COMP 4161
NICTA Advanced Course

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
- Well founded recursion, Calculational reasoning
- Hoare logic, proofs about programs
- Locales, Presentation
LAST TIME

→ Sets in Isabelle
LAST TIME

→ Sets in Isabelle
→ Inductive Definitions
LAST TIME

→ Sets in Isabelle
→ Inductive Definitions
→ Rule induction
LAST TIME

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

- **Define** closed 
  
  \[ f \mathcal{A} :: (\alpha \text{ set} \Rightarrow \alpha \text{ set}) \Rightarrow \alpha \text{ set} \Rightarrow \text{bool} \]

- **Show**
  
  \[ \text{closed } f \mathcal{A} \land \text{closed } f \mathcal{B} \Rightarrow \text{closed } f (\mathcal{A} \cap \mathcal{B}) \] 
  
  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$_1$: ”$[X \ s; A] \Longrightarrow X \ s'$”

: 

| rule$_n$: . . .
show "X x \rightarrow P x"

proof (induct rule: X.induct)
  fix s and s' assume "X s" and "A" and "P s"
  ...
  show "P s'"

next

: 

qed
show "\( X \ x \implies P \ x \)"

proof (induct rule: X.induct)
  case rule\(_1\)
  
  \ldots

  show ?case

next

: :

next

  case rule\(_n\)
  
  \ldots

  show ?case

qed
assume A: "\( X \ x \)"

::

show "\( P \ x \)"

using A proof induct

::

qed
I MPLICIT 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
case \( \text{rule}_{i} \, x_{1} \ldots x_{k} \)

Renames first \( k \) variables in \( \text{rule}_{i} \) to \( x_{1} \ldots x_{k} \).
A REMARK ON STYLE

→ case (rule_i x y) ... show ?case
   is easy to write and maintain
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)
WE HAVE SEEN SO FAR ...

➔ Formalising inductive sets and rule induction
WE HAVE SEEN SO FAR...

- Formalising inductive sets and rule induction
- Rule induction in Isar
WE HAVE SEEN SO FAR ...

➔ Formalising inductive sets and rule induction
➔ Rule induction in Isar
➔ Implicit induction rule selection
WE HAVE SEEN SO FAR ...

- Formalising inductive sets and rule induction
- Rule induction in Isar
- Implicit induction rule selection
- Case abbreviations
WE HAVE SEEN SO FAR ...

- Formalising inductive sets and rule induction
- Rule induction in Isar
- Implicit induction rule selection
- Case abbreviations
- Renaming case variables