Chapter 6

Inductive Definitions and Fixed Points

Overview of Chapter

6. Inductive Definitions and Fixed Points
6.1 Inductively defined sets and predicates
6.2 Fixed point theory for inductive definitions
6.3 Specifying and verifying transition systems

Introduction

Constructs for defining types and functions
Isabelle/HOL provides two core constructs for conservative extensions:

1. Constant definitions
2. Type definitions

Based on the core construct, there are further constructs:

- Recursive function definitions (`primrec, fun, function`)
- Recursive datatype definitions (`datatype`)
- Co-/inductively defined sets (`inductive_set, coinductive_set`)
- Co-/inductively defined predicates (`inductive, coinductive`)

Motivation

Goals

- Learn about inductive definitions:
  ~ important concept in computer science!
  E.g., to define operational semantics.
- Learn the underlying fixed point theory:
  ~ fundamental theory in computer science!
- Learn how to apply it to transition systems
  ~ central modeling concept for operational behavior!
6. Inductive Definitions and Fixed Points

6.1 Inductively defined sets and predicates

Introductory example

Informally:

- 0 is even
- If n is even, so is n + 2
- These are the only even numbers

In Isabelle/HOL:

```isabelle
-- The set of all even numbers
inductive_set even :: "nat set" where
  zero [intro!] "0 ∈ even" |
  step [intro!] "n ∈ even ⇒ n + 2 ∈ even"
```

Format of inductive definitions

```
inductive_set S :: "τ set" where
  "⟦ a₁ ∈ S; ...; aₙ ∈ S; A₁; ...; Aₖ ⟧ ⇒ a ∈ S" |
  ... |

where
- A₁, ..., Aₖ are side conditions not involving S and
- a is a term build from a₁, ..., aₙ.
```

The rules can be given names and attributes as seen in definition of even.

Embedding inductive definitions into HOL

Conservative theory extension

From an inductive definition, Isabelle generates a definition using a fixed point operator and proves theorems about it that can be used as proof rules.

The theory underlying the fixed point definition is explained in Subsect. 2.
### Generated rules

**Rules**

Generated rules include

- the introduction rules of the definition, e.g.,
  
  \[
  0 \in \text{even} \\
  n \in \text{even} \implies n + 2 \in \text{even}
  \]
  
  \(\text{(even.zero)}\)
  \(\text{(even.step)}\)

- an elimination rule for case analysis and

- an induction rule.

### Proving simple properties of inductive sets

**Example 1:**

**Lemma:** \(4 \in \text{even}\)

**Proof:**

\[
0 \in \text{even} \implies 2 \in \text{even} \implies 4 \in \text{even}
\]

**Discussion:**

- Simple: Use even.zero and apply rule even.step finitely many times.
- Works because there is no free variable

### Rule induction for even

To prove \(n \in \text{even} \implies P n\) by rule induction, one has to show:

- \(P 0\)
- \(P n \implies P (n + 2)\)

Isabelle provides the rule even.induct:

\[
\begin{aligned}
\ll n \in \text{even};
\forall n. P n \implies P(n + 2) \rr & \implies P n
\end{aligned}
\]
6. Inductive Definitions and Fixed Points

6.1 Inductively defined sets and predicates

Rule induction vs. natural/structural induction

Remarks:
- Rule induction uses the induction steps of the inductive definition and not of the underlying datatype! It differs from natural/structural induction.
- In the context of partial recursive functions, a similar proof technique is often called computational or fixed point induction.

Rule induction in general

Let $S$ be an inductively defined set.
To prove $x \in S \implies P x$ by rule induction on $x \in S$, we must prove for every rule:

$$\left[ a_1 \in S; \ldots; a_n \in S \right] \implies a \in S$$

that $P$ is preserved:

$$\left[ P a_1; \ldots; P a_n \right] \implies P a$$

In Isabelle/HOL: apply (induct rule: $S$.induct)

Inductive predicates

Isabelle/HOL also supports the inductive definition of predicates:

$$X \in S \leadsto S x$$

Example:

```isabelle
ingductive even :: "nat \Rightarrow bool" where
"even 0" |
"even n \Rightarrow even (n+2)"
```

Comparison:
- predicate: simpler syntax
- set: direct usage of set operation, like $\cup$, etc.

Inductive predicates can be of type $\tau_1 \Rightarrow \cdots \Rightarrow \tau_n \Rightarrow \text{bool}$

Further aspects

- Rule inversion and inductive cases (see IHT 7.1.5)
- Mutual inductive definitions (see IHT 7.1.6)
- Parameters in inductive definitions (see IHT 7.2)
**Motivation**

**Introduction:**

Inductive definitions can be considered as:

- **Constant definition:** define exactly one set (*semantic interpretation*)
- **Axiom system:** except all sets that satisfy the rules (*axiomatic interpretation*)
- **Derivation system:** show that an element is in a set by applying the rules (*derivational interpretation*)

Isabelle/HOL is based on the semantic interpretation. In addition, it allows to use the rules as part of the derivation system.

**Remark**

The interpretations have advantages and disadvantages/problems.

---

**Illustrating the problems**

**Problem of semantic interpretation:**

We have to assign a set to any well-formed inductive definition.

**Example:**

Which set should be assigned to `fooset`:

```
inductive_set fooset :: "nat set" where
  "n ∈ fooset ⇒ n+1 ∈ fooset"
```

**Problem of derivational interpretation**

The rules of the definition are too weak. E.g., we cannot prove:

```
3 ∉ even
```

---

**“Looseness” of rules**

**Problem of axiomatic interpretation:**

There are usually many sets satisfying the rules of an inductive definition.

**Example:**

The following set `even2` satisfies the rules of `even`:

```
definition even2 :: "nat set" where
  "even2 ≡ { n. n ≠ 1 }"
lemma "0 ∈ even2"
lemma "n ∈ even2 ⇒ n+2 ∈ even2"
```
6. Inductive Definitions and Fixed Points

6.2 Fixed point theory for inductive definitions

Semantics of inductive definition

Definition

Let $f : T \Rightarrow T$ be a function. A value $x$ is called a fixed point of $f$ if $x = f x$.

Semantics approach for inductive definitions

Three steps:

- Transform inductive definition $ID$ into “normalized form”
- “Extract” a fixed point equation for a function $F_{ID} :: nat \Rightarrow nat$
- Take the least fixed point

Assumption

For every (well-formed) inductive definition, the least fixed point exists.

Transformation to “normalized form”

A “normalized” inductive definition has exactly one implication of the form:

\[
\text{inductive_set } S :: \text{"nat set" where} \\
\forall m \in (F_S S) \Rightarrow m \in S
\]

Example:

\[
\text{inductive_set even :: \"nat set" where} \\
\forall \theta \in \text{even} \mid \forall n \in \text{even} \Rightarrow n+2 \in \text{even}
\]

has the normalized form:

\[
\text{inductive_set even :: \"nat set" where} \\
\forall m \in \{m. m=0 \lor (\exists n. n \in \text{nset} \land m=n+2)\} \Rightarrow m \in \text{even}
\]

That is, the function $F_{\text{even}}$ is

\[
F_{\text{even}} \text{nset} = \{m. m=0 \lor (\exists n. n \in \text{nset} \land m=n+2)\}
\]

Supremum and infimum

Definition (Supremum/infimum)

Let $(L, \leq)$ be partially ordered set and $A \subseteq L$.

- **Supremum**: $y \in L$ is called a supremum of $A$ if $y$ is an upper bound of $A$, i.e., $b \leq y$ for all $b \in A$ and $\forall y^' \in L : ((y^' \text{ upper bound of } A) \rightarrow y \leq y^')$

- **Infimum**: analogously defined, greatest lower bound
**Definition (Complete lattice)**
A partially ordered set \((L, \leq)\) is a complete lattice if every subset \(A\) of \(L\) has both an infimum (also called the meet) and a supremum (also called the join) in \(L\).
The meet is denoted by \(\wedge A\), the join by \(\vee A\).

**Lemma**
Complete lattices are non-empty.

**Lemma**
Let \(\mathcal{P}(S)\) be the power set of a set \(S\).
\((\mathcal{P}(S), \subseteq)\) is a complete lattice.

---

**Proof of Knaster-Tarski Corollary**

We prove:
The set of all fixed points \(P\), \(P \subseteq L\), has the following properties:
1. \(\vee P = \vee \{y \in L \mid y \leq F(y)\}\)
2. \((\vee P) \in P\)
3. \(\wedge P = \wedge \{y \in L \mid F(y) \leq y\}\)
4. \((\wedge P) \in P\)
That is, \((\vee P)\) is the greatest and \((\wedge P)\) the least fixed point.

**Proof:**
We show the first two properties. The proof of the third and forth property are analogous.

**Theorem (Knaster-Tarski)**
Let \((L, \leq)\) be a complete lattice and let \(F : L \rightarrow L\) be a monotone function.
Then the set of fixed points of \(F\) in \(L\) is also a complete lattice.

**Corollary (Knaster-Tarski)**
\(F\) has a (unique) least and greatest fixed point.

**Proof of Knaster-Tarski Corollary (2)**

**Show:** \(\vee P = \vee \{y \in L \mid y \leq F(y)\}\) and \((\vee P) \in P\)
Let \(D = \{y \in L \mid y \leq F(y)\}\) and \(u = \vee D\). We show:
\(u \in P\) and \(u = \vee P\), i.e., \(u\) is the greatest fixed point of \(F\).

For all \(x \in D\), also \(F(x) \in D\), because \(F\) is monotone and \(F(x) \leq F(F(x))\).
\(F(u)\) is an upper bound of \(D\), because for \(x \in D\), \(x \leq u\) and \(F(x) \leq F(u)\), i.e., \(x \leq F(x) \leq F(u)\).
As \(u\) is least upper bound, \(u \leq F(u)\). Thus, \(u \in D\).
As shown above, \(u \in D\) implies \(F(u) \in D\), thus \(F(u) \leq u\).
In summary, \(F(u) = u\), i.e., \(u\) is a fixed point, \(u \in P\).
Because \(P \subseteq D\), \(\vee P \leq \vee D\), hence \(u \leq \vee P \leq u\), i.e., \(u = \vee P\).
Remark
Isabelle/HOL handles:
- lattices in Chapter 5 of theory Main
- complete lattices in Chapter 8 of theory Main
- inductive definitions and Knaster-Tarski in Chapter 9

The natural numbers are introduced in Chapter 15, using an inductive definition!

Some related definitions and lemmas in Isabelle/HOL

\[
\text{mono } f \equiv \forall A \ B. \ A \leq B \rightarrow f \ A \leq f \ B
\]
where \( A, B \) are often sets and \( \leq \) is \( \subseteq \)

\[
\text{lfp } f \equiv \inf \{ u \mid f \ u \leq u \}
\]  \hspace{1cm} (lfp_def)

\[
\text{mono } f \implies \text{lfp } f = f \ (\text{lfp } f)
\]  \hspace{1cm} (lfp_unfold)

\[
\llbracket \text{mono } f; f \ (\inf \ (\text{lfp } f) \ P) \leq P \rrbracket \implies \text{lfp } f \leq P
\]  \hspace{1cm} (lfp_induct)

\[
\text{gfp } f \equiv \sup \{ u \mid u \leq f \ u \}
\]  \hspace{1cm} (gfp_def)

\[
\text{mono } f \implies \text{gfp } f = f \ (\text{gfp } f)
\]  \hspace{1cm} (gfp_unfold)

\[
\llbracket \text{mono } f; X \leq f \ (\sup X \ (\text{gfp } f)) \rrbracket \implies X \leq \text{gfp } f
\]  \hspace{1cm} (coinduct)