

基于行为逻辑的时序使用控制模型

徐长征,王清贤

(信息工程大学 信息工程学院, 郑州 450002)

(zachxu@gmail.com)

摘要:着眼于单主体访问操作中的原子行为和时序性质,采用时序模态逻辑,提出一种基于行为的时序使用控制模型。该逻辑框架包含了与授权相关的按时间节点预定义的行为、由主客体属性和系统所表达的状态序列及状态谓词。在给出了一种策略语言后,对其语法和语义进行了形式定义,并根据使用控制(UCON)模型族的行为特性给出了控制策略。该逻辑模型不仅可以形式化描述使用控制基本原理,而且有助于准确并自动化地判定策略的可满足性,具有良好的灵活性和表达能力。

关键词:使用控制;行为;状态序列;时序授权

中图分类号: TP309 **文献标志码:** A

Study on temporal usage control model based on action logic

XU Chang-zheng, WANG Qing-xian

(Institute of Information Engineering, Information Engineering University, Zhengzhou Henan 450002, China)

Abstract: By using temporal modality logic in this paper, an action-based temporal usage control model was proposed with focus on atomic actions and temporal properties during a single usage process. The specification of the logic model consists of pre-defined authorization actions, a sequence of states expressed by attributes of subjects, objects, and the system, and state predicates. A policy language was introduced and its syntax and semantics were defined formally, and control policies were given according to action characteristics of core models of Usage Control (UCON). The logic model can not only depict the basic principles of UCON, but be helpful to precisely and automatically determine whether the policies could be satisfied, which shows good flexibility and expressive capability.

Key words: Usage Control (UCON); action; sequence of states; temporal authorization

0 引言

传统的访问控制模型,如基于格的访问控制(Label-Based Access Control, LBAC)和基于角色的访问控制(Role-Based Access Control, RBAC),主要都是考虑主体对目标客体的访问许可,进行静态的授权判定。另一方面,近年来很多研究中,提出很多基于策略的授权管理系统。根据安全策略,主体一旦被授予访问许可,访问就不再受到干预。区别于传统访问控制模型,文献[1-2]作者提出了下一代使用控制(Usage Control, UCON) UCON_{ABC}模型。该概念原型中有两个显著性质:访问判定的持续性和主客体属性的可变性。在使用控制中,不仅在访问之前要检查并作出授权判定,而且在访问过程中还要作重复进行检查。可变性在传统的访问控制模型和策略中也可以发现类似影子。例如“中国墙”策略中^[3],在一个利益冲突集中主体 s 访问过客体 o ,之后 s 就不能访问任何与 o 冲突的客体。由于 s 的访问而引发副效应,主体可访问的潜在客体列表(这里我们视为主体属性)进而发生改变,该变化将约束 s 的下次访问授权。

UCON_{ABC}从原理层面上提供了一种高层指导模型^[2],如果要用程序语言自动实现或验证,需要某种逻辑规范的支持。文献[6]作者提出了一种时序逻辑规范,应用间隔时序逻辑来描述访问控制策略,他们的方法中关注的是更高层次的系

统策略或安全协议。文献[7]作者针对时序数据的访问控制提出了一种时序数据授权模型(Authorization Model for Temporal Data, TDAM),关注点是数据的时序属性。本文关注的重点是单主体访问操作中的原子行为和时序授权,采用Lamport的行为时序逻辑语言,提出了一种基于行为的时序使用控制TUCON(Temporal Usage Control)模型。该逻辑模型的基本组件是作用于主客体和系统属性之间的谓词;按照使用的时间节点,预定义各种行为,在访问期间这些行为将由主体或系统执行;进而给出了一种使用控制策略的描述语言及推理规则。该逻辑模型不仅可以描述使用控制基本原理,而且有助于准确并自动化地判定策略的可满足性,即可以比较直观地使用程序自动进行时序授权判定。

1 相关工作及对比

目前,在访问控制研究领域,大多数被提出的访问控制模型都属于静态访问控制模型。在这一类模型中,主体一旦被授予访问许可,访问就不再受到系统干预,传统的基于角色的访问控制(RBAC)、基于属性的访问控制(Attribute-Based Access Control, ABAC)^[8-9]等都属于这一类模型。

与此相对应的是,本文提出的模型则属于一种动态的访问控制模型。系统对主体的访问许可授权不仅可以在访问前进行判定,还能够在主体访问过程中持续进行判定,与之相伴

收稿日期:2009-04-15;修回日期:2009-05-25。

基金项目:国家 863 计划项目(2007AA01Z471);河南省基础与前沿技术研究计划(082300410150)。

作者简介:徐长征(1976-),男,江西南昌人,博士研究生,主要研究方向:安全操作系统结构研究、访问控制;王清贤(1960-),男,河南卫辉人,教授,博士生导师,主要研究方向:信息安全、算法分析。

随的是,主客体属性有可能发生持续的变化,进而影响到系统对主体的访问许可授权。

此外,本文提出的模型更加抽象,使得静态访问控制模型可以描述为本文模型的一种特例。

文献[1-2]作者提出的使用控制 $UCON_{ABC}$ 模型与本文模型相似,但没有给出模型的形式化描述,本文所提模型由于使用了形式化的逻辑语言进行刻画,使得更有助于明确模型的结构,以及访问判定自动化验证方法的正确性和安全性。

2 使用控制的特性

一个完整的使用过程,按时间节点分为:使用前、进行中使用、使用后,如图 1 所示。在前两个时间段分别实施预授权判定和进行中授权判定。在使用后阶段,因为主体在完成了对客体的使用之后,没有进一步访问控制需求,所有没有必要再执行任何判定。

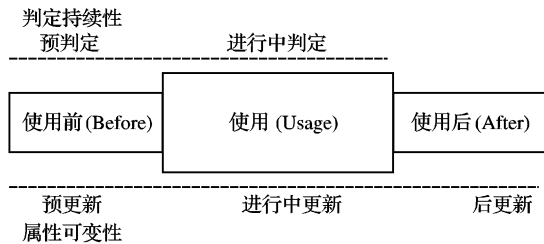


图 1 UCON 模型的持续性与可变性

属性可变性贯穿于全部时间段,相应地也就有三类更新:预更新、进行中更新和后更新,并由引用监控系统执行并监视所有这些更新。主体或客体的属性更新将导致某个系统行为,直接影响到当前使用,还将涉及到与同一主体或客体相关的其他访问。

基于属性可变与授权判定的内在联系,文献[1-2]定义了以下几种授权模型的概念原型:在访问之前,由授权规则做出使用判定,其中模型 $preA_0$ 表示整个使用过程中没有属性更新; $preA_1$ 表示在使用前主体或客体的属性有更新; $preA_3$ 表示在使用之后主体或客体的属性有更新。在访问过程中,检查使用控制并根据授权规则做出判定,其中模型 onA_0 表示整个使用过程中没有属性更新; onA_1 表示在使用之前主体或客体的属性有更新; onA_2 表示在使用中主体或客体的属性有更新; onA_3 表示在使用之后主体或客体的属性有更新。

3 行为时序逻辑

行为时序逻辑 (Temporal Logic of Actions, TLA)^[4] 由时序逻辑^[5] 扩展而来。本节简要介绍基本术语和时序公式的语法定义,然后介绍一些附加的时序模态词及语义。

3.1 基本术语和概念

变量、值、状态是 TLA 的基本要素,常量通常指固定赋值的一种变量。通过对变量 x 赋予 $s(x)$ 值,刻画了一种状态 s ,其中 $s(x)$ 的语义记为 $s[x]$ 。函数 f 就是一个非布尔表达式,其语义 $[f]$ 是指从状态到值的一种映射,如 $[x^2 + y - 3]$ 即是一种映射,将 $s[x]^2 + s[y] - 3$ 的值赋给状态 s ,其中 $s[x]$, $s[y]$ 分别表示 s 对 x , y 的赋值。谓词 P 是一种布尔表达式,其语义 $[P]$ 是状态到布尔值的一种映射。状态 s 满足谓词 P ,当且仅当状态 s 中 $[P]$ 的值 $s[P] = \text{true}$ 。行为 A 是一种布尔值表达式,表示了一种新旧状态的关系,对状态对 (s, t) 赋布尔值 $s[A]t$,如 $x' = y + 1$ 等于 $t[x] = s[y] + 1$ 的布尔值。若 $s[A]t = \text{true}$,我们称 (s, t) 是一个 A -step。在 TLA 中,由于谓词 P 同样是布尔表达式,我们称状态对 (s, t) 是一个 P -step,当且仅

当 $s[P] = \text{true}$ 。对 \forall 行为 A ,设“ $Enabled A$ ”为系统谓词。对状态 s 来说, $s[Enabled A] = \text{true}$,当且仅当在状态 s 中能够开始一个 A -step,从语义上即, $s[Enabled A] \equiv \exists t \in St: s[A]t$,其中 St 表示状态集。

3.2 行为时序公式及语法定义

考虑一个状态序列 $\langle s_0, s_1, s_2, \dots \rangle$,行为 A 的原子公式的语法定义: $\langle s_0, s_1, s_2, \dots \rangle [A] \equiv s_0[A]s_1; \langle s_0, s_1, s_2, \dots \rangle [\Box A] \equiv \forall n \geq 0: s_n[A]s_{n+1}$,其中 \Box 为基本的时序模态词“always”。由于谓词是一种特殊形式的行为,可以同样地定义谓词的语义。

在 TLA 中,公式由谓词和行为构成,并使用逻辑连接符和时序模态词。以下用 BNF 范式给出了公式 F 的语法定义:

$$\begin{aligned} \langle formula \rangle &::= \langle predicate \rangle \mid \langle action \rangle \mid \neg \langle formula \rangle \mid \\ &\langle formula \rangle \wedge \langle formula \rangle \mid \langle formula \rangle \vee \langle formula \rangle \mid \\ &\langle formula \rangle \rightarrow \langle formula \rangle \mid \Box \langle formula \rangle \end{aligned}$$

3.3 时序模态词

模态词 \Box (always) 可以扩展出其他模态词,给使用带来更多方便。用 \Diamond 表示最终地 (eventually), $\Box \Diamond$ 表示“永无止境地常常” (infinitely often)。其中模态词 \Box 和 \Diamond 之间的关系有: $\Diamond F \equiv \neg \Box \neg F$ 。

模态词下一个 (next), 记为 \circ , 其语法定义:

$$\langle s_0, s_1, s_2, \dots \rangle [\circ F] \equiv s_1[F]s_2。$$

模态词直到 (until), 记为 U , 这是一个二元模态词,其语法定义如下: $\langle s_0, s_1, s_2, \dots \rangle [FUG] \equiv \exists i \geq 0: (s_i[G]s_{i+1} \wedge (0 \leq j \leq i \rightarrow s_j[F]s_{j+1}))$ 。 U 和 \Diamond 之间的关系有: $\Diamond F \equiv (F \vee \neg F)UF$ 。

为了描述过去时间里的性质,可在 TLA 中增加过去时态的模态词。对于状态序列 $\langle s_0, s_1, s_2, \dots \rangle$, 设 s_0 为当前时间点的状态,则用状态序列 \dots, s_{-2}, s_{-1} 来表示过去时间点的状态。类似地可以定义过去时序模态词: \blacksquare (has-always-been)、 \blacklozenge (once)、 Θ (previous)、 S (since)。

4 逻辑描述语言

4.1 属性和状态

在 TLA 中,状态是对于变量的赋值。UCON 是一个基于属性的访问控制模型,其属性可以定义为三种不同类型的变量:主体属性、客体属性和系统属性。属性的数据类型依赖于属性的类型,如组成员、角色、安全级、信誉值等,特别地,系统属性是一种与主体或客体无直接关系的变量,如系统时间、地点等。主体或客体的状态是对其属性的赋值。

设函数 $state(s, o, r)$ 为 $\{(s, o, r)\}$ 到 $\{initial, requesting, denied, accessing, revoked, end\}$ 的映射。这 6 种状态值: $initial$ 表示访问 (s, o, r) 还未产生; $requesting$ 表示该访问已产生,正等待系统判定; $denied$ 表示在使用之前,根据授权策略,系统已拒绝该访问请求; $accessing$ 表示系统已经允许了访问,之后主体 s 正在访问该客体; $revoked$ 表示系统回收了进行中访问; end 表示主体 s 完成了使用。

4.2 谓词

谓词是由变量和常量组成的布尔表达式,包括主体属性、客体属性和系统属性。根据属性的不同形式,我们定义四类 UCON 谓词。

1) 一元谓词: 主体或客体属性的谓词, $p(sa)$ 、 $p(oa)$ 。

2) 二元谓词: 作用于主体属性和客体属性之间, $p(sa, oa)$ 。

3) 二元谓词: 属性间关系的谓词 $in(a_1, a_2)$, 其中 a_2 为 a_1 可能的取值集合。

4) 三元谓词: $permit(s, o, r)$, 若主体 s 可以对客体 o 进行 r 权限的访问, 则 $permit(s, o, r) = true$ 。该谓词的值为系统执行授权判定之后的返回结果。

4.3 行为

UCON 中定义了三种属性更新动作来改变主体或客体的状态: $preupdate$ 、 $ongoingupdate$ 、 $postupdate$, 分别在使用前、使用过程中、使用后由引用监控系统来执行。如果系统成功执行某动作, 那么属性值必须更改为新的值, 且该行为为 $true$, 否则为 $false$ 。在具体实现中, 有一种机制来监视进程, 并核查该更新, 如有任何错误发生, 系统可以恢复到先前状态。

前面提到, 在一个进程生命周期内, $state(s, o, r)$ 有 6 种不同的可能取值。从一种状态到另一种状态的转移, 就是我们所说的一种行为, 如图 2。要改变某个访问 (s, o, r) 的状态, 必须对上述三种动作进行扩充, 为描述方便, 统一称之为行为。

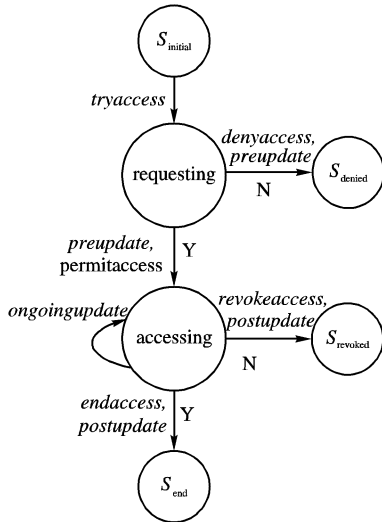


图 2 在一个使用过程中随行为发生的状态转移

在一个使用过程中, 按照时间节点预定义行为, 这些行为分别由主体和系统执行, 如图 3。

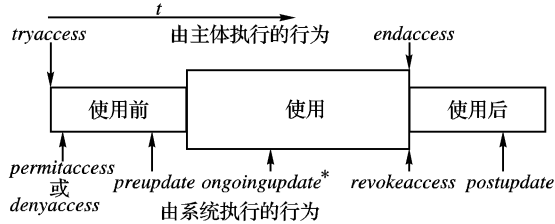


图 3 主体和系统行为

其中由主体执行的行为有 2 个: $tryaccess(s, o, r)$ 表示产生一个新的访问请求 (s, o, r) ; $endaccess(s, o, r)$ 表示结束一个访问 (s, o, r) 。由系统执行的行为有 6 个: $permitaccess(s, o, r)$ 表示同意访问请求 (s, o, r) ; $denyaccess(s, o, r)$ 表示拒绝访问请求 (s, o, r) ; $revokeaccess(s, o, r)$ 表示回收一个进行中访问 (s, o, r) ; $preupdate(attribute)$ 表示在访问之前更新主体或客体属性; $ongoingupdate(attribute)$ 表示在使用阶段更新主体或客体的属性, * 号表示该行为可以由系统重复执行, 不断更新属性; $postupdate(attribute)$ 表示在访问之后更新主体或客体属性。

4.4 策略语言的语法定义

以下用 BNF 范式给出使用控制策略语言的逻辑公式定义:

$\phi ::= a \mid p(t_1, \dots, t_n) \mid (\neg \phi) \mid (\phi \wedge \phi) \mid (\phi \rightarrow \phi) \mid \forall x: \phi \mid \exists x: \phi \mid \square \phi \mid \diamond \phi \mid \bigcirc \phi \mid \phi U \phi \mid \blacksquare \phi \mid \blacklozenge \phi \mid \Theta \phi$

$\mid \phi S \phi$

其中 a 表示行为, $p(t_1, \dots, t_n)$ 为 n 元谓词表达式, x 为变量。

令三元组 $M = (S, P, A)$ 表示时序授权模型, 其中 S 是状态序列; P 是关于主体属性、客体属性的状态谓词的有限集; A 是行为的有限集。在模型 M 中具有状态 s , 满足公式 ϕ , 记为 $M, s \models \phi$ 。据此, 上述公式的语义可以形式地定义如下:

- 1) $M, s_0 \models p$, 当且仅当 $s_0 \models p$, 其中 $p \in P$;
- 2) $M, s_0 \models a$, 当且仅当 $s_0 \xrightarrow{a} s_1$, 其中 $a \in A, s_1$ 为 S 中 s_0 的下一状态;
- 3) $M, s_0 \models \neg \phi$, 当且仅当 $M, s_0 \not\models \phi$;
- 4) $M, s_0 \models \phi_1 \wedge \phi_2$, 当且仅当 $M, s_0 \models \phi_1$, 且 $M, s_0 \models \phi_2$;
- 5) $M, s_0 \models \phi_1 \rightarrow \phi_2$, 当且仅当 $M, s_0 \models \phi_1$, 或 $M, s_0 \not\models \phi_2$;
- 6) $M, s_0 \models \forall x: \phi$, 当且仅当对所有的 $a \in A, M, s_0 \models \phi(x/a)$, 其中 x/a 表示用 x 来取代 a ;
- 7) $M, s_0 \models \exists x: \phi$, 当且仅当对某个 $a \in A, M, s_0 \models \phi(x/a)$, 其中 x/a 表示用 x 来取代 a ;
- 8) $M, s_0 \models \square \phi$, 当且仅当 $\forall n \geq 0: M, s_n \models \phi$;
- 9) $M, s_0 \models \diamond \phi$, 当且仅当 $\exists n \geq 0: M, s_n \models \phi$;
- 10) $M, s_0 \models \bigcirc \phi$, 当且仅当 $M, s_1 \models \phi$;
- 11) $M, s_0 \models \phi_1 U \phi_2$, 当且仅当 $\exists i \geq 0: M, s_i \models \phi_2 \wedge (0 \leq j < i \rightarrow M, s_j \models \phi_1)$;
- 12) $M, s_0 \models \blacksquare \phi$, 当且仅当 $\forall n < 0: M, s_n \models \phi$;
- 13) $M, s_0 \models \blacklozenge \phi$, 当且仅当 $\exists n < 0: M, s_n \models \phi$;
- 14) $M, s_0 \models \Theta \phi$, 当且仅当 $M, s_{-1} \models \phi$;
- 15) $M, s_0 \models \phi_1 S \phi_2$, 当且仅当 $\exists i < 0: M, s_i \models \phi_2 \wedge (i < j \leq 0 \rightarrow M, s_j \models \phi_1)$ 。

5 授权策略分析与应用

本节用前面给出的策略语言, 通过分析 UCON 模型族的各种行为特性, 给出控制策略。

1) $preA_0$ 模型

如文献[1-2]所述, 大多数传统的访问控制模型可以用 $preA_0$ 模型表示。其主要特征是在访问发生之前由系统作出授权判定, 主客体属性不会发生更新。使用控制策略具有如下形式:

$p_1 \wedge \dots \wedge p_i \rightarrow permit(s, o, r)$

$tryaccess(s, o, r) \wedge permit(s, o, r) \rightarrow \bigcirc(permitaccess(s, o, r))$

其中: p_1, \dots, p_i 为状态谓词, $permit$ 谓词表示 s 可以对 o 进行 r 方式的访问, $permitaccess$ 行为给予 s 许可并启动访问。

2) $preA_1$ 模型

在访问之前检查授权规则, 进行授权判定; 给予主体访问许可之前, 会有一个或多个属性更新行为。我们使用控制策略如下:

$p_1 \wedge \dots \wedge p_i \rightarrow permit(s, o, r)$

$permitaccess(s, o, r) \rightarrow \blacklozenge(tryaccess(s, o, r) \wedge permit(s, o, r) \wedge \diamond(preupdate(attribute)))$

其中 $attribute$ 指主体或客体属性。

3) $preA_3$ 模型

在访问之前检查授权规则, 进行授权判定; 在使用进程之后, 会有一个或多个更新行为。使用控制策略如下:

$p_1 \wedge \dots \wedge p_i \rightarrow permit(s, o, r)$

$permitaccess(s, o, r) \rightarrow \blacklozenge(tryaccess(s, o, r) \wedge permit(s, o, r) \wedge \diamond(postupdate(attribute)))$

需要说明的是,该模型在同意访问之后不再实施授权判定,也就不存在访问的回收。

4) $_{on}A_0$ 模型

这是一个预授权模型,在访问期间实施授权策略:

$permitaccess(s, o, r) \rightarrow \square(\neg(p_1 \wedge \dots \wedge p_i) \wedge (state(s, o, r) = accessing) \rightarrow revokeaccess(s, o, r))$

在 $permitaccess$ 行为之后,主体在 $accessing$ 客体期间进行授权检查,要么授权谓词 p_1, \dots, p_i 都为 true,要么访问立即被系统收回。

5) $_{on}A_1$ 模型

在主体开始访问客体之前存在属性更新行为,且没有授权检查。控制规则如下:

$permitaccess(s, o, r) \rightarrow \blacklozenge(tryaccess(s, o, r) \wedge \diamond(preupdate(attribute)))$

$permitaccess(s, o, r) \rightarrow \square(\neg(p_1 \wedge \dots \wedge p_i) \wedge (state(s, o, r) = accessing) \rightarrow revokeaccess(s, o, r))$

6) $_{on}A_2$ 模型

在使用期间存在属性更新行为,且要进行授权检查。控制规则如下:

$permitaccess(s, o, r) \rightarrow \square(\neg(p_1 \wedge \dots \wedge p_i) \wedge (state(s, o, r) = accessing) \rightarrow revokeaccess(s, o, r))$

$endaccess(s, o, r) \vee revokeaccess(s, o, r) \rightarrow \blacklozenge(permitaccess(s, o, r) \wedge \diamond(ongoingupdate(attribute)))$

需要指出的是,在第 2 条规则中,我们只说明了最终更新行为的这一种情况。在许多实际应用中,常常是进行中访问的每个状态下都会有更新。

7) $_{on}A_3$ 模型

在使用进程之后存在属性更新行为。控制规则如下:

$permitaccess(s, o, r) \rightarrow \square(\neg(p_1 \wedge \dots \wedge p_i) \wedge (state(s, o, r) = accessing) \rightarrow revokeaccess(s, o, r))$

若访问由主体结束:

$endaccess(s, o, r) \rightarrow \diamond(postupdate(attribute))$

若访问被系统回收:

$revokeaccess(s, o, r) \rightarrow \diamond(postupdate(attribute))$

6 实例说明

本节以 $_{pre}A_1$ 和 $_{on}A_2$ 模型为例说明如何应用第 4 章给出的策略语言进行描述。

假设在某个数字版权管理 DRM 中,主体 Alice 需花费一定存款才能阅读电子书 *ebook1*。则该访问控制可以用如下策略语言进行描述。

设 Alice 的属性 *credit* 表示其拥有的存款, *ebook1* 的属性 *value* 表示需要花多少存款才能具有对该书的 *read* 权限。

$dominate(Alice.credit, ebook1.value) \rightarrow permit(Alice, ebook1, read)$

$permitaccess(Alice, ebook1, read) \rightarrow \{ \blacklozenge(tryaccess(Alice, ebook1, read) \wedge \diamond(preupdate(Alice.credit))) \wedge permit(Alice, ebook1, read) \}$

其中,行为 *preupdate* 执行的是 $Alice.credit' = Alice.credit(ebook1.value)$ 。上述第 1 条规则说明,只要 Alice 拥有的存款大于 *ebook1.value*,就可以阅读 *ebook1*;而第 2 条规则隐含了对 Alice 所拥有存款数额的更新。

显然,上述访问控制属于 $_{pre}A_1$ 模型。现在假设在另外一种数字版权管理 DRM 中, Alice 要阅读 *ebook1*,不需要一次付清款项,系统将根据 Alice 阅读 *ebook1* 的时间长短来计费,即每隔一段时间,扣除 Alice 一定数额的存款(假设 ¥1),并检查其

存款是否已为 0,如果是,则拒绝 Alice 再继续阅读。该访问控制可用如下策略语言进行描述。

$permitaccess(Alice, ebook1, read) \rightarrow \square(\neg dominate(Alice, credit, 0) \wedge (state(Alice, ebook1, read) = accessing) \rightarrow revokeaccess(Alice, ebook1, read))$
 $endaccess(Alice, ebook1, read) \vee revokeaccess(Alice, ebook1, read) \rightarrow \blacklozenge(permitaccess(Alice, ebook1, read) \wedge \square \diamond(ongoingupdate(Alice.credit)))$

其中, *ongoingupdate* 表示每隔一段时间执行 $Alice.credit' = Alice.credit - ¥1$ 。上述第 1 条规则说明,只要 Alice 还有存款,则允许他阅读 *ebook1*,否则系统将收回阅读权限;第二条则隐含了 Alice 在阅读过程中,系统对其存款数额的持续更新。

7 结语

本文在分析使用控制中两个主要特性的基础上,采用行为时序模态逻辑语言,提出了一种基于行为的时序使用控制模型。该逻辑模型描述了包括主体、客体、系统在内的状态序列,分析了系统中主客体和系统的状态转移过程,按照时间节点,预定义系统的各种行为。这也是可信计算及评估中常见的一种方法。模型中时序性质是主客体属性的可变性造成的结果,由于使用行为的副效应,这些结果还会发生变化。

下一步我们需要讨论使用控制中义务和条件的逻辑方法。义务作为与授权行为分离的行为,必需在访问之前或期间执行;而条件需要使用系统属性相关的谓词进行说明。该逻辑系统的安全策略规范和描述能力还有待进一步完善和增强,以提高使用控制的建模能力。在实际应用中,要能够精确地且自动化地实施授权判定,还需要考虑将多个核心模型的控制策略加以组合。

参考文献:

- [1] SANDHU R, PARK J. Usage control: A vision for next generation access control[C]// The Second International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security. Heidelberg: Springer Berlin, 2003: 17–31.
- [2] PARK J, SANDHU R. The $UCON_{ABC}$ usage control model[J]. ACM Transactions on Information and Systems Security, 2004, 7(1): 128–174.
- [3] BREWER D, NASH M. The Chinese wall security policy[C]// Proceedings of the IEEE Symposium on Security and Privacy. Oakland: IEEE Computer Society Press, 1989: 206–214.
- [4] LAMPORT L. The temporal logic of actions[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1994, 16(3): 872–923.
- [5] MANNA Z, PNUELI A. The temporal logic of reactive and concurrent systems specification[J]. Operating Systems Review (SIGOPS), 1993, 27(4): 1–3.
- [6] SIEWE F, CAU A, ZEDAN H. Compositional framework for access control policies enforcement[C]// Proceedings of the ACM Workshop on Formal Methods in Security Engineering. New York: ACM Press, 2006: 32–42.
- [7] GAL A, ATLURI V. An authorization model for temporal data[C]// Proceedings of the 7th ACM conference on Computer and communications security, New York: ACM Press, 2002: 144–153.
- [8] 林莉, 怀进鹏, 李先贤. 基于属性的访问控制策略合成代数[J]. 软件学报, 2009, 20(2): 403–414.
- [9] WANG LING-YU, WIJESEKERA D, JAJODIA S. A logic based framework for attributed based access control[C]// Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering. New York: ACM Press, 2004: 45–55.