软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(4):912−927 [doi: 10.3724/SP.J.1001.2012.04023]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于良基语义的安全策略表达与验证方法*

包义保[1,2,3+], 殷丽华[1], 方滨兴[1], 郭 莉[1]

[1](中国科学院 计算技术研究所,北京 100190)

[2](解放军信息工程大学 电子技术学院,河南 郑州 450004)

[3](中国科学院 研究生院,北京 100049)

Approach of Security Policy Expression and Verification Based on Well-Founded Semantic

BAO Yi-Bao[1,2,3+], YIN Li-Hua[1], FANG Bin-Xing[1], GUO Li[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Institute of Electronic Technology, Information Engineering University of PLA, Zhengzhou 450004, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: baoyibao@software.ict.ac.cn

Bao YB, Yin LH, Fang BX, Guo L. Approach of security policy expression and verification based on well-founded semantic. *Journal of Software*, 2012,23(4):912−927. http://www.jos.org.cn/1000-9825/4023.htm

Abstract: This study proposes a logic-based security policy framework. First, the study proposes the security policy syntax and semantic. Next, four algoritms are proposed to transfer first-order logic based security policies into extended logic programs to evaluate queries with simple goals, to transfer complex queries into simple ones, and to verify security policies against complex security properties. Under well-founded semantics, all the algorithms are sound and completed, and their computational complexities are polynomial. In this framework, security policy declaration, evaluation and verification are executed under the same semantics, which is significant for security policy management. Furthmore, the framework can manage the security policies with advanced features, such as non-monotony and recursion, which is not supported in many existent security policy management frameworks.

Key words: security policy; security management; well-founded semantic; policy verification; logic programming

摘 要: 提出了一种基于一阶逻辑的安全策略管理框架.首先,研究安全策略的语法和语义,给出将安全策略转换成扩展型逻辑程序的算法,进而构造出安全策略基本查询算法;其次,给出将安全策略复杂查询转换成基本查询的算法,进而构造出安全策略验证算法.在良基语义下,上述算法是可终止的、可靠的和完备的,且计算复杂度都是多项式级的.该框架可以在统一的良基语义下实现安全策略表达、语义查询和验证,保证安全策略验证的有效性.此外,该框架不仅兼容现有主流的安全策略语言,还能够管理具有非单调和递归等高级特性的安全策略.

关键词: 安全策略;安全管理;良基语义;策略验证;逻辑编程

中图法分类号: TP393 **文献标识码**: A

* 基金项目: 国家自然科学基金(61070186); 国家高技术研究发展计划(863)(2009AA01Z438, 2009AA01Z43); 国家重点基础研究发展计划(973)(2007CB311100)

收稿时间: 2010-09-15; 修改时间: 2011-01-31; 定稿时间: 2011-03-21

近年来,信息安全研究领域逐渐从安全技术扩展到安全管理,基于策略的安全管理是研究的重点.在安全策略管理研究领域,如何表达安全策略、如何利用安全策略蕴涵的语义提供安全服务以及如何保障安全策略自身的安全性,这是 3 个基本而又重要的问题.由这 3 个基本问题延伸出 3 个重要的研究方向,即安全策略表达、安全策略语义查询和安全策略验证.

在安全策略表达方面,目前的主流方法是使用形式化的语言描述安全策略.文献[1]对安全策略语言进行了总结,认为基于逻辑的安全策略语言最容易被人们接受,文献[2–8]中提出的语言就是这类语言.文献[2]首次设计了一种基于逻辑的分布式安全策略语言,其语法与扩展型逻辑程序等价,但没有讨论复杂查询的计算方法,无法用于安全策略验证.文献[3,4]提出了一种基于分层逻辑程序[9]的安全策略语言,称为 FAF,它使用稳定语义[10]作为安全策略的语义.FAF 严格限制“非”的使用,不支持负递归(negative recursion),不支持量词,描述的安全策略也不够精炼,因而使用起来并不方便.由于已有的逻辑程序稳定语义查询算法的计算复杂度较高,因此 FAF 语义查询效率较低.文献[5]使用约束逻辑程序表达安全策略,约束逻辑程序的约束域能够方便地表达数值计算等特性,但其策略决策的计算复杂度较高.文献[6]基于动态逻辑提出了一种可用于表达职责和动态授权的安全策略语言,其安全策略验证问题是不可判定的.文献[7,8]仅讨论了安全策略语言的语法、语义及策略决策算法,并不支持安全策略验证这一关键特性.

虽然目前已经存在多种基于逻辑的安全策略语言,但这些语言基本上都是基于逻辑程序[9]的,局限于逻辑程序的语法,某些情况下无法容易地描述出一些安全策略.文献[11]提出了一种基于一阶逻辑的安全策略语言 Lithium.为了有限度地支持量词,Lithium 对其语法进行了严格的限制,普通管理人员难以理解,并且表达能力有限.但是,Lithium 在用一阶逻辑表达安全策略方面进行了有益的尝试.

在安全策略语义查询方面,安全策略究竟表达什么样的语义(即所谓的正则语义)以及在该语义下的查询算法是研究的重点.事实上,如何定义安全策略的正则语义是一个非常复杂的问题[12].现有的一些文献一般仅讨论具有特殊语法结构的安全策略语言,从而可以容易地定义其语义.例如,文献[1]仅讨论能够表示成 Datalog 程序[13]的安全策略,这是因为 Datalog 程序具有公认的正则语义.对于语法较为复杂的安全策略语言,其语义的研究往往被忽略了.例如,文献[2,7,11]中提出的安全策略语言要比文献[1]中提出的安全策略语言复杂得多,但却没有深入探讨安全策略语义问题.安全策略语义查询算法的设计难度取决于安全策略语言的复杂程度,一般来说,安全策略语言越复杂,语义查询算法的设计就越困难,计算复杂度也越高.因此,现有安全策略语言一般仅局限于具有特殊语法结构的逻辑系统.例如,文献[1]使用 Datalog 的语义查询算法作为安全策略的语义查询算法,其计算复杂度是多项式级的.文献[2,7,11]则没有明确地讨论语义查询问题.

在安全策略验证方面,目前已存在多种安全策略验证技术[1,14,15].这些技术的基本思想是:首先将待验证安全策略转换成抽象模型并通过形式化建模语言描述出来,同时将待验证安全属性转换成形式化的安全属性不变式,然后将它们输入到验证工具中完成验证.这需要保证不同表达方式之间转换的等价性,然而,① 有时从理论上无法实现不同表达方式之间转换的等价性;② 即使可以从理论上保证转换的等价性,但对于普通安全策略管理人员来说,保证实际转换过程的等价性也是非常困难的;③ 安全策略验证算法和安全策略语义查询算法采用的语义可能并不一致,从而有可能导致验证失效等问题的产生.

综上所述,安全策略表达、语义查询和安全策略验证之间是相互制约、相辅相成的.因此,在统一的安全策略管理框架中实现上述功能具有一定的优势,但也面临着巨大的挑战.现有安全策略语言大部分仅考虑到语义查询问题,而没有考虑到安全策略验证问题,而现有的验证技术又与安全策略语言严重脱节,因而有可能存在验证失效等问题.从公开的文献中还没有发现将上述三者统一起来进行形式化研究的成果,相关研究也不多见,一般仅关注于某些特定类型的安全策略,不具有普适性.例如,文献[16]介绍了一种在访问控制空间理论下研究系统特权安全问题的方法,指出隐式授权可能导致系统存在不必要的特权滥用风险.然而,隐式授权是现代安全策略系统中一种常用的授权方式,现有安全策略语言无一例外都支持隐式授权.文献[16]使用一种称为快速构造授权图的算法检测是否存在特权滥用,可以认为这是一种特殊类型的安全策略验证算法,但不支持负递归和非单调推理等特性.本文提出的方法可用于解决文献[16]中提出的问题,且具有更好的通用性.

基于上述考虑,本文提出了一种基于一阶逻辑的安全策略管理框架WF-SPevf,可以在统一的良基语义[17]下实现安全策略表达、语义查询和验证.一阶逻辑具有足够强的表达能力,良基语义具备一系列优良的特性(参见第 1.2 节和文献[17]),这使得 WF-SPevf 具有强大的表达能力,其所表达出的每个安全策略具有精确的语义定义方式和高效的语义查询算法及验证算法.

1 预备知识

本文假定读者熟悉一阶逻辑[18].为了便于阅读,本节简要介绍本文所使用的各种术语、符号与约定.有关逻辑程序及其良基语义的详细讨论可参考文献[17].

1.1 一阶逻辑及相关术语

本文讨论的一阶逻辑系统的字母表Σ由下列 7 种类型的符号集合构成:

① 常元符号集;

② 变元符号集;

③ 函数符号集;

④ 谓词符号集;

⑤ 逻辑连接符集$\mathcal{C}$={~,∧,∨,←,↔,↢};

⑥ 量题符号集{∀,∃};

⑦ 标点符号集.

本文所用术语,如项、原子、文字、表达式、自由变元、约束变元、闭式、Herbrand 域、Herbrand 基、Herbrand 解释、语义等的详细定义可参考文献[9,18].为了简单起见,本文约定:

① Σ为有限集;

② 所有常元符号都用首字母小写的字符串表示;

③ 所有变元符号都用首字母大写的字符串表示;

④ 函数符号集为空集;

⑤ 所有谓词符号也都用首字母小写的字符串表示;

⑥ $\vec{X},\vec{Y},\vec{Z}$ 等符号表示变元序列,$\vec{a},\vec{b},\vec{c}$ 等符号表示常元序列,$\vec{t}$ 等符号表示项序列;

⑦ 设 $X_1,\ldots,X_m$ 为一阶逻辑表达式 F 中全部的自由变元,则 $\forall_{X_1}\ldots\forall_{X_n}(F)$ 称为 F 的全称闭式,简写为 $\forall(F)$,$\exists_{X_1}\ldots\exists_{X_n}(F)$ 称为 F 的存在闭式,简写为$\exists(F)$.

定义 1(二值语义). 设 Fs 为由若干个一阶逻辑表达式构成的集合,H_B^{Fs} 为 Fs 上的 Herbrand 基,I 为 Fs 上的 Herbrand 解释(即 I 为基本原子集合且 $I\subseteq H_B^{Fs}$).如果 Fs 中的每个表达式 G 在 I 中的真值都为真,即 $I\models G$,则 I 是 Fs 的一个二值 Herbrand 语义,简称为二值语义.

1.2 扩展型逻辑程序及其语义

扩展型逻辑程序[9]是一种语法受限的一阶逻辑系统.每个逻辑程序都由一系列的规则构成,每个规则都具有 $A\leftarrow L_1\wedge\ldots\wedge L_m$ 形式的逻辑表达式,它等价于闭式$\forall(A\vee\sim L_1\vee\ldots\vee\sim L_m)$,一般写成 $A\leftarrow L_1,\ldots,L_m$ 的形式.这里,A 为原子,称为规则头部;L_i 为文字,$m\geqslant 0$,$\{L_1,\ldots,L_m\}$称为规则体部.特别地,不含变元的规则称为基本规则.

文献[17]从模型论的角度给出了通过三值逻辑定义扩展型逻辑程序良基语义的方法.不同于二值逻辑,三值逻辑认为,人们对客观世界的认识是不完全的,存在一些既无法给出肯定判断也无法给出否定判断的事实,其真值是“未定义的”.有关三值逻辑的详细描述可参考文献[17,19].

定义 2(三值解释)[19]. 设$\mathcal{L}$为一阶逻辑语言,$H_U^{\mathcal{L}}$ 为$\mathcal{L}$上的 Herbrand 域,$H_B^{\mathcal{L}}$ 为$\mathcal{L}$上的 Herbrand 基,$\mathcal{T}\subseteq H_B^{\mathcal{L}}$,$\mathcal{F}\subseteq H_B^{\mathcal{L}}$,且$\mathcal{T}\cap\mathcal{F}=\varnothing$,则序偶$\mathcal{I}=\langle\mathcal{T},\mathcal{F}\rangle$为定义在$\mathcal{L}$上的三值 Herbrand 解释,简称为三值解释.这里,$\mathcal{T}$中每个基本原子的真值为“真”,$\mathcal{F}$ 中每个基本原子的真值为“假”,$\mathcal{U}=H_B^{\mathcal{L}}-(\mathcal{T}\cup\mathcal{F})$ 中每个基本原子的真值为“未定义”.如果

$H_B^{\mathcal{L}}=\mathcal{T}\cup\mathcal{F}$,则$\mathcal{I}$蜕变成二值解释.每个二值解释$\mathcal{I}_2$都可以转化成一个三值解释$\mathcal{I}=\langle\mathcal{I}_2;H_B^{\mathcal{L}}-\mathcal{I}_2\rangle$.

三值解释$\mathcal{I}$=$\langle\mathcal{T},\mathcal{F}\rangle$可以等价地看成定义在$H_B^{\mathcal{L}}$上的全函数$I:H_B^{\mathcal{L}}\to\left\{0,\frac{1}{2},1\right\}$,如公式(1)所示:

$$I(A)=\begin{cases}1, & \text{if } A\in\mathcal{T}\\ \frac{1}{2}, & \text{if } A\in\mathcal{U}\\ 0, & \text{if } A\in\mathcal{F}\end{cases}\tag{1}$$

定义在$\mathcal{L}$上的每个逻辑表达式的真值可递归地定义为如下形式:

定义 3(逻辑表达式的真值)[19]. 设$\mathcal{I}$=$\langle\mathcal{T},\mathcal{F}\rangle$为定义在$\mathcal{L}$上的一个三值解释,则

(1) 如果 A 是基本原子,则 $I_{\mathcal{I}}(A)=I(A)$;

(2) 如果 S 是闭式,则 $I_{\mathcal{I}}(\sim S)=1-I_{\mathcal{I}}(S)$;

(3) 如果 S 和 G 都为闭式,则

$$\begin{aligned}&I_{\mathcal{I}}(S\wedge G)=\min(I_{\mathcal{I}}(S),I_{\mathcal{I}}(G)),\\ &I_{\mathcal{I}}(S\vee G)=\max(I_{\mathcal{I}}(S),I_{\mathcal{I}}(G)),\end{aligned}\quad I_{\mathcal{I}}(S\leftarrow G)=\begin{cases}1, & \text{if } I_{\mathcal{I}}(S)\geqslant I_{\mathcal{I}}(G)\\ 0, & \text{otherwise}\end{cases}.$$

(4) 对于任意的逻辑表达式 S,若 X 是 S 的一个自由变元,则

$$I_{\mathcal{I}}(\forall_X S)=\min\{I_{\mathcal{I}}(S|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\},\ I_{\mathcal{I}}(\exists_X S)=\max\{I_{\mathcal{I}}(S|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}.$$

由定义 3 可知,三值逻辑中的"←"和二值逻辑中的"←"是不同的.在二值逻辑中,$S\leftarrow G$ 和 $S\vee\sim G$ 等价,而在三值逻辑中,$I_{\mathcal{I}}(S\leftarrow G)$ 和 $\max(I_{\mathcal{I}}(S),1-I_{\mathcal{I}}(G))$ 并不恒等,因而 $S\leftarrow G$ 和 $S\vee\sim G$ 并不等价.为了平滑三值逻辑和经典二值逻辑之间的差异,本文引入一个新的逻辑连接符"↩",并且规定 $S\hookleftarrow G$ 等价于 $S\vee\sim G$.

定义 4(三值语义)[19]. 不防设扩展型逻辑程序$\mathcal{P}$中的规则都是基本规则(否则将其实例化),则三值解释$\mathcal{M}$是$\mathcal{P}$的三值语义,当且仅当对于$\mathcal{P}$中任一基本规则 $A\leftarrow L_1,\ldots,L_m$,公式(2)都成立.

$$I_M(A)\geqslant\min\{I_M(L_i):i\leqslant m\}\tag{2}$$

显然,每个二值语义都唯一对应着一个三值语义.因此,若无特殊说明,下文中的语义指的都是三值语义.由定义 4 可知,每个扩展型逻辑程序至少有一个三值语义Φ=$\langle\varnothing;\varnothing\rangle$,也可能有多个三值语义[19].研究人员普遍认为良基语义是比较合理的三值语义,具有一系列良好的特性:① 每个逻辑程序都具有唯一的良基语义;② 良基语义和一系列特殊类型逻辑程序公认的正则语义等同.例如,分层逻辑程序的良基语义等同于其完美语义[17],而完美语义被认为是分层逻辑程序的正则语义[9];③ 良基语义具备支撑性语义特性[19].所谓支撑性语义,是指该语义中任意一个基本文字都是由扩展型逻辑程序中某个相应的规则推理得到的.这一特性保证了安全策略描述安全需求的有效性;④ 存在高效的基于良基语义的语义查询算法,如 SLG 算法[20,21],具备良好的实用性.因此,本文选择良基语义作为扩展型逻辑程序的正则语义.

SLG 算法是由 Stony Brook 大学的 Chen 和 Warren 等人提出的一种高效的基于良基语义的逻辑程序语义查询算法.文献[20,21]详细讨论了 SLG 算法及其计算复杂度、可靠性和完备性等问题:SLG 算法的计算复杂度是多项式级的;对于 non-Floundering 查询,SLG 算法是可靠的和完备的;对于 Floundering 查询,SLG 算法是可靠的,但不是完备的.所谓 Floundering 查询,是指 SLG 算法在消解查询目标的过程中,如果按照某种原则选取出的子目标是负文字,且不是基本文字,则称这样的查询为 Floundering 查询.从公开的研究资料来看,几乎所有的语义查询算法都无法保证 Floundering 查询的完备性.有关 Floundering 及 non-Floundering 查询的详细讨论超出了本文的范围,但 SLG 算法在语义查询的过程中具备判定一个查询是否为 Floundering 查询的能力.如果 SLG 断定某个查询是 Floundering 查询,则报警并给出原因,然后退出语义查询过程.XSB 系统是目前被广泛认可而又有效的实现 SLG 算法的逻辑程序引擎.

2 安全策略语言

鉴于已有安全策略语言描述能力及对安全策略验证支持上的不足,本文基于一阶逻辑提出了一种支持安

全策略验证的新型安全策略语言,简记为 SPL.从语法形式上来看,SPL 兼容绝大数现有的安全策略语言,并且比它们具有更强的描述能力.

定义 5(规则). 设 *Wff* 为不含"←"逻辑连接符的一阶逻辑表达式,*A* 为原子,则称公式(3)形式的逻辑表达式为规则.公式(3)一般简写成 *A*←*Wff* 的形式.若 *Wff* 为空,则可简写成 *A*←或 *A*.此时,*A* 也称为事实.

$$\forall(A\leftarrow Wff) \tag{3}$$

定义 6(安全策略).在本文中,由有限个公式(3)形式的规则构成的集合称为安全策略.

给定安全策略$\mathcal{SP}$,它通过下列方式唯一地定义了一个与$\mathcal{SP}$对应的一阶逻辑系统 $\mathcal{L}_{\mathcal{SP}}$:① 在$\mathcal{SP}$中出现的所有常元符号构成 $\mathcal{L}_{\mathcal{SP}}$ 的常元符号集;② 在$\mathcal{SP}$中出现的所有谓词符号构成 $\mathcal{L}_{\mathcal{SP}}$ 的谓词符号集;③ 函数符号集为空集.因而, $\mathcal{L}_{\mathcal{SP}}$ 也可简记为$\mathcal{SP}$.上述类型的逻辑系统中使用的一阶语言,称为安全策略语言,简记为 SPL.

由定义 6 可知,每个扩展型逻辑程序都是一个 SPL 安全策略.由定义 6 还可以知道,SPL 兼容绝大多数已有的安全策略语言.例如,它兼容文献[2–8]中提出的安全策略语言.

安全策略语言关注两个方面的表达能力:总体表达能力和个体表达能力.前者指通过规则的不同组合最终能够达到的表达能力,一般直接称为表达能力;后者是指单个规则最终能够达到的表达能力,一般简称为描述能力.从安全策略管理的角度来看,具有相同表达能力的两种安全策略语言,描述能力越强的安全策略语言越容易受到安全管理人员的欢迎.由于安全策略语言直接面向安全管理员,因而目前的研究都非常注重安全策略语言的描述能力.

对于任意给定的安全策略$\mathcal{SP}$,设 $\mathcal{L}_{\mathcal{SP}}$ 为其对应的一阶逻辑系统, $F(\mathcal{L}_{\mathcal{SP}})$ 为 $\mathcal{L}_{\mathcal{SP}}$ 能够表达的所有一阶逻辑表达式的集合, $\mathcal{SP}(\mathcal{L}_{\mathcal{SP}})$ 为 $\mathcal{L}_{\mathcal{SP}}$ 能够表达的所有安全策略规则的集合, $\mathcal{LP}(\mathcal{L}_{\mathcal{SP}})$ 为 $\mathcal{L}_{\mathcal{SP}}$ 能够表达的所有逻辑程序规则的集合.由定义 5 可知, $\mathcal{SP}(\mathcal{L}_{\mathcal{SP}})\subseteq F(\mathcal{L}_{\mathcal{SP}})$. 同样,由第 1.2 节的内容可知, $\mathcal{LP}(\mathcal{L}_{\mathcal{SP}})\subseteq \mathcal{SP}(\mathcal{L}_{\mathcal{SP}})$. 因而,从语法的角度来看,SPL 的描述能力弱于相应的一阶逻辑语言,但强于相应的逻辑程序语言.此外,由本文定理 2 和定理 3 不难发现,SPL 的表达能力等价于扩展逻辑程序的表达能力.

3 安全策略良基语义

直接定义安全策略语义面临着巨大困难,为此,本文首先将安全策略转换成扩展型逻辑程序,然后利用扩展型逻辑程序的良基语义定义安全策略的良基语义.算法 1 通过转换运算①至转换运算⑩,将安全策略中的规则转换成扩展型逻辑程序中的规则,从而将安全策略转换成扩展型逻辑程序.

算法 1. 安全策略转换(security policy transformation).

PolicyTrans($\mathcal{SP}$)

For each rule *r* in $\mathcal{SP}$, apply one of the following transformations to *r*

① Replace $A\leftarrow W_H\wedge\sim(V\wedge W)\wedge W_T$
by $A\leftarrow W_H\wedge\sim V\wedge W_T$
and $A\leftarrow W_H\wedge\sim W\wedge W_T$

② Replace $A\leftarrow W_H\wedge(V\leftharpoondown W)\wedge W_T$
by $A\leftarrow W_H\wedge V\wedge W_T$
and $A\leftarrow W_H\wedge\sim W\wedge W_T$

③ Replace $A\leftarrow W_H\wedge\sim(V\leftharpoondown W)\wedge W_T$
by $A\leftarrow W_H\wedge\sim V\wedge W\wedge W_T$

④ Replace $A\leftarrow W_H\wedge(V\vee W)\wedge W_T$
by $A\leftarrow W_H\wedge V\wedge W_T$
and $A\leftarrow W_H\wedge W\wedge W_T$

⑤ Replace $A\leftarrow W_H\wedge\sim(V\vee W)\wedge W_T$
by $A\leftarrow W_H\wedge\sim V\wedge\sim W\wedge W_T$

⑥ Replace $A\leftarrow W_H\wedge\sim\sim W\wedge W_T$

by $A\leftarrow W_H\wedge W\wedge W_T$

⑦ Replace $A\leftarrow W_H\wedge\forall_{X_1}...\forall_{X_n}W\wedge W_T$

by $A\leftarrow W_H\wedge\sim\exists_{X_1}...\exists_{X_n}(\sim W)\wedge W_T$

⑧ Replace $A\leftarrow W_H\wedge\sim\forall_{X_1}...\forall_{X_n}W\wedge W_T$

by $A\leftarrow W_H\wedge\exists_{X_1}...\exists_{X_n}(\sim W)\wedge W_T$

⑨ Replace $A\leftarrow W_H\wedge\exists_{X_1}...\exists_{X_n}W\wedge W_T$

by $A\leftarrow W_H\wedge W|_{X_1\mapsto Z_1,...,X_n\mapsto Z_n}\wedge W_T$

where $Z_1,\ldots,Z_n$ are distinct variables that are not freely occurring in A, W_H and W_T.

⑩ Replace $A\leftarrow W_H\wedge\sim\exists_{X_1}...\exists_{X_n}W\wedge W_T$

by $A\leftarrow W_H\wedge\sim p(Y_1,\ldots,Y_k)\wedge W_T$

and $p(Y_1,...,Y_k)\leftarrow\exists_{X_1}...\exists_{X_n}W$

where p/k is a new k-arity predicate not occurring in $\mathcal{SP}$ and $Y_1,\ldots,Y_k$ are all the free variables in $\exists_{X_1}...\exists_{X_n}W$.

Until no transformation can be applied to $\mathcal{SP}$. □

End.

Note: W_H has the form $L_1\wedge\ldots\wedge L_m$ where L_i is a literal. W, V and W_T are first-order formulas.

定理 1 证明了算法 1 的可终止性和计算复杂度.这里,$|Wff|$为表达式 Wff 中的原子数目、“$\leftarrow$”连接符号数目、“$\vee$”连接符号数目、“$\exists$”连接符号数目、“$\forall$”连接符号数目及“~”连接符号数目之和,称为表达式 Wff 的尺寸.若$\mathcal{SP}$为一个安全策略,则称$|\mathcal{SP}|=\sum_{r\in\mathcal{SP}}|r|$为$\mathcal{SP}$的尺寸.

定理 1(算法 1 的可终止性). 对于任意给定的安全策略$\mathcal{SP}$,算法 1 可在有限步内执行完毕,并且其转换结果是一个扩展型逻辑程序.设$\mathcal{SP}$中规则数目为 v,“$\vee$”连接符号数目为 m,“$\forall$”连接符号数目为 n,$u=\max\{|r||r\in\mathcal{SP}\}$,$k$为使得 $u\leqslant 2^k$ 的最小值,则最坏情况下,算法 1 总的转换次数不大于 $v\cdot 2^{u-1}+m+n$;平均情况下,算法 1 总的转换次数不大于 $v\cdot(u+k\cdot 2^k-1)$,因而具有较好的实用价值.

证明:设 $A\leftarrow Wff$ 为$\mathcal{SP}$中的任一规则,算法 1 应用于 $A\leftarrow Wff$ 的转换运算为τ,同时,τ将$\mathcal{SP}$转换为$\mathcal{SP}_\tau$.

1) 若τ为转换运算①、转换运算②、转换运算④或转换运算⑩,则得到两个新规则,设为 $A_1\leftarrow Wff_1$ 和 $A_2\leftarrow Wff_2$.不难发现,$|Wff_{1\leqslant i\leqslant 2}|<|Wff|$;
2) 若τ为转换运算③、转换运算⑥或转换运算⑨,则得到一个新规则,设为$A_1\leftarrow Wff_1$.不难发现,$|Wff_1|<|Wff|$,因而$|\mathcal{SP}_\tau|<|\mathcal{SP}|$;
3) 若τ为转换运算⑧,则得到一个新规则,设为 $A_1\leftarrow Wff_1$,容易看出,$|Wff_1|=|Wff|$.但转换运算⑧后面紧跟着的转换运算一定是转换运算⑨,设其转换结果为 $A_2\leftarrow Wff_2$,容易看出,$|Wff_2|<|Wff_1|=|Wff|$;
4) 若τ为转换运算⑤,则所得结果一定为一个新规则,设为 $A_1\leftarrow Wff_1$,容易看出,$|Wff_1|=|Wff|$.若无后续转换运算可作用于$\mathcal{SP}_\tau$,则转换结束,即算法 1 可在有限次转换运算后终止;否则,在所有后续转换运算中,转换运算⑤最多出现 $m-1$ 次.这是因为,每应用一次转换运算⑤,都会使得原$\mathcal{SP}$中“$\vee$”连接符号至少减少一个,而所有其他的转换运算都不会引入“$\vee$”连接符号;
5) 若τ为转换运算⑦,则得到一个新规则,设为 $A_1\leftarrow Wff_1$,容易看出,$|Wff_1|=|Wff|+1$.但转换运算⑦后面紧跟着的转换运算一定是转换运算⑩,设其转换结果为 $A_2\leftarrow Wff_2$ 和 $A_3\leftarrow Wff_3$,并且$|Wff_2|\leqslant|Wff_1|-1=|Wff|$,$|Wff_3|\leqslant|Wff_1|-1=|Wff|$.同样,若无后续转换运算可作用于$\mathcal{SP}_\tau$,则转换结束;否则,在所有后续转换运算中,转换运算⑦最多出现 $n-1$ 次.这是因为,每应用一次转换运算⑦,都会使得原$\mathcal{SP}$中“$\forall$”连接符至少减少一个,而所有其他的转换运算都不会引入“$\forall$”连接符号.

综合上述步骤 1)~步骤 5)的分析可知,转换运算τ要么使得$|Wff_i|<|Wff|$(即转换得到的新规则的尺寸小于原规

则的尺寸),要么使得$|\mathcal{SP}_i|=|\mathcal{SP}|$.由于使得$|\mathcal{SP}_i|=|\mathcal{SP}|$的转换运算$\tau$在整个转换运算序列中最多出现 $m+n$ 次,因而对于任一安全策略$\mathcal{SP}$,算法 1 一定可在有限步转换运算后将其转换成逻辑程序$\mathcal{SP}$并终止.如果$\mathcal{SP}$不是逻辑程序,则$\mathcal{SP}$中一定存在可以继续应用转换运算的规则,因而算法 1 还没有终止,这显然是矛盾的.

在极端情况下,算法 1 选择的转换运算序列中的每个转换运算都使一个尺寸为 i 的规则分裂为尺寸为 i–1 的两个规则.因而,针对每个规则的转换运算序列最大长度不超过 $v\cdot 2^{u-1}+m+n$,转换结果$\mathcal{SP}$中规则数目最多不超过 $v\cdot u$ 个.在现实世界中,这种极端情况极其罕见,因而转换运算序列长度的平均值更具有参考价值.

在平均情况下,算法 1 针对每个规则选择的转换运算序列中的每个转换运算,使一个尺寸为 i 的规则转换成尺寸为 i–1 的另一个规则,或分裂为尺寸为$\lceil i/2\rceil$的两个规则,因而总的转换次数不超过

$$v\cdot\left(u+2\cdot\left\lceil\frac{u}{2}\right\rceil+2^2\cdot\left\lceil\frac{u}{2^2}\right\rceil+\ldots+2^k\cdot\left\lceil\frac{u}{2^k}\right\rceil\right)\leqslant v\cdot\left(u+2\cdot\frac{2^k}{2}+2^2\cdot\frac{2^k}{2^2}+\ldots+2^k\cdot\frac{2^k}{2^k}\right)=v\cdot(u+k\cdot 2^k-1).$$

这个值非常接近于$(k+1)\cdot u\cdot v$. □

由下述引理 1~引理 4 可以看出算法 1 中转换运算①~转换运算⑩的正确性.

引理 1. 下述表达式在三值逻辑中是恒等式:

$$I_{\mathcal{I}}(\sim\sim V)\equiv I_{\mathcal{I}}(V),$$
$$I_{\mathcal{I}}(\sim(V\wedge W))\equiv I_{\mathcal{I}}(\sim V\vee\sim W),$$
$$I_{\mathcal{I}}(\sim(V\vee W))\equiv I_{\mathcal{I}}(\sim V\wedge\sim W),$$
$$I_{\mathcal{I}}(\forall_{X_1}\ldots\forall_{X_n}W)\equiv I_{\mathcal{I}}(\sim\exists_{X_1}\ldots\exists_{X_n}(\sim W)),$$
$$I_{\mathcal{I}}(\sim\forall_{X_1}\ldots\forall_{X_n}W)\equiv I_{\mathcal{I}}(\exists_{X_1}\ldots\exists_{X_n}(\sim W)),$$
$$I_{\mathcal{I}}(V\leftharpoondown W)\equiv I_{\mathcal{I}}(V\vee\sim W).$$

证明:利用定义 3 即可完成证明.设$\mathcal{I}$为三值解释.

(1) $I_{\mathcal{I}}(\sim\sim V)\equiv 1-I_{\mathcal{I}}(\sim V)\equiv 1-(1-I_{\mathcal{I}}(V))\equiv I_{\mathcal{I}}(V)$;

(2) $I_{\mathcal{I}}(\sim(V\wedge W))\equiv 1-I_{\mathcal{I}}(V\wedge W)\equiv 1-\min(I_{\mathcal{I}}(V),I_{\mathcal{I}}(W))\equiv\max(1-I_{\mathcal{I}}(V),1-I_{\mathcal{I}}(W))\equiv I_{\mathcal{I}}(\sim V\vee\sim W)$;

(3) $I_{\mathcal{I}}(\sim(V\vee W))\equiv 1-I_{\mathcal{I}}(V\vee W)\equiv 1-\max(I_{\mathcal{I}}(V),I_{\mathcal{I}}(W))\equiv\min(1-I_{\mathcal{I}}(V),1-I_{\mathcal{I}}(W))\equiv I_{\mathcal{I}}(\sim V)\wedge I_{\mathcal{I}}(\sim W)$;

(4) $I_{\mathcal{I}}(\sim\exists_X(\sim W))\equiv 1-I_{\mathcal{I}}(\exists_X(\sim W))\equiv 1-\max\{I_{\mathcal{I}}(\sim W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}\equiv\min\{1-I_{\mathcal{I}}(\sim W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}$
$\equiv\min\{1-(1-I_{\mathcal{I}}(W|_{X\mapsto c})):c\in H_U^{\mathcal{L}}\}\equiv\min\{I_{\mathcal{I}}(W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}\equiv I_{\mathcal{I}}(\forall_X(W))$;

(5) $I_{\mathcal{I}}(\sim\forall_X W)\equiv 1-I_{\mathcal{I}}(\forall_X W)\equiv 1-\min\{I_{\mathcal{I}}(W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}\equiv\max\{1-I_{\mathcal{I}}(W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}$
$\equiv\max\{I_{\mathcal{I}}(\sim W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\}\equiv I_{\mathcal{I}}(\exists_X(\sim W))$.

(6) 由“$\leftharpoondown$”逻辑连接符的含义可立即得到$I_{\mathcal{I}}(V\leftharpoondown W)\equiv I_{\mathcal{I}}(V\vee\sim W)$. □

引理 2. 设 r:$A\leftarrow W_H\wedge(V\wedge W)\wedge W_T$,$r_1$:$A\leftarrow W_H\wedge V\wedge W_T$,$r_2$:$A\leftarrow W_H\wedge W\wedge W_T$,$\mathcal{I}=\langle\mathcal{I},\mathcal{F}\rangle$为三值解释.$\mathcal{I}$是 r 的语义,当且仅当$\mathcal{I}$是 r_1 和 r_2 的语义.

证明:“$\Rightarrow$”的证明:不妨设 A 为一个基本原子.

(1) 若 $A\in\mathcal{I}$,即 $I_{\mathcal{I}}(A)=1$,由 $I_{\mathcal{I}}$ 的定义可知,$I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge V\wedge W_T)$ 且 $I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge W\wedge W_T)$. 由定义 3 可知,$I_{\mathcal{I}}(r_1)=1$,$I_{\mathcal{I}}(r_2)=1$,因而$\mathcal{I}$是 r_1 和 r_2 的语义;

(2) $A\notin\mathcal{I}$且 $A\notin\mathcal{F}$,即 $I_{\mathcal{I}}(A)=\frac{1}{2}$. 由于$\mathcal{I}$是 r 的语义,则由定义 3 可知,$I_{\mathcal{I}}(W_H\wedge(V\vee W)\wedge W_T)\leqslant\frac{1}{2}$,因而 $I_{\mathcal{I}}(W_H),I_{\mathcal{I}}(W_T),I_{\mathcal{I}}(V\vee W)$ 中必有一个不大于 $\frac{1}{2}$:

a) 若 $I_{\mathcal{I}}(W_H)\leqslant\frac{1}{2}$ 或 $I_{\mathcal{I}}(W_T)\leqslant\frac{1}{2}$,则 $I_{\mathcal{I}}(W_H\wedge V\wedge W_T)\leqslant\frac{1}{2}$ 且 $I_{\mathcal{I}}(W_H\wedge W\wedge W_T)\leqslant\frac{1}{2}$;

b) 若 $I_{\mathcal{I}}(W_H)>\frac{1}{2}$ 且 $I_{\mathcal{I}}(W_T)>\frac{1}{2}$,即 $I_{\mathcal{I}}(W_H)=1$ 且 $I_{\mathcal{I}}(W_T)=1$,则

$$I_{\mathcal{I}}(V\vee W)=\max(I_{\mathcal{I}}(V),I_{\mathcal{I}}(W))\leqslant\frac{1}{2}.$$

因而 $I_{\mathcal{I}}(V)\leqslant\frac{1}{2}$ 且 $I_{\mathcal{I}}(W)\leqslant\frac{1}{2}$, 从而可得 $I_{\mathcal{I}}(W_H\wedge V\wedge W_T)\leqslant\frac{1}{2}$ 且 $I_{\mathcal{I}}(W_H\wedge W\wedge W_T)\leqslant\frac{1}{2}$;

由上述 a)和 b)可知, $I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge V\wedge W_T)$ 且 $I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge W\wedge W_T)$. 由定义 3 可知, $I_{\mathcal{I}}(r_1)=1$, $I_{\mathcal{I}}(r_2)=1$. 因而$\mathcal{I}$是 r_1 和 r_2 的语义;

(3) 若 $A\in\mathcal{F}$,即 $I_{\mathcal{I}}(A)=0$. 由于$\mathcal{I}$是 r 的模型,则由定义 3 可知, $I_{\mathcal{I}}(W_H\wedge(V\vee W)\wedge W_T)=0$, 因而 $I_{\mathcal{I}}(W_H)$, $I_{\mathcal{I}}(W_T)$, $I_{\mathcal{I}}(V\vee W)$ 中必有一个为 0:

a) 若 $I_{\mathcal{I}}(W_H)=0$ 或 $I_{\mathcal{I}}(W_T)=0$, 则 $I_{\mathcal{I}}(W_H\wedge V\wedge W_T)=0$ 且 $I_{\mathcal{I}}(W_H\wedge W\wedge W_T)=0$;

b) 若 $I_{\mathcal{I}}(W_H)\neq 0$ 且 $I_{\mathcal{I}}(W_T)\neq 0$, 则 $I_{\mathcal{I}}(V\vee W)=\max(I_{\mathcal{I}}(V),I_{\mathcal{I}}(W))=0$, 因而 $I_{\mathcal{I}}(V)=I_{\mathcal{I}}(W)=0$, 从而可得 $I_{\mathcal{I}}(W_H\wedge V\wedge W_T)=0$ 且 $I_{\mathcal{I}}(W_H\wedge W\wedge W_T)=0$;

由上述 a)和 b)可知, $I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge V\wedge W_T)$ 且 $I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge W\wedge W_T)$. 由定义 3 可知, $I_{\mathcal{I}}(r_1)=1$, $I_{\mathcal{I}}(r_2)=1$, 因而$\mathcal{I}$是 r_1 和 r_2 的语义.

"⇐"的证明与"⇒"证明的逆过程类似,这里不再赘述. □

引理 3. 设 r:$A\leftarrow W_H\wedge\exists_X W\wedge W_T$, $r_1: A\leftarrow W_H\wedge W|_{X\mapsto Z}\wedge W_T$, 变量 Z 不在规则 r 中出现,$\mathcal{I}=\langle\mathcal{T};\mathcal{F}\rangle$为三值解释.$\mathcal{I}$是 r 的语义,当且仅当$\mathcal{I}$是 r_1 的语义.

证明:$\mathcal{M}$是 r 的语义 $\Leftrightarrow I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge\exists_X W\wedge W_T)$

$$\Leftrightarrow I_{\mathcal{I}}(A)\geqslant\min(I_{\mathcal{I}}(W_H),\max\{I_{\mathcal{I}}(W|_{X\mapsto c}):c\in H_U^{\mathcal{L}}\},I_{\mathcal{I}}(W_T))$$
$$\Leftrightarrow I_{\mathcal{I}}(A)\geqslant\min(I_{\mathcal{I}}(W_H),I_{\mathcal{I}}(W|_{X\mapsto Z}),I_{\mathcal{I}}(W_T))$$
$$\Leftrightarrow I_{\mathcal{I}}(A)\geqslant I_{\mathcal{I}}(W_H\wedge W|_{X\mapsto Z}\wedge W_T)$$
$$\Leftrightarrow\mathcal{M}\text{是 }r_1\text{ 的语义.}$$ □

引理 4. 设 r:$A\leftarrow W_H\wedge\sim\exists_X W\wedge W_T$,$r_1$:$A\leftarrow W_H\wedge\sim p(Y_1,\ldots,Y_k)\wedge W_T$,$r_2$:$p(Y_1,\ldots,Y_k)\leftarrow\exists_X W$,$\mathcal{I}=\langle\mathcal{T};\mathcal{F}\rangle$为三值解释.这里, p/k 为新引入的 k-元谓词,它不在表达式 A,W_H,$\sim\exists_X W$ 及 W_T 中出现,$Y_1,\ldots,Y_k$ 为$\exists_X W$ 中的全部自由变元.那么,$\mathcal{I}$是 r 的语义,当且仅当$\mathcal{I}'=\langle\mathcal{T}';\mathcal{F}'\rangle$是 r_1 和 r_2 的语义.这里,

$$\mathcal{T}'=\mathcal{T}\cup\{p(c_1,\ldots,c_k)\mid c_1,\ldots,c_k\in H_U^{\mathcal{L}},I_{\mathcal{M}}(\exists_X W|_{Y_1\mapsto c_1,\ldots,Y_k\mapsto c_k})=1\},$$
$$\mathcal{F}'=\mathcal{F}\cup\{p(c_1,\ldots,c_k)\mid c_1,\ldots,c_k\in H_U^{\mathcal{L}},I_{\mathcal{M}}(\exists_X W|_{Y_1\mapsto c_1,\ldots,Y_k\mapsto c_k})=0\}.$$

证明:"⇒"的证明:由于$\mathcal{T}\cap\mathcal{F}=\varnothing$,显然$\mathcal{T}'\cap\mathcal{F}'=\varnothing$.容易验证 r_2 在$\mathcal{I}'$中的真值为"真",即 $I_{\mathcal{I}'}(r_2)=1$. 下面证明 $I_{\mathcal{I}'}(r_1)=1$.不妨设 A 为一个基本原子.由于 p/k 是新引入的谓词,则由$\mathcal{I}'$的定义可知:

$$I_{\mathcal{I}}(A)=I_{\mathcal{I}'}(A),I_{\mathcal{I}}(W_H)=I_{\mathcal{I}'}(W_H),I_{\mathcal{I}}(W_T)=I_{\mathcal{I}'}(W_T),I_{\mathcal{I}}(\sim\exists_X W)=I_{\mathcal{I}'}(\sim\exists_X W).$$

由于$\mathcal{I}$是 r 的语义,则

(1) 若 $A\in\mathcal{T}$,即 $I_{\mathcal{I}}(A)=I_{\mathcal{I}'}(A)=1$, 因而一定有 $I_{\mathcal{I}'}(A)\geqslant I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,\ldots,Y_k)\wedge W_T)$. 由定义 3 可知, $I_{\mathcal{I}'}(r_1)=1$, 即$\mathcal{I}'$是 r_1 和 r_2 的语义;

(2) 若 $A\notin\mathcal{I}$且 $A\notin\mathcal{F}$,即 $I_{\mathcal{I}}(A)=I_{\mathcal{I}'}(A)=\frac{1}{2}$.由于$\mathcal{I}$是 r 的模型,则由定义 3 可知, $I_{\mathcal{I}}(W_H\wedge\sim\exists_X W\wedge W_T)\leqslant\frac{1}{2}$,

因而 $I_{\mathcal{I}}(W_H)$, $I_{\mathcal{I}}(W_T)$, $I_{\mathcal{I}}(\sim\exists_X W)$ 中至少有一个小于或等于 $\frac{1}{2}$:

a) 若 $I_{\mathcal{I}}(W_H)\leqslant\frac{1}{2}$ 或 $I_{\mathcal{I}}(W_T)\leqslant\frac{1}{2}$,则一定有 $I_{\mathcal{I}'}(W_H)\leqslant\frac{1}{2}$ 或 $I_{\mathcal{I}'}(W_T)\leqslant\frac{1}{2}$,因而一定有

$$I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,\ldots,Y_k)\wedge W_T)\leqslant\frac{1}{2}.$$

b) 若 $I_{\mathcal{I}}(W_H)=1$ 且 $I_{\mathcal{I}}(W_T)=1$, 则 $I_{\mathcal{I}}(\sim\exists_X W)=I_{\mathcal{I}'}(\sim\exists_X W)\leqslant\frac{1}{2}$,即 $I_{\mathcal{I}'}(\exists_X W)=1-I_{\mathcal{I}'}(\sim\exists_X W)\geqslant\frac{1}{2}$.由

于 $I_{\mathcal{I}'}(r_2)=1$, 即 $I_{\mathcal{I}'}(p(Y_1,...,Y_k))\geqslant I_{\mathcal{I}'}(\exists_X W)\geqslant\frac{1}{2}$, 因而 $I_{\mathcal{I}'}(\sim p(Y_1,...,Y_k))=1-I_{\mathcal{I}'}(p(Y_1,...,Y_k))\leqslant\frac{1}{2}$, 即 $I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,...,Y_k)\wedge W_T)\leqslant\frac{1}{2}$.

由上述 a)和 b)可知, $I_{\mathcal{I}'}(A)\geqslant I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,...,Y_k)\wedge W_T)$. 由定义 3 可知, $I_{\mathcal{I}'}(r_1)=1$, 即$\mathcal{I}'$是 r_1 的语义;

(3) 若 $A\in\mathcal{F}$,即 $I_{\mathcal{I}}(A)=I_{\mathcal{I}'}(A)=0$. 由于$\mathcal{I}$是 r 的模型,则由定义 3 可知, $I_{\mathcal{I}}(W_H\wedge\sim\exists_X W\wedge W_T)=0$, 因而 $I_{\mathcal{I}}(W_H), I_{\mathcal{I}}(W_T), I_{\mathcal{I}}(\sim\exists_X W)$ 中必有一个为 0:

a) 若 $I_{\mathcal{I}}(W_H)=0$ 或 $I_{\mathcal{I}}(W_T)=0$, 则一定有 $I_{\mathcal{I}'}(W_H)=0$ 或 $I_{\mathcal{I}'}(W_T)=0$, 因而一定有

$$I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,...,Y_k)\wedge W_T)=0.$$

b) 若 $I_{\mathcal{I}}(W_H)\neq 0$ 且 $I_{\mathcal{I}}(W_T)\neq 0$, 则 $I_{\mathcal{I}}(\sim\exists_X W)=I_{\mathcal{I}'}(\sim\exists_X W)=0$, 即 $I_{\mathcal{I}'}(\exists_X W)=1-I_{\mathcal{I}'}(\sim\exists_X W)=1$. 由于 $I_{\mathcal{I}'}(r_2)=1$, 即 $I_{\mathcal{I}'}(p(Y_1,...,Y_k))\geqslant I_{\mathcal{I}'}(\exists_X W)=1$, 因而 $I_{\mathcal{I}'}(\sim p(Y_1,...,Y_k))=1-I_{\mathcal{I}'}(p(Y_1,...,Y_k))=0$, 即 $I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,...,Y_k)\wedge W_T)=0$.

由上述 a)和 b)可知, $I_{\mathcal{I}'}(A)=I_{\mathcal{I}'}(W_H\wedge\sim p(Y_1,...,Y_k)\wedge W_T)$. 由定义 3 可知, $I_{\mathcal{I}'}(r_1)=1$, 即$\mathcal{I}'$是 r_1 的语义.

综上所述,若$\mathcal{I}$是 r 的语义,则$\mathcal{I}'$是 r_1 和 r_2 的语义.

“⇐”的证明过程与“⇒”证明的逆过程类似,这里不再赘述. □

定理 2 说明,算法 1 在转换安全策略的过程中引入的新谓词虽然向原安全策略的语义中增加了一些新的基本原子,但这对原安全策略的语义并不构成实质性的影响,因而是可靠的.定理 3 说明,算法 1 具有保持安全策略语义的能力,因而是完备的.由引理 1~引理 4,容易证明算法 1 的可靠性和完备性.

定理 2(算法 1 的可靠性). 设$\mathcal{SP}$为一个安全策略,$\mathcal{SP}$经算法 1 转换后所得结果为$\mathcal{SP}'$,$\mathcal{M}'=\langle\mathcal{I}';\mathcal{F}'\rangle$为$\mathcal{SP}'$的一个三值语义,$\mathcal{N}$为转换过程中由转换运算⑩新引入的所有谓词构成的集合,则$\mathcal{M}=\langle\mathcal{I},\mathcal{F}\rangle$是$\mathcal{SP}$的一个三值语义.这里, $\mathcal{I}=\mathcal{I}'-\{p(\vec{t})\mid p\in\mathcal{N}, p(\vec{t})\in\mathcal{I}'\}$, $\mathcal{F}=\mathcal{F}'-\{p(\vec{t})\mid p\in\mathcal{N}, p(\vec{t})\in\mathcal{F}'\}$.

证明:由$\mathcal{M}'=\langle\mathcal{I}';\mathcal{F}'\rangle$为三值模型可得,$\mathcal{I}'\cap\mathcal{F}'=\varnothing$,容易看出 $\mathcal{I}\subseteq\mathcal{I}'$, $\mathcal{F}\subseteq\mathcal{F}'$, 因而$\mathcal{I}\cap\mathcal{F}=\varnothing$.

由引理 1~引理 4 可直接推导出,$\mathcal{SP}$中的每个规则在$\mathcal{M}'=\langle\mathcal{I}';\mathcal{F}'\rangle$中的真值都为真.

由于集合$\mathcal{I}$和$\mathcal{F}$仅是从$\mathcal{I}'$和$\mathcal{F}'$中去除了那些对$\mathcal{SP}$中规则的真值赋值不会产生影响的基本原子(这是因为这些原子所对应的谓词都是在转换过程中新引入的),因而$\mathcal{SP}$中的每个规则在$\mathcal{M}$中的真值也都为真.

综上所述,$\mathcal{M}=\langle\mathcal{I},\mathcal{F}\rangle$是$\mathcal{SP}$的一个三值语义. □

定理 3(算法 1 的完备性). 设$\mathcal{SP}$为安全策略,$\mathcal{M}=\langle\mathcal{I},\mathcal{F}\rangle$为$\mathcal{SP}$的三值语义,$\mathcal{SP}$经算法 1 转换后所得结果$\mathcal{SP}'$,则存在$\mathcal{M}'=\langle\mathcal{I}';\mathcal{F}'\rangle$,其中, $\mathcal{I}\subseteq\mathcal{I}'\subseteq H_B^{\mathcal{SP}'}$, $\mathcal{F}\subseteq\mathcal{F}'\subseteq H_B^{\mathcal{SP}'}$, $\mathcal{M}'$是$\mathcal{SP}'$的三值语义.

证明:由定理 1 可知,算法 1 可在有限步内转换完毕.设算法 1 应用于$\mathcal{SP}$总的转换次数为 m,且这些转换运算构成的序列依次为 $\tau_1,...,\tau_m$.设$\mathcal{SP}$经过前 $i-1$ 次转换后所得到的安全策略记为$\mathcal{SP}_{i-1}$,令$\mathcal{M}_0=\mathcal{M}$,$\mathcal{M}_i=\langle\mathcal{I}_i;\mathcal{F}_i\rangle$构造方法如下所示:

1) 若τ_i为转换运算①~转换运算⑨,令$\mathcal{M}_i=\mathcal{M}_{i-1}$.由引理 1~引理 3 可知,若$\mathcal{M}_{i-1}$是$\mathcal{SP}_{i-1}$的语义,则$\mathcal{M}_i$是$\mathcal{SP}_i$的语义;

2) 若τ_i为转换运算⑩,则被转换的规则为 $A\leftarrow W_H\wedge\sim\exists_X W\wedge W_T, p_i/k$ 为新引入的 k-元谓词,$Y_1,\ldots,Y_k$为$\exists_X W$中的全部自由变元.令$\mathcal{M}_i=\langle\mathcal{I}_i;\mathcal{F}_i\rangle$,其中,

$$\mathcal{I}_i=\mathcal{I}_{i-1}\cup\{p_i(c_1,...,c_k)\mid c_1,...,c_k\in H_U^{\mathcal{L}}, I_{\mathcal{M}}(\exists_X W|_{Y_1\mapsto c_1,...,Y_k\mapsto c_k})=1\},$$

$$\mathcal{F}_i=\mathcal{F}_{i-1}\cup\{p_i(c_1,...,c_k)\mid c_1,...,c_k\in H_U^{\mathcal{L}}, I_{\mathcal{M}}(\exists_X W|_{Y_1\mapsto c_1,...,Y_k\mapsto c_k})=0\}.$$

由引理 4 可知,若$\mathcal{M}_{i-1}$是$\mathcal{SP}_{i-1}$的语义,则$\mathcal{M}_i$是$\mathcal{SP}_i$的语义. □

定理 2 和定理 3 保证了通过扩展型逻辑程序的良基语义定义安全策略语义的合理性.

定义 7(安全策略的良基语义). 设$\mathcal{SP}$为安全策略,$\mathcal{SP}$经算法 1 转换后所得结果为$\mathcal{SP}'$,$\mathcal{M}'=\langle\mathcal{I}';\mathcal{F}'\rangle$为$\mathcal{SP}'$的良基语义,$\mathcal{N}$为转换过程中由转换运算⑩新引入的所有谓词构成的集合,则称$\mathcal{M}=\langle\mathcal{I},\mathcal{F}\rangle$是$\mathcal{SP}$的良基语义.这里,

$$\mathcal{T}=\mathcal{T}'-\{p(\vec{t})\mid p\in\mathcal{N},p(\vec{t})\in\mathcal{T}'\},\mathcal{F}=\mathcal{F}'-\{p(\vec{t})\mid p\in\mathcal{N},p(\vec{t})\in\mathcal{F}'\}.$$

4 安全策略语义查询与验证

本节讨论基于良基语义的安全策略语义查询问题,进而构造出安全策略决策算法和验证算法.首先讨论查询目标为 $\leftarrow p(\vec{t})$ 形式的表达式(即基本查询目标)时安全策略语义查询问题,并给出相应的算法,该算法可应用于安全策略决策等方面;其次,讨论查询目标为一阶逻辑表达式(即复杂查询目标)时安全策略语义查询问题,并给出相应的算法,该算法可应用于安全策略验证等方面.

4.1 安全策略基本查询及安全策略决策

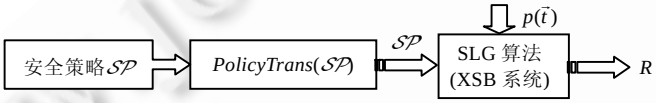
设$\mathcal{SP}$为安全策略,$\mathcal{M}$为$\mathcal{SP}$的良基语义,p/n 为$\mathcal{SP}$中的一个谓词, $p(\vec{t})$ 为 n 元原子.那么存在哪些 n 元基本置换θ可以使得 $p(\vec{t})\theta$ 为$\mathcal{SP}$关于$\mathcal{M}$的逻辑结果呢?这个问题称为安全策略基本查询问题.本节基于算法 1 给出了一种安全策略基本查询问题的求解方法,其基本思想如算法 2 所示.这里,R 为 SLG 算法返回的查询结果.SLG 算法和 XSB 系统的相关信息可参考文献[20,21].

(1) 如果 $R=\varnothing$,则说明不存在基本置换使得原子 $p(\vec{t})$ 在$\mathcal{SP}$的良基语义中成立.由于本文认为良基语义是扩展型逻辑程序的正则语义,因而可以断言,$\exists(p(\vec{t}))$ 在$\mathcal{SP}$中是不成立的;

(2) 如果 $R\neq\varnothing$,则说明存在基本置换使得原子 $p(\vec{t})$ 在$\mathcal{SP}$的良基语义中成立,因而可以断言,$\exists(p(\vec{t}))$ 在$\mathcal{SP}$中是成立的;

(3) 如果 XSB 系统返回 Floundering 警告,则无法判定 $\exists(p(\vec{t}))$ 在$\mathcal{SP}$中是否成立.

算法 2. 安全策略语义基本查询算法.



定理 4(安全策略基本查询算法的可靠性和完备性). $p(\vec{t})\theta$ 是$\mathcal{SP}$相对于$\mathcal{M}$的逻辑结果,当且仅当 $p(\vec{t})\theta$ 是$\mathcal{SP}'$相对于其良基语义$\mathcal{M}'$的逻辑结果.

证明:由算法 1 的可靠性(定理 2)和完备性(定理 3)可直接得证. □

安全策略语义基本查询算法经过简单变换后可处理 $\leftarrow p(\vec{t})$ 形式的查询目标,主要用于策略决策等方面.

4.2 安全策略复杂查询及安全策略验证

安全属性不变式通常是比较复杂的一阶逻辑表达式,安全策略语义基本查询算法只能处理形式较为简单的(仅含有一个原子的)查询目标,因而无法满足安全策略验证的需求.为了解决安全策略验证问题,本节首先给出了将安全策略复杂查询问题转化为安全策略简单查询问题的算法(参见算法 3),并且证明了该算法的可靠性和完备性;其次,利用安全策略语义基本查询算法构造出安全策略复杂查询算法,该算法主要用于安全策略验证,因而本文称其为安全策略验证算法.由于安全策略验证算法是基于安全策略语义基本查询算法的,因而安全策略验证和安全策略决策使用相同的语义,可有效避免本文引言中提到的验证失效等问题的产生.

算法 3. 安全策略复杂查询转换.

QueryTrans(⟨$\mathcal{SP}$,Q⟩)

  $\mathcal{SP}'=\mathcal{SP}\cup\{ans(X_1,\ldots,X_n)\leftarrow Q\}$;

  Return ⟨*PolicyTrans*($\mathcal{SP}'$),*ans*(X_1,…,X_n)⟩

End.

设$\mathcal{SP}$为安全策略,$\mathcal{M}$为$\mathcal{SP}$的良基语义,Q 为不含"←"的任意形式的一阶逻辑表达式,X_1,…,X_n 为 Q 中的全部自由变元,那么存在哪些 n 元基本置换可以使得 Q 为$\mathcal{SP}$关于$\mathcal{M}$的逻辑结果呢?这个问题称为安全策略复杂查询问题,记为⟨$\mathcal{SP}$,Q⟩.设 *ans*/n 为一个未在$\mathcal{SP}$中出现过的 n 元谓词符号,则算法 3 可以将⟨$\mathcal{SP}$,Q⟩转换成另一个等价的安全策略基本查询问题⟨$\mathcal{SP}''$,*ans*(X_1,…,X_n)⟩.这里,$\mathcal{SP}''$为扩展型逻辑程序.定理 5 证明了算法 3 的可靠性和完

备性.定理 6 证明了算法 3 的可终止性.

定理 5(算法 3 的可靠性和完备性). 在算法 3 中,设$\mathcal{M}$=〈$\mathcal{T}$,$\mathcal{F}$〉是$\mathcal{SP}$的良基语义,令

$$\mathcal{T}'=\mathcal{T}\cup\{ans(c_1,...,c_n)\mid c_1,...,c_n\in H_U^{\mathcal{SP}},I_{\mathcal{T}}(q|_{X_1\mapsto c_1,...,X_n\mapsto c_n})=1\},$$

$$\mathcal{F}'=\mathcal{F}\cup\{ans(c_1,...,c_n)\mid c_1,...,c_n\in H_U^{\mathcal{SP}},I_{\mathcal{T}}(q|_{X_1\mapsto c_1,...,X_n\mapsto c_n})=0\},$$

则有:

(1) $\mathcal{M}'$=〈$\mathcal{T}'$;$\mathcal{F}'$〉是扩展型逻辑程序$\mathcal{SP}'$的良基语义;

(2) $I_{\mathcal{M}}(\exists(Q))=1$ 当且仅当 $I_{\mathcal{M}'}(\exists(ans(X_1,...,X_n)))=1$.

证明:(1) 由文献[17]的定义 3.3 及$\mathcal{T}'$和$\mathcal{F}'$的构造方法可直接得出$\mathcal{M}'$是$\mathcal{SP}'$的良基语义,即上面的(1)成立.

(2) "⇒"的证明:由于$\mathcal{M}'$是$\mathcal{SP}'$的语义,则 $I_{\mathcal{M}'}(ans(X_1,...,X_n)\leftarrow Q)=1$. 若 $I_{\mathcal{M}}(\exists(Q))=1$, 易知 $I_{\mathcal{M}'}(\exists(Q))=1$, 即存在基本置换$\theta$={$X_1\mapsto a_1,...,X_n\mapsto a_n$}使得 $I_{\mathcal{M}'}(Q|_{X_1\mapsto a_1,...,X_n\mapsto a_n})=1$. 由定义 3 可知:

$$I_{\mathcal{M}'}(ans(a_1,...,a_n))\geqslant I_{\mathcal{M}'}(Q|_{X_1\mapsto a_1,...,X_n\mapsto a_n}).$$

即 $I_{\mathcal{M}'}(ans(a_1,...,a_n))\equiv I_{\mathcal{M}'}(\exists(ans(X_1,...,X_n)))=1$.

反之,若 $I_{\mathcal{M}'}(\exists(ans(X_1,...,X_n)))=1$, 不妨设 $I_{\mathcal{M}'}(ans(a_1,...,a_n))=1$. 这里, $a_1,...,a_n\in H_U^{\mathcal{SP}}$. 由于 $ans(a_1,\ldots,a_n)$仅能由规则 $ans(X_1,\ldots,X_n)\leftarrow Q$ 推导出来,并且$\mathcal{M}'$是$\mathcal{SP}'$的良基语义,因而 $I_{\mathcal{M}'}(Q|_{X_1\mapsto a_1,...,X_n\mapsto a_n})=1$. 由$\mathcal{M}'$=〈$\mathcal{T}'$;$\mathcal{F}'$〉构造方法可知, $I_{\mathcal{M}}(Q|_{X_1\mapsto a_1,...,X_n\mapsto a_n})=1$, 即 $I_{\mathcal{M}}(\exists(Q))=1$.

"⇐"的证明与"⇒"证明的逆过程类似,这里不再赘述. □

定理 6(算法 3 的可终止性). 算法 3 可在有限步内转换完毕,其总的转换次数是多项式级的.

证明:该结论是定理 1 的直接推论. □

至此,本文给出了将安全策略转换成扩展型逻辑程序及将安全策略查询问题转换成扩展型逻辑程序查询问题的算法,并且证明了这些算法在良基语义下的可终止性、可靠性和完备性.下面将利用这些结论构建安全策略验证算法.为了使验证算法具有足够高的效率,本文假设:

① 安全策略中每个规则 $A\leftarrow Wff$ 的 Wff 中不含有任何函数符号;

② 待验证的安全属性都可以表示成一个不含逻辑连接符"←"和函数符号的一阶逻辑表达式;

③ 为了保证安全策略验证过程的可靠性和完备性,每个待验证的安全属性都是一个 non-Floundering 查询.

对于那些无法表示成 non-Floundering 查询的安全属性,则不在本文考虑的范围之内.

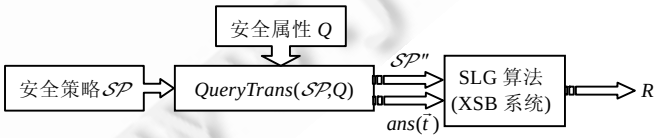
安全策略验证算法的结构如算法 4 所示.在该算法中,首先利用算法 3,将安全策略$\mathcal{SP}$和待验证的安全属性不变式 Q 构成的安全策略复杂查询问题〈$\mathcal{SP}$,Q〉转换成安全策略简单查询问题〈$\mathcal{SP}''$,$ans(X_1,\ldots,X_n)$〉.由于$\mathcal{SP}''$是扩展型逻辑程序,$ans(X_1,\ldots,X_n)$是基本查询目标,因而 XSB 系统可高效地计算它并返回查询结果 R:

(1) 如果 $R=\varnothing$,则说明不存在基本置换使得安全属性不变式 Q 在$\mathcal{SP}$的良基语义中成立.由于本文认为良基语义是扩展型逻辑程序的正则语义,因而可以断言,$\exists(Q)$在$\mathcal{SP}$中是不成立的;

(2) 如果 $R\neq\varnothing$,则说明存在基本置换使得 Q 在$\mathcal{SP}$的良基语义中成立,因而$\exists(Q)$在$\mathcal{SP}$中成立;

(3) 如果 XSB 系统返回 Floundering 警告,则说明无法判定$\exists(Q)$在$\mathcal{SP}$中是否成立.

算法 4. 安全策略验证算法.



定理 7(安全策略查询算法的可靠性和完备性). 设$\mathcal{M}$是$\mathcal{SP}$的良基语义,则$\exists(Q)$是$\mathcal{SP}$相对于$\mathcal{M}$的逻辑结果,当且仅当$\exists(ans(X_1,\ldots,X_n))$是$\mathcal{SP}''$相对于其良基语义$\mathcal{M}''$的逻辑结果.

证明:该结论是定理 5 的直接推论. □

至此,我们已经完成了安全策略语言的设计、安全策略语义的定义及安全策略语义查询算法和验证算法的

设计.接下来,我们将这些研究成果组合起来,形成一个切实可用的安全策略管理框架 WF-SPevf.

5 WF-SPevf 安全策略管理框架

WF-SPevf 的整体架构如图 1 所示,可在统一的安全策略良基语义下实现以下主要功能:

- 安全策略表达:安全管理员根据系统安全需求,通过安全策略语言 SPL 制定安全策略,并存储到策略库中;
- 安全策略查询:信息系统根据具体应用场景形成基本查询目标提交给 WF-SPevf.WF-SPevf 根据安全策略及查询目标,通过安全策略语义基本查询算法计算出查询结果并反馈给信息系统;
- 安全属性表达:安全专家根据系统特性提炼出系统应具备的安全属性,然后通过安全策略语言 SPL 表达出来,形成安全属性不变式.值得注意的是,WF-SPevf 使用相同的语言表达安全属性和安全策略,这是 WF-Spevf 与其他安全策略验证工具的一个显著不同点;
- 安全策略验证:WF-SPevf 根据安全策略和安全属性不变式,自动地验证两者之间的符合性.如果安全策略满足安全属性不变式,则验证通过;否则,安全策略中存在错误或不足.此时,WF-SPevf 提供完整的出错原因,以便于安全管理员调试安全策略.值得注意的是,安全策略验证和安全策略查询使用相同的语义,这是 WF-Spevf 与其他验证技术最重要的区别.由于安全属性不变式本质上是一阶逻辑表达式,因此 WF-SPevf 还可用于其他能够表达成一阶逻辑表达式的属性判定问题,如策略冲突检测等.

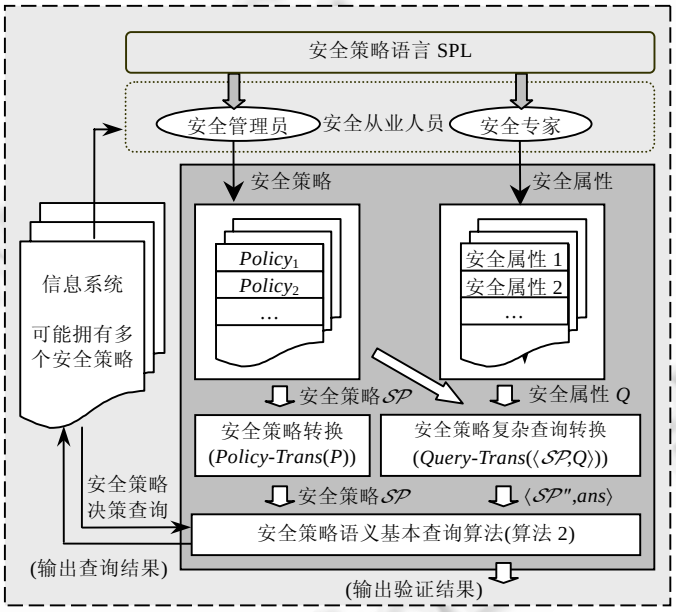


Fig.1 Security policy framework of WF-SPevf

图 1 WF-SPevf 的安全策略框架

在 WF-SPevf 中,安全策略语言、安全策略语义查询算法和安全策略验证算法构成一个有机的整体.它们在统一的语义下对外提供策略表达、策略决策、策略验证等服务.WF-SPevf 具备强大的描述能力、高效的策略决策算法及强健的策略验证能力.与现有的安全策略框架相比,具有鲜明的特色.因此有理由相信,WF-SPevf 可为管理人员提供快捷、方便和易用的安全策略管理服务.

下面通过实例说明 WF-SPevf 的使用方法.设信息系统中存在由下列规则构成的信息流安全策略[22]:

规则 1. 对于任意给定的用户 U 和文件 F(每个文件至少具有一个祖先目录),如果 F 的安全级别不支配 U 的安全级别,同时至少有一个管理员允许 U 读取 F 的所有祖先目录,且没有管理员禁止 U 读取 F

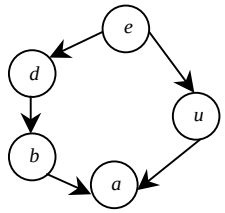
的任一祖先目录,则允许 U 读取 F.

规则 2. 对于任意给定的用户 U 和文件 F,如果 U 的安全级别不支配 F 的安全级别,则允许 U 写 F.

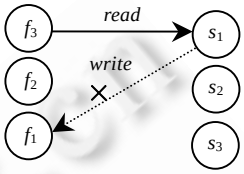
规则 3. 系统中已有的实体(用户或文件)及其安全级别分别为$(f_1,b),(f_2,u),(f_3,d),(s_1,d),(s_2,u),(s_3,a)$,其中,$f_i$ 为文件,s_j 为用户.

规则 4. 安全级别$\mathcal{L}=\{a,b,u,d,e\}$之间的支配关系如图 2(a)所示,它们构成格.

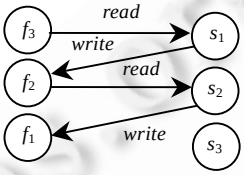
规则 5. 文件夹 t 为文件 $f_{1\leqslant i\leqslant 3}$ 的祖先,且管理员 adm_1 和 adm_2 授予用户 s_1 和 s_2 读取文件夹 t 的权限.



(a) $\mathcal{L}$在谓词$\prec$上构成格(箭头表示支配关系)



(b) $\mathcal{P}$不允许出现该信息流(箭头表示信息流向)



(c) 在$\mathcal{P}$中查找到一条不安全信息流路径(箭头表示信息流向)

Fig.2 An insecurity information flow in security policy $\mathcal{P}$

图 2 安全策略$\mathcal{P}$中安全级的构成及不安全的信息流路径

(1) 用 SPL 表示安全策略

文献[2−8]中提出的安全策略语言不支持量词,无法简单地表示上述安全策略.然而,SPL 可简单地将上述安全策略表示成下列形式:

$$\mathcal{P}=\left\{\begin{array}{l} R_1:\ permit(U,F,read)\leftarrow secLevel(U,L_U)\wedge secLevel(F,L_F)\wedge\sim L_U\prec L_F\wedge\exists_O ancestor(O,F)\wedge \\ \qquad\qquad \exists_Z\forall_O[(authoz(U,O,read,Z)\wedge\sim\exists_{Z'}deny(U,O,read,Z'))\longleftarrow ancestor(O,F)] \\ R_2:\ permit(U,F,write)\leftarrow secLevel(U,L_U),secLevel(F,L_F),\sim L_F\prec L_U \\ \left[\begin{array}{l} C_1:\ secLevel(f_1,b) \\ C_2:\ secLevel(f_2,u) \\ C_3:\ secLevel(f_3,d) \\ C_4:\ secLevel(s_1,d) \\ C_5:\ secLevel(s_2,u) \\ C_6:\ secLevel(s_3,a) \end{array}\right] \quad \left[\begin{array}{l} D_1:\ a\prec b \\ D_2:\ a\prec u \\ D_3:\ b\prec d \\ D_4:\ d\prec e \\ D_5:\ u\prec e \\ D_6:\ X\prec Z\leftarrow X\prec Y,Y\prec Z \end{array}\right] \quad \left[\begin{array}{l} E_1:\ ancestor(t,f_1) \\ E_2:\ ancestor(t,f_2) \\ E_3:\ ancestor(t,f_3) \\ E_4:\ authoz(s_1,t,read,adm_1) \\ E_5:\ authoz(s_2,t,read,adm_2) \\ E_6:\ deny(s_3,t,read,adm_3) \end{array}\right] \end{array}\right\},$$

其中,R_1 对应规则 1,R_2 对应规则 2,C_1~C_6 对应规则 3,D_1~D_6 对应规则 4,E_1~E_6 对应规则 5.*permit*/3,*ancestor*/2,*authen*/2,*authoz*/4,*deny*/4,*secLevel*/2 和$\prec$/2 都为谓词符号,表达的含义是不言自明的.

(2) 安全策略决策

判断$\mathcal{P}$是否允许 s_1 读取 f_1 的过程如下(参见算法 2):首先,利用算法 1 将$\mathcal{P}$转换成扩展型逻辑程序$\mathcal{P}'$;然后,利用 XSB 系统评估查询$\langle\mathcal{P}',permit(s_1,f_1,read)\rangle$,即可获得最终的判决结果.

利用算法 1 转换$\mathcal{P}$中 R_1 的过程如下所示:连续 3 次应用转换运算⑨后,R_1 可被转换成公式(4)所示的规则:

$$\begin{array}{l} permit(U,F,read)\leftarrow secLevel(U,L_U)\wedge secLevel(F,L_F)\wedge\sim L_U\prec L_F\wedge ancestor(O,F)\wedge \\ \qquad\qquad \forall_O[(authoz(U,O,read,Z)\wedge\sim\exists_{Z'}deny(U,O,read,Z'))\longleftarrow ancestor(O,F)] \end{array} \tag{4}$$

应用转换运算⑦,公式(4)所示的规则可转换成公式(5)所示的规则:

$$permit(U,F,read) \leftarrow secLevel(U,L_U) \wedge secLevel(F,L_F) \wedge \sim L_U \prec L_F \wedge ancestor(O,F) \wedge \sim \exists_O[\sim((authoz(U,O,read,Z) \wedge \sim \exists_{Z'} deny(U,O,read,Z')) \leftarrowtail ancestor(O,F))] \quad (5)$$

应用转换运算⑩,公式(5)所示的规则可转换成公式(6)和公式(7)所示的两个规则:

$$T_1: permit(U,F,read) \leftarrow SecLevel(U,L_U) \wedge SecLevel(U,L_U) \wedge \sim L_U \prec L_F \wedge ancestor(O,F) \wedge \sim p(U,Z) \quad (6)$$

$$p(U,Z) \leftarrow \exists_O[\sim((authoz(U,O,read,Z) \wedge \exists_{Z'} deny(U,O,read,Z')) \leftarrowtail ancestor(O,F))] \quad (7)$$

已经没有转换运算可应用于公式(6).应用转换运算⑨,公式(7)所示的规则可转换为公式(8)所示的规则:

$$p(U,Z) \leftarrow \sim((authoz(U,O,read,Z) \wedge \exists_{Z'} deny(U,O,read,Z')) \leftarrowtail ancestor(O,F)) \quad (8)$$

应用转换运算③,公式(8)所示的规则可转换成公式(9)所示的规则:

$$p(U,Z) \leftarrow \sim(authoz(U,O,read,Z) \wedge \exists_{Z'} deny(U,O,read,Z')) \wedge ancestor(O,F) \quad (9)$$

应用转换运算①,公式(9)所示规则可转换成公式(10)和公式(11)所示的两个规则:

$$T_2: p(U,Z) \leftarrow \sim authoz(U,O,read,Z) \wedge ancestor(O,F) \quad (10)$$

$$p(U,Z) \leftarrow \exists_{Z'} deny(U,O,read,Z') \wedge ancestor(O,F) \quad (11)$$

已经没有转换运算可应用于公式(10).应用转换运算⑨,公式(11)所示规则可转换成公式(12)所示的规则:

$$T_3: p(U,Z) \leftarrow deny(U,O,read,Z') \wedge ancestor(O,F) \quad (12)$$

已经没有转换运算可应用于公式(12),转换过程结束,因而 R_1 被转换成 3 个扩展型逻辑程序规则$\{T_1,T_2,T_3\}$.最终,算法 1 将$\mathcal{P}$转换成扩展型逻辑程序$\mathcal{P}'=(\mathcal{P}-\{R_1\})\cup\{T_1,T_2,T_3\}$.

将$\mathcal{P}$转换成$\mathcal{P}'$后,利用 XSB 逻辑程序查询引擎评估查询$\langle\mathcal{P}',permit(s_1,f_1,read)\rangle$,查询结果集为空集,具体的查询评估过程参见文献[20,21].这说明,$\mathcal{P}$不允许 s_1 直接读取 f_1,如图 2(b)所示.

(3) 安全属性验证

$\mathcal{P}$是一个信息流安全策略.从信息流的角度来看,为了保护机密性,高安全级别客体中的信息应无法流向低安全级别的客体,即$\mathcal{P}$应该能够满足 SPL 安全属性不变式 Q:

$$Q = \forall_{O_1,L_1,O_2,L_2}(\sim(canFlowTo(O_1,O_2) \wedge secLevel(O_1,L_1) \wedge secLevel(O_2,L_2) \wedge L_2 \prec L_1)).$$

这里,安全属性不变式 Q 表示"高安全级别客体中的信息应无法流向低安全级别的客体".为了判定$\mathcal{P}$是否满足 Q,需要向原安全策略$\mathcal{P}$中加入下列两个用于刻画信息流向的规则:

$$V_1: canFlowTo(O_1,O_2) \leftarrow permit(S,O_1,read), permit(S,O_2,write),$$

$$V_2: canFlowTo(O_1,O_3) \leftarrow canFlowTo(O_1,O_2), canFlowTo(O_2,O_3),$$

其中,规则 V_1 表示,对于给定的用户 S、客体 O_1 和 O_2,如果 S 能够读取 O_1,同时 S 能够向 O_2 中写入信息,则 O_1 中的信息能够流向 O_2;规则 V_2 表示信息流向具有传递性.显然,将 V_1 和 V_2 加入$\mathcal{P}$中不会改变$\mathcal{P}$原有的安全属性.

令$\mathcal{P}''=\mathcal{P}\cup\{V_1,V_2\}$,验证$\mathcal{P}$是否满足 Q 的过程如下所示(参见算法 4):

首先,利用算法 3 将复杂查询$\langle\mathcal{P}'',Q\rangle$转换成基本查询$\langle\mathcal{P}_{flow},tmp\rangle$.由算法 3 和算法 1 容易得到扩展型逻辑程序$\mathcal{P}_{flow}$,其中,$tmp$ 和 p 是在转换$\langle\mathcal{P}'',Q\rangle$的过程中新引入的 0 元谓词;

$$\mathcal{P}_{flow} = PolicyTrans(\mathcal{P}'' \cup \{tmp \leftarrow Q\})$$

$$= \mathcal{P}' \cup \{V_1,V_2\} \cup \left\{ \begin{array}{l} tmp \leftarrow \sim p \\ p \leftarrow canFlowTo(O_1,O_2) \wedge secLevel(O_1,L_1) \wedge secLevel(O_2,L_2) \wedge L_2 \prec L_1 \end{array} \right\}.$$

然后,利用算法 2 评估查询$\langle\mathcal{P}_{flow},tmp\rangle$.如果$\mathcal{P}''$能够满足 Q,则语义查询$\langle\mathcal{P}_{flow},tmp\rangle$的结果集应该为空集.然而,算法 2 却能够查询到一条规则序列(当 XSB 系统工作于调试模式时,能够跟踪查询过程中所调用的规则序列).相应地,由这条规则序列可得到一条信息流,如图 2(c)所示.在这条信息流中,信息能够从安全级别为 d 的客体 f_3 不安全地流向安全级别为 b 的客体 f_1.由于安全级别 d 支配安全级别 b,因而$\mathcal{P}$存在安全隐患.

由上例可知,WF-SPevf 可以高效地验证安全策略是否满足某个查询,进而判定是否满足某个安全属性.

6 结 论

本文首先定义了一种基于一阶逻辑的安全策略语言 SPL.它不仅兼容现有主流的安全策略语言,还可以表达具有非单调和递归等高级特性的复杂安全策略;其次,本文通过安全策略转换算法(算法 1)将 SPL 安全策略转换成扩展型逻辑程序,进而利用扩展型逻辑程序的良基语义给出了 SPL 安全策略的语义;再次,本文利用逻辑程序理论构造出安全策略语义基本查询算法(算法 2),可用于处理简单查询目标,主要应用于安全策略决策.在此基础上,本文给出了将安全策略语义复杂查询问题转换成安全策略语义基本查询问题的算法(算法 3),进而构造出安全策略验证算法(算法 4).显然,安全策略验证算法和安全策略决策算法基于相同的语义,有效地保证了安全策略验证的有效性;最后,本文将上述研究成果组合起来,形成安全策略管理框架 WF-SPevf.总体来说,WF-SPevf 可在统一的良基语义基础上实现安全策略表达、语义查询和安全策略验证,具有强大的策略描述能力、高效的策略决策能力以及可靠的策略验证能力,可应用于各种类型的策略管理系统中.

References:

[1] Bonatti PA, Shahmehri N, Duma C, Olmedilla D, Nejdl W, Baldoni M, Baroglio C, Martelli A, Patti V, Coraggio P, Antoniou G, Peer J, Fuchs NE. Rule-Based policy specification: State of the art and future work. Technical Report, IST506779, Project Deliverable D1, Working Group I2. 2004. http://rewerse.net/deliverables/i2-d1.pdf

[2] Woo TYC, Lam SS. Authorization in distributed systems: A new approach. Journal of Computer Security, 1993,2(2-3):107−136.

[3] Jajodia S, Samarati P, Sapino ML, Subrahmanian VS. Flexible support for multiple access control policies. ACM Trans. on Database System, 2001,26(2):214−260. [doi: 10.1145/383891.383894]

[4] Jajodia S, Samarati P, Subrahmanian VS, Bertino E. A unified framework for enforcing multiple access control policies. In: Peckman JM, Franklin M, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'97). New York: ACM, 1997. 474−485. [doi: 10.1145/253260.253364]

[5] Barker S, Stuckey PJ. Flexible access control policy specification with constraint logic programming. ACM Trans. on Information System Security, 2003,6(4):501−546. [doi: 10.1145/950191.950194]

[6] Gelfond M, Lobo J. Authorization and obligation policies in dynamic systems. In: De La Banda MG, Pontelli E, eds. Proc. of the 24th Int'l Conf. on Logic Programming. LNCS 5366, Berlin, Heidelberg: Springer-Verlag, 2008. 22−36. [doi: 10.1007/978-3-540-89982-2_7]

[7] Becker MY, Fournet C, Gordon AD. Design and semantics of a decentralized authorization language. In: Proc. of the 20th IEEE Computer Security Foundations Symp. (CSF 2007). Washington: IEEE Computer Society, 2007. http://dl.acm.org/citation.cfm?id=1270639 [doi: 10.1109/CSF.2007.18]

[8] Gurevich Y, Neeman I. DKAL: Distributed-Knowledge authorization language. In: Proc. of the 21st IEEE Computer Security Foundations Symp. Washington: IEEE Computer Society, 2008. 149−162. http://dl.acm.org/citation.cfm?id=1381251 [doi: 10.1109/CSF.2008.8]

[9] Lloyd JW. Foundations of Logic Programming. 2nd ed., New York: Springer-Verlag, 1993.

[10] Gelfond M, Lifschitz V. The stable model semantics for logic programming. In: Kowalski R, Bowen KA, eds. Proc. of the 5th Int'l Conf. on Logic Programming. MIT Press, 1988. 1070−1080. http://www.mendeley.com/research/the-stable-model-semantics-for-logic-programming/

[11] Halpern JY, Weissman V. Using first-order logic to reason about policies. ACM Trans. on Information System Security, 2008, 11(4):1−41. [doi: 10.1145/1380564.1380569]

[12] Zhou AY, Shi B. The fixpoint characteristic of semantics of datalog with negation. Journal of Software, 1995,6(5):257−264 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19950501.htm

[13] Ceri S, Gottlob G, Tanca L. What you always wanted to know about datalog (and never dared to ask). IEEE Trans. on Knowledge and Data Engineering, 1989,1(1):146−166. [doi: 10.1109/69.43410]

[14] Jaeger T, Sailer R, Zhang XL. Analyzing integrity protection in the SELinux example policy. In: Proc. of the 12th Conf. on USENIX Security Symp., Vol.12. Washington, 2003. http://dl.acm.org/citation.cfm?id=1251358&CFID=89924345&CFTOKEN=34546777

[15] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engneering, 1997,23(5):279−295. [doi: 10.1109/32.588521]

[16] Cai JY, Qing SH, Liu W, He JB. Analysis on implicit authorization in privilege through rule deduction. Journal of Software, 2008,19(8):2102−2113 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/2102.htm [doi: 10.3724/SP.J.1001.2008.02102]

[17] Van Gelder A, Ross KA, Schlipf JS. The well-founded semantics for general logic programs. Journal of ACM, 1991,38(3):620−650. [doi: 10.1145/116825.116838]

[18] Enderton HB. Introduce to Mathematical Logic. 2nd ed., San Diego: Academic Press, 2001.

[19] Przymusinski T. Well-Founded semantics coincides with three-valued stable semantics. Fundamenta Informaticae, 1990,13(4): 445−463.

[20] Chen WD, Warren DS. Tabled evaluation with delaying for general logic programs. Journal of ACM, 1996,43(1):20−74. [doi: 10.1145/227595.227597]

[21] Chen WD, Warren DS. Query evaluation under the well-founded semantics. In: Proc. of the 12th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Washington, 1993. [doi: 10.1145/153850.153865]

[22] Elliott BD, Padula L, Leonard J. Secure computer system: Unified exposition and multics interpretation. Technical Report, ESD-TR-75-306, MITRE Corporation, 1976.

附中文参考文献:

[12] 周傲英,施伯乐.带否定的 DATALOG 的语义的不动点特性.软件学报,1995,6(5):257−264. http://www.jos.org.cn/1000-9825/19950501.htm

[16] 蔡嘉勇,卿斯汉,刘伟,何建波.基于规则推导的特权隐式授权分析.软件学报,2008,19(8):2102−2113. http://www.jos.org.cn/1000-9825/19/2102.htm [doi: 10.3724/SP.J.1001.2008.02102]

包义保(1976−),男,安徽无为人,博士生,讲师,主要研究领域为智能安全,网络安全.

方滨兴(1960−),男,博士,教授,博士生导师,中国工程院院士,CCF 高级会员,主要研究领域为网络安全,信息内容安全,并行处理,互联网技术.

殷丽华(1973−),女,博士,副研究员,CCF 会员,主要研究领域为网络安全,安全性分析.

郭莉(1969−),女,高级工程师,CCF 会员,主要研究领域为网络与信息安全.