Ambient 演算的数据流分析方法及其实现*)

张 晶[1] 张丽翠[2] 金成植[1]

(吉林大学计算机科学与技术学院 长春 130012)[1] (吉林大学通信工程学院 长春 130012)[2]

摘 要 针对 Ambient 演算的定义,提出了 Ambient 演算语法结构树的概念,并且给出一个基于 Ambient 语法结构树的 Ambient 演算的数据流分析方法及其实现。为深入研究 Ambient 演算的性质和应用提供了分析手段。

关键词 Ambient 演算,数据流分析,Ambient 语法结构树

Method and Implementation of Data Flow Analysis for Ambient Calculus

ZHAGN Jing[1] ZHANG Li-Cui[2] JIN Cheng-Zhi[1]

(College of Computer Science and Technology, Jilin University, Changchun 130012)[1]

(College of Communication Engineering, Jilin University, Changchun 130012)[2]

Abstract According to the definition of ambient calculus, we propose the concept of ambient syntactic structure tree, and introduce a data flow analysis method for ambient calculus based on the structural tree of ambient calculus and the implementation of the analysis algorithms. The present paper provides a way to further studying the features and application of ambient calculus.

Keywords Ambient calculus, Data flow analysis, Ambient syntactic structure tree

1 引言

Mobile Ambients[1]演算是作为 Web 的核心编程语言及推理移动进程属性如安全性的模型提出来的。经典的移动进程形式化模型 π 演算中移动是通过通信(communication)实现的,而 MA 中移动是通过进程迁移(movement)实现的。Ambient 是一个有名字的位置,Ambient 是迁移的基本单位。Ambient 内部还可以包含顺序进程和子 Ambient. 同一 Ambient 内部的进程可以相互交换消息。Ambient 内部还可以有子 Ambient,因此形成一种层次结构。

Ambient 演算中的名字有两类,一类是变量名字,一类是 ambient 名字,其中变量名字的使用遵循先定义后使用的原则,而 Ambient 名字则不遵循这一原则,因此传统的数据流分析技术不能直接用于对 Ambient 演算进行分析,本文在文[2]中给出的 Action 演算的数据流分析技术的基础上,基于 Ambient 演算的层次结构,提出了 Ambient 演算的语法结构树的概念,并分析了 Ambient 程序的数据流动情况,最终给出 Ambient 的数据流分析算法。为深入研究 Action 演算的性质和应用提供了必要的分析手段。

2 Ambient 演算

本文中 Ambient 演算的定义是基于文[1,3,4]给出的。

(1)语法

进程 $P,Q ::= 0 \mid (vn:T)P \mid P \mid Q \mid !\ P \mid n[P] \mid M.\ P \mid (x).\ P \mid \langle M \rangle$

表达式 $M,N ::= x \mid n \mid \text{in } M \mid \text{out } M \mid \text{open } M \mid \varepsilon \mid M.\ M'$

语法意义:0 是无活动进程;$(vn)P$ 引入新名字 n 且限定作用域是 P;$P|Q$ 代表 P,Q 并行执行;$!\ P$ 表示 P 的无限复制;$n[P]$表示名字为 n 的环境,进程 P 运行于其中;$M.\ P$ 进程表示执行由能力 M 调节的动作,然后继续进程 P;(x)是输入动作,进程$(x).\ P$ 在其包围的环境中接收名字 x;$\langle M \rangle$是异步输出动作,将 M 放于包围的环境。M 表示表达式,它可以是变量 x,环境名字 n,进入环境能力 in,退出能力 out,开放能力 open,空或能力组合路径。

(2)语义

Ambient 演算以非确定的化学反应风格给出语义。操作语义由描述进程演化的归约关系→ 和结构全等关系≡定义。

1) $P \equiv P$; 2) $P \equiv Q \Rightarrow Q \equiv P$;3)③ $P \equiv Q, Q \equiv R \Rightarrow P \equiv R$;

4) $P \equiv Q \Rightarrow (vn)P \equiv (vn)Q$;5) $P \equiv Q \Rightarrow P|R \equiv Q|R$;

6) $P \equiv Q \Rightarrow !\ P \equiv !\ Q$;7) $P \equiv Q \Rightarrow n[P] \equiv n[Q]$;

8) $P \equiv Q \Rightarrow M.\ P \equiv M.\ Q$;9) $P \equiv Q \Rightarrow (n).\ P \equiv (n).\ Q$;

10) $P|Q \equiv Q|P$;11) $(P|Q)|R \equiv P|(Q|R)$;

12) $!\ P \equiv P|!\ P$;13) $(vn)(vm)P \equiv (vm)(vn)P$ if $n \neq m$

14) $(vn)(P|Q) \equiv P|(vn)Q$ if $n \notin fn(P)$;

15) $(vn)m[P] \equiv m[(vn)P]$ if $n \neq m$;16) $P|0 \equiv P$;

17) $(vn)\ 0 \equiv 0$;18) $!\ 0 \equiv 0$;

19) $\varepsilon.\ P \equiv P$;20) $(M.\ M').\ P \equiv M.\ M'.\ P$

进程行为由归约关系定义:

① $n[\text{in } m.\ P|Q]|m[R] \rightarrow m[n[P|Q]|R]$

② $m[n[\text{out } m.\ P|Q]|R] \rightarrow n[P|Q]|m[R]$

③ $open\ n.\ P|n[Q] \rightarrow P|Q$

④ $(x).\ P|\langle M \rangle \rightarrow P\{x \leftarrow M\}$

⑤ $P \rightarrow Q \Rightarrow (vn)P \rightarrow (vn)Q$

⑥ $P \rightarrow Q \Rightarrow n[P]\ n[Q]$

⑦ $P \rightarrow Q \Rightarrow P|R \rightarrow Q|R$

⑧ $P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$

⑨ $\rightarrow^*$ 归约关系的传递反射闭包

3 Ambient 演算的数据流分析

定义 3.1 设 ambient a 的一般形式为:$n[P_1|\cdots|P_p|m_1[\cdots]|m_q[\cdots]]$ for $p,q \geqslant 0$ 其中每个 P_i 是以 in,out 或 open 开始的顺序进程,则满足下面条件的树称为 a 的语法结构树:每个节点都标有相关的信息:节点的种类,节点的编号,输入变量名字集合,输入 ambient 名字集合,指向父节点的指针,

*)基金项目:吉林省科技厅项目(批准号:20050527)。张 晶 博士生,讲师,从事形式化方法和程序理论方面的研究。

指向子节点的指针，指向兄弟节点的指针，并且满足：

(1)根节点标有“ambient 种类 ambiTy，节点的编号 0，节点的输入变量名字集合为空，输入 Ambient 名字集合{Super-Ambient}”；

(2)如果一个节点是 ambient 类的节点，其输入变量名字集合为$(\tilde{x})$，输入 ambient 名字集合为$(\tilde{m})$，且有 s 个子节点(从左至右)对应的 ambient 为 $a_1[],\cdots,a_s[]$，有 t 个弟节点(从左至右)对应的进程形式为 $P_1\cdots P_t$，则一定有$(\nu\tilde{x},\nu\tilde{m})a[a_1[]\parallel\cdots\parallel a_s[]\parallel P_1\parallel\cdots\parallel P_t]$，其中 a 是该节点的名字；

(3)如果一个节点是进程类的节点，其输入变量名字为$(\tilde{x})$，输入 ambient 名字集合为$(\tilde{m})$，且有 t 个弟节点(从左至右)对应的进程形式为 $P_1\cdots P_t$，则一定有$(\nu\tilde{x},\nu\tilde{m})(P_1\parallel\cdots\parallel P_t)$；

(4)进程类的节点都是叶节点，没有子节点的 ambient 类节点也是叶节点。

Ambient 演算中的名字有两类，一类是变量名字，一类是 ambient 名字。在 ambient 进程 $n[P]$中，当 ambient 名字 n 的直接外层，次外层，乃至最外层 ambient 中均没有名字 n 的定义性出现，则称 $n[P]$中的 n 是定义性出现。在约束进程$(\nu\tilde{x},\nu\tilde{m})(P_1\parallel\cdots\parallel P_t)$和输入进程中$(\tilde{x}).P$或$(\tilde{m}).P$中，变量名字序列 $\tilde{x}$ 和 ambient 名字序列 $\tilde{m}$ 是定义性出现，其它情况下变量名字序列和 ambient 名字序列的出现都是使用性出现。

定义 3.2 在 Ambient 演算的某个 ambient 演算或某个进程中，如果一个名字的使用性出现是在该名字的某个定义的作用域内，则称该名字的使用性出现为绑定出现，或称该名字为被绑定的。

定义 3.3 在 Ambient 演算的某个 ambient 演算或某个进程中，如果一个名字是使用性出现，而且在该 ambient 或进程中没有绑定该名字的定义性出现，则称该名字是自由出现的。

由于在 Ambient 演算中可能会出现名字的嵌套定义，同时，Ambient 演算中变量名字和 Ambient 名字的作用域规则是不同的，因此，下面分别给出 Ambient 演算中变量名字和 Ambient 名字的的嵌套作用域规则。

(1)变量名字的作用域规则

1)Ambient 层变量的作用域：在 Ambient 名字前面声明的变量，其作用域为这个 ambient 演算内的各个没有相同变量声明的并发进程和这个 ambient 演算内的没有相同变量声明的各个子 Ambient。例如：在

$(\nu y:t_1,\nu x:t_2)s[P_1\parallel s_1[P_2\parallel(\nu x:\text{integer})P_3]\parallel(\nu x:\text{integer})s_2[P_5]\|P_4]$

中，由于变量 y 只在 ambient s 前声明一次，因此变量 y 的作用域是 ambient s 及其所有子 ambient 的进程体和 s 的所有进程体，即 P_1,P_2,P_3,P_4 和 P_5。而 ambient s 中变量 x 的作用域则不同，它的作用域是进程 P_1,P_2 和 P_4。由于 ambient s_1 中的进程 P_3 前又重新声明了变量 x，因此第一个变量 x 的作用域不包括 P_3，由于 ambient s_2 前又重新声明了变量 x，所以第一个变量 x 的作用域也不包括 ambient s_2。总之，第一个变量 x 的作用域包括 P_1,P_2,P_4。第二个变量 x 的作用域为 P_3。第三个变量 x 的作用域为 P_5。

2)进程层变量的作用域：在一个 ambient 演算内的某个参与并发的顺序进程前声明的变量的作用域就是这个顺序进程本身。例如：$(\nu x:t_1)P$，则变量 x 的作用域就是进程 P。

3)里层优先原则：在有嵌套的 ambient 层次结构中，作用域以最近一层为主，即里层优先。例如：在

$(\nu x:t_1)s_1[(\nu x:t_2)P_1\|(\nu y:t_3)P_3\|(\nu z:t_4)\ s_2[P_4]]$

中，有两个地方声明了变量 x。在 P_1 中出现的 x 具有类型 t_2(P_1 中的变量 x 是里层变量)，在进程 P_3 和进程 P_4 中出现的 x 具有类型 t_1(因为这两个进程没有重新声明变量 x)。

(2) Ambient 名字的作用域规则

Ambient 名字有三种定义性出现，一是在 ambient 定义时出现的 ambient 名字，如给定一个 $n[P]$，则 ambient 名 n 就是在定义 ambient 时候出现的；二是在声明中出现的 ambient 名字，如在$(\nu n)P$中，进程 P 前面的声明中出现的 ambient 名字 n 就是名字 n 的定义性出现；三是在$(x).P$中出现的 ambient 名字。对于第一种定义性出现的 ambient 名字，如果没有父 ambient，则其作用域为其本身。若有父 ambient，则其作用域为其父 ambient 中非 ambient 的并发进程。第二和第三种定义性出现的 ambient 名字，其作用域是进程 P。

根据上述作用域原则，基于 ambient 语法结构树，我们将分析一个 Ambient 演算程序的数据流情况，主要内容包括如下两个方面：

(1)对于一个名字 x 的使用性出现，确定绑定它的定义性出现。分析方法：对于一个给定路径 p 中的名字 x，可以沿着该路径 p，自顶向下遍历语法结构树，确定当前有效的定义性名字，当找到当前名字 x 的使用性出现时，如果存在有效的 x 的定义性出现，即为绑定 x 的定义性出现，否则名字 x 为自由出现。

(2)对于一个名字 x 的定义性出现，确定它所绑定的所有使用性出现。分析方法：对于一个给定路径 p 的名字 x，可以沿着该路径 p，找到当前名字 x 的定义性出现的节点，对以该节点为根节点的子树，先根遍历，名字 x 有效的节点上的 x 使用性出现皆为所绑定的所有使用性出现。

4 数据流分析的实现算法

为实现 Ambient 演算的数据流分析，本文给出了相应的 ambient 语法结构树(构造方法略)，并给出了相应的实现算法。

定义 ambient 语法结构树中节点的数据结构为：

```
NODE=Struct
        Flag:BOOLEAN;
        NodeName:STRING;
        NodeNumber:INTEGER;
        InVarNames:* STRING;
        InAmbiNames:* STRING;
        FatherNode :* NODE;
        SonNodes:ARRAY[1..MaxSon] of * NODE;
        BrotherNodes:ARRAY[1..MaxBro] of * NODE;
END
```

其中 Flag：TRUE 表示 ambient，FALSE 表示顺序进程；NodeName 表示对应节点的名字；NodeNumber 表示对应节点的编号；InVarName 为输入变量名字序列；InAmbiName 为输入 Ambient 名字序列；FatherNode 为父节点指针；SonNodes 为子节点指针数组；BrotherNode 为兄弟节点指针数组。

由于变量名字和 Ambient 名字的作用域规则是不同的，所以对于变量名字和 Ambient 名字分开进行分析。

(1)对于一个名字 x 的使用性出现，确定绑定它的定义性出现

算法 FindVarDefPath(AmbiTree：TREE，boundPath：PATH，x：NAME)：PATH；

参数：Ambi 语法树 AmbiTree，一个使用性变量名字 x 及其出现路径；

输出：绑定该出现 x 的定义性出现的路径。

1)根据 boundPath，找到使用性出现节点 node。

2)逆向搜索找到对应的定义性出现：

(下转第 255 页)

念格使用用例图只是一种方法，对于 aspect 挖掘可以根据 aspect 挖掘的目标使用其它的构造概念格的方法。

结束语 论文提出一种结合 Fan-In 理论和概念分析的技术进行 aspect 挖掘的方法，这种方法比单纯的对目标系统做概念分析更有效。单纯的概念分析对于较大的系统，收集的信息过多，构造的概念格变得非常庞大，构造的过程和对概念格分析的过程费时费力。该方法先通过 Fan-In 分析，将那些难以成为 aspect 种子的方法过滤掉，实验证明这种先作扇入分析的方法可以剔出 95% 左右的非 aspect 种子的方法。但是该方法由于是通过收集动态信息来生成用例，因此由于程序运行很难遍历所有的分支，所以收集的信息不能完全反映目标系统，因此对 aspect 的挖掘也不可能非常准确，但是这种方法还是能将大多数的 aspect 挖掘出来。同是该方法现在还没有完全的自动完成，对于概念分析仍然需要手工完成，论文下一步的工作是要将该方法能自动实现。

参 考 文 献

1 Fowler M. Refactoring: Improving the design of existing code. Addison-Wesley Publishing Company, Reading, MA, 1999
2 Tonella P, Ceccato M. Migrating interface implementation to aspect oriented programming. In: Proc. of the International Conference on Software Maintenance (ICSM), page (to appear), Chicago, Illinois, USA, September 2004. IEEE Computer Society Press
3 Hannemann Jan, Kiczales Gregor. Design Pattern Implentation with Java and AspectJ [M]. JavaWorld, 2003. 9
4 Loughran N, Rashid A. Mining aspects. In: Proc. Of the Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (with AOSD), Enschede, The Netherlands, April 2002
5 Marin M, van Deursen A, Moonen L. Identifying aspects using fan-in analysis. Technical Report SENR0413, CWI, 2004
6 Tonella P, Ceccato M. Aspect Mining through the Formal Concept Analysis of Execution Traces. WCRE, 2004. 9
7 Ganter B, Wille R. Formal Concept Analysis. Springer-Verlag, Berlin, Heidelberg, New York, 1996
8 Kiselev I. Aspect-Oriented Programming with AspectJ. Sams Publishing, Indianapolis, Indiana, USA, 2002

（上接第 248 页）

如果 node 对应的是顺序进程，则如果 node 的输入名字集合中存在 x，则{goto 3；}否则{node= 它的长兄节点（即包含该进程的 ambient 节点），goto 2}；

如果 node 对应的是 ambi 进程，则如果 node 是根节点，则{goto 4；}否则如果 node 的输入名字集合中存在 x，则{goto 3；} 如果 node 的输入名字集合中不包含 x，则{node= 它的父节点，goto 2；}；

3）找到对应的定义性出现在节点 node 上，确定它的出现路径：从根节点到 node 的路径上的内部编号序列确定了名字的出现路径 path，return path；

4）没有找到对应的定义性出现，表明出错，retrun −1。

算法 FindAmbiDefPath（AmbiTree：TREE，x：NAME，flag：BOOLEAN）：PATH

参数：Ambi 语法树 AmbiTree，一个使用性 ambient 名字 x；

输出：绑定该出现 x 的定义性出现的路径。如果 ambient 名字 x 的定义点不是根节点，则 flag=true；否则 flag=false。

1）找到名字为 x 的 ambi 类型节点 node：对给定的 AmbiTree 从根节点开始进行部分先序搜索，即搜索顺序为节点、它的所有儿子节点，不包括它的弟节点。若找到名字为 x 的 ambi 类型节点 node，则转 2，否则转 6。

2）确定 node 的出现路径：从根节点到 node 的路径上的内部编号序列，确定了名字 x 的一个可能定义路径 path1。

3）逆向搜索找到对应的定义性出现：如果 node 是根节点，则{flag=false；goto 4；}；否则如果 node 的输入 ambient 名字集合中包含 x，则{flag= true；goto 5；} 否则{node= node 的父亲节点，goto 3；}。

4）找到名字 x 的定义性出现路径：retrun path1。

5）确定 node 的出现路径：从根节点到 node 的路径上的内部编号序列，确定了名字 x 的出现路径 path. return path。

6）没有找到对应的定义性出现，表明出错，retrun −1。

（2）对于一个名字 x 的定义性出现，确定它所绑定的所有使用性出现

算法 FindVarBoundPaths（AmbiTree：TREE，x：NAME，path：PATH）：set of PATH；

参数：Ambi 语法树 AmbiTree，一个定义性变量名字 x 及其出现路径；

输出：定义性名字 x 所绑定的所有使用性出现的路径集合。

1）初始化 set={}；

2）根据 path 在 AmbiTree 中找到对应定义性变量名字 x 出现的节点 node；

3）如果 node 对应的是一个顺序进程，则 return path；

4）如果 node 对应的是一个 ambi 进程，则

如果 node 有弟节点，则

{ 对于 node 的每个弟节点 brother，做：如果 brother 的输入名字集合中没有 x，则

{ sibpath= path 加上对应 brother 的编号；

set= set∪{sibpath}；}}

如果 node 有子节点，则

{ 对于 node 的所有子节点 son，做：如果 son 的输入名字集合中没有 x，则

{ sonpath= path 加上对应 son 的编号；

set= set ∪{sonpath}；}

node=son；

goto 3；}}

return set；

算法 FindAmbiBoundPaths（AmbiTree：TREE，x：NAME，path：PATH）：set of PATH；

参数：Ambi 语法树 AmbiTree，一个定义性 Ambient 名字 x 及其出现路径；

输出：定义性名字 x 所绑定的所有使用性出现的路径集合。

1）初始化 set={}；

2）在 AmbiTree 中找到对应定义性 ambient 名字出现的节点 node 及出现路径 path：

path= FindAmbiDefPath(AmbiTree, x, flag)；

如果 flag= true && node 有弟节点，则

{ 对于 node 的所有弟节点 brother，做：

{ sibpath= path 加上对应 brother 的编号；

set= set ∪sibpath；}

return set}

3）如果 flag= false，则 return path；

结论 本文针对 Ambient 演算的层次定义，提出了 Ambient 演算语法结构树的概念，并且在 Windows XP 环境下用 Visual C++ 实现了 Ambient 演算的数据流分析程序，该程序以 ambient 的语法结构树为输入，对于名字的使用性出现，可以给出绑定它的定义性出现路径，以及对于名字的定义性出现，可以给出它所绑定的所有使用性出现路径。为进一步研究 Ambient 演算的性质和实现技术提供了有效的手段和环境。

参 考 文 献

1 Cardelli L, Gordon A D. Mobile Ambients In: Proc. FoSSaCS'98, Lecture Notes in Computer Science, Springer Verlag, 1998, 1378: 140～155
2 金英，金成植，郑爽. Action 演算中动作内部数据流分析方法及其实现. 吉林大学学报（理学版），2003，41：94～98
3 Cardelli L, Gordon A D. Types for mobile ambients. In: Proc. 26th Annual ACM Symposium on Principles of Programming Languages, 1999. 79～92
4 Cardelli L, Gordon A D. Mobility types for Mobile Ambients. In Proc. Of ICALP, Lecture Notes in Computer Science, Springer Verlag, 1999, 1644: 230～239
5 张丽翠. 具有并行性和移动性的演算语言的形式语义研究：[吉林大学博士学位论文]. 2005. 4