# How to Write a Definitional Package for Isabelle

Stefan Berghofer

Institut für Informatik
Technische Universität München

Isabelle Developers Workshop, 14.8.2009

# Outline

# Outline

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*
  Properties are described by axioms

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*
  Properties are described by axioms
- More comples concepts must be defined using these constants

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*
  Properties are described by axioms
- More comples concepts must be defined using these constants
  Properties have to be derived from definitions by formal proof

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*
  Properties are described by axioms
- More comples concepts must be defined using these constants
  Properties have to be derived from definitions by formal proof
- **Note:** reducing everything to primitive concepts "by hand" is tedious!

# The Definitional Approach

- HOL is based on just a few primitives: $=$, $\longrightarrow$, *THE* / *SOME*
  Properties are described by axioms
- More comples concepts must be defined using these constants
  Properties have to be derived from definitions by formal proof
- **Note:** reducing everything to primitive concepts "by hand" is tedious!

> *The method of 'postulating' what we want has many advantages; they are the same as the advantages of theft over honest toil. Let us leave them to others and proceed with our honest toil.*
>
> — Bertrand Russell, Introduction to Mathematical Philosophy

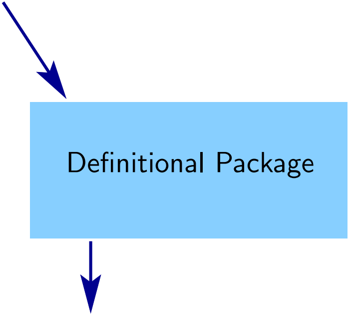
# Definitional Packages

**inductive** $\mathit{even} :: \mathit{nat} \Rightarrow \mathit{bool}$
**where**
$\quad \mathit{even}\ 0$
$|\ \mathit{even}\ n \Longrightarrow \mathit{even}\ (\mathit{Suc}\ (\mathit{Suc}\ n))$

Definitional Package

# Definitional Packages

**inductive** $even :: nat \Rightarrow bool$
**where**
$\quad even\ 0$
$\mid\ even\ n \Longrightarrow even\ (Suc\ (Suc\ n))$




$$even \equiv lfp\ (\lambda p\ x.\ x = 0 \lor (\exists\, n.\ x = Suc\ (Suc\ n) \land p\ n))$$

# Definitional Packages

**inductive** *even* :: *nat* ⇒ *bool*
**where**
  *even 0*
| *even n* ⟹ *even* (*Suc* (*Suc n*))

**Theorem**

*even 0*
*even n* ⟹ *even* (*Suc* (*Suc n*))

Definitional Package

$$even \equiv lfp\ (\lambda p\ x.\ x = 0 \vee (\exists\, n.\ x = Suc\ (Suc\ n) \wedge p\ n))$$

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules
  - Only proves a weaker form of the rule induction theorem

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules
  - Only proves a weaker form of the rule induction theorem

**How to tackle the problem?**

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules
  - Only proves a weaker form of the rule induction theorem

**How to tackle the problem?**

1. Try out the construction on some examples

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules
  - Only proves a weaker form of the rule induction theorem

### How to tackle the problem?

1. Try out the construction on some examples
2. Figure out the general construction principle

# Running Example: Simple Inductive Package

- Does not use the general Knaster-Tarski fixpoint theorem on complete lattices [Paulson, 2000]
- Uses a simple impredicative encoding [Melham, 1992]
- **Limitations**
  - No support for introduction rules involving arbitrary monotone operators
  - Does not prove case analysis (inversion) rules
  - Only proves a weaker form of the rule induction theorem

**How to tackle the problem?**

1. Try out the construction on some examples
2. Figure out the general construction principle
3. Write code implementing the construction principle

# Outline

## Reminder: Definition of Basic Logical Operators

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R. \ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$
- $P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$
- $P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$
- $Ex\ P \equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$
- $P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$
- $Ex\ P \equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

## Generalizes to recursive definitions

$even'\ 0$
$even'\ n \Longrightarrow even'\ (Suc\ (Suc\ n))$

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R$
- $P \vee Q \equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$
- $Ex\ P \equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q$

## Generalizes to recursive definitions

$even'\ 0$
$even'\ n \Longrightarrow even'\ (Suc\ (Suc\ n))$

## *even'* is least predicate closed under above introduction rules

$even' \equiv$
$\lambda z.\ \forall\, even'.\ even'\ 0 \longrightarrow (\forall\, n.\ even'\ n \longrightarrow even'\ (Suc\ (Suc\ n)))$
$\longrightarrow even'\ z$

# The Impredicative Encoding

## Reminder: Definition of Basic Logical Operators

- $P \wedge Q \equiv \forall R. (P \longrightarrow Q \longrightarrow R) \longrightarrow R$
- $P \vee Q \equiv \forall R. (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$
- $Ex\ P \equiv \forall Q. (\forall x. P\ x \longrightarrow Q) \longrightarrow Q$

## Generalizes to recursive definitions

$even'\ 0$
$even'\ n \Longrightarrow even'\ (Suc\ (Suc\ n))$

## $even'$ is least predicate closed under above introduction rules

$even' \equiv$
$\lambda z. \forall even'. even'\ 0 \longrightarrow (\forall n. even'\ n \longrightarrow even'\ (Suc\ (Suc\ n)))$
$\longrightarrow even'\ z$

### Intuition

$even'\ x$ holds iff $P\ x$ holds for every predicate $P$ closed under the above rules.

# Deriving the Properties "by hand"

**Demo**

# Outline

# Characteristic Rules

## Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

# Characteristic Rules

## Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

## Definition of Predicates

$$R_i \equiv \lambda \vec{p}\ \vec{z}_i.\ \forall \vec{P}.\ K_1 \longrightarrow \cdots \longrightarrow K_r \longrightarrow P_i\ \vec{z}_i$$

$$K_i \equiv \forall \vec{x}_i.\ \vec{A}_i \longrightarrow \left(\forall \vec{y}_{ij}.\ \vec{B}_{ij} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \longrightarrow P_{l_i}\ \vec{t}_i$$

# Characteristic Rules

## Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

## Definition of Predicates

$$R_i \equiv \lambda \vec{p}\ \vec{z}_i.\ \forall \vec{P}.\ K_1 \longrightarrow \cdots \longrightarrow K_r \longrightarrow P_i\ \vec{z}_i$$

$$K_i \equiv \forall \vec{x}_i.\ \vec{A}_i \longrightarrow \left(\forall \vec{y}_{ij}.\ \vec{B}_{ij} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \longrightarrow P_{l_i}\ \vec{t}_i$$

## Induction rules (weak)

$$R_i\ \vec{p}\ \vec{z}_i \Longrightarrow I_1 \Longrightarrow \cdots \Longrightarrow I_r \Longrightarrow P_i\ \vec{z}_i$$

$$I_i \equiv \bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \Longrightarrow P_{l_i}\ \vec{t}_i$$

# Proof of Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\ldots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

**Unfolding the definition**

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow \forall \vec{P}.\ \vec{K} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow \\ \forall \vec{P}.\ \vec{K} \longrightarrow P_{l_i}\ \vec{t}_i$$

$$K_i \equiv \forall \vec{x}_i.\ \vec{A}_i \longrightarrow \left(\forall \vec{y}_{ij}.\ \vec{B}_{ij} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \longrightarrow P_{l_i}\ \vec{t}_i$$

# Proof of Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

**Applying introduction rules for $\forall$ and $\longrightarrow$**

$$\bigwedge \vec{x}_i\ \vec{P}.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow \forall \vec{P}.\ \vec{K} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow \vec{K} \Longrightarrow P_{l_i}\ \vec{t}_i$$

$$K_i \equiv \forall \vec{x}_i.\ \vec{A}_i \longrightarrow \left(\forall \vec{y}_{ij}.\ \vec{B}_{ij} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \longrightarrow P_{l_i}\ \vec{t}_i$$

# Proof of Introduction Rules

$$\bigwedge \vec{x}_i.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow R_{k_{ij}}\ \vec{p}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow R_{l_i}\ \vec{p}\ \vec{t}_i$$

**Applying $K_{l_i}$**

$$\begin{array}{l}\bigwedge \vec{x}_i\ \vec{P}.\ \vec{A}_i \Longrightarrow \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow \forall \vec{P}.\ \vec{K} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \Longrightarrow \\ \qquad \vec{K} \Longrightarrow \left\{ \begin{array}{l} \vec{A}_i \\ \left(\bigwedge \vec{y}_{ij}.\ \vec{B}_{ij} \Longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \end{array} \right. \end{array}$$

$$K_i \equiv \forall \vec{x}_i.\ \vec{A}_i \longrightarrow \left(\forall \vec{y}_{ij}.\ \vec{B}_{ij} \longrightarrow P_{k_{ij}}\ \vec{s}_{ij}\right)_{j=1,\dots,m_i} \longrightarrow P_{l_i}\ \vec{t}_i$$

# Outline

# Stages of a Definitional Package

1. Parse the specification

# Stages of a Definitional Package

1. Parse the specification
2. Make definitions

# Stages of a Definitional Package

1. Parse the specification
2. Make definitions
3. Prove characteristic properties

# Stages of a Definitional Package

1. Parse the specification
2. Make definitions
3. Prove characteristic properties
4. Store theorems