2012年9月28日
16:51

Part I

Categorical Algebra and Coalgebra.

2012年9月28日
16:52

Chap. 1

Introduction :

Induction and Coinduction.

References:
- Bart Jacobs, Introduction to Coalgebra (a draft of a book, available on his website)
- Dirk Pattinson, NASSLLI 2003 Lecture Notes (available on his website)

§1. Intro.

You all are familiar with $\begin{pmatrix}\text{def.}\\ \text{proof.}\end{pmatrix}$ by induction ...

Ex. 1.1. (Inductive def. of terms)

$\Sigma = (\Sigma_n)_{n\in\omega}$. Σ_n : a set of n-ary operation symbols

The set $T_\Sigma X$ of Σ-terms over variables from X is defined by

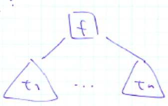
$$T_\Sigma X \ni t ::= \underset{(x\in X)}{x} \mid \underset{(f\in\Sigma_n)}{f(t_1,\dots,t_n)}$$

That is,

$$\frac{}{x\in T_\Sigma X}\ (x\in X)$$

$$\frac{t_1\in T_\Sigma X \quad \dots \quad t_n\in T_\Sigma X}{f(t_1,\dots,t_n)\in T_\Sigma X}\ (f\in\Sigma_n)$$

That is equivalently:

$x \in T_\Sigma X$  $\in T_\Sigma X$

画面の領域の取り込み日時: 2012/07/17 15:50

画面の領域の取り込み日時: 2012/07/17 15:50

Ex. 1.2. (Inductive def. of a function)

Let A be a Σ-algebra with a valuation $J: X\to A$. That is,

- For each $n\in\mathbb{N}$ and $f\in\Sigma_n$,

$$[\![f]\!]_A : A^n \longrightarrow A$$

- $J: X\to A$ is a function.

We define the map

$$[\![-]\!]_J : T_\Sigma X \longrightarrow A$$

by :

- $[\![x]\!]_J := J(x) \quad (x\in \mathbb{V}\mathrm{ar})$
- $[\![f(t_1,\dots,t_n)]\!]_J := \underbrace{[\![f]\!]}_{\boxed{\text{The interpretation of } f\in\Sigma_n \text{ in } A}}\left([\![t_1]\!]_J,\dots,[\![t_n]\!]_J\right)$

That is,

2012年10月2日
12:32

NB If you're not really familiar with (formal) logic, you may be having difficulties understanding the last two pages.

The relevant definitions are given (in more detail) in Chap. 2 of

http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro/courseNotes/2012infLogic/textbook.pdf

Ex. (Proof by Induction)

$t \in T_\Sigma X$ with $FV(t) \subseteq X$

with an obv inductive def.

free var's

$J, J' : X \to A$, valuations

$J(x) = J'(x)$, $\forall x \in FV(t)$

Then $[\![t]\!]_J = [\![t]\!]_{J'}$

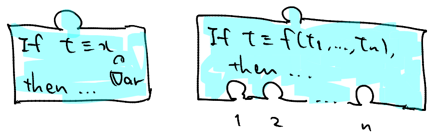
Proof. syntactic eq.

- If $t \equiv x \in Var$, $FV(t) = \{x\}$ and
  $[\![t]\!]_J = J(x) \underset{\text{asmp.}}{=} J'(x) = [\![t]\!]_{J'}$.
- If $t \equiv f(t_1, \dots, t_n)$, $FV(t) := FV(t_1) \cup \dots \cup FV(t_n)$
  $FV(t_i) \subseteq FV(t)$, thus by ind. hyp.
  $[\![t_i]\!]_J = [\![t_i]\!]_{J'}$, $\forall i \in [1, n]$
  Use this in:
  $$[\![t]\!]_J = [\![f]\!]([\![t_1]\!]_J, \dots, [\![t_n]\!]_J) \underset{\text{above}}{=} [\![f]\!]([\![t_1]\!]_{J'}, \dots, [\![t_n]\!]_{J'}) = [\![t]\!]_{J'}$$ ∎

$= [\![t]\!]_{J'}$. ∎

Inductive proof, schematically

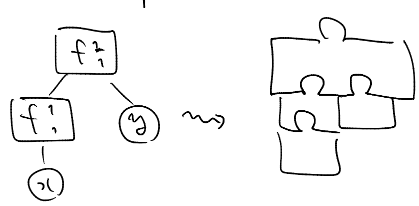
1. I provide the modules for proofs, namely



2. and I leave the use of these to somebody else

"You have a term $t \in T_\Sigma X$; for which you can use the p'f modules I give you. This'll do. Good luck. I'm now going out for beer..."

For example:



We now exhibit a dual notion of <u>Coinduction</u>.

- What is similar:
  * data structure
  * Defs / Pf's by "modules"
- What is <u>not</u> similar:
  * Well-founded vs. non well-fdd
  * Possible infinity vs. <u>real</u> infinity

Let us now turn to data that are "truly infinite" ...

<u>Def. 1.4</u> A <u>stream</u> over L is

$$a_0 a_1 a_2 \ldots \qquad \text{with } a_i \in L$$

that is, a function

$$a : \underset{\omega}{\overset{}{\mathbb{N}}} \longrightarrow L \qquad (\mathbb{N} = \omega)$$

The set of streams over L is denoted by L^ω.

<u>Ex. 1.5</u> (Def. by coinduction)

- The stream $abab\ldots$ $(a, b \in L)$

  What is its "precise" / ("finitary") / "useful" definition?

  Try this:

  * A "state machine" / "automaton"

$$\begin{tikzcd} x \arrow[r, bend left, "a"] & y \arrow[l, bend left, "b"] \end{tikzcd}$$

  * $abab\ldots$ as its "behavior", i.e. the stream it produces (eventually)

- The stream $aa\ldots$ is by

$$\begin{tikzcd} x \arrow[loop right, "a"] \end{tikzcd}$$

(Ex. 1.5 cont'd)

- $\mathrm{alt} : L^\omega \times L^\omega \longrightarrow L^\omega$

  $(a_0 a_1 \ldots, b_0 b_1 \ldots) \longmapsto a_0 b_1 a_2 b_3$.

  is defined, or "implemented", by the "state machine"

$$\begin{tikzcd} (a \cdot \alpha,\ b \cdot \beta) \arrow[d, "a"] \\ (\beta,\ \alpha) \end{tikzcd} \qquad \left(\begin{array}{l} a, b \in L \\ \alpha, \beta \in L^\omega \end{array}\right)$$

E.g.

$$(a_0 a_1 \ldots, b_0 b_1 \ldots) \xrightarrow{a_0} (b_1 b_2 \ldots, a_1 a_2 \ldots) \xrightarrow{b_1} (a_2 a_3 \ldots, b_2 b_3 \ldots) \xrightarrow{a_2} \ldots$$

<u>Point</u>

Represent a stream by means of a certain kind of an automaton

<u>Def.</u> (Stream automaton)

An <u>L-stream automaton</u> is

$$(\underset{\text{a set}}{X},\ \underset{\text{"transition function"}}{c : X \to L \times X})$$

$$\frac{c(x) = (a, x')}{x \xrightarrow{a} x'}$$

"is the same thing as"

Ex. (Proof by coinduction)

Claim. $alt(\alpha, \alpha) = \alpha$

Intuitively obvious:

$\alpha = a_0 a_1 a_2 \ldots$

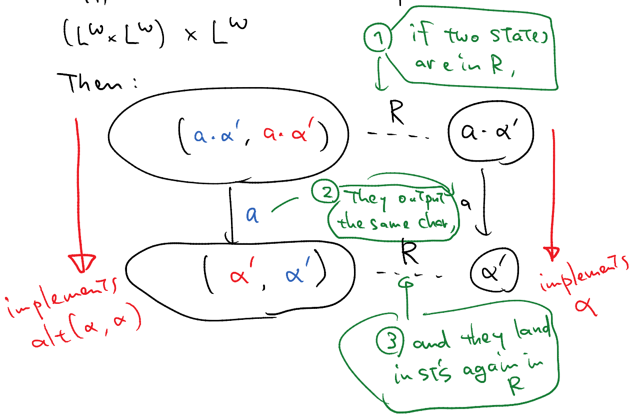
$alt(\alpha, \alpha) = \overline{a_0}\ \underline{a_1}\ \overline{a_2}\ \underline{a_3} \ldots$

But: "precise proof"? (generalizes)

We define

$R = \{ ((\alpha, \alpha), \alpha) \mid \alpha \in L^\omega \}$
$\cap$
$(L^\omega \times L^\omega) \times L^\omega$

Then:

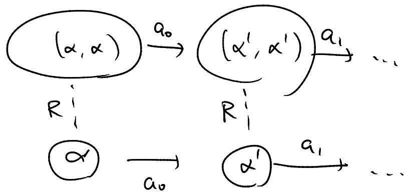


The proof says:

" Here is $R \subseteq (L^\omega \times L^\omega) \times L^\omega$ s.t.

- $(\alpha, \beta) R \gamma \Rightarrow$ the first char's of $\{ alt(\alpha, \beta), \gamma \}$ are the same
- $(\alpha, \beta) R \gamma \Rightarrow$ the next states are again related by R
- $((\alpha, \alpha), \alpha) \in R$

You have time, so you can try



I'm mortal and I prefer going out for beer."

So this R is like a proof module, or a witness.

bisimulation

画面の領域の取り込み日時: 2012/07/17 16:49

This argument with a relation R as an invariant — "initially R, and R is preserved under a transition" — is one of the several different things referred to as coinduction.

They include: the argument using largest fixed pt.
We'll see that they are all instances of the categorical principle of coinduction


クリック なか見! 検索

Advanced Topics in Bisimulation and Coinduction

Edited by Davide Sangiorgi and Jan Rutten

Cambridge

Advanced Topics in Bisimulation and Coinduction (Cambridge Tracts in Theoretical Computer Science) [ハードカバー]

Davide Sangiorgi (編集), Jan Rutten (編集)

この商品の最初のレビューを書き込んでください。 いいね (0)

価格: ￥ 7,795 通常配送無料 詳細

在庫あり。 在庫状況について

この商品は、Amazon.co.jp が販売、発送します。ギフトラッピングを利用できます。

4点在庫あり。ご注文はお早めに。

7/18 水曜日 にお届けします。関東へのお届けの場合、今から16 時間 と 40 分以内に「お急ぎ便」または「当日お急ぎ便」を選択して注文を確定してください(有料オプション。Amazonプライム会員は無料)。関東以外の場合は、注文確定画面でお届け予定日をご確認ください。

Summary

- Induction / Coinduction are two principal methods for dealing with infinity.
  - * But with different flavors ...
    - induction $\Rightarrow$ "possible infinity" well-fdd
    - coinduction $\Rightarrow$ "real infinity" non-well-fdd
  - * CS indeed involve the latter aspect (cf. while loops, μ, ...)
- We now present a categorical formalization of these