ODRL权利描述语言逻辑实施机制研究

钟 勇[1,2,4] 秦小麟[4] 刘凤玉[2,3]

(佛山科学技术学院信息与教育技术中心 佛山 528000)[1]

(南京理工大学计算机科学与技术博士后流动站 南京 210094)[2]

(南京理工大学计算机科学与技术学院 南京 210094)[3] (南京航空航天大学信息安全研究所 南京 210016)[4]

摘 要 针对缺乏正式语义使基于XML的ODRL等权利描述语言的确切含义依赖应用程序的特定理解、易产生二义性和不确定性等问题,将ODRL语言转换成一种基于逻辑的权利描述语言,为ODRL语言提供了正式语义和策略实施的逻辑框架,为ODRL语言实施提供可信和形式化分析基础。在ODRL规范基础上对转换方法进行了说明和示例,最后描述了实施平台。

关键词 权利描述语言,ODRL语言,数字权利管理

中图法分类号 TP309

Logical Implementation Mechanism for ODRL Rights Expression Language

ZHONG Yong[1,2,4] QIN Xiao-lin[4] LIU Feng-yu[2,3]

(Information and Educational Technology Center, Foshan University, Foshan 528000, China)[1]

(Postdoctoral Mobile on Computer Application, Nanjing University of Science and Technology, Nanjing 210094, China)[2]

(School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China)[3]

(Institute of Information Security, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[4]

Abstract The absence of formal semantics of ODRL rights expression language and the others that are based on XML makes the exact meaning of these languages rely heavily on the specific interpretation of application programs, which brings the ambiguous and uncertain problems. The paper transited the ODRL language to a logic-based rights expression language, which brings a formal semantics and a logic implementation framework to the language, and also a foundation of trust and formal analysis to the language. Based on the ODRL specification, the methods of transition were discussed and exampled, and an implementation platform was also described.

Keywords Rights expression languages (REL), Open digital rights language (ODRL), Digital rights management (DRM)

1 引言

随着Internet和数字化技术的快速发展,特别是P2P技术的发展,网上交易和传播的数字内容越来越多。由于数字的易复制、修改和传播性,盗版和侵权问题也日益严重。传统的以访问控制为主的安全保护技术只能限制非法用户的访问,对于合法用户获取数字内容后的非法复制和传播造成的盗版、侵权和敏感信息泄密,传统的访问控制技术往往无能为力。

数字权利(版权)管理(DRM)技术是近年来发展起来的信息安全技术,它通过数字权利管理和防复制技术实现对数字内容的保护,维护版权人合法权益。数字权利管理的核心在于对数字资源的权利保护和使用控制。国际上许多著名计算机公司和研究机构也纷纷推出各自的系统和产品,如Micorsoft WMRM, IBM EMMS, Adobe Content Server和InterTrust DigiBox等[1]。

在DRM系统中,许可证(license)是重要的组成部分,包含数字资源的使用权限、密钥、元数据等关键信息。权利描述语言(REL)用来构造许可证,描述数字资源或服务的使用权利,是数字权利保护的重要研究课题。目前的权利描述语言主要包括基于XML的语言(如XrML(http://www.xrml.org))、ODRL(http://www.ordl.net)、MPEG-21 REL、OMA(Open Mobile Alliance)DRM-REL和OeBF(Open e-Book Forum)REL等。基于XML的权利描述语言发展较为完善,已处于实用阶段。但权利描述语言要求开放性、灵活性、可扩展性以及支持各类使用控制描述使基于XML的语言难于满足要求。

(1) 当使用条件变得复杂时,基于XML的REL语言的

到稿日期:2008-05-13 本文得到中国博士后科学基金(20070421015),国家自然科学基金(60673127),国家863计划(2007AA01Z404)等资助。

钟 勇(1970-),男,博士后,副教授,主要研究方向为信息安全、数据库技术、数字版权保护技术等,E-mail:zhongyong@fosu.edu.cn;秦小麟(1953-),男,博士生导师,教授,主要研究方向为安全数据库、空间数据库、时空数据库、信息安全等;刘凤玉(1943-),女,教授,博士生导师,主要研究方向为网络性能保护和信息安全。

语法变得复杂和难以理解。

(2) 缺乏正式语义使基于 XML 的 REL 语言的确切含义严重依赖特定理解,并容易产生二义性和不确定性。如近来有学者证明 XrML 和 MPEG REL 等语言的评价(evaluation)算法并不能保证完全的可终止性[2]。缺乏正式语义也使基于 XML 的 REL 语言的安全实施缺乏可信性和形式化分析基础。

(3)缺乏权限协商和交互机制。如无法表达许多有用的版权法相关的权限要求[3]。

针对基于 XML 的权利描述语言缺乏正式语义的问题,Pucella 等[4]通过将 ODRL 的一个子集转换成一阶逻辑的方法增加 ODRL 语言的正式语义;Halpern[5]等针对 XrML 语言提出一个类似的方法;Garcia[7]等提出使用 Web 拓扑结构表达 ODRL 的正式语义,但他们的方法都只是逻辑概念的表达,并不具备实施的可行性;Holzer[8]等提出使用自动机来表达 ODRL 语言的正式语义,但他们的方法只是使用自动机来表示不同的简单条件,无法表达复杂权限和复合条件。针对基于 XML 的权利描述语言缺乏资源权限协商和交互的表达机制问题,Arnab 等[9]提出在 ODRL 语言中扩展 Request 等语句,使 ODRL 语言能表达双向交流语义并用于协商合理使用权限;Arnab[10]也将该方法扩展于 XrML 语言,做了有益的探索,但仍无法满足真正开放性权限管理要求。

本文在我们提出的分布式使用控制和权利描述语言(LucScript 语言)[11]的基础上,实现 ODRL 语言与 LucScript 语言转换和互操作机制。本文的研究目的包括:

(1) 由于 LucScript 不仅包含正式的逻辑语义,而且包含相应的逻辑实施框架。将 ODRL 语言转换成 LucScript,为 ODRL 提供正式逻辑语义,并提供了 ODRL 权利保护策略实施的逻辑框架,形成 ODRL 权利保护策略表达和实施的统一逻辑框架,为 ODRL 权利保护策略实施提供可信和形式化分析基础。

(2) 扩展 ODRL 语言的功能,LucScript 语言不仅包括 DRM 领域的权利描述功能,而且包括丰富的使用控制等语义。通过将 ODRL 语言转换成 LucScript 语言,能扩展 ODRL 语言的使用控制语义。

(3) 增加 DRM 语言的互操作性和用户的接受度。由于现有的 DRM 领域中不同的机制使用不同的权利描述语言,而 DRM 领域中权利描述语言的标准化仍无法做到,通过建立各类 DRM 语言的互操作性可以提高 DRM 机制的用户接受度。

2 LucScript 分布式使用控制和权利描述语言

逻辑语言由于在表达力、灵活性和语义完整性上的优势,基于逻辑的 REL 语言的研究受到重视。但现存的逻辑语言存在一些问题,如 LicenseScript 语言[3]规则不统一、权限管理不开放、缺乏表达授权决策持续性和数字内容实时动态使用控制等语义能力。LucScript(Logic-based Usage Control License Script)分布式使用控制和权利描述语言是我们提出的一种基于逻辑框架的语言,该语言基于 Active-U-Datalog[12]逻辑。其具有触发功能的授权机制起源于我们前期提出的集中式使用控制授权框架 LUC[13],具有较强表达力、灵活性和开放式权限管理能力等。

Active-U-Datalog 是一种结合主动规则、具有可更新能力的 Datalog 程序,其谓词原子包括表示插入和删除的更新原子 $\pm p(t_1,t_2,\cdots,t_n)$。通过 Active-U-Datalog 的主动规则,权利描述语言能描述环境条件变化产生的自适应规则,具有表达授权决策持续性和授权主客体属性可变性(mutability)下进行实时动态使用控制的能力。

LucScript 语言中许可证 *lic* 是四元组 $\{D, IDB, AR, BV\}$,其中 D 是许可证所保护的数字内容的唯一识别符,IDB 是内涵规则集,AR 是触发规则集,BV 是表示为 $name=value$ 形式的属性绑定形式。称 $P=IDB\cup AR\cup BV$ 为许可证程序部分。许可证表示成 $lic(D,\Delta)$,其中 Δ 是许可证程序标识符。图 1 是许可证示例。

```
(1)License ( //许可证示例
(2)  e_film_star_war, //许可证数字媒体标识
(3)  { play(D)← counter(n),n≥1,update_counter();
(4)  update_counter()←counter(n),n = n1 +1,-counter(n),+counter
     (n1); }, //IDB
(5)  { -total_counter(n),+total_counter(m)←total_counter (n),m=n+1,
     +play_a(D) }, //AR,
(6)  { counter=50;total_counter=0';version='11.6.1'; expire='07/12/
     31' ; type='use' }, //BV
(7)  )
```

图 1 使用许可证示例

图 1 语句(2)说明该许可证管理的数字媒体标识。语句(3)-(4)是 *IDB* 部分,语句(3)说明只有当计数器不小于 1 才能播放该媒体并使用语句(4)进行计数器更新。语句(4)将计数器的值减 1,其中 $-counter(n)$ 表示删除计数器的当前值,$+counter(n_1)$ 表示插入新的计数值。语句(5)是主动规则,该规则在用户进入播放状态时触发(由 $+play\_a(D)$ 更新谓词触发),将总计数值 *total_count* 加 1,其中 $-total\_counter(n)$ 表示删除现在的总计数值,$+total\_counter(m)$ 表示插入加 1 后的计数值。通过该触发规则,许可证可记录用户的总使用次数。语句(6)是许可证外延绑定部分,如 *counter* 谓词说明该许可证允许播放媒体 50 次。

LucScript 许可证使用逻辑程序调用谓词进行交互和更新。逻辑程序调用谓词如表 1 所列,是我们提出的一种处理逻辑事务的特殊谓词。设事务 T,Δ 是许可证程序 $P=IDB\cup EDB\cup AR$ 的标识符,程序调用谓词包括表 1 的两类谓词。

表 1 调用谓词

谓词形式	谓词的使用
call(T) (Ⅰ型)	确定调用该谓词的逻辑程序是否可满足 T,如果 T 是可满足的且引发的更新是一致的,谓词返回 true,否则返回 false
call(Δ,(T)) (Ⅱ型)	确定逻辑程序 Δ 是否可满足 T,如果可满足且 T 引发的更新对程序 Δ 来说是一致的,谓词返回 true,否则返回 false

增加调用语义后,许可证具有自拆分能力。如假定在图 1 的许可证中增加下列 *IDB* 规则:

$$clone(lic(D,\Delta))\leftarrow create(lic(D,\Delta)),call(\Delta,(+IDB(r))),IDB(r) \quad (1)$$

其中 $create(lic(D,\Delta))$ 创建许可证 $lic(D,\Delta)$,$IDB(r)$ 是许可证结构谓词,代表许可证中的 *IDB* 规则;$call(\Delta,(+IDB(r)))$ 是Ⅱ型调用谓词,在程序 Δ 中执行 $+IDB(r)$ 插入。整个规则创建临时许可证 $lic(D,\Delta)$ 并将图 1 许可证中的 *IDB*

规则全部插入到该临时许可证中，通过这种方法，可以拆分和创建新许可证。

3 ODRL 语言转换成 LucScipt 语言

ODRL 是基于 XML 的权利描述语言[①]，ODRL 语言主体包括权利所有人(*rights holders*)和用户(*end users*)。〈*party*〉标签用于说明权利所有人，数据对象称为 *assets*。ODRL 包含权利所有人的权利要求标签〈*offers*〉和权利双方同意遵守的权利协议标签〈*agreements*〉。在 ODRL 语言中，*permission* 是在数据对象*assets* 执行一定活动 *activities* 的权利，活动 *activities* 的例子如 *play*，*print*，*display* 和 *execute* 等。〈*permission*〉标签由限制标签〈*constraints*〉和需求标签〈*requirement*〉说明。限制标签〈*constraints*〉说明用户在执行一定操作前必须满足的条件，需求标签〈*requirement*〉说明用户在执行一定操作前必须采取的额外操作(如付费等)。ODRL 语言基础模型如图 2 所示。

```
<rights>
    <context>
      <uid>...</uid>
    </context >
    <offer>
      <asset> ...</asset>
      <permission>
        <permission-type>
          <requirement>...</requirement>
          <constraint>...</ constraint>
        </permission-type>
        <condition>...</condition>
      </permission>
      <party>
        <context>...</context>
        <rightsholder>...</rightsholder>
      </party>
    </offer>
    <agreement>
      <context> ...</context >
      <party>...</party>
      <permission>...</permission>
      <asset>...</asset>
    </agreement>
</rights>
```

图 2　ODRL 基础模型

3.1 总体转换

〈*rights*〉是 ODRL 的根元素，〈*rights*〉包括〈*offer*〉和〈*agreement*〉等子元素。为简单起见，假定 ODRL 语言中只包括单个数字内容(*asset*)的权利管理。

总体上，将 ODRL 的 *offer* 实体转换为 LucScript 中的原始许可证(权利所有人的最初许可证)，*agreement* 实体转换成 LucScript 中的使用许可证。*asset* 实体标识(*uid*)作为许可证媒体标识，*permission* 实体转换为许可证规则，*constraints* 实体转换为许可证的绑定(外延)谓词。图 3 所示的 ODRL 权利描述，可转换成图 4 所示的 *LucScript* 许可证形式。在图 4 的许可证中，为了区分不同的绑定谓词名，我们使用分层的谓词名表示 ODRL 中的分层结构，如谓词 *play.hardware.uid* 绑定 *play* 操作所需要的硬件 *id*。图 4 中的规则(3)说明在播放数字内容之前需要检查当前客户机的机器标识与许可证绑定的硬件标识是否相等。其中 *get_hardwareid* 是系统调用谓词，获取当前客户机的机器标识。

```
<rights>
    <agreement>
      <asset>
        <context>
          <uid> e_film_star </uid>
        </context >
      </asset>
      <permission>
        <play>
          <constraint>
            <hardware>
              <context>
                <uid> pl </uid>
              </context>
            </hardware>
          </constraint>
        </play>
      </permission>
    </agreement>
</rights>
```

图 3　ODRL 示例

```
(1)License ( //许可证示例
(2)e_film_star_war, //许可证数字媒体标识
(3)  { play()←play. hardware. uid(hd1),get_hardwareid(hd2),hd1 = hd2;},
     //IDB
(4)  { }, //AR,
(5)  { play. hardware. uid≡pl; type≡'use' }, //BV
(6)  )
```

图 4　ODRL 示例的 LucScript 许可证形式

3.2 ODRL 授权(permission)模型转换

ODRL 的授权实体如图 5 所示，包括 *usage*，*reuse*，*transfer*，*asset management* 4 个虚拟实体部分。

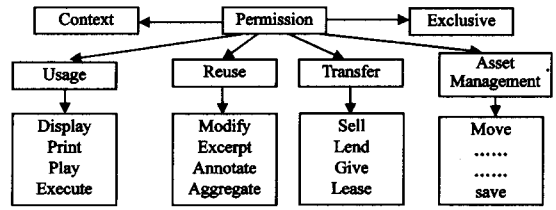

图 5　ODRL 授权模型

Exclusivity 属性指出权限是否限制在指定部件，*Context* 实体用于指定权限集的标识。*Permission* 实体通过 *agreement* 实体与 *asset* 实体发生联系，*Permission* 实体自身可包含多个 *Constraints*，*Conditions* 和 *Requirements* 实体。*Permission* 实体转换成 *LucScript* 许可证 *IDB* 规则中的规则头，包含的 *Constraints* 等实体作为规则体内容。转换过程如图 6 所示。

```
<permission>
    <display/>
    <print>
        <constraint>...</constraint>
        <conditions>...</conditions>
        <requirements>...</ requirements>
    </print>
</permission>
```

⇩

display()←

print()←<constraint>, <conditions>, <requirements>

图 6　ODRL 授权模型的转换

① 本文的 ODRL 相关文字和图例均来自 ODRL 标准规范(http://www.odrl.net/1.1/ODRL-11.pdf)。

3.3 ODRL 限制(constraint)模型转换

ODRL 限制模型如图 7 所示,用于限制相关权限的使用。限制实体转换成 LucScript 许可证中的外延绑定谓词,并且产生相应的系统谓词,获取当前限制状态进行对比。如对 CPU 实体,LucScript 有相应的 *get_cpu* 系统谓词获取当前 CPU 标识进行对比。对 *Count* 等实体,还需要产生触发规则进行更新。如图 8 所示,该示例说明 *display* 操作限制在固定 CPU,*print* 操作限制 5 次,*play* 操作限制在每星期内最多 10 次(ODRL 日期和时间使用 ISO8601 标准,P7D 表示 7 天时间)。

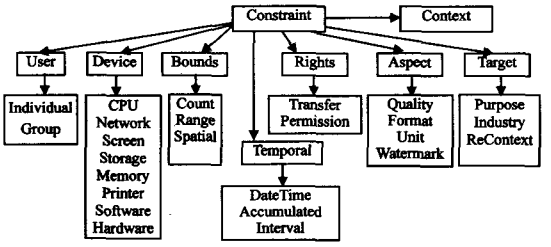

图 7 ODRL 限制模型

图 8 的 ODRL 限制模型转换的 LucScript 许可证如图 9 所示,其中对 Count 等实体的更新操作转换成触发规则(第 6,7 行)[②]。

```
<display>
    <constraint>
        <cpu> BFEBFBFF00000F34 </cpu>
    </constraint >
</display>
<print>
    <constraint>
        <count>5 </count>
    </constraint >
</print>
<play>
    <constraint>
        <interval> P7D </interval>
        <constraint>
            <count>10 </count>
        </constraint>
    </constraint >
</play>
```

图 8 ODRL 限制模型示例

图 9 许可证中,第 3 行规则在使用 display 操作之前需要验证当前 CPU 是否与许可证绑定的 CPU 一致。第 4 行规则说明执行打印操作必须许可证的可打印次数大于零。第 5 和第 6 行规则分别判断 2 个日期是否是在同一周内。第 7 行规则说明如果当前播放日期与上次播放日期(*play. date*)在同一周内且可播放次数大于零的话,则允许播放操作。第 8 行规则说明如果当前播放日期与上次播放日期不在同一周内则允许播放。第 9 行触发规则说明在每次打印后将可打印次数减 1。第 10 行触发规则说明如果当前播放日期与上次播放日期在同一周内则将可播放次数减 1。第 11 行触发规则说明如果当前播放日期与上次播放日期不在同一周内则将可播放次数重新设置为 10 且将上次播放日期设置为当前日期。

(1)License //许可证
(2)(e_film_star, //asset 标识
(3) { display()←get_cpu(c_1),cpu(c_2),$c_1 = c_2$;
(4) print()←print. count(n),n>0;
(5) same_week(d_1,d_2)←$d_2 \geqslant d_1$, $n = d_2 - d_1$, $n < 7$;
(6) not_same_week(d_1,d_2)←$d_2 \geqslant d_1$, $n = d_2 - d_1$, $n \geqslant 7$;
(7) play()←play. count(n),n>0,date(d_1),play. date(d_2),same_week(d_2,d_1);
(8) play()←date(d_1),play. date(d_2),not_same_week(d_2,d_1) }, //IDB
(9) { −print. count(n),+print. count(n_1)←+print_a(),print. count(n),$n_1 = n-1$;
(10) −play. count(n),+play. count(n_1)←+play_a(),date(d_1),play. date(d_2),same_week(d_2,d_1),play. count(n),$n_1 = n-1$;
(11) −play. count(n),+play. count(10),−play. date(d_2),−play. date(d_1),←+play_a(),date(d_1),play. date(d_2),not_same_week(d_2,d_1),play. count(n) }, //AR,触发规则
(12) { cpu≡'BFEBFBFF00000F34'; print. count≡5; play. count≡10; play. date≡ * * / * * / * * },//BV
(13))

图 9 ODRL 限制模型转换成 LucScript 许可证示例

3.4 ODRL 需求(requirement)模型

ODRL 需求模型如图 10 所示,指的是在获取相关操作允许前必须获得的预条件集。每个 requirement 实体可转换成一条 IDB 规则,并将规则头或相应的标志加入相应的动作实体规则中,如图 11 所示的 ODRL 需求模型。该模型说明在每次使用媒体时需要支付 20 $ AUD(外加 10%的税率)。

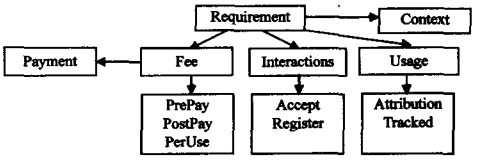

图 10 ODRL 需求模型

图 11 所示的需求模型转换的 LucScript 许可证如图 12 所示,该图第 3 行是需求规则,该规则从许可证费用中减去支付费用[③],第 4 行规则要求每次使用前必须支付费用。

```
<play>
    <requirement>
        <peruse>
            <payment>
              <amount currency="AUD">20.00</amount>
              <taxpercent code="GST">10.0</taxpercent>
            </payment>
        <peruse>
    </requirement>
</play>
```

图 11 ODRL 需求模型示例

(1)License //许可证
(2)(e_film_star, //asset 标识
(3) { play. payment()←rate(r),tax_percent(t),n=r+r * t,money(n_1),$n_1 \geqslant$ n,−money(n_1),$n_2 = n_1 - n$,+money(n_2);
(4) play()←play. payment() }, //IDB
(5) { }, //AR,触发规则
(6) { money≡ * ,currency≡'AUD',rate≡20.00,tax_percent≡0.10},//BV
(7))

图 12 ODRL 需求模型转换成 LucScript 许可证示例 1

图 12 所示的许可证的付款方式是每次使用前付款。如果是一次性预付款,可以使用相应的标识位表示,如图 13 所

② 当然,更新操作也可以转换成 IDB 规则中的更新操作。

③ 简单起见,本文的示例均未使用 LucScript 中的支付模式,而只是简单地使用 *money* 绑定代表用户的钱包。

示，该许可证说明在一次性预付款 1000 元，则可以不受限制地播放数字内容。

```
(1)License //许可证
(2)(e_film_star, //asset 标识
(3)  { play. payment()←money(n1), n1≥1000, −money(n1), n2 = n1 −
     1000, +money(n2), −prepay(false), +prepay(true);
(4)    play()←prepay(true) }, //IDB
(5)  { }, //AR,触发规则
(6)  { money=＊, currency='AUD', prepay=false}, //BV
(7))
```

图 13　ODRL 需求模型转换成 LucScript 许可证示例 2

图 13 的第 3 行规则说明，如果用户拥有的金额数不小于 1000，那么支付成功并将 *prepay* 标志设置为 *true*。第 4 行规则说明如果 *prepay* 标志为 *true* 的话，则用户可以播放数字内容。

3.5　ODRL 条件(condition)模型转换

ODRL 条件模型说明当条件满足时相应的操作不再允许。条件模型如图 14 所示。

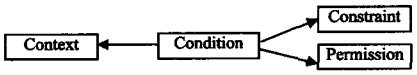

图 14　ODRL 条件模型

条件实体转换成 LucScript 许可证中的外延绑定谓词，并且产生相应的系统谓词，获取当前限制状态进行对比。如果不相等，则允许操作。图 15 所示的条件模型，说明不允许使用软件 RealPlay 播放对象，以及在澳大利亚不允许播放和销售该对象。该模型转换的 LucScript 许可证如图 16 所示。

```
<permission>
    <sell/>
    <play>
        <condition>
            <constraint>
                <software> RealPlay </software>
            </constraint>
        </condition>
    </play>
    <constraint>
        <spatial>
            <context> <uid>iso3166:AU</uid> </context>
        <spatial>
    </constraint >
</permission>
```

图 15　ODRL 条件模型

```
(1)License //许可证
(2)(e_film_star, //asset 标识
(3)  { sell()←get_spatial(s1), spatial (s2), s1<>s2;
(4)    play()←get_software(t1), software(t2), t1<>t2, get_spatial(s1),
     spatial (s2), s1<>s2}, //IDB
(5)  { }, //AR,触发规则
(6)  { play. oftware='RealPlay', spatial='iso3166:AU''
(7)  },//BV
(8))
```

图 16　ODRL 条件模型转换成 LucScript 许可证示例

图 16 许可证中，*get_software* 系统调用谓词返回当前使用的软件，*get_spatial* 系统调用谓词返回当前使用许可证所在的位置。规则(3) 说明如果当前所在的位置不是澳大利亚，则允许销售。规则(4)说明如果当前所在的位置不是澳大利亚且所使用的软件并非 RealPlay，则允许播放该软件。

3.6　ODRL 容器(container)模型转换

ODRL 语言支持 3 种将实体聚合成容器的关系：(1) And 关系。必须支持所有实体；(2) Exclusive Or 关系。仅支持单一实体；(3) Inclusive Or 关系。支持单一或多个实体。

图 17 所示是 And 关系转换成单一规则。

```
<play>
    <constraint>
        <container type="and">
            <cpu/>
            <storage/>
        </container>
    </constraint>
</play>
⇩
play()←<cpu>, <storage>
```

图 17　And 关系转换

图 18 所示是 Inclusive Or 转换成多条规则。

```
<play>
    <constraint>
        <container type="in-or">
            <cpu/>
            <storage/>
        </container>
    </constraint>
</play>
⇩
play()←<cpu>
play()←<storage>
```

图 18　Inclusive Or 关系转换

图 19 所示是 Exclusive Or 关系转换成带有负外延原子的多条规则。

```
<play>
    <constraint>
        <container type="ex-or">
            <cpu/>
            <storage/>
        </container>
    </condition>
</play>
⇩
play()←<cpu>, ¬<storage>
play()←¬<cpu>, <storage>
```

图 19　Exclusive Or 关系转换

由于 LucScript 语言的负原子只使用在外延原子和调用原子中，对内涵原子可以使用外延原子作为标志符的方法实现。图 20 的示例，分别采用标记外延谓词 prepay 和 peruse 作为采取该方法付款的标记。用户在采取且只采取一种方式付款后，相应的标记位设置为 true，数字内容才能被使用。

```
<play>
    <requirement>
        <container type="ex-or">
            <prepay>
                <payment> ..... </payment>
            </prepay>
            <peruse>
                <payment> .....</payment>
            </peruse>
        </container>
    </ requirement >
</play>
⇩
play()←prepay(true), peruse(false), ...
play()←prepay(false), peruse(true), ...
{ prepay=false; peruse=false } //BV
```

图 20　使用标记谓词的 Exclusive Or 关系转换

ODRL 的其他模型如 sequence 模型[④]、context 模型、

Rights Holder 模型、Security 模型等可使用类似的方法实现，在此不再详叙。对于 ODRL 的权限撤消 Revoke 模型，由于 LucScript 许可证是一种具有自我更新和增量更新能力的动态许可证，其许可证规则和谓词绑定的更新易于实现，因而 ODRL 的 Revoke 模型易于转换，在此也不再详述。ODRL 的链接(linking)模型和继承(inheritance)模型由于牵涉到多个数字内容实体的保护，而目前 LucScript 暂未考虑单一许可证的多内容实体的保护方法，在此暂不讨论。

4 转换示例

ODRL 规范中提出 8 个 ODRL 的应用场景，下面将部分场景转换成 LucScript 许可证形式。为简单起见，一些 XML 的描述语句和解释语句因纯粹表示为 LucScript 许可证的绑定描述，未做转换和说明。

4.1 Ebook 场景＃1

Corky(作者)和 Addison(插图作者)通过出版社 EBooksRus 出版图书，他们希望该图书售价 20 $ AUD 外加 10%的税，只能在单个 CPU 使用且最多只能打印 2 份，图书的开始 5 页能够免费观看。购书款的 60%属于作者，10%属于插图作者，30%属于出版社。该场景属于〈*offer*〉实体，产生如图 21 的原始许可证。

```
(1)License //许可证
(2)(urn:ebook. world/999999/ebook/rossi-000001, //asset 标识
(3)  { display()←get_page_no(n),n≤5;
(4)    display()←get_page_no(n),n>5,prepay(true);
(5)    print()←get_cpu(c1),cpu(c2),c1= c2,prepay(true),print. count(n),
       n<2;
(6)    pay(CPU,Money)←－prepay(false),＋prepay(true),cpu(c),－cpu(c),
       ＋cpu(CPU),rate(r),tax_percent(t),Money=r+r*t,distribute_money(Money);
(7)    distribute_money(Money)←ToA=Money*0. 6,ToI=Money*0. 1,ToP=Money*0. 3,set_money(ToA,ToI,ToP);
       }, //IDB
(8)  { －print. count(n),＋ print. count(n1)←＋print_a(),print. count(n),n1=n+1 }, //AR,触发规则
(9)  { asset. name≡'Why cats sleep and we don't'; currency≡'AUD'; rate≡20. 00; tax_percent≡0. 10; cpu≡nil; prepay≡false; type='origin'; print. count≡0;author≡'x500:c=AU;o=RightsDir;cn=CorkyRossi'; illustrator≡'x500:c=AU;o=RightsDir;cn=AddisonRossi'; publisherr≡'x500:c=AU;o=RightsDir;cn=EBooksRUS'; },//BV
(10))
```

图 21 E-Book 场景＃1 的 LucScript 原始许可证

图 21 中规则(3)说明图书的前 5 页能够免费观看，其中 *get_page_no* 是系统谓词，得到当前的页码号。规则(4)说明在支付费用后才能观看除前 5 页外的其他页码。规则(5)说明在当前 CPU 与许可证绑定的 CPU 相同且已支付费用的情况下，如果已打印次数小于 2 次，则可以打印。规则(6)是费用支付规则，在支付费用时将使用方的 CPU 绑定，并将预付款标识 *prepay* 设置为 *true*。变量 CPU，Money 分别表示用户 CPU 和支付费用数，Money 必须等于数字内容价格(含税)。规则(7)将支付费用分配到各个权利所有方。触发规则(8)记录用户的打印次数。

4.2 Ebook 场景＃2

假定用户 Mary Smith 购买该电子书，在图 21 由〈*offer*〉实体产生的原始许可证(*type*≡'*origin*')基础上，用户付款后执行规则(6)，产生图 22 所示的使用许可证。

```
(1)License //许可证
(2)(urn:ebook. world/999999/ebook/rossi-000001, //asset 标识
(3)  { display()←get_page_no(n),n≤5;
(4)    display()←get_page_no(n),n>5,prepay(true);;
(5)    print()←get_cpu(c1),cpu(c2),c1= c2,prepay(true),print. count(n),
       n<2;
(6)    pay(Money)←－prepay(false),＋prepay(true),rate(r),tax_percent(t),Money=r+r*t;}, //IDB
(7)  { －print. count(n),＋ print. count(n1)←＋print_a(),print. count(n),n1=n+1 }, //AR,触发规则
(8)  { asset. name≡'Why cats sleep and we don't'; currency≡'AUD'; rate≡20. 00; tax_percent≡0. 10; cpu≡'Adobe-WebBuy:CPD-ID:ER-393939-Dss-787878'; prepay≡true; type='use'; print. count≡0; consumer. id≡'urn:ebook. world/999999/users/msmth-000111'; consumer. name≡'Mary Smith';},//BV
(9))
```

图 22 E-Book 场景＃2 的 LucScript 使用许可证

图 22 的许可证已绑定用户的 CPU 并假定用户已付款(*prepay*≡*true*)。

4.3 Ebook 场景＃3

场景＃3 包括两个 *agreement* 实体，一部分是关于图书销售公司有权销售至多 5000 份图书份额。另一部分是关于用户 John Doe 的使用许可，其授权有效期从 2001 年初至 2004 年底，他有权在 30 整天内观看该书并在指定的可信打印机上打印 5 份该书拷贝，在其它打印机上每星期最多可从 1 到 100 页中打印 5 页，总打印次数不超过 100 页，每星期也能摘录 5000 字节到内存(剪贴板)，但总数不能超过 1,000,000 字节。1 年后，John Doe 可将该书转让他人。场景＃3 的两个 agreement 实体分别转换为图 23 和图 24 所示的销售公司和最终用户许可证。

```
(1)License //许可证
(2)(urn:ebook. world/999999/voucher/2001/1234567890,//标识
(3)  { sell()←sell. count(n),n<5000; }, //IDB
(4)  { －sell. count(n),＋sell. count(n1)←＋sell_a(),sell. count(n),n1=n+1 }, //AR,触发规则
(5)  { asset. id≡'isbn:872-2345-981' asset. name≡'XML:Ananager's Guide'; owner. id≡'http://distributors. net/registry/xyz'; owner. name≡'XYZ Company'; owner. role≡'marc:dst'; sell. count≡5000; type≡'use' },//BV
(6))
```

图 23 E-Book 场景＃3 的出版商 LucScript 许可证

图 23 中规则(3)说明已销售数小于 5000 时才能销售，触发规则(4)在每次销售前将已销售数加 1。

```
(1)License //许可证
```

④ 序列模型，部分详细内容和转换示例可参照下列技术报告：http://www. fosu. edu. cn/zhongyong/LucReport5. pdf，http://www. fosu. edu. cn/zhongyong/LucReport2. pdf

```
(2)(urn:ebook.world/999999/voucher/2001/1234567890,//标识
(3) { valid()←Date(d),issue_date(d1),d≥d1,expire(d2),d≤d2;
(4)   same_week(d1,d2)←d2≥d1, n=d2−d1,n<7;
(5)   not_same_week(d1,d2)←d2≥d1, n=d2−d1,n≥7;
(6)   display()←valid(),display.days(n),n>0;
(7)   display()←valid(),Date(d1),display.date(d2),d1=d2;
(8)   print(P)←valid(),trustprinter.id(q),P=q,trustprinter.count(n),n>0;
(9)   print(P)←valid(),trustprinter.id(q),P<>q,printer.date(d1),date
      (d2),same_week(d1,d2),printer.count(n1),n1>0,printer.total-
      count(n2),n2>0,get_page_no(n),n≤100;
(10)  print(P)←valid(),trustprinter.id(q),P<>q,printer.date(d1),date
      (d2),not_same_week(d1,d2),printer.totalcount(n2),n2>0,get_
      page_no(n),n≤100;
(11)  excerpt(N)←valid(),excerpt.date(d1),date(d2),same_week(d1,
      d2),excerpt.count(n1),N≤n1,excerpt.totalcount(n2),N≤n2;
(12)  excerpt(N)←valid(),excerpt.date(d1),date(d2),not_same_week
      (d1,d2),excerpt.totalcount(n1),N≤n1;
(13)  give()←date(d1),give.date(d2),d1≥d2}, //IDB
(14) { −display.days(n),+display.days(n1),−display.date(d2),+dis-
      play.date(d1)←+display_a(),Date(d1),display.date(d2),d1<>d2,
      n1=n−1;
(15)  −trustprinter.counter(n),+trustprinter.counter(n1)←+print_a
      (P),trustprinter.id(q),P=q,trustprinter.counter(n),n1=n−1;
(16)  −printer.count(n),+printer.count(n1),−printer.totalcount(m),+
      printer.totalcount(m1)←+print_a(P),trustprinter.id(q),P<>q,
      printer.date(d1),date(d2),same_week(d1,d2),printer.count(n),n1
      =n−1,printer.totalcount(m),m1=m−1;
(17)  −printer.count(n),+printer.count(4),−printer.totalcount(m),+
      printer.totalcount(m1),−printer.date(d1),+printer.date(d2)←+
      print_a(P),trustprinter.id(q),P<>q,printer.date(d1),date(d2),
      not_same_week(d1,d2),printer.count(n),printer.totalcount(m),m1
      =m−1;
(18)  −excerpt.count(n),+excerpt.count(n1),−excerpt.totalcount(m),
      +excerpt.totalcount(m1)←+excerpt(N),excerpt.date(d1),date
      (d2),same_week(d1,d2),excerpt.count(n1),n2=n1−N,excerpt.to-
      talcount(m),m1=m−N;
(19)  −excerpt.count(n),+excerpt.count(n1),−excerpt.totalcount(m),
      +excerpt.totalcount(m1),−excerpt.date(d1),+excerpt.date(d2)←
      +excerpt(N),excerpt.date(d1),date(d2),not_same_week(d1,d2),
      excerpt.count(n),n1=5000−n excerpt.totalcount(m),m1=m−N;
(20)  }, //AR,触发规则
(21) { user.id≡'http://people.net/registry/john-doe-9999'; user.name≡
      'John Doe'; issue_date≡'2001-01-01 T00:00:00'; expire≡'2004-
      12-31 T23:59:59'; sell.count≡5000 display.days≡30; display.date
      ≡''; trustprinter.id≡'guid:TrustPrint/4747474742222'; trust-
      printer.count≡5;printer.date≡''; printer.count≡5; printer.total-
      count≡100;.excerpt..count≡5000; excerpt.totalcount≡1000000;
      excerpt.date≡''; give.date≡'202-01-01 T00:00:00'; },//BV
(22))
```

图 24　E-Book 场景 #3 的用户 LucScript 许可证

图 24 的规则(3)说明当前日期在有效期内，其中 *date* 是系统调用谓词，返回当前时间。规则(4)和(5)分别判断 2 个日期是否是在同一周内。规则(6)说明在有效期内如果可观看的天数大于零，那么可以观看该书。规则(7)说明在有效期内如果在同一天内已观看过该书，则允许该日继续观看。规则(8)说明在有效期内如果在可信打印机上的可打印次数大于零，那么可以在可信打印机上打印。规则(9)和(10)分别说明在任意打印机上每星期可打印 1 至 100 页中任意的 5 页，总的打印不超过 100 页，其中 *get_page_no* 谓词获取当前打印的页码。规则(11)说明在同一周内摘录总字数不超过该周可摘录字数和总的可摘录字数的情况下可摘录该书。规则(12)说明如果与上次摘录日期不在同一周且摘录的字数总计不超过总可摘录字数情况下，可以摘录该书。规则(13)说明一年后该书可转让他人。触发规则(14)在用户对非当前日期观看时，将可观看天数减 1 并重新设置当前观看日期。触发规则(15)在可信打印机上打印之前将可打印次数减 1。触发规则(16)对同一周内在非可信打印机打印之前，分别将该周可打印次数和总可打印次数减 1。触发规则(17)对非同一周在非可信打印机打印之前，将总可打印次数减 1 并设置该周的可打印次数和打印日期。触发规则(17)对同一周内对数字内容进行摘录前，分别将该周可摘录次数和总可摘录次数减去将摘录的字数。触发规则(18)对非同一周内对数字内容进行摘录前，将总可摘录次数减去将摘录的字数并设置该周的可摘录字数和摘录日期。

5　LucScript 语言对 ODRL 语言的转换

由于 LucScript 语言具有超出 ODRL 语言的表达力，因此 LucScript 只有部分语义能够转换成 ODRL 语言。LucScript 语言的许可证转换成 ODRL 语言之前必须遵循下列规则：

(1) LucScript 许可证谓词名称与 ODRL 语言的对应标签名称保持一致，具有可转换性；

(2) LucScript 许可证采用分层谓词名表示 ODRL 中的分层结构，如谓词 *play.hardware.uid* 绑定 *play* 操作所需要的硬件 *id*；

(3) LucScript 许可证表达的语义范围在 ODRL 可表达的范围内。

LucScript 许可证转换成 ODRL 语言是上述转换的逆过程，在此不再详述。

6　LucScript 数字版权保护管理平台

LucScript 数字版权管理平台是我们正在研制的基于逻辑语言的应用平台。该平台的逻辑部分由具有分层负文字分析能力的 Datalog 解释器 Desl. 2.0 经过改进后重新实现在 Visual Prolog 中。该平台主要实现架构如图 25 所示。

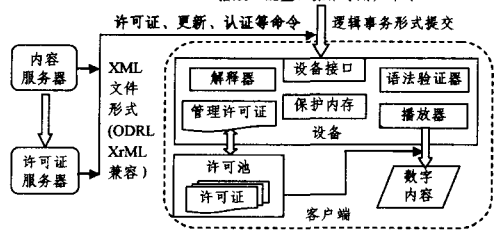

图 25　LucScript 数字版权管理平台实现原型

内容服务器是存储数字内容的内容仓库，实现对数字内容的打包分发，并将数字内容密钥和原始许可证发送给许可证服务器。许可证服务器生成 LucScipt 许可证，发送到客户端。协议形式采取本文提出的权利描述语言，以 XML 文件的形式交流，兼容 ODRL 或 XrML 语言。在客户端，设备接

(下转第 178 页)

```
56.  memleak_error9++;
57.  free(p);
58.  }
59.  void f9(){
60.   int *q=NULL;
61.   int* memleak_error10=NULL;
62.   memleak_error10=(int *)malloc(100);
63.   q=(int *)memleak_error10;
64.   free(q);
65.  }
```

程序 2　被测程序范例

表 3　测试结果

泄漏内存申请时的变量	替换前	替换后
memleak_error1	Y	N
memleak_error2	Y	N
memleak_error3	Y	N
memleak_error4	Y	N
memleak_error5	Y	N
memleak_error6	Y	N
memleak_error7	Y	Y
memleak_error8	Y	N
memleak_error9	Y	N
memleak_error10	Y	N

下面的函数中每个 memleak_error 变量都有内存泄漏，如果将/* */中的语句替换成之前的语句，则内存泄漏故障就都消除了。在测试结果表(如表 3 所列)中分别列出了每个 memleak_error 变量在程序替换前和替换后的测试结果。其中"Y"表示报出了故障，"N"表示没有报出故障。

从测试结果看出，替换前的 10 个故障点全部检测出来，而替换后的 10 个非故障点有一个误报，误报率是 10%。因此可以看出本项目的检测效果还是非常好的。

结束语　本文提出基于区间运算的内存泄漏静态的检测模型，并对其检测效果进行了测试，测试结果表明该模型能较为准确地检测出方法内的内存泄漏故障，而基于区间运算的静态检测能有效地减少漏报率和误报率。但目前系统还仅限于方法内的故障检测，不能做方法间内存泄漏的检测，因此在现实的检测中有一定的限制，需要进一步的改进。

参 考 文 献

[1] 宫云战. 软件测试的故障模型[J]. 装甲兵工程学院学报，2004，18(2)：1-5

[2] 宫云战. 软件测试[M]. 北京：国防工业出版社，2006

[3] 肖庆. 内存泄漏的一种静态分析方法[J]. 装甲兵工程学院学报，2004，18(2)：23-26

[4] https://javacc.dev.java.net/

[5] 王德人，张连生，邓乃扬. 非线性方程的区间算法[M]. 上海：上海科学技术出版社，1987

[6] 周高嵌. 基于白箱测试的源代码在线评测系统[D]. 北京：北京化工大学，2005

(上接第 139 页)

口负责接收许可证、更新、认证等服务器端的命令和本地用户的播放、设备配置等命令，并转换成逻辑程序的事务(查询)命令。客户端的许可证存放在许可池中，许可池本身由管理许可证规则管理。用户播放数字内容的过程如下：用户首先向设备提出播放命令，由设备接口将该命令转换成相应的逻辑事务并进入该许可证的使用控制过程，否则拒绝该命令。

通过本文的转换方式，内容服务器、版权服务器和版权控制器之间的交互可使用 ODRL 且与相关使用 ODRL 的 DRM 系统进行互操作。

结束语　缺乏正式语义使基于 XML 的 ODRL 等 REL 语言的确切含义严重依赖特定理解，并容易产生二义性和不确定性。本文在我们提出的分布式使用控制和权利描述语言 LucScript 语言的基础上，实现 ODRL 语言与 LucScript 语言转换和互操作机制。由于 LucScript 不仅包含正式的逻辑语义，而且包含相应的逻辑实施框架。将 ODRL 语言转换成 LucScript，为 ODRL 提供正式逻辑语义，并提供 ODRL 权利保护策略实施的逻辑框架，形成 ODRL 权利保护策略表达和实施的统一逻辑框架，为 ODRL 权利保护策略实施提供可信和形式化分析基础。

下一步将对 XrML 语言与 LucScript 语言的转换和互操作机制进行研究。

参 考 文 献

[1] 俞银燕，汤帜. 数字版权保护技术研究综述[J]. 计算机学报，2005，28(12)：1962-1968

[2] Becker M Y，Fournet C，Gordon A D. Design and Semantics of a Decentralized Authorization Language// 20th IEEE Computer Security Foundations Symposium (CSF). 2007：3-15

[3] Chong C N，Corin R，et al. LicenseScript：a logical language for digital rights management. Annales des Telecommunications，2006，61(3/4)：284-331

[4] Pucella R，Weissman V. A Formal Foundation for ODRL // Proc. Workshop on Issues in the Theory of Security. 2004

[5] Halpern J，Weissman V. A formal Foundation for XrML// Proc. 17th IEEE Computer Security Foundations Workshop. 2004：251-265

[6] Holzer M，Katzenbeisser S，Schallhart C. Towards a Formal Semantics for ODRL // Proc. 1st International Workshop on ODRL. 2004：137-148

[7] Gunter C A，Weeks S T，Wright A K. Models and languages for digital rights // Proc. of the 34th Annual Hawaii International Conference on Systems Sciences. Maui，Hawaii，2001：4034-4038

[8] Garcia R，Gil R，Gallego I，et al. Formalizing ODRL Semantics using Web Ontologies// Proc. 2nd Intl. ODRL Workshop. 2005：1-10

[9] Alapan A，Andrew H. Extending ODRL to Enable Bi-Directional Communication// Proceedings of the 2nd International ODRL Workshop. Lisbon，Portugal，2006：43-52

[10] Alapan A，Andrew H. Extending ODRL and XrML to Enable Bi-directional Communication. Technical Report CS04-28-00. Cape Town，South Africa：Department of Computer Science，University of Cape Town，2004

[11] Zhong Y，Zhu Z，Lin D M，et al. A Method of Fair Use in Digital Rights Management// Proc. of the 10th International Conference on Asian Digital Libraries. LNCS 4822. Hanoi，Vietnam，2007：160-164

[12] Bertino E，Catania B，Gori R. Active-U-Datalog：integrating active rules in a logical update Languages// Lecture Notes in Computer Science. 1998：107-133

[13] 钟勇，秦小麟，郑吉平，等. 一种灵活的使用控制授权语言框架[J]. 计算机学报，2006，29(8)：1408-1418