移动灰箱演算中强干扰问题的进一步控制*

管旭东, 杨怡玲, 尤晋元

(上海交通大学 计算机科学与工程系 分布计算技术中心,上海 200030)
E-mail: guan-xd@cs.sjtu.edu.cn; yang-yl@cs.sjtu.edu.cn; you-jy@cs.sjtu.edu.cn.ac.cn
http://dctc.sjtu.edu.cn

摘要: 为了消除移动灰箱演算中的强干扰问题,Levi 等人提出了安全灰箱演算.然而,安全灰箱演算中引入的反动作却带来了新的安全隐患.为了消除上述安全隐患,提出了鲁棒灰箱演算.鲁棒灰箱演算在依靠反动作解决强干扰问题的同时,利用反动作的参数明确了该反动作的使用对象,有效地消除了安全灰箱演算中的不安全因素.对防火墙跨越的描述和对多元异步π-演算的翻译显示鲁棒灰箱演算依然具有较强的类似移动灰箱演算和安全灰箱演算的表达能力.同时还就鲁棒灰箱演算的类型问题作了初步的探讨,给出并证明了一套可以描述进程和能力的移动性和线程数两个属性的类型系统.研究结果初步表明,鲁棒灰箱演算可以成为移动计算形式化描述中的有力工具.

关 键 词: 进程代数;移动灰箱演算;安全移动灰箱演算;鲁棒灰箱演算;类型
中图法分类号: TP391 文献标识码: A

Cardelli 和 Gordon 在文献[1]中首先提出了可以统一描述移动计算中的计算平台移动性和计算代码移动性的移动灰箱(mobile Ambients,简称 MA)演算(箱字取自其整体移动性,灰字取自其内部结构部分可见、部分不可见之意——发生归约的部分可见,其余不可见).灰箱(ambient)是一个可以整体移动的计算场所.灰箱之间表现为树状嵌套关系,可以直观地描述互联网上的自治域与防火墙、自治域中的子域与计算设备、移动代理运行环境与其内部的移动代理程序等对象之间的位置包含关系.无论是一个企业 Intranet 网络,一台便携式计算机,还是一个移动代理,在 MA 中都统一地被形式化为一个灰箱.同时,灰箱的边界具有保护作用,只有穿越边界进入某灰箱的内部,才能与之发生信息交换.MA 对移动计算中的计算性和移动性两方面都有很好的形式化支持.

随着研究的不断深入[2~5],MA 也暴露出一些设计上的不足.文献[2~4]试图从类型的角度加以弥补,但是引入了太多的条件限制,且效果不理想.Levi 等人在文献[6]中首次指出 MA 在语法上存在强干扰(grave interference)问题——由于灰箱移动的单方参与性,归约顺序的不同将会导致完全不同的结果.例如以下的 MA 表达式:

$$h[\ ] \mid n[\text{in } h \mid m[\text{out } n.P]\ ] \tag{1}$$

可归约为(首先灰箱 n 经 in n 动作进入 h,之后灰箱 m 经 out n 动作移出 n,即:先 in 后 out):

$$\rightarrow h[\ n[\ m[\text{out } n.P]\ ]\ ] \rightarrow h[\ n[\ ] \mid m[P]\ ], \tag{2}$$

也可归约为(先 out 后 in):

$$\rightarrow h[\ ] \mid n[\text{in } h] \mid m[P] \rightarrow h[\ n[\ ]\ ] \mid m[P]. \tag{3}$$

由于存在式(2)和式(3)两种不同归约结果,P 无法判断 m 的具体位置,从而影响后续的归约.

* 收稿日期: 2000-06-09; 修改日期: 2001-03-05
基金项目: 上海市科技发展基金资助项目(995115014)

作者简介: 管旭东(1976－),男,江苏常熟人,博士生,主要研究领域为分布移动计算,数据挖掘;杨怡玲(1973－),女,山西太原人,博士生,主要研究领域为数据挖掘,分布移动计算;尤晋元(1939－),男,江苏常州人,教授,博士生导师,主要研究领域为操作系统,分布移动计算,构件与协调技术.

文献[6]在指出强干扰问题的同时,也提出了一种解决方法,通过引入移动的双方参与性来约束归约的顺序,即为每个动作(action)配置相应的反动作(coaction),这种改进的演算被称为安全灰箱(mobile safe ambients,简称SA)演算.

然而,SA 演算虽然解决了 MA 中的强干扰问题,但却同时引入了不安全的因素.其反动作很容易被第三方挪用,从而使双方的移动协议失效.例如以下的两个 SA 表达式:

$$n[\text{in } m.\ \overline{\text{open}}\ n.P] \mid m[\ \overline{\text{in}}\ m.\text{open } n.Q] \mid h[\text{in } m], \tag{4}$$

$$m[\ \overline{\text{out}}\ m.\ \text{in } n.P \mid n[\text{out } m.\ \overline{\text{in}}\ n.Q] \mid h[\text{out } m]\ ]. \tag{5}$$

在式(4)中,n 希望进入 m 后被打开,同样 m 等待 n 的进入并对其执行打开操作.但是,这个协议很容易被与 n 同层的灰箱 h 所破坏:只要 h 也发出一个进入 m 的请求,m 中的 $\overline{\text{in}}\ m$ 动作就有可能先同 h 中的 in m 动作发生归约,并让 h 进入,从而无法正常完成打开 n 的协议.式(5)中也存在类似的问题.

针对以上问题,本文提出一种可以限定反动作对象的抗干扰鲁棒灰箱 (robust ambients,简称 ROAM)演算,在克服 MA 中强干扰问题的同时,也不存在 SA 中的安全漏洞.同时,还给出了一套 ROAM 的类型系统.在表达能力上,ROAM 与 MA 和 SA 一样可以达到描述多元异步π-演算[7]的能力,文章最后给出了一种翻译方案.

1 ROAM 语法的取舍分析

SA 中反动作挪用问题的根源在于没有限制反动作的归约对象.为了达到与 MA 相同的表达能力,SA 将反动作的参数统一设置为包含该反动作的灰箱的名字.这样,对任何 MA 表达式,只要在所有的灰箱 x 中增加进程$!\ \overline{\text{in}}\ x|!\ \overline{\text{out}}\ x|!\ \overline{\text{open}}\ x$,就可转化成等价的 SA 表达式.然而,这种表达能力上的简单性却带来了上述的反动作被挪用的问题.

为了在控制强干扰的同时避免引入负面效应,应该谨慎选择动作和反动作的归约对象.在 SA 中,反动作的参数就是其所在灰箱的名字,在归约中并没有起到任何实质的作用.本文尝试在利用反动作控制强干扰问题的同时,进一步利用反动作的参数限制其归约对象,以预防反动作被挪用.

以下简单讨论各反动作参数的取舍.在随后的章节中,我们将对新演算系统进行形式化引入.

i). $\overline{\text{in}}$: 假设灰箱外部有多个灰箱等待进入,这些灰箱进入的先后次序必须用一定的手段来控制.显然,一个直观的方法是:用 $\overline{\text{in}}$ 所带的参数来指定允许使用该反动作的灰箱.如:$h[\ \overline{\text{in}}\ n\ ] \mid n[\text{in } h]$就可归约为 $h[n[\ ]]$,而$h[\ \overline{\text{in}}\ n\ ] \mid m[\text{in } h]$就不能.

ii). $\overline{\text{out}}$: 同样,在多个灰箱等待跳出同一灰箱时, $\overline{\text{out}}$ 的参数可用来指定哪个灰箱有权使用它.如:$h[\ \overline{\text{out}}\ \ n \mid n[\text{out } h]\ ] \rightarrow n[\ ] \mid h[\ ]$.

iii). $\overline{\text{open}}$: 与上述 $\overline{\text{in}}$ 和 $\overline{\text{out}}$ 相反,只有父灰箱会使用 open 试图打开某一个子灰箱(与该子灰箱的 $\overline{\text{open}}$ 进行归约),这里不存在多个 open 竞争同一个 $\overline{\text{open}}$ 的问题,因此, $\overline{\text{open}}$ 后没有必要再增加一个多余的参数.

另外,对于 3 个传统的动作(in,out,open)中的 out 来说,它所在的灰箱唯一所要跳出的灰箱就是其父灰箱.故此,在引入反动作后,out 后所带的参数也是没有实际作用的.

采用以上描述方法,式(4)和式(5)分别可表示为

$$n[\text{in } m.\ \overline{\text{open}}\ .P] \mid m[\ \overline{\text{in}}\ n.\text{open } n.Q] \mid h[\text{in } m], \tag{6}$$

$$m[\ \overline{\text{out}}\ n.\text{in } n.P \mid n[\text{out}.\ \overline{\text{in}}\ m.Q] \mid h[\text{out}]\ ]. \tag{7}$$

经过上述修改,反动作 $\overline{\text{in}}\ n$ 和 $\overline{\text{out}}\ n$ 被指定只能由灰箱 n 使用,灰箱 h 无法挪用.

2 ROAM 演算

设 N 为名字集,元素用 n 表示,定义 ROAM 演算中的进程集 $Proc$ 和能力集 Cap 如下(其元素分别用 P 和 M 表示):

$$P ::= 0 \mid (\boldsymbol{\nu} n{:}W)P \mid P|Q \mid !P \mid M.P \mid M[P] \mid (n_1{:}W_1,\ldots,n_k{:}W_k).P \mid \langle M_1,\ldots,M_k\rangle$$

$$M ::= \varepsilon \mid n \mid \text{in } M \mid \overline{\text{in}}\ M \mid \text{out} \mid \overline{\text{out}}\ M \mid \text{open } M \mid \overline{\text{open}} \mid M.M'$$

这里的“ν”,“|”,“!”,“.”分别为进程代数中通常使用的名字创建、并置、重复和动作记号;$M[P]$为灰箱记号,代表一个名字为 M、内部活动进程为 P 的灰箱;随后的“( )”和“〈 〉”分别表示消息的输入和输出记号(本文使用了

多元输入输出操作,即以元组作为输入输出的基本单位);3 种基本动作进入、移出、打开分别由 in,out 和 open 表示.这些基本结构的具体含义可参考文献[1].

ROAM 引入了类似 SA 的反动作: $\overline{\text{in}}$, $\overline{\text{out}}$ 和 $\overline{\text{open}}$,同时进一步规定了反动作的使用对象,其具体制约关系参见下文的归约规则.

为减少括号的使用,这里规定"|"的优先级最低.自由名和约束名采用习惯的定义方式:在$(vn{:}W)P$ 和$(n_1{:}W,\ldots,n_k{:}W).Q$ 中,n 和 $n_1,\ldots,n_k$ 分别为约束名,在其他情况下出现的名字为自由名.用 $fn(P)$表示进程 P 中所有自由名的集合.为避免名字捕获,对约束名可使用α-换名规则.

类似其他演算系统,ROAM 定义了对进程 P 和能力 M 的自由名替换操作.用 $P\{M'/n\}$,$M\{M'/n\}$分别表示将 P,M 中的自由名 n 替换为 M'.为书写简单起见,使用 $P\{M_1/n_1,\ldots,M_k/n_k\}$ 表示$(\ldots(P\{M_1/n_1\})\ldots)\{M_k/n_k\}$,$M\{M_1/n_1,\ldots,M_k/n_k\}$表示$(\ldots(M\{M_1/n_1\})\ldots)\{M_k/n_k\}$.

引入 ROAM 演算的归约规则之前,定义 *Proc* 集上(自反、对称、传递)的结构同余关系"≡"如下:

$!0 \equiv 0$ $\quad !P \equiv P \mid !P$ $\quad \varepsilon.P \equiv P$ $\quad (M.M').P \equiv M.(M'.P)$

$P \mid 0 \equiv P$ $\quad P \mid Q \equiv Q \mid P$ $\quad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$

$P \equiv Q \Rightarrow M.P \equiv M.Q$ $\quad P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$ $\quad P \equiv Q \Rightarrow (vn{:}W)P \equiv (vn{:}W)Q$

$P \equiv Q \Rightarrow !P \equiv !Q$ $\quad P \equiv Q \Rightarrow M[P] \equiv M[Q]$ $\quad P \equiv Q \Rightarrow (n_1{:}W_1,\ldots, n_k{:}W_k).P \equiv (n_1{:}W_1,\ldots, n_k{:}W_k).Q$

$(vn{:}W)0 \equiv 0$ $\quad (vn{:}W)\,(vm{:}V)P \equiv (vm{:}V)\,(vn{:}W)P$,如果 $n \neq m$

$(vn{:}W)m[P] \equiv m[(vn{:}W)P]$, 如果 $n \neq m$ $\quad (vn{:}W)\,(P \mid Q) \equiv P \mid (vn{:}W)Q$,如果 $n \notin fn(P)$

ROAM 演算有以下 8 条归约规则:

$n[\,\overline{\text{in}}\ m.P_1 \mid P_2] \mid m[\text{in } n.Q_1 \mid Q_2] \rightarrow n[P_1 \mid P_2 \mid m[Q_1 \mid Q_2]\,]$ $\quad n[\,\overline{\text{out}}\ m.P_1 \mid P_2 \mid m[\text{out}.Q_1 \mid Q_2]] \rightarrow\ n[P_1 \mid P_2] \mid m[Q_1 \mid Q_2]$

$\text{open } n.P \mid n[\,\overline{\text{open}}\ .Q_1 \mid Q_2] \rightarrow P \mid Q_1 \mid Q_2$ $\quad \langle M_1,\ldots,M_k\rangle \mid (n_1{:}W_1,\ldots,n_k{:}W_k).P \rightarrow P\{M_1/n_1,\ldots,M_k/n_k\}$

$$\frac{P \rightarrow P'}{(vn{:}W)P \rightarrow (vn{:}W)P'} \qquad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q} \qquad \frac{P \rightarrow P'}{n[P] \rightarrow n[P']} \qquad \frac{P \equiv P' \quad P' \rightarrow P'' \quad P'' \equiv P'''}{P \rightarrow P'''}$$

ROAM 较之 MA 和 SA,最明显的不同在于归约双方的参与性和相互控制——动作的执行者和动作的授权者必须同时参与归约,且授权者必须明确指定执行者的身份.

3 ROAM 演算的类型系统

结合文献[6]中的线程数概念和文献[3]中的移动性概念,本节提出一套 ROAM 的类型系统.在 ROAM 中,进程和能力包含两个的属性:移动性α和线程数β.进程的消息交换类型概念和能力的 *Amb* 和 *Cap* 两种类型源自 MA 的类型系统[2],基本保留了原有概念的涵义.

设移动性标志集合 *Alpha* 为{Ω,↑},其元素用α表示;线程数标志集 *Beta* 为$\{0,1,1^+,n\}$,其元素用β表示.定义 ROAM 演算的类型语法如下:

W	::= $V\ \alpha\ \beta$,	能力的类型表达式	α	::=	两类移动性标志:
V	::=	有以下两种:		↑,	移动(包含 in 或 out 动作)
	Amb[*S*],	(1) 灰箱能力,如:$n[P]$中的 n		Ω,	固定(否则)
	Cap[*S*],	(2) 动作能力,如:open n,in n,等.	β	::=	四类线程数标志:
T	::= $S\ \alpha\ \beta$,	进程的类型表达式		n,	多线程
S	::=	消息交换有以下两种:		1+,	单线程,但以 open n 结束
	$W_1\times\ldots\times W_k$,	(1) 交换类型为 $(W_1,\ldots,W_k)$的元组		1,	单线程,不以 open n 结束
	Shh,	(2) 不进行消息交换(上式中 k=0)		0,	没有活动线程

此外,*Alpha* 和 *Beta* 集上的 4 种运算定义如下:

i) 二元运算⊗ ii) 二元运算∘ iii) 二元运算 ⊕ iv) 一元运算 ↑

$\alpha_1 \otimes \alpha_2$:

α_1 \ α_2	$\uparrow$	Ω
$\uparrow$	$\uparrow$	$\uparrow$
Ω	$\uparrow$	Ω

$\beta_1 \circ \beta_2$:

β_1 \ β_2	n	1^+	1	0
n	n	n	n	n
1^+	n	n	n	1^+
1	n	1^+	1	1
0	n	1^+	1	0

$\beta_1 \oplus \beta_2$:

β_1 \ β_2	n	1^+	1	0
n	n	n	n	n
1^+	n	n	n	1^+
1	n	n	n	1
0	n	1^+	1	0

β	$\beta^\uparrow$
n	n
1^+	1^+
1	1^+
0	1

同时,为描述集合元素间的关系,分别引入 *Alpha* 和 *Beta* 集上的全序关系"≤".

i) *Alpha* 上的关系"<"为{(Ω,↑)},"≤"为"<"的自反、传递闭包.容易得到以下关于"≤"及运算"⊗"的一些性质:

$\alpha_1 \otimes \alpha_1 = \alpha_1$ $\quad \alpha_1 \otimes \alpha_2 = \alpha_2 \otimes \alpha_1$ $\quad (\alpha_1 \otimes \alpha_2) \otimes \alpha_3 = \alpha_1 \otimes (\alpha_2 \otimes \alpha_3)$

$\alpha_1 \otimes \alpha_2 = \alpha_3 \Rightarrow \alpha_1 \le \alpha_3 \wedge \alpha_2 \le \alpha_3$ $\quad \alpha_1 \le \alpha_2 \Rightarrow \alpha_1 \otimes \alpha_3 \le \alpha_2 \otimes \alpha_3$

ii) *Beta* 上的关系"<"为{(0,1),(1,1$^+$),(1$^+$,*n*)},"≤"为"<"的自反、传递闭包.容易得到以下关于"≤"及运算"⊕","∘","$^\uparrow$"的一些性质:

$(\beta_1 \circ \beta_2) \circ \beta_3 = \beta_1 \circ (\beta_2 \circ \beta_3)$ $\quad \beta_1 \circ \beta_2 = \beta_3 \Rightarrow \beta_1 \le \beta_3 \wedge \beta_2 \le \beta_3$ $\quad \beta_1 \le \beta_2 \Rightarrow \beta_1 \circ \beta_3 \le \beta_2 \circ \beta_3$

$\beta_1 \oplus \beta_2 = \beta_2 \oplus \beta_1$ $\quad (\beta_1 \oplus \beta_2) \oplus \beta_3 = \beta_1 \oplus (\beta_2 \oplus \beta_3)$ $\quad \beta_1 \oplus \beta_1 = \beta_1 \oplus (\beta_1 \oplus \beta_1)$

$\beta_1 \oplus \beta_2 = \beta_3 \Rightarrow \beta_1 \le \beta_3 \wedge \beta_2 \le \beta_3$ $\quad \beta_1 \le \beta_2 \Rightarrow \beta_1 \oplus \beta_3 \le \beta_2 \oplus \beta_3$ $\quad \beta_1 \oplus \beta_2 \le \beta_1^\uparrow \circ \beta_2$

ROAM 的类型推导规则采用与文献[2,3]中相同的判定形式.令Γ表示一个类型环境:

i) $\Gamma \vdash \Diamond$表示Γ为一个合法的类型环境;

ii) $\Gamma \vdash M:W$ 表示能力 M 在环境Γ下是一个合法的能力表达式,其类型为 W;

iii) $\Gamma \vdash P:T$ 表示进程 P 在环境Γ下是一个合法的进程表达式,其类型为 T.

ROAM 的类型推导规则如下(S_{any} 表示任意某个消息传递类型):

$$\frac{}{\phi \vdash \Diamond} \qquad \frac{\Gamma \vdash \Diamond,\ n \notin dom(\Gamma)}{\Gamma, n:W \vdash \Diamond} \qquad \frac{\Gamma', n:W, \Gamma'' \vdash \Diamond}{\Gamma', n:W, \Gamma'' \vdash n:W}$$

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \varepsilon : Cap[S_{\text{any}}]\Omega 0} \qquad \frac{\Gamma \vdash M : Amb[S_1]\alpha\beta}{\Gamma \vdash \text{in } M : Cap[S_{\text{any}}]\uparrow 1} \qquad \frac{\Gamma \vdash M : Amb[S_1]\alpha\beta}{\Gamma \vdash \overline{\text{in}}\ M : Cap[S_{\text{any}}]\Omega 1}$$

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \text{out} : Cap[S_{\text{any}}]\uparrow 1} \qquad \frac{\Gamma \vdash M : Amb[S_1]\alpha\beta}{\Gamma \vdash \overline{\text{out}}\ M : Cap[S_{\text{any}}]\Omega 1} \qquad \frac{\Gamma \vdash M : Amb[S_1]\alpha\beta}{\Gamma \vdash \text{open } M : Cap[S_1]\alpha\beta^\uparrow}$$

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash \overline{\text{open}} : Cap[S_{\text{any}}]\Omega 1} \qquad \frac{\Gamma \vdash M_1 : Cap[S_1]\alpha_1\beta_1,\ \Gamma \vdash M_2 : Cap[S_1]\alpha_2\beta_2}{\Gamma \vdash M_1.M_2 : Cap[S_1](\alpha_1 \otimes \alpha_2)(\beta_1 \circ \beta_2)}$$

$$\frac{\Gamma \vdash \Diamond}{\Gamma \vdash 0 : S_{\text{any}}\Omega 0} \qquad \frac{\Gamma, n : Amb[S_1]\alpha_1\beta_1 \vdash P : S_2\alpha_2\beta_2}{\Gamma \vdash (\nu n : Amb[S_1]\alpha_1\beta_1)P : S_2\alpha_2\beta_2} \qquad \frac{\Gamma \vdash P : S_1\alpha_1\beta_1,\ \Gamma \vdash Q : S_1\alpha_2\beta_2}{\Gamma \vdash P \mid Q : S_1(\alpha_1 \otimes \alpha_2)(\beta_1 \oplus \beta_2)}$$

$$\frac{\Gamma \vdash P : S_1\alpha\beta}{\Gamma \vdash !P : S_1\alpha\ (\beta \oplus \beta)} \qquad \frac{\Gamma \vdash M : Amb[S_1]\alpha_1\beta_1,\ \Gamma \vdash P : S_1\alpha_2\beta_2,\ \alpha_2 \le \alpha_1 \wedge \beta_2 \le \beta_1}{\Gamma \vdash M[P] : S_{\text{any}}\Omega 0}$$

$$\frac{\Gamma \vdash M : Cap[S_1]\alpha_1\beta_1,\ \Gamma \vdash P : S_1\alpha_2\beta_2}{\Gamma \vdash M.P : S_1(\alpha_1 \otimes \alpha_2)(\beta_1 \circ \beta_2)}$$

$$\frac{\Gamma, n_1 : W_1, \ldots, n_k : W_k \vdash P : W_1 \times \ldots \times W_k\alpha\beta}{\Gamma \vdash (n_1 : W_1, \ldots, n_k : W_k)P : W_1 \times \ldots \times W_k\alpha\beta} \qquad \frac{\Gamma \vdash M_1 : W_1, \ldots, \Gamma \vdash M_k : W_k}{\Gamma \vdash \langle M_1, \ldots, M_k \rangle : W_1 \times \ldots \times W_k\Omega 0}$$

以下定理保证了上述类型系统的归约一致性(subject reduction).

定理 1. 如果$\Gamma \vdash P:S_1\alpha_1\beta_1, P \to Q$,那么$\Gamma \vdash Q:S_1\alpha_2\beta_2$ 并且$\alpha_2 \le \alpha_1 \wedge \beta_2 \le \beta_1$.

限于篇幅关系,该命题的完整证明此处省略."$\alpha_2 \le \alpha_1 \wedge \beta_2 \le \beta_1$"表示对进程表达式的归约不会增加其移动性和线程数目:非移动进程不会因归约而变为可移动进程;单线程进程也不会因归约而变为多线程.

4 举 例

在增强了反动作参数限制以后,ROAM 演算仍然具有类似 MA 和 SA 的很强的表达能力.我们以灰箱演算中经典的防火墙跨越(firewall-crossing)例子以及对多元异步π-演算的翻译来说明以上事实.

4.1 防火墙跨越

在文献[1,5]中多次使用防火墙跨越的例子,来说明灰箱演算在安全性上的描述能力.一个授权的代理(agent)可以正确地穿越一个防火墙(firewall)到达其内部.该防火墙(一个灰箱)的名字对外部完全保密,为了让授权的代理正确进入防火墙,防火墙首先送出一个向导灰箱,由于授权代理拥有预先约定好的密钥 k,k'和 k'',因此可以由向导灰箱带领进入防火墙内部.在 ROAM 中,防火墙跨越的例子表示为

$$\text{Agent}::=k'[\overline{\text{in}}\ k.\text{open}\ k.(x{:}W_w).\text{in}\ x.\ \overline{\text{open}}\ |k''[\ \overline{\text{open}}\ |Q]], \tag{8}$$

$$\text{Firewall}::=(\nu w)w[k[\text{out}.in\ k'.\ \overline{\text{open}}\ .\langle w\rangle]|\ \overline{\text{out}}\ k.\ \overline{\text{in}}\ k'.\text{open}\ k'.\text{open}\ k''.P]. \tag{9}$$

在式(8)、式(9)中,进程 P 和 Q 可以是任意的.假设其类型分别为$\Gamma\vdash P{:}S_p\alpha_p\beta_p$,$\Gamma\vdash Q{:}S_q\alpha_q\beta_q$,则灰箱 w,k,k',k''以及 W_w 可分别赋予以下类型:

$$\Gamma\vdash\ w:Amb\ [S_p]\uparrow n \qquad \Gamma\vdash\ k:Amb[S_p]\uparrow 1 \qquad \Gamma\vdash\ k':Amb\ [S_p]\uparrow n$$

$$\Gamma\vdash\ k'':Amb\ [S_q]\alpha_q\beta_q \qquad W_w=Amb\ [S_p]\uparrow n$$

只要保证密钥 k,k',k''的保密性,授权代理利用密钥可顺利进入防火墙内部,并让其内部的进程 Q 在防火墙内部运行,我们可以得到以下结论:

$$(\nu k\ k'\ k'')(\text{Agent}\mid\text{Firewall})\approx(\nu w)w[P\mid Q].$$

这里关系"≈"是一种同余关系(congruence)(由于篇幅关系,在文中并未引入).无论将两个同余的进程放入什么上下文中,所得到的结果都表现出完全相同的外部特性.在 SA 中,防火墙跨越前后的同余性必须通过禁止上下文中存在某类动作来达到[6].ROAM 的鲁棒性表现在对任何存在恶意干扰的上下文,该同余性都成立.

4.2 翻译多元异步π-演算

对多元异步π-演算(polyadic asynchronous π-calculus)的翻译通常可以体现一套演算系统的表达能力.在多元异步π-演算中,通道(channel)n^π 的类型为 $Ch[W_1{}^\pi,\ldots,W_k{}^\pi]$,表示该通道所传递元组中第 i 个元素的类型为 $W_i{}^\pi$.类型判定使用 $\Gamma^\pi\vdash P^\pi$ 的形式,其中Γ^π为类型环境,P^π 为符合正确类型规则的π-进程.ROAM 演算对多元异步π-演算的一种翻译方案如下:

$$[\![0^\pi]\!]::=0 \qquad [\![P^\pi\mid Q^\pi]\!]::=[\![P^\pi]\!]\mid[\![Q^\pi]\!] \qquad [\![!P^\pi]\!]::=!\,[\![P^\pi]\!] \qquad [\![\Gamma^\pi\vdash P^\pi]\!]::=[\![\Gamma^\pi]\!]\vdash[\![P^\pi]\!]:Shh\uparrow n$$

$$[\![\phi,\ n_1{}^\pi{:}W_1{}^\pi,\ldots,n_k{}^\pi{:}W_k{}^\pi]\!]::=\phi,\ n_1{:}\ [\![W_1{}^\pi]\!],\ldots,n_k{:}\ [\![W_k{}^\pi]\!]$$

$$[\![Ch[W_1{}^\pi,\ \ldots,\ W_k{}^\pi]]\!]::=Amb[[\![W_1{}^\pi]\!]\times\ldots\times[\![W_k{}^\pi]\!]]\uparrow n$$

$$[\![(\nu^\pi n^\pi{:}Ch[W_1{}^\pi,\ldots,\ W_k{}^\pi])P^\pi]\!]::=(\nu n{:}\ [\![Ch[W_1{}^\pi,\ldots,\ W_k{}^\pi]]\!])\ (n[!(\overline{in}\ n.\text{open}\ n)]\mid[\![P^\pi]\!])$$

$$[\![n^\pi\ (n_1{}^\pi{:}W_1{}^\pi,\ldots,\ n_k{}^\pi{:}W_k{}^\pi).P^\pi]\!]::=(\nu p{:}Amb[Shh]\uparrow n)(\text{open}\ p\mid n[in\ n.\ \overline{\text{open}}\ .(n_1{:}\ [\![W_1{}^\pi]\!],\ldots,\ n_k{:}\ [\![W_k{}^\pi]\!])$$
$$(p[\text{out}.\ \overline{\text{open}}\ .\ [\![P^\pi]\!]\ ]\mid\overline{\text{out}}\ p)])$$

$$[\![n^\pi\langle n_1{}^\pi,\ldots,\ n_k{}^\pi\rangle]\!]::=n[in\ n.\ \overline{\text{open}}\ .\langle n_1,\ldots,\ n_k\rangle]$$

这里,一个π-通道 n^π 表示为一个同名灰箱 n.该灰箱中的进程!($\overline{in}\ n$.open n)不断地让试图在该通道上进行通信的灰箱进入,并将它们打开.在 n^π 上进行输入和输出操作被翻译为若干进入 n 并被打开的灰箱.输入和输出灰箱被相继打开以后,消息传递便在 n 中进行.消息传递完毕,得到输入的进程跳出 n 并继续运行.这样,在 n^π上进行元组通信被转化为在灰箱 n 中进行多元消息传递.

以上翻译方案对一个π-通道名只使用了一个灰箱名,既表示该π-通道,又代表进入该通道进行消息交换的输入输出灰箱.当然,可以分别再使用两个不同的灰箱名分别表示输入和输出灰箱,这样,每个灰箱的类型还可以定义得更加具体,如表示通道的灰箱还可更加明确.但是,上述简单方案足以说明 ROAM 的表达能力问题.

5 结论和进一步的工作

本文基于 MA 和 SA 的工作,提出了鲁棒灰箱演算系统——ROAM,进一步改进了 SA 中对 MA 强干扰的控制,克服了 SA 中存在的安全隐患.通过对防火墙跨越和多元异步π-演算的描述,表明 ROAM 与 MA 和 SA 类似,具有很强的描述能力.同时,本文提出了一套可以描述进程和能力的移动性和线程数两个属性的类型系统,并证明了该类型系统的归约一致性.针对 ROAM 的进一步工作有子类型系统的建立和证明、单线程进程的性质研究等.

References:

[1] Cardelli, L., Gordon, A.D. Mobile ambients. Lecture Notes in Computer Science, 1998,1378:140～155.

[2] Cardelli, L., Gordon A.D. Types for mobile ambients. In: ACM, ed. Proceedings of the POPL'99, New York: ACM Press, 1999. 79～92.

[3] Cardelli, L., Ghelli, G., Gordon, A.D. Mobility types for mobile ambients. Lecture Notes in Computer Science, 1999,1644:230～239.

[4] Cardelli, L., Ghelli, G., Gordon, A.D. Ambient groups and mobility types. Lecture Notes in Computer Science, 2000,1872:333～347.

[5] Gordon, A.D., Cardelli, L. Equational properties of mobile ambients. Lecture Notes in Computer Science, 1999.1578:212～226.

[6] Levi, F., Sangiorgi, D. Controlling interference in ambients. In: Thomas, R., ed. Proceedings of the POPL 2000. New York:ACM Press, 2000. 352～364.

[7] Milner, R., Parrow, J.G., Walker, D.J. A calculus of mobile process. Information and Computation, 1992,100(1):1～77.

Further Control on the Grave Interference in Mobile Ambient*

GUAN Xu-dong, YANG Yi-ling, YOU Jin-yuan

(*Distributed Computing Technology Center*, *Department of Compuer Science and Engineering*, *Shanghai Jiaotong University*, *Shanghai* 200030, *China*)

E-mail: guan-xd@cs.sjtu.edu.cn; yang-yl@cs.sjtu.edu.cn; you-jy@cs.sjtu.edu.cn.ac.cn

http://dctc.sjtu.edu.cn

Abstract: In order to control the grave interference in mobile ambient (MA), Levi *et al.* proposed mobile safe ambients (SA). However, the coactions introduced in SA brought new security breaches. In this paper, robust ambients (ROAM) is proposed to eliminate those security breaches. In ROAM, coactions are still utilized to control the grave interference. In addition, the parameter of every coaction is explicitly specified to name the consumer of that coaction. This mechanism effectively eliminates the security breaches in SA. The firewall crossing example and the encoding of polyadic asynchronous π-calculus in ROAM show that ROAM still keeps the strong expressiveness of its ancestors. A fundamental type system for ROAM with both thread count and mobility attributes is also proposed and proved. The result in this paper shows that ROAM is a good candidate in the formalization of mobile computation.

Key words: process algebra; mobile ambient; mobile safe ambient; robust ambient; type

* Received June 9, 2000; accepted March 5, 2001
Supported by the Science and Technology Development Foundation of Shanghai of China under Grant No.995115014