第48卷第6期
2008年11月

大 连 理 工 大 学 学 报
Journal of Dalian University of Technology

Vol. 48, No. 6
Nov. 2008

电子与信息工程

文章编号：1000-8608(2008)06-0912-07

一类高级 Petri 网:XML 代数网

唐 达*[1]， 李 晔[1,2]， 王秀坤[1]

（1.大连理工大学 电子与信息工程学院，辽宁 大连 116024；
2.大连海事大学 交通运输管理学院，辽宁 大连 116026）

摘要： 综合运用 Petri 网和 XML 代数的理论和方法，提出 XML 代数网.通过对代数高级网在 XML 代数下的解释和赋值，给出了 XML 代数网的形式化定义，从而建立了 XML 代数网的规范化描述.哲学家问题的实例研究展示了 XML 代数网在动态系统建模和仿真中的应用.研究的结果表明 XML 代数网作为一种工具对 XML 应用领域的建模和分析具有实际意义.

关键词： Petri 网；代数高级网；XML 代数；XML 代数网；哲学家就餐
中图分类号： TP311 **文献标志码：** A

0 引 言

在高级 Petri 网中，令牌不再是无区别的个体，而是作为具有类型的数据对象，反映了实际系统的一种资源，网中的变迁则实现了数据对象之间的变换.在 XML 应用领域，基于模式的编程(schema based programming，SBP)不同于传统的程序设计，是一种新兴的编程方法实践.一个 SBP 应用程序是通过一系列 XSLT 在 XML 模型上的转换来实现的.因此，将具有 XML 类型的静态对象，结合 Petri 网的动态建模特性，构造具有 XML 对象作为令牌的 Petri 网，对 Petri 网在 XML 领域中的应用具有实际意义.

高级网系统是由库所/变迁系统(P/T_系统)折叠而来，较为成熟的高级网系统有谓词/变迁系统(Pr/T_系统)和更一般的有色网系统(colored nets system)[1].代数高级网的出现[2]，丰富和扩展了高级 Petri 网的代数规范化理论，在这一理论基础上，有色网系统是它的一种可实现的形式，它是代数高级网通过在基本种类(basic sorts)和操作下解释得出的语义模型[3]，是最常用的一种建模方法和工具，因此，对于具有基本种类的应用，采用有色网建模是足够的.但是，对于有些应用，要求使用高阶的对象(比如图[4])作为令牌，甚至要考虑动态对象(比如 Petri 网自身[5])作为令牌，这就需要在代数高级网下，对复杂种类及其操作进行解释，以规定其语义.不同的解释，可以得出不同的高级网(精确地应称为高阶网，higher order nets).代数高级网通常提供二层建模技术[4,6]，系统层和对象层，对象层描述作为令牌的高阶对象的建模，而系统层则描述结构的组织和对象的处理.系统的结构是不变的，而处理的对象是可以动态改变的.因此，复杂种类的高级网系统适合于对动态、分布的系统建模.

近年来，将 Petri 网和 XML 相结合的研究大多集中在 Petri 网的 XML 表示和处理方面[7]，而将 XML 对象作为令牌的高级 Petri 网的研究很少.已有研究[8,9]尝试结合使用 XML 和 Petri 网(称之为 XML 网)对应用系统建模.该方法采用了图形化 XML 模式定义语言(GXSL)，使用图形建立 XML 数据模型和模型上的操作.但是，该方法没有在理论上完成对 XML 网的形式化描述，图形限制了 XML 的描述能力，因此在系统的仿真和实现上存在问题.

XML 代数[10,11]提供了对 XML 数据模型的形式化说明和对 XML 数据操作的规范.本文结合 Petri 网和 XML 代数的理论和方法，提出 XML 代数网.它将 XML 对象作为令牌，将 XML

收稿日期： 2006-12-09； **修回日期：** 2008-10-08.
作者简介： 唐 达*(1961-)，男，副教授，E-mail：tangda@dlut.edu.cn；王秀坤(1945-)，女，教授，博士生导师.

对象模型上的操作定义为运算，在此基础上定义变迁转换的条件和结果. 在理论上，通过对代数高级网在 XML 代数下的解释和赋值，给出了 XML 代数网的形式化定义，从而建立了 XML 代数网的描述规范. 匆忙的哲学家(hurried philosophers)问题是哲学家就餐(dining philosophers)问题的扩充，也是研究动态系统建模的典型实例，本文基于 XML 代数网对该实例的研究显示 XML 代数网在动态系统建模和仿真中的应用.

1　XML 查询代数

XML 查询代数定义了 XML 数据查询语言的核心操作符和语义. 它建立了一个完整的类型系统用于描述 XML 数据模型，并确保了在 XML 数据模型上查询操作的封闭性. 本文采用文献[10]中提出的 XML 查询代数作为 XML 代数网中令牌对象的运算基础.

一个关于图书信息的 XML 应用的数据和相应的模式(Schema)如图 1、2 所示，用以对 XML 查询代数作简单介绍. 使用 XML 查询代数(以下称 XML 代数)，可以将图 1、2 中的 XML 模式和数据表示成 XML 代数的类型和变量的形式，如图 3 所示.

在 XML 代数数据模型下的运算和函数详细描述在文献[10]中，表 1 仅给出了在图 3 所示图书信息的 XML 数据模型下的几个简单实例，用来说明 XML 代数的运算和函数.

```
〈bib〉
  〈book〉
    〈title〉Data on the Web〈/title〉
    〈year〉1999〈/year〉
    〈author〉Abiteboul〈/author〉
    〈author〉Buneman〈/author〉
    〈author〉Suciu〈/author〉
  〈/book〉
  〈book〉
    〈title〉XML Query〈/title〉
    〈year〉2001〈/year〉
    〈author〉Fernandez〈/author〉
    〈author〉Suciu〈/author〉
  〈/book〉
〈/bib〉
```

图 1　图书信息 XML 数据实例

Fig. 1　An XML data example of the books

```
〈xsd:group name="Bib"〉
  〈xsd:element name="bib"〉
    〈xsd:complexType〉
      〈xsd:group ref="Book"
        minOccurs="0" maxOccurs="unbounded"/〉
    〈/xsd:complexType〉
  〈/xsd:element〉
〈/xsd:group〉
〈xsd:group name="Book"〉
  〈xsd:element name="book"〉
    〈xsd:complexType〉
      〈xsd:element name="title" type="xsd:string"/〉
      〈xsd:element name="year" type="xsd:integer"/〉
      〈xsd:element name="author" type="xsd:integer"
        minOccurs="1" maxOccurs="unbounded"/〉
    〈/xsd:complexType〉
  〈/xsd:element〉
〈/xsd:group〉
```

图 2　图书信息 XML 数据模式

Fig. 2　An XML data model of the books

```
type Bib = bib [ Book* ]
type Book = book [
      title [ String ],
      year [ Integer ],
      author [ String ]+
    ]
```

(a) 图书信息 XML 数据类型

```
let bib0 : Bib = bib [
    book [
      title [ "Data on the Web" ],
      year [ 1999 ],
      author [ "Abiteboul" ],
      author [ "Buneman" ],
      author [ "Suciu" ]
    ],
    book [
      title [ "XML Query" ],
      year [ 2001 ],
      author [ "Fernandez" ],
      author [ "Suciu" ]
    ]
  ]
```

(b) 图书信息 XML 数据变量和赋值

图 3　图书信息 XML 数据模型的代数表示

Fig. 3　The algebraic representation of XML data model

表 1 XML 代数的投影、选择和函数的实例

Tab. 1 The samples of XML algebra: projection, selection and function

操作	符号	实例	结果
Projection	e/a	*bib0/book/author*	*author* ["*Abiteboul*"], *author* ["*Buneman*"], *author* ["*Suciu*"], *author* ["*Fernandez*"], *author* ["*Suciu*"]
Selection	Where *e* then *e*	*for b in bib0/book do* *where value(b/year)* ⟨= 2000 *do b*	*book* [*title* ["*Data on the Web*"], *year* [1999], *author* ["*Abiteboul*"], *author* ["*Buneman*"], *author* ["*Suciu*"] *author* ["*Suciu*"]]
Function	Fun $f(v_1 t_1; \cdots; v_n t_n) : t = e$	*fun notauthor* (*s* : *String*; *b* : *Book*) : *Boolean* = *empty*(*for a in b/author do* *where value*(*a*) = *s do a*) *for b in bib0/book do* *where notauthor*("*Buneman*"; *b*) *do b*	*book* [*title* ["*XML Query*"], *year* [2001], *author* ["*Fernandez*"], *author* ["*Suciu*"]]

2 代数高级网

代数高级网包括两部分描述：一是在系统层，说明 Petri 网的位置和变迁节点以及它们之间结构上的关系，令牌被视为抽象对象. 该层通常被形式化为自由交换的独异点[12,13](free commutative monoid)，也有称其为多重集并规定其上的运算；二是在对象层，说明高级网中令牌对象的属性和行为，在代数高级网中，通常也用一种代数系统来说明. 对于“Petri 网作为令牌”的高级网系统，有使用与系统层相同[6,14]和与系统层不同[4,5]的两种形式化方法来说明令牌对象. 而对于其他种类的令牌对象，则使用与系统层不同的方法来定义. 一个代数高级网表示如下：

$$AN = (SPEC, P, T, pre, post, cond, type, A)$$

其中 $SPEC = (\Sigma, E; X)$ 为代数说明规范，$\Sigma = (S, OP)$ 为基调(signature)，E 为等式集合，X 为变量集合；P 为位置节点集合；T 为变迁节点集合；变迁节点的前驱和后继 $pre, post: T \to (T_\Sigma(X) \otimes P)^{\oplus}$；变迁条件 $cond: T \to P(Eqns(\Sigma; X))$；位置节点的类型 $type: P \to S$；A 为 (Σ, E) 上的代数.

这里，基调 $\Sigma = (S, OP)$ 定义了代数的种类 S 和操作符 OP；$T_\Sigma(X)$ 是基调 Σ 上含有变量 X 的项的集合；$T_\Sigma(X) \otimes P$ 被定义为

$$(T_\Sigma(X) \otimes P) = \{(term, p) \mid term \in T_\Sigma(X)_{type(p)}, p \in P\}$$

而 $(T_\Sigma(X) \otimes P)^{\oplus}$ 表示集合 $T_\Sigma(X) \otimes P$ 上的自由交换独异点. 因此，$pre(t)$ 具有形式

$$pre(t) = \sum_{i=1}^{n} (term_i, p_i)$$

这里 $n \geqslant 0$，$p_i \in P$，$term_i \in T_\Sigma(X)_{type(p_i)}$. 即变迁 t 的前驱位置节点 p_i 到 t 的弧有权 $term_i$. 对于 $post(t)$ 也类似具有形式；$Eqns(\Sigma; X)$ 是 Σ 上包含变量 X 的等式的有限集合，因此，$cond(t)$ 是等式集合 $Eqns(\Sigma; X)$ 的子集.

代数高级网的变迁规则描述如下：设代数高级网的标记为 $m \in (A \otimes P)^{\oplus}$，其中

$$A \otimes P = \{(a, p) \mid a \in A_{type(p)}, p \in P\}$$

对于变迁 $t \in T$，$Var(t)$ 表示出现在 $pre(t)$、$post(t)$ 和 $cond(t)$ 的变量集合. 设该集合在 A 上的一个赋值为 $asg_A: Var(t) \to A$，如果在 A 中的赋值使得变迁条件 $cond(t)$ 成立，则称赋值 asg_A 在 A 中满足. 对应于 $pre(t) = \sum_{i=1}^{n} (term_i, p_i)$，标记 $pre(t, asg_A)$ 被定义为

$$pre_A(t, asg_A) = \sum_{i=1}^{n} (\overline{asg}_A(term_i), p_i)$$

其中 $\overline{asg_A}: T_\Sigma(Var(t)) \to A$ 是赋值 asg_A 对项的一个扩展；类似地，对于标记 $post_A(t, asg_A)$ 也有同样的定义.

对于变迁 $t \in T$，如果一个赋值 $asg_A: Var(t) \to A$ 是可满足的，且对于标记 $m \in (A \otimes P)^{\oplus}$，有 $pre_A(t, asg_A) \leqslant m$，则变迁 t 被使能，在这种情况下，变迁可以发生，其后继标识 m' 被定义为

$$m' = m \ominus pre_A(t, asg_A) \oplus post_A(t, asg_A)$$

3　XML 代数网

在 XML 应用领域，模式是用来说明 XML 数据模型的一种形式，而由第 2 章可知，XML 代数可以将这种形式转换为代数的类型. 本文采用具有代数形式的 XML 数据模式来定义 XML 代数的种类.

设 $SCHEMAS$ 是在某一应用中具有代数形式的 XML 模式的集合，即 XML 代数类型的集合，$OPNS$ 是该应用中 XML 代数操作符号的集合. 一个 XML 代数网系统定义为

$$XAN = (AN, INIT)$$

其中代数高级网 AN(见第 2 章) 具有代数规范 $SPEC = (XAN\_System\_Signature; X)$ 和在基调 $XAN\_System\_Signature = (XML\_S, XML\_OP)$ 上的 XML 代数 A. 这里，X 是这基调上的变量.

种类 XML_S 在 XML 代数 A 下被解释为具有模式 $t \in SCHEMAS$ 的 XML 对象，即

$$A_{XML\_S} = \{XS \mid (\exists t)(t \in SCHEMAS \wedge XS: t)\}$$

这里，$XS: t$ 表示 XS 具有类型 t，可见，具有相同代数类型的 XML 对象属于相同的种类.

操作 XML_OP 在 XML 代数 A 下被解释为 $OPNS$，即 $A_{XML\_OP} = OPNS$.

对 AN 中所有的 $p \in P$ 都是 XML 类型的位置节点，即

$$p \in P_{XML} = \{p \mid type(p) \in A_{XML\_S}\}$$

$INIT$ 是 XML 代数网系统的初始标识.

XML 代数网是代数高级网的一种可实现的应用形式，它是将代数高级网中的抽象代数系统解释为 XML 代数. XML 代数使用的类型系统具有比 XML 模式更强的表达能力[10]，因此，任何 XML 模式都可以表示为 XML 代数的类型. 另外，XML 代数中的类型之间可以具有子类型关系，以及类型之间的等价表示，也都直接使用在 XML 代数网的规范化描述中.

4　实例研究

匆忙的哲学家问题是从哲学家就餐问题扩展而来，是研究动态系统建模的典型实例. 哲学家就餐问题描述了 n 个哲学家和 n 只筷子，哲学家围坐在一张桌子周围，筷子分别放在每一位哲学家的左右两边，即每一只筷子被二位哲学家共享. 哲学家就餐问题是资源共享问题，它已成为使用 Petri 网分析的经典案例[1,15]. 哲学家就餐问题中的哲学家和筷子的数量是固定的，当数量发生变化时，即有某一哲学家带着一只筷子离开餐厅，或者有一个哲学家带着一只筷子加入进来，仍然要保持桌子周围的哲学家的就餐机制，这就是匆忙的哲学家问题.

根据上一章给出的 XML 代数网的形式化定义，建立基于 XML 代数网的匆忙哲学家问题的模型及其描述. 为说明问题简单，在此只考虑哲学家编号和筷子编号等必要信息. 设哲学家的 XML 代数类型：

$$\begin{aligned} type\ PHILO = philo[& \\ & id[String], \\ & leftchops[String], \\ & rightchops[String]\{0,1\} \\ ] & \end{aligned}$$

其中 id 是哲学家的编号；$leftchops$ 是该哲学家左边的筷子，这里规定这只筷子由哲学家自带，即在加入和离开餐厅时带着它；$rightchops$ 是哲学家右边的筷子，只有哲学家加入到餐厅时才表示出来. 桌子周围哲学家旁边的筷子，不论左右，都与邻接的哲学家共享. 筷子的 XML 代数类型则有简单的表示：

$$type\ CHOPS = chops[String]$$

其中 $chops$ 表示筷子的编号.

在 XML 代数网中，设

$$\begin{aligned} & XAN\_System\_Signature = (\\ & \quad \cdot\ XML\_S: Philosopher, Chopstick \\ & \quad \cdot\ XML\_OP: \\ & \qquad invitee, inviter, neighbor: \\ & \qquad\quad Philosopher \times Philosopher \to Philosopher \\ & \qquad leaver: Philosopher \to Philosopher \\ & \qquad getleftchops, getrightchops: Philosopher \to Chopstick \\ &) \end{aligned}$$

$$X = \{p, p_1: Philosopher;\ c, c_1: Chopstick\}$$

XML 代数 A 下种类 XML_S、操作符 XML_OP 的解释如下：

哲学家 $A_{philosopher}=\{ph_i \mid ph_i:PHILO \wedge i=1,\cdots,n\}$

筷子 $A_{chopstick}=\{ch_i \mid ch_i:CHOPS \wedge i=1,\cdots,n\}$

函数 *invitee* 解释为被邀请操作，表示被邀请哲学家进入餐厅的变换，带有参数被邀请人（自己）和邀请人作为参数，定义为

$invitee_A:A_{philosopher}\times A_{philosopher}\rightarrow A_{philosopher}$

$fun\quad invitee_A(iee:A_{philosopher},ier:A_{philosopher}):A_{philosopher}=$

$\quad if(getrightchops_A(ier)=())\quad then$

$\quad\quad philo[iee/id,iee/leftchops,ier/leftchops]$

$\quad else\quad philo[iee/id,iee/leftchops,$

$\quad\quad ier/rightchops]$

函数 *inviter* 解释为邀请操作，表示邀请后哲学家回到餐桌的变换，带有参数被邀请人和邀请人（自己）作为参数，定义为

$inviter_A:A_{philosopher}\times A_{philosopher}\rightarrow A_{philosopher}$

$fun\quad inviter_A(iee:A_{philosopher},ier:A_{philosopher}):A_{philosopher}=$

$\quad philo[ier/id,ier/leftchops,iee/leftchops]$

函数 *leaver* 解释为哲学家离开餐厅，表示哲学家带着（左边的）一只筷子离开餐厅的变换，定义为

$leaver_A:A_{philosopher}\rightarrow A_{philosopher}$

$fun\quad leaver_A(ler:A_{philosopher}):$

$\quad A_{philosopher}=philo[ler/id,ler/leftchops]$

函数 *neighbor* 解释为离开餐厅哲学家左边的邻居的变换操作，操作完成对这位哲学家右边筷子的设置，定义为

$neighbor_A:A_{philosopher}\times A_{philosopher}\rightarrow A_{philosopher}$

$fun\quad neighbor_A(ler:A_{philosopher},nor:A_{philosopher}):A_{philosopher}=$

$\quad if(getrightchops_A(ler)=getleftchops(nor))$

$\quad then\quad philo[nor/id,nor/leftchops]\quad else$

$\quad philo[nor/id,nor/leftchops,ler/rightchops]$

函数 *getleftchops* 和 *getrightchops* 解释为获得哲学家左边和右边筷子标识的操作，分别定义为

$getleftchops_A:A_{philosopher}\rightarrow A_{chopstick}$

$fun\quad getleftchops_A(ph:A_{philosopher}):A_{chopstick}=$

$\quad chops[value(ph/leftchops)]$

$getrightchops_A:A_{philosopher}\rightarrow A_{chopstick}$

$fun\quad getrightchops_A(ph:A_{philosopher}):A_{chopstick}=$

$\quad chops[value(ph/rightchops)]$

另外，设

$type(Think)=type(Eat)=type(Hall)=Philosopher,$

$type(Chopsticks)=Chopstick$

哲学家 $ph_1,\cdots,ph_5$ 定义 XML 代数形式为

$let\ ph_1:PHILO=philo[id["ph_1"],leftchops["ch_1"],rightchops["ch_2"]]$

$let\ ph_2:PHILO=philo[id["ph_2"],leftchops["ch_2"],rightchops["ch_1"]]$

$let\ ph_3:PHILO=philo[id["ph_3"],leftchops["ch_3"]],$

$let\ ph_4:PHILO=philo[id["ph_4"],leftchops["ch_4"]]$

$let\ ph_5:PHILO=philo[id["ph_5"],leftchops["ch_5"]],$

筷子 $ch_1,\cdots,ch_5$ 定义 XML 代数形式为

$let\ ch_1:CHOPS=chops["ch_1"],$

$let\ ch_2:CHOPS=chops["ch_2"],$

$let\ ch_3:CHOPS=chops["ch_3"],$

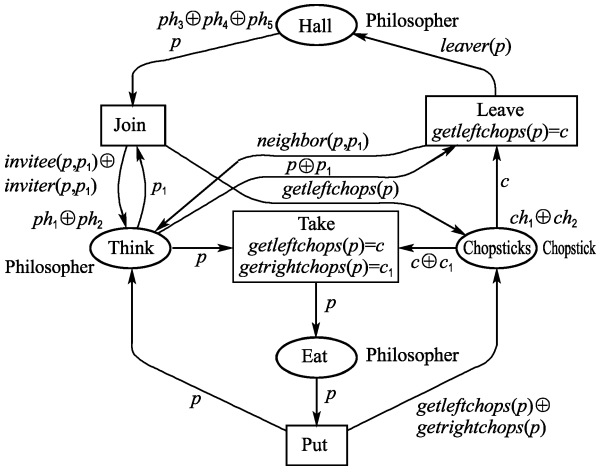
$let\ ch_4:CHOPS=chops["ch_4"],$

$let\ ch_5:CHOPS=chops["ch_5"]$

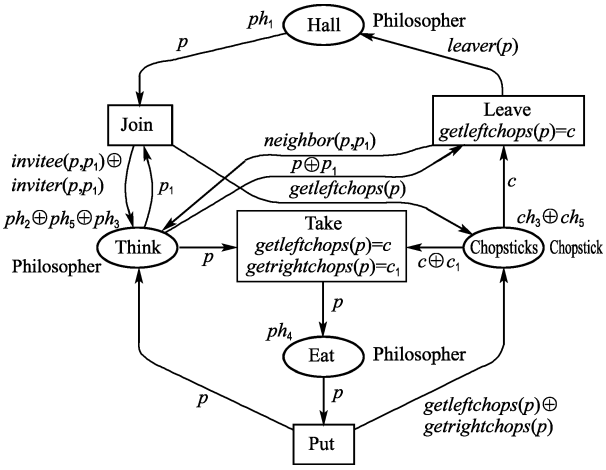
匆忙的哲学家问题的 XML 代数网初始标记设置为

$$INIT=\sum_{i=1}^{2}(ph_i,Think)\oplus\sum_{i=3}^{5}(ph_i,Hall)\oplus\sum_{i=1}^{2}(ch_i,Chopsticks)$$

可见，系统在初始状态有哲学家 ph_1 和 ph_2 围在餐桌准备就餐，他们共享 ch_1 和 ch_2，其他的哲学家都在大厅，如图 4(a) 所示.



(a) 系统初始时 ph_1 和 ph_2 共享筷子 ch_1 和 ch_2 处于准备就餐



(b) 系统某一时刻 ph_2、ph_3、ph_4 和 ph_5 围坐在餐桌，ph_4 使用筷子 ch_4 和 ch_2 正在就餐

图 4 匆忙哲学家问题的 XML 代数网模型与仿真

Fig. 4 The modeling and simulation of hurried philosophers based on XML algebraic nets

当系统经过多次交互，既有哲学家进入餐厅，也有哲学家离开餐厅，在某一时刻，有哲学家 ph_2、ph_3、ph_4 和 ph_5 围坐在餐桌，ph_4 使用筷子 ch_4 和 ch_2 正在就餐，如图 4(b) 所示. 此时，ph_1、ph_2、ph_3、ph_4 和 ph_5 表示的 XML 数据如下：

$let\ ph_1:PHILO = philo[id["ph_1"],leftchops["ch_1"]]$

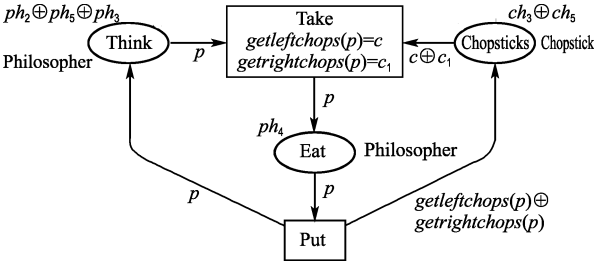
$let\ ph_2:PHILO = philo[id["ph_2"],leftchops["ch_2"],rightchops["ch_5"]]$

$let\ ph_3:PHILO = philo[id["ph_3"],leftchops["ch_3"],rightchops["ch_4"]]$

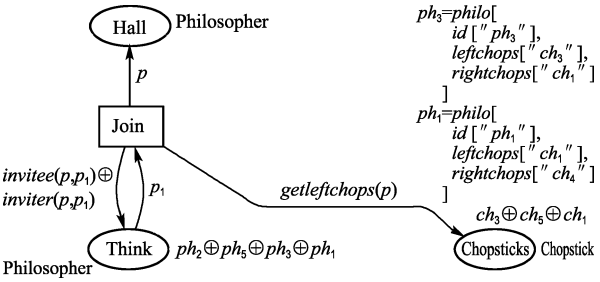
$let\ ph_4:PHILO = philo[id["ph_4"],leftchops["ch_4"],rightchops["ch_2"]]$

$let\ ph_5:PHILO = philo[id["ph_5"],leftchops["ch_5"],rightchops["ch_3"]]$

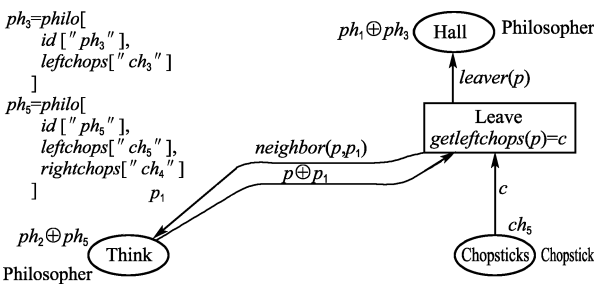
餐厅的就餐机制，是根据餐桌上的哲学家数量及其邻接关系确定的，图 4(b)的一部分表示这一就餐过程，如图 5(a)所示. 如果系统在图 4(b)所示的状态下，哲学家 ph_3 邀请 ph_1 进入餐厅就餐，即有变量 $p = ph_1$ 且 $p_1 = ph_3$，变迁的结果如图 5(b)所示. 如果系统在图 4(b)所示的状态下，哲学家要 ph_3 离开餐厅，则有 $p = ph_3$ 且 $p_1 = ph_5$，变迁的结果如图 5(c)所示. 哲学家进入和离开餐厅动态地改变了餐厅中的哲学家数量及其邻接关系，从而完成了哲学家就餐问题的动态模拟.



(a) 哲学家就餐问题模型



(b) 哲学家进入餐厅模型



(c) 哲学家离开餐厅模型

图 5　匆忙哲学家问题分解说明

Fig. 5　The disassembled illustration of the hurried philosophers

5　结　语

本文提出的 XML 代数网，是代数高级网的一种可执行的系统模型，它给出了“XML 对象作为令牌”的高级 Petri 网的形式化语义，建立了一种将 XML 数据模型与 Petri 网相结合的规范化方法.

本文的基于 XML 代数网的实例是通过匆忙哲学家问题实现了哲学家就餐问题的动态仿真. 虽然使用的数据模型简单，但它却是 XML 对象，可以容纳更多的信息(比如哲学家姓名等). XML 代数具有很强的处理能力，它既可以简化复杂问题的建模和仿真，也可以对所建模型作更为详细的描述. 比如公文流转中的流转单可以处理为基于 XML 代数网中的 XML 令牌对象. 可见，基于 XML 代数网的应用会更具体和贴近实际.

参考文献：

[1] 袁崇义. Petri 网原理与应用 [M]. 北京：电子工业出版社，2005

[2] EHRIG H，HOFFMANA K，PADBERG J，*et al*. High-level net processes [M] // **Formal and Natural Computing，Lecture Notes in Computer Science (2300)**. New York：Springer-Verlag，2002：191-219

[3] GIRAULT C，VALK R. **Petri Nets for System Engineering：a Guide to Modeling，Verification，and Applications** [M]. New York：Springer-Verlag，2001

[4] HOFFMANN K，MOSSAKOWSKI T. Algebraic higher-order nets：graphs and Petri nets as tokens [C] // **The 16th International Workshop on Recent Trends in Algebraic Development Techniques**. Frauenchiemsee：Springer，2002：253-267

[5] HOFFMANN K，EHRIG H，MOSSAKOWSKI T. High-level nets with nets and rules as tokens [C] // **The 26th International Conference on Applications and Theory of Petri Nets**. Miami：Springer，2005：268-288

[6] VALK R. Concurrency in communicating object Petri

nets [M] // **Concurrent Object-Oriented Programming and Petri Nets: Advances in Petri Nets**. New York: Springer-Verlag, 2001:164-195

[7] BILLINGTON J, CHRISTENSEN S, VAN HEE K, *et al*. The Petri net markup language:concepts, technology, and tools [C] // **The 24th International Conference on Applications and Theory of Petri Nets**. Eindhoven:Springer, 2003:1023-1024

[8] LENZ K, OBERWEIS A. Inter-organizational business process management with XML nets [M] // **Petri Net Technology for Communication-Based Systems, Lecture Notes in Computer Science(2472)**. Berlin / Heidelberg:Springer-Verlag, 2003:243-263

[9] VON MEVIUS M, PIBERNIK R. Process management in supply chains — a new Petri-net based approach [C] // **The 37th Annual Hawaii International Conference on System Sciences**. USA: IEEE Computer Society, 2004:71-80

[10] FERNANDEZ M, SIMEON J, WADLER P. An algebra for XML query [C] // **The 20th Conference on Foundations of Software Technology and Theoretical Computer Science**. New Delhi: Springer-Verlag, 2000:11-45

[11] JAGADISH H, LAKSHMANAN L, SRIVASTAVA D, *et al*. TAX:a tree algebra for XML [M] // **Database Programming Languages, Lecture Notes in Computer Science (2397)**. Berlin / Heidelberg:Springer-Verlag, 2002:149-164

[12] BADOUEL E, CHENOU J, GUILLOU G. Petri algebras [C] // **The 32nd International Colloquium on Automata, Languages and Programming**. Lisbon: Springer, 2005:742-754

[13] MESEGUER J, MONTANARI U. Petri nets are monoids:a new algebraic foundation for net theory [C] // **The 3rd Annual Symposium on Logic in Computer Science**. USA:IEEE Xplore, 1988: 155-164

[14] VALK R. Petri nets as token objects:an introduction to elementary object nets [M] // **Application and Theory of Petri Nets 1998, Lecture Notes in Computer Science (1420)**. Berlin / Heidelberg:Springer-Verlag, 1998:1-24

[15] CHENG A, MORTENSEN K H. Model checking coloured Petri Nets exploiting strongly connected components [C] // **International Workshop on Discrete Event Systems**. Edinburg: [s n], 1996: 169-177

A class of high-level Petri Nets: XML algebraic nets

TANG Da[*1], LI Ye[1,2], WANG Xiu-kun[1]

(1. School of Electronic and Information Engineering, Dalian University of Technology, Dalian 116024, China;
2. Transportation Management College, Dalian Maritime University, Dalian 116026, China)

Abstract: XML algebraic nets are proposed by means of theories and methods combining Petri Nets with XML algebra. A formal definition of XML algebraic nets is given by an interpretation and an assignment of algebraic high-level nets under XML algebra. Consequently, the specifications of XML algebraic nets are formed. A case study of philosopher's problem is investigated to illustrate the applications of dynamic system modeling and simulation based on XML algebraic nets. The research results show that XML algebraic nets, as a modeling and analyzing tool in the field of XML application, are significant in practice.

Key words: Petri Nets; algebraic high-level nets; XML algebra; XML algebraic nets; dining philosopher