基于XML的Web应用模型抽取

程广金[1,2] 缪淮扣[1] 方明科[1,3] 梅 佳[1,2] 高洪皓[1]
(上海大学计算机工程与科学学院 上海 200072)[1] (上海市计算机软件评测中心 上海 201112)[2]
(信阳师范学院计算机与信息技术学院 信阳 464000)[3]

摘 要 以模型检验为目标,从时间的约束角度出发,提出一种基于XML文档的Web应用的模型抽取方法。模型抽取由时间及相关链接的提取、模型构造和结果显示3部分组成。首先,通过对Web应用进行逆向分析,从带时间约束的XML源代码对链接及时间约束等相关信息进行提取、规整和存储。其次,对Web应用中的链接、时间约束等建模元素进行分析,应用映射与聚合等抽象技术对获得的信息进行重构,得到适合于形式化验证的时间自动机(TA, Timed Automata)模型,并对时间约束下的并发进行模型组合。最后,以电子邮箱系统为实例阐述如何实现模型抽取。

关键词 XML文档,时间约束,模型抽取,时间自动机
中图法分类号 TP311 **文献标识码** A

Extracting Model of Web Application Based on XML

CHENG Guang-jin[1,2] MIAO Huai-kou[1] FANG Ming-ke[1,3] MEI Jia[1,2] GAO Hong-hao[1]
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)[1]
(Shanghai Key Laboratory of Computer Software Evaluating & Testing, Shanghai 201112, China)[2]
(School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China)[3]

Abstract For model checking, an approach to extracting TA models of Web application based on XML document with time constraint was proposed. The extraction process is divided into three phases: time and link extraction, model construction and display. Firstly, analyzed Web application reversely to extract, structure and store information related to link and time constraint from XML source code. Then, analyzed elements for model construction such as link and time constraint, and restructure the information obtained with mapping and aggregation technology. Finally the TA(Timed Automata) model applying for formal checking was obtained. A particular case study, a mail box system is taken to illustrate the method to be feasible.

Keywords XML documents, Time constraint, Model extraction, Timed automata

在Internet时代,Web应用给企业发展和人们生活带来了巨大改变,且正发挥着日益重要的作用。人们通过Internet和Intranets可以获取越来越多的服务和信息,Web应用程序也变得越来越复杂。然而迫于市场需求和竞争的压力,许多Web应用没有进行充分的验证和测试就投入了运营,导致其质量难以保证。检验Web应用模型是减少Web应用的缺陷及保证和提高Web应用质量和可靠性的重要手段。XML[1]作为一种新的、Web上的数据交换标准,扮演了极其重要的角色。

Web应用系统日益复杂和庞大,且频繁变化,通过分类、映射等抽象表示,可以建立针对Web应用某方面特征的模型[2]。这些抽象表示为Web应用的建模和性质抽取奠定了基础。因此,分析Web应用程序代码,建立Web应用模型,是模型抽取中的重要手段之一。这种逆向抽取模型方法可以用于验证和确认Web应用模型的正确性和有效性,并可为Web应用的维护与更新提供必要的信息。

关于Web应用程序的抽取的研究,Ricca等人[3]开发了一个网站分析工具ReWeb。该工具通过网站的页面与链接进行分析,从而对站点的维护与更新提供支持。但只对纯HTML格式的页面进行分析,且未考虑后续验证的问题。Lucca等人[4,5]设计了一种用于对Web应用进行逆向工程的工具WARE,但未对链接进行分析,且产生的结果不用形式化语言描述。文献[6]给出了一种链接提取流程。该流程主要针对URI文法,改进了链接提取,但是相关的改进还是仅限于纯HTML格式的页面,且对提取出的链接未做进一步的分析与处理。文献[7]提出了一种用分层的有限状态机对Web应用的行为进行建模的方法,但这种方法也不是从程序代码来实现抽取的,对页面之间的依赖关系使用的是UML扩展模型,而不是形式化模型,也不能直接用于验证。文献[8]提出了一个XML的子集,用于实时分布式多媒体系统的

到稿日期:2010-10-12 返修日期:2011-01-27 本文受国家自然科学基金项目(60673115,60970007),国家重大基础研究(973)项目(2007CB310800),上海市自然科学基金(09ZR1412100),上海市科委项目(10510704900),上海市重点学科建设项目(J50103)资助。
程广金(1986—),女,硕士生,主要研究方向为软件工程,E-mail:chenggj@shu.edu.cn;缪淮扣 男,教授,博士生导师,主要研究方向为软件工程、软件形式方法,E-mail:hkmiao@shu.edu.cn(通信作者);方明科 男,硕士,讲师;梅 佳 男,博士生;高洪皓 男,博士生。

规格说明，但没有关于时间约束的系统模型的抽取。文献[9]采用逆向工程的思想从源代码进行分析，抽取了链接等相关信息，但没有涉及到时间约束的问题。

本文基于时间约束研究，以 XML 文档形式的 Web 应用程序进行模型抽取，获取适合于模型检验的时间自动机(TA)模型。本方法在借鉴传统方法的基础上，利用逆向工程思想提取 Web 应用的链接关系和时间约束等信息，分析和抽取 Web 页面对应的 XML 文档中时间约束下的状态的变化信息，构建出适用于形式化模型检验的 TA 模型。通过实例研究阐述本方法的可行性。

1 模型抽取框架

模型抽取过程描述如下：针对具体的 Web 应用，从应用程序页面的源代码开始收集信息，采用深度优先算法遍历 Web 应用程序的 XML 文档。抽取出与时间和链接相关的信息后，按照一定格式存储这些信息。接着重构信息组织方式，建立 Web 应用的 TA 模型，为后续的模型检验提供基础。

如图 1 所示，模型抽取主要包括链接和时间约束信息提取、模型构造和结果呈现等步骤。首先利用 XML 解析器将 Web 应用的 XML 代码解析为一定的结构模型，如树状模型；采用提取算法提取、收集结构模型中的链接和时间约束信息；信息加工模块包括数据的过滤、转换等操作，过滤、规整已提取的链接信息和时间信息等。信息存储模块是将加工后的信息存储在一定的数据结构中，最终将存储的信息构造为形式化的 TA 模型。

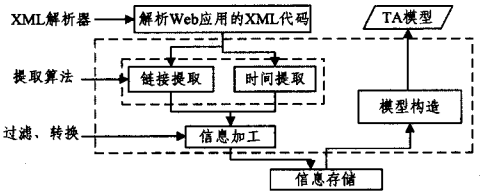


图 1 模型抽取的过程

2 信息的提取

Web 应用通过 Web 服务器站点来部署 Web 信息页面、视频音频文档和组件，通过超链接方式将这些资源彼此进行关联。模型抽取工作是从 Web 应用程序的 XML 代码中抽取页面和组件间的链接关系及组织结构以及时间约束条件。

XML(eXtensible Markup Language)是一种元标记语言，作为一种新的 Web 上的数据交换标准，其扮演了极为重要的角色。XML 是面向内容的，且它不是一种具体的标记语言，没有固定的标记符号，是一种用来定义标记的元标记语言。它允许用户自己定义一套适宜应用的 DTD(Document Type Definition)，所以具有更多的语义、良好的可扩展性、灵活性、简单易用、自描述等特点，而且特别适用于 Web 上的半结构数据，并可用于数据交换，正成为数据组织和交换的事实标准。同时，大量的 XML 也迅速地出现在了 Web 上。要将 XML 文件中的信息以某种形式显示出来，如通过浏览器显示，则可引用一个样式表文件来定义浏览器怎样显示 XML 文件中的信息。Web 设计人员不仅能创建文字和图形，而且能构建文档类型定义的多层次、相互依存的系统、数据树、元数据、超链接结构和样式表。

1)XML 中的链接

Web 页面上不仅包含大量的文本信息，也包含很多的超链接，如文字、图像、视频等。而且同一个站点的多个页面之间也通过超链接关联在一起。所以抽取 Web 应用的信息，不仅仅是抽取 Web 应用的某一个特定的独立的页面，也要将关联这些页面的链接抽取出来，以便抽取整个 Web 应用的相关信息，因此主要从链接等方面抽取。

XML 链接语言[10](即 XLink)使用 XML 语法来定义资源之间的链接。XLink 类似于 HTML 文档中的超链接，只是 XLink 功能更强大，能够定义双向的甚至更复杂的链接。XLink 包括两种链接：简单链接和扩展链接。简单链接指的是单向链接，扩展链接描述两个或多个资源之间的一组单向或双向链接。定义命名空间后，需给出 XLink 的两个属性：xlink:type 和 xlink:href。

XPointer 用于标识 XML 文档中特定位置的语法。从某方面来讲，XPointer 与 HTML〈a〉锚标签中的 # name 引用有相同的功能，允许用户直接导航到文档中的特定位置，但是不需要特定的标签。

2)XML 中的时间约束

随着 Internet 的发展和技术的进步，现在的 Web 应用系统越来越广泛，而且有时间要求。时间系统又称实时系统(Real-time System)，是指在确定的时间内能够完成特定功能并且对外部和内部的随机事件做出及时响应的计算机应用系统。时间约束不仅涉及计算结果逻辑上的正确性，而且要与其在什么时候输出计算结果相关联。一方面对外部事件的响应必须在一定时间内完成，同样要求各种输出也必须在一定时间内完成。另一方面，一个实时系统的负荷可能是不均匀的，必须满足一定的峰值负荷要求。自引入时间系统概念后，时间自动机已广泛应用于各类系统，如航天飞机的自动控制、核电站的安全监控、智能高速公路的监控、空中交通管制以及国防应用领域等，Web 应用系统也不例外。一般来说，这些对时间要求严格的系统，或是实时系统[11]；或是为了用户信息的安全性，Web 应用程序往往会由于系统长时间没有操作而会自动退出的系统。

例如在银行系统中，假如服务未在规定时间段内收到一笔资金转款，将会导致很大的经济损失。一个 Web 页面通常是很复杂的，对应的 XML 代码也比较复杂。如下 XML 代码片段描述的是 Web 应用程序的时间约束，Session 表示超时的时间约束条件，以秒为单位。

```
〈session-config〉
  〈timeout〉10〈/timeout〉
〈/session-config〉
```

其中，session-config 包含一个子元素 timeout 标签，定义了 Web 的 session 的有效时间是 10s，即一旦延迟时间超过 10s，服务器将拒绝服务，系统回到初始状态。对于带有时间约束的系统，在小于 timeout 的时间内可进行一系列的操作；当时间大于等于 timeout 时，系统会自动执行一些其他操作。

3)提取算法

为了抽取出时间自动机模型，抽取的链接和操作也是与时间约束相关的。解析器作为 XML 文档处理的一个基本部分，首先检查 XML 文档的良构性，其次解析 XML 文档的内容以对处理软件可用。XML 解析器有很多，目前主要使用 DOM(文档对象模型)和 SAX(Simple API for XML，XML 简单 API)解析器。本文使用 DOM 解析树访问和处理 XML 文

档，将 XML 文档解析为一个树图。深度优先遍历该树图的每一个节点，将遍历的节点放在一个堆栈中。当回溯到的节点是根节点时，对已出栈的节点进行分析、规整。

提取算法描述如下。

```
Input:A DOM Tree-Chart,action,xlink,xpoint,PageNode,session
Output:location,transition,guard,time
String[] AllLocations;//定义操作和状态的全局变量
String[] AllXLinks
Parser1=CreateParser(DOM);//创建 DOM 解析器
  Parser. findElement(DOM);//解析 XML 中的元素
StartLocation=initPage(initLocation) //将 XML 中初始的页面装换为初始的状态
LocationStack←rootNode;
LocationStack←StartPage;//将初始状态放在 location 栈中
char * p=StartPage;
while(! LocationStack) //当栈中有元素时
{
  do{
    if(p→child) //如果节点有子节点，则将它的子节点入栈
      p= p→child;
    else //若无子节点，则将给节点退栈，并回溯到它的父节点
        {pop(p);
          p=p→parent;
  }while(p! =rootNode);//当回溯到根节点时，将之前出栈的元素抽取
  Parser(Elemnts poped from stack)
}
While(LocationStack! =Null)
{ Location←PageNode;
    if(session) //当有 session 元素时
      guard←time;//时间为守卫条件
    While(Link∨Action∨session) //存在链接、操作或时间约束时
    PageNode=PageNode—>next;//发生迁移
    Location=Location—>next;//当遍历到动作和时间时，发生迁移
    structured(location);//规整与时间约束相关的状态及迁移
    Related(Info(action));
      }
```

4)信息存储

利用上述算法遍历的信息存储在如图 2 所示的栈中。栈是一种后进先出的数据结构，当遍历到某一元素没有子节点时，遵循后进先出的原则，将该数据（如图 2 中 e_{21}）从栈中取出，并回溯到该节点的父节点（e_{11}）。若父节点还有其他的子节点，将下一个子节点（e_{22}）进栈。当节点 e_{11} 的所有子节点都没有不被遍历的，将 e_{11} 退栈，回溯到 e_0，并访问 e_0 的其他子节点。当栈顶元素是 e_0 时，分析和加工抽取的已出栈的数据信息，过滤掉那些不正确、不合理或与时间约束无关的一些数据，优化信息，然后将其存储在字符串变量等数据结构中。

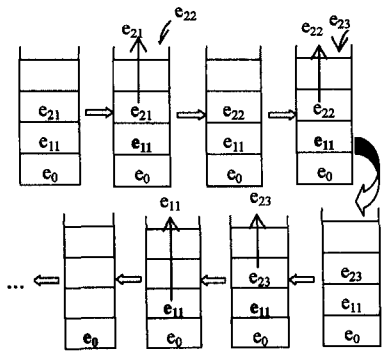


图 2　抽取的信息的存储

3　模型构造

通过上述过程的抽取，Web 信息可能比较复杂繁琐，甚至会出现一些与后续的验证无关的信息。需过滤掉这些冗余的信息，简化模型。抽象主要是划分系统中的多个状态，实现状态集合之间的映射，以得到一个保留了部分原始系统行为和特征的模型，是简化 Web 应用模型的有效手段之一。对于 Web 应用的逆向工程而言，模型抽象是对页面链接、操作、时间以及它们之间的关联的抽象。

页面链接和操作等 Web 页面内的实体可以被建模成具有属性和行为的对象[12]。Web 页面在运行时，这些页面都具有一定的状态，且通过状态的迁移完成一些功能。可将这种带有时间约束的 Web 应用模型用时间自动机[13,14]来表示。

有限状态自动机能够很方便地描述有限状态系统，但是在描述与时间相关的状态变化的系统时却显得无能为力。一般的自动机不能够表示系统中某一个状态的持续时间，也不能表示状态在哪个时间点发生迁移。例如自动售票系统、电子商务系统、火车交叉口交通灯控制和激光手术等越来越关注时间限制，这些时间苛求系统进行某些操作时需要适时地做出响应。

带有时间约束的有限状态自动机又称为时间自动机（Timed Automata，TA）。TA 是带有时钟集合的有限状态自动机，用于对有限状态实时系统的建模。时钟用于测量系统处于某一状态上时流逝的时间。随着时间的流逝，系统处在某一状态，或者发生迁移。迁移是不需要时间的，也就是说迁移是即时的。时钟值只有满足了时间约束，迁移才可能发生。时钟的不变量（invariants）限制了在某一个状态时的时间量。时钟值和时钟的不变量是关于时间的约束。

定义 1（时间约束）　假设 C 是一个时钟集合，其中 x 是一个时钟，c 是一个自然数常量，在集合 C 上的时间约束满足以下规则：

（1）$x<c$ 及 $x\leqslant c$ 是时间约束；

（2）如果 α 是时间约束，那么 $\neg\alpha$ 也是时间约束；

（3）如果 α,β 是时间约束，那么 $\alpha\wedge\beta$ 也是时间约束；

（4）其他情况都不是时间约束。

约束常常表示为时间常量和时钟值的相比，用 *Constraint*(C)表示集合 C 上的时间约束的集合。模型检查的可判定性问题不能对分数做出判断，如果 c 出现分数形式，我们可以将其乘以分母的最小公倍数而转换为整数，以后时钟约束中用到的常量均取自于 N。

定义 2（时间自动机）　时间自动机用一个八元组来表示，即 $TA=(Loc,Loc_0,Act,C,\rightarrow,Inv,AP,L)$，其中

1) Loc 是有限的位置集合；

2) $Loc_0\subseteq Loc$ 是初始化所处的位置；

3) Act 是有限动作的集合；

4) C 是有限时钟；

5) $\rightarrow\subseteq Loc\times CC(C)\times Act\times 2^C\times Loc$ 是状态/位置迁移关系；

6) $Inv:Loc\rightarrow CC(C)$ 是不变量赋值函数；

7) AP 是有限的原子命题的集合；

8) $L:Loc\rightarrow 2^{AP}$ 是状态/位置的标记函数。

令 $ACC(TA)$ 代表那些或是督导条件或是状态/位置不变式上的原子时钟限制的集合。时间自动机的边用元组（g，

a,D)表示，其中 g 是一个时钟限制，a 是一个动作并且 $D\subseteq C$ 是时钟的集合。公式 $l\xrightarrow{g,a,D}l'$ 表明一旦时间限制 g 被满足，时间自动机就可以从一个状态/位置向另外一个状态/位置迁移，同时任何 D 中的时钟将被重值为 0，通常用 $reset(D)$ 表示。函数 Inv 将每一个状态/位置赋予一个局部的不变量，用来描述自动机可以在该状态/位置停顿的时间，如 $Inv(l)$ 限制在 l 状态/位置最大消耗的时间。也就是说，在不变量 $Inv(l)$ 无效之前，必须离开 l 状态/位置。如果此时没有向外迁移的传递，那么流程将不可能再进一步，这可能导致终止，称为时间锁 timelock。

定义 3(时间语义)　时间的流逝可能引起状态变化，也可能不发生变化(时间自动机上相应的位置不变)。为了区分状态和位置，设状态 s 是$\langle l,v\rangle$位置和时间的二元关系。时间自动机上有两种时间语义：

(1)对于状态$\langle l,v\rangle$和非实数值的时间增量 $\delta\geqslant 0$，如果 $\forall\delta,0\leqslant\delta'<\delta,v+\delta'$ 满足 $Inv(l)$，则有$\langle l,v\rangle\xrightarrow{\delta}\langle l,v+\gamma\rangle$，将其称为延迟传输。

(2)由于位置迁移引起的状态改变(相应的时间自动机中的所有时钟值不变)：对于状态$\langle l,v\rangle$和一个迁移$\langle l,a,\theta,\lambda,l'\rangle$，使得$\langle l,v\rangle\xrightarrow{a}(l',v[\lambda:=0])$迁移上的动作触发，将其称为离散传输。

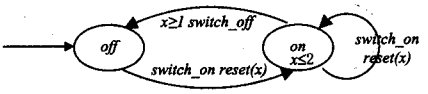


图 3　信号灯切换模型

在用于轨道交通控制的信号灯切换模型上，时间约束所描述的状态空间为 $S=\{\langle off,t\rangle\mid t\in IR_{\geqslant 0}\}\cup\{\langle on,t\rangle\mid t\in IR_{\geqslant 0}\}$，其中 t 是在 v 上的时间解释 $v(x)=t$，$IR_{\geqslant 0}$ 是非负数集，如图 3 所示。在初始化状态 $s=\langle off,0\rangle$：

$\langle off,t\rangle\xrightarrow{d}\langle off,t+d\rangle$ for all $t\geqslant 0,d\geqslant 0$

$\langle off,t\rangle\xrightarrow{switch}\langle on,0\rangle$ for all $t\geqslant 0$

$\langle off,t\rangle\xrightarrow{d}\langle off,t+d\rangle$ for all $t\geqslant 0,d\geqslant 0$

$\langle on,t\rangle\xrightarrow{d}\langle off,t+d\rangle$ for all $t\geqslant 0,d\geqslant 0$ 且 $t+d\leqslant 2$

从$\langle off,0\rangle$出发的可达状态集合为 $S=\{\langle off,t\rangle\mid t\in IR_{\geqslant 0}\}\cup\{\langle on,t\rangle\mid 0\leqslant t\leqslant 2\}$。在任何状态$\langle on,t\rangle$上的不变量 $t\leqslant 2$，并且 $t>2$ 是不成立的。

定义 4(组合时间自动机)　M_i 是 $TA(Loc_i,Act,C_i,\rightarrow,Loc_0,Inv_i,AP_i,L_i)$，其中 $i=1,2,C_1\cap C_2=\Phi$，M_1 与 M_2 的同步组合 $M_1||M_2$ 是时间自动机，$(Loc_1\cup Loc_2,Loc_{01}\times Loc_{02},Act_1\cup Act_2,C_1\cup C_2,\rightarrow,Inv,AP_1\cup AP_2,L)$，其中→要满足以下定义的迁移关系：

①当 $a\in Act_1\cap Act_2$ 时，$(Loc_1,Loc_2)\xrightarrow{a,a_1\wedge a_2,C_1\cup C_2}(Loc_1',Loc_2')$；当操作 a 只属于 Act_1 或 Act_2 时，那么分别只有状态 Loc_1 或 Loc_2 发生迁移；

②$Inv(Loc_1,Loc_2)=Inv(Loc_1)\wedge Inv(Loc_2)$，即组合状态的 Inv 值是它的各 Inv 值的连接。

这实际上是一个包含节点和边的状态迁移图。每个节点代表状态，边代表状态的迁移。每个 Web 页面对应一个状态，迁移是指 Web 页面从一个状态转换为另一个页面；而状态的迁移总是和事件联系在一起的，或者是从某个条件出发的。例如某个页面上，用户点击了一个“提交”按钮，或者由于超时而发生的页面跳转。

4　实例研究

```
<?xml version="1.0" encoding="utf-8" ?>
<body>
 <RoleRoot xlink:type="simple"
          xlink:href="http://email.123.com/">
    <author>ID="C0" password</author>
    <title>mm's 123 mail</title>
    <role>mm's online email</role>

    <Roles ID="C1" Action="NULl">
      <session-config>
        <session-timeout>900</session-timeout>
        <execute-action>non-action</execute-action>
        <error-message>auto-exite for no action</error-message>
        <error-page>http://email.123.com/#123</error-page>
      </session-config>
    </Roles>

    <Roles ID="C2" Action="write email">
        <auto-save>
          <session-config>
            <session-time>300</session-time>
            <save-mail mail="currentMail">autosave</save-mail>
            <display>autosave to draft box</display>
          <session-config>
        </auto-save>
        <write-action>
          <fromMail>A@163.com</fromMail>
          <ToMail>B@126.com,C@163.com,D@sina.com</ToMail>
          <title>hello</title>
          <content>hello world !</content>
          <attachment>a.rar</attachment>
        </write-action>
    </Roles>

    <Roles ID="C3" Action="send" ToMail1="B@126.com" ToMail2=
          "C@163.com" ToMail3="D@sina.com" fromMail="A@163.com"
          title="hello" content="hello world !">
        <failed-send>
          <send-timeout>
            <session-config>
              <session-timeout>10</session-timeout>
              <display>send failed for timeout</display>
            </session-config>
            <resend-action acion="send"
                mail="currentMail"">send
            </resend-action>
          </send-timeout>
          <delay-reply>
            <session-config>
              <session-timeout>3</session-timeout>
              <display>send successfully</display>
            </session-config>
          </delay-reply>
          <partial-send>
            <display>to B is successful , to C is
                failed, to D is failed
            </display>
          </partial-send>
        </failed-send>
        <success-send>
            <display>send successfully</display>
        </success-send>
    </Roles>

    <Roles ID="C4" Action="non-write action">
      <action>
        ......
      </action>
    </Roles>

 </RoleRoot>
</body>
```

图 4　电子邮件系统的部分 XML 文档

本节以电子邮箱系统为例。当用户登录邮箱后，若长时间无操作，系统会自动退出该用户的登录；当用户写邮件时，为防止网络异常地突然中断，一般系统会在几分钟后将其自动保存到草稿箱；当邮件写好发送时，可能会由于超时而致使邮件不能成功发送，或者返回的是发送成功，但因延迟太长而实际上并没有发送成功。图 4 是自定义的邮箱系统的 Web 页面的一段带有时间约束的 XML 代码。有的页面在某一个状态的时候，可能会有多个时钟计时。

该 XML 文档可分为 4 个部分进行分析和抽取。用户登录后，若长时间(900s 内)没有进行任何操作，系统将自动退出该用户的登录；在 900s 内开始写邮件，为了防止异常导致的内容丢失，每 300s 自动保存一次；发送邮件时可能由于网络等原因导致发送超时、reply 超时而发送失败；在一个操作结束后的 900s 内做一些其他方面的操作。

利用第 2 节给出的提取算法，分析抽取邮件系统的 XML

的信息，可得到图 5 所示对系统模型的 XML 描述。此描述主要由 transition 组成，对于每一个 transition，有源状态、目的状态以及迁移条件，即 source ref、target ref 和 guard，guard 是时间约束和操作的组合。

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE nta (View Source for full doctype...)>
- <nta>
  <declaration>// Place global declarations here. Chan login;
    chan logout; chan send; chan resend; chan write; chan autosave;
    chan reply; chan other;</declaration>
 - <template>
    <name x="5" y="5">Template</name>
    <declaration>// Place local declarations here. Clock x; clock y;
        clock z; clock t;</declaration>
 - <location id="id0" x="-96" y="-40">
    <name x="-160" y="-48">success</name>
    <label kind="invariant" x="-104" y="-32">x<=900</label>
  </location>
 - <location id="id1" x="48" y="-64">
    <label kind="invariant" x="24" y="-56">z<=10&t<=3</label>
  </location>
 - <location id="id2" x="-152" y="-112">
    <label kind="invariant" x="-208" y="-112">x<=900</label>
  </location>
  <location id="id3" x="-16" y="64" />
  <location id="id4" x="200" y="-56" />
  <location id="id5" x="192" y="-184" />
 - <location id="id6" x="48" y="-200">
    <label kind="invariant" x="0" y="-232">x<=900&y<=300</label>
  </location>
 - <location id="id7" x="-88" y="-192">
    <label kind="invariant" x="-104" y="-224">x<=900</label>
  </location>
  <location id="id8" x="-248" y="-192" />
  <init ref="id8" />
 - <transition>
  <source ref="id0" />
   <target ref="id2" />
   <label kind="synchronisation" x="-152" y="-72">other?</label>
   <label kind="assignment" x="-160" y="-88">x:=0</label>
  </transition>
 - <transition>
    <source ref="id6" />
    <target ref="id1" />
    <label kind="synchronisation" x="40" y="-152">send?</label>
    <label kind="assignment" x="48" y="-128">z:=0&t:=0</label>
  </transition>
  ......
</nta>
</xml>
```

图 5　邮件系统的 TA 的 XML 描述片段

从 XML 文档抽取的信息以栈的形式存储。为了形式验证，需将信息退出栈来构造模型。

图 6 是用户登录邮件系统后长时间没有任何操作的自动机模型。当用户使用用户名和密码登录后，系统时钟值 x 置为 0，然后开始计时。在 900s 之内没有操作，那么 Web 页面状态不变。当到 900s($timeout=900$)时，该页面自动退出用户的登录，也即是该页面状态跳转到了登录页面。其中 $Loc=\{0,1\}$，$Loc_0=\{0\}$，$Act=\{login, logout\}$，$C=\{x\}$，$CC(C)=\{x\leqslant 900, x\geqslant 900\}$，$Inv(Loc_0)=\Phi$，$Inv(Loc_1)=\{x\leqslant 900\}$。若状态 0 迁移到状态 1，需满足 Act 是 *login*，并将时钟 x 置为零时发生迁移，即$\langle Loc_0, login, \{x\}, Loc_1\rangle$。状态 1 可保持 900s，位置 Loc_1 满足$\langle Loc_1, v\rangle \xrightarrow{\delta} \langle Loc_1, v+\delta\rangle$，其中 $v(x)+\delta\leqslant 900$。当 $x\geqslant 900$ 时，状态 1 迁移到状态 0，使得$(Loc_1, v) \xrightarrow{\log\ out} (Loc_0, v[\lambda:=0])$迁移上的动作触发。

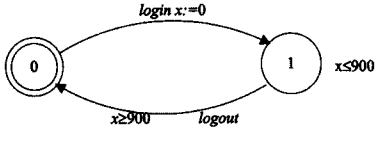


图 6　长时间无操作模型

图 7、图 8 分别所示的是从邮件系统抽取出的写邮件和发送邮件的模型。写邮件时每 300s 将邮件自动保存一次；邮件发送的结果会有 3 种情况。当有时间约束时，进入该状态时时钟首先置为 0，然后开始计时。

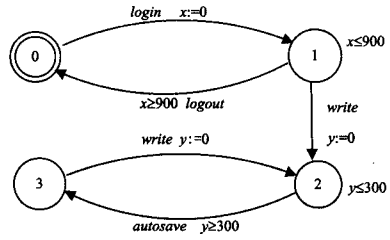


图 7　写邮件模型

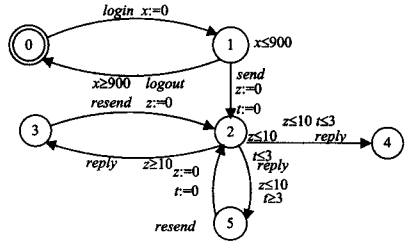


图 8　发送电子邮件模型

图 9 涵盖了除写邮件之外的任何其他操作以及相应状态、时间约束的自动机模型。

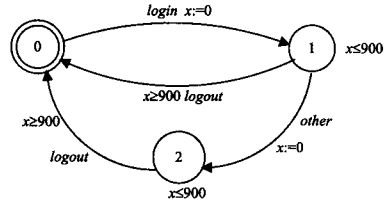


图 9　非写邮件时模型

根据组合时间自动机的定义，将以上 4 个模型进行同步组合，可得到如图 10 所示的邮件系统的时间自动机模型。模型中 $Loc=\{0,1,2,3,4,5,6,7\}$，动作 *login* 是以上 4 个小模型共同的动作。根据组合自动机的定义，当时间约束也被满足时，该状态发生迁移。同理可对其他的状态和动作进行分析组合，最终得到的组合时间自动机模型如图 10 所示。

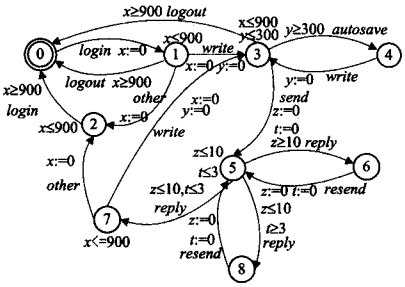
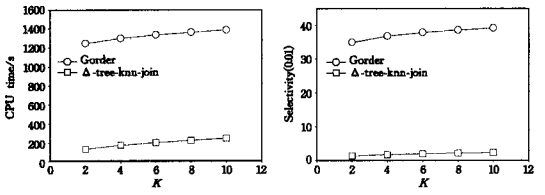


图 10　电子邮件系统组合模型

在该模型中，$Loc=\{0,1,2,3,4,5,6,7,8\}$，$Loc_0=\{0\}$，$Act=\{login, logout, write, other, autosave, send, reply, resend\}$，$C=\{x,y,z,t\}$，$CC(C)=\{x\geqslant 900, x\leqslant 900, y\geqslant 300, y\leqslant 300, z\geqslant 10, z\leqslant 10, t\geqslant 3, t\leqslant 3\}$，$Inv(Loc_0)=Inv(Loc_4)=Inv(Loc_6)=Inv(Loc_8)=\emptyset$，$Inv(Loc_1)=Inv(Loc_2)=Inv(Loc_7)=\{x\leqslant 900\}$，$Inv(Loc_3)=\{x\leqslant 900, y\leqslant 300\}$，$Inv(Loc_5)=\{z\leqslant 10, t\leqslant 3\}$。以状态 5 和状态 6、8 之间的迁移为例，$\langle Loc_5, v\rangle \xrightarrow{\gamma} \langle Loc_5, v+\gamma\rangle$ 中 $v(z)+\gamma\leqslant 10$，$v(t)+\gamma\leqslant 3$。当 $z\geqslant 10$ 时，状态 5 迁移到状态 6(发送失败时的状态)，使得

(下转第 149 页)

的 CPU 性能提高 5.9～9.3 倍，距离计算选择度优化 17.2～26.9 倍。



(a) CPU 时间　　(b) 距离计算选择度

图 2　16 维真实数据集上进行 kNN 连接

Goder 采用无索引块嵌套循环连接技术，为寻找 kNN 需要反复对数据块进行排序和扫描，CPU 消耗相当可观；Δ-tree-KNN-Join 的索引结构 Δ-tree-R 和 Δ-tree-S 的核心算法使用编码和节点聚类中心重合等技术，使得该算法在 kNN 连接过程中能够快速定位位置对应节点，并迅速找到较优的 kNN 候选、快速缩小 kNN 修剪距离，从而提高剪枝能力、大幅减少计算量。Δ-tree-KNN-Join 进行连接时使用较少的维数计算节点之间的距离，所以 CPU 性能较好。

结束语　本文针对 kNN-Join 的特征，给出了构建主存 kNN 连接索引结构的核心算法，实现了编码、定位技术，使得基于该索引结构的连接算法快速定位距离查询点较近的被查询节点，从而提高连接效率。通过实验对比分析了索引 Δ-tree-R 和 Δ-tree-S 的有效性。在后续工作中将继续研究使用较优的降维技术为 kNN 和 RkNN 连接设计索引及查询算法。

参 考 文 献

[1] Böhm C, Krebs F. The k-nearest neighbor join: turbo charging the kdd process [J]. Knowledge and Information Systems (KAIS), 2004, 6(6): 728-749

[2] Xia C, Lu J, Ooi B C, et al. GORDER: An efficient method for KNN join processing[C]// Proceedings of the 30th International Conference on Very Large Data Bases(VLDB'04). San Francisco, CA: Morgan Kaufmann, 2004: 756-767

[3] Cui Y, Cui B, Wang S, et al. Efficient index-based KNN join processing for high-dimensional data[J]. Information and Software Technology, 2007, 49(4): 332-344

[4] 何洪辉，王丽珍，周丽华. pgi-distance：一种高效的并行 KNN-join 处理方法[J]. 计算机研究与发展，2007，44(10)：1774-1781

[5] 梁俊杰，冯玉才. LBD：基于局部位码比较的高维空间 KNN 搜索算法[J]. 计算机科学，2007，6(34)：145-159

[6] Emrich T, Graf F, Kriegel H-P, et al. Optimizing All-Nearest-Neighbor Queries with Trigonometric Pruning [C]// Proceedings of the 22nd International Conference on Scientific and Statistical Database Management(SSDBM). Heidelberg, Germany: Springer, 2010: 501-518

[7] Wang J J, Lin L, Huang T, et al. Effcient K-Nearest Neighbor Join Algorithms for High Dimensional Sparse Data. Computing Research Repository(CoRR)[R]. cs. DB/1011. 2807. 2010-11

[8] Yu C, Zhang R, Huang Y, et al. High-dimensional k NN Joins with Incremental Updates[J]. GeoInformatica, 2010, 14(1): 55-82

[9] 刘艳，郝忠孝. 一种基于主存 Δ-tree 的高维数据 KNN 连接算法[J]. 计算机研究与发展，2010，47(7)：1234-1243

[10] Cui B, Ooi B C, Su J W, et al. Indexing high-dimensional data for efficient in-memory similarity search[J]. IEEE Trans on Knowledge and Data Engineering, 2005, 17(3): 339-353

(上接第 134 页)

$(Loc_5, v) \xrightarrow{reply} (Loc_6, v[\lambda:=0])$迁移上的动作 *reply* 触发；当 $z \leqslant 10$ 且 $t \geqslant 3$ 时，状态 5 迁移到状态 8(*reply* 延迟状态)，使得$(Loc_5, (v_z, v_t)) \xrightarrow{reply} (Loc_8, (v_z[\lambda:=0], v_t[\lambda:=0]))$迁移上的 *reply* 动作触发；另外状态 7 表示发送成功。

结束语　Web 应用程序抽取模型可用于形式化验证。如何抽取基于时间约束的安全模型，也是一个挑战性研究课题。本文应用逆向工程的思想，通过解析带有时间约束的 Web 应用的 XML 源代码，提取和生成时间自动机模型，并且时间自动机模型是可并行组合的。最后通过一个实例阐述了时间约束的抽取过程，而抽取出的模型方便于后续的模型检验以及性质抽取。

本方法可以有效提取 Web 应用的 XML 文档中的时间约束等信息，而且可以通过形式化验证工具 UPPAAL 验证生成的时间自动机模型的正确性。

参 考 文 献

[1] For technical reports, work drafts, recommendations, and specifications of XML and related technologies [EB/OL]. http://www.w3.org

[2] 邓小鹏，邢春晓，蔡莲红. Web 应用测试技术进展[J]. 计算机研究与发展，2007，44(8)：1273-1283

[3] Ricca F, Tonella P. Web site analysis: structure and evolution [C]// International Conference on Software Maintenance. California, 2000: 76-86

[4] Tramontant P. Reverse engineering Web applications[C]// Proceedings of the 21st IEEE International Conference on Software Maintenance. Budapest, 2005: 705-708

[5] Di Lucca G A, Fasolino A R, Pace F, et al. Ware: a tool for the reverse engineering of Web applications [C]// Proceedings of the Sixth European Conference on Software Maintenance and Reengineering. Budapest, 2002: 241-250

[6] 苏杭，严建援. 一种新的 Web 链接提取模型[J]. 清华大学学报，2006，46(1)：975-982

[7] 胡蓉，缪淮扣，刘焕洲. 一种基于 Web 软件集成测试的建模方法[J]. 计算机科学，2007，34(6)：253-257

[8] Tsang T. Using XML-based Real-time Model for Distributed Real-time Multimedia Systems[C]// the 9th IEEE International Conference on Networks (IEEE-ICON2001). Bangkok, Thailand, 10-12October 2001

[9] 方明科，缪淮扣. 一种用于模型验证的 Web 应用模型抽取方法[J]. 应用科学学报，27(1)：90-96

[10] Aitken P G. XML—The Microsoft Way [M]. 谢君英，译. 北京：中国电力出版社，2003. 146-155

[11] Diaz G. Automatic Translation of WS-CDL Choreographies to Timed Automata[M]. Springer Berlin/Heidelberg, 2005: 230-242

[12] Kung D, Liu CH, Hsia P. An object-oriented Web test model for testing Web applications[C]// Proc. of IEEE 24th Annual International Computer Software and Applications Conference (COMPSAC2000). Taipei, 2000: 537-542

[13] Alur R, Dill D L. A theory of timed automata[J]. Theor. Comput. Sci., 1994, 126: 183-235

[14] Alur A R, Dill D. Automata for modeling real-time systems[C]// Proceedings of the 17th International Colloquium on Automata, Languages and Programming. Warwick University, England, 1990: 322-335