RBAC模型中角色的继承与互斥问题的研究*)

胡金柱　陈娟娟

(华中师范大学计算机科学系　武汉430079)　(软件工程国家重点实验室　武汉430072)

Research for Inheritance and Mutual Exclusion of Role in RBAC Model

HU Jin-Zhu　CHEN Juan-Juan

(Department of Computer Science, Central China Normal University, Wuhan 430079)

(State Key Laboratory of Software Engineering, Wuhan 430072)

Abstract　RBAC(Role-Based Access Control) maps naturally to an organization's structure and facilitates safety administration by separating logically users and permissions via roles as well as constructing role hierarchies, and therefore RBAC offers a powerful means of specifying access control decisions and is attracting increasing attention. In role hierarchies of RBAC, superroles inherit all properties and permissions of subroles. This paper classifies role inheritance into two types : generalization inheritance and supervision inheritance . Furthermore, it outlines two problems in relation to role inheritance : one is how to maintain data integrity, another is how to reduce the effect of absent roles on the normal running of the system . At last, this paper discusses solutions to them .

RBAC is attracting increasing attention as a security mechanism . Separation of duty is an important safety requirement which is implemented by means of mutual exclusion of roles in RBAC . This paper presents a basic RBAC model, then explores some properties of mutual exclusion of roles, which helps enforcing security policies efficiently. At last, this paper describes how mutual exclusion of roles affects role hierarchies.

Keywords　RBAC, Role inheritance, Mutual exclusion of roles, Role hierarchy, Separation of duty

1　引言

基于角色访问控制RBAC(Role-Based Access Control)的基本思想是引入角色将用户和访问权限间接联系起来，根据系统用户的工作职责设置角色，授予角色相应的访问权限，再为用户分配角色。图1给出了RBAC模型的基本思想。

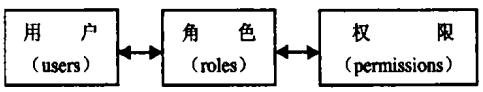

图1　RBAC模型示意图

RBAC模型根据企业组织内部视图中不同的工作岗位设置角色，所以很自然地反映了企业内部特定的安全管理策略。在RBAC模型中，角色是访问控制的主体，用户只能执行通过角色而拥有的访问权限，对于角色不具备访问权限的用户被强制不允许执行。所以RBAC较好地实现了最基本的安全规则：最小特权规则(the Least Permission Rule)，即分配给角色的权限在保证用户充分工作的基础之上，不能超越工作权限范围。

RBAC相对于DAC(Discretionary Access Control，自主访问控制)以及MAC(Mandatory Access Control，强制访问控制)将用户与访问权限直接对应的做法，减少了用户工作职责的变动对系统稳定性的影响，因为用户职责或任务易于变化，而根据工作职责设置的角色则相对稳定。同时由于安全管理员的主要工作只是为用户分配、取消角色，因此大大减少了授权管理的复杂性和工作量。正是由于RBAC模型作为存取控制策略的种种优点，在现代信息系统中，RBAC逐渐取代DAC和MAC并得到广泛应用。目前已有越来越多的系统支持RBAC，例如NIST用于Intranet的管理工具RBAC/Web，Schumann公司的SAM。

RBAC模型将用户和访问权限通过角色进行逻辑分离，并且构造了角色之间的层次继承关系，即角色可以通过角色继承而自动拥有子角色的属性和访问权限。故可自然灵活地反映企业内部组织结构，方便系统安全管理，所以在信息系统中得到了广泛应用。

RBAC本身作为一种存取控制策略，主要是为了解决应用系统日益复杂但又极其重要的安全管理问题，所以安全机制在RBAC中是一关键部分。信息系统的安全需求要求一个计算机应用系统应能实现两个最基本的安全规则：最小特权规则和职责分离(Separation of Duty)规则。在RBAC模型中，角色互斥是实现职责分离规则自然而有效的形式，所以有着极其重要的研究价值。

2　RBAC基本模型描述

定义1(用户)　用户是可以独立访问计算机系统资源或数据的主体，一般是指使用计算机系统资源的人，本文用Users表示用户集合。

定义2(权限)　权限是访问计算机系统资源或数据的许可权，又称做访问权限，本文用Permissions表示权限集合。

RBAC是策略中立的，并没有具体规定应该如何定义权限，权限的定义依赖于具体系统的实现。

定义3(角色)　角色对应于企业内部组织中某一特定的

*)本文受软件工程国家重点实验室开放研究基金(SKL(4)018)和湖北省科技攻关项目(2001AA101C31)支持。

工作岗位，是一组访问权限的集合，本文用 Roles 表示角色集合。

定义4(角色权限授权) PA: Roles→$2^{Permissions}$，PA(r)是角色 r 被授予的访问权限集合。

角色与权限之间是 n-n 的关系。由于用户是通过角色间接拥有访问权限，因此在进行角色权限分配时，应注意满足最小特权规则。

定义5(用户角色授权) UA: Users→2^{Roles}，UA(u)是用户 u 被授予的角色集合。

一个用户根据其工作职责可以被授予多个角色，一个角色也可被授予多个用户，故用户与角色之间也是 n-n 的关系。

定义6(会话) 会话是一个用户的一次活跃进程，代表用户执行其访问权限。用户是静态概念，会话是动态概念；一个用户可以同时打开多个会话，但是一个会话只能由一个用户打开，用户与会话之间是一对多的关系。本文用 Sessions 表示会话集合。

定义7(会话的用户) U: Sessions→Users，U(s)是与会话 s 有关的用户。

定义8(会话的活跃角色集) A: Sessions→2^{Roles}，A(s)是会话 s 的当前活跃角色集。会话作为用户的一次活跃进程，激活了此用户角色集的某个子集，这个子集就称作会话的活跃角色集，它决定了在此次会话中用户实际拥有的访问权限。

下面是一组本文所使用的各种类型的变量：

u_i: Users s_i: Sessions r, r_1, r_2: Roles p, p_1, p_2: Permissions

性质1 会话的活跃角色集是与此会话相关的用户的角色集的子集。

$(\forall s)(\forall u)|U(s)=u_i$:

$A(s)\subseteq UA(u)$

性质2 会话仅能执行其活跃角色集之中的角色的访问权限。

性质1说明了用户的角色集与会话的活跃角色集之间的关系，性质2说明了权限执行的必要条件。由性质1可以推出如下定理1。

定理1 一角色若是会话的当前活跃角色，则一定属于与此会话相关的用户的角色集；反之不成立。

$(\forall s)(\forall u)(\forall r)|U(s)=u_i: r\in A(s)\Rightarrow r\in UA(u)$

$(\forall s)(\forall u)(\forall r)|U(s)=u_i: r\in UA(u)\not\Rightarrow r\in A(s)$

以上描述了角色与权限、用户与角色之间的相互关系。

3 角色继承及其有关的问题

3.1 角色继承的定义

在 RBAC 模型中，为了避免相同权限的重复设置，角色除了具有自身的属性与权限之外，还可以继承其他角色，从而自动拥有被继承的角色的属性与权限。

定义9(角色继承) 如果角色 r_1继承了角色 r_2，则称 r_1是 r_2的父角色，r_2是 r_1的子角色，记为 $r_1\geqslant r_2$。

角色继承(Roles Hierarchy)是一种偏序关系。定义角色继承不但可以避免相同权限的重复设置，还可以很好地描述角色之间的层次关系。本文用角色层次图图形化地描述角色层次关系，角色层次图是一有向非循环图：父角色结点处于子角色结点的上方，从父角色结点出发到子角色结点终止的有向边代表二者之间的继承关系。图2是一个简单的角色层次图[2]。

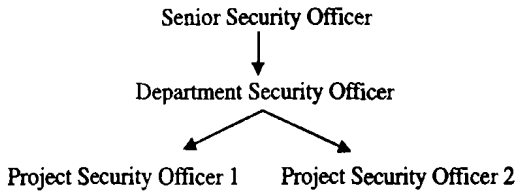

图2 角色层次图举例

性质3 任一用户若被授予了父角色，则也被授予了子角色。

$(\forall u)(\forall r_1,r_2)r_1\in UA(u)\wedge r_1\geqslant r_2\Rightarrow r_2\in UA(u)$

性质4 父角色若是会话的当前活跃角色，则子角色也是此会话的当前活跃角色。

$(\forall s)(\forall r_1,r_2)r_1\in A(s)\wedge r_1\geqslant r_2\Rightarrow r_2\in A(s)$

我们认为，在 RBAC 模型中可以将角色继承划分为泛化继承和监管继承两大类，划分角色类型有利于保证信息系统中数据的完整性和一致性，以及更有效地预防滥用权限。

3.2 泛化继承

所谓泛化继承(Generalization Inheritance)主要是根据父、子角色之间的具体与抽象的关系确定角色继承关系，即父角色是子角色的抽象化，子角色是父角色的具体化，本文用 $r1\rightarrow_g r2$ 表示 r1泛化继承 r2 。

如图3所示，图3是一个“法院综合信息管理系统”的部分角色层次图，图中的院长居于最高层次、是副院长的父角色，具有最高的抽象；而所有用户处于最低层次，是具体的用户。所以，图3中的纵向关系是一种角色的泛化继承。

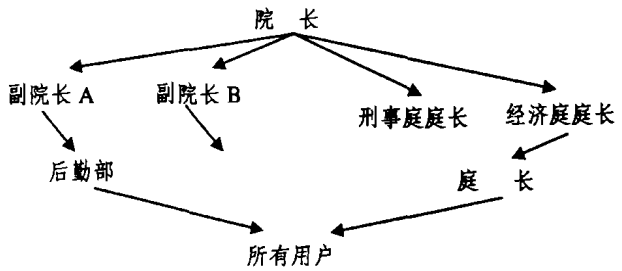

图3 法院部分角色层次

定义10(虚角色与实角色) 没有授权给具体用户的角色叫作“虚角色(Virtual Role)”，用 VR 表示虚角色集合；已经授权给具体用户的角色叫作“实角色(Real Role)”，本文用 RR 表示实角色集合。

定义虚角色和实角色有利于更加清楚、方便地表示角色之间的层次关系和角色的共性。

定义11 isVR: Roles→Boolean，判断一角色是虚角色或实角色 IsVR(r)=True 当且仅当 r 是虚角色。

下面的性质5、6、7用以具体说明虚角色和实角色的性质及其相互关系。

性质5 角色集合是虚角色集合与实角色集合的非交并：

VR I RR=Φ VR Y RR =Roles

性质6 虚角色不可授权给用户：$(\forall r)$isVR(r)=True⇒UA(r)=Φ

性质7 虚角色不可泛化继承实角色：

$(\forall r1)$isVR(r1)=True⇒¬$(\exists r2)$((isVR(r2)= False)∧$(r1\rightarrow_g r2)$)

性质7在角色层次图中体现为虚角色不可能在实角色的上方。图3中，处于最下方的角色“所有用户”就是为了反映其他角色的共性而设置的虚角色，它没有授予具体用户，此角色

具有系统最基本的访问权限，其他角色都是从直接泛化继承或间接泛化继承而来。除此之外，角色“庭长”也是为了反映角色“刑事庭庭长”、“经济庭庭长”等的共性而设置的，显然“庭长”也是泛化继承“所有用户”。

3.3 监管继承

所谓监管继承(Supervision Inheritance)主要是根据父、子角色所代表的工作岗位的上下级关系确定角色继承关系。若 r1监管继承 r2，且 r1被授权给用户 u1，r2被授权给用户 u2，则 u1和 u2实质上是上下级关系。图3中，角色“院长”就直接监管继承了“副院长 A”、“副院长 B”等，而“副院长 A”又直接监管继承了“后勤部”，所以“院长”间接监管继承了“后勤部”。本文用 r1→r2表示 r1监管继承 r2，称 r1是监管角色，r2是被监管角色。

在设置角色和确定角色之间的层次关系时，泛化继承和监管继承可以同时体现在角色层次图中，图3就同时具有泛化继承和监管继承。事实上，被监管角色只可能是实角色，而不可能是虚角色。确定角色之间的监管继承关系后，上级用户既可随时监督下级用户的工作情况，又可及时发觉下级用户的非法行为对系统造成的危害。

3.4 与角色继承有关的问题

(1)数据完整性和一致性问题：根据角色继承的定义，如果被监管角色可以访问某数据，那么监管角色也可对之进行访问，但是由此却可能破坏数据的完整性和一致性。一个十分典型的例子是：在法院日常办公的公文流转过程中，对于一份尚处于起草阶段而未提交审批的公文，如果“院长”对之进行读取或备份，就破坏了此公文数据的完整性和一致性。

为此可以为数据设置状态，称作“数据状态(Data State)”，且数据的初态是“ready”，并对监管角色应有以下约束：当被监管角色完成相应动作之后，数据状态转为“completed”，此时监管角色方可访问此数据。

(2)缺席角色(Absent Role)问题：当拥有某一关键角色(所谓关键角色，是指在整个系统运作中必不可少、并且如果不通过角色继承就只能直接授予一位用户的角色)的用户因无法预料的情况(如生病)而不能工作时，为了让系统继续正常运行，可由它的一个直接监管角色执行其工作职责，此时将执行缺席角色工作职责的直接监管角色称作“备份角色(Back-up Role)”。显然，引入“备份角色”可以解决关键角色的缺席问题。

但是为了防止备份角色滥用权利，同样应对之进行约束：仅当关键角色缺席时，才可由其备份角色代替执行工作职责。约束是基于角色的访问控制中重要的安全策略，是对用户执行权限的一些限制。

4 角色互斥

在现实领域中，为了安全起见，用户不能同时独立完成某些操作，每个操作要由不同用户完成，这种现象称之为职责分离(Separation of Duty)。例如在办公自动化系统中的公文审批过程中，公文起草与公文审批不能由同一人进行。职责分离规则是一种最基本的安全措施，主要是在保证应用系统各部分正常通信的基础之上防止滥用权利和维护信息的完整性。

4.1 角色互斥的定义

定义12(权限互斥) 对于任意两个访问权限 p_1和 p_2，若任何用户都不可能同时拥有它们，则称访问权限 p_1和 p_2互斥，记为$(p_1,p_2)\in E_p$。

定义13(角色互斥) 对于任意两个角色 r_1和 r_2，若授予 r_1的某一访问权限与授予 r_2的某一访问权限互斥，则称角色 r_1和 r_2互斥，记为$(r_1,r_2)\in E$。用公式表示为：

$$(\forall r_1,r_2)(\exists p_1\in PA(r_1))\wedge(\exists p_2\in PA(r_2))\wedge(p_1,p_2)\in Ep\Leftrightarrow(r_1,r_2)\in E$$

定义14(静态互斥) 静态互斥(Static mutual exclusion)是指在给用户分配角色时，如果一个用户已被授予了角色 r_1，并且存在另外一个不同的角色 r_2与 r_1互斥，则此用户不能再被指定为角色 r_2的用户，记为$(\forall r_1,r_2)(r_1,r_2)\in ATE$。用公式表示为：

$$(\forall r_1,r_2)(\forall u)\mid r_1\neq r_2:(r_1,r_2)\in ATE\wedge r_1\in UA(u)\Rightarrow r_2\notin UA(u)$$

定义15(动态互斥) 动态互斥(Dynamic mutual exclusion)是指用户可以同时拥有互斥角色，但是此用户的任何会话在执行权限时却不能同时激活互斥的角色，记为$(\forall r_1,r_2)(r_1,r_2)\in RTE$。用公式表示为：

$$(\forall r_1,r_2)(\forall s)\mid r_1\neq r_2:(r_1,r_2)\in RTE\wedge r_1\in A(s)\Rightarrow r_2\notin A(s)$$

例如，办公自动化系统中的公文流转可以分为如下三个阶段：

(1)公文起草阶段：由具有“起草者”角色的用户进行；

(2)公文审批阶段：由具有“审批者”角色的用户进行；

(3)公文发布阶段：若公文审批后确认为可行，则由具有“发布者”角色的用户进行。

其中，“起草者”角色和“审批者”角色因为不能分配给同一用户，所以属于静态互斥；虽然公文的起草与发布可以由同一用户进行，但对同一公文而言，用户不能同时激活“起草者”角色和“发布者”角色，所以它们属于动态互斥。由于静态互斥从角色授权时就一直存在，而动态互斥只在角色激活时发生作用，因此动态互斥比静态互斥的约束作用弱。动态互斥是静态互斥的有益补充，它可以灵活地运用于企业内部组织中。

4.2 角色互斥的性质

性质8 任一角色与其自身不具有互斥关系：$(\forall r)(r,r)\notin E$，即角色互斥关系不满足自反性，所以角色互斥均指互不相同的角色之间的互斥。

性质9 角色互斥关系具有对称性：$(\forall r_1,r_2)(r_1,r_2)\in E\Leftrightarrow(r_2,r_1)\in E$。

定理2 角色互斥不具备传递性：

$$(\forall r_1,r_2)(r_1,r_2)\in E\wedge(r_2,r_3)\in E\not\Rightarrow(r_1,r_3)\in E。$$

证明：用反证法，假设由$(r_1,r_2)\in E$以及$(r_2,r_3)\in E$得出：$(r_1,r_3)\in E$。

当用 r_1代替 r_3时，则有：$(r_1,r_2)\in E\wedge(r_2,r_1)\in E\Rightarrow(r_1,r_1)\in E$。这与性质8相矛盾。命题得证。

定理3 任何两个角色如果满足静态互斥，则一定满足动态互斥；反之不成立。

$$(\forall r_1,r_2)(r_1,r_2)\in ATE\Rightarrow(r_1,r_2)\in RTE$$

$$(\forall r_1,r_2)(r_1,r_2)\in RTE\not\Rightarrow(r_1,r_2)\in ATE。$$

定理4 任何互斥角色不能同时成为当前活动角色。

证明：令$\forall r_1,r_2$且$(r_1,r_2)\in E$。

①当$(r_1,r_2)\in ATE$，不失一般性，设$\exists s$使得 $r_1\in A(s)$且 $U(s)=u$。

用反证法证明，假设 $r_2\in A(s)$。$\Theta r_2\in A(s)$，由定理2可知：$r_2\in UA(u)$；又$\Theta r_1\in A(s)$，由定理2可知：$r_1\in UA(u)$。

故由假设 $r_2\in A(s)$可得出 $r_1\in UA(u)$且 $r_2\in UA(u)$，这

与前提条件$(r_1,r_2)\in ATE$相矛盾。命题得证。

②当$(r_1,r_2)\in RTE$

由运行时互斥的定义可直接证明，在此略去。

定理4是用角色互斥实现职责分离的必要条件。

4.3 与角色继承有关的角色互斥性质

定理5 具有继承关系的角色不可能互斥，即任何互斥的角色之间均没有继承关系：

$(\forall r_1,r_2)(r_1,r_2)\in E\Rightarrow\neg(r_1\geqslant r_2\vee r_2\geqslant r_1)$

证明：不失一般性，设$\exists$ s 使得 $r_1\in A(s)$且 $U(s)=u$。

用反证法证明，不失一般性假设 $r_1\geqslant r_2$。

由性质4得，$(r_1\in A(s)\wedge r_1\geqslant r_2)\Rightarrow r_2\in A(s)$

故得如下结论：$r_1\in A(s)\wedge r_2\in A(s)$，这与前提条件$(r_1,r_2)\in E$相矛盾。命题得证。

定理6 任何角色不能多重继承互斥的角色，即若$(r_1,r_2)\in E$，则不存在 r：

$r\geqslant r_1\wedge r\geqslant r_2$。

证明：不失一般性，设$\forall r_1,r_2,r$ 且$(r_1,r_2)\in E$ 且 $r\geqslant r_1$；$\exists$ s 使得 $r_1\in A(s)$。现需证明 $r\geqslant r_2$不成立。

用反证法证明，假设 $r\geqslant r_2$成立。

由性质4得，$(r\in A(s)\wedge r\geqslant r_1)\Rightarrow r_1\in A(s)$

$(r\in A(s)\wedge r\geqslant r_2)\Rightarrow r_2\in A(s)$

故可得如下结论：$r_1\in A(s)\wedge r_2\in A(s)$，这与前提条件$(r_1,r_2)\in E$相矛盾。命题得证。

定理7 一角色若继承了互斥角色中的某一方，则该角色也与另一方互斥，即若

$(r_1,r_2)\in E$ 且$\exists$ $r\geqslant r_1$，则$(r,r_2)\in E$。

证明：由互斥定义得，

$(r_1,r_2)\in E\Rightarrow(\exists\ p_1\in PA(r_1),\exists\ p_2\in PA(r_2))(p_1,p_2)\in Ep$

由角色继承定义得，$r\geqslant r_1\wedge p_1\in PA(r_1)\Rightarrow p_1\in PA(r)$

故$\exists\ p_1\in P_A(r),\exists\ p_2\in PA(r_2)$且$(p_1,p_2)\in Ep$，由角色互斥定义得，$(r,r_2)\in E$。命题得证。

在公文审批中，“起草者”与“审批者”是互斥角色，由定理5可以防止这两个互斥角色中的任一方通过继承另一方而破坏职责分离规则；由定理6和定理7可以防止其他角色通过多重继承而同时进行公文起草与审批。

结束语 本文将角色继承划分为泛化继承和监管继承，分析了由于角色监管继承而可能破坏数据的完整性和一致性、角色缺席可能对系统正常运行造成破坏的问题，在此基础上提出了解决这两个问题的方案。在 RBAC 模型中利用角色互斥可以实现现实领域中职责分离的安全需求，认真研究角色互斥及其性质，可以更好地划分用户职责，防止滥用权利对系统安全造成的危害。本文所研究的内容不仅具有理论意义，而且具有较大的实用价值。本文所研究的内容在我们开发的“法院综合信息管理系统”中得到了很好的应用。

参 考 文 献

1 Yan Han, Liu Feng-Yu, Zhang Hong. An object-oriented model of access control based on role. ACM SIGSOFT Software Engineering Notes, 2000, 25(2): 64～68

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

3 Sandhu R, Ferraiolo D, Kuhn R. The NIST Model for Role-based Access Control: Towards A Unified Standard [A]. In: Proc. of 5th ACM2000. Workshop on Role-Based Access Control [C]. ACM, Berlin, Germany, July 2000

4 Simon R T, Zurko M E. Separation of Duty in Role-Based Environments. In: Proc. of Computer Security Foundations Workshop X, Rockport, Massachusetts, 1997

5 Moffett J D. Control Principles and Role Hierarchiees. In: 3rd ACM Workshop on Role-Based Access Control(RBAC). 1998

6 Moffett J D, Lupu E C. The Uses of Role Hierarchies in Access Control. In: 4th ACM Workshop on Role-Based Access Control (RBAC). 1999

（上接第159页）

出了 Petri 网的抽象方法，它可使人们用不同的粒度，对不同抽象级别的概念进行有效的建模，化简了问题的复杂度，使人们在可控制的前提下充分地利用 Petri 网这种直观的形式化工具对问题进行正确的建模和分析。

结束语 本文描述了信牌驱动模型的分层结构（块和子过程）所对应的正则 Petri 网。至此，通过文[6～8]和本文完整地论述了一个功能强大、使用方便的工作流模型——扩展的信牌驱动模型，解决了过程定义的建模问题。同时，也详细地讨论了一个与之对应的形式化模型——正则 Petri 和非确定 Petri 模型，以及将信牌驱动这种工作流过程定义模型转化为了一个 Petri 网的方法。即我们不仅可以用灵活、方便、表达能力强的信牌驱动模型对工作流的过程进行适当、准确和全面的建模，而且，还能将这种模型能转化为一个 Petri 网。这就为进一步对信牌驱动模型进行分析、验证、甚至仿真、执行和其他性能分析等等提供了坚实的理论基础和保障。

参 考 文 献

1 WfMC. The Workflow Reference Model. http://www.aiim.org/wfmc

2 史美林，向勇，杨光信. 计算机支持的协同工作理论与应用. 北京：电子工业出版社，2000

3 范玉顺. 工作流管理技术基础. 北京：清华大学出版社，2001

4 van der Aalst W M P. The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 1998. 1～53

5 Wang Binjun, Hao Kegang. Three Levels of Workflow Model, in Proc. ISFST2001

6 王斌君. 工作流过程模型的层次研究及其分析：[西北大学博士论文]. 2002. 4

7 岳晓丽，杨 斌，郝克刚. 信牌驱动式工作流计算模型. 计算机研究与发展，2000，37(12)：1513～1519

8 郝克刚，王斌君. 非确定 petri 网. 小型微型计算机系统(已录)

9 Booch G. Object_Oriented Design With Applications. The Benjamin/lemmings, 1991

10 Best E, Koutny M. Petri net semantics of priority system. Theoretical Computer Science, 1992. 175～215

11 Peterson J L. Petri Net Theory and The Modeling of Systems. Prentice-hall, 1981

12 袁崇义. Petri 网原理. 北京：电子工业出版社，1998