

# Experiments in Verification

SS 2009

Christian Sternagel (VO)<sup>1</sup>

Computational Logic  
Institute of Computer Science  
University of Innsbruck

27 March 2009

---

<sup>1</sup>[christian.sternagel@uibk.ac.at](mailto: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  $\Rightarrow$  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:  $A \text{ Int } B$ )
- ▶ IntI:  $\llbracket c \in A; c \in B \rrbracket \implies c \in A \cap B$
- ▶ IntD1:  $c \in A \cap B \implies c \in A$
- ▶ IntD2:  $c \in A \cap B \implies c \in B$

## Union

- ▶ notation:  $A \cup B$  (ASCII:  $A \text{ Un } B$ )
- ▶ UnI1:  $c \in A \implies c \in A \cup B$
- ▶ UnI2:  $c \in B \implies c \in A \cup B$
- ▶ UnE:  $\llbracket c \in A \cup B; c \in A \implies P; c \in B \implies P \rrbracket \implies P$

# Basic Operations on Sets (cont'd)

## Complement and Difference

- ▶ complement:  $\neg A$
- ▶ Compl\_iff:  $(c \in \neg A) = (c \notin A)$
- ▶ difference:  $A - B$

## Subsets

- ▶ notation:  $A \subseteq B$  (ASCII:  $A \leq B$ )
- ▶ subsetI:  $(\bigwedge x. x \in A \implies x \in B) \implies A \subseteq B$
- ▶ equalityI:  $\llbracket A \subseteq B; B \subseteq A \rrbracket \implies A = B$

# Set Notation

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

# An Example Proof

lemma

**lemma** " $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ "

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** " $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ "  
**by** `blast`



# Set Comprehension - By Example

Mathematics	Isabelle
$\{x \mid P(x)\}$	$\{x. P\ x\}$
$\{(x, y) \mid x \in A, y \in B\}$	$\{(x, y) \mid x\ y. x \in A \wedge y \in B\}$

# 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:  $(\bigwedge x. x \in A \implies P x) \implies \forall x \in A. P x$
- ▶ bspec:  $\llbracket \forall x \in A. P x; x \in A \rrbracket \implies P x$
- ▶ bexI:  $\llbracket P x; x \in A \rrbracket \implies \exists x \in A. P x$
- ▶ bexE:  $\llbracket \exists x \in A. P x; \bigwedge x. \llbracket x \in A; P x \rrbracket \implies Q \rrbracket \implies Q$

# Relations

## Basics

- ▶ a relation is a **set of pairs** ( $(a \times b)$  set)
- ▶ **identity relation**:  $\text{Id} \equiv \{p. \exists x. p = (x, x)\}$
- ▶ **composition**:  $r \circ s \equiv \{(x, z). \exists y. (x, y) \in s \wedge (y, z) \in r\}$
- ▶ **converse**:  $((a, b) \in r^{-1}) = ((b, a) \in r)$

## Reflexive and Transitive Closure

- ▶ reflexive and transitive closure:  $r^*$
- ▶ transitive closure:  $r^+$
- ▶ `rtrancl_refl`:  $(a, a) \in r^*$
- ▶ `r_into_rtrancl`:  $p \in r \implies p \in r^*$
- ▶ `rtrancl_trans`:  $[(a, b) \in r^*; (b, c) \in r^*] \implies (a, c) \in r^*$

# Example

## Lemma

**lemma** "(r 0 s)<sup>-1</sup> = s<sup>-1</sup> 0 r<sup>-1</sup>"

## Proof.

Isabelle



# An Introductory Definition

## Example (Even Numbers)

```
inductive_set even :: "nat set"  
where zero[intro!]: "0 ∈ even"  
      | step[intro!]: "n ∈ even  $\implies$  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  $\implies$  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** " $(x,y) \in r^* \implies (y,z) \in r^* \implies (x,z) \in r^*$ "

## 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 were 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>