Theory of Programming Languages
程序设计语言理论

张昱

Scool of Computer Science and Technology
University of Science and Technology of China

September, 2009

第2章　PCF语言-代数数据类型

[FPL Ch2, Ch3]

简要介绍PCF语言的组成部分之一：布尔类型和自然数类型这些代数数据类型。

介绍定义和研究代数数据类型的一般数学框架——泛代数。

PCF: Programming Computable Functions,可计算函数程序设计语言

PCF语言简介

PCF语言最初由Dana Scott于1969年阐述

A Type-Theoretic Alternative to cuch, iswim, pwhy, Notes,. Oxford (1969). Annotated version in Theor. Comp. Sc. 121, 411-440, 1993.

- 基于λ演算的类型化函数式语言

❖PCF语言的组成

- 代数数据类型：自然数类型和布尔类型
- 纯类型化λ演算：函数类型→ 、积类型×
- 不动点算子：递归类型

PCF语言简介

- 通过列出PCF的类型来概述PCF的语言构造
  - **基本类型：** 自然数类型nat、布尔类型bool
    基本值： 自然数0、1、…，布尔值true、false
  - **类型构造符：** × 和 →
    - 笛卡儿积(cartesian product)$\sigma \times \tau$ 类型、函数类型$\sigma \to \tau$
    - 规则：→自右向左结合， ×的优先级高于→

    例如，nat×nat类型是自然数序对的类型

    nat→nat是论域为nat、值域为nat的函数类型，如自然数上的恒等函数

    nat×nat→nat即(nat×nat)→nat，是二元函数类型，如求2个自然数中的较大值的函数max

    nat→nat→nat即nat→(nat→nat)，是二阶函数类型，它表示一类参数类型为nat,返回类型为一阶函数类型nat→nat函数

本章主题

- 本章介绍 **PCF**语言中代数数据类型部分
  - 语言构造（**constructs**）：语法和语义
    - 自然数类型和布尔类型的表达式
  - 公理语义、操作语义和指称语义以及它们之间的联系

2.1 语言简介-表达式-1

❖布尔类型和自然数类型的表达式

- 布尔类型常量：*true*，*false*
- 布尔类型条件表达式
  if ⟨ *bool_exp* ⟩ **then** ⟨ *bool_exp* ⟩ **else** ⟨ *bool_exp* ⟩
- 自然数类型常量：0，1，2，3，…，（数码 ）
- 自然数类型条件表达式
  if ⟨ *bool_exp* ⟩ **then** ⟨ *nat_exp* ⟩ **else** ⟨ *nat_exp* ⟩
- 自然数相等测试
  Eq? ⟨ *nat_exp* ⟩⟨ *nat_exp* ⟩ 如 *Eq? 5 0 = false*
- ⟨*σ_exp*⟩ :: = … | **if** ⟨*bool_exp*⟩ **then** ⟨*σ_exp*⟩ **else** ⟨*σ_exp*⟩
  尚未列出PCF中所有的自然数类型表达式和布尔类型表达式，因为一些函数调用的结果是自然数或布尔值

2.1 语言简介-表达式-2

❖ 良形的(*well-formed*)表达式

只有满足某种定型约束的表达式才是PCF的合法表达式

- 例如， **true+5不是良形的**
- 对于含有变量的表达式，其定型条件依赖于该表达式所处的上下文（称为定型环境，它由一组变量定型假设 $x_1:\sigma_1,\ldots,x_k:\sigma_k$ 组成）

  例如，对于y+5，若y是nat类型，则y+5是nat类型；若y是bool类型，则y+5不是良形的。

——通过语言的静态语义（类型、值、定型规则）来描述，静态语义预测表达式值将具有的形式。

2.1 语言简介-表达式- 3

❖等式公理

- **0 + 0 = 0, 0 + 1 =1, …, 1 + 0 = 1, 1 + 1 = 2, …**
- 与条件表达式有关的公理模式

  if *true* then *M* else *N* = *M*

  if *false* then *M* else *N* = *N*
- 相等测试也有无数个公理，其形式如下:

  对任意数码***n***，***Eq? n n = true***

  对任意不相同数码***m***和***n***，***Eq? m n = false***

  没有等式公理***Eq? M M = true***

2.1 语言简介-表达式- 4

- 操作语义由一组归约公理来定义
  - 把每个等式公理自左向右定向可得到对应的归约公理
  - 归约公理用于对表达式求值
  - 可归约式(*redex*)：匹配一个归约公理左部的项
  - 例：　　**if *Eq*? (6 +6) 7 then (2+2) else 36**
    - → **if *Eq*? 12 7 then (2 + 2) else 36**
    - → **if *false* then (2 + 2) else 36**
    - → **36**
- 指称语义
  - 选择一个自然数值集*N* 和一个布尔值集*B*
  - 为表达式中的每个自由变元选择一个值，从而给表达式指称一个数学含义

2.1 语言简介-程序和结果-1

❖程序和结果

- PCF语言有2个语法范畴（表达式的种类）：类型和项
  项：具有类型的表达式

$$\tau \quad ::= \text{nat} \mid \text{bool}$$

$$e \quad ::= < nat\_exp > \mid < bool\_exp >$$

$$< nat\_exp > ::= 0 \mid 1 \mid 2 \mid \ldots \mid < nat\_var > \mid < nat\_exp > + < nat\_exp > \mid$$
$$\text{if} < bool\_exp > \text{then} < nat\_exp > \text{else} < nat\_exp >$$

$$< bool\_exp > ::= \text{true} \mid \text{false} \mid < bool\_var > \mid Eq? < nat\_exp >< nat\_exp > \mid$$
$$\text{if} < bool\_exp > \text{then} < bool\_exp > \text{else} < bool\_exp >$$

- 闭项：项中不含自由出现的变元
  自然数类型和布尔类型的闭项具有可观测的值
  函数类型的闭项具有不可观测的值(不能精确说明其究竟定义了什么样的数学函数)

2.1 语言简介-程序和结果-2

❖程序和结果

- 自然数类型和布尔类型是**可观测的(observable)类型**.
- **PCF程序**：是一个良形的(合式的)具有可观测类型的闭项；是一个无须额外的输入并产生一个输出的程序.
- **结果**：计算或执行一个程序所产生的可观测的效果.
- **PCF的结果**：是可观测类型的(闭)范式(normal form). 即，自然数0,1,2,……和布尔常量true和false.
- **语言语义的一般定义**：程序的语义是程序集合与结果集合之间的一种关系——最小情形.
  一般讨论的语义不止是程序的执行结果信息.

2.1 语言简介-公理语义-1

❖ 公理语义

- **公理语义**是一个证明系统.
  用于推导程序及其组成部分的性质.
- **性质的表示形式**
  - 等式
  - 程序在给定输入下的输出断言
  - ……
- 本课程主要讨论等式公理语义

  两个程序在公理语义中等价，只要从它们推导出来的断言完全相同。

2.1 语言简介-公理语义-2

❖ 公理语义

➢ PCF拥有的公理语义的3个一般性质

公理语义决定任一PCF程序有一个结果（输出或可观测效果）. 因为 归约公理是等式公理的子集.

公理语义关于操作语义的可靠性：当2个表达式在公理语义中等价时，在任意程序中可以用其中1个安全地代换另1个，而不改变该程序的操作语义.

公理语义关于指称语义的可靠性：如果能证明任意一对PCF项在公理语义中是等价的，则这些项一定有相同的指称，而不管其中的自由变元被赋予什么样的值.

2.1 语言简介-公理语义-3

❖ 公理语义

➢ PCF的等式证明系统(nat和bool部分)

公理

自反性 (*ref*) $M = M$

加法 (*add*) 0+0=0, 0+1=1, … , 3+5=8, …

相等测试 (*Eq?*) $Eq?\ n\ n = true$, $Eq?\ n\ m = false$

（*n和m是不同的数码*）

条件 (*Cond*) if *true* then M else N = M

if *false* then M else N = N

其中， *M和N是项*

2.1 语言简介-公理语义-4

❖ 公理语义

➢ PCF的等式证明系统(nat和bool部分)

推理规则

等价 (*sym*) $\dfrac{M = N}{N = M}$ (*trans*) $\dfrac{M = N,\ N = P}{M = P}$

同余(Congruence) $\dfrac{M = N,\ P = Q}{M + P = N + Q}$ $\dfrac{M = N,\ P = Q}{Eq?\,M\,P = Eq?\,N\,Q}$

$$\frac{M_1 = M_2,\ N_1 = N_2,\ P_1 = P_2}{\text{if } M_1 \text{ then } N_1 \text{ else } P_1 = \text{if } M_2 \text{ then } N_2 \text{ else } P_2}$$

2.1 语言简介-指称语义-1

❖指称语义

- 指称语义是一个模型.
- 为每个类型选择一个值集来给出这种类型的项的指称.
- ***nat***的语义论域是自然数值集$N \cup \{\perp_{nat}\}$.
  $\perp_{nat}$表示不终止
- ***bool***的语义论域是布尔值集$B \cup \{\perp_{bool}\}$.
- ***nat***和***bool***类型的任何项被指派到相应语义论域上的一个值，也称为它们的语义解释.
- 环境η是指从变量到值的映射.
  若$x:\sigma$, $\eta(x)$是类型σ的值集上的元素.

2.1 语言简介-指称语义-2

❖指称语义

- 通过归纳可以定义项M在环境η中的含义$[\![M]\!]_\eta$
  - $[\![x]\!]_\eta$ 是变量x在环境η中的值
- 指称语义的一般性质

指称语义是可合成的(compositional).任何表达式的含义由它的子表达式的含义决定.

$$[\![\text{if } B \text{ then } M \text{ else } N]\!]_\eta = \begin{cases} [\![M]\!]_\eta & \text{if } [\![B]\!]_\eta \text{ is true} \\ [\![N]\!]_\eta & \text{if } [\![B]\!]_\eta \text{ is false} \\ \perp & \text{otherwise} \end{cases}$$

如果B'、M'和N'同B、 M和N分别有同样的指称，那么

$$[\![\text{if } B \text{ then } M \text{ else } N]\!]_\eta = [\![\text{if } B' \text{ then } M' \text{ else } N']\!]_\eta$$

2.1 语言简介-指称语义-3

❖指称语义

➢指称语义的一般性质

公理语义关于指称语义的可靠性：如果能证明项M和N在公理语义中有相同的断言，则项M和N在指称语义中一定有相同的含义.

由公理语义关于操作语义的可靠性，如果项M能归约到项N，则项M和N在指称语义中一定有相同的含义.

2.1 语言简介-操作语义-1

❖操作语义

➢操作语义的表示形式

（数学上）是一个证明系统，用于推导计算的最终结果，或用于对一个表达式进行一步步变换.

（实际实现上）定义一个抽象机，通过一系列机器状态的向前进展来计算程序.

（实际实现上）解释器或编译器.

2.1 语言简介-操作语义-2

❖操作语义

➢ PCF的操作语义：归约公理

加法 (*add*) 0+0→ 0, 0+1→ 1, … , 3+5→ 8, …

相等测试 (*Eq?*) *Eq? n n* → *true*, *Eq? n m* → *false*

（*n*和*m*是不同的数码）

条件 (*Cond*) if *true* then *M* else *N* → *M*

if *false* then *M* else *N* → *N*

其中*M* → *N*表示1个计算步, 并且*N*比*M*短

➢ 由归约系统定义部分计算函数： *eval*(*M*)=*N*

当且仅当*M* 经若干步（含零步）可以归约到范式 *N*.

➢ 这种操作语义相对抽象，它没有指定计算次序.

2.1 语言简介-操作语义-3

❖操作语义

➢PCF的操作语义的性质

如果 *eval*(*M*)=*N*，则*N* 或者是自然数，或者是布尔常量，取决于*M*的类型.

如果 *eval*(*M*)=*N*，则*M*和*N* 在公理语义和指称语义中都是等价的.

如果 *M*是一个程序并且*M*有与其结果*N* 相同的指称，则在操作语义中执行程序*M*的结果是*N*.

泛代数和代数数据类型简介-1

- 代数数据类型(algebraic datatype)
  - 包括一个或多个值集以及一组在这些集合上的函数.
  - 函数不能以函数作为变元.
  - 基本“类型”(*type*)符号被称为“类别” (*sort*).
- 泛代数（也叫做等式逻辑）
  - 定义和研究代数数据类型的一般数学框架.
  - 代数数据类型的**公理语义**：项之间的一组等式.
  - 代数数据类型的**指称语义**：代数.
    包括一组集合(每个对应一个类别)和一组函数(每个对应项中使用的函数符号).
  - 代数项的**操作语义**：有向的代数等式，归约公理又称重写规则.

泛代数和代数数据类型简介-2

2.2 代数、基调和项

代数数据类型的指称语义：代数项(语法)及其在代数中的解释(指称语义)

2.3 等式、可靠性和完备性

代数数据类型的公理语义及其与指称语义的等价

2.4 同态和初始性

同态是一个代数到另一个代数的保结构的映射（翻译）

2.5 代数数据类型

数据类型公理化方法的一般特征

2.6 重写系统

代数项的归约系统

2.2 代数、基调和项-代数

❖ 代数(*algebra*)

一个代数由一个或多个集合（称为载体*carrier*），以及一组特征元素和一阶函数(也称代数函数)组成.

例，$\mathcal{N} = \langle N, 0, 1, +, * \rangle$

- 载体N是自然数集合.
- 特征元素$0, 1 \in N$，也叫做零元函数.
- 函数$+, * : N \times N \to N$

例，多个载体的例子

$$\mathcal{A}_{PCF} = \langle N, B, 0, 1, +, true, false, Eq?, \ldots \rangle$$

2.2 代数、基调和项-代数项的语法-1

❖代数项的语法

- 由基本符号及其类型来定，这些信息收集在基调中.

基调 $\Sigma = \langle S, F \rangle$

- S是一个集合，其元素叫做**类别**
- F是一组函数符号$f: s_1 \times \ldots \times s_k \to s$的集合，其中表达式$s_1 \times \ldots \times s_k \to s$是$S$上的一阶函数类型.

  当k=0时，$f: s$称为零元函数符号，也称作**常量符号**.

例**2.1**，$\Sigma_{\mathcal{N}} = \langle S, F \rangle$

sorts : *nat*

fctns : 0, 1 : *nat*

$+, * : nat \times nat \to nat$

2.2 代数、基调和项-代数项的语法-2

含变量的代数项

假设有一个无穷的符号集合$\mathcal{V}$，其元素称为变量.

类别指派$\Gamma = \{x_1 : s_1, \ldots, x_k : s_k\}$是一个有限集，一个变量只有一个类别.

给定基调$\Sigma = \langle S, F \rangle$和使用$S$中类别的类别指派$\Gamma$，在$\Sigma$和$\Gamma$上的类别$s$的代数项集合$Terms^s(\Sigma, \Gamma)$定义为:

- 如果$x : s \in \Gamma$，那么$x \in Terms^s(\Sigma, \Gamma)$.
- 如果$f : s_1 \times \ldots \times s_k \to s \in F$并且$M_i \in Terms^{s_i}(\Sigma, \Gamma)$ $(i = 1, \ldots, k)$，那么$f\, M_1 \ldots M_k \in Terms^s(\Sigma, \Gamma)$.
- 当$k = 0$时，如果$f : s \in F$，那么$f \in Terms^s(\Sigma, \Gamma)$.

2.2 代数、基调和项-代数项的语法-3

例，$0, 0+1 \in Terms^{nat}(\Sigma_{\mathcal{N}}, \varnothing)$

$0+x \in Terms^{nat}(\Sigma_{\mathcal{N}}, \Gamma)$，其中$\Gamma = \{x : nat, \ldots\}$

代换

- 代数项中无约束变元，故$[N/x]M$就是简单地把M中x的每个出现都用N代替
- 记号　$\Gamma, x : s' = \Gamma \cup \{x : s'\}$

引理2.1 如果$M \in Terms^{s}(\Sigma, \Gamma, x : s')$且$N \in Terms^{s'}(\Sigma, \Gamma)$，那么$[N/x]M \in Terms^{s}(\Sigma, \Gamma)$.

证明　按$Terms^{s}(\Sigma, \Gamma)$中项的结构进行归纳来证明.

2.2 代数、基调和项-代数项的语法-4

例2.2 用基调$\Sigma_{stk} = \langle S, F \rangle$来写自然数和栈表达式

sorts : *nat*, *stack*

fctns : 0, 1, 2, … : *nat*

$+, * : nat \times nat \to nat$

$empty : stack$

$push : nat \times stack \to stack$

$pop : stack \to stack$

$top : stack \to nat$

push 2 (*push* 1 (*push* 0 *empty*))是该基调的项

$push\ 1\ (push\ 0\ s) \in Terms^{stack}(\Sigma_{stk}, \{s{:}stack\})$

2.2 代数、基调和项-代数项的语义-1

- 代数是为代数项提供含义(指称语义)的数学结构
- Σ是一个基调，则Σ代数$\mathcal{A}$包含
  - 对每个类别符号$s \in S$，正好有一个载体A^s.
  - 一个解释映射I

把函数$I(f): A^{s1} \times \ldots \times A^{sk} \to A^s$指派给函数符号

$$f: s_1 \times \ldots \times s_k \to s \in F .$$

把$I(f) \in A^s$指派给常量符号$f: s \in F$.

例，$\Sigma_{\mathcal{N}}$代数$\mathcal{N}$写成

$$\mathcal{N} = \langle N, 0^{\mathcal{N}}, 1^{\mathcal{N}}, +^{\mathcal{N}}, *^{\mathcal{N}} \rangle$$

2.2 代数、基调和项-代数项的语义-2

- 含变量的代数项的含义(指称语义)
  - 代数$\mathcal{A}$的环境η：$\mathcal{V} \to \cup_s A^s$.
  - 环境η满足Γ，如果对每个$x:s \in \Gamma$都有$\eta(x) \in A^s$.
  - $M \in Terms^s(\Sigma, \Gamma)$的含义$\mathcal{A}[\![M]\!]_\eta$定义为
    - $\mathcal{A}[\![x]\!]_\eta = \eta(x)$
    - $\mathcal{A}[\![f\ M_1 \ldots M_k]\!]_\eta = f^{\mathcal{A}}(\mathcal{A}[\![M_1]\!]_\eta, \ldots, \mathcal{A}[\![M_k]\!]_\eta)$
    - 若$f:s$是常量符号，则$\mathcal{A}[\![f]\!]_\eta = f^{\mathcal{A}}$

  如果在上下文中$\mathcal{A}$是清楚的，则省略$\mathcal{A}$而写成$[\![M]\!]_\eta$

  如果M不含变量，则省略η而写成$\mathcal{A}[\![M]\!]$.

例2.3，将例2.1中基调上的项解释到前面定义的代数$\mathcal{N}$.

若$\eta(x) = 0^{\mathcal{N}}$

$$[\![x+1]\!]_\eta = +^{\mathcal{N}}([\![x]\!]_\eta, [\![1]\!]_\eta) = +^{\mathcal{N}}(\eta(x), 1^{\mathcal{N}}) = +^{\mathcal{N}}(0^{\mathcal{N}}, 1^{\mathcal{N}}) = 1^{\mathcal{N}}$$

2.2 代数、基调和项-代数项的语义-3

例2.4 例2.2的基调$\Sigma_{stk}=\langle S, F\rangle$的代数$\mathcal{A}_{stk}$

$\mathcal{A}_{stk}=\langle N, N^*, 0^{\mathcal{A}}, 1^{\mathcal{A}}, \ldots, +^{\mathcal{A}}, *^{\mathcal{A}}, empty^{\mathcal{A}}, push^{\mathcal{A}}, pop^{\mathcal{A}}, top^{\mathcal{A}}\rangle$

$empty^{\mathcal{A}}=\varepsilon$, 空序列　　$push^{\mathcal{A}}(n, s)=n :: s$

$pop^{\mathcal{A}}(n :: s)=s$　　$pop^{\mathcal{A}}(\varepsilon)=\varepsilon$

$top^{\mathcal{A}}(n :: s)=n$　　$top^{\mathcal{A}}(\varepsilon)=0^{A}$

若η把$x:nat$映射到$3^{\mathcal{A}}$，把$s:stack$映射到$\langle 2^{\mathcal{A}}, 1^{\mathcal{A}}\rangle$

$$\begin{aligned}[\![pop(push\ x\ s)]\!]_{\eta} &= pop^{\mathcal{A}}(push^{\mathcal{A}}([\![x]\!]_{\eta}, [\![s]\!]_{\eta}))\\ &= pop^{\mathcal{A}}(push^{\mathcal{A}}(3^{\mathcal{A}}, \langle 2^{\mathcal{A}}, 1^{\mathcal{A}}\rangle))\\ &= pop^{\mathcal{A}}(\langle 3^{\mathcal{A}}, 2^{\mathcal{A}}, 1^{\mathcal{A}}\rangle)\\ &= \langle 2^{\mathcal{A}}, 1^{\mathcal{A}}\rangle\end{aligned}$$

2.2 代数、基调和项-代数项的语义-4

引理2.2 令$\mathcal{A}$是一个Σ代数，$M \in Terms^s(\Sigma, \Gamma)$，并且$\eta$是满足$\Gamma$的环境，那么$[\![M]\!]_\eta \in \mathcal{A}^s$.

证明 按$Terms^s(\Sigma, \Gamma)$中项的结构进行归纳.

引理2.3 令$x_1, \ldots, x_k$是由出现在$M \in Terms^s(\Sigma, \Gamma)$中的所有变量构成的变量表，$\eta_1$和$\eta_2$是满足$\Gamma$的两个环境，并且对$i = 1, \ldots, k$有$\eta_1(x_i) = \eta_2(x_i)$，那么$[\![M]\!]_{\eta_1} = [\![M]\!]_{\eta_2}$.

证明 按$Terms^s(\Sigma, \Gamma)$中项的结构进行归纳.

2.2 代数、基调和项-代换引理

代换引理

引理2.4 令$M \in Terms^s(\Sigma, \Gamma, x : s')$且$N \in Terms^{s'}(\Sigma, \Gamma)$，那么$[N/x]M \in Terms^s(\Sigma, \Gamma)$。并且对任何环境$\eta$，有

$$[\![[N/x]M]\!]_\eta = [\![M]\!]_{(\eta[x \mapsto a])},$$

其中$a = [\![N]\!]_\eta$是N在η下的含义.

证明 按项M的结构进行归纳.

2.3 等式、可靠性和完备性

2.3.1 等式

代数数据类型的公理语义

2.3.2 项代数：一种特殊的代数

类别s的项集作为s的载体，$I(f)(M_1,\ldots,M_k) = f\,M_1\,\ldots\,M_k$

2.3.3 语义蕴含和等式证明系统

公理语义关于指称语义的可靠性

2.3.4 完备性的形式

最弱形式，演绎完备性，最小模型完备性

2.3.5 同余、商和演绎完备性

可以有空载体的代数的演绎完备性：每个语义蕴含的等式都可证

2.3.6 非空类别和最小模型性质

没有空载体的代数有最小模型完备性

2.3.1 等式-1

- 代数数据类型的公理语义由基调上的项之间的一组等式给出.
- 代数规范：一个基调 + 一组等式
  - 使用等式证明系统推导项之间的新的等式，或者调查什么样的代数满足这些等式强加的要求.

代数证明系统的可靠性与完备性

- **可靠性(soundness)**：从规范可证明的等式在任何满足该规范的代数中都成立.
- **完备性(completeness)**：在满足规范的所有代数中都成立的等式都可从该规范证明.

2.3.1 等式-2

- 允许空载体，从而每个代数规范都有初始代数

  空载体提出了一个技术问题

  - 逻辑公式
    - 若$A = \varnothing$，则公式$\forall x{:}A.\ F(x)$为真
    - 若$A = \varnothing$，则公式$\exists x{:}A.\ F(x)$为假
  - $M \in Terms^s(\Sigma, \Gamma)$的含义仅在环境$\eta$满足$\Gamma$时才被定义，如果$x : s' \in \Gamma$且$A^{s'}$为空，则任何环境都不能满足$\Gamma$，从而不清楚$Terms^s(\Sigma, \Gamma)$中的任意一对项是否相等或这样的等式是否有意义.

    最简单的理论是认为如果A^s为空，则任意一对在类别指派中含有$x : s'$的项是相等的.
  - 只有一个类别时，假定该类别非空是有意义的.

2.3.1 等式-3

- 等式是公式$M = N[\Gamma]$　对某个s，M、$N \in Terms^s(\Sigma, \Gamma)$
  - 如果η满足Γ，那么若$[\![M]\!]_\eta = [\![N]\!]_\eta$，就说代数$\mathcal{A}$在环境$\eta$下满足$M = N[\Gamma]$，写成$\mathcal{A}, \eta \models M = N[\Gamma]$
  - 若$\mathcal{A}$在任何环境下都满足$M = N[\Gamma]$，写成$\mathcal{A} \models M = N[\Gamma]$
  - 如果一类代数C中的任何一个代数$\mathcal{A}$都满足$M = N[\Gamma]$，写成$C \models M = N[\Gamma]$

例2.5 令Σ有两个类别a和b，令$\mathcal{A}$是一个Σ代数，其中$A^a = \{0,1\}$并且$A^b = \emptyset$.

$\mathcal{A}$不可能满足$x = y[x : a, y : a]$，即$\mathcal{A} \models x = y[x : a, y : a]$不成立

但是，$\mathcal{A} \models x = y[x : a, y : a, z : b]$无意义地成立，因为不可能在$A^b$上对变量$z$作出任何指派.

2.3.1 等式-4

- 等式是公式$M = N[\Gamma]$　对某个s，M、$N \in Terms^s(\Sigma, \Gamma)$
  - 任何一个Σ代数都满足等式$M = N[\Gamma]$，写成$\models M = N[\Gamma]$，称等式$M = N[\Gamma]$是**永真**的、**有效**的.
    例如，$x = x\,[x{:}s]$是有效的.
  - 如果$\mathcal{A}$满足Σ上的所有等式，就说Σ代数$\mathcal{A}$是平凡的.

习题2.6 代数$\mathcal{A}$是平凡的，当且仅当任何载体都是空集或是仅有一个元素的集合.

2.3.2 项代数-1

- 项代数$Terms(\Sigma, \Gamma)$是一个Σ代数

$Terms(\Sigma, \Gamma) = \langle \{Terms^s(\Sigma, \Gamma)\}, \mathcal{I} \rangle, \quad \Sigma = \langle S, F \rangle$

  - S中类别s的载体：集合$Terms^s(\Sigma, \Gamma)$
  - F中函数符号$f : s_1 \times \ldots \times s_k \to s$的解释

$$\mathcal{I}(f)(M_1, \ldots, M_k) = f\, M_1 \ldots M_k$$

$Terms(\Sigma, \Gamma)$既用来表示项集，也用来表示项代数.

用代换描述项代数中项的含义:

  - 项代数$Terms(\Sigma, \Gamma)$的环境η也叫做一个代换.
  - 如果S是代换，则用SM表示同时把M中的各个变量x用项Sx替换的结果.
  - 因此用ηM表示把代换η作用于M的结果.

2.3.2 项代数-2

例2.6 令$\Sigma = \langle \{u\}, \{f : u \to u, g : u \times u \to u\} \rangle$，
项代数$\mathcal{T} = Terms(\Sigma, \Gamma)$，其中 $\Gamma = \{x : u, y : u, z : u\}$

- u的载体是类别u在a, b, c以及函数f和g上的所有项的集合

  $\mathcal{T}^u = \{a, b, c, f\,a, f\,b, f\,c, g\,a\,a, g\,a\,b, g\,a\,c, g\,b\,b, \ldots,$
  $g\,(f\,(f\,a))\,(f\,(g\,b\,c)), \ldots\}$
- 函数符号f的解释$f^{\mathcal{T}}$是把任何项M映射到$f\,M$的函数，$g^{\mathcal{T}}$类似.
- 若环境η把变量x映射到a，把y映射到$f\,b$，则

  $\mathcal{T}[\![g\,(f\,x)\,(f\,y)]\!]_\eta = g(f\,a)\,(f\,(f\,b))$

2.3.2 项代数-3

引理2.5 令$M \in Terms(\Sigma, \Gamma)$，并且$\eta$是满足$\Gamma$的项代数$Terms(\Sigma, \Gamma)$的环境，那么$[\![M]\!]_\eta = \eta M$.

证明 对项M的结构进行归纳 .

从项代数可以知道，只有M和N是语法上相同的项时，等式$M = N[\Gamma]$才会永真.

2.3.3 语义蕴含和等式证明系统-1

- 代数规范$Spec = \langle \Sigma, \mathcal{E} \rangle$：一个基调$\Sigma$和一组等式$\mathcal{E}$
- $\mathcal{E}$语义蕴含等式 $M = N[\Gamma]$,写成$\mathcal{E} \models M = N[\Gamma]$，如果

  满足$\mathcal{E}$的所有Σ代数都满足等式$M = N[\Gamma]$

  - 在所有满足代数规范$Spec = \langle \Sigma, \mathcal{E} \rangle$ 的代数中成立的等式是由$\mathcal{E}$语义蕴含的Σ等式.
- $\mathcal{E}$是一个语义理论，如果

  $\mathcal{E} \models M = N[\Gamma]$蕴含$M = N[\Gamma] \in \mathcal{E}$.即$\mathcal{E}$在语义蕴含下封闭.
- 代数$\mathcal{A}$的理论$Th(\mathcal{A})$

  是在$\mathcal{A}$中成立的所有等式的集合.

  习题2.11 证明一个代数的理论是一个语义理论.

2.3.3 语义蕴含和等式证明系统-2

- 证明系统的可靠性
  - 若一个等式从一组假设$\mathcal{E}$是可证明的，那么$\mathcal{E}$语义上蕴涵该等式.
- 证明系统的完备性
  - 若$\mathcal{E}$语义上蕴涵一个等式，那么该等式可从$\mathcal{E}$证明.

接下来，证明代数证明系统的可靠性，2.3.5节证明完备性.

2.3.3 语义蕴含和等式证明系统-3

❖代数证明系统

➢语义相等是等价关系.因此有

公理

(*ref*) $M = M[\Gamma]$

推理规则

(*sym*) $\dfrac{M = N[\Gamma]}{N = M[\Gamma]}$ (*trans*) $\dfrac{M = N[\Gamma],\ N = P[\Gamma]}{M = P[\Gamma]}$

增加类别指派的规则

(*add var*) $\dfrac{M = N[\Gamma]}{M = N[\Gamma, x : s]}\ x \text{ not in } \Gamma$

x不在Γ中

等价代换

(*subst*) $\dfrac{M = N[\Gamma, x : s],\ P = Q[\Gamma]}{[P / x]M = [Q / x]N[\Gamma]}\ P, Q \in Terms^{s}(\Sigma, \Gamma)$

2.3.3 语义蕴含和等式证明系统-4

等式$M = N[\Gamma]$从一组等式集合$\mathcal{E}$可证明，记为$\mathcal{E} \vdash M = N[\Gamma]$

- 仅当利用推理规则*(sym)*、*(trans)*、*(add var)*、*(subst)*可从$\mathcal{E}$中的等式和*(ref)*公理的实例推导出 $M = N[\Gamma]$.

$\mathcal{E}$是一个语法理论

- 如果$\mathcal{E} \vdash M = N[\Gamma]$蕴含$M = N[\Gamma] \in \mathcal{E}$,即$\mathcal{E}$封闭于可证明性.

$\mathcal{E}$的语法理论写成$Th(\mathcal{E})$

- 它是从$\mathcal{E}$可证明的所有等式的集合.

等式集合$\mathcal{E}$是语义一致的

- 如果存在某个等式$M = N[\Gamma]$，它不能由$\mathcal{E}$语义蕴含.

等式集合$\mathcal{E}$是语法一致的

- 如果存在某个等式$M = N[\Gamma]$，它不能从$\mathcal{E}$证明.

2.3.3 语义蕴含和等式证明系统-5

例2.7 例2.2(定义Σ_{stk})和例2.4(定义$\mathcal{A}_{stk}$)的延续

在基调$\Sigma_{stk} = \langle S, F \rangle$上增加在代数$\mathcal{A}_{stk}$中成立的等式:

$$top(\ push\ x\ s) = x[s{:}stack, x{:}nat]$$

$$pop(\ push\ x\ s) = s[s{:}stack, x{:}nat]$$

使用这些等式可以证明等式

$$top(push\ 3\ empty) = 3$$

证明

$$\dfrac{\dfrac{top(push\,x\,s) = x[s : stack, x : nat], \overbrace{empty = empty[x : nat]}^{(\ ref\ )}}{top(push\,x\,empty) = x[x : nat]} \qquad 3 = 3[\ ]}{top(push\,3\,empty) = 3[\ ]}$$

2.3.3 语义蕴含和等式证明系统-5

例2.7 例2.2(定义Σ_{stk})和2.4(定义$\mathcal{A}_{stk}$)的延续

在基调$\Sigma_{stk} = \langle S, F \rangle$上增加在代数$\mathcal{A}_{stk}$中成立的等式:

$top(push\ x\ s) = x[s{:}stack, x{:}nat]$

$pop(push\ x\ s) = s[s{:}stack, x{:}nat]$

使用这些等式可以证明等式

$top(push\ 3\ empty) = 3$

证明

(*subst*) 代换项中的*s*

$$\frac{\dfrac{top(push\,x\,s) = x[s:stack, x:nat],\ empty = empty[x:nat]}{top(push\,x\,empty) = x[x:nat]} \qquad 3 = 3[\,]}{top(push\,3\,empty) = 3[\,]}$$

2.3.3 语义蕴含和等式证明系统-5

例2.7 例2.2(定义Σ_{stk})和2.4(定义$\mathcal{A}_{stk}$)的延续

在基调$\Sigma_{stk} = \langle S, F \rangle$上增加在代数$\mathcal{A}_{stk}$中成立的等式:

$$top(\ push\ x\ s) = x[s{:}stack, x{:}nat]$$

$$pop(\ push\ x\ s) = s[s{:}stack, x{:}nat]$$

使用这些等式可以证明等式

$$top(push\ 3\ empty) = 3$$

证明

$$\frac{top(push\,x\,s) = x[s : stack, x : nat],\ empty = empty[x : nat]}{\dfrac{top(push\,x\,empty) = x[x : nat] \qquad 3 = 3[\,]\ (ref)}{top(push\,3\,empty) = 3[\,]}}$$

2.3.3 语义蕴含和等式证明系统-5

例2.7 例2.2(定义Σ_{stk})和2.4(定义$\mathcal{A}_{stk}$)的延续

在基调$\Sigma_{stk} = \langle S, F \rangle$上增加在代数$\mathcal{A}_{stk}$中成立的等式:

$$top(\ push\ x\ s) = x[s{:}stack, x{:}nat]$$

$$pop(\ push\ x\ s) = s[s{:}stack, x{:}nat]$$

使用这些等式可以证明等式

$$top(push\ 3\ empty) = 3$$

证明

$$\dfrac{\dfrac{top(push\,x\,s) = x[s:stack, x:nat],\ empty = empty[x:nat]}{top(push\,x\,empty) = x[x:nat]} \qquad 3 = 3[\ ]}{top(push\,3\,empty) = 3[\ ]}$$

(*subst*) 代换项中的x

2.3.3 语义蕴含和等式证明系统-6

证明系统的导出规则 $\dfrac{antecedent}{consequent}$

- 例如，$\dfrac{M = N[\Gamma], N = P[\Gamma], P = Q[\Gamma]}{M = Q[\Gamma]}$

- 再如，同余规则(相等的项应用于相等的项时，产生相等的项)

$$(cong)\quad \frac{M_1 = N_1[\Gamma], \ldots, M_k = N_k[\Gamma]}{f\, M_1 \ldots M_k = f\, N_1 \ldots N_k[\Gamma]}\ f : s_1 \times \ldots \times s_k \to s \text{ and } M_i, N_i \in Terms^{s_i}(\Sigma, \Gamma)$$

是使用(*add var*)、(*ref*)、(*subst*)的导出规则

$$M_i = N_i[\Gamma, x_1 : s_1, \ldots, x_k : s_k]$$

$$f\, x_1 \ldots x_k = f\, x_1 \ldots x_k[\Gamma, x_1 : s_1, \ldots, x_k : s_k]$$

$$f\, M_1 \ldots M_k = f\, N_1 \ldots N_k[\Gamma]$$

2.3.3 语义蕴含和等式证明系统-7

证明系统的导出规则 $\dfrac{antecedent}{consequent}$

例2.9，下面的导出规则允许从任何等式的类别指派中删除多余变量

$$\frac{M = N[\Gamma, x : s]}{M = N[\Gamma]}$$ M和N中没有x，$Terms^s(\Sigma, \Gamma)$非空

因为$Terms^s(\Sigma, \Gamma)$非空，故存在某个$P \in Terms^s(\Sigma, \Gamma)$，使得

$$P = P[\Gamma]$$

假设 $M = N[\Gamma, x{:}s]$，用**(subst)**可以证明$[P/x]M = [P/x]N[\Gamma]$,

因为变量x 在M和N中不出现，故有$M = N[\Gamma]$

2.3.3 语义蕴含和等式证明系统-8

定理2.6（可靠性）如果$\mathcal{E} \vdash E$，那么$\mathcal{E} \models E$.

证明　可以根据该证明的长度进行归纳

- **基本情形**：长度为1的证明，它是公理或$\mathcal{E}$的一个等式. 不管是哪种情况，任何满足$\mathcal{E}$的代数$\mathcal{A}$一定满足该等式.
- **归纳情形**：假设E的最后一步证明是从$E_1, \ldots, E_n$得到. 由归纳假设，如果$\mathcal{A} \models \mathcal{E}$，则$\mathcal{A}$ 满足$E_1, \ldots, E_n$.
  要证明：如果$\mathcal{A}$ 满足最后一条规则的这些前提，那么$\mathcal{A}$ 满足E.
  证明 根据证明规则的集合，分情况进行分析.

2.3.3 语义蕴含和等式证明系统-9

定理2.6（可靠性）如果$\mathcal{E} \vdash E$，那么$\mathcal{E} \models E$.

要证明：如果$\mathcal{A}$满足最后一条规则的这些前提，那么$\mathcal{A}$满足E.

证明 根据证明规则的集合，分情况进行分析.

如针对(*subst*) $\dfrac{M = N[\Gamma, x:s],\ P = Q[\Gamma]}{[P/x]M = [Q/x]N[\Gamma]}$ $P, Q \in Terms^s(\Sigma, \Gamma)$

假设$\mathcal{A} \models M = N[\Gamma, x{:}s]$，$\mathcal{A} \models P = Q[\Gamma]$. η是满足Γ的任意环境，设$a = [\![P]\!]_\eta = [\![Q]\!]_\eta$，环境$\eta[x \mapsto a]$满足$\Gamma, x{:}s$.

由代换引理2.4, $[\![[P/x]M]\!]_\eta = [\![M]\!]_{(\eta[x \mapsto a])}$

$[\![[Q/x]N]\!]_\eta = [\![N]\!]_{(\eta[x \mapsto a])}$

由于$\mathcal{A} \models M = N[\Gamma, x{:}s]$，故有 $[\![M]\!]_{(\eta[x \mapsto a])} = [\![N]\!]_{(\eta[x \mapsto a])}$

从而，$[\![[P/x]M]\!]_\eta = [\![[Q/x]N]\!]_\eta$

2.3.3 语义蕴含和等式证明系统-10

命题2.7　存在一个代数理论$\mathcal{E}$和不含x的项 M和N，使得$\mathcal{E} \vdash M = N[\Gamma, x{:}s]$，但是$\mathcal{E} \nvdash M = N[\Gamma]$.

证明　令基调有类别a和b，函数符号$f : a \to b$和$c, d : b$.

令$\mathcal{E}$是包含能从$f\,x = c[x : a]$和$f\,x = d\,[x : a]$证明的所有等式的理论，显然$c = d\,[x : a] \in \mathcal{E}$（ **(trans)** ）

可以找到一个使等式$c = d\,[\varnothing]$不成立的模型：

a的载体为空，c和d指称到b对应的非空载体中的2个不同的元素.

由可靠性，$c = d\,[\varnothing]$是不可能从$\mathcal{E}$证明的.

如何判断等式E是否由$\mathcal{E}$语义蕴含?

1)从$\mathcal{E}$证明E，或者2)找出1个满足$\mathcal{E}$而不满足E的代数.

2.3.3 语义蕴含和等式证明系统-11

例2.10 栈代数规范.

sorts : *nat, stack*

fctns : $0, 1, 2, \ldots : nat$ $\quad +, * : nat \times nat \to nat$

$empty : stack$ $\quad push : nat \times stack \to stack$

$pop : stack \to stack$ $\quad top : stack \to nat$

eqns : $[\, s : stack; x : nat]$

$0 + 0 = 0,\ 0 + 1 = 1, \ldots$ $\quad 0 * 0 = 0,\ 0 * 1 = 0, \ldots$

$top\ (push\ x\ s\ ) = x$ $\quad pop\ (push\ x\ s\ ) = s$

- 等式$push\ (top\ s)\ (pop\ s) = s[s : stack]$是不可证明的.
- 任何形式为$top\ empty = M[\Gamma]$的等式都是不可证明的，假定M是类别为nat的项，并且不含$empty$.

2.3.4 完备性的形式-1

- 用于不同逻辑系统的三种形式的完备性
  - 最弱的形式：所有的永真公式都是可证的.
    对代数而言，所有的永真公式是(*ref*) 公理的实例.
  - 演绎完备性：每个语义蕴涵在证明系统中都可证.
    对等式而言，如果$\mathcal{E} \models M = N[\Gamma]$，那么$\mathcal{E} \vdash M = N[\Gamma]$.
  - 最小模型完备性：每个语法理论是某个“最小”模型的语义理论.
    对代数而言，每个语法理论是某个代数$\mathcal{A}$的$Th(\mathcal{A})$.
    “最小模型”是指它的理论包含的内容最少，而不是其载体的规模最小.
    最小模型完备性的语义条件：最小模型性质(每个语义理论是某单一模型的理论)

2.3.4 完备性的形式-2

- 最小模型完备性不一定成立
  - 对代数而言，若有空载体则最小模型性质不成立.
  - 通常，当逻辑公式可以表示“析取”的信息时，最小模型性质不成立.
  - 空载体引入了一种析取形式.

考虑例2.5中的等式$E =_{\text{def}} x = y\,[x:a, y:a, z:b]$，满足$E$的任意代数一定满足以下等式之一:

- $E_1 =_{\text{def}} x = y\,[x:a, y:a]$，当$a$的载体只含一个元素
- $E_2 =_{\text{def}} x = y\,[z:b, w:b]$，当$b$的载体为空
- E_1和E_2都不是E的语义蕴含.
- 不存在一个代数，其理论是E的所有等式推论组成的语义理论 .

2.3.5 同余、商和演绎完备性-1

本节给出可以有空载体的多类别代数的演绎完备性定理，该定理的证明要用到同余关系和商代数.

❖ 同余关系

- **同余关系**：在等价关系(自反、对称、传递的二元关系)上增加(*cong*)规则，即每个函数保持可证明的相等性.
- 对单类别代数$\mathcal{A}=\langle A, f_1^{\mathcal{A}}, f_2^{\mathcal{A}},\ldots\rangle$，同余关系是载体$A$上的等价关系，使得对每个$k$元函数$f^{\mathcal{A}}$，如果$a_i \sim b_i (i=1,\ldots,k)$,即$a_i$和$b_i$等价，那么$f^{\mathcal{A}}(a_1, \ldots, a_k) \sim f^{\mathcal{A}}(b_1, \ldots, b_k)$.
- 对多类别Σ代数$\mathcal{A}=\langle\{A^s\}, I\rangle$，同余关系是一簇等价关系$\sim=\{\sim_s\}, \sim_s \subseteq A^s \times A^s$，使得对每个$f: s_1 \times \ldots \times s_k \to s$及变元序列$a_1, \ldots, a_k$和$b_1, \ldots, b_k$(其中$a_i \sim_{s_i} b_i \in A^{si}$),有

$$f^{\mathcal{A}}(a_1, \ldots, a_k) \sim_s f^{\mathcal{A}}(b_1, \ldots, b_k).$$

2.3.5 同余、商和演绎完备性-2

❖ 商代数

- $\mathcal{A}$ 模~的商的代数$\mathcal{A}/\sim$：把$\mathcal{A}$中有关系的元素$a \sim a'$压缩成$\mathcal{A}/\sim$的一个元素.
- $a \in A^s$的等价类$[a]_\sim$ (或记为$a/\sim$)： $[a]_\sim = \{a' \in A^s \mid a \sim a'\}$
- Σ商代数$\mathcal{A}/\sim$

1) $(A/\sim)^s$是由A^s的所有等价类组成的集合

$$A^s/\sim_s = \{[a]_{\sim_s} \mid a \in A\}$$

2)函数$f^{\mathcal{A}/\sim}$由$\mathcal{A}$的函数$f^{\mathcal{A}}$确定

对相应载体的所有$a_1, \ldots, a_k$，

$$f^{\mathcal{A}/\sim}([a_1], \ldots, [a_k]) = [f^{\mathcal{A}}(a_1, \ldots, a_k)].$$

上述定义要求$f^{\mathcal{A}/\sim}([a_1], \ldots, [a_k])$必须只依赖于$[a_1], \ldots, [a_k]$，而不是依赖于所选的等价类中的代表$a_1, \ldots, a_k$.

2.3.5 同余、商和演绎完备性-3

例 单类别代数$\langle N, 0, 1, +\rangle$上的同余关系 “模$k$等价”

这个商代数是大家熟悉的整数模k加结构。如果$k = 5$, 那么$[3] + [4] = [3+4] = [7] = [2]$

- 商代数$\mathcal{A}/\sim$的环境$\eta_\sim$

$\eta_\sim(x) = [\eta(x)]_\sim$，如果$\eta$是$\mathcal{A}$的一个环境

对于$\mathcal{A}/\sim$的环境η'，可以选择任意$\eta(x) \in \eta'(x)$来为$\mathcal{A}$定义环境η.

引理2.8 令$\sim$是Σ代数$\mathcal{A}$上的同余关系，项$M \in Terms(\Sigma, \Gamma)$并且$\eta$是满足$\Gamma$的环境。那么项$M$在商代数$\mathcal{A}/\sim$和环境$\eta_\sim$下的含义$(\mathcal{A}/\sim)[\![M]\!]_{\eta_\sim}$由下式决定

$$(\mathcal{A}/\sim)[\![M]\!]_{\eta_\sim} = [\,\mathcal{A}[\![M]\!]_\eta\,]_\sim$$

证明 基于M的结构进行归纳.

2.3.5 同余、商和演绎完备性-3

项集$Terms(\Sigma, \Gamma)$上的一个关系$\sim_{\mathcal{E},\Gamma}$定义为：

$M \sim_{\mathcal{E},\Gamma} N$当且仅当$\mathcal{E} \vdash M=N[\Gamma]$.

引理2.9 令$\mathcal{E}$是一组Σ等式集合，令$Terms(\Sigma, \Gamma)$是基调Σ上的项集. 由$\mathcal{E}$的可证明性确定的关系$\sim_{\mathcal{E},\Gamma}$是$Terms(\Sigma, \Gamma)$上的同余关系.

定理2.10（完备性）如果$\mathcal{E} \models M=N[\Gamma]$，那么$\mathcal{E} \vdash M=N[\Gamma]$.

完备性定理加上可靠性定理表明，语法理论和语义理论是相同的.

2.3.5 同余、商和演绎完备性-4

定理2.10（完备性）如果$\mathcal{E} \models M=N[\Gamma]$，那么$\mathcal{E} \vdash M=N[\Gamma]$.

证明 用反证法. 假设$M_0=N_0[\Gamma_0]$不能从$\mathcal{E}$证明，若能找到一个代数满足$\mathcal{E}$，但不满足该等式，则完备性得证.

所找的代数是商代数$\mathcal{A} = \mathcal{T}(\Sigma, \Gamma_0)/\sim_{\mathcal{E},\Gamma_0}$，要证明$\mathcal{A}$满足$\mathcal{E}$，但不满足$M_0=N_0[\Gamma_0]$.

为简单起见，用~和$\mathcal{T}$分别表示$\sim_{\mathcal{E},\Gamma_0}$和$\mathcal{T}(\Sigma, \Gamma_0)$.

该问题转变为：找一个环境,使得M_0和N_0在该环境下的解释不相同. 令η是$\mathcal{T}$的一个环境，它把变量都映射到自己.

由引理2.5,$\mathcal{T}[\![M_0]\!]_\eta$就是$M_0$.

由引理2.8, 有$\mathcal{A}[\![M_0]\!]_\eta{\sim}=[M_0]_\sim$

同理，$\mathcal{T}[\![N_0]\!]_\eta$就是$N_0$，$\mathcal{A}[\![N_0]\!]_\eta{\sim}=[N_0]_\sim$.

2.3.5 同余、商和演绎完备性-5

定理2.10（完备性）如果$\mathcal{E} \models M=N[\Gamma]$，那么$\mathcal{E} \vdash M=N[\Gamma]$.

证明

由假设($M_0=N_0[\Gamma_0]$不能从$\mathcal{E}$证明)，有$[M_0]_{\sim} \neq [N_0]_{\sim}$，故在环境$\eta_{\sim}$中，$M_0=N_0[\Gamma_0]$不成立．

接下来证明$\mathcal{A} \models \mathcal{E}$．假定$M=N[\Gamma] \in \mathcal{E}$，由引理2.8，对项代数$\mathcal{T}$的一个环境$\eta$，$\mathcal{A}$的环境可以写成$\eta_{\sim}$．

只要证明对每个满足Γ的$\mathcal{T}$的环境η，$\mathcal{A}[\![M]\!]_{\eta_{\sim}} = \mathcal{A}[\![N]\!]_{\eta_{\sim}}$.

再由引理2.5,得$\mathcal{A}[\![M]\!]_{\eta_{\sim}} = [\mathcal{T}[\![M]\!]_{\eta}]_{\sim} = [\eta M]_{\sim}$．

同样，$\mathcal{A}[\![N]\!]_{\eta_{\sim}} = [\eta N]_{\sim}$．

使用规则(*ref*)和(*subst*)，可以证明$\mathcal{E} \vdash \eta M=\eta N[\Gamma]$(习题2.14)

故有$[\eta M]_{\sim} = [\eta N]_{\sim}$，从而$M$和$N$在$\mathcal{A}$中有同样的含义.

2.3.6 非空类别和最小模型性质

- 类别s是Σ非空的：Σ包含s的一个常量符号，或者有函数符号$f: s_1 \times \ldots \times s_k \to s$并且$s_1, \ldots, s_k$是$\Sigma$非空的.
- 如果s是Σ非空的，则至少有一个项$M \in Terms^s(\Sigma, \varnothing)$, 故在任何$\Sigma$代数中，$s$的载体不可能为空.
- 没有空载体时，可以增加推理规则**(nonempty)**

$$\frac{M = N[\Gamma, x : s]}{M = N[\Gamma]} \quad x\text{不在}M\text{和}N\text{中}$$

定理2.11 令$\mathcal{E}$是封闭于规则**(nonempty)**的语法理论，那么存在所有载体都非空的代数$\mathcal{A}$，使得$\mathcal{E} = Th(\mathcal{A})$.

- 没有空载体的代数有最小模型完备性.

2.4 同态和初始性

❖ 同态

- 是一个代数到另一个代数的保结构的映射(或称翻译).
- 从Σ 代数$\mathcal{A}$ 到Σ 代数$\mathcal{B}$的同态$h:\mathcal{A}\to\mathcal{B}$是一簇映射 $h=\{h^s \mid s\in S\}$，使得对Σ的每个函数符号$f: s_1\times\ldots\times s_k\to s$, 有 $h^s(f^{\mathcal{A}}(a_1,\ldots,a_k)) = f^{\mathcal{B}}(h^{s_1}a_1,\ldots,h^{s_k}a_k)$

例2.11 令$\mathcal{N}=\langle N, 0, 1, +\rangle$，令~是模$k$的等价关系.那么把$n\in\mathcal{N}$映射到它的等价类$[n]_{\sim}$是从$\mathcal{N}$到$\mathcal{N}/\sim$的一个同态, 因为

$$h(0) = 0^{\mathcal{N}/\sim} = [0]_{\sim}$$

$$h(n+m) = h(n) +^{\mathcal{N}/\sim} h(m) = [n+m]_{\sim}$$

- 任何代数到它的商代数的同态都用这种方式定义 .

2.4 同态和初始性

例2.12 含义函数是从项代数$\mathcal{T}=Terms(\Sigma,\Gamma)$到任意代数$\mathcal{A}$的一个同态$h:\mathcal{T}\to\mathcal{A}$. 如果$\eta$是$\mathcal{A}$的一个满足$\Gamma$的环境，该同态的定义是

$$h\,(M)=\mathcal{A}[\![M]\!]_{\eta}$$

这是一个同态，因为

$$\begin{aligned} h\,(f\,M_1\ldots M_k) &= \mathcal{A}[\![f\,M_1\ldots M_k]\!]_{\eta} \\ &= f^{\mathcal{A}}(\mathcal{A}[\![M_1]\!]_{\eta},\ldots,\mathcal{A}[\![M_k]\!]_{\eta}) \\ &= f^{\mathcal{A}}(h\,(M_1),\ldots,h\,(M_k)) \end{aligned}$$

➢ 同态和含义之间的联系

如果$h:\mathcal{A}\to\mathcal{B}$，$\eta$是$\mathcal{A}$的一个满足$\Gamma$的环境，那么定义$\mathcal{B}$的环境$\eta^h$为：$\eta^h(x)=h\,(\eta(x))$ 对任何变量x.

2.4 同态和初始性

引理2.13 令$h:\mathcal{A}\to\mathcal{B}$是任意同态，并且$\eta$是满足类别指派$\Gamma$的任意$\mathcal{A}$环境。那么对任何项$M\in Terms(\Sigma,\Gamma)$，有

$$h\,(\mathcal{A}[\![M]\!]_{\eta})=\mathcal{B}\,[\![M]\!]_{\eta h}$$

当M中不含变量时，$h\,(\mathcal{A}[\![M]\!])=\mathcal{B}\,[\![M]\!]$

证明 按项M的结构进行归纳.

引理2.14 令$h:\mathcal{A}\to\mathcal{B}$和$k:\mathcal{B}\to C$都是$\Sigma$代数的同态，那么合成$k\circ h:\mathcal{A}\to C$也是$\Sigma$代数的同态.

$k\circ h$是合成各类别映射而得到的一簇映射$(k\circ h)^s=k^s\circ h^s$

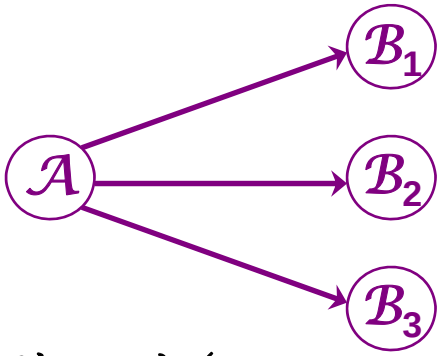
- **同构**：一个双射(入射并且满射)的同态，写成$\mathcal{A}\cong\mathcal{B}$.

同构就是重新命名元素而不改变代数结构，故同构代数本质上是相同的.

2.4 同态和初始性

❖初始代数

如果C是一类Σ代数并且$\mathcal{A}\in C$，若对每个$\mathcal{B}\in C$，存在唯一的同态$h:\mathcal{A}\to\mathcal{B}$，那么$\mathcal{A}$在$C$中叫做**初始代数**.



- 初始代数是“典型”的(可将其翻译到该类中的所有代数)
- 初始代数有尽可能少的非空类别(初始代数的每一元素必须通过某个同态映射到其他代数的一个元素)
- 初始代数满足尽可能少的等式(引理2.13 $h(\mathcal{A}[\![M]\!])=\mathcal{B}[\![M]\!]$表明在代数$\mathcal{A}$中满足的等式一定在代数$\mathcal{B}$中满足)

2.4 同态和初始性

例2.13 基调Σ_0有类别*nat* 和函数符号**0**: *nat*和*S*: *nat* → *nat*.
令$\mathcal{C}$是所有Σ_0代数构成的代数类.

项代数$\mathcal{T}$=*Terms*(Σ_0, ∅) 是$\mathcal{C}$的初始代数，其载体是所有不含变元的项**0**, *S* **0** , *S* (*S* **0**), …, S^k **0**, …

该代数的函数*S*把S^k **0**映射到 S^{k+1} **0**.

该代数拥有解释所有函数符号至少需要的元素，并且满足这些元素之间的尽可能少的等式 .

如何证明项代数$\mathcal{T}$在$\mathcal{C}$中是初始的？首先证明从$\mathcal{T}$到$\mathcal{C}$中的任何代数都有一个同态. (由例2.12知含义函数$\mathcal{A}[\![\cdot]\!]$这样的同态) 剩下是证明这个同态是唯一的.

2.4 同态和初始性

先证明任何代数类的所有初始代数都是同构的.

引理2.15 假设$h:\mathcal{A}\to\mathcal{B}$和$k:\mathcal{B}\to\mathcal{A}$都是同态，并且$h\circ k = Id_{\mathcal{B}}$，$k\circ h = Id_{\mathcal{A}}$，那么$\mathcal{A}$和$\mathcal{B}$同构. 其中$Id$是恒等函数.

同构代数本质上是相同的，故任何代数类的初始代数如果存在，则它唯一到同构.

命题2.16 如果$\mathcal{A}$和$\mathcal{B}$在代数类C中都是初始代数，那么$\mathcal{A}$和$\mathcal{B}$是同构的.

命题2.17 令$\mathcal{E}$是一组Σ等式，$\mathcal{A} = Terms(\Sigma, \varnothing)/\sim_{\mathcal{E},\varnothing}$是且模可证相等性的、不含变元的项代数，则$\mathcal{A}$对$\mathcal{E}$来说是初始代数.

- 由项代数和商的性质可知，从$\mathcal{E}$可证的等式在$\mathcal{A}$中都成立.
- 还要证明从$\mathcal{A}$到任何满足$\mathcal{E}$的代数有唯一的同态.

任何代数规范都有初始代数.

2.4 同态和初始性

例2.14 基调Σ_1有类别*nat*和函数符号$0: nat$、$S: nat \to nat$和$+: nat \times nat \to nat$. 令等式集合$\mathcal{E}$：

$$x+0=x \qquad x+(Sy)=S(x+y)$$

- $S^k0 + S^l0 = S^{k+l}0$
- 对任何不含变元的项M，存在某个自然数k，使得$M = S^k0$
- 等式$S^k0 = S^l0$是不可证明的，除非$k = l$
- 每个不含变元的项的等价类只包含一个形式为S^k0的项
- 等价类和形式为S^k0的项之间有一个双射
- 初始代数：载体看成由不含变元的项$0, S0, \ldots, S^k0, \ldots$构成的集合，函数$S$映射$S^k0 \mapsto S^{k+1}0$，$+$映射$(S^k0, S^l0) \mapsto S^{k+l}0$
- 该初始代数和该基调的标准模型（有后继算子和加法的自然数）同构.

2.4 同态和初始性

该规范的初始代数可与其他有更多或较少元素的代数相对照

- 如果一个代数有更多元素的话，那么这些多余的元素不能由项定义（称为垃圾*junk*）
  - 整数代数Z
  - $\mathcal{A}=\langle A^{nat}, 0^{\mathcal{A}}, S^{\mathcal{A}}, +^{\mathcal{A}}\rangle$

    $A^{nat} = (\{0\} \times N) \cup (\{1\} \times Z)$

    $0^{\mathcal{A}} = \langle 0, 0\rangle$

    $S^{\mathcal{A}}\langle i, n\rangle = \langle i, n+1\rangle$

    $\langle i, n\rangle +^{\mathcal{A}} \langle j, m\rangle = \langle max(i, j), n+m\rangle$
- 如果一个代数有较少元素的话，那么就有一些不能被证明为相等的项在该代数中被等同（出现混淆）
  - 模k的自然数

2.4 同态和初始性

初始代数可能满足不能由$\mathcal{E}$证明的额外的等式.

（直观上，增加元素到一个代数可能导致带变元的等式不成立.）

例2.15 例2.14的初始代数满足+交换性，$x+y=y+x$

因为$\mathcal{E} \vdash M = S^k 0$ 和$\mathcal{E} \vdash N = S^l 0$蕴涵

$$\mathcal{E} \vdash M+N = S^{k+l}0 = N+M$$

- 不满足交换性的代数

  如$\mathcal{A} = \langle A^{nat}, 0^{\mathcal{A}}, S^{\mathcal{A}}, +^{\mathcal{A}} \rangle$满足$\mathcal{E}$，但是函数合成$+^{\mathcal{A}}$无交换性

  - A^{nat}是所有$f: X \to X$的函数的集合（X至少有两个元素）
  - $0^{\mathcal{A}}$是X上的恒等函数，$S^{\mathcal{A}}$是A^{nat}上函数之间的恒等映射
  - $+^{\mathcal{A}}$是A^{nat}上的函数合成

2.4 同态和初始性

- 基项：不含变元的项.
- 基代换：将Γ中的变元映射到Σ上的基项的代换.
- 基实例（项、等式）：

  如果$M \in Terms^s(\Sigma, \Gamma)$并且$S$是一个Σ、Γ基代换，则$S\ M$是$M$的基实例.

  如果$M=N[\Gamma]$是$Terms^s(\Sigma, \Gamma)$的项之间的等式，并且S是一个Σ、Γ基代换，则$S\ M=SN[\varnothing]$称为是$M=N[\Gamma]$的基实例.

命题2.18 令$\mathcal{E}$是一组Σ等式，并且$\mathcal{A}$对$\mathcal{E}$来说是初始代数。对任何Σ等式$M=N[\Gamma]$，下面三个条件等价:

1) $\mathcal{A}$满足$M=N[\Gamma]$.
2) $\mathcal{A}$满足$M=N[\Gamma]$的每一个基实例.
3) $M=N[\Gamma]$的每一个基实例都可以从$\mathcal{E}$证明.

2.5 代数数据类型

2.5.1 规范与数据抽象

数据类型被公理化地定义的一般特征

2.5.2 初始代数语义和数据类型归纳

研究选择什么样的代数作为代数规范的指称语义

2.5.3 解释没有意义的项

怎样规定没有意义的项(如除数为零的表达式，调用不终止的函数)的值

2.5.1 规范和数据抽象-1

- 公理化地定义数据类型
  - 很多数据类型(如优先队列)不存在标准的数学构造，没有单一和标准的计算机实现.
    常通过列出其操作并描述操作的行为来描述，即公理化地定义这些数据类型，而不是由数学构造来定义.
  - 公理化方法的好处：精确描述所有实现的共同要求，而不偏向任何一个特定实现.
  - 对于任意数据类型，难以断定是否有“足够”的公理.
  - 从“公理化地定义”数据类型看，这些数据类型是抽象的.

本节讨论数据类型公理化方法的一般特征.

2.5.1 规范和数据抽象-2

例，有限集合数据类型

sorts: *set, nat, bool*

fctns: 0, 1, 2, … : *nat* $\quad$ +: $nat \times nat \to nat$

$Eq? : nat \times nat \to bool$ $\quad$ *true, false*: *bool*

empty : *set* $\quad$ *insert* : $nat \times set \to set$

union : $set \times set \to set$ $\quad$ *ismem?* : $nat \times set \to bool$

$cond_n : bool \times nat \times nat \to nat$ $\quad$ $cond_b : bool \times bool \times bool \to bool$

$cond_s : bool \times set \times set \to set$

eqns : [*x, y* : *nat*; *s, s'* : *set*; *u,v* : *bool*]

$0 + 0 = 0,\ 0 + 1 = 1, \ldots$ $\quad$ *Eq* ? 0 1 = *false*, *Eq* ? 0 2 = *false*, …

Eq ? *x x* = *true* $\quad$ *ismem? x empty* =*false*

ismem? x (*insert y s*) = if *Eq?x y* then *true* else *ismem? x s*

union empty s = *s* $\quad$ *union* (*insert y s*) *s'* = *insert y* (*union s s'*)

$cond_n$ *true x y* = *x* $\quad$ $cond_b$ *true u v* = *u* $\quad$ $cond_s$ *true s s'* = *s*

$cond_n$ *false x y* = *y* $\quad$ $cond_b$ *false u v* = *v* $\quad$ $cond_s$ *false s s'* = *s'*

2.5.1 规范和数据抽象-3

- 数据抽象的一般原理
  - 抽象数据类型由它的规范定义。使用数据类型的程序应该只依赖于由规范保证的性质，而不依赖于其任何特定实现.
  - 一个数据类型的函数可以划分成

    构造算子：构造该数据类型的一个新元素.
    - 如，*set*类型的*empty*和*insert*

    运算算子：是该数据类型上的函数，不产生新元素.
    - 如，*set*类型的*union*

    观察算子：返回其他数据类型的元素.
    - 如，*set*类型的*ismem*？

2.5.2 初始代数语义和数据类型归纳-1

- 代数规范有几种不同的“语义”形式
  - 宽松语义：满足一个代数规范的所有代数所构成的代数类.
  - 初始代数语义：满足一个代数规范的所有初始代数所构成的同构类.

    初始代数的主要性质：没有**junk**垃圾, 没有混淆， 支持基于数据类型构造算子的归纳(称为**数据类型归纳**).
  - 终结代数语义：满足一个代数规范的所有终结代数所构成的同构类.
  - 生成语义：满足一个代数规范的所有生成代数所构成的代数类.

2.5.2 初始代数语义和数据类型归纳-2

- 构造符和其他函数之间的区别
  - 构造符集合：$Spec=\langle \Sigma, \mathcal{E} \rangle$的函数符号子集$F_0$，使得$Terms(\Sigma,\varnothing)/\sim_{\mathcal{E},\varnothing}$的每个等价类只含某个仅由$F_0$中的函数符号所构成的基项.
  - 可以基于对构造符的归纳来证明初始代数的性质.
- 终结代数
  - 在一个代数类C中，代数$\mathcal{A} \in C$是C的**终结代数**，如果从每个$\mathcal{B} \in C$到$\mathcal{A}$都存在唯一的同态.
  - 一个代数类中的所有终结代数都同构.
  - 在代数类C中，如果某代数的所有载体都是单元素集合，则该代数一定是C的终结代数.

2.5.2 初始代数语义和数据类型归纳-3

❖初始代数语义和终结代数语义

- 初始语义：不能证明为相等的就是不相等的.
- 终结语义：不能证明为不相等的则是相等的.
- 从初始语义角度看，前面给出的不是集合的规范,而是表的规范。因为不能证明

***insert x*(*insert y z*)=*insert y*(*insert x z*) [*x*: *nat*, *y*: *nat*, *z*: *set*]**

***insert x* (*insert x z*) = *insert x z*[*x* : *nat*, *z* : *set*]**

- 若用终结语义，且*nat*和*bool*的解释固定到自然数和布尔值，则前面的规范就是集合的规范.

2.5.2 初始代数语义和数据类型归纳-4

❖ 生成代数

- 一个Σ代数$\mathcal{A}$是**生成的**(或**可达的**、或**项生成的**)，如果对每个元素$a \in A^s$，存在一个基项M使得$a = \mathcal{A}[\![M]\!]$.

生成代数和初始代数之间的联系

- 如果Σ生成代数$\mathcal{A}$满足$\mathcal{E}$，那么从初始代数$\mathcal{T} = Terms(\Sigma, \varnothing)/\sim_{\mathcal{E},\varnothing}$到生成代数$\mathcal{A}$有一个满射.
- 如果Σ生成代数$\mathcal{A}$满足$\mathcal{E}$，那么$Th(\mathcal{A})$包含规范$\langle \Sigma, \mathcal{E} \rangle$的初始代数的理论.
- 等式$M = N[\Gamma]$在满足$\mathcal{E}$的所有生成代数中都成立当且仅当该等式在规范$\langle \Sigma, \mathcal{E} \rangle$的初始代数中成立.

2.5.3 解释没有意义的项

- 表达式没有意义的情况
  - 除法是一个部分函数，除数为零的表达式没有定义.
  - 调用不终止的函数.
- 错误值
  - 如果想在代数规范中表示上述这些情况，如说明除数为0时产生一个错误，必须在基调中增加表示错误的项.
  - 这样的项也是良形项，怎样规定这些项的值?
    - 什么也不规定
    - 任意做一个决定
    - 非常仔细地说明什么是所需要的

2.6 重写系统

2.6.1 基本定义

重写系统、终止性、合流性、可变换性

2.6.2 合流性和可证的相等性

对于合流的重写系统，当1个等式的2个项都能归约到同一个项时，该等式一定可证.

2.6.3 终止性

用项在良基代数上的解释来证明重写系统的终止性.

2.6.4 临界对

通过检查临界对判定重写系统是否局部合流.

2.6.5 左线性无重迭重写系统

这类系统的合流性不受终止性影响.

2.6.6 局部合流、终止和合流之间的联系

用终止性分析规则之间的相互影响.

2.6.1 基本定义-1

- 重写系统
  - 项重写系统(简称重写系统)是关于代数项的归约系统.
  - 两种重要的应用
    - 为代数项提供一种计算模型
    - 自动定理证明
  - 由一组叫做重写规则($L \to R$)的有向等式组成.
    - 其中$L, R \in Terms^s(\Sigma, \Gamma)$， L不是变量并且$Var(R) \subseteq Var(L)$
    - 规则$L \to R$允许把一个项中的L的实例$S\ L$化简为$S\ R$
  - 由$\mathcal{R}$ 确定的一步归约关系$\to_{\mathcal{R}}$

$$[SL/x]M \to_{\mathcal{R}} [SR/x]M$$

关系$\to\!\!\!\to_{\mathcal{R}}$是$\to_{\mathcal{R}}$的自反传递闭包.

当不出现混淆时，下标$_{\mathcal{R}}$可省略.

2.6.1 基本定义-2

例2.17 基调

sorts : ***nat***

fctns : **0:** ***nat***

$-$**:** $nat \rightarrow nat$

$+$**:** $nat \times nat \rightarrow nat$

- 在这个基调上的一些归约规则如下

$$x + 0 \rightarrow x$$

$$x + (-x) \rightarrow 0$$

$$(x + y) + z \rightarrow x + (y + z)$$

- 实例: $\underline{(x + y) + (-y)} \rightarrow x + \underline{(y + (-y))} \rightarrow \underline{x + 0} \rightarrow x$

加下划线的子项是被重写的部分，称为可归约式(redex).

2.6.1 基本定义-3

- 归约保类别

  引理2.15 令$\mathcal{R}$是Σ上的重写系统. 如果$M \in Terms^s(\Sigma, \Gamma)$，并且$M \twoheadrightarrow_R N$，那么$N \in Terms^s(\Sigma, \Gamma)$.

- 合流性和终止性

  $N \twoheadleftarrow M$表示$M \twoheadrightarrow N$

  $M \twoheadrightarrow \circ \twoheadleftarrow N$表示存在项$P$使得$M \twoheadrightarrow P \twoheadleftarrow N$

  - 关系$\twoheadrightarrow$是合流的(confluent)，如果$N \twoheadleftarrow M \twoheadrightarrow P$蕴涵$N \twoheadrightarrow \circ \twoheadleftarrow P$.
  - 关系$\rightarrow$是终止的(terminating)，如果不存在一步归约的无穷序列$M_0 \rightarrow M_1 \rightarrow M_2 \ldots$.
  - 不能再归约的项称为范式(normal form).
  - 合流且终止的重写系统通常被称为是规范的(canonical).

2.6.1 基本定义-4

❖ 可变换性(conversion)—归约的无向版本

项$M, N \in Terms^s(\Sigma, \Gamma)$是可变换的，记为$M \leftrightarrow_{\mathcal{R}} N[\Gamma]$, 是指:

如果M和N相等，或存在$M' \in Terms^s(\Sigma, \Gamma)$ 使得$M \twoheadrightarrow_{\mathcal{R}} M'$ 或$M' \twoheadrightarrow_{\mathcal{R}} M$ 并且$M' \leftrightarrow_{\mathcal{R}} N$ [Γ].

(也即，如果存在$M \twoheadrightarrow M_1 \twoheadleftarrow M_2 \twoheadrightarrow M_3 \ldots M_k \twoheadleftarrow N$)

- 箭头的方向并没有什么意义.
- 对任何重写系统，可变换性和可证明的相等性是一致的

如果重写系统是合流的，则

$M \leftrightarrow_{\mathcal{R}} N[\Gamma]$当且仅当$M \twoheadrightarrow \circ \twoheadleftarrow N$.

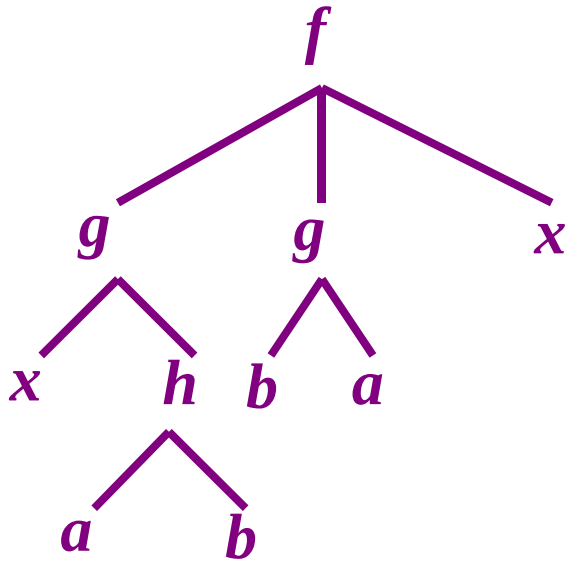
2.6.1 基本定义-5

❖项中的一个位置

- 是用于方便地引用子项在项中的某个特定出现.

例

- 位置 $\vec{n}=1,2$的子项是hab.
- 用$M|_{\vec{n}}$表示M在位置 $\vec{n}$的子项.
- 用$[N/\vec{n}]M$表示M在位置 $\vec{n}$ 的子项由N代换的结果.



$f\ (gx(hab))(gba)\ x$

- 一步归约可表示为:
  $M\rightarrow_{\mathcal{R}} N$当且仅当$M$中有一个在位置$\vec{n}$使得$M|_{\vec{n}} \equiv SL$并且 $N \equiv [SR/\vec{n}]\ M$.

2.6.2 合流性和可证明的相等性

如果$\mathcal{R}$是一组重写规则，$\mathcal{E}_{\mathcal{R}}$用来表示对应的无向等式集合

定理2.16.

对任何重写系统$\mathcal{R}$, $M \leftrightarrow_{\mathcal{R}} N[\Gamma]$当且仅当 $\mathcal{E}_{\mathcal{R}} \vdash M = N[\Gamma]$.

对任何合流的重写系统$\mathcal{R}$, $\mathcal{E}_{\mathcal{R}} \vdash M = N[\Gamma]$
当且仅当$M \twoheadrightarrow_{\mathcal{R}}^{\circ} \twoheadleftarrow_{\mathcal{R}} N$.

从定理2.16，可以得到下面的推论

推论2.17. 如果$\twoheadrightarrow_{\mathcal{R}}$是合流的，并且类别s存在2个不同的范式，那么是$\mathcal{E}_{\mathcal{R}}$是一致的.

2.6.3 终止性-1

❖ 良基关系与终止性

- 良基关系：集合A上的一个关系$\prec$是良基的，如果不存在A上元素的无穷递减序列$a_0 \succ a_1 \succ a_2 \succ \ldots$.

- 如果能在项和有良基关系的集合A的元素间建立起一个对应，那么可以利用它去证明不存在无穷的归约序列$M_0 \to M_1 \to M_2 \to \ldots$

- 有几种方式可把项映射到有良基关系的集合：利用代数的语义结构

2.6.3 终止性-2

❖ 良基代数

代数$\mathcal{A}=\langle A^{s1}, A^{s2}, \ldots, f_1^{\mathcal{A}}, f_2^{\mathcal{A}}, \ldots \rangle$是良基的，如果

- 每个载体A^s上有一个良基关系$\prec_s$.
- 对每个n元函数符号f，如果$x_1 \preceq y_1, \ldots, x_n \preceq y_n$并且对某个$i$（$1 \le i \le n$）有$x_i \prec y_i$，那么$f^{\mathcal{A}}(x_1, \ldots, x_n) \prec f^{\mathcal{A}}(y_1, \ldots, y_n)$.

❖ 语义蕴含

- 如果$\mathcal{A}$是一个良基代数，并且M和N都是类别s上的项，如果$[\![M]\!]_\eta \prec_s [\![N]\!]_\eta$，那么$\mathcal{A}, \eta \models M \prec N$.

类似地，$\mathcal{A} \models M \prec N$当$\mathcal{A}, \eta \models M \prec N$对任何环境$\eta$都成立.

2.6.3 终止性-3

定理2.18 令$\mathcal{R}$是Σ项上的一个重写系统，并且令$\mathcal{A}$是一个良基的Σ代数。如果对$\mathcal{R}$中每条规则$L \to R$有$\mathcal{A} \models L \succ R$，那么$\mathcal{R}$是终止的.

例2.18

$\neg\neg x \to x$

$\neg (x \vee y) \to (\neg x \wedge \neg y)$

$\neg \neg (x \wedge y) \to (\neg x \vee \neg y)$

$x \wedge (y \vee z) \to (x \wedge y) \vee (x \wedge z)$

$(y \vee z) \wedge x \to (y \wedge x) \vee (z \wedge x)$

$\mathcal{A} = \langle A^{bool}, or^{\mathcal{A}}, and^{\mathcal{A}}, not^{\mathcal{A}} \rangle$

载体$A^{bool} = N - \{0, 1\}$

$\vee^{A}(x, y) = x + y + 1$

$\wedge^{A}(x, y) = x * y$

$\neg^{A}(x) = 2^{x}$

$\mathcal{A}$是一个良基代数.

每条重写规则的左部定义的值都大于其右部定义的值 .

2.6.4 临界对-1

❖ 局部合流

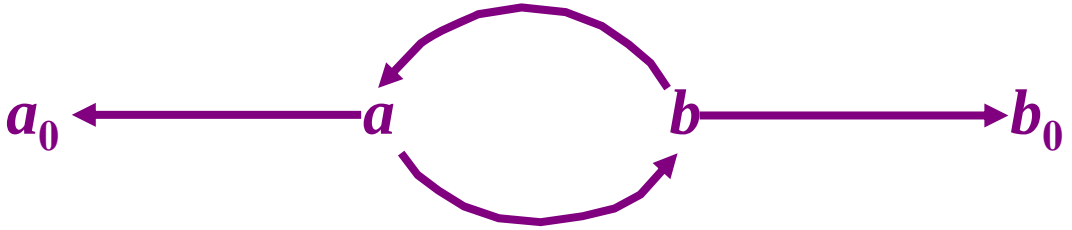
关系$\rightarrow$是局部合流的，如果$N \leftarrow M \rightarrow P$蕴涵$N \rightarrow\!\!\!\!\!\rightarrow \circ \leftarrow\!\!\!\!\!\leftarrow P$.

➢ 局部合流关系严格弱于合流关系.

例2.20 一个局部合流但不合流的重写系统

$a \rightarrow b, b \rightarrow a$

$a \rightarrow a_0, b \rightarrow b_0$



a和b都能归约到a_0和b_0，但是a_0和b_0都不能归约到对方.

2.6.4 临界对-2

- 如何判断重写系统是局部合流的
  - 不相交的归约

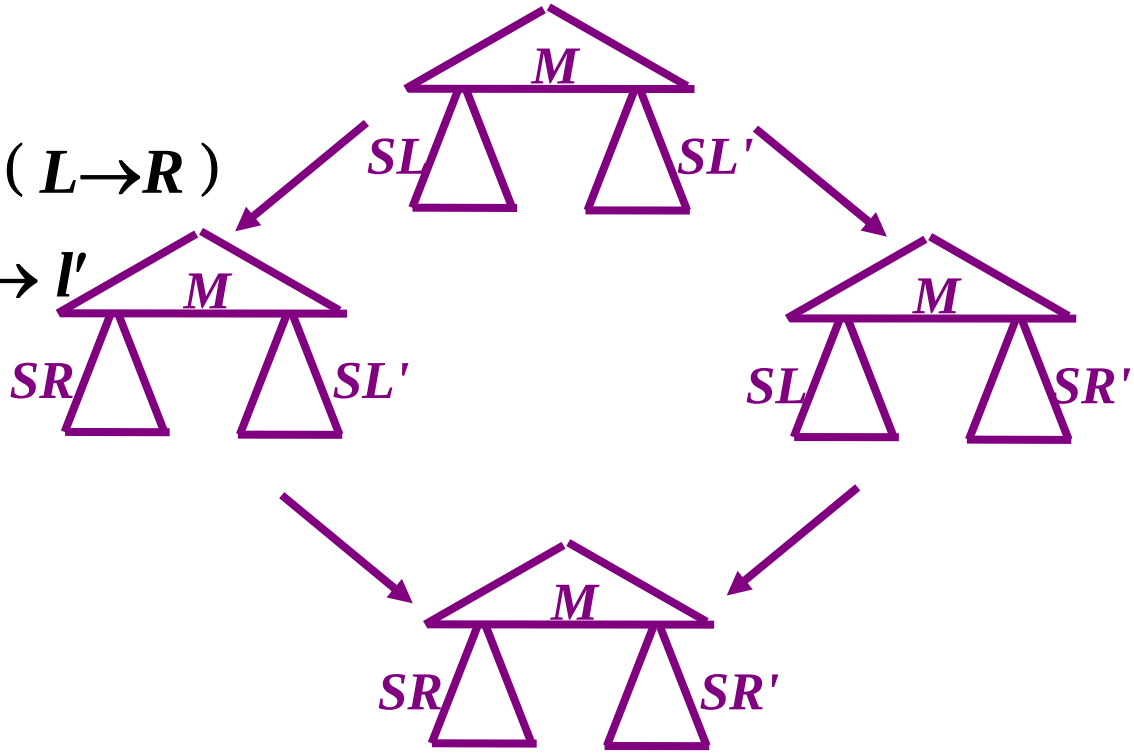
cond B (cdr(cons s l)) (cons(car l′)(cdr l′))

规则如下：

cdr(cons x l) $\rightarrow$ ***l*** （***L*** $\rightarrow$ ***R***）

cons(car l′)(cdr l′) $\rightarrow$ ***l′***

（***L′*** $\rightarrow$ ***R′***）



2.6.4 临界对-3

- 如何判断重写系统是局部合流的
  - 平凡的重迭

cdr(cons x(cons(car l')(cdr l')))

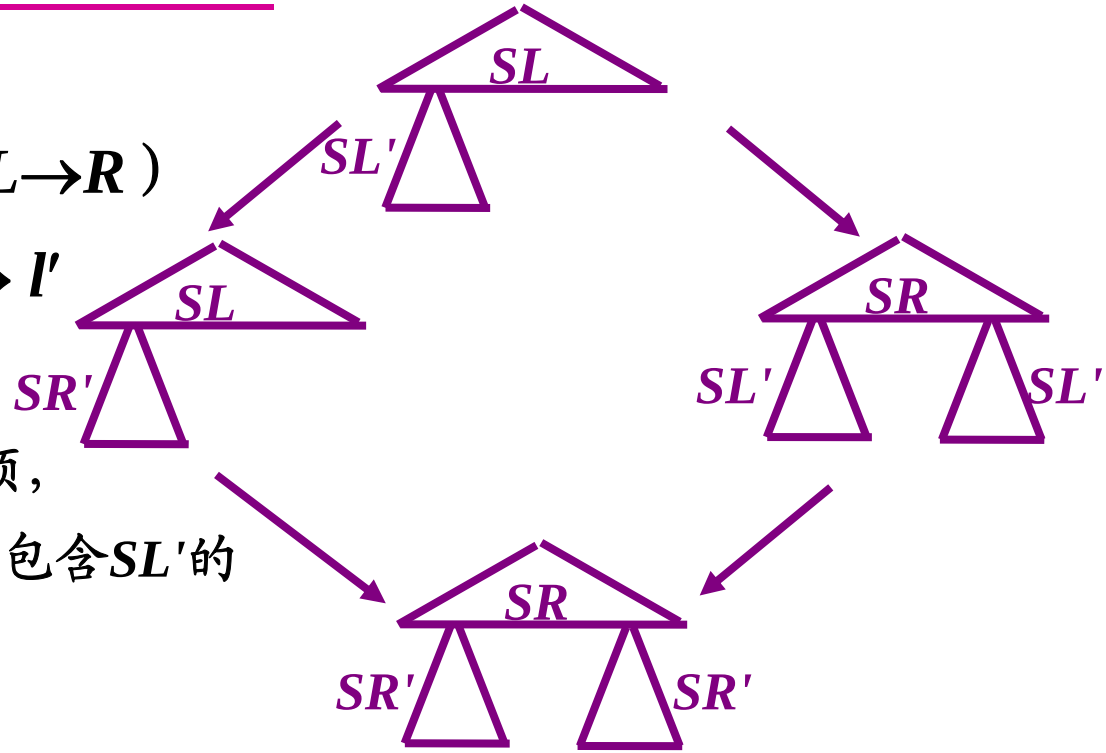
规则如下:

cdr(cons x l) $\rightarrow$ ***l*** （***L***$\rightarrow$***R***）

cons(car l')(cdr l') $\rightarrow$ ***l'***

（***L'***$\rightarrow$***R'***）

平凡的：***SL'***是***SL***的子项，且***S***把***L***中的某个变量用包含***SL'***的一个项代换.



2.6.4 临界对-4

- 如何判断重写系统是局部合流的
  - 非平凡的重迭

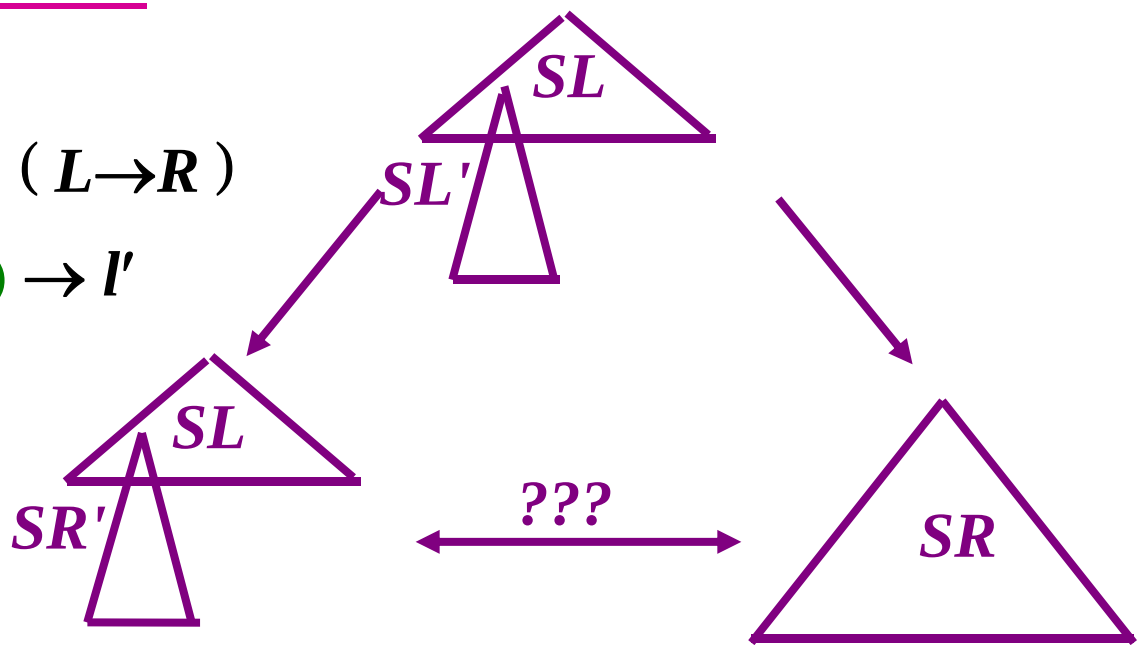
cdr(cons(car l′)(cdr l′))

规则如下：

$cdr(cons\ x\ l) \rightarrow l$ （$L \rightarrow R$）

$cons(car\ l')(cdr\ l') \rightarrow l'$

（$L' \rightarrow R'$）



2.6.4 临界对-5

❖临界对

- 对于规则$L \to R$和$L' \to R'$，若存在代换使得三元组$\langle SL, SL', \vec{m} \rangle$是一个非平凡重迭，则序对$\langle SR, [SR'/\vec{m}]SL \rangle$叫做临界对.
- 前面例子的规则有两个临界对，

1) $L \to R$: $cdr(cons\ x\ l) \to l$; $L' \to R'$: $cons(car\ l')(cdr\ l') \to l'$

SL是$cdr(cons(car\ l)(cdr\ l))$，临界对是$\langle cdr\ l, cdr\ l \rangle$

2) $L \to R$:$cons(car\ l')(cdr\ l') \to l'$; $L' \to R'$: $cdr(cons\ x\ l) \to l$

SL是$cons(car(cons\ x\ l))\ (cdr(cons\ x\ l))$

临界对是$\langle cons\ x\ l\ , cons(car(cons\ x\ l))\ l \rangle$

2.6.4 临界对-6

❖临界对

- 对于规则$L \to R$和$L' \to R'$，只有L'的主函数符号f出现在L中，临界对才有可能出现.
- 若有规则$f\,(g\,x\,y) \to R'$，L中有子项$f\,z$或$f\,(gPQ)$时，才可能产生临界对. 子项$f\,(h\,\ldots)$不会引起临界对（$f\,(h\,\ldots)$不可能与$f\,(gxy)$合一）.
- 同一条规则的两次使用可能引起重迭，如

  $f\,(f\,x) \to a$　　$f\,(f\,(f\,x))$

  从一条规则$L \to R$的两次应用引起的相同项构成的临界对$\langle R, R\rangle$，称为平凡临界对. (引起重迭$\langle L, L, \varepsilon\rangle$)
- 左部完全相同的情况：$L \to R$和$L \to R'$. $\langle R, R'\rangle$是临界对

2.6.4 临界对-7

命题2.19 一个重写系统$\mathcal{R}$是局部合流的，当且仅当对每个临界对$\langle M, N\rangle$有$M \rightarrow\!\!\!\!\!\rightarrow_{\mathcal{R}}{}^{\circ} \leftarrow\!\!\!\!\!\leftarrow_{\mathcal{R}} N$.

证明.从左到右的蕴涵是简单明了的.

另一个方向的证明仿照定义临界对时采用的推理：不相交、平凡的重迭、临界对的代换实例.

如果一个有限的重写系统$\mathcal{R}$是终止的，那么该命题就给出一个算法，可用于判定$\mathcal{R}$是否局部合流.

2.6.5 左线性无重迭重写系统-1

一个重写系统没有非平凡的临界对，则称它是无重迭的.

无重迭的重写系统不一定是合流的.

例2.22 关于有穷数和无穷数的重写系统没有临界对，因此它是局部合流的，但它不是合流的.

$$\infty \to S\ \infty$$

$$Eq\ ?\ x\ x \to true$$

$$Eq\ ?\ x\ (S\ x) \to false$$

$Eq\ ?\ \infty\ \infty$有两个不同的范式

$$Eq\ ?\ \infty\ \infty \to true$$

$$Eq\ ?\ \infty\ \infty \to Eq\ ?\ \infty\ (S\ \infty) \to false$$

2.6.5 左线性无重迭重写系统-2

- 左线性的规则：任何变量在规则左部的出现不超过一次.
- 左线性的重写系统：每一条规则都是左线性的.

命题2.20 如果$\mathcal{R}$是左线性和无重迭的，那么$\rightarrow\!\!\!\!\rightarrow_{\mathcal{R}}$是合流的.

例2.23 下面是一个左线性无重迭重写系统.

$car\ (cons\ x\ l) \rightarrow x$　　$cdr\ (cons\ x\ l) \rightarrow l$

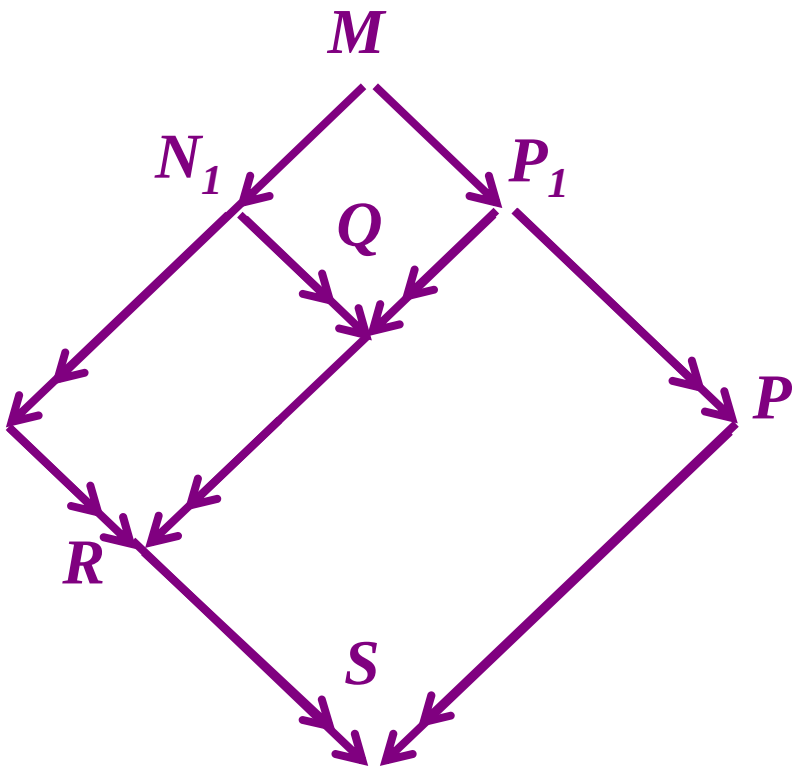
$isempty\ ?\ nil \rightarrow true$　　$isempty\ ?\ (cons\ x\ l) \rightarrow false$

加入非左线性规则$cons\ (car\ l)\ (cdr\ l) \rightarrow l$后不合流

$isempty?\ (cons\ (car\ l)\ (cdr\ l))$有两个范式，即$false$和$isempty?\ (l)$.

2.6.6局部合流、终止和合流之间的联系

命题2.21 令$\mathcal{R}$是终止的重写系统，那么$\mathcal{R}$是合流的当且仅当它是局部合流的.



例2.25 下面的重写系统是局部合流的并且是终止的，但不是左线性的.

$$\text{if } \mathit{true} \text{ then } x \text{ else } y \to x$$

$$\text{if } \mathit{false} \text{ then } x \text{ else } y \to y$$

$$\text{if } u \text{ then } x \text{ else } x \to x$$

2.6.6局部合流、终止和合流之间的联系

***Knuth-Bendix*完备化过程**

测试：通过检查每个临界对是否合流来判断一个终止的重写系统$\mathcal{R}$是否局部合流.

完备化：在$\mathcal{R}$不合流时添加规则，添加的规则对原来的规则集是守恒的.(守恒：$\mathcal{E}_{\mathcal{R}'}$和$\mathcal{E}_{\mathcal{R}}$确定同样的代数理论)

完备化过程可能不终止.

例2.26 由等式定向成一组重写规则

$$f\,x + f\,y \to f\,(x + y) \qquad (x + y) + z \to x + (y + z)$$

产生一个临界对$\langle f\,(x + y) + z,\ f\,x + (f\,y + z)\rangle$

增加一条重写规则$f\,(x + y) + z \to f\,x + (f\,y + z)$

本例增加无数条重写规则$f^{n}(x + y) + z \to f^{n}x + (f^{n}\,y + z)$

作业

2.2 [TPL]2.3

2.3 [TPL]2.4 2.10 2.13 2.16

2.4 [TPL]2.19 2.22 2.23

2.6 [TPL]2.30 2.33