# Experiments in Verification

SS 2009

Christian Sternagel (VO)[1]

Computational Logic
Institute of Computer Science
University of Innsbruck

27 March 2009

[1] christian.sternagel@uibk.ac.at



## This Time

### Session 1
formal verification, Isabelle/HOL basics, functional programming in HOL

### Session 2
simplification, function definitions, induction, calculational reasoning

### Session 3
natural deduction, propositional logic, predicate logic

### Session 4
sets, relations, inductively defined sets, advanced topics

# Sets in Isabelle

## Type

```
(* A set is defined by its characteristic function. *)
types 'a set = ('a ⇒ bool)
```

## Membership

- $x$ is member of set $S$ if characteristic function returns `True`
- **`lemma`** `mem_def: "`$x \in S \equiv S\ x$`"`





# Basic Operations on Sets

## Intersection

- notation: $A \cap B$ (ASCII: $A$ `Int` $B$)
- `IntI`: $[\![c \in A; c \in B]\!] \Longrightarrow c \in A \cap B$
- `IntD1`: $c \in A \cap B \Longrightarrow c \in A$
- `IntD2`: $c \in A \cap B \Longrightarrow c \in B$

## Union

- notation: $A \cup B$ (ASCII: $A$ `Un` $B$)
- `UnI1`: $c \in A \Longrightarrow c \in A \cup B$
- `UnI2`: $c \in B \Longrightarrow c \in A \cup B$
- `UnE`: $[\![c \in A \cup B; c \in A \Longrightarrow P; c \in B \Longrightarrow P]\!] \Longrightarrow P$

# Basic Operations on Sets (cont'd)

## Complement and Difference

- complement: $-A$
- `Compl_iff`: $(c \in -A) = (c \notin A)$
- difference: $A - B$

## Subsets

- notation: $A \subseteq B$ (ASCII: $A <= B$)
- `subsetI`: $(\bigwedge x.\ x \in A \Longrightarrow x \in B) \Longrightarrow A \subseteq B$
- `equalityI`: $[\![A \subseteq B; B \subseteq A]\!] \Longrightarrow A = B$





# Set Notation

- the empty set: `{}`
- the universal set: `UNIV`
- a singleton set: $\{x\}$
- insertion (`insert_is_Un`): `insert` $x\ A = \{x\} \cup A$
- finite sets, e.g., $\{a, b, c, d\}$

# An Example Proof

lemma

```
lemma "A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)"
```

Proof.
Isabelle □





# A Shorter Proof

## The Method `blast`

- applies introduction and elimination rules automatically
- suitable for many goals concerning logical and/or set operations

```
lemma "A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)"
  by blast
```

# Set Comprehension - By Example

| Mathematics | Isabelle |
|---|---|
| $\{x \mid P(x)\}$ | `{x. P x}` |
| $\{(x, y) \mid x \in A, y \in B\}$ | `{(x,y) \| x y. x∈A ∧ y∈B}` |





# Binding Operators for Sets

## Operators

- bounded universal quantifier: $\forall x \in A.\ P\ x$
- bounded existential quantifier: $\exists x \in A.\ P\ x$

## Useful Lemmas

- `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`: $[\![\exists x \in A.\ P\ x; \bigwedge x.\ [\![x \in A; P\ x]\!] \Longrightarrow Q]\!] \Longrightarrow Q$

# Relations

## Basics

- a relation is a set of pairs (`('a×'b)set`)
- identity relation: `Id` $\equiv \{p.\ \exists x.\ p = (x, x)\}$
- composition: $r$ `O` $s \equiv \{(x, z).\ \exists y.\ (x, y) \in s \wedge (y, z) \in r\}$
- converse: $((a, b) \in r\hat{}{-}1) = ((b, a) \in r)$

## Reflexive and Transitive Closure

- reflexive and transitive closure: $r\hat{}*$
- transitive closure: $r\hat{}+$
- `rtrancl_refl`: $(a, a) \in r\hat{}*$
- `r_into_rtrancl`: $p \in r \Longrightarrow p \in r\hat{}*$
- `rtrancl_trans`: $[\![(a, b) \in r\hat{}*; (b, c) \in r\hat{}*]\!] \Longrightarrow (a, c) \in r\hat{}*$





# Example

## Lemma

```
lemma "(r O s)^-1 = s^-1 O r^-1"
```

## Proof.

Isabelle □

# An Introductory Definition

## Example (Even Numbers)

```
inductive_set even :: "nat set"
where zero[intro!]: "0 ∈ even"
    | step[intro!]: "n ∈ even ⟹ Suc(Suc n) ∈ even"
```

## Remarks

- `intro`: declares a lemma as introduction rule (for blast/auto)
- `elim`: declares a lemma as elimination rule (for blast/auto)
- adding a `!` tells the system that a rule is safe (i.e., it can always be applied without making the goal unprovable)
- `even` is the smallest set constructed by finitely many applications of the two rules `zero` and `step` (i.e., it contains only elements that can be added via the rules)





# Even Numbers Are Divisible By 2

```
lemma even_imp_dvd: "n ∈ even ⟹ 2 dvd n"
proof (induct rule: even.induct)
  case zero show ?case by simp
next
  case (step n)
  hence IH: "2 dvd n" by simp
  then obtain k where "n = 2*k"
    unfolding dvd_def by (rule exE)
  hence "Suc(Suc n) = 2*(Suc k)" by simp
  thus ?case unfolding dvd_def by (rule exI)
qed
```

# Advanced Inductive Sets

## Arguments

- an inductive definition may take arguments
- hence it is possible to define functions yielding sets inductively
- the keyword **for** is used to introduce arguments

## Example (Reflexive Transitive Closure)

```
inductive_set
  rtc :: "('a × 'a)set ⇒ ('a × 'a)set" ("_*" [1000] 999)
  for r :: "('a × 'a)set"
where rtc_refl[iff]: "(x,x) ∈ r*"
    | rtc_step: "(x,y) ∈ r ⟹ (y,z) ∈ r* ⟹ (x,z) ∈ r*"
```





# rtc is Transitive

## Lemma

```
lemma "(x,y) ∈ r* ⟹ (y,z) ∈ r* ⟹ (x,z) ∈ r*"
```

## Proof.

Isabelle □

# Evaluation

## LVA-Code

# 703523-0

## Additional Questions

(1) I can prove simple lemmas in Isabelle/HOL.
(2) I would prefer having a final exam instead of a project.
(3) The slides where generally helpful.
(4) There was too little theory.





# Projects

http://isabelle.in.tum.de/exercises/advanced/sorting/ex.pdf
http://isabelle.in.tum.de/exercises/advanced/mergesort/ex.pdf
http://isabelle.in.tum.de/exercises/advanced/tries/ex.pdf
http://isabelle.in.tum.de/exercises/advanced/interval/ex.pdf
http://isabelle.in.tum.de/exercises/advanced/regmachine/ex.pdf
http://isabelle.in.tum.de/exercises/proj/hanoi/ex.pdf
http://isabelle.in.tum.de/exercises/proj/euclid/ex.pdf
http://isabelle.in.tum.de/exercises/proj/compSE/ex.pdf
http://isabelle.in.tum.de/exercises/proj/bignat/ex.pdf
http://isabelle.in.tum.de/exercises/proj/optComp/ex.pdf