R}^1 \{ \} \cup \hat{R}^2 \{ \} \cup \ldots$
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

Formalize the lecture in Isabelle:

- Define **closed** \( f \ A :: (\alpha \ \text{set} \Rightarrow \alpha \ \text{set}) \Rightarrow \alpha \ \text{set} \Rightarrow \text{bool} \)
- Show \( \text{closed} \ f \ A \land \text{closed} \ f \ B \Rightarrow \text{closed} \ f \ (A \cap B) \) if \( f \) is monotone (\text{mono} is predefined)
- Define \( \text{lfpt} \ f \) as the intersection of all \( f \)-closed sets
- Show that \( \text{lfpt} \ f \) is a fixpoint of \( f \) if \( f \) is monotone
- Show that \( \text{lfpt} \ f \) 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 \( \text{lfpt} \ \hat{R} \)
Rule Induction in Isar
inductive $X :: \alpha \Rightarrow \text{bool}$

where

rule\_1: "$[X \ s; A] \Longrightarrow X \ s'$"

\vdots

| rule\_n: \ldots |
Rule induction

\[ \text{show } \forall x \left( X \implies P \right) \]

\textbf{proof} (induct rule: X.induct)
\begin{itemize}
  \item \textbf{fix} \( s \) and \( s' \) \textbf{assume} \( X \ s \) and \( A \) and \( P \ s \)
  \item \ldots
  \item \textbf{show} \( P \ s' \)
\end{itemize}

\textbf{next}
\begin{itemize}
  \item \ldots
\end{itemize}

\textbf{qed}
Abbreviations

\textbf{show} \ "X\ x \Longrightarrow\ P\ x"\\
\textbf{proof} (induct rule: X.induct)\\
\hspace{1cm} \textbf{case} \ rule_1\\
\hspace{1cm} \ldots\\
\hspace{1cm} \textbf{show} \ ?\text{case}\\
\textbf{next}\\
\vdots\\
\textbf{next}\\
\hspace{1cm} \textbf{case} \ rule_n\\
\hspace{1cm} \ldots\\
\hspace{1cm} \textbf{show} \ ?\text{case}\\
\textbf{qed}
Implicit selection of induction rule

assume A: "X x"

show "P x"
using A proof induct

qed

lemma assumes A: "X x" shows "P x"
using A proof induct

qed
Renaming free variables in rule

\textbf{case } (\textit{rule}_i \ x_1 \ldots x_k )

Renames first $k$ variables in rule$_i$ to $x_1 \ldots x_k$. 
A remark on style

→ **case** (rule \( i \) \( x \) \( y \)) \( \ldots \) **show** ?case
  is easy to write and maintain

→ **fix** \( x \) \( y \) **assume** \( formula \) \( \ldots \) **show** \( formula' \)
  is easier to read:
  
  - all information is shown locally
  - no contextual references (e.g. ?case)
DEMO: RULE INDUCTION IN ISAR
We have learned today ...

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