Advanced Topics in Software Verification

Gerwin Klein

Formal Methods
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
- Calculational reasoning, mathematics style proofs
- Hoare logic, proofs about programs
LAST TIME

- Permutative rewriting, AC rules
- More confluence: critical pairs
- Knuth-Bendix Algorithm, Waldmeister
- More Isar: forward, backward, obtain, abbreviations, moreover
Give an Isar proof of the rich grandmother theorem
(automated methods allowed, but proof must be explaining)
BUILDING UP SPECIFICATION TECHNIQUES
Type ’a set: sets over type ’a

\[ \{\}, \{e_1, \ldots, e_n\}, \{x. P \, x\} \]
\[ e \in A, \quad A \subseteq B \]
\[ A \cup B, \quad A \cap B, \quad A - B, \quad -A \]
\[ \bigcup x \in A. \ B \, x, \quad \bigcap x \in A. \ B \, x, \quad \bigcap A, \quad \bigcup A \]
\[ \{i..j\} \]
\[ \text{insert :: } \alpha \Rightarrow \alpha \, \text{set} \Rightarrow \alpha \, \text{set} \]
\[ f^\prime A \equiv \{y. \exists x \in A. y = f \, x\} \]
\[ \ldots \]
Natural deduction proofs:

- equalityI: \[ A \subseteq B; \ B \subseteq A \] \implies A = B
- subsetI: \((\forall x. x \in A \implies x \in B)\) \implies A \subseteq B
- ... (see Tutorial)
Bounded Quantifiers

\[ \forall x \in A. P x \equiv \forall x. x \in A \rightarrow P x \]
\[ \exists x \in A. P x \equiv \exists x. x \in A \land P x \]
\[ \text{ballI: } (\forall x \in A. P x \rightarrow P x) \rightarrow \forall x \in A. P x \]
\[ \text{bspec: } [\forall x \in A. P x; x \in A] \rightarrow P x \]
\[ \text{bexI: } [P x; x \in A] \rightarrow \exists x \in A. P x \]
\[ \text{bexE: } [\exists x \in A. P x; \forall x. [x \in A; P x] \rightarrow Q] \rightarrow Q \]
DEMO: SETS
INDUCTIVE DEFINITIONS
\[ \langle \text{skip}, \sigma \rangle \rightarrow \sigma \quad \text{[e]} \sigma = v \quad \langle x := e, \sigma \rangle \rightarrow \sigma[x \leftarrow v] \]

\[ \langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma'' \quad \langle c_1; c_2, \sigma \rangle \rightarrow \sigma'' \]

\[ \text{[b]} \sigma = \text{False} \quad \langle \text{while b do c, } \sigma \rangle \rightarrow \sigma \]

\[ \text{[b]} \sigma = \text{True} \quad \langle c, \sigma \rangle \rightarrow \sigma' \quad \langle \text{while b do c, } \sigma' \rangle \rightarrow \sigma'' \quad \langle \text{while b do c, } \sigma \rangle \rightarrow \sigma'' \]
What does this mean?

$\langle c, \sigma \rangle \rightarrow \sigma'$ fancy syntax for a relation $(c, \sigma, \sigma') \in E$

$\Rightarrow$ relations are sets: $E :: (\text{com} \times \text{state} \times \text{state})$ set

$\Rightarrow$ the rules define a set inductively

But which set?
Simpler Example

\[ 0 \in \mathbb{N} \quad n \in \mathbb{N} \implies n + 1 \in \mathbb{N} \]

\(\Rightarrow\) \(\mathbb{N}\) is the set of natural numbers \(\mathbb{N}\)

\(\Rightarrow\) But why not the set of real numbers? \(0 \in \mathbb{R}\), \(n \in \mathbb{R} \implies n + 1 \in \mathbb{R}\)

\(\Rightarrow\) \(\mathbb{N}\) is the smallest set that is consistent with the rules.

Why the smallest set?

\(\Rightarrow\) Objective: no junk. Only what must be in \(X\) shall be in \(X\).

\(\Rightarrow\) Gives rise to a nice proof principle (rule induction)

\(\Rightarrow\) Alternative (greatest set) occasionally also useful: coinduction
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 $R \subseteq A \times A$ ($R, X$ 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:

$R \equiv \{\{\}, 0\} \cup \{\{n\}, n + 1\}$, $n \in \mathbb{R}$

$\hat{R} \{3, 6, 10\} = \{0, 4, 7, 11\}$
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

$A$

$R$-closed

$X$

$R$-closed

$R$-closed
Rule Induction

\[ \begin{array}{c}
0 \in N \\
n \in N \\
n + 1 \in N
\end{array} \]

induces induction principle

\[ [P \ 0; \ \wedge n. \ P \ n \ \Rightarrow \ P \ (n + 1)] \ \Rightarrow \ \forall x \in X. \ P \ x \]

In general:

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

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

\[\{x. P x\} \text{ is } R\text{-closed}\]

but: \[X \text{ is the least } R\text{-closed set}\]

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:

\[
\begin{align*}
(\forall \{a_1, \ldots, a_n\}, a) \in R. & \quad P a_1 \land \ldots \land P a_n \land \\
& \quad C_1 \land \ldots \land C_m \land \\
& \quad \{a_1, \ldots, a_n\} \subseteq X \implies P a \\
\implies \\
& \forall x \in X. \ P x
\end{align*}
\]
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:

\[
X_0 = \hat{R}^0 \ \{\} = \{\} \\
X_1 = \hat{R}^1 \ \{\} = \text{rules without hypotheses} \\
\vdots \\
X_n = \hat{R}^n \ \{\} \\
X_\omega = \bigcup_{n \in \mathbb{N}} (R^n \ \{\}) = X
\]
Generation from Below

\[
\hat{R}^0 \cup \hat{R}^1 \cup \hat{R}^2 \cup \ldots
\]
DEMO: INDUCTIVE DEFINITIONS
WE HAVE SEEN TODAY ...

- Sets in Isabelle
- Inductive Definitions
- Rule induction
- Fixpoints
Formalize this lecture in Isabelle:

- Define \( \text{closed} \ f \ A :: (\alpha \ \text{set} \Rightarrow \alpha \ \text{set}) \Rightarrow \alpha \ \text{set} \Rightarrow \text{bool} \)
- Show \( \text{closed} \ f \ A \land \text{closed} \ f \ B \iff \text{closed} \ f \ (A \cap B) \) if \( f \) is monotone (\texttt{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 \( \text{lfpt} \ \hat{R} \)