

# Structured Induction Proofs in Isabelle/Isar

Makarius Wenzel  
Technische Universität München

August 2006

# Motivation

# Background

- **Isabelle/Pure:** minimal logical framework
  - abstract syntax
  - primitive inferences
- **Isabelle/Isar:** framework for human-readable proofs
  - proof context
  - proof structure
- Common aims:
  - generic
  - simple foundations
  - practically useful

## From rules to proofs – example (1)

- Textbook:  $\frac{A \quad B}{A \wedge B}$
- Isabelle/Pure:  $A \implies B \implies A \wedge B$
- Isabelle/Isar:

**have**  $A$  **and**  $B$   $\langle proof \rangle$   
**then have**  $A \wedge B$  ..

**have**  $A \wedge B$

**proof**

**show**  $A$   $\langle proof \rangle$

**show**  $B$   $\langle proof \rangle$

**qed**

## From rules to proofs – example (2)

- Textbook: 
$$\frac{P\ 0 \quad \begin{array}{c} [n, P\ n] \\ \vdots \\ P\ (Suc\ n) \end{array}}{P\ n}$$
- Isabelle/Pure:  $P\ 0 \implies (\bigwedge n. P\ n \implies P\ (Suc\ n)) \implies P\ n$
- Isabelle/Isar:

```
have  $P\ n$   
proof (rule nat-induct)  
  show  $P\ 0$   $\langle proof \rangle$   
  fix  $n$   
  assume  $P\ n$   
  show  $P\ (Suc\ n)$   $\langle proof \rangle$   
qed
```

# Why induction is non-trivial

- Observation: real applications use compound inductive predicates
  - local parameters  $\bigwedge x. \dots$
  - local premises  $A \implies \dots$
  - local definitions  $x \equiv a y$
  - simultaneous goals  $P x \ \& \ Q y$
- Problem: explicit logical connectives need to be handled 3 times
  - reformulate statement
  - elimination/introduction in induction body
  - reformulate result

## Example: complex induction

```
have  $\forall x. A\ n\ x \longrightarrow P\ n\ x$   
proof (rule nat-induct)  
  show  $\forall x. A\ 0\ x \longrightarrow P\ 0\ x$   
  proof  
    fix  $x$   
    show  $A\ 0\ x \longrightarrow P\ 0\ x$   
    proof  
      assume  $A\ 0\ x$   
      show  $P\ 0\ x$   $\langle proof \rangle$   
    qed  
  qed  
next  
  fix  $n$   
  assume raw-hyp:  $\forall x. A\ n\ x \longrightarrow P\ n\ x$   
  have hyp:  $\bigwedge x. A\ n\ x \Longrightarrow P\ n\ x$   
  proof —
```

**fix**  $x$   
**from** *raw-hyp* **have**  $A\ n\ x \longrightarrow P\ n\ x ..$   
**also assume**  $A\ n\ x$   
**finally show**  $P\ n\ x .$   
**qed**  
**show**  $\forall x. A\ (Suc\ n)\ x \longrightarrow P\ (Suc\ n)\ x$   
**proof**  
**fix**  $x$   
**show**  $A\ (Suc\ n)\ x \longrightarrow P\ (Suc\ n)\ x$   
**proof**  
**assume** *prem*:  $A\ (Suc\ n)\ x$   
**show**  $P\ (Suc\ n)\ x$  *<proof>*  
**qed**  
**qed**  
**then have**  $A\ n\ x \longrightarrow P\ n\ x ..$   
**also assume**  $A\ n\ x$   
**finally show**  $P\ n\ x .$

# The *induct* method

## Introducing *induct*

- Idea: sophisticated wrapper for Pure *rule* method
  - hides the gory details
  - enables native reasoning, using first-class Isar language elements
  - uses sane proof structure, instead of ad-hoc automation
- Method syntax:

*facts*

*(induct insts fixing: vars rule: rule)*

- *facts*: current facts passed to any Isar method (cf. **then**, **using**)
- *insts*: induction variables  $x$ , optionally with definition  $x \equiv a$
- *vars*: fixed variables
- *rule*: actual induction rule

## Example: complex induction done right

```
assume  $A\ n\ x$   
have  $P\ n\ x$  using  $\langle A\ n\ x \rangle$   
proof (induct n fixing: x)  
  case 0  
    from  $\langle A\ 0\ x \rangle$   
    show  $P\ 0\ x$   $\langle proof \rangle$   
next  
  case ( $Suc\ n$ )  
    from  $\langle \bigwedge x. A\ n\ x \implies P\ n\ x \rangle$   
    and  $\langle A\ (Suc\ n)\ x \rangle$   
    show  $P\ (Suc\ n)\ x$   $\langle proof \rangle$   
qed
```

## Internal operations (1)

1. context: declare local *defs* for defined induction variables  $x \equiv a$
2. rule: apply *insts* according to conclusion  $P\ x\ y\ z$
3. rule: expand *defs* in major premises
4. rule: consume prefix of *facts* according to major premises
5. goal: insert remaining *facts* and *defs*
6. goal: closeup fixed variables, using  $(\bigwedge x. B\ x) \implies B\ a$
7. goal: internalize  $\bigwedge/\implies/\equiv$  into the object-logic
8. rule: unify conclusion against goal ( $\rightarrow$  fully-instantiated rule)
9. rule: carefully recover internalized  $\bigwedge/\implies/\equiv$  in the inductive cases
10. context: extract inductive cases from rule (for **case**)
11. context: discharge *defs*
12. goal: apply fully-instantiated rule

## Internal operations (2) — simultaneous goals

1. goal: internalize  $A$  &  $B$  into object-logic
2. goal: apply induction rule
3. goal: recover  $A$  &  $B$  and apply congruences wrt.  $\wedge/\implies$
4. goal: eliminate & by *currying*
5. context: extract nested cases, numbered for each conjunct

# **Common induction patterns**

# Overview of patterns

## 1. local premises

```
have  $P\ n$  using  $\langle A\ n \rangle$   
proof (induct  $n$ ) ...
```

## 2. local parameters

```
have  $P\ x\ n$   
proof (induct  $n$  fixing:  $x$ ) ...
```

## 3. local definitions

```
have  $P\ (a\ x)$   
proof (induct  $n \equiv a\ x$  fixing:  $x$ ) ...
```

## 4. simultaneous goals

```
have  $P\ n$  and  $Q\ n$   
proof (induct  $n$ ) ...
```

## Pattern 1/2: local premises and parameters

**lemma**

**fixes**  $n :: \text{nat}$  **and**  $x :: 'a$

**assumes**  $A\ n\ x$

**shows**  $P\ n\ x$  **using**  $\langle A\ n\ x \rangle$

**proof** (*induct n fixing: x*)

**case** 0

**note**  $\langle A\ 0\ x \rangle$

**show**  $P\ 0\ x$   $\langle \text{proof} \rangle$

**next**

**case** ( $Suc\ n$ )

**note**  $\langle \bigwedge x. A\ n\ x \implies P\ n\ x \rangle$

**and**  $\langle A\ (Suc\ n)\ x \rangle$

**show**  $P\ (Suc\ n)\ x$   $\langle \text{proof} \rangle$

**qed**

## Pattern 3: local definitions

**lemma**

**fixes**  $a :: 'a \Rightarrow \text{nat}$

**assumes**  $A (a x)$

**shows**  $P (a x)$  **using**  $\langle A (a x) \rangle$

**proof** (*induct*  $n \equiv a x$  *fixing*:  $x$ )

**case** 0

**note**  $\langle A (a x) \rangle$

**and**  $\langle 0 = a x \rangle$

**show**  $P (a x)$   $\langle \text{proof} \rangle$

**next**

**case** (*Suc*  $n$ )

**note**  $\langle \bigwedge x. A (a x) \Longrightarrow n = a x \Longrightarrow P (a x) \rangle$

**and**  $\langle A (a x) \rangle$

**and**  $\langle \text{Suc } n = a x \rangle$

**show**  $P (a x)$   $\langle \text{proof} \rangle$

**qed**

## Pattern 4: simultaneous goals

**lemma**

**fixes**  $n :: nat$

**and**  $x :: 'a$  **and**  $y :: 'b$

**shows**  $A\ n\ x \implies P\ n\ x$

**and**  $B\ n\ y \implies Q\ n\ y$

**proof** (*induct n fixing: x y*)

**case** 0

{ **case** 1

**note**  $\langle A\ 0\ x \rangle$

**show**  $P\ 0\ x$   $\langle proof \rangle$  }

{ **case** 2

**note**  $\langle B\ 0\ y \rangle$

**show**  $Q\ 0\ y$   $\langle proof \rangle$  }

**next**

**case** (*Suc n*)

**note**  $\langle \bigwedge x. A\ n\ x \implies P\ n\ x \rangle$   $\langle \bigwedge y. B\ n\ y \implies Q\ n\ y \rangle$

```
then have some-intermediate-result <proof>
{ case 1
  note < $A (Suc\ n)\ x$ >
  show  $P (Suc\ n)\ x$  <proof> }
{ case 2
  note < $B (Suc\ n)\ y$ >
  show  $Q (Suc\ n)\ y$  <proof> }
qed
```

# Conclusion

# Achievements

- Minimal requirements on induction rule format  
possible extensions include:
  - nominal induction: additional “freshness” context  
(*nominal-induct n avoiding: a b c fixing: x y*)
  - coinduction: dualized version  
(*coinduct n fixing: x y*)
- Clear separation of proof interpretation from automated reasoning
- Isabelle/Isar framework is sufficiently flexible to support domain specific proof patterns

## Related work

- General: little support for complex induction proofs in existing interactive systems
- Coq: basic support for including full (!) proof context in induction
- HOL/HOL-Light: recommend use of adhoc automation
- Mizar: induction as manual forward proof  
(less puristic than Isabelle/Isar, but still requires many intermediate steps for  $\forall$  and  $\longrightarrow$ )