计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
42(11):1946~1953, 2005

基于角色和任务的工作流授权模型及约束描述

邢光林　　洪　帆
(华中科技大学计算机科学与技术学院　武汉　430074)
(xgldaj@yahoo.com.cn)

A Workflow Authorization Model Based on Role and Task and Constraints Specification

Xing Guanglin and Hong Fan
(*College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract　A workflow authorization model based on role and task is first described. The basic idea of this model is that roles and permissions are not connected directly but are put together by tasks. This is more convenient for controlling and managing the granularity of permissions. And then an intuitive formal language called RTCL is proposed, which takes the model as context to specify workflow authorization constraints based on role and task. RTCL uses system functions, sets and variable symbols as its basic elements and is proved to be equivalent to a restricted form of first order predicate logic called RFOPL on semantics. Finally, the expressive power of RTCL is demonstrated by showing how it can be used to express a variety of constraints.

Key words　role and task; workflow; authorization model; constraint specification

摘　要　首先描述了一个基于角色和任务的工作流授权模型,其基本思想是角色和权限不直接挂钩而是通过任务把它们联系在一起,更方便权限粒度的控制和管理,然后以此模型为上下文背景提出了一个描述基于角色和任务的工作流授权约束的直观的形式化语言,称为 RTCL. 它以系统函数、集合以及变量符作为基本元素,证明了在语义上 RTCL 与严格形式的一阶谓词逻辑 RFOPL 是等价的. 最后通过用 RTCL 表示各种各样的约束来说明 RTCL 的表现能力.

关键词　角色和任务;工作流;授权模型;约束描述

中图法分类号　TP309.2

1　引　　言

工作流中的活动通常划分为一些预先定义好的任务[1]. 这些任务相互依赖以一种协调的方式执行,但这些任务并不是任何人都能执行,只有经过授权的用户才能执行. 为了保证任务只能被合法的用户执行,就应该有一个合适的授权机制:首先,应当保证授权只有在任务开始执行时才授予且任务一结束就收回,即授权流尽可能与工作流同步,否则用户拥有权限的时间就会长于他需要权限的时间;其次,应当保证合法用户在执行某个任务实例时只能拥有该任务实例所允许的权限,即最小特权,否则该用户拥有的权限就会比他所需要的权限大;再次,为了防止欺骗及维护工作的完整性,还应该保证执行完某个任务的用户不能再执行其他某个任务,或是保证执行某个任务的用户必须是执行前面某个任务的同一用户,即职责分离[2].

收稿日期:2004-05-12;修回日期:2005-01-27

为此提出了一个基于角色和任务的工作流授权模型,同 RBAC96 模型[3]相比,其基本思想是:角色和权限不直接挂钩而是通过任务联系在一起,给用户指派合适的角色,用户通过其指派的角色获得可以执行的任务,在执行任务的某个具体实例时获得该任务所允许访问的客体的权限,这样更方便权限粒度的控制和管理. 授权约束是工作流授权中的一个非常重要的部分,已经有方法对授权约束进行形式化的描述[4~8]. 在基于角色和任务的工作流授权模型的基础上提出了一个约束描述语言 RTCL(role and task based constraint language). 该描述语言沿用了 RCL2000[4]的思想,但同 RCL2000 相比,有以下几点重要区别:第 1,RCL2000 的基本元素源自 RBAC96 模型,它不完全适合于工作流授权约束的描述,而 RTCL 则完全适合于工作流授权约束的描述. 第 2,RCL2000 在语义上与严格形式的一阶谓词逻辑是不能划等号的,有些 RFOPL 表达式就不能转换成 RCL2000 表达式,这主要是 RCL2000 的两个不确定函数的局限性造成的. RTCL 对 RCL2000 的两个不确定函数进行了扩展,从而保证 RTCL 在语义上与严格形式的一阶谓词逻辑是等价的. 第 3,RCL2000 没有考虑时态约束,而 RTCL 加入了时态约束. 第 4,RTCL 的表现能力比 RCL2000 更强,并更正了 RCL2000 中一个不太准确的静态职责分离约束 SSOD-CU,它表示的是两个相互冲突的用户不能指派给同一个冲突角色集中的角色,该约束排除了两个相互冲突的用户可以指派给同一个冲突角色集中的同一个角色的可能性.

2 基于角色和任务的工作流授权模型

2.1 基本术语定义

基于角色和任务的工作流授权模型,其主要组成部件如图 1 所示:

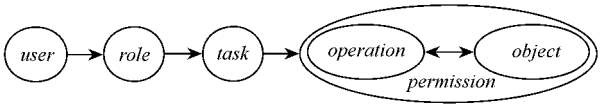

Fig. 1 The sketch diagram of workflow authorization model based on role and task.

图 1 基于角色和任务的工作流授权模型简图

一个工作流由一些预先定义好的任务组成,令 WT 表示任务集. 当一个任务执行一次时就会产生一个任务实例,所以一个任务可以产生多个任务实例,令 TI 表示任务实例集.

定义 1. 任务映射. 令 $\Im_{WT}: TI \rightarrow WT$ 表示一个任务映射,它将每个任务实例映射到相应的任务,使得对于一个给定的任务实例 wt_i^k,如果有 $\Im_{WT}(wt_i^k) = wt_i$,那么 wt_i^k 就是 wt_i 的实例.

定义 2. 实例映射. 令 $\Im_{TI}: WI \rightarrow 2^{TI}$ 表示一个实例映射,它将每个任务映射到相应的任务实例集,使得如果 $wt_i \neq wt_j$ 且 $wt_i, wt_j \in WT$,则 $\Im_{TI}(wt_i) \cap \Im_{TI}(wt_j) = \varnothing$.

定义 3. 时间集. 令二元组 $(TS; \leqslant)$ 是一个时态域,其中 $TS = \{t \in \mathbb{R} \mid t \geqslant 0\}$ 是一个时间集,$\leqslant$ 表示 TS 上的全序,$\mathbb{R}$ 表示实数集.

定义 4. 令 $[t_s, t_e]$ 表示一个时间区间,其中 t_s, $t_e \in TS$ 且 $t_s \leqslant t_e$,令 $TR = \{[t_s, t_e] \in TS \times TS \mid t_s \leqslant t_e\}$ 表示所有时间区间的集合.

如果说区间 $[t_1, t_2]$ 在区间 $[t_3, t_4]$ 中,当且仅当 $t_3 \leqslant t_1$ 且 $t_4 \geqslant t_2$,同样,说一个时间点 t_1 在区间段 $[t_3, t_4]$ 中,当且仅当 $t_3 \leqslant t_1 \leqslant t_4$.

定义 5. 任务执行时间模板. 给定一个任务 wt,令 $TT(wt) = (wt, [t_l, t_u])$ 为 wt 的执行时间模板,表示 wt 只能在时间区间 $[t_l, t_u]$ 内执行. 令 $TT(WT)$ 表示任务执行时间模板集合.

2.2 模型部件描述

基于角色和任务的工作流授权模型由如下的部件组成:

(1) U(用户集)、R(角色集)、WT(任务集)、TI(任务实例集)、OP(操作集)、OBJ(客体集)、P(权限集)、S(会话集)、$TT(WT)$(任务执行时间模板集).

(2) $UA \subseteq U \times R$(用户角色指派关系)、$RH \subseteq R \times R$(角色层次关系)、$RTA \subseteq R \times WT$(角色任务指派关系)、$PTA \subseteq P \times WT$(权限任务指派关系).

(3) $\Im_{WT}: TI \rightarrow WT$ 是一个任务映射函数,$\Im_{TI}: WT \rightarrow 2^{TI}$ 是一个实例映射函数.

(4) $user: R \rightarrow 2^U$ 是一个用户映射函数,它把每个角色 r_i 映射到一个用户集,

$$user(r_i) = \{u \in U \mid (u, r_i) \in UA\}.$$

(5) $role: U \cup WT \rightarrow 2^R$ 是一个角色映射函数,它把集合 U,WT 映射到一个角色集,$RH\_role: U \cup WT \rightarrow 2^R$ 是在存在角色层次的情况下对函数 $role$ 的扩展,

$$role(wt_i) = \{r \in R \mid (r, wt_i) \in RTA\},$$

$$role(u_i) = \{r \in R \mid (u_i, r) \in UA\},$$

$$RH\_role(u_i) = \{r \in R \mid (\exists r' \geqslant r)[(u_i, r') \in UA]\},$$

$$RH\_role(wt_i) = \{r \in R \mid (\exists r' \geqslant r)[(r', wt_i) \in RTA]\}$$

(6) $permission: WT \rightarrow 2^P$ 是一个权限映射函数,它把每个任务 wt_i 映射到一个权限集,

$$permission(wt_i) = \{p \in P \mid (p, wt_i) \in PTA\}.$$

(7) $task: R \cup P \rightarrow 2^{WT}$ 是一个任务映射函数,它把集合 R,P 映射到一个任务集,

$$task(r_i) = \{wt \in WT \mid (r_i, wt) \in RTA\},$$

$$task(p_i) = \{wt \in WT \mid (p_i, wt) \in PTA\}.$$

2.3 授权规则

定义 6. 授权. 一个授权可定义为一个三元组 $az = (u, wt_i^k, [t_{b_i}^k, t_{f_i}^k])$,表示用户 u 在时间点 $t_{b_i}^k$ 开始执行任务 wt_i 的第 k 次实例,在时间点 $t_{f_i}^k$ 结束对任务实例的执行.

随着工作流的执行,相应会产生很多的授权,把所有这些授权称做一个授权基,记为 $AZ = \{az_1, az_2, \cdots\}$.

定义 7. 授权非时态投影. 给定一个授权 $az = (u, wt_i^k, [t_{b_i}^k, t_{f_i}^k])$,定义 az 的非时态投影为 $az_{NT} = (u, wt_i^k)$,授权基 AZ 的非时态投影记为 $AZ_{NT} = \{az_{NT1}, az_{NT2}, \cdots\}$.

作为一个安全原则,职责分离是一个根本技巧,职责分离的主要目的是通过把责任和权限分配给不同的多个用户,要求这些用户合作完成敏感任务以减少欺诈行为或重要错误的可能性. 职责分离可以表示成为一系列约束规则[9].

假定每个约束 c_i 都是形如 $q \leftarrow p$ 的逻辑表达式,其中 p 是包含非时态投影 az_{NT} 的任何逻辑表达式,q 是一个单一原子,要么是 $must\_do(u, wt_i^k)$,要么是 $cannot\_do(u, wt_i^k)$,用 $U(p)$ 表示 p 中所描述的用户. 实施约束要么是强制要么是阻止把某个具体的用户指派给某个任务,故约束可分为两种类型:排他型,q 总是 $cannot\_do(u, wt_i^k)$ 的形式;肯定型,q 总是 $must\_do(u, wt_i^k)$ 的形式. 这样一来,有资格执行每个任务的用户集合会根据当前的授权集合状态自动改变. 而且,不同的任务实例,其有资格的用户集合也可能是不同的.

只有与某个任务相关的约束规则才能起到决定有资格执行该任务的用户集合,下面给出与任务 wt_i 相关的约束集合.

定义 8. 令 $C_{wt_i^k}$ 表示与任务实例 wt_i^k 相关的约束集合,其中每个约束 c_j^k 可表示为 $q_i^k \leftarrow p_j^k$ 的形式.

定义 9. 给定约束集合 $C_{wt_i^k}$,有资格执行任务实例 wt_i^k 的用户集合 $EU(wt_i^k)$ 定义为:

(1) $EU(wt_i^k) = user(RH\_role(\Im_{WT}(wt_i^k)))$,如果 $C_{wt_i^k} = \varnothing$;

(2) $EU(wt_i^k) = user(RH\_role(\Im_{WT}(wt_i^k))) - u(p_j^k)$,如果 $c_j^k: cannot\_do(u, wt_i^k) \leftarrow p_j^k \in C_{wt_i^k}$ 且 p_j^k 关于 AZ_{NT} 为真;

(3) $EU(wt_i^k) = u(p_j^k)$,如果 $c_j^k: must\_do(u, wt_i^k) \leftarrow p_j^k \in C_{wt_i^k}$ 且 p_j^k 关于 AZ_{NT} 为真.

一个真正的授权是在任务开始执行时生成的,授权生成规则的形式化定义如下:

定义 10. 授权规则. 给定任务 wt_i 的执行时间区间模板 $TT(wt_i) = (wt_i, [t_{l_i}, t_{u_i}])$,一个授权 $az = (u, wt_i^k, [t_{b_i}^k, t_{f_i}^k])$ 可按如下的方式生成:

授予规则:假定用户 u_i 在时间点 t_{a_i} 启动任务实例 wt_i^j,如果 $\Im(wt_i^j) = wt_i$,$u_i \in EU(wt_i^j)$,$t_{a_i} \leqslant t_{u_i}$,则 $u \leftarrow u_i$,$wt_i^k \leftarrow wt_i^j$,$t_{f_i}^k \leftarrow t_{u_i}$;若还有 $t_{a_i} \leqslant t_{l_i}$,则 $t_{b_i}^k \leftarrow t_{l_i}$;否则,$t_{b_i}^k \leftarrow t_{a_i}$.

回收规则:假定任务实例 wt_i^j 在时间点 t_{f_i} 结束,如果 $t_{f_i} \leqslant t_{u_i}$,则 $t_{f_i}^k \leftarrow t_{f_i}$.

2.4 实 例

有一个发文工作流由 5 个任务组成:拟稿、审稿、核稿、稿件签发和校稿. 假定用户为 5 人:张三,李四,王五,赵六,陈七. 角色分为 3 类:职员、科长、处长,其层次关系为处长支配科长,科长支配职员. 5 个用户的角色分别是张三和李四是职员,王五和赵六是科长,陈七是处长. 系统的安全策略为①职员只能进行拟稿和校稿;②审稿和核稿只能由科长或处长来处理;③稿件签发必须由处长来完成;④执行校稿的用户必须是执行拟稿的用户;⑤执行核稿的用户不能是执行审稿的用户. 图 2 即为上述发文工作流的任务图.

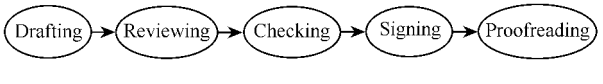

Fig. 2 The task diagram of dispatch workflow.

图 2 发文工作流任务图

上述工作流用基于角色和任务的工作流授权模型描述如下:

$WT = \{wt_1, wt_2, wt_3, wt_4, wt_5\}$,$wt_1$,$wt_2$,$wt_3$,$wt_4$,$wt_5$ 分别是拟稿、审稿、核稿、稿件签发和

校稿. $TT(WT)=\{(wt_1,[10,40]),(wt_2,[20,50]),(wt_3,[30,60]),(wt_4,[40,70]),(wt_5,[50,80])\}$,$U=\{u_1,u_2,u_3,u_4,u_5\}$,$u_1,u_2,u_3,u_4,u_5$ 分别是张三、李四、王五、赵六和陈七,$R=\{r_1=$职员,$r_2=$科长,$r_3=$处长$\}$,$P=\{$(准备,稿件),(审查,稿件),(核对,稿件),(签发,稿件),(校对,稿件)$\}$,根据系统假设,用户指派关系为 $UA=\{(u_1,r_1),(u_2,r_1),(u_3,r_2),(u_4,r_2),(u_5,r_3)\}$,$RH=\{r_3\geqslant r_2\geqslant r_1\}$,角色任务指派关系为 $RTA=\{(r_1,wt_1),(r_1,wt_5),(r_2,wt_2),(r_2,wt_3),(r_3,wt_4)\}$,权限任务指派关系为 $PTA=\{(wt_1,p_1),(wt_2,p_2),(wt_3,p_3),(wt_4,p_4),(wt_5,p_5)\}$,系统的安全策略 4 属于肯定型约束,描述为 c_1:$must\_do(u,wt_5)\leftarrow(u,wt_1)\in C_{wt_5}$,系统的安全策略⑤属于排他型约束,描述为 c_2:$connot\_do(u,wt_3)\leftarrow(u,wt_2)\in C_{wt_3}$. 下面是该工作流的一个实例执行过程:

由于 $C_{wt_1^1}=\varnothing$,所以 $EU(wt_1^1)=user(RH\_role(\Im_{TI}(wt_1^1)))=\{u_1,u_2,u_3,u_4,u_5\}$,假设 u_1 在时间点 30 开始拟稿 wt_1^1,因为 u_1 属于 $EU(wt_1^1)$,$TT(wt_1)=(wt_1,[10,40])$ 且 $30\leqslant 40$,所以产生授权 $(u_1,wt_1^1,[30,40])$,假定 u_1 在 37 时完成拟稿,则授权收回并将上边界 40 用 37 代替,形成授权 $(u_1,wt_1^1,[30,37])$. 类似地,在 wt_2^1 执行完后产生授权 $(u_3,wt_2^1,[37,45])$. 由于 $C_{wt_3^1}=\{c_2\}$ 且 c_2 是一个排他型约束,所以 $EU(wt_3^1)=user(RH\_role(\Im_{TI}(wt_3^1)))-U(u,wt_2^1)=\{u_3,u_4,u_5\}-u_3=\{u_4,u_5\}$,$TT(wt_3)=(wt_3,[30,60])$,假定是 u_4 进行核稿 wt_3^1,最后形成授权 $(u_4,wt_3^1,[45,53])$. 由于 $C_{wt_5^1}=\{c_1\}$ 且 c_1 是一个肯定型约束,所以 $EU(wt_5^1)=U(u,wt_1^1)=\{u_1\}$,换句话说,只有 u_1 才允许执行校稿任务 wt_5^1,$TT(wt_5)=(wt_5,[50,80])$,最后产生的授权为 $(u_1,wt_5^1,[65,72])$.

3 基于角色和任务的约束描述语言

本节在上节模型的框架上描述基于角色和任务的约束描述语言 RTCL,RTCL 约束描述语言可以看成 RCL2000[4] 的扩展,首先介绍 RTCL 所使用的基本元素和系统函数.

3.1 基本元素和系统函数

RTCL 使用的基本元素和系统函数的定义如上节所示,图 3 展示了这些定义的上下文背景. RTCL 主要有 9 个实体集,分别为用户(U)、角色(R)、任务(WT)、任务实例(TI)、权限(P)、操作(OP)、客体(OBJ)、会话(S)和授权(AZ).

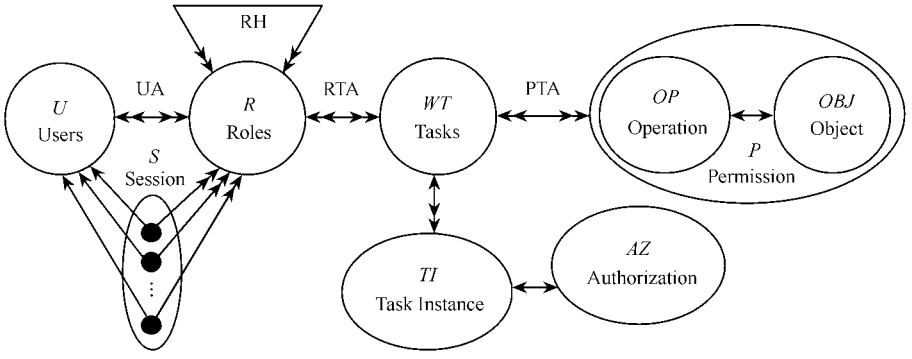

Fig. 3 Basic elements and system functions from workflow authorization model based on role and task.

图 3 源自基于角色和任务的工作流授权模型的基本元素和系统函数

RTCL 用到的其他元素和系统函数定义如下:

- $CR=\{cr_1,\cdots,cr_m\}$ 为冲突角色集,其中 $cr_i\subseteq R$;
- $CP=\{cp_1,\cdots,cp_s\}$ 为冲突权限集,其中 $cp_i\subseteq P$;
- $CU=\{cu_1,\cdots,cu_u\}$ 为冲突用户集,其中 $cu_i\subseteq U$;
- $CT=\{ct_1,\cdots,ct_o\}$ 为冲突任务集,其中 $ct_i\subseteq WT$;
- $v_x=$隶属于实体集 X 的变量符;
- $OE(v_X,X)=x_i$,其中 $x_i\in X$,是变量符 v_X 的值;
- $AO(v_X,X)=X-\{OE(v_x,X)\}$;
- is_in:$TS\times TR\rightarrow\{\text{true},\text{false}\}$,$is\_in(t,[t_s,t_e])=(t_s\leqslant t)\wedge(t\leqslant t_e)$;
- $authorization$:$TI\rightarrow AZ$ 是一个授权映射函数,把每个工作流实例映射到相应的授权;
- $az\_starttime$:$AZ\rightarrow TS$,$az\_starttime(az)=$

t_b,取每个授权的开始时间;

$-az\_endtime: AZ \rightarrow TS, az\_endtime(az) = t_f$,取每个授权的结束时间;

$-getregion: TT(WT) \rightarrow TR, getregion(TT(wt)) = [t_s, t_e]$,取每个任务的执行时间模板区间.

$OE(v_X, X)$和 $AO(v_X, X)$是两个不定函数. $OE(v_X, X)$表示从实体集 X 中任取一个元素,v_X 是该元素的变量符表示. 如果两个 OE 函数中的变量符不同,则这两个 OE 函数不同. 单一 RTCL 表达式中多处出现相同的 OE 函数其取值是相同的. $AO(v_X, X)$表示从实体集 X 中取出除去某个元素后的所有元素,v_X 是该除去元素的变量符表示. $OE(v_X, X)$和 $AO(v_X, X)$是上下文相关的,如对某个实体集合 X,$\{OE(v_X, X)\} \cup \{AO(v_X, X)\} = X$,但 $OE(v_X, X)$与 $AO(v_X, X)$都不是确定函数.

下面来看看如何用 RTCL 描述基于角色和任务的约束,以静态职责分离约束中的冲突用户约束为例. 约束要求为"两个相互冲突的用户不能指派给同一个冲突角色集中的不同的角色",该约束的 RTCL 表达式如下:

$$|RH\_role(OE(u, U)) \cap OE(cr, CR)| \leqslant 1 \wedge OE(r, OE(cr, CR)) \in RH\_role(OE(u_1, OE(cu, CU))) \Rightarrow RH\_role(OE(u_2, OE(cu, CU))) \cap AO(r, OE(cr, CR)) = \varnothing.$$

在上述表达式中,$OE(cr, CR)$表示某个冲突角色集,$RH_role(OE(u, U))$表示指派给某个用户的所有角色,故 $|RH\_role(OE(u, U)) \cap OE(cr, CR)| \leqslant 1$ 表示一个用户不能同时拥有两个或两个以上的相互冲突的角色,余下的部分表示的是,如果把某个冲突用户集中一个用户指派给了某个冲突角色集中的一个角色,那么就不能把该冲突用户集中的其他用户指派给同一冲突角色集中的其他角色.

3.2 RTCL 的形式化语义

本小节通过识别一个与 RTCL 等价的称之为 RFOPL 的严格形式的一阶谓词逻辑来讨论 RTCL 的形式化语义. 用 RTCL 表示的任何约束称之为 RTCL 表达式,可转换成一个 RFOPL 表达式,反之亦然. 把 RTCL 表达式转换成 RFOPL 表达式的转换算法称做归约算法,如下所示:

算法 1. 归约算法.

输入:RTCL 表达式;

输出:RFOPL 表达式.

(1) 消除 AO 函数

在所有出现 $AO(v_{expr}, expr)$的地方都用 $(expr - \{OE(v_{expr}, expr)\})$替换;

(2) 消除 OE 函数

当 RTCL 表达式中有 OE 函数

选择 OE 函数;

调用归约程序;

结束.

归约程序:

情况 1. OE 函数是 $OE(v_{SET}, SET)$的形式:

把 $\forall v_{SET} \in SET$ 放在已存在量词的右边;

用 V_{SET}替换掉所有的 $OE(v_{SET}, SET)$;

情况 2. OE 函数是 $OE(v_{function}, funcion(v_{SET}))$的形式:

把 $\forall v_{function} \in function(v_{SET})$放在已存在量词的右边;

用 $v_{function}$替换所有的 $OE(v_{function}, funcion(v_{SET}))$;

结束.

归约算法首先从 RTCL 表达式中消除 AO 函数,然后从左到右反复地把 OE 函数转换成全称量词. 当 RTCL 表达式中有嵌套的 OE 函数时,转换从最里面的 OE 函数开始. 例如,下面的 RTCL 表达式就可以按照以下的顺序转换成一个 RFOPL 表达式.

例 1. RTCL 表达式如下:

$$OE(r, OE(cr, CR)) \in RH\_role(OE(u, U)) \Rightarrow AO(r, OE(cr, CR)) \cap RH\_role(OE(u, U)) = \varnothing.$$

RFOPL 表达式为

(1) $OE(r, OE(cr, CR)) \in RH\_role(OE(u, U)) \Rightarrow (OE(cr, CR) - \{OE(r, OE(cr, CR))\}) \cap RH\_role(OE(u, U)) = \varnothing$;

(2) $\forall cr \in CR: OE(r, cr) \in RH\_role(OE(u, U)) \Rightarrow (cr - \{OE(r, cr)\}) \cap RH\_role(OE(u, U)) = \varnothing$;

(3) $\forall cr \in CR, \forall r \in cr: r \in RH\_role(OE(u, U)) \Rightarrow (cr - \{r\}) \cap RH\_role(OE(u, U)) = \varnothing$;

(4) $\forall cr \in CR, \forall r \in cr, \forall u \in U: r \in RH\_role(u) \Rightarrow (cr - \{r\}) \cap RH\_role(u) = \varnothing$.

把 RFOPL 表达式转换成 RTCL 表达式的转换算法称做构造算法,如下所示:

算法 2. 构造算法.

输入:RFOPL 表达式;

输出:RTCL 表达式.

(1) 用 RFOPL 表达式构造 RTCL 表达式

当 RFOPL 表达式中有全称量词时

选择最右边的全称量词 $\forall x \in X$;

用 $OE(x,X)$ 替换谓词部分的所有的 x;

结束.

(2) 复位 AO 函数

如果在 RFOPL 表达式中有 $(expr - \{OE(v_{\text{expr}}, expr)\})$

用 $AO(v_{\text{expr}}, expr)$ 替换 $(expr - \{OE(v_{\text{expr}}, expr)\})$;

结束.

构造算法首先反复地选择 RFOPL 表达式中最右边的全称量词,通过消除谓词部分中该量词中的变量来构造 OE 函数,当所有的全称量词都消除后,再根据 AO 函数的定义来构造 AO 函数.

由于 RCL2000 中的两个不确定函数的局限性,造成了从 RCL2000 表达式转换到 RFOPL 表达式没有问题,但有些 RFOPL 表达式却转换不成 RCL2000 表达式. 例如下面的 RFOPL 表达式用 RCL2000 的构造算法就不能转换成 RCL2000 表达式,但用经过扩展的 RTCL 构造算法就可以.

例 2.

RFOPL 表达式:

$\forall cr \in CR, \forall r \in cr, \forall cu \in CU, \forall u_1 \in cu, \forall u_2 \in cu : r \in RH\_role(u_1) \Rightarrow RH\_role(u_2) \cap (cr - \{r\}) = \varnothing$.

RTCL 表达式:

(1) $\forall cr \in CR, \forall r \in cr, \forall cu \in CU, \forall u_1 \in cu : r \in RH\_role(u_1) \Rightarrow RH\_role(OE(u_2, cu)) \cap (cr - \{r\}) = \varnothing$;

(2) $\forall cr \in CR, \forall r \in cr, \forall cu \in CU : r \in RH\_role(OE(u_1, cu)) \Rightarrow RH\_role(OE(u_2, cu)) \cap (cr - \{r\}) = \varnothing$;

(3) $\forall cr \in CR, \forall r \in cr : r \in RH\_role(OE(u_1, OE(cr, CR))) \Rightarrow RH\_role(OE(u_2, OE(cu, CU))) \cap (cr - \{r\}) = \varnothing$;

(4) $\forall cr \in CR : OE(r, cr) \in RH\_role(OE(u_1, OE(cu, CU))) \Rightarrow RH\_role(OE(u_2, OE(cu, CU))) \cap (cr - \{OE(r, cr)\}) = \varnothing$;

(5) $OE(r, OE(cr, CR)) \in RH\_role(OE(u_1, OE(cu, CU))) \Rightarrow RH\_role(OE(u_2, OE(cu, CU))) \cap (OE(cr, CR) - \{OE(r, OE(cr, CR))\}) = \varnothing$;

(6) $OE(r, OE(cr, CR)) \in RH\_role(OE(u_1, OE(cu, CU))) \Rightarrow RH\_role(OE(u_2, OE(cu, CU))) \cap AO(r, OE(cr, CR)) = \varnothing$.

构造算法可以把一个 RFOPL 表达式构造成一个 RTCL 表达式,归约算法又可以把该 RTCL 表达式还原成一个同原 RFOPL 表达式等价的 RFOPL 表达式. 下面给出的两个定理就保证了 RTCL 表达式和 RFOPL 表达式之间这种相互转换关系的合理性和完整性.

定理 1. 合理性. 给定 RTCL 表达式 α,α 可以转换成 RFOPL 表达式 β,再由 β 可以重新构造回 α,即 $C(R(\alpha)) = \beta$. 其中 $R(rtcl\_expr)$ 表示由归约算法转换得到的 RFOPL 表达式,$C(rfopl\_expr)$ 表示由构造算法构造得到的 RTCL 表达式.

定理 2. 完整性. 给定 RFOPL 表达式 β,由 β 可以构造出 RTCL 表达式 α,再由 α 可以转换成多个 RFOPL 表达式,但任何一个由 α 转换而来的 RFOPL 表达式 β' 一定与 β 等价,即 $R(C(\beta)) = \beta'$.

上述两个定理可由构造算法和归约算法进行证明,具体过程从略.

4 RTCL 的表现能力

本节通过用 RTCL 来表示各种各样的约束来说明 RTCL 的表现能力.

4.1 时态约束

文中所讨论的时态约束主要是授权流与工作流同步的时态约束,分为两种:授权的开始时间约束和授权的结束时间约束. 表 1 列出了用 RTCL 表示的上述两种时态约束:

Table 1 Time Constraint Expressed by RTCL

表 1 RTCL 表示的时态约束

Time Constraint	RTCL Expression
Start-time Constraint of Authorization	$OE(wt, WT) \in task(RH\_role(user(OE(s, S)))) \Rightarrow is\_in(az\_starttime(authorization(\Im_{TI}(wt))), getregion(TT(wt)))$
End-time Constraint of Authorization	$OE(wt, WT) \in task(RH\_role(user(OE(s, S)))) \Rightarrow is\_in(az\_endtime(authorization(\Im_{TI}(wt))), getregion(TT(wt)))$

4.2 职责分离约束

安全工作流系统中,职责分离约束有很多种,大体可以分为三大类:静态职责分离约束(SSOD)、动态职责分离约束(DSOD)和混杂职责分离约束(HSOD).

混杂职责分离约束实际上是静态职责分离约束和动态职责分离约束的混合形式,所以下面只考虑如何用 RTCL 来表示静态职责分离约束和动态职

责分离约束.

静态职责分离约束是在工作流执行之前可进行验证的约束,表 2 列出了几种用 RTCL 表示的 SSOD 约束:

Table 2 SSOD Expressed by RTCL

表 2 RTCL 表示的静态职责分离约束

SSOD Constraint	RTCL Expression
1. SSOD-CR	$\|RH\_role(OE(u,U)) \cap OE(cr,CR)\| \leqslant 1$
2. SSOD-CT	$\|task(RH\_role(OE(u,U))) \cap OE(ct,CT)\| \leqslant 1$
3. SSOD-CP	$\|permission(task(RH\_role(OE(u,U)))) \cap OE(cp,CP)\| \leqslant 1$
4. SSOD-CU	(1) $\wedge \|user(OE(cr,CR)) \cap OE(cu,CU)\| \leqslant 1 \wedge \|user(OE(r,OE(cr,CR))) \cap OE(cu,CU)\| \geqslant 1$
5. Another variation of 4	(1) $\wedge OE(r,OE(cr,CR)) \in RH\_role(OE(u_1,OE(cu,CU))) \Rightarrow RH\_role(OE(u_2,OE(cu,CU))) \cap AO(r,OE(cr,CR)) = \varnothing$

Note: (1) Represents the first expression (SSOD-CR) in table 2.

约束 1 表示一个用户不能同时拥有两个相互冲突的角色.约束 2 表示不能同时把两个相互冲突的任务通过角色指派和用户指派而指派给同一个用户.约束 3 表示一个用户同一时间最多只能拥有一个冲突权限.约束 4 表示两个相互冲突的用户不能指派给同一个冲突角色集中的不同的角色. RCL2000 中也列出了这一约束,但它表示的是两个相互冲突的用户不能指派给同一个冲突角色集中的角色,该约束排除了两个相互冲突的用户可以指派给同一个冲突角色集中的同一个角色的可能性.

动态职责分离约束是只能在工作流执行过程中进行验证的约束,分为两种:排他型约束和同一型约束. 排他型约束是指执行某个任务的用户不能是执行其他某个任务的用户,而同一型约束是指执行某个任务的用户只能是执行其他某个任务的用户. 表 3 列出了上述两种 DSOD 约束的 RTCL 表示:

Table 3 DSOD Expressed by RTCL

表 3 RTCL 表示的动态职责分离约束

DSOD Constraint	RTCL Expression
1. Exclusive Type	$(OE(u,user(RH\_role(wt_i))),OE(ti_i,\Im_{TI}(wt_i))) \in AZ_{NT} \Rightarrow cannot\_do(OE(u,user(RH\_role(wt_i))),OE(ti_j,\Im_{TI}(wt_j)))$
2. Assertive Type	$(OE(u,user(RH\_role(wt_i))),OE(ti_i,\Im_{TI}(wt_i))) \in AZ_{NT} \Rightarrow must\_do(OE(u,user(RH\_role(wt_i))),OE(ti_j,\Im_{TI}(wt_j)))$

5 结束语

工作流授权是 WFMS 研究的一个重要课题. 在分析了工作流授权的需求的基础上描述了一个基于角色和任务的工作流授权模型. 同基于角色的访问控制模型相比,该模型通过任务把角色和权限联系在一起,然后给用户指派合适的角色,用户通过其指派的角色获得可以执行的任务,然后在执行某个任务的某个具体实例时获得该任务所允许访问的客体的权限,这样更方便权限粒度的控制和管理;为每个任务指定一个执行时间模板,表示只能在某个时间段内执行该任务,只有在任务开始执行时才授权给用户并且任务一旦结束授权就要被收回,这样可以保证授权有效时间与任务执行时间尽可能同步;在工作流的执行过程中,系统会保存一个授权基,即任务的历史执行信息,根据这些历史执行信息求出有资格执行任务的用户集,从而实现动态职责分离. 还以此模型为框架提出了一个基于角色和任务的工作流授权约束描述语言 RTCL. 证明了在语义上 RTCL 与严格形式的一阶谓词逻辑 RFOPL 是等价的,并通过用 RTCL 表示各种各样的约束来说明 RTCL 的表现力.

参 考 文 献

1 Hong Fan, Li Jing. A task-based authorization model. Journal of Computer Research and Development, 2002, 39(8): 998~1003 (in Chinese)
(洪帆,李静. 基于任务的授权模型. 计算机研究与发展, 2002, 39(8): 998~1003)

2 R. S. Sandhu. Separation of duties in computerized information systems. In: S. Jajodia, C. E. Lanwehreds. Database Security IV. North Holland: Elsevier Science Publisher, 1991. 179~189

3 R. S. Sandhu, E. J. Coyne, H. L. Feinstein, *et al*. Role-based access control models. IEEE Computer, 1996, 29(2): 38~47

4 G. J. Ahn, R. S. Sandhu. Role-based authorization constraints specification. ACM Trans. Information and System Security, 2000, 3(4): 207~226

5 V. Atluri, W. K. Huang. An authorization model for workflows. In: Proc. 5th European Symposium on Research in Computer Security, Lecture Notes in Computer Science. New York: Springer-Verlag, 1996. 44~64

6 V. Atluri, W. K. Huang. A Petri net based safety analysis of workflow authorization models. Journal of Computer Security, 2000, 8(2): 83~94

7 E. Bertino, P. A. Bonatti, E. Ferrari. TRBAC: A temporal role-based access control model. ACM Trans. Information and System Security, 2001, 4(3): 191~223

8 Dong Guangyu, Qing Sihan, Liu Kelong. Role-based authorization constraint with time character. Journal of Software, 2002, 13(8): 1521~1527 (in Chinese)
(董光宇, 卿斯汉, 刘克龙. 带时间特性的角色授权约束. 软件学报, 2002, 13(8): 1521~1527)

9 N. R. Adam, V. Atluri, W. K. Huang. Modeling and analysis of workflows using Petri nets. Journal of Intelligent Information Systems, 1998, 10(2): 131~158

Xing Ganglin, born in 1972. Ph. D. candidate. His main research interests are access control and secure workflow.
邢光林,1972 年生,博士研究生,主要研究方向为访问控制技术和安全工作流.

Hong Fan, born in 1942. Professor and Ph. D. supervisor. Her main research interests are database security, cryptography, secure model and formalization method.
洪帆,1942 年生,教授,博士生导师,主要研究方向为数据库安全、密码学、安全模型和形式化方法.

Research Background

Workflow management systems are being widely used today by organizations to coordinate the execution of various applications representing their day-to-day tasks. To ensure that these tasks are executed by authorized users, a proper authorization mechanism must be provided. Constraints specification is also an important aspect of workflow authorization. In this paper, based on analyzing the requirements of workflow authorization, we introduce a workflow authorization model based on role and task, which should be very suitable for secure workflow, distributed management and office automation. The basic idea of this model is that roles and permissions are not connected directly but are put together by tasks. This is more convenient for controlling and managing the granularity of permissions. Then an intuitive formal language called RTCL is proposed, which takes the model as context to specify workflow authorization constraints based on role and task and is proved to be equivalent to a restricted form of first order predicate logic called RFOPL on semantics.