基于安全进程代数的非演绎安全模型的分析与验证

王精明[1,2] 虞慧群[1]

(华东理工大学计算机科学与工程系 上海 200237)[1] (滁州学院计算机与信息工程学院 滁州 239012)[2]

摘 要 就刻画安全的性质而言,基于非演绎信息流的安全模型较基于访问控制的安全模型更为确切和本质。在基于迹语义对非演绎信息流安全模型进行分析的基础上,基于安全进程代数给出非演绎模型的形式化描述,然后基于系统的安全进程代数表达式给出非演绎模型的验证算法且开发了相应的验证工具,最后通过实例说明该算法的正确性和验证工具的方便适用性。

关键词 迹语义,安全进程代数,信息流安全模型,非演绎模型

中图法分类号 TP309 **文献标识码** A

Security Process Algebra-based Analysis and Verification of Non-deducibility Security Model

WANG Jing-ming[1,2] YU Hui-qun[1]

(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]

(School of Computer and Information Engineering, Chuzhou University, Chuzhou 239012, China)[2]

Abstract In characterizing security, non-deducibility security model is more essential and accurate than access control security model. This paper described and formally defined non-deducibility information flow security model based on trace semantics and security process algbra. Furthermore, this paper provided the verification algorithm for non-deducibility security model based on security process algebra and offered the verification tools, whose use was illustrated with several examples.

Keywords Trace semantics, Security process algebra, Information flow security model, Non-deducibility model

信息的机密性是分级安全系统研究的核心问题之一,安全模型的研究主要分为基于信息流的安全模型和基于访问控制的安全模型两大类[1]。基于访问控制的安全模型主要侧重于定义怎样做才能保证系统是安全的,而信息流安全模型侧重于安全是什么,从这点上说,就刻画安全的性质而言,基于信息流的安全模型较基于访问控制的安全模型更为确切和本质[2]。自 Sutherland 于 1986 年首次提出非演绎(non-deducibility)信息流安全模型以来,基于非演绎模型的研究不断深入,且其应用也越来越广泛,如文献[3]对信息流安全模型组合问题进行了较深入的研究;文献[4]着重研究了大型基础设施(如国家电网)的信息流安全问题;文献[5]深入探讨了当前研究热点之一:信息物理融合系统中的信息流安全模型。

Simon N. Foley 等人[2,5]是将进程代数引入信息流领域研究的先导者,之后进程代数就作为研究信息流理论的主要工具之一。安全进程代数 Security Process Algebra(缩写为 SPA)是在 Communication of Concurrency System(CCS)基础上做了简单扩展而成,它给研究者探索信息流安全模型提供了方便,有很多基于安全进程代数研究信息流的成果,见文献[6-10]。

文献[1-10]主要工作为基于安全进程代数对部分信息流安全模型进行形式化描述和部分性质的证明,虽然文献[13,14]基于 Petri 网对部分信息流安全模型进行了分析和验证,但上述文献均没有对非演绎安全模型进行分析并提出相应的验证算法。鉴于非演绎安全模型及其基于进程代数的分析和验证方面的研究成果相对较少,本文的主要工作为基于安全进程代数对非演绎安全模型进行形式化表述,进一步基于安全进程代数表达式给出非演绎模型的验证算法并开发相应的验证工具,最后通过实例说明该算法的正确性和验证工具的方便适用性。

本文第 1 节简要介绍本文涉及的基本概念和定义;第 2 节基于迹语义对非演绎信息流安全模型进行了分析和描述,进一步基于安全进程代数给出了非演绎模型的形式化描述;第 3 和第 4 节给出了基于安全进程代数的非演绎模型的验证算法,开发了相应的验证工具,且通过实例说明了该算法的正确性和验证工具的方便适用性;最后对本文进行了简要总结。

1 相关定义和概念

1.1 标识变迁系统

作为并发语言操作语义描述的首选模型,标识变迁系统 Labelled Transition Systems(缩写为 LTS)可以理解为具有无

到稿日期:2011-04-08 返修日期:2011-06-30 本文受国家自然科学基金(61173048,60773094,60473055),上海市曙光计划(07SG32),安徽省高校省级自然科学研究基金项目(KJ2011Z279),安徽省滁州市滁州学院重点自然科学基金项目(2010kj008Z)资助。

王精明(1979—),男,博士生,主要研究方向为信息流安全模型、形式化方法;虞慧群(1967—),男,教授,博士生导师,主要研究方向为软件工程、形式化方法,E-mail:yhq@ecust.edu.cn(通信作者)。

穷状态的自动机，具体可参考文献[11]。本文仍然采用标识变迁系统表示文中涉及到的 SPA 操作语义。

定义 1 变迁系统可以理解为一个三元组$\langle S,T,\rightarrow\rangle$，其中：

①S 表示状态集合；

②T 表示动作集合；

③$\rightarrow\subseteq S\times T\times S$ 表示标记变迁集合。

如$(S_1,\alpha,S_2)\in\rightarrow$(可简记为 $S_1\xrightarrow{\alpha}S_2$)表示系统通过发生动作 α 从状态 S_1 到达状态 S_2。

1.2 迹语义

由于本文基于迹语义讨论非演绎安全模型，故本节简要介绍与迹相关的定义和概念，关于迹语义的详细定义可参考文献[12]。

定义 2 表达式 $E\overset{\mu}{\Rightarrow}E'$代表 $E(\xrightarrow{\tau})^*E1\xrightarrow{\mu}E2(\xrightarrow{\tau})^*E'$的缩写形式，其中$(\xrightarrow{\tau})^*$代表 0 个或若干个 τ 动作组成的动作序列。$E\overset{\mu}{\Rightarrow}$表示存在 E'使得 $E\overset{\mu}{\Rightarrow}E'$，$E\overset{K}{\Rightarrow}$表示对任意 $K\subseteq\mathcal{L}$，$\forall\mu\in K$，均有 $E\overset{\mu}{\Rightarrow}$。$E\overset{\sigma}{\Rightarrow}E'$，其中 $\sigma\in\mathcal{L}^*$，$\sigma=\alpha_1\alpha_2\cdots\alpha_n$ 代表存在着 E_1，$E_2\cdots,E_{n-1}$使得 $E\overset{\alpha_1}{\Rightarrow}E_1\overset{\alpha_2}{\Rightarrow}E_2\cdots\overset{\alpha_{n-1}}{\Rightarrow}E_{n-1}\overset{\alpha_n}{\Rightarrow}E'$，若对于空序列$\langle\rangle$有 $E\overset{\langle\rangle}{\Rightarrow}E'$表示 $E(\xrightarrow{\tau})^*E'$，也说 E'是从 E 可达的。

定义 3 ε 表示所有进程的集合，对任意 $E\in\varepsilon$，$T(E)=\{\gamma\in\mathcal{L}^*\mid\exists E':E\overset{\gamma}{\Rightarrow}E'\}$，任意 $E\in\varepsilon$ 和 $F\in\varepsilon$，E 和 F 是迹等价的(符号表示为 $E\approx_{tr}F$)当且仅当 $T(E)=T(F)$。

1.3 安全进程代数

为了便于形式化描述系统(进程)，本节简要介绍扩展于进程代数的 SPA 的相关概念。

SPA 语法构成要素如下：集合 $I=\{a,b,\cdots\}$表示输入动作集合；集合 $O=\{\bar{a},\bar{b},\cdots\}$表示输出动作集合；集合 $\mathcal{L}=I\cup U$ 表示可见动作集，其元素一般用 α,β 表示，在该集合上定义函数 $\overline{\cdot}:\mathcal{L}\to\mathcal{L}$ 使得 $a\in I\Rightarrow\bar{a}\in O$ 且 $\bar{a}\in O\Rightarrow\bar{\bar{a}}=a\Rightarrow I;Act_H$ 和 Act_L 分别表示高级和低级动作集合且$\overline{Act_H}=Act_H$，$\overline{Act_L}=Act_L$，$Act_H\cup Act_L=\mathcal{L}$，$Act_H\cap Act_L=\emptyset$，$L\underline{def}\{\bar{a}:a\in L\}$；集合 $Act=\mathcal{L}\cup\{\tau\}$表示动作集，其元素一般用 μ 表示，其中 τ 表示内部动作即不可见动作；集合 $Act_{LI}=Act_L\cap I$、$Act_{LO}=Act_L\cap O$、$Act_{HI}=Act_H\cap I$ 和 $Act_{HO}=Act_H\cap O$ 分别表示低级输入动作、低级输出动作、高级输入动作和高级输出动作；K 代表进程常量集合，其元素一般用 Z 等表示。

安全进程代数进程的语法定义如下：

$E::=0\mid\mu.E\mid E+E\mid E\mid E\mid E\backslash L\mid E\mid\mid E\mid E/L\mid E[f]\mid Z$

式中，$L\subseteq\mathcal{L}$，函数 $f:Act\to Act$ 使得 $f(\bar{\alpha})=\overline{f(\alpha)}$，$f(\tau)=\tau$。0、$\mu.E$、$E+E$、$E\mid E$、$E\backslash L$、$E[f]$和 $Z\underline{def}E$ 与 CCS 中含义基本相同，具体含义为 0 表示空进程，不发生任何动作；前缀算子 $\mu.E$ 表示发生 μ 动作后到达进程 E；选择算子 E_1+E_2 表示在 E_1 和 E_2 中选且仅选一个；并行算子 $E_1\mid E_2$ 代表两个进程并行交叉执行；限制算子 $E\backslash L$ 能够执行 E 中的所有动作，只要该动作不属于集合 L；交替算子 $E\mid\mid E$ 表示两个进程无同步并行交叉执行；隐藏算子 E/L 将 L 中的所有动作转化为内部动作 τ，其余动作等同于进程 E；重标识算子表示若 E 能执行动作 μ，那么 $E[f]$就能执行动作 $f(\mu)$；最后若将常量进程 $Z\underline{def}E$，那么 Z 的动作完全等同于进程 E。基于标识变迁系统的安全进程代数的操作语义如图 1 所示。

算子名称	操作语义
前缀	$\dfrac{-}{\alpha.E\xrightarrow{\alpha}E}$
选择	$\dfrac{E\xrightarrow{\alpha}E'}{E+F\xrightarrow{\alpha}E'}\quad\dfrac{F\xrightarrow{\alpha}F'}{E+F\xrightarrow{\alpha}F'}$
并行	$\dfrac{E\xrightarrow{\alpha}E'}{E\mid F\xrightarrow{\alpha}E'\mid F}\quad\dfrac{F\xrightarrow{\alpha}F'}{E\mid F\xrightarrow{\alpha}E\mid F'}\quad\dfrac{E\xrightarrow{\alpha}E'F\xrightarrow{\bar{\alpha}}F'}{E\mid F\xrightarrow{\tau}E'\mid F'}$
交替并行	$\dfrac{E\xrightarrow{\alpha}E'}{E\mid\mid\mid F\xrightarrow{\alpha}E'\mid\mid\mid F}\quad\dfrac{F\xrightarrow{\alpha}F'}{E\mid\mid\mid F\xrightarrow{\alpha}E\mid\mid\mid F'}$
限制	$\dfrac{E\xrightarrow{\alpha}E'}{E\backslash L\xrightarrow{\alpha}E'\backslash L(\alpha\notin L)}$
隐藏	$\dfrac{E\xrightarrow{\alpha}E'}{E/L\xrightarrow{\alpha}E'/L}(\alpha\notin L)\quad\dfrac{E\xrightarrow{\alpha}E'}{E/L\xrightarrow{\tau}E'/L}(\alpha\in L)$

图 1 安全进程代数(部分)操作语义

2 基于迹语义和安全进程代数的非演绎信息流安全模型的形式化表示

非演绎信息流安全模型由 Sutherland 于 1986 年首次提出，所谓系统是非演绎安全的是指对系统的低级动作的观察是不能推导出任何高级输入事件的性质。换句话说，一个系统是非演绎安全的，如果系统的任意动作的低级观察与系统的任何高级输入动作是兼容的。基于迹语义非演绎信息流安全模型描述如下：

$$\forall\tau\in T(E\upharpoonright Act_L)\cdot\{\tau'\mid\xi\in T(E)\wedge\xi\upharpoonright Act_L=\tau\wedge\xi\upharpoonright Act_{HI}=\tau'\}=T(E\upharpoonright Act_{HI})\tag{1}$$

可以把上式写成下述等价的基于安全进程代数的代数式，系统满足非演绎安全模型，当且仅当

$$(\Pi_1\mid E)/Act_{HO}\approx_{may}(\Pi_2\mid E)/Act_{HO}\tag{2}$$

式中，Π_1，Π_2 任意仅含高级输入动作的进程，$\approx_{may}$表示测试等价，关于测试等价参考文献[9]。

我们可以这样理解，如果将系统的低级动作映射为对非机密数据的处理，与之对应将系统的高级动作映射为对机密数据的处理，根据式(1)(2)，非演绎安全模型强调系统的任何低级观察所产生的动作序列可由多条系统的执行路径得到，并且从系统的低级观察所产生的动作序列不能推导出系统某个特定的高级输入动作是否发生，而一般研究认为系统的不安全是系统的高级输入导致的，保护了高级输入动作的安全就相当于保证了系统的安全性。换句话说，系统的低级观察者通过观察到的低级动作是不能推导出任意一个高级输入动作是否发生，这就防止了机密信息泄密从而达到了安全。

之所以说非演绎信息流安全模型相对于访问控制安全模型来说其刻画安全更为本质，是因为访问控制安全模型如 BLP 模型是通过事先制定访问权限及相应规则来达到保证系统的安全性，其强调如何才能保证安全，而非演绎安全模型则强调的是安全是什么。

3 基于迹语义的非演绎信息流安全模型的验证算法

一般来说，任意一个使用进程代数表示系统的表达式，最终都能化为仅含前缀算子和选择算子连接的表达式[12]，所以为了简化起见，本文非演绎验证算法仅需考虑只含前缀算子和选择算子连接而成的表达式。按照上述约定，只需依次考察进程代数表达式的各个选择分支(如进程表达式 $E=l_1.l_2.$

$0+h_1.l_1.\overline{h_1}.l_2.0$，它有两个分支，对应了系统的两个可能运行的轨迹）是否满足式(1)(2)。

这里设 n 表示系统的进程代数表示的分支个数，也就是系统可能执行的迹，τ_{low} 代表各个可能执行的迹在低级动作集上限制操作所得到的结果，$\tau[i]$或者 $\tau[j]$表示系统可能执行的迹。集合 *Set*1 代表系统 E 在高级输入动作集上通过限制操作所得到的结果，集合 *Set*2 是由判断是否满足非演绎安全模型累积的高级输入动作序列组成的集合。为清晰起见，集合 *Set*1 的产生算法用流程图 2 阐述，流程图 3 为判断系统 E 是否满足非演绎安全模型的算法，具体如下。

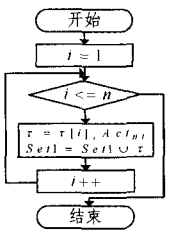

图 2　Set1 的生成算法

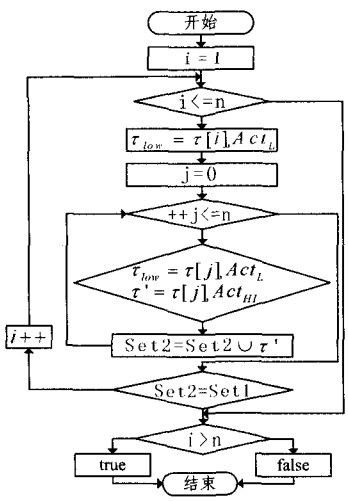

图 3　Nondeducibility 验证算法

4　安全模型检查工具 NDChecker

根据上述算法，本文基于 Java 语言开发了非演绎安全模型验证工具 NDChecker，图 4 为 NDChecker 验证工具的界面。验证工具界面由任意系统的安全进程代数表达式输入部分、结果显示以及 OK 按钮 3 部分组成，其中输出结果部分如果满足非演绎安全模型，则结果输出 true，否则输出 false，并且给出一个反例。

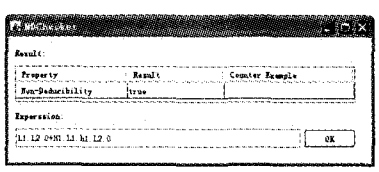

图 4　NDChecker 演示 1

为了简化输入，进程表达式输入格式规定高级动作用 H 和 h 表示，其中 H 表示高级输入动作，h 表示高级输出动作，L 表示低级动作。“+”操作算子为进程代数选择操作子，“.”操作算子为进程代数中前缀操作子。如图 4 中进程表达式 $E=l_1.l_2.0+h_1.l_1.\overline{h_1}.l_2.0$ 满足非演绎安全特性，验证结果为 true，如图 5 所示，系统 $E=l_1.l_2.0+h_1.l_1.\overline{h_1}.0$ 则不满足非演绎安全模型，输出结果为 false，且给出了个反例，反例表明观察者通过观察 L1. L2. 0 低级动，可以知道 H1 动作是否发生，从而导致了隐通道的不安全。

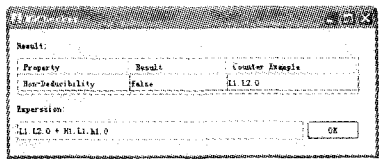

图 5　NDChecker 演示 2

结束语　基于迹语义和安全进程代数形式化描述和定义了非演绎信息流安全模型，并且基于系统的安全进程代数表达式给出了验证算法，然后基于 Java 语言实现该算法，开发了安全模型验证工具 NDChecker，最后通过实例显示其操作的正确性和易用性。

参 考 文 献

[1] 周伟，尹青，王清贤. 计算机安全中的经典模型[J]. 计算机科学，2004，31(3)：195-200

[2] Ryan P Y A, Schneider S A. Process algebra and non-interference[J]. Journal of Computer Security, 2001, 9(1/2): 75-103

[3] McCullough D. Noninterference and the composition of security properties[C]// Proc. of the IEEE Symposium on Research in Security and Privacy. 1988

[4] Tang, McMillin B. Security of information flow in the electric power grid, in Critical Infrastructure Protection[C]// Goetz E, Shenoi S, eds. Boston, Massachusetts: Springer, 2007: 43-56

[5] Akella R, Tang Han, Mcmillin B M. Analysis of information flow security in cyber-physical systems[J]. International Journal of Critical Infrastructure Protection, 2010, 3(3/4): 157-173

[6] O'hailoran C. A calculus of information flow[C]// Proc. of European Symposium on Research in Computer Security. 1990: 147-159

[7] Focardi R, Gorrieri R. A Classification of Security Properties for process algebras[J]. Journal of Computer Security, 1994/1995, 3(1): 5-33

[8] Focardi R, Gorrieri R. Classification of security properties (Part I: Information Flow)[C]// Focardi R, Gorrieri R, eds. Foundations of Security Analysis and Design-Tutoral Lectures. LNCS, Vol. 2171. Springer-verlag, 2001: 331-396

[9] 周伟，尹青，郭金庚. 计算机安全中的无干扰模型[J]. 计算机科学，2005，32(2)，159-165

[10] Akella R, Tang Han, Mcmillin B M. Analysis of information flow security in cyber-physical systems[J]. International Journal of Critical Infrastructure Protection, 2010, 3(3/4): 157-173

[11] Busi N, Gorrieri R. A Survey on Non-interference with Petri Nets[J]. Lecture Notes in Computer Science, 2004, 3098: 91-113

[12] Milner R. A Calculus of Communicating Systems[C]// LNCS 92. Springer, 1980

[13] 陈松，周从华，鞠时光，等. 基于 Petri 网的信息流安全属性的分析与验证[J]. 计算机应用研究，2010，27(12)

[14] Frau S, Gorrieri R, Ferigato C. Petri Net Security Checker: Structural Non-interference at Work[J]. Lecture Notes in Computer Science, 2009, 5491: 210-225