基于角色的操作系统完整性控制模型

刘 尊 王 涛 李伟华
(西北工业大学计算机学院 西安 710072)

摘 要 完整性是操作系统的安全目标之一。分析了安全操作系统的完整性保护策略,详细说明了用户职责和进程可信度是决定系统中用户、进程和文件完整级的重要因素。提出了一个完整性控制模型,该模型使用角色来简化对用户完整级的管理。分别给出了模型的完整性公理、模型元素、系统状态的定义、系统设置方法和以及状态转换规则,并对模型进行了正确性证明,最后介绍了模型的应用实例和实现方法。

关键词 角色,完整级,完整性控制,安全模型,操作系统
中图法分类号 TP309 **文献标识码** A

Role Based Integrity Control Model for Operating System

LIU Zun WANG Tao LI Wei-hua
(School of Computer, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Integrity is one of the security objectives of operating system. This paper analysed the integrity protection policies of security operating system, and elaborated user responsibility and process reliability are important factors in determining the integrity level of the users, the processes and the files in operating system. Then we proposed an integrity control model with detailed description of its integrity axiom, model elements, system states, system configurations and state transition rules. The model employs roles to simplify user integrity management. The correctness of given model was proven. The application case and implementation method were given at the end of the paper.

Keywords Role, Integrity level, Integrity control, Security model, Operating system

1 引言

完整性是信息系统的一项重要安全需求。作为信息系统软硬件资源的管理者,安全操作系统应提供完整性保护特性。在安全操作系统设计中的"不向上破坏的原则"即低等级主体不能破坏高等级客体信息的完整性原则,就是从完整性保护方面对安全操作系统设计提出的要求。

J. P. Anderson 指出,要开发安全系统首先必须建立系统的安全模型[1],要开发具有完整性保护特性的安全操作系统,必须从建立操作系统完整性模型开始。有关安全操作系统完整性模型的研究从 1977 年 Biba 模型提出开始,受到了研究者的广泛关注,已提出了不少完整性模型,其中比较典型的有 Biba 等人于 1977 年提出的 Biba 模型[2]和 David Clark 等人于 1987 年提出的 Clark-Wilson 模型[3]。

Biba 模型基于 MLS 策略,为系统中的主客体分别指定一个服从偏序关系的完整级,制定了一系列完整性策略,对系统中主体对客体的访问进行限制。Biba 模型中较为常用的严格完整性策略与 BLP 模型安全策略类似。但是,由于在实际中缺乏与完整级相对应的分级机制,因此很难确定主客体的完整标签。

Clark-Wilson 模型以良构事务为基础实现在商务环境中所需的完整性策略,要求只能由一种能够确保数据完整性的受控方式来操作数据,从而确保数据的完整性。但是,由于该完整性策略的内在复杂性,使得有关它的形式化工作至今仍未完成,因此造成了 Clark-Wilson 完整性策略难以被精确理解和有效使用的局面。

中国科学院软件研究所的季庆光博士提出了基于 DTE 技术的完整性保护形式模型 DTE_IPM[4],提出为完成完整性保护进行配置 DTE 的一些基本规则,但在具体的系统设计中仍然需要进一步细化。

2 安全操作系统完整性保护策略

完整性策略包括数据完整性及系统完整性两部分[5],其目标是阻止非授权用户对保护对象的修改行为、阻止授权用户对保护对象的不恰当的修改行为、保持保护对象的内在及外在的一致性。如果将对信息的授权修改视为新信息的产生,则完整性保护的本质就是保证信息在从生产者向使用者的传递过程中不被修改,一直保持原来的状态。

完整性保护的核心是让使用者能够确定信息的真伪和可信程度。信息的真伪取决于信息在传递过程中有无非授权修改,信息的可信程度取决于信息是由谁产生的。如果使用者不能获得所接收信息的产生及修改情况,他就不能信任并使

到稿日期:2009-02-20 返修日期:2009-04-23 本文受国家 863 项目(2006AA01Z406)资助。
刘 尊(1979—),男,博士研究生,讲师,CCF 学生会员,主要研究方向为操作系统安全、网络信息安全,E-mail:liuzun@nwpu.edu.cn;王 涛(1980—),男,博士研究生,讲师,主要研究方向为操作系统安全;李伟华(1951—),男,教授,博士生导师,主要研究方向为信息安全、决策支持。

用这条信息。可见，信息的完整性问题实质上是对信息的信任问题。

完整级是完整性保护的基础，因此需要为影响信息完整性的元素分别确定完整级。这些元素主要有响应存取操作的信息载体(客体)、直接操作信息的主动元素进程(主体)和使用进程对信息进行处理的操作者(用户)。

客体的完整级是指客体中所承载信息的可信程度，取决于生成信息的主体和用户的可信程度，与信息的具体内容无关。客体的完整级由产生该客体的用户和主体的完整级决定。如果信息被非授权修改，信息的完整性级别则降到最低。

用户的完整性级别是指系统中不同用户之间互相信任的程度，与用户在系统中所承担的职责有关。不同职责的用户所生成的信息的可信程度是不同的，所以他们的完整性级别也不相同。

主体的完整性级别是指主体在操作信息过程中能够严格执行用户指令，确保所操作信息不被破坏的程度。主体完整级主要取决于完整性管理员对其验证的结果及对其行为的熟悉程度。如果一个主体能够被验证是经过适当构造并正确安装和使用的，则其完整级可以定义为系统最高完整级；反之，任何一个未经验证的主体其完整级都为系统最低完整级。如专门破坏系统完整性的特洛伊木马，由于没有经过验证过程，其完整级为系统最低完整级，以便最大限度地避免其对系统的破坏。

客体修改完整级来保护客体的完整性，具体级别由客体的所有者根据需要确定。只有在用户的完整级和所使用主体的完整级均支配客体修改完整级时，才允许用户通过主体对客体进行修改操作。

可见，完整性级别的确定与机密性级别的确定有所不同。客体所承载的信息的特性在机密性级别确定中起主导作用，而主体及用户的特征在完整性级别的确立中起主导作用。

3 基于角色的完整性控制模型

用户在系统中的职责是用户完整性级别的决定因素，用户的完整性级别又是系统完整性的关键因素，因此需要根据用户职责，为不同用户设置合适的完整性级别。而角色就是是管理用户职责的一个好工具。基于角色的完整性控制模型(Role Based Integrity Control Model，RBICM)将利用角色对用户的权限进行管理，并依据用户所承担的角色来确定用户当前的完整性级别。

3.1 模型元素及系统状态

下面给出 RBICM 模型中使用的一些元素。

用户集合 $U:=\{U_1,U_2,\cdots,\}$。

角色集合 $RO:=\{intadmin,RO_1,RO_2,\cdots,RO_m\}$，包括完整性管理员和若干其他角色。

主体集合 $S:=\{S_1,S_2,\cdots,S_n\}$，表示系统中运行的执行对客体访问的进程。

客体集合 $O:=\{O_1,O_2,\cdots,O_p\}$。

完整级 $I:=\{I_{low},I_1,I_2,\cdots,I_n,I_h\}$，包括系统最低完整级 I_{low} 和系统最高完整级 I_h 以及若干普通完整级，完整级之间为偏序关系，$I_{low}\leqslant I_1\leqslant I_2\leqslant\cdots\leqslant I_n\leqslant I_h$。

访问模式集合 $A:=\{r,w,e\}$，其中 r 为可读不可写方式，w 为可读且可写方式，e 为执行方式。

请求集合 $R:=\{R^1,R^2,R^3,R^4,R^5\}$，其中 R^1 请求访问，R^2 请求产生一个新客体，R^3 请求删除一个客体，R^4 请求改变完整级，R^5 表示授予/撤销访问权限。

判定集合 $D:=\{yes,no,undefined\}$，yes 表示请求被成功执行，no 表示请求未被成功执行，$undefined$ 表示规则 ρ 无法处理该请求。

当前访问集合 $b:b\subseteq R\times S\times O\times A$。

访问矩阵 $M:=\{M_{ik}\}$。

完整级函数集合 $F:=\{(f_r,f_s,f_o,f_l)\}$。

系统状态 v 由角色、主体、客体、访问模式、访问矩阵和完整级函数组成。v 由一个有序四元组 (b,M,f,H) 组成，其中 b 表示在某特定状态下哪些角色通过哪些主体以何种形式访问哪些客体；M 是访问矩阵，其中元素 $M_{ik}\in A$，表示角色 r_i 对客体 o_k 的访问权限；$f\in F$，是完整级函数，$f=(f_r,f_s,f_o,f_l)$，其中 f_r 称为角色完整等级函数；f_s 称为主体完整等级函数，f_o 称为客体完整等级函数，f_l 为客体最低修改完整等级函数；H 是表示当前的层次结构，即当前客体的树形结构 $O_k\in H(O_p)$ 表示在此树形结构中 O_k 为叶子节点，O_p 为父节点。

3.2 状态转换

系统状态间的转换由一组操作规则定义。一个规则可以表示为一个函数，它对每个状态和每个请求(输入)决定系统产生下一状态和一个响应。规则 ρ 定义为

$$\rho:R\times V\to D\times V'$$

其中，R 为系统中为请求定义的请求集，V 为状态集，D 为系统中为请求定义的判定(decision)集合。

3.3 模型的完整性公理

为了保证系统的完整性，模型定义了一组必须满足的完整性公理。一个系统的状态是安全的，当且仅当这些安全公理被满足。

公理 1 系统状态 $v=(b,M,f,H)$ 满足完整性公理 1，

iff $\forall b=(RO_i,S_j,O_k,x)\in B$：

$x=w(f_c(RO_i)\geqslant f_l(O_k)$ and $f_s(S_j)\geqslant f_l(O_k))$

这个公理说明，如果某角色通过某主体对某客体进行写操作，必须满足该角色的当前完整级和该进程的完整级均大于客体的修改完整级。

公理 2 系统状态 $v=(b,M,f,H)$ 满足安全公理 2，

iff $\forall(RO_i,O_k,x)\in b\Rightarrow x\in M_{ik}$

3.4 系统设置

在 RBICM 中，由系统完整性管理员负责系统的设置。完整性管理员根据组织需要设置若干角色和完整级序列，根据各角色所承担的职责为其设置合适的完整级，并为各用户指定可以登录的角色。

因为主体是由二进制可执行文件运行产生的，所以需要对系统中的二进制可执行文件进行行为验证，并根据验证结果为对应主体设置合适的完整级，未验证程序的完整级设为 I_{low}。

对于客体完整级，通常为系统文件设置较高的完整级，而用户文件的完整级由创建、修改文件的角色和主体决定。

对于客体的修改完整级，通常将不需要更新的系统配置文件和经过验证的可执行文件的修改完整级设为 I_h，将其他需要修改的重要配置文件设置为较高的修改完整级，将临时文件等多数用户需要修改的文件的修改完整级设置为 I_{low}。

用户文件的修改完整级由用户根据需要确定。

若需对修改完整级设为 I_h 的客体进行更新维护，则需在停止该客体的使用后降低其修改完整级。待更新维护后重新验证其行为，并设置其修改完整级和对应主体的完整级。

3.5 状态转换规则

如无特别说明，在下述规则中，RO_i 指用户登录的一个角色 i，S_j 指用户用来存取客体的进程 j，O_k，O_{new} 指客体。

规则 1　create-object 表示请求创建一个客体。

请求描述 $(RO_i, S_j, O_k) \in R^2$

if $[\min(f_r(RO_i), f_s(s_j)) \geqslant f_l(O_k)] \& [O_k \in b(RO_i, S_j, w)]$

then 规则输出 *yes*，

系统状态 $v' = (b, M', f', H')$，其中

$M' = M \cup \{M_{ik} = \{r, w, e\}\}$

$f' = (f \setminus f_o \leftarrow f_o \cup (f_o(O_{new}) = \min(f_r(RO_i), f_s(S_j)))) \cup (f \setminus f_l \leftarrow f_l \cup (f_l(O_{new}) = L_u))$，其中 $L_u \in [I_{low}, I_h]$

$H' = H \cup (O_j, O_{new})$

else 规则输出 *no*，

系统状态保持不变。

其中，$\min(f_r(RO_i), f_s(S_j))$ 表示取 $f_r(RO_i)$，$f_s(S_j)$ 二者中的较小值；$L_u \in [I_{low}, I_h]$，表示 L_u 可根据需要由用户决定在 $[I_{low}, I_h]$ 范围内取任意值。

规则说明：当符合以下条件时，用户可以创建一个父节点为 O_k 的客体，并设置其完整级为 RO_i 完整级与 S_j 完整级中的较小值，设置其修改完整级为 I_{low} 到 I_h 间的任意值。

- RO_i 及 S_j 的完整级支配 O_k 的修改完整级；
- RO_i 及 S_j 当前以写权限访问客体 O_k。

规则 2　get-write 表示请求修改一个客体。

请求描述 $(RO_i, S_j, O_k, w) \in R^1$

if $[(f_r(RO_i) \geqslant f_l(O_k)] \& [f_s(S_j) \geqslant f_l(O_k)] \& [w \in M_{ik}]$

then if $[(f_r(RO_i) \geqslant f_o(O_k)] \& [f_s(S_j) \geqslant f_o(O_k)]$

then 规则输出 *yes*，

系统状态 $v' = (b', M, f', H)$，其中

$b' = b \cup (RO_i, S_j, O_k, w)$

$f' = (f \setminus f_o \leftarrow (f_o'(O_k) = L_{uo}) \cup (f \setminus f_l \leftarrow f_l'(O_k) = L_{ul})$

其中 $L_{uo} \in [\min(f_r(RO_i), f_s(s_j)), f_o(O_k)]$

$L_{ul} \in [f_l(O_k), I_h]$

else 规则输出 *yes*，

系统状态 $v' = (b', M, f', H)$

$b' = b \cup b(RO_i, S_j, O_k, w)$

$f' = f \setminus f_o \leftarrow (f_o(O_k) = \min(f_r(RO_i), f_s(S_j)))$

else 规则输出 *no*，

系统状态保持不变。

规则说明：当符合以下条件时，用户可以修改客体 O_k，可设置其完整级为 O_k 原有完整级与 RO_i 完整级和 S_j 完整级中较小值之间的任意值，可设置客体修改完整级为原修改完整级与 I_h 间的任意值。

- RO_i 及 S_j 的完整级支配客体 O_k 的修改完整级；
- 角色 RO_i 的访问属性中有对客体 O_k 的写权限；
- RO_i 及 S_j 的完整级支配客体 O_k 的完整级。

当符合以下条件时，用户可以修改客体 O_k，并设置其完整级为用户登录角色 RO_i 的完整级及用户使用进程 S_j 的完整级的最小值。

- RO_i 及 S_j 的完整级支配客体 O_k 的修改完整级；
- 角色 RO_i 的访问属性中有对客体的写权限；
- 客体 O_k 的完整级支配 RO_i 或 S_j 的完整级。

规则 3　delete-object 表示请求删除一组客体。

请求描述 $(RO_i, S_j, O_k) \in R^3$

if $[f_r(RO_i) \geqslant f_l(O_{f(k)})] \& [f_s(S_j) \geqslant f_l(O_{f(k)})] \& [O_{f(k)} \in b(RO_i, S_j, w)] \& [\forall O_s \in inferior(O_k), [f_r(RO_i) \geqslant f_l(O_s)] \& [f_s(S_j) \geqslant f_l(O_s)]]$

then 规则输出 *yes*，

统状态 $v' = (b', M', f', H')$

$b' = b - access(O_k)$

$M' = M - column(O_k)$

$f' = f/(f_o \cup f_l) \cup (f_o - f_o(inferior(O_k))) \cup (f_l - f_l(inferior(O_k)))$

$H' = H - inferior(O_k)$

else 规则输出 *no*，

系统状态保持不变。

其中，$O_{f(k)}$ 表示 O_k 的父节点，$inferior(O_k)$ 表示节点 O_k 及其所有子节点；$access(O_k)$ 表示与节点 O_k 及其子节点相关的当前访问集中的项；$column(O_k)$ 表示节点 O_k 及其子节点在访问矩阵 M 中对应的列。

规则说明：当符合以下条件时，用户可以删除客体 O_k 及其所有子节点。

- RO_i 及 S_j 的完整级支配 O_k 父节点的修改完整级；
- RO_i 及 S_j 当前以写权限访问 O_k 的父节点；
- RO_i 及 S_j 的完整级支配 O_k 及其所有子节点的修改完整级。

规则 4　decrease-integrity-class 表示请求降低客体最低修改安全级。

请求描述 $(RO_i, S_j, O_k, I_x) \in R^4$

if $[RO_i = intadmin] \& [S_j = depro]$

then 规则输出 *yes*，

系统状态 $v' = (b, M, f', H)$

$f' = f \setminus f_l \leftarrow f_l \cup f_l'(O_k) = I_x$

else 规则输出 *no*，

系统状态保持不变。

其中，*depro* 表示专门用来降低客体最低修改安全级的进程。

规则说明：只有完整性管理员角色通过特定进程才能降低客体的最低修改安全级。其他改变客体安全级的请求按规则 2 处理。

规则 5　give-power 表示授予另一角色对客体的访问权限。

请求描述 $(RO_\lambda, RO_i, O_k, x) \in R^5, x \in A$

if $[O_{f(k)} \in b(RO_\lambda, w)] \& [O_k \in b(RO_\lambda, x)]$

then 规则输出 *yes*，

系统状态 $v' = (b, M', f, H)$

$M' = M \setminus M_{ik} \leftarrow \{x\}$

else 规则输出 *no*，

系统状态保持不变。

规则说明：当满足下列条件时，角色 RO_λ 可以授予另一

角色 RO_i 对客体 O_k 的访问权限。

· 角色 RO_λ 拥有对节点 O_k 父节点 $O_{f(k)}$ 的写访问权限；

· 角色 RO_λ 拥有对节点 O_k 的相应访问权限。

规则 6　resciend-power 表示撤销另一角色对客体的访问权限。

请求描述 $(RO_\lambda, RO_i, O_k, x) \in R^5, x \in A$

if $[O_{f(k)} \in b(RO_\lambda, w)] \& [O_k \in b(RO_\lambda, x)] \& [f_r(RO_\lambda) > f_r(RO_i)]$

then 规则输出 *yes*,

系统状态 $v' = (b, M', f, H)$

$M' = M \backslash M_{ik} - \{x\}$

else 规则输出 *no*,

系统状态保持不变。

规则说明：当满足下列条件时，角色 RO_λ 可以撤销另一角色 RO_i 对客体 O_k 的访问权限。

· 角色 RO_λ 拥有对节点 O_k 父节点 $O_{f(k)}$ 的写访问权限；

· 角色 RO_λ 拥有对节点 O_k 的相应访问权限；

· 角色 RO_λ 的完整级绝对支配角色 RO_i 的完整级。

规则 7　get-read-excute 表示请求读或执行一个客体。

请求描述 $(RO_i, S_j, O_k, x) \in R^2$

if $x = r | x = e$

then 规则输出 *yes*,

系统状态 $v' = (b', M, f, H)$

$b' = b \bigcup b(RO_i, S_j, o_k, x)$

else 规则输出 *undefiend*,

系统状态保持不变。

规则说明：本模型不对系统的只读或执行访问进行限制，为了模型的完整，设置该规则。

4　模型的安全性分析

RBICM 模型是一个有限状态机模型。如果前一状态是安全状态，经过 RBICM 模型的规则转变后的状态也是安全状态，则可说明这一转换规则是安全的。因此，模型正确性证明就是证明一个安全的状态 v 经过 RBICM 模型的 7 条规则的转换后形成的状态 v' 依然能够满足 RBICM 模型的完整性公理，即下述定理成立。

定理 1　若状态 v 满足 RBICM 公理，则由 RBICM 模型任一规则得到的状态 v' 均满足 RBICM 模型的完整性公理。

证明：根据规则的条件直接进行验证。

5　模型应用实例

某组织机构中有局长、处长、科长、科员 4 个角色，完整级分别为 I_1, I_2, I_3, I_4。假定科员甲欲到财务部门报销其差旅费，则科员甲需要首先创建一个报销申请，请科长签字证明出差事实，请处长签字确定支出帐目，请局长批准，最后到财务部门报销，而财务部门只有看到局长的批准后才能为其办理报销手续。用本模型处理，科员创建申请文件，完整级别为 I_1，最低修改完整级为 I_2，这样可以防止其他科员篡改本文件。而科长可以对文件进行修改，科长批准后，文件完整级别为 I_2，最低修改完整级为 I_3。同理，局长批准后文件完整级别为 I_4，最低修改完整级为 I_h。而财务部门只需查看文件完整级是否为 I_4 就可以确定是否为其报销，同时局长审批后的文件将无法修改。再假定科员要直接给局长递交报告，则可以将创建的报告文件最低修改完整级为 I_4，这样可以避免科长、处长修改报告。

除此之外，该模型还可以防止完整性特洛伊木马对信息的破坏。假设某科员在系统中放置了木马，试图盗用局长的角色为自己的虚假报销文件签名。即使局长没有发现木马的存在，并且木马也使用局长角色完成了签名，但因为木马程序未经认证，所以签名文件的完整性级别为木马的完整级别，仍为系统最低完整级别 I_{low}，这份虚假申请无法取得财务部门的认可。

6　模型的实现

在 Linux 内核版本为 2.6.18 的 fedora core 6 系统上实现了基于角色的完整性控制模型的原型系统。

在系统中增加了对角色的定义和管理，并根据角色的职责指定其完整级。通过设置用户-角色关系表，指定用户可以登录的角色。同时，为系统中的所有可执行文件设置完整级，对系统中基础程序进行严格测试，确认无误后赋予系统最高完整级。对经常使用的较成熟的应用程序赋予相对较高的完整级，对使用较少、对其行为了解不多的应用程序赋予较低的完整级，对临时下载程序赋予系统最低完整级。具体完整级的确定与程序的功能、复杂程度、潜在使用者有关。

模型的控制部分通过 Linux 系统的 LSM 框架实现。当进程试图请求以写方式打开文件时，激发本模型规则，根据规则的条件做出相应的判断，并更新系统状态。

在模型的原型实现中也发现了一些问题，主要就是进程间的调用问题。比如在 Linux 中，所有由命令生成的进程都是由用户的 shell 进程产生的。再比如脚本操作是由系统的基础命令对应的进程来完成的。此外，应用程序中也存在调用动态库或者其他应用程序的行为。在设计模型过程中，为了简化模型设计，没有考虑进程间调用问题。实现过程中，通过在系统中设置进程，调用链，可以追溯每一个访问客体的进程的上级进程直至启动操作的角色，取调用链中所有进程完整级的最小值作为模型中进程的完整级，这样就可以避免特洛伊木马通过调用其他程序破坏系统完整性的行为。

结束语　与机密性模型相比，完整性模型中完整级的概念并不明确，缺乏现实世界中的实例与之对应，造成应用上的困难。本文从信息系统对完整性的需求出发，指出完整性需求是由使用者对信息可信程度进行评估而产生的，还指出了影响完整性级别的两个重要因素，使用者和对信息执行存取操作的进程，提出了使用角色对使用者完整级进行管理，给出了进程完整性评估方法。在此基础上，提出了一个完整性控制模型，给出了完整性公理、模型元素、系统状态、系统设置和模型的状态转换规则，并对模型进行了证明，给出了模型的应用实例和实现方法。

下一步的工作将研究 RBICM 与机密性模型的统一，力争建立一个融机密性、完整性于一体的操作系统安全模型，并应用在本单位开发的安全操作系统原型系统中。

参 考 文 献

[1]　Anderson J P. Computer Security Technology Planning Study VolumeII[R]. MA, USA: Air Force Systems Command, 1972

（下转第 210 页）

每一个语句串作为一个处理单元进行初始化处理。将语句串利用分词程序进行分解，形成对应的词语序列串，从词语概念库提取每一个词语对应的概念符号串，把存在多个义项的词语和包含有关键性信息的词语（包含领域概念基元符号的词语）挑选出来，进入下一步匹配计算阶段。

3.2 匹配计算

将多义项词语的概念符号串分解为多个单一义项的概念基元符号串，这些概念基元符号串与包含信息的概念基元串进行逐层匹配计算（利用同行概念关联式进行计算），或者通过领域范围内给出的交式概念关联式来进行匹配计算。如果中间存在匹配的义项概念串，则形成义项概念串的匹配序列，进入下一步消岐处理阶段；如果这些义项概念符号串中不存在匹配的义项概念符号串，则进入统计算法计算模式，通过统计数据给出该义项概念在该词语中所出现的概率值，通过统计算法给出它的置信权值。由于这一部分不是本文研究的重点，作者在此不过多阐述。当然这一步骤也可以相对应地调整，可以先采用统计算法计算出对应的结果，只有在置信度低于一定的数值时才采用规则的方法进行义项的匹配计算。

3.3 消岐处理

提取对应的义项概念串进行排列，同时进行必要的全语句串的一个验证过程，使得整个语句的概念符号串符合 HNC 的句类结构，从中得到最后的“句类－概念－词语”结果，实现语句级别的语义深层结构。至此，输出对应的词语义项的概念符号串结果，整个消岐算法过程结束。

4 实验及其结果

表 1 新闻语料测试结果

多义词	义项数	统计准确率	本文准确率	提高百分比
底子	6	83.25	92.64	9.39
拉扯	6	84.67	96.51	11.84
左右	5	81.62	91.05	9.43
门户	5	84.48	97.16	12.68
扑腾	5	82.94	92.38	9.44
点子	5	80.69	92.59	11.9
后身	5	78.26	85.57	7.31
昏天黑地	5	84.78	98.37	13.59
当头	4	81.68	92.18	10.5
机关	4	83.72	93.62	9.9
开放	4	82.95	96.48	13.53
出世	4	81.63	93.85	12.22
关闭	4	84.65	94.38	9.73
调理	4	82.64	93.67	11.03
分散	4	83.45	98.37	14.92
站住	4	84.16	92.28	8.12
狠心	4	82.39	93.42	11.03
教育	4	82.57	96.29	13.72
到底	4	80.27	93.67	13.4
沐浴	4	82.71	97.64	14.93

实验选取了 20 个汉语文本中经常出现的多义词作为测试对象，从新闻语料中随机找了 1011 个包含上述多义词的语句进行测试，采用上面所述的基于概念关联式的词义排岐方法在句子中进行消岐，验证了本文提出的方法的有效性。实验测试的 20 个多义词如表 1 所列。

实验获得了较好的结构及平均将近 94.11％的准确率，比单纯用统计方法进行词义消岐的结果（82.68％）平均提高了 11.43 个百分点，提升效果还是令人满意的。

我们发现，如果在进行词语义项激活之前，准确给出语句所属文章对应的领域符号串，则消岐准确率在此基础上更能得到进一步的提高，这说明了篇章或语句的领域信息对词语义项排岐的重要作用，也为下一步准确率的提高指明了方向。

结束语 综上所述，词语义项的多义排岐，是对自然语言理解处理所面临的一个研究难点也是重点之一，它的准确率在很大程度上影响着其它领域诸如机器翻译、中文信息检索等的研究成果。本文利用 HNC 理论概念基元符号体系中概念的关联式，从多义词语所蕴含的义项的概念符合串入手，结合统计方法和规则方法各自的优势，证实了这种方法为多义词的义项排岐探索了一条行之有效的途径。

下一步的工作将从丰富和完善概念的交式关联式方面入手，提高利用词语概念对语句领域判断的准确率，把它形式化后添加到初始化处理中，使得在做义项匹配计算之前能够得到对应的领域指导，以进一步提高消岐的正确率。

参 考 文 献

[1] 陈笑蓉，秦进. 基于最大熵原理的汉语词义消岐[J]. 计算机科学，2005，32(5)：174-176

[2] 王广正，王喜凤. 基于知网语义相关度计算的词义消岐方法[J]. 安徽工业大学学报，2008，25(1)：71-75

[3] 缪建明，张全. 词语义项的概念激活方法[C]//第九届汉语词汇语义学研讨会论文集. 新加坡：COLIPS PUBLICATAION，2008：191-197

[4] 黄曾阳. HNC（概念层次网络）理论[M]. 北京：清华大学出版社，1998

（上接第 90 页）

[2] Biba K J. Integrity considerations for secure computer systems [R]. MA, USA: Air Force Systems Command, 1977

[3] Clark D D, Wilson D R. A Comparison of Commercial and Military Computer Security Policies[C]// Proceedings of the IEEE Symposium on Security and Privacy. CA, USA: IEEE Computer Society, 1987: 184-194

[4] 季庆光. 高安全级操作系统形式设计的研究[D]. 北京：中国科学院研究生院，2004

[5] Bell D E. Modeling the multipolicy machine[C]// Proceedings of the 1994 Workshop on New Security Paradigms. CA, USA: IEEE Computer Society, 1994: 2-9