Declarative Programming and (Co)Induction

Davide Ancona and Elena Zucca

University of Genova

PhD Course, DIBRIS, June 23-27, 2014

Course description

- Induction and conduction: different ways to interprete recursive definitions
- Self-contained introduction to functional and logic programming (languages Haskell and Prolog)
- Semantics and type system of programming languages
- Organized in two modules:
  1. 10 hours: basis for the second
     Induction, small step and big step semantics, lambda calculus, inductive type system, soundness
     Functional programming in Haskell
  2. 10 hours: induction and coinduction, lowest and greatest fixed points, abstract and operational semantics of Prolog and coProlog
     Programming in Prolog and coProlog

First module

1. [Monday 10.30-13] Induction: inductive definitions and proofs by induction
2. [Monday 14.30-17] Functional programming in Haskell + Lab: simple programs in Haskell
3. [Wednesday 10.30-13] Small step and big step semantics, lambda calculus, type system, soundness
4. [Wednesday 14.30-17] Lab: programs in Haskell

Part I
Induction

Induction

What is induction useful for?

- definition of sets whose elements can be generated in a finite number of steps:
  - natural numbers, finite lists, finite trees
  - relations and functions over such sets
- proving properties by the induction principle

Simple examples

- Mathematical style
  The set of even numbers is the least set s.t. (or: the set inductively defined by)
  - 0 is an even number
  - if n is an even number, then $n + 2$ is an even number
- Recursive function definitions in programming languages

  f x = if x == 0 then 0 else f (x-1) + 1

- Syntax of programming languages

$$t \quad ::= \quad \mathtt{true} \mid \mathtt{false} \mid \mathtt{if}\ t\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2 \mid \mathtt{succ}\ t \mid \mathtt{pred}\ t \mid \mathtt{0} \mid \mathtt{iszero}\ t$$

Inference systems

- $\mathcal{U}$ universe
- a rule is a pair $\frac{Pr}{c}$, with $Pr \subseteq \mathcal{U}$ set of premises, $c \in \mathcal{U}$ consequence
- an inference system Φ is a set of rules
- Φ is finitary if, for all $\frac{Pr}{c} \in \Phi$, Pr is finite
- $X \subseteq \mathcal{U}$ is closed w.r.t. $\frac{Pr}{c}$ iff $Pr \subseteq X$ implies $c \in X$
- X is Φ-closed (closed w.r.t. Φ) iff it is closed w.r.t all rules in Φ
- the set $I(\Phi)$ inductively defined by Φ is the intersection of all the Φ-closed sets
- it is easy to see that $I(\Phi)$ is Φ-closed, hence we can equivalently say the least Φ-closed set
- $\mathcal{U}$ is always Φ-closed hence $I(\Phi)$ is well-defined
- given Φ, we can take as universe the set of consequence elements, hence it is not necessary to fix $\mathcal{U}$

Inductive definitions

- an inductive definition is any finite description, in some meta-language, of an inference system Φ, hence of $I(\Phi)$
- typically consisting of a set of meta-rules of the form $\frac{pre}{ce}\,cond$
- *pre*, *ce*, *cond* are expressions with meta-variables
- each meta-rule represents a (possibly infinite) set of rules, one for each assignment of values to the meta-variables satisfiyng *cond*
- meta-rules with empty set of premises are the basis, others are the inductive step of the inductive definition
- however, there are many other styles for giving inductive definitions ...

Example: mathematical style

The set of even numbers is the least set s.t. (or: the set inductively defined by)

- 0 *is an even number*
- *if n is an even number, then $n+2$ is an even number*

- corresponds to the following (meta-)rules, where n ranges over $\mathbb{N}$:

$$\frac{}{0} \quad \frac{n}{n+2}$$

- closed sets: $\{n \mid n \text{ even}\}$, $\{n \mid n \text{ even or } n \geq k\}$ for some $k \in \mathbb{N}$
- non closed sets: e.g., $\emptyset$

Variants

$\frac{n}{n+2}$			*empty set*
$\frac{}{10}$	$\frac{n+1}{n}$		0..10
$\frac{}{0}$	$\frac{n}{n+2}$	$\frac{\{n \mid n \textit{ even}\}}{1}$	$\mathbb{N}$

- it is easy to see that $I(\Phi) \neq \emptyset$ only if there is some rule with empty set of premises

Recursive function definitions in programming languages

f x = if x == 0 then 0 else f (x-1) + 1

- corresponds to the following (meta-)rules, where x, r range over $\mathbb{Z}$:

$$\frac{}{(0,0)} \qquad \frac{(x-1,r)}{(x,r+1)} x \neq 0$$

- (some) closed sets: all the partial identity functions defined from some $x \leq 0$, the total identity function, ...
- exercise: show that $I(\Phi) = \{(x,x) \mid x \geq 0\}$
  - $I(\Phi) \subseteq \{(x,x) \mid x \geq 0\}$ is proved showing that $\{(x,x) \mid x \geq 0\}$ is closed
  - $\{(x,x) \mid x \geq 0\} \subseteq I(\Phi)$ by arithmetic induction

Example: syntax of programming languages

$$\begin{array}{rcl} T & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{if}\ T\ \texttt{then}\ T\ \texttt{else}\ T \\ & & \mid \texttt{0} \mid \texttt{succ}\ T \mid \texttt{pred}\ T \mid \texttt{iszero}\ T \end{array}$$

- corresponds to the following (meta-)rules:

$$\frac{}{\texttt{true}} \quad \frac{}{\texttt{false}} \qquad \frac{t\ t_1\ t_2}{\texttt{if}\ t\ \texttt{then}\ t_1\ \texttt{else}\ t_2}$$

$$\frac{}{0} \quad \frac{t}{\texttt{succ}\ t} \quad \frac{t}{\texttt{pred}\ t} \quad \frac{t}{\texttt{iszero}\ t}$$

- context free grammars correspond to a special class of inductive definitions where premises are distinct metavariables

$$\begin{array}{rcl} t & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{if}\ t\ \texttt{then}\ t_1\ \texttt{else}\ t_2 \\ & & \mid \texttt{0} \mid \texttt{succ}\ t \mid \texttt{pred}\ t \mid \texttt{iszero}\ t \end{array}$$

An alternative view

Definition (Signature)

A signature Σ is a family of operators indexed over natural numbers. If $op \in \Sigma_n$, then we say that op has arity n and write op/n

Definition (Terms over a signature)

Given a signature Σ, the set of terms over Σ or Σ-terms is inductively defined by:
for each operator op with arity n, if $t_1, \ldots, t_n$ are terms, then $op(t_1, \ldots, t_n)$ is a term

- for simplicity we consider the uni-sorted case
- a context-free grammar implicitly defines a signature and, for each operator, a concrete syntax for writing $op(t_1, \ldots, t_n)$, e.g., `if` t `then` t_1 `else` t_2
- the signature is the abstract syntax

Induction principle

Φ inference system, $I(\Phi) \subseteq \mathcal{U}$, $P\colon \mathcal{U} \to \{T, F\}$

Theorem

If for all $\dfrac{Pr}{c} \in \Phi$

$$(\star) \qquad (P(d) = T \text{ for all } d \in Pr) \text{ implies } P(c) = T$$

then $P(d) = T$ *for all* $d \in I(\Phi)$

Proof.

Set $C = \{d | P(d) = T\}$
The condition $(\star)$ can be equivalently written: $Pr \subseteq C$ implies $c \in C$.
That is, C is Φ-closed, hence $I(\Phi) \subseteq C$. $\square$

Remark

If $Pr = \emptyset$, then $(\star)$ is equivalent to $P(c) = T$

Particular case: arithmetic induction

Theorem

P predicate on natural numbers s.t.

- $P(0) = T$
- *for all* $n \in \mathbb{N}$, $P(n) = T$ *implies* $P(n+1) = T$

Then $P(n) = T$ *for all* $n \in \mathbb{N}$.

Proof.

$\mathbb{N}$ can be seen as the set inductively defined by:

- $0 \in \mathbb{N}$
- if $n \in \mathbb{N}$ then $n + 1 \in \mathbb{N}$.

$\square$

Particular case: complete arithmetic induction

Theorem

P predicate on natural numbers s.t.

- $P(0) = T$
- *for all* $n \in \mathbb{N}$, $P(m) = T$ *for all* $m < n$ *implies* $P(n) = T$

Then $P(n) = T$ *for all* $n \in \mathbb{N}$.

Proof.

$\mathbb{N}$ can be seen as the set inductively defined by:

- $0 \in \mathbb{N}$
- if $m \in \mathbb{N}$ for all $m < n$ then $n \in \mathbb{N}$.

$\square$

Particular case: structural induction

Theorem

Σ signature, P predicate on Σ-terms s.t.

$$\text{for all } op \in \Sigma_n,\ P(t_1) = T, \ldots, P(t_n) = T \text{ implies } P(op(t_1, \ldots, t_n)) = T$$

Then $P(t) = T$ for all t term over Σ.

Multiple inference definitions (sketch)

- all previous definitions and results can be generalized to families
- a family of sets A indexed over S (S-family of sets) is a function which associates to each $s \in S$ a set A_s
- also written $\{A_s\}_{s \in S}$
- in a multiple inference system a rule has shape $\dfrac{\{Pr_s\}_{s \in S}}{c : \underline{s}}$
- $I(\Phi)$ is an S-family of sets
- examples: definitions of mutually recursive functions, general form of syntax (many syntactic categories = indexes, many-sorted signature)
- multiple induction principle: Φ multiple inference system, $I(\Phi) \subseteq \mathcal{U}$, $\{P_s\}_{s \in S}$ family of predicates s.t. $P_s : \mathcal{U}_s \to \{T, F\}$

If for all $\dfrac{\{Pr_s\}_{s \in S}}{c : \underline{s}} \in \Phi$

$$(\star) \qquad (P_s(d) = T\ \forall d \in Pr_s, \forall s \in S) \text{ implies } P_{\underline{s}}(c) = T$$

then $P_s(d) = T\ \forall d \in I(\Phi), \forall s \in S$

Inductive definitions as fixed points

- given $f\colon A \to A$ and $a \in A$, a is a fixed point of f iff $f(a) = a$
- given $f\colon \wp(\mathcal{U}) \to \wp(\mathcal{U})$ and $X \subseteq \mathcal{U}$, X is a pre-fixed point of f (X is f-closed) iff $f(X) \subseteq X$
- X is a least pre-fixed point of f iff $f(Y) \subseteq Y$ implies $X \subseteq Y$
  equivalently, X is the intersection of pre-fixed points
- f is monotone if $X \subseteq Y$ implies $f(X) \subseteq f(Y)$

Theorem

Given Φ an inference system with universe $\mathcal{U}$, set $f_\Phi : \wp(\mathcal{U}) \to \wp(\mathcal{U})$ defined by:

$$\text{for each } X \subseteq \mathcal{U},\ f_\Phi(X) = \{c \mid \frac{Pr}{c} \in \Phi, Pr \subseteq X\}$$

Then, f_Φ is monotone and $I(\Phi)$ is the least pre-fixed point of $f_\Phi(X)$.

Theorem

Given $f\colon \wp(\mathcal{U}) \to \wp(\mathcal{U})$ monotone, set Φ_f defined by:

$$\Phi_f = \{\frac{Pr}{c} \mid Pr \subseteq \mathcal{U}, c \in f(Pr)\}$$

Then, $I(\Phi_f)$ is the least pre-fixed point of f.