Theory of Programming Languages
程序设计语言理论

张昱

Department of Computer Science and Technology
University of Science and Technology of China

December, 2007

第九章 子定型(Subtyping)

9.1 子定型概述[PFPL, 32.1]

9.2 子定型的种类[PFPL, 32.2]

9.3 子定型的变迁[PFPL, 32.3]

子类型关系是类型上的偏序(自反且传递的)关系，它满足以下包含原理(subsumption principle):

如果σ是τ的子类型，则当需要一个类型为τ的值时，都可以将一个类型为σ的值提供给它。(σ的元素集是τ的元素集的子集)

包含原理放宽了类型系统的限制，即允许一个类型的值被当成另一个类型的值。例如，如果一个函数期待一个超类型的值作为参数，则该函数可以应用到一个子类型的参数上。

包含原理在实际中难以正确应用，应用的关键是正确使用引入和消去原理。

在判断两个类型是否是子类型关系时，需要考虑以下两个问题:

1. 子类型的每一引入形式是否都能看作是超类型的引入形式? (即判断子类型的元素集是否是超类型的元素集的子集)
2. 如果对1的回答是肯定的，则进一步判断超类型的每一消去形式是否都能被应用到子类型的值上?

如果对上述两个问题的回答是肯定的，那么就能保证子类型的每个值是超类型的合法值(**legitimate value**)，而且超类型上的操作可以限定到子类型的值上。

9.1 子定型-1

- 子定型断言
  - $\sigma <: \tau$表示σ是τ的子类型
  - 子定型断言至少满足以下结构规则(自反性、传递性)

$$\frac{}{\tau <: \tau}$$

$$\frac{\rho <: \sigma \quad \sigma <: \tau}{\rho <: \tau} \qquad \text{(PFPL 32.1)}$$

- 包含规则
  - 在语言的定型规则中增加包含规则，可以扩大良类型程序的集合

$$\frac{\Gamma \vdash e : \sigma \quad \sigma <: \tau}{\Gamma \vdash e : \tau} \qquad \text{(PFPL 32.2)}$$

  - 包含规则不是语法制导的，因为它不限制e的形式，即包含规则可以应用到任意形式的表达式上。
  - 动态语义应不受是否使用包含规则的影响！因为动态语义一般是定义在表达式上，而不是定义在定型推导上。

9.1 子定型-2

❖包含规则

➢类型安全的证明必须考虑子定型的影响!

- 置换引理
  - 对类型化推导(如 $\Gamma, x:\tau \vdash e:\tau'$)进行归纳.
- 保持性
  - 对转换规则归纳证明，不受包含规则的影响。
- 进展性
  - 对定型规则归纳证明，必须考虑包含规则的影响。
- 范式引理
  - 在证明进展性时需要用到范式引理
  - 范式引理也必须考虑包含规则
  - 对范式引理的证明也可以按定型规则归纳证明

9.2 子定型的种类-数-1

本节将在L{—}的不同扩展版本上讨论子定型的不同形式!

❖ 数(Numbers)

整数**int**、有理数**rat**、实数**real**有： **int <: rat <: real**

- **int <: rat**
  - 整数**int**、有理数**rat**及其上的操作有加法、乘法、加法恒等元(0)、乘法恒等元(1)、加法逆元素(指定数的负数)...
  - 如果**int <: rat** ，则必须保证:
    每个整数是有理数，在有理数上的操作局限到在整数上的对应操作.
    例如，两个整数的有理数和(是有理数)一定是它们的整数和(所得的整数也是有理数)
    如何保证？将整数***n***表示成有理数***n*/1**.
- **rat <: real** ： 可以将有理数表示成实数

——以上反映了整数、有理数和实数在数学上的特点

9.2 子定型的种类-数-2

❖ 数(Numbers)

在计算机中如何表示数?

从效率考虑，希望整数的表示比有理数的表示更有效，有理数的表示比实数的表示更有效。

这样就必须有一个显式的包含函数：它会为每一个整数指定关联的有理数表示，为每一有理数指定关联的实数表示。

但是，当要考虑机器表示的有限精度时，如

- 整数：二进制的补码形式
- 实数：浮点数（考虑有限精度）

这时，就不可能将每个整数正好关联到一个浮点数表示(由于精度的问题！)

9.2 子定型的种类-积-1

❖ 积(Products)

➢ 元组的广度子定型(width subtyping)

$$\frac{n \leq m}{\langle \tau_1, \ldots, \tau_m \rangle <: \langle \tau_1, \ldots, \tau_n \rangle}$$

(PFPL 32.3)

一个较宽的元组类型是一个较窄的元组类型的子类型

为了进行子定型，只要求对第$i(1 \leqslant i \leqslant n)$个成员的选择与元组的实际长度无关！——在子类的元组中，超类的元组成员放在前面.

➢ 记录的广度子定型

$$\frac{n \leq m}{\langle l_1 : \tau_1, \ldots, l_m : \tau_m \rangle <: \langle l_1 : \tau_1, \ldots, l_n : \tau_n \rangle}$$

(PFPL 32.4)

记录类型中的域是次序无关的(按标签访问)，设π是{1,…,n}的排列

则有 $\mathtt{rcd}[l_1, \ldots, l_n](\tau_1, \ldots, \tau_n) \equiv \mathtt{rcd}[l_{\pi(1)}, \ldots, l_{\pi(n)}](\tau_{\pi(1)}, \ldots, \tau_{\pi(n)})$

9.2 子定型的种类-积-2

❖ 积(Products)

在按标签访问记录中的域时，如何确定域的位置？

- 在没有广度子定型时，可以选择一种规范的排序方式，如让各个域按标签的字母序来存放，如**<cno:str, score:int, sname:str, sno:str>**

- 在有广度子定型时，类型只反映出关于哪些域的偏序信息
  - 采用字母序：对在记录中定位域是无益的
    如超类记录是**<cno:str, score:int, sname:str, sno:str>**
    子类记录是**<cname:str, cno:str, score:int, sname:str, sno:str>**

  - 为每个记录建立一个标签映射到偏移量的字典，用来确定记录中每个域的位置
    记录中的域按元组类型形式进行存放(前面的域是超类中的域)
    字典可以按标签的字母序排列，通过字典可以快速标签访问域

9.2 子定型的种类-变式

❖ 和(Sums)—变式

- 和类型中选项越多，则关于成员的信息就越少
  如$[l_1:T_1]$能精确地说明成员的标签是什么

$$\frac{m \leq n}{[l_1:\tau_1,\ldots,l_m:\tau_m] <: [l_1:\tau_1,\ldots,l_n:\tau_n]}$$

(PFPL 32.5)

具有较少选项的类型是较多选项的类型的子类型

9.2 子定型的种类-递归类型-1

❖ 递归类型(Recursive Types)

例如：带标签的二叉树类型，结点上的标签为整型

$$\mu(t.[\mathtt{empty}:\mathtt{unit},\mathtt{binode}:\langle\mathtt{data}:\mathtt{int},\mathtt{lft}:t,\mathtt{rht}:t\rangle])$$

以及结点上无标签的二叉树类型

$$\mu(t.[\mathtt{empty}:\mathtt{unit},\mathtt{binode}:\langle\mathtt{lft}:t,\mathtt{rht}:t\rangle])$$

其中一个是另一个的子类型？

直观认识：带标签的二叉树是不带标签的二叉树的子类型。(从记录类型的子类型关系来得到)

进一步考虑如下不带标签的2-3树

$$\mu(t.[\mathtt{empty}:\mathtt{unit},\mathtt{binode}:\langle\mathtt{lft}:t,\mathtt{rht}:t\rangle,\mathtt{trinode}:\langle\mathtt{lft}:t,\mathtt{mid}:t,\mathtt{rht}:t\rangle])$$

2-3树与前面两种树存在什么样的子类型关系？

直观认识：2-3树是不带标签的二叉树的超类(从变式的子类型关系来得到)

9.2 子定型的种类-递归类型-2

- 根据上述例子，递归类型的子定型规则似乎可以定义为

$$\frac{t \mid \sigma <: \tau}{\mu(t.\sigma) <: \mu(t.\tau)} \;??$$

(PFPL 32.6)

上述规则表明：判断一个递归类型是否是另一个的子类型，只需要简单比较二者的递归体，而将约束变元处理成参数。

- 规则(32.6)证实了关于"带标签的二叉树是不带标签的二叉树的子类型"这一直观认识，但是也证实了一些错误的子定型关系！

例如，$\sigma = \mu(s.\langle self{:}s,\ a{:}int\rangle \rightarrow \langle x{:}int, y{:}int\rangle)$

σ是函数类型，函数参数中的self域为该函数类型本身，函数以两个整数组成的记录为结果。

对于类型为σ的函数，其self参数可以是该函数本身，也可以不要求是该函数本身。

9.2 子定型的种类-递归类型-3

例如，$\sigma = \mu(s.<self:s, a:int> \rightarrow <x:int, y:int>)$

设e是类型为σ的函数

$e = fold(\lambda(u:<self:t, a:int>.<x= unfold(u.self)(u).y, y=8>))$

1) 在参数u上调用$u.self$ (即将$u.self$的展开形式应用到u上)

2) 选择1)中调用结果的y域，作为整个结果的x域

进一步考虑递归类型$\tau = \mu(t.<self:s, a:int> \rightarrow <x:int>)$

设e_0是类型为τ的函数：$fold(\lambda(y:<self:t, a:int>.<x=7>))$

由规则(32.6)，有$\sigma <: \tau$。从而，由于$e:\sigma$，故$e:\tau$。

现在考虑表达式$unfold(e)(<self= e_0, a=0>)$，它是良类型的，因为

$e_0:\tau$，但是它在执行时会因应用e_0所得的结果不存在y域而受阻！

故规则(32.6)是不正确的。

原因：t表示2个递归类型，从而没有正确地对自引用建模。

e中的self是子类型σ，而$unfold(e)$应用的参数则是超类τ

9.2 子定型的种类-递归类型-4

以要证明的结论为假设(自引用)

- 关于递归类型的正确子定型规则

$$\frac{\mu(s.\sigma) <: \mu(t.\tau) \vdash [\mu(s.\sigma)/s]\sigma <: [\mu(t.\tau)/t]\tau}{\mu(s.\sigma) <: \mu(t.\tau)}$$

(PFPL 32.7)

递归类型的子定型规则的另一种表示:

$$\frac{s, t \mid s <: t \vdash \sigma <: \tau}{\mu(s.\sigma) <: \mu(t.\tau)}$$

(PFPL 32.8)

这种表示使用参数，而不是置换

9.3 变迁-1

本节讨论在类型构造子上的子定型！

例如，如果一个元组类型的域是另一个元组类型对应域的子类型，那么这两个元组类型也是子定型关系？

❖ 类型构造子的分类

称类型构造子在参数上是:

- 协变的(covariant): 如果类型构造子保持在参数上的子定型。
  如: 积类型构造子、和类型构造子;
  函数类型构造子在输出参数(值域)上是协变的
- 逆变的(contravariant): 如果类型构造子所保持的子定型关系与参数上的子定型关系相反。
  如: 函数类型构造子在输入参数(定义域)上是协变的
- 不变的(invariant): 如果所构造的类型不受参数上的子定型影响。

9.3 变迁-2

- 积
  - 元组、记录和对象类型构造子都是协变式，因为它们保持每个参数上的子定型。
  - 元组类型的协变式原理(深度子定型):

$$\frac{\sigma_1 <: \tau_1 \quad \ldots \quad \sigma_n <: \tau_n}{\langle \sigma_1, \ldots, \sigma_n \rangle <: \langle \tau_1, \ldots, \tau_n \rangle}$$

(PFPL 32.9)

若e是子类型表达式，则其第i投影的类型为σ_i,由包含规则类型也为τ_i，从而可以将e的类型提升为超类型。

- 和
  - 无标签的和类型以及带标签的和类型都是协变式。
  - 带标签的和类型的协变式原理

$$\frac{\sigma_1 <: \tau_1 \quad \ldots \quad \sigma_n <: \tau_n}{[l_1 : \sigma_1, \ldots, l_n : \sigma_n] <: [l_1 : \tau_1, \ldots, l_n : \tau_n]}$$

(PFPL 32.10)

对超类型的值进行分情况分析，在第i个分支上将接受类型为τ_i的值。由(32.10)的前提，则可以提供给第i个分支σ_i的值。

9.3 变迁-3

❖ 函数

- 函数类型在值域上的变迁：协变的

$$\frac{\tau <: \tau'}{\sigma \to \tau <: \sigma \to \tau'}$$

(PFPL 32.11)

假设$e : \sigma \to \tau$，若$e_1 : \sigma$，则$e(e_1) : \tau$. 如果$\tau <: \tau'$，则$e(e_1) : \tau'$

- 函数类型在定义域上的变迁：逆变的

$$\frac{\sigma' <: \sigma}{\sigma \to \tau <: \sigma' \to \tau}$$

(PFPL 32.12)

假设$e : \sigma \to \tau$，则e可以应用到任意类型为σ的值，由包含规则，e也可应用到σ的任意子类型σ'，两种情况都得到类型为τ的值.

Thanks!