一种基于改进 $UCON_C$ 的网格授权策略规范*)

桂劲松[1] 陈志刚[1] 郭 迎[2]

(中南大学信息科学与工程学院 长沙 410083)[1]

(上海交通大学区域光纤通信网与新型光通信系统国家重点实验室 上海 200030)[2]

摘 要 由基于 TLA(Temporal Logic of Action)的使用控制策略规范所表达的授权策略得出的决策结果仅能表达简单的"允许"或"拒绝",这在服务网格中难以实现决策组件与执行组件的合理分工,也不利于独立授权过程的并发执行。因此,本文首先提出了委托凭证作为决策结果的细粒度表达方式,然后对基于条件谓词决策的策略规范进行了改进与扩充,将原来简单的访问状态改进与扩充为委托凭证处理过程的状态组合。决策组件能根据访问请求时的系统状态输出合理的委托凭证,也能根据随后的系统状态变化进行再决策,以转换委托凭证的处理状态。最后对新的策略规范的完备性和正确性进行了证明,并通过实例展示了策略规范的表达能力和访问请求的决策过程。

关键词 服务网格,授权决策,委托凭证,策略规范

An Improved $UCON_C$-based Authorization Policy Specification in Grid

GUI Jin-song[1] CHEN Zhi-gang[1] GUO Ying[2]

(School of Information Science and Engineering, Central South University, Changsha 410083, China)[1]

(National Key Laboratory on Optical Communication and New Optical System, Shanghai Jiaotong University, Shanghai 200030, China)[2]

Abstract The decision-making result of the usage control policy specification based on temporal logic of action only expresses "permission" or "rejection". In service grid, it is difficult to reasonably divide the work of PDP(Policy Decision Point) and PEP(Policy Enforcement Point), and the independent authorization processes are not implemented simultaneously. For that, the paper firstly presents the delegation certification that can express fine-grained rights, and the decision-making result can be expressed by it. Secondly, the paper improves and expands the policy specification based on condition predication decision-making, and defines the delegation certification processing statuses to replace the simple access status. PDP can make the reasonable delegation certification based on the system status when a request arrives, and also make decision to change the delegation certification processing status when the system status is changed. Finally, the completeness and soundness of the new policy specification are proved, and its expressive capability and the decision-making process of the access request are exhibited through an example.

Keywords Service grid, Authorization decision, Delegation certification, Policy specification

1 引言

UCON[1](Usage CONtrol)概念首先由 J. Park 和 R. Sundhu 提出。它扩展了传统的访问控制功能,定义了三个授权决策因素:授权(Authorization)、职责(Obligation)、条件(Condition),提出了访问控制的连续性(Continuity)和易变性(Mutability)两个重要属性。文献[2]提出了一个概念模型 $UCON_{ABC}$,从授权、职责和条件三方面给出了 16 个基本模型及其逻辑规范,但没有给出相应的授权语言和实用框架。文献[3]基于 TLA(Temporal Logic of Action)对 $UCON_{ABC}$ 进行了形式化描述,提出了一组确定的模式规则并证明了它的完备性和正确性。通过这组模式规则可以为有安全需求的系统制定 UCON 策略。

在类似文献[3]的模式规则中,当一个访问请求被递交到决策组件时,决策组件仅输出简单的决策结果:允许或拒绝。这在传统的授权机制下是可行的。例如,在访问控制基本组件(定义在 RFC2094 中)中,通常 PDP(Policy Decision Point)是独立于服务和应用的,而 PEP(Policy Enforcement Point)则是与这些部件紧密耦合的,因此 PEP 不能过于复杂,简单的决策结果适合这种情况。但在授权负荷较重的网格环境下,授权功能过分集中于一个单一组件 PDP,不利于多个独立授权过程的并发执行,从而影响网格授权系统的效率。

文献[3]形式化描述的 $UCON_{ABC}$ 是一个庞大的模型族。依据三个授权决策因素可划分为三个模型:只依据授权谓词(authorization predicate)进行决策的 $UCON_A$,只依据职责行动(obligation action)进行决策的 $UCON_B$,只依据条件谓词(condition predicate)进行决策的 $UCON_C$。本文针对服务网格授权的新特点,对 $UCON_C$ 进行研究和改进,提出一套满足服务网格授权决策需求的策略规范并证明其完备性和正确性。

2 服务网格授权的新特点

从抽象层次看,现有网格授权系统[4,5]存在相似的语义:输入发起人(Initiator)身份和属性等描述信息、请求的行动(Action)描述信息、目标资源的访问策略信息、授权上下文信

*)国家自然科学基金资助项目(60573127)。桂劲松 讲师,博士研究生,主要研究方向为网格授权管理;陈志刚 教授,博士生导师,主要研究方向为并行分布式处理、信息安全;郭 迎 副教授,博士后,主要研究方向为信息安全、量子加密及应用。

息。输出授权决策结果,以指示请求的行动是被执行还是被拒绝。这种过于简单的授权决策结果导致授权执行组件缺乏细粒度控制权限执行的依据,同时不利于授权过程中并行成分的挖掘。

在按需、动态、即时构建服务虚拟组织(Virtual Organization,简称 VO)以协同进行问题求解的网格发展必然趋势[6]下,VO 要对聚集的资源进行统一的授权管理。但从资源的角度来看,每个资源只关心其自身的访问控制策略和运行于其上的应用,并不关心其它资源的情况,因此有必要在 VO 中设置一个授权执行组件 PEP,以充当全局控制器。这样的设置不仅可以合理分担 PDP 繁重的工作,从而使得独立的多个授权过程的决策与执行能够并行,而且可以大大简化资源本地域 PEP 的工作。在这样的背景下,PDP 输出的授权决策结果就不能是简单的"允许"或"拒绝",而应该是一个表示细粒度权限的授权证书。这个证书就是下面要定义的委托凭证。

委托凭证用到了下面部分基础元素[7]:(1)$U=\{u_1,u_2,\cdots,u_m\}$是所有用户的集合;(2)$R=\{r_1,r_2,\cdots,r_n\}$是所有角色的集合;(3)$UA\subseteq U\times R$ 是从 U 到 R 的多对多映射。常规用户是由系统管理员指派角色的用户;委托用户是通过委托指派角色的用户。UA 划分为 UAO 和 UAD,分别表示用户与被授予角色的关系和用户与其被委托角色的关系。委托凭证的定义如下:

定义 1 角色树是委托授权的基本单位,表示为 $r_{tree}=(r_0(r_1(r_{11})(r_{111}(\cdots),\cdots,r_{11n}(\cdots)),\cdots,r_{1m}(\cdots)),\cdots,r_k(r_{k1}(\cdots))))$。

定义 2 在委托票据 $dt=(uad,pt,n,ae,dep)$[8]中,用 r_{tree} 替代 $uad=(u,r)\in UAD$ 中的 r,并将 r 看成 r_{tree} 的未给出树形结构的缩写式。

定义 3 委托传播树表示为 $spr_{tree}=(dt_0(dt_1(dt_{11}(dt_{111}(\cdots),\cdots,dt_{11n}(\cdots)),\cdots,dt_{1m}(\cdots)),\cdots,dt_k(dt_{k1}(\cdots))))$。树的根结点拥有从常规用户得到的委托票据 dt_0。$dt_1,dt_2,\cdots,dt_k$ 是根结点可签发的委托票据,根结点可将 dt_0 的角色树的一部分或全部委托给 $dt_1,dt_2,\cdots,dt_k$ 的持有者。$dt_{11},dt_{12},\cdots,dt_{1m}$ 是 dt_1 的持有者可签发的委托票据,dt_1 的持有者可将 dt_1 的角色树的一部分或全部委托给 $dt_{11},dt_{12},\cdots,dt_{1m}$,其余依次类推。

定义 4 委托凭证由 $dc=(n_d,n_b,spr_{tree})\in DC$ 组成。其中 n_d 表示委托权限传播的许可步数,即委托深度;n_b 表示在每步中权限被委托的许可用户数,即委托宽度。

我们定义的委托凭证全面支持委托授权的临时性、关联性、部分性、传播性限制等需求的细粒度表达。例如,其中的委托票据中的 pt,n,ae 提供了丰富的时间限制功能;委托票据中的 dep 能够限制存在利害关系的角色[9]同时激活;通过对其中的角色树 r_{tree} 进行剪枝很容易实现部分委托功能;利用其中的委托传播树 spr_{tree} 能够方便地表达委托传播的上下文信息。而现有工作[8,10]仅支持上述委托需求的某些方面。角色树的提出解决了现有工作[11]采用分解角色实现部分委托带来的系统角色管理的复杂性问题。委托传播树的提出克服了现有工作[8,10,11]缺乏委托传播上下文信息的不足。

3 基于改进 $UCON_C$ 的授权策略规范

3.1 对现有 $UCON_C$ 的改进

文献[3]定义的一些概念如下:(1)系统状态定义为对一组变量的一次赋值,其中变量包括主体属性、客体属性、系统属性这三类变量。(2)实体(主体或客体)表示为一个有限属性集。(3)函数被定义为一个或多个属性和常量构成的表达式。在形式上,函数是从一组属性值到一个新值的映射。(4)单一使用过程由(s,o,r)来表示,其中 s 表示主体,o 表示客体,r 表示权限。(5)(s,o,r)的状态被定义为特殊系统变量,并定义函数 $state(s,o,r)$为从$\{(s,o,r)\}$到$\{initial,requesting,denied,acces\ sing,revoked,end\}$的映射。其中状态 *initial* 的语义是$(s,o,r)$尚未产生,而状态 *requesting* 的语义是(s,o,r)已经产生并正等待系统的使用决策;状态 *denied* 的语义是在使用前根据 UCON 策略系统拒绝了访问请求;状态 *acces sing* 的语义是系统已允许(s,o,r)并且在此后主体一直在访问客体。当系统在使用期间撤销了访问,此(s,o,r)就改变到状态 *revoked*,或者当主体完成了使用,(s,o,r)改变到状态 *end*。(6)谓词是由变量、函数、常量构成的逻辑表达式。其语义是从系统状态到布尔值的映射。若状态中所指派的属性值满足谓词,则状态满足谓词。(7)使用控制行动包括更新属性值的行动和改变单一使用过程状态 $state(s,o,r)$的行动。对前者,存在三种更新行动:*preupdate*,*onupdate*,*postupdate*,各自在使用前、使用中、使用后执行。若系统成功执行了一个更新,属性值就改变为新值,行动为 true,否则为 false。对后者,使用控制行动由改变一个(s,o,r)的状态的主体或系统执行。在一个(s,o,r)的生命周期中,$state(s,o,r)$存在六种状态值,从一种状态到另一种状态的转换就是一种使用控制行动。在这些行动中,*tryaccess*(s,o,r)由主体 s 执行,产生一个新的访问请求(s,o,r);*permitaccess*(s,o,r)由系统执行,准予访问请求(s,o,r);*denyaccess*(s,o,r)由系统执行,拒绝访问请求(s,o,r);*revokeaccess*(s,o,r)由系统执行,撤销一个正在进行的访问(s,o,r);*endaccess*(s,o,r)由主体 s 执行,结束一个访问(s,o,r);*preupdate*(*attribute*)由系统执行,在准予访问前或在拒绝访问后更新主体或客体属性;*onupdate*(*attribute*)由系统执行,在使用期间更新主体或客体属性;*postupdate*(*attribute*)由系统执行,在访问后更新主体或客体属性。

为了适应服务网格授权的新特点,我们将状态值 *acces sing*(访问状态)细化为图 1 中虚线框中的状态组合,并将 *permitaccess*(s,o,r)(允许访问)的语义改为:将状态值 *requesting*(请求状态)改变为"凭证请求"状态,即将主体 s 对客体 o 的权限 r 的请求转换为主体 s 对委托凭证 dc 的请求。另外,到达状态值 *revoked*(撤销状态)的前提是"凭证撤销"状态而非访问进行中。

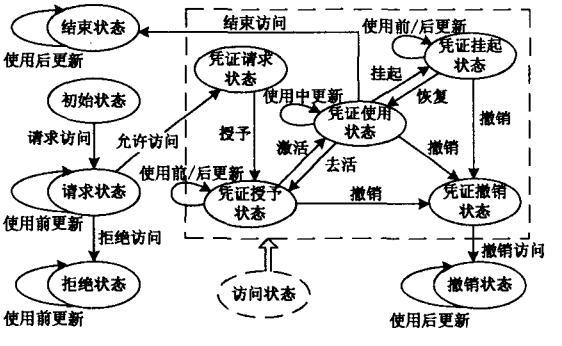

图 1 单一使用过程的状态变迁

定义 5 凭证处理过程用(s,dc)来表示,其中 s 为主体,dc 为委托凭证。

定义 6 凭证处理状态集{*request_dc*(凭证请求),*grant_*

dc(凭证授予),u sin g_dc(凭证使用),$hold\_dc$(凭证挂起),$revoke\_dc$(凭证撤销)}被定义来表示(s,dc)的状态。其中 $request\_dc$ 的语义是(s,dc)已经产生并正等待系统授予 dc;$grant\_dc$ 的语义是 dc 已被授予并随时可被激活使用;u sin g_dc 的语义是 dc 处于 PEP 的使用中;$hold\_dc$ 的语义是 dc 处于更新中而暂时不能被 PEP 使用;$revoke\_dc$ 的语义是 dc 已被系统撤销而不能再被 PEP 使用。

定义 7 (s,dc)的状态表示为特殊系统变量,并定义函数 $state(s,dc)$为从{(s,dc)}到凭证处理状态集的映射。

定义 8 改变(s,dc)状态的使用控制行动定义如下:$grant(s,dc)$(授予)由系统执行,若为 true,则为主体 s 授予委托凭证 dc;$activate(s,dc)$(激活)由系统执行,若为 true,则激活主体 s 的委托凭证 dc;$inactivate(s,dc)$(去活)由系统执行,若为 true,则去活主体 s 的委托凭证 dc;$hold(s,dc)$(挂起)由系统执行,若为 true,则挂起主体 s 的委托凭证 dc;$restore(s,dc)$(恢复)由系统执行,若为 true,则恢复主体 s 的委托凭证 dc;$revoke(s,dc)$(撤销)由系统执行,若为 true,则撤销主体 s 的委托凭证 dc。

定义 9 一个 $UCON_C$ 的抽象形式定义为三元组 $M_A=(S,P_C,A_A)$,其中 S 是系统状态序列集,P_C 是由系统属性构成的条件谓词的有限集,A_A 是使用控制行动的有限集。

定义 10 在 $UCON_C$ 中,逻辑公式 ϕ 由条件谓词和使用控制行动通过逻辑连接符和时态操作符构成。其 BNF 语法为 $\phi::=a|p|(\neg\phi)|(\phi\wedge\phi)|(\phi\rightarrow\phi)|\Box\phi|\Diamond\phi|\bigcirc\phi|\phi U\phi|\boxminus\phi|\blacktriangle\phi|\Theta\phi|\phi S\phi$,其中 a 是使用控制行动,p 是条件谓词;$\neg$,$\wedge$,$\rightarrow$是逻辑连接符,其语义分别是"非","与","蕴涵";$\Box$,$\Diamond$,$\bigcirc$,U,$\boxminus$,$\blacktriangle$,Θ,S 是时态操作符,其语义分别是"*Always*","*Eventually*","*Next*","*Until*","*Has-always-been*","*Once*","*Previous*","*Since*"。

时态操作符语义的详细解释和使用示例可参见文献[3]。我们可以根据服务网格授权的新特点对文献[3]中形式化描述的基于条件谓词决策的八种核心模型进行改进并统称为 SG_UCON_C。这些模型的形式化细节不是本文所要讨论的内容,而本文讨论的是 SG_UCON_C 的一般授权策略规范。

3.2 SG_UCON_C 授权策略规范

在 SG_UCON_C 中的几个假设:第一,一条 SG_UCON_C 策略被看成一组单一使用过程(s,o,r)和凭证处理过程(s,dc)的逻辑公式,在此不考虑并行使用过程的相互影响。第二,在访问请求产生前,请求主体和目标客体存在于系统中,因而未在逻辑规范中对创建和销毁主体和客体进行详细说明。第三,时间轴被限定在单一使用过程的使用期间。第四,SG_UCON_C 的所有授权策略被定义为正权限(授予权限),即对一个访问请求,若无根据属性值授予权限的策略,则默认为拒绝访问。

在 SG_UCON_C 中,使用控制决策由授权谓词决定。对(s,o,r)、(s,dc),条件谓词 $pc_1,pc_2,\cdots,pc_i$,触发属性更新的谓词 $pu_1,pu_2,\cdots,pu_j$,根据 SG_UCON_C 中的核心模型规范,SG_UCON_C 策略可由两种逻辑规则表示——使用控制决策规则和更新规则。使用控制决策规则如下:

$CR_1:permitaccess(s,o,r)\rightarrow\Delta(tryaccess(s,o,r)\wedge(\wedge_n pc_n))$;

$CR_2:grant(s,dc)\rightarrow\Delta(permitaccess(s,o,r)\wedge(\wedge_n pc_n))$;

$CR_3:activate(s,dc)\rightarrow\Delta(grant(s,dc)\wedge(\wedge_n pc_n))$;

$CR_4:\Box(\neg(\wedge_n pc_n)\wedge(state(s,dc)=u\ \text{sin}\ g\_dc)\rightarrow inactivate(s,dc))$;

$CR_5:\Box(\neg(\wedge_n pc_n)\wedge(state(s,dc)=u\ \text{sin}\ g\_dc)\rightarrow hold(s,dc))$;

$CR_6:restore(s,dc)\rightarrow\Delta(hold(s,dc)\wedge(\wedge_n pc_n))$;

$CR_7:\Box(\neg(\wedge_n pc_n)\wedge(state(s,dc)=u\ \text{sin}\ g\_dc)\rightarrow revokes(s,dc))$;

$CR_8:\Box(\neg(\wedge_n pc_n)\wedge(state(s,dc)=grant\_dc)\rightarrow revokes(s,dc))$;

$CR_9:\Box(\neg(\wedge_n pc_n)\wedge(state(s,dc)=hold\_dc)\rightarrow revokes(s,dc))$;

$CR_{10}:restore(s,o,r)\rightarrow\Delta revoke(s,dc)$。

若使用前的决策结果为真,则访问请求被授予;若进行中的决策结果为真,则进行中的访问被继续。对一个访问来说,它的使用前决策和进行中决策组件可能相同或不同。三种类型的更新行动被表示为如下更新规则:

$UR_1:permitaccess(s,o,r)\rightarrow\Delta preupdate(attribute)$;

$UR_2:activate(s,dc)\rightarrow\Delta preupdate(attribute)$;

$UR_3:activate(s,dc)\rightarrow\Diamond(onupdate(attribute)\wedge\Diamond endaccess(s,o,r))$;

$UR_4:activae(s,dc)\rightarrow\Diamond(onupdate(attribute)\wedge\Diamond revoke(s,dc))$;

$UR_5:\Box((state(s,dc)=u\ \text{sin}\ g\_dc)\rightarrow onupdate(attribute))$;

$UR_6:\Box((state(s,dc)=u\ \text{sin}\ g\_dc)\wedge pu_1\wedge\cdots\wedge pu_j\rightarrow onupdate(attribute))$;

$UR_7:restore(s,dc)\rightarrow\Delta preupdate(attribute)$;

$UR_8:inactivate(s,dc)\rightarrow\Diamond postupdate(attribute)$;

$UR_9:hold(s,dc)\rightarrow\Diamond postupdate(attribute)$;

$UR_{10}:endaccess(s,o,r)\rightarrow\Diamond postupdate(attribute)$;

$UR_{11}:revokeaccess(s,o,r)\rightarrow\Diamond postupdate(attribute)$。

定理 1(SG_UCON_C 逻辑规则的完备性) 任意 SG_UCON_C 策略都能用非空使用控制决策规则集和更新规则集表示。

证明:由使用控制决策规则和更新规则的构建可知:$CR_1,CR_2,\cdots,CR_{10}$用于单一使用过程和凭证处理过程的不同阶段以进行控制决策,因而它们不冲突。对更新规则也是如此。而且,使用控制决策规则集详细说明了单一使用过程和凭证处理过程的所有可能决策,更新规则集详细说明了单一使用过程和凭证处理过程的所有可能更新。因此,一般 SG_UCON_C 策略能被非空使用控制决策规则集和更新规则集表示。

定理 2(SG_UCON_C 逻辑规则的正确性) 对由非空使用控制决策规则集和更新规则集实例化的任意策略规则,至少存在一个 SG_UCON_C 的核心模型,其中的系统状态转换满足策略规则。

证明:针对一个单一使用过程(s,o,r)和凭证处理过程(s,dc),我们构建一个 SG_UCON_C 模型来满足分别由 21 个模式规则实例化得到的 21 个逻辑公式,每一个对应唯一的模式规则。设定 $CR'_1,CR'_2,\cdots,CR'_{10}$ 分别是 $CR_1,CR_2,\cdots,CR_{10}$ 的实例;$UR'_1,UR'_2,\cdots,UR'_{11}$ 分别是每个唯一模式更新规则的实例。不失一般性,我们假设在这些更新规则中所有属性都不同,因为在同一属性上的多次更新能被约减到一个单次更新。设定在一个系统中,一个状态由所有规则中的属性(主体的、客体的、系统的)表示。最初,系统状态是 s_0,$state(s,o,r)$

$=initial$。如图2所示的状态转换由下列步骤构建：

(1)在 s_0 中，主体 s 产生一个对客体 o 要求权限 r 的访问请求($tryaccess$)，$state(s,o,r)$ 的值被改变为 $requesting$，并且系统状态转换为 s_1。其它属性值与 s_0 中的相同。

(2)若存在 s_1 中的系统属性不满足 CR'_1 中条件谓词的情况，那么行动 $denyaccess(s,o,r)$ 将系统状态 s_1 转换到 s_2，且 $state(s,o,r)=denied$。

(3)若 s_1 中的系统属性满足 CR'_1 中的所有条件谓词，则 UR'_1 中的更新行动被执行，系统状态转换到 s_3。

(4)在 s_3 中，由系统执行行动 $permitaccess(s,o,r)$ 将系统状态转换到 $s_{4\text{-}1}$，且转换单一使用过程的状态 $state(s,o,r)=requesting$ 为凭证处理过程的状态 $state(s,dc)=request\_dc$。

(5)在 $s_{4\text{-}1}$ 中，由系统执行行动 $grant(s,dc)$ 将系统状态转换到 $s_{4\text{-}2}$，且 $state(s,dc)=grant\_dc$。

(6)若 $s_{4\text{-}2}$ 中的系统属性满足 CR'_3 中的所有条件谓词，则 UR'_2 中的更新行动被执行，系统状态转换到 $s_{4\text{-}3}$。

(7)在 $s_{4\text{-}3}$ 中，由系统执行行动 $activate(s,dc)$ 将系统状态转换到 $s_{4\text{-}4}$，且 $state(s,dc)=u\ \sin\ g\_dc$。

(8)若 $s_{4\text{-}4}$ 中的系统属性满足 CR'_4，CR'_5，CR'_7 中的所有条件谓词，则 UR'_5 中的更新行动被系统执行；若在状态 $s_{4\text{-}4}$ 中的主客体属性满足中 UR'_6 的所有谓词，则系统执行 UR'_6 中的更新操作。在这两种情况下，系统状态都转换到 $s_{4\text{-}5}$。

(9)若存在 $s_{4\text{-}5}$ 中的系统属性不满足 CR'_4 中条件谓词的情况，则 UR'_8 中的更新行动以及行动 $inactivate(s,dc)$ 先后被系统执行，系统状态转换到 $s_{4\text{-}6}$，且 $state(s,dc)=grant\_dc$。

(10)$s_{4\text{-}6}\rightarrow s_{4\text{-}7}\rightarrow s_{4\text{-}8}\rightarrow s_{4\text{-}9}$ 的状态转换过程的操作函数与 $s_{4\text{-}2}\rightarrow s_{4\text{-}3}\rightarrow s_{4\text{-}4}\rightarrow s_{4\text{-}5}$ 相同，但操作函数的参数(如属性值)可以不同。

(11)若存在 $s_{4\text{-}8}$ 中的系统属性不满足 CR'_7 中的条件谓词的情况，则 UR'_4 中的更新行动以及行动 $revoke(s,dc)$ 先后被系统执行，系统状态转换到 $s_{4\text{-}10}$，且 $state(s,dc)=revoke\_dc$。

(12)在 $s_{4\text{-}9}$ 中，由系统执行行动 $revoke(s,dc)$ 将系统状态转换到 $s_{4\text{-}11}$，且 $state(s,dc)=revoke\_dc$。

(13)若存在 $s_{4\text{-}5}$ 中的系统属性不满足 CR'_5 中的条件谓词的情况，则 UR'_9 中的更新行动以及行动 $hold(s,dc)$ 先后被系统执行，系统状态转换到 $s_{4\text{-}13}$，且 $state(s,dc)=hold\_dc$。

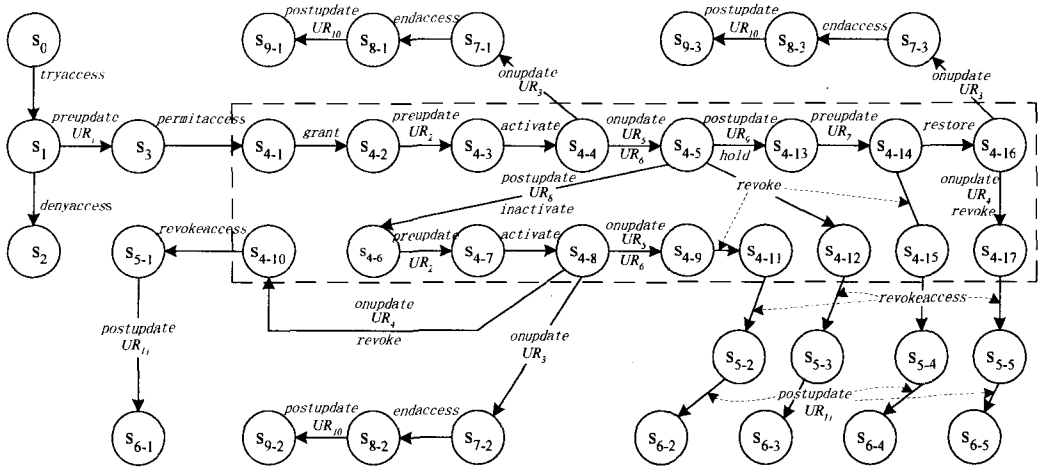

图2 SG_UCON_C 的系统状态变迁图

(14)若 $s_{4\text{-}13}$ 中的系统属性满足 CR'_3 中的所有条件谓词，则 UR'_7 中的更新行动被执行，系统状态转换到 $s_{4\text{-}14}$。

(15)$s_{4\text{-}5}\rightarrow s_{4\text{-}12}$ 和 $s_{4\text{-}14}\rightarrow s_{4\text{-}15}$ 的状态转换过程的操作函数与 $s_{4\text{-}9}\rightarrow s_{4\text{-}11}$ 相同，但操作函数的参数(如属性值)可以不同。

(16)在 $s_{4\text{-}14}$ 中，由系统执行行动 $restore(s,dc)$ 将系统状态转换到 $s_{4\text{-}16}$，且 $state(s,dc)=u\ \sin\ g\_dc$。

(17)$s_{4\text{-}16}\rightarrow s_{4\text{-}17}$ 的状态转换过程的操作函数与 $s_{4\text{-}8}\rightarrow s_{4\text{-}10}$ 相同，但操作函数的参数(如属性值)可以不同。

(18)在 $s_{4\text{-}10}$ 中，由系统执行行动 $revokeaccess(s,o,r)$ 将系统状态转换到 $s_{5\text{-}1}$，且转换凭证处理过程的状态 $state(s,dc)=revoke\_dc$ 为单一使用过程的状态 $state(s,o,r)=revoked$。

(19)在 $s_{5\text{-}1}$ 中，UR'_{11} 中的更新行动被执行，并且系统状态转换到 $s_{6\text{-}1}$。

(20)$s_{4\text{-}11}\rightarrow s_{5\text{-}2}\rightarrow s_{6\text{-}2}$，$s_{4\text{-}12}\rightarrow s_{5\text{-}3}\rightarrow s_{6\text{-}3}$，$s_{4\text{-}15}\rightarrow s_{5\text{-}4}\rightarrow s_{6\text{-}4}$ 和 $s_{4\text{-}17}\rightarrow s_{5\text{-}5}\rightarrow s_{6\text{-}5}$ 的状态转换过程的操作函数与 $s_{4\text{-}10}\rightarrow s_{5\text{-}1}\rightarrow s_{6\text{-}1}$ 相同，但操作函数的参数(如属性值)可以不同。

(21)在状态 $s_{4\text{-}4}$ 下，当主体 s 完成了使用，UR'_3 中的更新行动被执行，并且系统状态转换到 $s_{7\text{-}1}$。

(22)在状态 $s_{7\text{-}1}$ 下，主体 s 发出的命令——行动 $endaccess(s,o,r)$ 得以执行，系统状态转换到 $s_{8\text{-}1}$，且 $state(s,o,r)=end$。

(23)在 $s_{8\text{-}1}$ 中，UR'_{10} 中的更新操作被执行，并且系统状态转换到 $s_{9\text{-}1}$。

(24)$s_{4\text{-}8}\rightarrow s_{7\text{-}2}\rightarrow s_{8\text{-}2}\rightarrow s_{9\text{-}2}$ 和 $s_{4\text{-}16}\rightarrow s_{7\text{-}3}\rightarrow s_{8\text{-}3}\rightarrow s_{9\text{-}3}$ 的状态转换过程的操作函数与 $s_{4\text{-}4}\rightarrow s_{7\text{-}1}\rightarrow s_{8\text{-}1}\rightarrow s_{9\text{-}1}$ 相同，但操作函数的参数(如属性值)可以不同。

以简单的模型校核，我们验证了上述状态转换中所有规则的正确性。也就是说，这个模型满足了逻辑规则集。因此，任何控制规则和更新规则能被至少一个 SG_UCON_C 核心模型满足。

4 应用实例分析

在 e-Learning Grid 中，为满足大量短期软件工程师培训需求，需构建一个“软件培训”虚拟组织 VO_{ST}。它需要聚集广域分布的闲散资源——课件。依据文献[12]的定义将课件封装成网格服务，它对外公开了两个基本交互机制：操作和服务数据。网格服务也提供对服务数据的操作。因此，可将网格服务的授权表示为发起者调用操作的权限。课件这类网格服

务对外提供四个基本操作：读(Read)、下载(Download)、写(Write)、上传(Upload)。这些操作的使用权限分别表示为 R,D,W,U，并分别被指派给四个基本角色：r_R，r_D，r_W，r_U。

自治域 AD_1 愿意共享基础课服务。AD_1 的资源组织片断如图 3 所示。以“应用数学 AM”为例，其课件被封装成网格服务并指派给角色 r_{AM}，而 r_{AM} 继承了角色 r_R，r_D，r_W，r_U，其余课件类似。角色 r_{MT} 继承了角色 r_{AM}，r_{AS}，r_C。

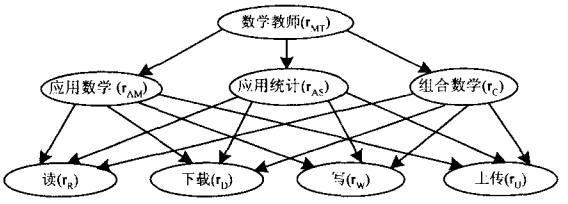

图 3　基础课角色层次示例图

自治域 AD_2 愿意共享专业课服务。AD_2 的资源组织片断如图 4 所示。以“现代软件工程 MSE”为例，其课件被封装成网格服务并指派给角色 r_{MSE}，而 r_{MSE} 继承了角色 r_R，r_D，r_W，r_U，其余课件类似。角色 r_{ST} 继承了角色 r_{MSE}，r_{SMTT}，r_{OOCT}。

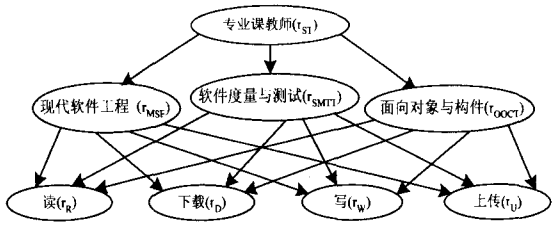

图 4　专业课角色层次示例图

VO_{ST} 根据自己的目标与 AD_1 协商获得“应用数学 AM 和“应用统计 AS”的“读”和“下载”权，有效期从 2007 年 7 月 1 日到 2007 年 8 月 31 日，其委托凭证为 $dc_{VO_{ST}\leftarrow AD_1}=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_1}))$，其中委托票据为 $dt_{VO_{ST}\leftarrow AD_1}=((VO_{ST},r'_{MT\_tree}),([07\text{-}07\text{-}01,07\text{-}08\text{-}31],P),n,ae,dep)$，其中 $r'_{MT\_tree}=(r_{MT}(r_{AM}(r_R,r_D),r_{AS}(r_R,r_D)))$，$pt=([07\text{-}07\text{-}01,07\text{-}08\text{-}31],p)$，$pt$ 的详细解释参见文献[8]。r'_{MT_tree} 表示由图 3 中完整的角色树 r_{MT_tree} 剪枝得到的子角色树。

VO_{ST} 根据自己的目标与 AD_2 协商获得“现代软件工程 MSE、“软件度量与测试 SMTT”和“面向对象与构件 OOCT”的“读”和“下载”权，有效期同前，其委托凭证为 $dc_{VO_{ST}\leftarrow AD_2}=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_2}))$，其中委托票据为 $dt_{VO_{ST}\leftarrow AD_2}=((VO_{ST},r'_{ST\_tree}),([07\text{-}07\text{-}01,07\text{-}08\text{-}31],P),n,ae,dep)$，其中 $r'_{ST\_tree}=(r_{ST}(r_{MSE}(r_R,r_D),r_{SMTT}(r_R,r_D),r_{OOCT}(r_R,r_D)))$。$r'_{ST\_tree}$ 的解释类似 r'_{MT_tree}。

上述委托凭证中未给出具体值的参数在本例中对授权无限制。为了充分利用聚集的资源，VO_{ST} 需要制定合理的授权策略。其中一条授权策略如下：注册用户可以浏览课件；每人每周可对每门课的课件浏览 5 次，每次浏览不能超过 45min；每门课的课件被同时浏览的人数不能超过 30 人；在每天 8 时到 12 时只能浏览基础课课件；在每周 14 时到 18 时只能浏览专业课课件。

我们定义的部分属性如下：主体属性有“委托凭证 dc”、“主体类别 cid”、“浏览时间 bt”、“浏览次数 bm_o(下标表示课件)”、“开始使用时间 $start$”。cid 可取两种值——“注册用户 reg”和“注册并缴费用户 enr”；客体属性有“正在浏览的主体数量 bsn”；系统属性有“系统时钟 $clock$”。定义常量为 DD=07-01-01，UD=07-08-31。客体 o 的取值范围为 $o\in\{AM,AS,MSE,SMTT,OOCT\}$。用 SG_UCON$_C$ 策略规范表示上述授权策略如下：

(1) $permitaccess(s,o,R)\rightarrow\Delta(tryaccess(s,o,R)\wedge((s.CID=reg)\wedge(o\in\{AM,AS,MSE,SMTT,OOCT\})\wedge preupdate(s,dc))$；

$preupdate(s,dc):switch(o)$

$\{case\ o==AM:dc=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_1}(dt_{s\leftarrow VO_{ST}}))),$
$dt_{s\leftarrow VO_{ST}}=((s,(r_{AM}(r_R))),([DD,UD],P),n,ae,dep);$

$case\ o==AS:dc=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_1}dt_{s\leftarrow VO_{ST}}))),$
$dt_{s\leftarrow VO_{ST}}=((s,(r_{AS}(r_R))),([DD,UD],P),n,ae,dep);$

$case\ o==MSE:dc=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_2}(dt_{s\leftarrow VO_{ST}}))),$
$dt_{s\leftarrow VO_{ST}}=((s,(r_{MSE}(r_R))),([DD,UD],P),n,ae,dep);$

$case\ o==SMTT:dc=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_2}(dt_{s\leftarrow VO_{ST}}))),$
$dt_{s\leftarrow VO_{ST}}=((s,(r_{SMTT}(r_R))),([DD,UD],P),n,ae,dep);$

$case\ o==OOCT:dc=(n_d,n_b,(dt_{VO_{ST}\leftarrow AD_2}(dt_{s\leftarrow VO_{ST}}))),$
$dt_{s\leftarrow VO_{ST}}=((s,(r_{OOCT}(r_R))),([DD,UD],P),n,ae,dep)\}$;

(2) $grant(s,dc)\rightarrow\Delta permitaccess(s,o,R)$;

(3) $activate(s,dc)\rightarrow\Delta(grant(s,dc)\wedge(o.bsn<30)\wedge(s.bm_o<5)\wedge$
$((o\in\{AM,AS\}\wedge 8am\leqslant sys.clock\leqslant 12am)\vee$
$(o\in\{MSE,SMTT,OOCT\}\wedge 2pm\leqslant sys.clock\leqslant 6pm))$
$\wedge preupdate(o.bsn)\wedge preupdate(s.bm_o)$
$\wedge preupdate(s.bt)\wedge preupdate(s.start)),$
$preupdate(o.bsn):o.bsn=o.bsn+1,$
$preupdate(s.bm_o):s.bm_o=s.bm_o+1,$
$preupdate(s.bt):s.bt=0,$
$preupdate(s.start:s.start=sys.clock;$

(4) $\Box((state(s,dc)=u\ \sin\ g\_dc)\wedge(s.bt<45)\wedge$
$((o\in\{AM,AS\}\wedge 8am\leqslant sys.clock\leqslant 12am)\vee$
$(o\in\{MSE,SMTT,OOCT\}\wedge 2pm\leqslant sys.clock\leqslant 6pm))$
$\rightarrow onupdate(s.bt)),$
$onupdate(s.bt):s.bt=sys.clock-s.start;$

(5) $\Box(\neg(o\in\{AM,AS)\wedge 8am\leqslant sys.clcok\leqslant 12am)\vee$
$(o\in\{MSE,SMTT,OOCT\}\wedge 2pm\leqslant sys.clock\leqslant 6pm))$
$\wedge(state(s,dc)=u\ \sin\ g\_dc)\rightarrow inactivate(s,dc))$;

(6) $\Box((s.bm_o\leqslant 4)\wedge(s.bt>45)\wedge(state(s,dc)=u\ \sin\ g\_dc)\rightarrow inactivate(s,dc))$;

(7) $inactivate(s,dc)\rightarrow\Diamond postupdate(o.bsn),$
$postupdate(o.bsn):o.bsn=o.bsn-1;$

(8) $\Box((s.bm_o>4)\wedge(s.bt>45)\wedge(state(s,dc)=u\ \sin\ g\_dc)\rightarrow hold(s,dc))$;

(9) $hold(s,dc)\rightarrow\Diamond postupdate(o.bsn)$;

(10) $restore(s,dc)\rightarrow\Delta(hold(s,dc)\wedge(s.bm_o==0)$

$\wedge((o\in\{AM,AS\}\wedge 8am\leqslant sys.clock\leqslant 12am)\vee(o\in\{MSE,SMTT,OOCT\}\wedge 2pm\leqslant sys.clock\leqslant 6pm))$

$\wedge preupdate(o.bsn)\wedge preupdate(s.bn_o)\wedge preupdate(s.bt)\wedge preupdate(s.start))$;

(11)$\square((sys.clock>UD)\wedge(state(s,dc)=u\ \sin\ g\_dc)\rightarrow revoke(s,dc))$;

(12)$\square((sys.clock>UD)\wedge(state(s,dc)=grant\_dc)\rightarrow revoke(s,dc))$;

(13)$\square((sys.clock>UD)\wedge(state(s,dc)=hold\_dc)\rightarrow revoke(s,dc))$;

(14)$revokeaccess(s,o,r)\rightarrow\Delta revoke(s,dc)$;

(15)$activate(s,dc)\rightarrow\Diamond endaccess(s,o,r)$;

(16)$revokeaccess(s,o,r)\vee endaccess(s,o,r)\rightarrow\Diamond postupdate(s.dc),postupdate(s.dc);s.dc=null$

在上述形式化授权策略中，bn_o 和 bsn 的初始化工作由系统自动完成，故本文未给出形式化说明。在 DD=07-01-01 时，bn_o 被初始化为 0，以后每 7 天都要重新初始化为 0，但 bsn 仅在 DD=07-01-01 时被初始化为 0。

我们以主体 s 发出浏览客体 $o=MSE$ 的请求为例，展示其决策的主要过程：当与(1)匹配时，若主体 s 已注册，且(1)中的其它谓词也为真，则使用前更新行动执行，相应的委托凭证产生；当与(2)匹配时，由于(1)匹配成功知 $permitaccess(s,o,R)$ 为真，故 $grant(s,dc)$ 为真，主体 s 获得委托凭证 dc；当与(3)匹配时，若客体 MSE 的在线浏览人数 $MSE.bsn$ 小于 30，主体 s 对 MSE 的周使用次数 $s.bn_{MSE}$ 小于 5，而且在 8 时到 12 时期间请求浏览的是基础课课件或在 14 时到 18 时期间请求浏览的是专业课课件，则执行使用前更新行动将 $MSE.bsn$ 和 $s.bn_{MSE}$ 各增 1，将 $s.bt$ 初始化为 0，并将开始使用时间 $s.start$ 设置为当前系统时钟值。然后主体 s 的委托凭证 dc 被激活并处于使用中；当与(4)匹配时，若主体 s 的浏览时间 $s.bt$ 未超过 45 分钟并且处于规定的浏览时间段，则执行使用中更新 $onupdate(s.bt)$ 将 $s.bt$ 的值更新为当前系统时钟值与开始使用时间 $s.start$ 的差值；当与(5)匹配时，若主体 s 的浏览时间未处于规定的浏览时间段，则去活委托凭证 dc；当与(6)匹配时，若 $s.bn_{MSE}$ 的指标未用完且本次使用时间超时，则去活委托凭证 dc；若(5)或(6)匹配，则在(7)匹配时将 $MSE.bsn$ 减 1；当与(8)(9)匹配时，若 $s.bn_{MSE}$ 的指标用完且本次使用时间超时，则挂起委托凭证 dc 并相应地将 $MSE.bsn$ 减 1；当与(10)匹配时，若 $s.bn_{MSE}$ 被系统重新初始化为 0 并且主体 s 的访问请求处于规定的浏览时间段，则将挂起的委托凭证 dc 恢复为激活状态并做(3)中同样的更新；若有效使用期过时，则由(11)(12)(13)都能将委托凭证 dc 转换到撤销状态并由(14)撤销相应权限；若在使用中，主体 s 完成了应用，则主动执行 $endaccess(s,o,r)$；无论 $revokeaccess(s,o,r)$ 还是 $endaccess(s,o,r)$ 为真，都将导致系统按(16)进行使用后更新，即将委托凭证 dc 所占用的系统资源释放。

在整个有效使用期，委托凭证只在主体第一次请求时产生，除非主体主动撤销，否则要到有效使用期满才会销毁。在委托凭证销毁前，它可依据系统状态的变化而转换到不同的凭证处理状态。

结束语 $UCON_{ABC}$ 被称为下一代访问控制技术，但现有的 $UCON_{ABC}$ 策略规范不能适应服务网格授权的新特点。本文针对 $UCON_C$ 进行改进与扩充，给出了一套基于条件谓词决策的形式化策略规范，并对策略规范的完备性和正确性进行了证明。通过实例分析展示了改进的策略规范的表达能力和访问请求的主要决策过程。针对基于 $UCON_A$ 和 $UCON_B$ 的策略规范的改进与扩充，我们将另文详述。

参 考 文 献

[1] Park J, Sandhu R. Towards Usage Control Models: Beyond Traditional Access Control// Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT02). Monterey, California, USA: ACM, 2002: 57-64

[2] Park J, Sandhu R. The UCONABC usage control model. ACM Transaction on Information and System Security, 2004, 7(1): 128-174

[3] Zhang Xinwen, Parisi-Presicce F, Sandhu R, et al. Formal model and policy specification of usage control. ACM Transactions on Information and System Security(TISSEC), 2005, 8(4): 351-387

[4] Pearlman L, Welch V, Foster I, et al. A community authorization service for group collaboration// IEEE 3rd International Workshop on Policies for Distributed Systems and Networks. Washington D. C: IEEE Computer Society, 2002: 50-59

[5] Thompson M, Essiari A, Mudumbai S. Certificate based Authorization Policy in a PKI Environment. ACM Transactions on Information and System Security (TISSEC), 2003, 6(4): 566-588

[6] Foster I, Jennings N R, Kesselman C. Brain meets brawn: Why grid and agents need each other// Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems(AAMAS'04). New York, USA, 2004: 8-15

[7] Ferraiolo D F, Sandhu R, Gavrila S. Proposed NIST standard for role-based access control. ACM Transaction on Information and System Security, 2001, 4(3): 224-274

[8] 徐震，李斓，冯登国. 基于角色的受限委托模型. 软件学报，2005, 16(5): 970-978

[9] Strembeck M. Conflict checking of separation of duty constraints in RBAC - implementation experiences// Proceedings of the Conference on Software Engineering (SE'04). Innsbruck, Austria, 2004: 224-229

[10] 翟征德. 基于量化角色的可控委托模型. 计算机学报，2006, 29(8): 1401-1407

[11] Zhang X W, Oh S, Sandhu R S. PBDM: A flexible delegation model in RBAC// Ferrari E, Ferraiolo D, eds. Proc. of the 8th ACM Symp. on Access Control Models and Technologies. New York: ACM Press, 2003: 149-157

[12] Tuecke, et al. Open Grid Service Infrastructure, Version 1.0. http://www.ggf.org/ogsi-wg. June 2003