γ演算到 Action 演算的转换*

金 英+, 金成植

(吉林大学 计算机科学与技术学院,吉林 长春 130012)

Translating γ-Calculus into Action Calculus

JIN Ying+, JIN Cheng-Zhi

(College of Computer Science and Technology, Jilin University, Changchun 130012, China)

+ Corresponding author: Phn: 86-431-5166480, E-mail: jin0119@mail.jl.cn

http://www.jlu.edu.cn

Received 2001-06-29; Accepted 2001-12-27

Jin Y, Jin CZ. Translating γ-calculus into action calculus. *Journal of Software*, 2003,14(1):16~22.

Abstract: Action calculi is introduced as a mathematical framework for expressing different interactive behaviors, which shows the advantages in representing different interactive models with some common features. In this paper, action calculi is used to include γ-calculus (a computational calculus for higher-order concurrent programming) in its setting. First, a concrete action calculus $AC(K_\gamma)$ is defined. Then the formal compositional translation of the γ-calculus into $AC(K_\gamma)$ is presented. Finally, upon definitions of the observability, the weak barbed bisimularity as well as the weak barbed congruence for $AC(K_\gamma)$, it is proved that such translation preserves the weak behavioural equivalence of the γ-calculus with the π-calculus as intermediate. This work not only shows the expressiveness of action calculi, but also provides precondition for uniting and comparing γ-calculus with other concurrent models under the theory of action calculi.

Key words: action calculi; γ-calculus; translation; weak barbed congruence

摘 要: Action 演算簇(action calculi)作为描述不同并发交互行为的数学框架,可以表示一大类具有某些相同特性的并发形式化模型.试图把γ演算(一种基于约束的高阶并发计算模)也包含在 action 演算簇的框架下.首先定义了一个具体的 action 演算 $AC(K_\gamma)$,然后给出了从γ演算到 $AC(K_\gamma)$转换的形式描述,最后在定义 $AC(K_\gamma)$的可观察性、弱互模拟关系和弱等价关系的基础上,以π演算为中间表示,证明了这种转换保持了γ演算的弱行为等价性.研究表明,action 演算簇可以表示基于约束的并发模型,从而充分说明了 action 演算簇的描述能力,并且为在 action 演算簇框架下把γ演算与其他并发模型结合并进行比较提供了前提.

关键词: action 演算簇;γ演算;转换;弱等价关系

中图法分类号: TP301 **文献标识码**: A

Action 演算簇(action calculi)是由英国爱丁堡大学教授 Robin Milner 于 1993 年提出来的[1,2].它可以作为描述一大类并发交互行为模型的数学框架,是具有某些相同特性的一类演算的集合,其中一个具体的演算称为

* Supported by the National Natural Science Foundation of China under Grant No.60073041 (国家自然科学基金)

第一作者简介: **金英**(1971－),女,吉林长春人,博士,讲师,主要研究领域为程序设计语言,软件形式化.

action 演算.Action 演算簇的研究工作主要集中在以下几个方面:(1) Action 演算簇本身的一些性质和理论[1~3];(2) Action 演算簇与其他并行演算之间的关系[4,5];(3) Action 演算簇的动态性质及其分类[6,7];(4) Action 演算簇的应用[8].目前已经取得了一些重要的成果.其中在与其他并行演算之间的关系方面,已经给出了λ演算、π演算、ambient 演算、Petri 网以及一个函数式语言的 action 演算定义[1,2,8],并且在文献[4]中给出了从线性逻辑到 action 演算簇的转换,在文献[5]中以π演算为中介对 tile logic 和 action 演算簇进行了比较等等.这些充分说明 action 演算可以把不同的演算(尤其是并行演算)统一在共同的框架下,并为对它们进行比较、归并和结合,进而更好地理解并发行为提供了一个很好的途径.

本文试图把γ演算包含在 action 演算簇的框架下,以便把γ演算与其他并发模型进行结合与比较.γ演算是一个高阶并发计算模型[9].对γ演算进行扩充后可以得到一个基于约束的问题求解系统.它具有把一阶约束和高阶程序设计相结合的特性,是成熟的程序设计语言与系统 Oz 的核心[9,10].我们首先定义了一个具体的 action 演算 $AC(K_\gamma)$,然后针对γ演算的特点,引入了引用的句柄和句柄接口的概念,并在此基础上给出了γ演算到 $AC(K_\gamma)$的转换函数,最后我们采用类似于文献[11]的方法,给出了 $AC(K_\gamma)$的可观察性(observability)、弱互模拟关系(weak barbed bisimularity)和弱等价关系(weak barbed congruence)的定义,并以π演算为中间表示,证明了本文给出的转换保持了γ演算的弱行为等价性.本文的工作表明,action 演算簇不仅可以表示进程演算(如π演算),而且也可以表示基于约束的并发模型,从而充分说明了 action 演算簇的描述能力.同时,应用本文的思路可以考虑其他并发模型是否也包含在 action 演算簇之中,从而为在 action 演算簇这个统一的框架下把γ演算与其他并发模型进行结合与比较,并进一步理解和统一各种并发模型提供了可能.

1 Action 演算与γ演算

1.1 Action演算

这里,我们采用文献[3]中的定义和表示方法.

一个 action 演算是由一个基调 $\mathbb{K}=(P,K)$来定义的,包括以下几个部分:(1) 一个项的集合;(2) 定义在项集合上的等式理论;(3) 建立在项的等价类上的前序关系(又称为反作用关系或者动态关系).

在给出每个部分的具体内容之前,我们首先给出一些约定:

(1) 一个基调 $\mathbb{K}=(P,K)$,其中 P 是基本类型(prime)集合,每个元素用 $p,q,\ldots$表示;K 是一个控制操作集合;

(2) 基本类型的有限序列,称为元(arity),用ε表示空序列,⊗表示序类的连接操作,M 表示元的集合,则$(M,\varepsilon,\otimes)$构成一个幺半群;

(3) 假设有一个可数名字集合 X,其中每个名字 x 都有一个基本类型 p 与之相关联,记作 $x{:}p$ 或者 x^p.

定义 1. 项 $a,b,c,\ldots$是由基本操作符(恒等式 id_m,复合操作 •,积操作⊗,置换 $P_{m,n}$,抽象(x)或者 ab_x,数据$\langle x^p\rangle$和控制操作符 K)构造而成,其语法形式为

$$a,b,c,\ldots::=id|\langle x\rangle|P_{m,n}|a\bullet b|a\otimes b|ab_xa|K(a_1,\ldots,a_n).$$

每个项 a 都有一个元对(m,n)相对应,记作:$a{:}m\to n$,且满足一定的元规则(详见文献[1~3]).

定义 2. 等式理论 AC 是建立在项上的等价关系,由下面的公理组成:

$a\bullet id=a=id\bullet a,\quad id\otimes id=id,\quad a\bullet(b\bullet c)=(a\bullet b)\bullet c,\quad a\otimes id=a=id\otimes a,\quad p_{m,n}\bullet p_{n,m}=id_{m\otimes n},$

$a\otimes(b\otimes c)=(a\otimes b)\otimes c,\qquad (a\bullet b)\otimes(a'\bullet b')=(a\otimes a')\bullet(b\otimes b'),$

$p_{m,n}\bullet(b\otimes a)=(a\otimes b)\bullet p_{m,n},\qquad p_{lm,n}=(id_l\otimes p_{m,n})\bullet(p_{l,n}\otimes id_m),$

$(\langle y\rangle\otimes id_m)\bullet(x)a=a(\langle y\rangle/\langle x\rangle)(x\notin fn(a)),\qquad (x)((\langle x\rangle\otimes id_m)\bullet a)=a(x\notin fn(a)).$

定义 3. 动作是满足等式理论 AC 的项的等价类,我们仍旧用 $a,b,c,\ldots$表示.

定义 4. 设 t 为一个项,则控制规则的一般形式为 $t[\overline{a}]\searrow t'[\overline{a}]$ (其中 $\overline{a}$ 表示动作序列).

定义 5. 一个 action 演算是由动作集合以及建立在动作集合上的前序关系↘(满足所有控制规则的最小关系)组成的.

action 演算簇中各个操作的含义如下:

• 恒等式 id 表示恒等动作,对交互行为没有任何影响.

• $\langle x\rangle$表示数据.

• 置换操作 $P_{m,n}$ 是把输入名字的顺序进行置换以后输出.

• 复合操作 $a\bullet b$ 并不代表顺序复合.我们可以把一个动作看做是一次活动,那么,对于 $a{:}k\rightarrow m$ 和 $b{:}m\rightarrow p$ 来讲,元 m 描述了在复合 $a\bullet b$ 中 a 影响 b 的接口.我们可以把这种影响看做是在 a 和 b 的后续活动中通过接口流动的信息,即"数据流"或者是"管道".

• 积操作⊗可以看做是并行复合,在 $a\otimes b$ 中,a 和 b 的归约将相互独立地并行进行.但是,对于 $a\otimes b\searrow c$ 来说,并不意味着这种作用仅仅是从 a 或者 b 产生的,它表示的是 a 和 b 之间的通信或者交互作用.特别是,当 a 和 b 可能"使用"了相同的名字 x 时,可以把 x 作为通道来形成通信.

• 抽象操作 ab_x(或者(x))允许对名字 x 进行参数化(并不是像λ演算那样可以指代任意的值).当 x 指代一个通道时,对 x 的抽象可以改变通信接口 x.因此,可以通过 x 来改变与之通信的对象.

• 控制操作符 $K(a_1,\ldots,a_n)$允许在原有动作的基础上构造新的动作.

1.2 γ演算

γ演算是一个高阶并发计算模型.它的原语是逻辑变量、名字、过程抽象和单元.假设有分别表示变量和名字的互不相交的无穷字母表,则又把变量和名字合称为引用,其中名字是惟一的值,变量是名字的占位符.简单地说,γ演算中表达式的含义是:T表示空动作;符合操作∧中的两个项将并行归约;声明$\exists u\ E$ 声明了一个新的引用 u,其作用域是整个 E;等式给出了两个项在所有项中的等价关系;一个抽象 $a{:}\ \bar{x}\ /E$ 和应用 $u\,\bar{v}$ 可以进行通信,产生新的项 $E\{\bar{v}\,/\,\bar{x}\}$;一个单元 $a{:}u$ 可以看做是用来保存值 u,并且也可以和一个应用 avw 相通信,其结果是单元的值变为 w,同时旧的值将产生一个等式 $u{=}v$;条件表达式 if $u{=}v$ then E else F 通过测试两个引用 u 和 v 是否等价来选择产生新的项.

γ演算的语法定义如下:

符号:x,y,z 表示变量;a,b,c 表示名字;$u,v,w{::=}x|a$ 表示引用;

表达式:$E,F,G{::=}\top|E\wedge F|\exists u\ E|u{=}v|a{:}\ \bar{x}\ /E|\ u\bar{v}\ |$if $u{=}v$ then E else $F|a{:}u$.

γ演算的结构等价性:

(1) $E\equiv F$,如果 E 和 F 可以α转换;

(2) $E\wedge F\equiv F\wedge E, E\wedge(F\wedge G)\equiv(E\wedge F)\wedge G, E\wedge\top\equiv E$;

(3) $\exists u\ \top\equiv\top, \exists u\ \exists v\ E\equiv\exists v\ \exists u\ E, \exists u\ E\wedge F\equiv\exists u\ (E\wedge F)$,如果 $u\notin fr(F)$.

γ演算的规约规则:

STRUCT: $\dfrac{E\equiv E' \quad E'\rightarrow F' \quad F\equiv F'}{E\rightarrow F}$.

COMP: $\dfrac{E\rightarrow E'}{E\wedge F\rightarrow E'\wedge F}$.

DECL: $\dfrac{E\rightarrow E'}{\exists u\,E\rightarrow\exists u\,E'}$.

APPL: $a\,\bar{u}\wedge a{:}\ \bar{x}\ /E\rightarrow E\{\,\bar{u}\,/\,\bar{x}\,\}\wedge a{:}\ \bar{x}\ /E$,若 $\bar{u}$ 在 E 是自由的,且$|\,\bar{u}\,|=|\,\bar{x}\,|$.

THEN: if $u{=}u$ then E else $F\rightarrow E$.

ELSE: if $a{=}b$ then E else $F\rightarrow F$,如果 $a\neq b$.

ELIM: $\exists x\ (x{=}u\wedge E)\rightarrow E\{u/x\}$,如果 $x\neq u$ 且 u 在 E 是自由的.

2 定义 action 演算 $AC(K_\gamma)$

下面给出一个具体的 action 演算 $AC(K_\gamma)$的定义,其中基本类型为 1,因此幺半群为自然数$(N,0,+)$.这里,我们主要给出控制及其规则.

控制:

v, *out* (动作参数个数为 0); *box*, *rep* (动作参数个数为 1);

match (动作参数个数为 2); *in_choose*, *out_choose* (动作参数个数为任意).

控制的元规则:

$\nu: 0\to 1$, $out: 1+m\to 0$,

$$\frac{a:m\to n}{box\ a:1\to n},\quad \frac{a:m\to 0}{rep\ a:1\to 0},\quad \frac{a:m\to n,\ b:m\to n}{match\,(a,b):2\to n},$$

$$\frac{a_1:0\to n,\ldots,a_n:0\to n}{in\_choose(a_1,\ldots,a_n):0\to n},\quad \frac{a_1:0\to n,\ldots,a_n:0\to n}{out\_choose(a_1,\ldots,a_n):0\to n}.$$

控制规则:

(1) $((\langle x\rangle\otimes id_m)\cdot out)\otimes(\langle x\rangle\cdot box)\searrow id_m$.

(2) $((\langle x\rangle\otimes id_m)\cdot out)\otimes(\langle x\rangle\cdot box\ a)\searrow a$.

(3) $((\langle x\rangle\otimes id_m)\cdot out)\otimes in\_choose\ (\ldots,\langle x\rangle\cdot box\ a,\ldots)\searrow a$.

(4) $(\langle x\rangle\cdot box\ a)\otimes out\_choose\ (\ldots,(\langle x\rangle\otimes id_m)\cdot out,\ldots)\searrow a$.

(5) $((\langle x\rangle\otimes id_m)\cdot out)\otimes(\langle x\rangle\cdot rep\ a)\searrow a\otimes(\langle x\rangle\cdot rep\ a)$.

(6) $\langle xx\rangle\cdot match\ (a,b)\searrow a$.

(7) $\langle xy\rangle\cdot match\ (a,b)\searrow b(x\neq y)$.

3 转　换

3.1 问　题

从γ演算到 action 演算的转换,主要的困难在于逻辑变量的处理,因为 action 演算中只有名字而没有变量.为了解决这个问题,我们引进了如下内容:

(1) 针对γ演算中的引用 u,我们引入了句柄的概念.一个句柄是一个参数化的动作,它将与其他的动作相互作用,进行必要的归约或者传递值.一个句柄的初始值就是引用本身,随着γ演算中等式的出现将不断地改变.针对不同情况,有 3 种不同的句柄:

- $N(\underline{a})$:针对名字 a 的句柄,它的值总是 a.
- $R(x,\underline{u})$:针对变量 x 的句柄,它将把 x 的值变为 u,并且在收到询问其值或者改变其值的请求时,把该请求传递给 u.
- $V(\underline{x})$:针对变量 x 的句柄,它将处理 x 的值,当收到改变其值的请求时,把其状态转换为 $R(x,u)$.

(2) 针对γ演算中的每个引用,我们在 action 演算中引进了 5 个不同的名字,称为句柄的接口.为了简洁,引用 u 对应的接口将用 u, $u.name$, $u.value$, $u.update$ 和 $u.equ$ 表示,它们的含义是:

- u 本身用于抽象的 action 演算表示和单元的 action 演算表示之间的通信;
- $u.name(a)$声明 u 的值为 action 演算的名字 a;
- $u.value(c,v)$通过向 u 传递两个名字 c 和 v 来询问其值,其中值将通过 c 传递出去,而 v 表示提出申请的通道;
- $u.update(v)$将通知 u 其值修改为 v;
- $u.equ(v,y,n)$将向 u 询问它是否与 v 相同,如果是,则通知 y,否则通知 n.

在下面的转换中,我们将统一用 $\underline{u}$ 表示 u 对应的 5 个接口名字.

3.2 预定义的动作

下面给出句柄的动作定义:

$$\begin{aligned}N(a)\equiv\ & rep(\langle a.name\ \underline{a}\rangle\cdot out)\otimes rep(\langle a.value\rangle\cdot box((c,\_)\langle c\ a\rangle\cdot out))\otimes\\ & rep(\langle a,update\rangle\cdot box((u)\langle u.update\ a\rangle\cdot out))\otimes\\ & rep(\langle a.equ\rangle\cdot box((\underline{u}\ y\ n)\langle u.name\rangle\cdot box((\underline{b})\langle a\ b\rangle\cdot match(\langle y\rangle\cdot out,\langle n\rangle\cdot out)))).\end{aligned}$$

$V(\underline{x})\ \underline{\mathrm{def}}\ in\_choose(\langle x.value\rangle \bullet box((c,\_)\langle c\ x\rangle \bullet out \bullet V(\underline{x})),\langle x.update\rangle \bullet box((\underline{u})) \bullet \langle x\ u\rangle \bullet match(V(\underline{x}),$

$\nu \bullet (c)\langle u.value\ c\ x\rangle \bullet out \bullet \langle c\rangle \bullet box((\underline{u})\langle x\ u\rangle \bullet match(\langle y\rangle \bullet out,\langle x.equ\ c\ v\rangle \bullet out)))).$

$R(\underline{x},\underline{u})\equiv rep(\langle u.name\rangle \bullet box((\underline{a})\langle x.name\ \underline{a}\rangle \bullet out))\otimes$

$rep(\langle x.value\rangle \bullet box((c\ v)\langle u\ v\rangle \bullet match(\langle c\ \underline{u}\rangle) \bullet out,\langle u.value\ c\ v\rangle \bullet out))\otimes$

$rep(\langle x.update\rangle \bullet box((\underline{v})\ \langle u.update\ \underline{v}\rangle \bullet out))\otimes$

$rep(\langle x.equ\rangle \bullet box((\underline{v}\ y\ n)\langle u.equ\ \underline{v}\ y\ n\rangle \bullet out)).$

3.3 转换函数

下面给出每个γ演算的表达式对应的 action 演算表示:

$Tr〖\top〗=id_0,$

$Tr〖E\wedge F〗=Tr〖E〗\otimes Tr〖F〗,$

$Tr〖\exists a\ E〗=\nu \bullet (\underline{a})(N(\underline{a})\otimes Tr〖E〗),$

$Tr〖\exists x\ E〗=\nu \bullet (\underline{x})(V(\underline{x})\otimes Tr〖E〗),$

$Tr〖u=v〗=\langle u.update\ \underline{v}\rangle \bullet out,$

$Tr〖a{:}\ \bar{x}\ /E〗=rep(\langle u.name\rangle \bullet box((a)\langle a\rangle \bullet box(\ \underline{\bar{x}}\ ) \bullet Tr〖E〗)),$

$Tr〖u\ \bar{v}\ 〗=\langle u.name\rangle \bullet box((\underline{a})\langle a\ \ \underline{\bar{v}}\ \rangle \bullet out),$

$Tr〖\text{if } u=v \text{ then } E \text{ else } F〗=\nu \bullet (yn)((\langle u.equ\ \underline{v}\ y\ n\rangle \bullet out)\otimes(\langle y\rangle \bullet box(Tr〖E〗))\otimes(\langle n\rangle \bullet box(Tr〖F〗))),$

$Tr〖a{:}u〗=\langle u.name\rangle\cdot box((\underline{a})(\langle a\rangle \bullet box((\underline{v}\ \underline{w})(Tr〖u=v〗\otimes Tr〖a{:}w〗)))).$

4 行为等价性

在这一节中我们先介绍一些概念,然后结合文献[2,11]中的结论,以π演算为中间表示,证明上面的转换保持了γ演算的弱行为等价性.

4.1 概 念

在文献[11]中,针对γ演算中的项 E 定义了对应的π演算表示 $\mathcal{V}(E)$,类似地,我们也给出了项 E 对应的 action 演算 $AC(K_\gamma)$表示 $\mathcal{U}(E)$的定义.

定义 6. 如果 $fr(E)=\{a_1,\ldots,a_n\},n>0$,则有 $\mathcal{U}(E)\equiv\nu \bullet (\bar{a})\ (N(\underline{a_1})\otimes\ldots\otimes N(\underline{a_n})\otimes Tr〖E〗)$,其中 E 是γ演算中的项,而且 $a\equiv\bigcup_{i=1}^{n}\{a_i.name, a_i.value, a_i.update, a_i.equ\}$.

同样地,类似于文献[11]中γ演算和π演算的可观察性和弱互模拟关系以及弱相似性和弱等价关系的定义,我们给出了对应 $AC(K_\gamma)$的相关定义.

定义 7($AC(K_\gamma)$可观察性 $\downarrow^x_{AC(K_\gamma)}$).

$\langle x\rangle \bullet out$ 是 $AC(K_\gamma)$可观察的;

$\langle x\rangle \bullet box\ a$ 是 $AC(K_\gamma)$可观察的;

如果 a 是 $AC(K_\gamma)$可观察的且 $x\neq y$,则 $\nu \bullet (y)a$ 是 $AC(K_\gamma)$可观察的;

如果 a 是 $AC(K_\gamma)$可观察的或者 b 是 $AC(K_\gamma)$可观察的,则 $a\otimes b$ 是 $AC(K_\gamma)$可观察的;

如果 a_i 是 $AC(K_\gamma)$可观察的且 $1\le i\le n$,则 $in_choose(a_1,\ldots,a_n)$是 $AC(K_\gamma)$可观察的;

如果 a_i 是 $AC(K_\gamma)$可观察的且 $1\le i\le n$,则 $out_choose(a_1,\ldots,a_n)$是 $AC(K_\gamma)$可观察的;

如果一个项 t 是 $AC(K_\gamma)$可观察的,则记作 $t\downarrow^x_{AC(K_\gamma)}$;

如果 $a\searrow\downarrow^x_{AC(K_\gamma)}$,我们记作 $a\Downarrow^x_{AC(K_\gamma)}$.

定义 8. 称定义在 $AC(K_\gamma)$项上的一个对称关系 R 是 $AC(K_\gamma)$弱互模拟关系(weak $AC(K_\gamma)$-barbed bisimulation),如果对于所有的$(a,b)\in R$,有:

(1) 如果 $a\searrow a'$,则 $b\searrow b'$且满足 $a'\ R\ b'$;

(2) 如果 $a\downarrow^{x}_{AC(K_\gamma)}$,则 $b\Downarrow^{x}_{AC(K_\gamma)}$.

称两个项 a 和 b 是 $AC(K_\gamma)$弱相似的(weak $AC(K_\gamma)$-barbed bisimilar),如果存在某个 $AC(K_\gamma)$弱互模拟关系 R,使得 $a\,R\,b$.

定义 9. 称两个项 a 和 b 是 $AC(K_\gamma)$弱等价的(weak $AC(K_\gamma)$-barbed congruent),记作 $a\approx_{AC(K_\gamma)} b$,如果在所有的 $AC(K_\gamma)$演算上下文 $C[.]$下,满足 $C[a]$和 $C[b]$是 $AC(K_\gamma)$弱相似的.

4.2 结 论

我们首先给出从π演算到 $AC(K_r)$的转换函数 *trans* 以及它们之间关系的定理.

trans【0】$=id_0$;

trans【$\bar{x}\langle y\rangle$】$=\langle xy\rangle\cdot out$;

trans【$x(y).P$】$=\langle x\rangle\cdot box((y)$*trans*【$P$】$)$;

trans【$P|Q$】=*trans*【P】$\otimes$*trans*【Q】;

trans【$(\nu x)P$】$=\nu\cdot(x)$*trans*【P】;

trans【$x_1(y_1).P_1+\ldots+x_n(y_n).P_n$】$=in\_choose\,(\langle x_1\rangle\cdot box((y_1)$*trans*【$P_1$】$),\ldots,\langle x_n\rangle\cdot box((y_n)$*trans*【$P_n$】$))$;

trans【$\bar{x}_1\langle y_1\rangle+\ldots+\bar{x}_n\langle y_n\rangle$】$=out\_choose\,(\langle\bar{x}_1y_1\rangle\cdot out,\ldots,\langle\bar{x}_ny_n\rangle\cdot out)$;

trans【$!P$】$=rep$ *trans*【P】;

trans【if $x=y$ then P else Q】$=\langle xy\rangle\cdot match($*trans*【$P$】,*trans*【$Q$】$)$.

定理 1. 对于π演算中任意两个进程 P 和 Q,有:

(1) $P\equiv Q$ 当且仅当 *trans*(P)=*trans*(Q);

(2) 如果 $P\rightarrow Q$,则 *trans*$(P)\searrow$*trans*(Q);

(3) 如果 *trans*$(P)\searrow a$,则存在 P',满足 $P\rightarrow P'$且 *trans*$(P')=a$.

证明:证明类似于文献[2]中的定理证明. □

引理 1. P 是π可观察的,当且仅当 *trans*(P)是 $AC(K_\gamma)$可观察的.

证明:根据π可观察性和 $AC(K_\gamma)$可观察性的定义,对 P 进行归纳,可以很自然地得到 P 的π可观察性和 *trans*(P)的 $AC(K_\gamma)$可观察性之间的对应关系. □

推论 1. $P\Downarrow^{a}_{\pi}$ 当且仅当 *trans*$(P)\Downarrow^{x}_{AC(K_\gamma)}$.

证明:根据定义得证. □

引理 2. 如果 R 是弱π互模拟关系,则存在关系 $R'=\{($*trans*(P),*trans*$(Q))|(P,Q)\in R\}$是弱 $AC(K_r)$互模拟关系.

证明:对于所有(*trans*(P),*trans*$(Q))\in R'$,有:

(1) 如果 *trans*$(P)\searrow a$,则 *trans*$(Q)\searrow b$,且 $a\,R'\,b$.

由 *trans* $(P)\searrow a$ 可知,存在 P',满足 $P\rightarrow P'$且 *trans* $(P')=a$,而$(P,Q)\in R$,因此有:如果 $P\rightarrow P'$,则 $Q\rightarrow Q'$,且 $P'\,R\,Q'$.这样,由定理 1,有 *trans*$(Q)\searrow$*trans*(Q'),同时 *trans*$(P')\,R'$ *trans*(Q'),令 $b=$*trans*(Q'),则 $a\,R'\,b$.

(2) 如果 *trans*$(P)\downarrow^{x}_{AC(K_\gamma)}$,则 *trans*$(Q)\Downarrow^{x}_{AC(K_\gamma)}$(显然可以由引理 1 和推论 1 得到).

最后由弱 $AC(K_\gamma)$互模拟关系定义可以得证. □

引理 3. 如果 R 是弱 $AC(K_\gamma)$互模拟关系,且对于所有的$(a,b)\in R$,有 P,Q 为π演算中的进程,使得 $a=$*trans*(P),$b=$*trans*(Q),则存在关系 $R'=\{(P,Q)|($*trans*(P),*trans*$(Q))\in R\}$,是弱π互模拟关系.

证明:对于所有(*trans*(P),*trans*$(Q))\in R$,有:

(1) 如果 $P\rightarrow P'$,则 $Q\rightarrow Q'$,且 $P'\,R\,Q'$.

由 $P\rightarrow P'$得到 *trans*$(P)\searrow$*trans*(P'),因此 *trans*$(Q)\searrow b$,且 *trans*$(P')\,R\,b$.由定理 1 可知,存在 Q',满足 $Q\rightarrow Q'$且 *trans*$(Q')=b$,因此 *trans*$(P')\,R$ *trans*(Q'),进而 $P'\,R\,Q'$.

(2) 如果 $P\downarrow_\pi^a$,则 $Q\Downarrow_\pi^a$(显然可以由引理 1 和推论 1 得到).

最后由弱π互模拟关系的定义可以得证. □

引理 4. P 和 Q 是弱π相似的,当且仅当 $\mathit{trans}(P)$和 $\mathit{trans}(Q)$是弱 $AC(K_\gamma)$相似的.

证明:由引理 3、引理 4 以及弱π相似、弱 $AC(K_\gamma)$相似的概念可以得证. □

定理 2. $P\approx_\pi Q$,当且仅当 $\mathit{trans}(P)\approx_{AC(K_\gamma)}\mathit{trans}(Q)$.

证明:由$\approx_\pi$和 $\approx_{AC(K_\gamma)}$ 的定义可以得证. □

前提. 对于γ演算中任意的项 E,我们有 $\mathit{trans}(\mathcal{V}(E))\equiv\mathcal{U}(E)$.

证明:应用 $AC(K_\gamma)$中定义的控制规则,很容易得到该结论. □

推论 2. $\mathcal{V}(E)\approx_\pi\mathcal{V}(F)$当且仅当 $\mathcal{U}(E)\approx_{AC(K_\gamma)}\mathcal{U}(F)$.

证明:由定理 2 和前提显然得证. □

此外,我们引入文献[11]中的定理作为定理 3.

定理 3. $E\approx_\gamma F$ 当且仅当 $\mathcal{V}(E)\approx_\pi\mathcal{V}(F)$.

定理 4. $E\approx_\gamma F$ 当且仅当 $\mathcal{U}(E)\approx_{AC(K_\gamma)}\mathcal{U}(F)$.

证明:由推论 2 和定理 3 可以得证. □

5 总 结

本文定义了一个具体的action演算$AC(K_\gamma)$,并且给出了γ演算到$AC(K_\gamma)$的转换;同时,还证明了在弱行为等价性概念下,本文给出的转换忠实地表示了γ演算.本文的工作表明,γ演算也可以包含在 action 演算簇的框架下,从而为把γ演算与其他演算进行比较与结合,并进一步深入理解各种并发行为提供了广阔的前景.

References:

[1] Milner R. Action calculi, or concrete action structures. In: Borzyszkowski AM, Sokolowski S, eds. Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science. LNCS 711, Berlin: Springer-Verlag, 1993. 105~121.

[2] Milner R. Calculi for interaction. Acta Informatica, 1996,33(8):707~737.

[3] Gardner P, Hasegawa M. Higher-Order and reflexive action calculi: their type theory and models. 1998. http://www.cl.cam.ac.uk/users/pag20/horac.ps.gz.

[4] Barber A, Gardner P, Hasegawa M, Plotkin G. From action calculi to linear logic. In: Nielsen M, Thomas W, eds. Computer Science Logic, Proceedings of the 11th International Workshop (CSL'97). LNCS 1414, Berlin: Springer-Verlag, 1998. 78~97.

[5] Gadducci F, Montanari U. Comparing action calculi and tile logic: on the operational semantics of p-calculus. 1999. http://www.dsi.unive.it/~lopstr99/W40/PS/gadducci.pdf.

[6] Gardner P, Wischik L. Explicit fusions. In: Nielson M, Rovan B, eds. Proceedings of the 25th International Symposium, Mathematical Foundations of Computer Science 2000. LNCS 1893, Berlin: Springer-Verlag, 2000. 373~383.

[7] Leifer J. Operational congruences for reactive systems [Ph.D. Thesis]. Computer Laboratory, University of Cambridge, 2001.

[8] Gardner P, Milner R, Sewell P. Calculi for interactive systems: theory and experiment. 2001. http://www.doc.ic.ac.uk/~pg/final2001.html.

[9] Smolka G. A foundation for higher-order concurrent constraint programming. Research Report, RR-94-16, Saarbrücken: The German Research Center for Artificial Intelligence GmbH (DFKI), 1994.

[10] Henz M, Mehl M, Müller M, Müller T, Niehren J, Scheidhauer R, Schulte C, Smolka G, Treinen R, Würtz J. The Oz Handbook. Saarbrücken: The German Research Center for Artificial Intelligence GmbH (DFKI), 1994.

[11] Victor B, Parrow J. Constraints as processes. In: Montanari U, Sassone V, eds. Proceedings of the CONCUR'96: Concurrency Theory. LNCS 1119, Berlin: Springer-Verlag, 1996. 389~405.