软件体系结构的属性图文法描述及其约束验证*

石 兵, 冉 平, 马晓星, 陶先平, 吕 建
(南京大学 计算机软件研究所 计算机软件新技术国家重点实验室, 江苏 南京 210093)

摘 要: 在前人工作的基础上,使用了一种利于约束检查和属性刻画的属性图文法,该方法形式地描述了体系结构及其演化;接着给出了一个算法检查演化动作是否会破坏体系结构约束。在对属性图文法系统 AGG 的图形解析器进行定制和改进的基础上,设计并实现了体系结构自动检查器。该检查器已应用于面向体系结构的服务集成开发平台 Artemis-ARC 系统中。

关键词: 软件体系结构; 动态演化; 属性图文法; 验证
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2007)03-0163-06

Attributed Graph Grammar-based Description and Constraints Verification of Software Architectures

SHI Bing, RAN Ping, MA Xiao-xing, TAO Xian-ping, LV Jian
(State Key Laboratory for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing Jiangsu 210093, China)

Abstract: Based on existing researches on graph grammar-based formalisms of software architecture, firstly an Attributed Graph Grammar-based(AGG) approach was proposed to describe software architectures, which emphasized the description and verification of attribute-related constrains. Secondly an algorithm to verify the simulated evolved architecture against the constraints was discussed. Finally an automatic architecture checker was implemented based on TU Berlin's AGG parser with some enhancing customization and adaptation. The checker has been integrated into the Artemis-ARC system.

Key words: software architecture; dynamic evolution; attributed graph grammar; verification

0 引言

软件体系结构是指该系统的构件组成、构件间的相互关系和连接方式所形成的系统结构配置及动机、模式和约束等[1]。动态软件体系结构则是指系统运行时刻软件体系结构可以演化,进行重配置。动态体系结构研究中,如何保证体系结构的演化不会破坏体系结构的约束是关键技术之一[2]。一般情况下,动态体系结构演化是在系统运行时刻进行的,因此体系结构约束验证必须在线、自动进行。体系结构约束包括属性值限制、体系结构拓扑限制等,它实际上给出了合法体系结构的集合。

体系结构约束验证方法一般包括两个部分,即约束描述和约束检查算法。越来越多的机构和个人在动态体系结构演化及其约束验证方面开展工作并取得良好成果,其中体系结构描述语言 ACME 使用一阶谓词逻辑作为约束描述语言。此外还引入函数表示体系结构的特定特征,在给出约束描述后,在 Armani 原型系统中实现了一个基于深度优先算法的体系结构检查器[4,5]。其他一些动态体系结构描述语言,如 Wright[6,7]、Darwin[8~10]等,均给出了约束描述和约束检查算法。这些方法在完善性、直观性、应用复杂性等方面不同程度地存在不足之处。例如 ACME 限制类型的表示,使得结构类型元素是一个有限集合,从而解决一阶谓词逻辑的不可判定性问题; Wright 中对状态机进行穷尽搜索,因此在时间和空间上存在限制; Darwin 提供的编译器尚不完善,有些违反约束的情况无法检查出来。

Daniel Le Metayer[11]提出使用图文法(Graph Grammar)形式地描述体系结构及其风格。文献[13, 14]提出了使用图文法验证体系结构是否满足约束;这种基于图文法的约束验证方法优点在于直观、易理解,并且图文法还可对面向图系统中的图进行形式描述。但文献[11, 13, 14]的工作还只是理论上的研究,没有真正地实现,另外也没有明确定义体系结构的属性信息该如何描述和验证。本文在文献[11, 13, 14]的基础上,讨论了一种基于属性图文法的体系结构描述及约束验证方法。基于属性图文法的体系结构描述方法不但刻画了包括属性在内的体系结构信息,还有利于使用图形解析技术进行包括属性在内的约束验证。在此理论基础上,利用属性图文法系统 AGG[3]的图形解析器设计和实现了体系结构自动检查器,并将其应用于 Artemis-ARC 系统[1]中。该系统采用面向图的运

收稿日期: 2005-12-30; 修返日期: 2006-02-17 基金项目: 国家"973""863"计划资助项目(2002CB312002, 2005AA113160, 2004AA112090);国家自然科学基金资助项目(60273034, 60233010, 60403014)

作者简介: 石兵(1982-),男,硕士研究生,主要研究方向为 Internet 软件技术、软件 Agent 技术; 冉平(1982-),男,硕士研究生,主要研究方向为 Internet 软件技术、软件 Agent 技术; 马晓星(1975-),男,副教授,主要研究方向为 Internet 软件技术、软件体系结构; 陶先平(1970-),男,教授,主要研究方向为 Internet 软件技术、移动 Agent 技术; 吕建(1960-),男,教授,博导,博士,主要研究方向为软件自动化、并行程序形式化方法、面向对象语言和环境.

行时体系结构对象技术,该技术与属性图文法描述体系结构模型是一致的。体系结构的演化首先将被模拟执行;然后对模拟执行后的体系结构进行约束验证;如约束满足,则演化被真正执行,并能在约束满足方面得到保障。

1 体系结构的属性图文法描述和约束验证

1.1 体系结构的属性图文法描述

本文分别用体系结构实例属性图(AAG)、体系结构风格属性图文法(ASGG)来形式地描述体系结构实例、体系结构风格,用属性图转换描述体系结构演化。它们的形式定义如下。

定义 1 体系结构实例属性图。L_{VT}是终结节点标记集合,L_{ET}是终结边标记集合,基调(Signature) Sig = (S, OP),则 AAG = (V, E, s, t, lv, le, AAG_A, av, ae)是基调为 Sig 的属性图,其中:

(1) V是有限个终结节点的集合;E是有限个终结边的集合。

(2) s: $E\to V$是源节点函数;t: $E\to V$是目标节点函数。

(3) lv: $V\to L_{VT}$是终结节点标记函数;le: $E\to L_{ET}$是终结边标记函数。

(4) AAG_A 是一个 Sig 代数(Sig-Algebra)。

(5) av: $V\to U(AAG_A)$ 和 ae: $E\to U(AAG_A)$ 分别是终结节点和终结边的属性函数。

体系结构实例实际上包含了:①构件(Component)和连接子(Connector)。②构件和连接子组成的体系结构的配置信息。③属性信息。

AAG 形式地描述了上述体系结构实例:

①体系结构实例中出现的构件 com∈V、lv(com)表明了构件类型;连接子 con∈E、le(con)表明了连接子类型。

②配置信息表现为 V、E 上的图,函数 s、t 定义了拓扑结构。

③函数 av: $V\to U(AAG_A)$ 和 ae: $E\to U(AAG_A)$ 赋予体系结构实例属性信息。

与属性图定义不同,AAG 中节点和边必须是终结的。当属性图中含有非终结元素时,它可能是某推导序列中的中间图。中间图定义与附录中属性图定义一致。

定义 2 体系结构风格属性图文法。基调 Sig = (S, OP),符号集合 $V=\{x_1, x_2, \cdots, x_k\}$。其元素 x_i 称为变量,ASGG = (L_V, L_{VT}, L_E, L_{ET}, P, G_0, Γ, avt, aet),其中:

(1) L_V 是节点标记集合;$L_{VT}\subset L_V$ 是终结节点标记集合;L_E 是边标记集合,$L_{ET}\subset L_E$ 是终结边标记集合。

(2) $P=\{p_1, p_2, \cdots\}$ 是带应用条件的产生式有限集合。产生式 $p_i=(L\xrightarrow{q}R, A(p))$,$q$: $L\to R$ 是属性图态射,$A(p)=(AP(p), AN(p), \mathrm{Exp})$ 是该产生式的应用条件。对于子属性图匹配 $L\xrightarrow{m}G$,当 $m\models_P A(p)$ 时 G 才能使用该产生式进行推导。

(3) G_0 是体系结构风格属性图文法的初始属性图。

(4) 类别指派 $\Gamma=\{x_1:s_1, \cdots, x_k:s_k\}$ 是变量与类别二元组的有限集合,op: $s_1\times s_2\cdots\times s_k\to s$ 是运算符号。其中,$op\in OP$,$x_i\in V$, $s_i\in S$ $(1\le i\le k)$, $s\in S$。

(5) avt: $L_{VT}\to\Gamma$ 和 aet: $L_{ET}\to\Gamma$ 是对节点标记和边标记指派属性的函数。

与一般图文法定义不同,ASGG 进行了扩展,除了有初始图和产生式集合外,还有节点标记、边标记集合以及属性指派,从而可以很好地形式描述体系结构风格。在上述 ASGG 定义中,属性由基调 Sig = (S, OP)来描述,还是抽象描述;在属性图中,具体属性信息将由 Sig 代数描述。

从初始图出发,使用满足应用条件的产生式进行推导,可以产生体系结构实例图。$G_0\xrightarrow{p_i, m_i}G_1\xrightarrow{p_j, m_j}\cdots\xrightarrow{p_k, m_k}AAG$。其中,$G_0$ 是初始图;AAG 是体系结构实例图;p_i, p_j, $\cdots$, $p_k\in P$;m_i, m_j, $\cdots$, m_k 是产生式左部 L 与待推导图中的子属性图匹配,此推导序列记为 $G_0\to_p^* AAG$。ASGG 生成的图形语言如下:

L(ASGG) = {[AAG] | AAG 是 L_{VT}和 L_{ET}上的属性图,且有推导序列 $G_0\to_p^* AAG$}

其中[AAG]表示所有与 AAG 属性图同构的属性图集合。如果 AAG∈L(ASGG),则体系结构实例属性图遵循体系结构风格属性图文法。

体系结构风格实际上规定了:

(1) 体系结构实例中出现的构件类型和连接子类型;

(2) 上述类型的构件和连接子构造的体系结构实例所要满足的拓扑结构约束、体系结构实例中构件和连接子要满足的属性约束。

ASGG = (L_V, L_{VT}, L_E, L_{ET}, P, G_0, Γ, avt, aet)形式化地描述了上述概念:

(1) 每一个构件类型定义为一个终结节点标记 $X\in L_{VT}$,每一个连接子类型定义为一个终结边标记 $T\in L_{ET}$。

(2) L_{VT}和 L_{ET}上的体系结构实例属性图所要满足的拓扑结构约束和属性约束定义为 AAG∈L(ASGG)。

为了适应应用需求的变化,体系结构必须可以重配置,有良好的演化能力,对用属性图描述的体系结构,演化就表现为基于规则集的属性图转换。规则集定义为 $R=\{r_1, r_2, \cdots\}$。其中 $r_i=(L\xrightarrow{q}R, A(p))$(它与产生式 $p_i=(L\xrightarrow{q}R, A(p))$的定义是一样的,但是表达不同的含义:$p_i$ 是用于从初始图出发产生满足图文法约束的图;而 r 则是对任意图进行图转换。也就是说,根据规则 r 进行图转换后的图未必属于图文法定义的图形语言)。规则集 R 上的体系结构实例属性图演化表现为 $AAG\xrightarrow{r_i, m_i}G_1\xrightarrow{r_j, m_j}G_2\to\cdots\xrightarrow{r_k, m_k}AAG'$,将它简记为 $AAG\to_R^* AAG'$,r_i, r_j, $\cdots$, r_k 称为一个规则序列,m_i, m_j, $\cdots$, m_k 是规则左部在待转换图中的子属性图匹配。对于刻画了体系结构演化的属性图转换,要求它满足以下性质:对任何由 ASGG 生成的属性图 AAG,在其上施以规则进行图形转换后,产生的属性图 AAG 仍然可以由 ASGG 生成,即

$$(AAG\in L(ASGG))\wedge(AAG\to_R^* AAG')\Rightarrow(AAG'\in L(ASGG))$$

也就是说,演化合法,即要保证演化后体系结构仍然遵循原体系结构风格,在属性图文法中就是要保证 AAG′∈L(ASGG)成立。判断它是否成立,形式语言中称为从属问题(Membership Problem),是一个 NP 完全问题[17]。在一个受限图文法中,这个问题是可以解决的,如基于图形解析(Graph Parsing)技术,判断待检查图是否可达(Reachable)。可达是指在图形转换系统中,待检查图不含非终结元素,从初始图出发使用产生式经一系列推导得到的图与待检查图同构[19](在属性图文

法中是属性图同构)。本文就是基于图形解析技术进行约束验证的,下面给出约束描述和约束验证算法。

1.2　约束验证

定义 3　约束满足。对于体系结构实例属性图,如果 $\exists AAG'$: $AAG' \in L(ASGG) \wedge AAG \cong_{AG} AAG$,则 AAG 对于 ASGG 是约束满足的。AAG 描述的体系结构实例遵循 ASGG 描述的体系结构风格。

定义了约束满足后,给出约束检查算法。约束检查算法基于图形解析技术,判断是否可找到合适的产生式序列,初始图按此产生式序列进行推导保证待检查图可达。为了能让算法正常终止,给出限制条件如下:

$\forall p_i \in P(|L| \leq |R| \wedge \neg(L \cong R))$ ($\neg(L \cong R)$) 指 L 与 R 图拓扑结构不同构)

该条件要求产生式右部元素(节点、边)不比左部少,至少创建或修改其中一个元素。它和算法中的回溯条件一起保证算法终止。在 CHOMSKY 文法体系中将满足此限制条件的文法称为 1 型文法,即上下文相关文法[20]。由于从初始图进行图形推导判断待检查图是否可达,待检查图元素数一般大于初始图或是对初始图元素的修改,在推导过程中不会用到右部元素比左部少或左、右部图同构的产生式,故该限制条件是合理的。约束检查算法如下:

```
算法 1
PROCESS Parse
ASGG, AAG;
BEGIN
  G1 = G0
  ST = Stack;
  if( ¬ isAllTerminateElement( AAG) )
      Terminate Wrongly;
  else
    LOOPBEGIN
    if( isomorphism( G1, AAG) )
        Terminate Correctly;
    else
      if( ElementNum( G1) > ElementNum( AAG) )
        if( ST = Φ)
          Terminate Wrongly;
        else
          ( G1, S) = POP( ST) ;
    else if ( ∃G( ST. contain( G, S) ∧ isomorphism( G1,
  G) ) )
      ( G1, S) = POP( ST) ;
      if( - isInitialized( set S of G1) )
      initialize a set S for G1;
      if( ∃p( p∈P∧ismatch( p, G1) ∧ match 〈m, p, G1〉
  ∉S) )
        addmatch 〈m, p, G1〉into S;
        PUT( G1, S) into ST;
        G1 = apply p to G1;
      else
        if( ST = Φ)
            Terminate Wrongly;
        else
            ( G1, S) = POP( ST) ;
  end
END
```

该算法是深度优先的回溯搜索算法。一般情况下回溯算法的可终止性和正确性是能得到保证的。当它用于语言解析时,要防止由于循环文法(Cyclic Grammar)[21]引起推导无限循环,造成检查无法终止。前述限制条件虽然没有防止循环文法的出现,但是在算法中采取了措施保证跳出循环,从而使算法可终止,正确性也可得到保证。

定理 1　给定满足前述限制条件的体系结构风格属性图文法和体系结构属性图,验证 AAG 对于 ASGG 是否约束满足。算法 1 总可以保证验证过程是可终止的。

证明　用 $|G|$ 表示图中元素数目,分循环文法存在与否两种情况讨论:

(1) 不存在循环文法,即对任意推导序列 $G_0 \xrightarrow{p_r, m_r} G_1 \xrightarrow{p_l, m_l} G_2 \to \cdots \to G_m \to \cdots$,满足 $\forall G_p, G_q\ \neg(G_p \cong_{AG} G_q)\quad 0 \leq p, 0 \leq q, p \neq q)$。因为 $\forall p_i \in P(|L| \leq |R| \wedge \neg(L \cong R))$,故 $|G_0| \leq |G_1| \leq \cdots \leq |G_m| \leq \cdots$。假设上述推导序列是无穷的,由于产生式数目是有限的,可尝试的产生式匹配(match $\langle m, p, G_1\rangle$)也是有限的。当推导序列无穷且不由循环文法产生(由形如 $p = (A \to aA)$ 的产生式产生),则必 $\vee\ G_k.\ |G_k| > |AAG|$。在算法中,此种情况出现时会要求回溯,从而不可能产生无穷序列。由于产生式数目和可尝试的产生式匹配都是有限的,因此算法最终会终止。

(2) 存在循环文法,即存在推导序列 $G_0 \xrightarrow{p_r, m_r} G_1 \xrightarrow{p_l, m_l} G_2 \to \cdots$,其中有 G_p, G_q 满足 $G_p \cong_{AG} G_q (0 \leq p < q)$。在算法中,当 $G_{q-1} \xrightarrow{p_t, m_t} G_q$,由于 $G_p \cong_{AG} G_q$,算法要求回溯到 G_{q-1},选择该产生式其他未使用匹配(当无未使用匹配时,尝试其他产生式的匹配),从而跳出循环,此时与不存在循环文法的情况相同,算法最终会终止。

综上所述,算法可终止性得证。

定理 2　给定满足前述限制条件的体系结构风格属性图文法和体系结构属性图,验证 AAG 对于 ASGG 是否约束满足。算法 1 总可以保证验证结果是正确的。

证明　使用数学归纳法证明当 AAG 对于 ASGG 约束满足时,算法总可以找到有限步推导序列 $G_0 \xrightarrow{p_r, m_r} G_1 \xrightarrow{p_l, m_l} G_2 \to \cdots \to G_m$,并且满足 $G_m \cong_{AG} AAG$。

(1) 对于一步推导序列 $G_0 \xrightarrow{p_r, m_r} G_1$,由算法可知,满足 G_0 的所有产生式匹配总会被尝试到。因此,这个一步推导序列是可找到的。

(2) 假设算法可以找到任意 $k-1$ 步推导序列,下面证明对于 k 步推导序列,算法仍可找到。由于 $k-1$ 步推导序列可找到,因此有 $G_1 \xrightarrow{p_r, m_r} G_2 \xrightarrow{p_l, m_l} G_3 \to \cdots \to G_k$(对于算法来说,初始图是 G_0 还是 G_1 都是一样的)。下面分三种情况讨论:

①如果 $\neg\exists\ G_i.\ G_i \cong_{AG} G_0 (1 \leq i \leq k)$,则根据回溯算法,总可以找到推导步 $G_0 \xrightarrow{p_q, m_q} G_1$,而 $G_1 \xrightarrow{p_r, m_r} G_2 \xrightarrow{p_l, m_l} G_3 \to \cdots \to G_k$ 总是可以被算法找到的(推导尝试过程中可能会推导到 G_0,但根据算法,此时会回溯,不会产生循环)。因此推导序列 $G_0 \xrightarrow{p_q, m_q} G_1 \xrightarrow{p_r, m_r} G_2 \to \cdots \to G_k$ 可被找到,正确性得证。

② $\exists G_i.\ G_i \cong_{AG} G_0\ (1 \leq i \leq k)$ 此时已存在推导序列 $G_0 \xrightarrow{p_t, m_t} G_{i+1} \cdots \to G_k$。这是小于 k 步的推导序列,显然可由算法得到。

综上所述,算法的正确性得证。

算法还可以优化,如子图匹配优化、分层规则、冲突对分析等[22]。该算法不但可用于检查体系结构实例属性图是否遵循体系结构风格属性图文法,还可对某些非基于属性图文法描述的体系结构进行检查。此时首先使用 ASGG 描述体系结构所遵循的体系结构风格,然后用 AAG 描述待检查体系结构,最后用该算法来验证。

2　Master/Slave 结构例子

Master/Slave 结构中,Master 负载被分配给各 Slave,其基本约束是:所有 Slave 总的计算能力必须不小于 Master 要分配的负载。下面用 $ASGG_{MS}$ 描述 Master/Slave 结构风格。假定 Master 负载和 Slave 计算能力都是自然数。

基调 Sig = { *S*, *OP* } ,其中 nat 是自然数类别, bool 是布尔类别。

sorts: nat, bool

opns:

+ , - : nat ×nat→ nat

> , ≤ : nat ×nat→ bool

Sig 代数(Sig-Algebra) 的载体集是自然数集合和布尔数集合,“+ ”“- ”操作就是自然数上的加、减操作,“> ”“≤ ”是自然数的比较大小操作。

符号集合 $V = \{ w, c\}$, $ASGG_{MS} = (L_V, L_{VT}, L_E, L_{ET}, P, G_0, \Gamma, avt, aet)$ 是一个九元组,其中:

(1) $L_V = \{ M, S, X\}$, $L_{VT} = \{ M, S\}$, $L_E = \{ D, T\}$, $L_{ET} = \{ D\}$, M、S 是 M/S 结构中出现的构件类型, M 表示 Master, S 表示 Slave, *D* 是 M/S 结构中出现的连接子类型。

(2) 产生式集合 $P = \{ p_1, p_2\}$, p_1, p_2 的描述分别如图 1、2 所示。

(3) 初始图 G_0 如图 3 所示。

(4) 类别指派 $\Gamma = \{ w: nat, c: nat\}$, *OP* 中操作符号表明了可以进行的操作。

(5) $avt(M) = w: nat$, $avt(S) = c: nat$。对 M 类型构件指派属性 *w*, 表示其将分配的负载; S 类型构件指派属性 *c*, 表示其可提供的计算能力。

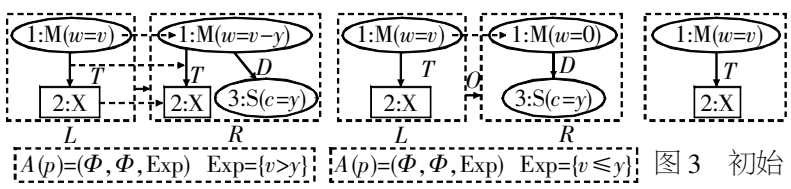

图 1　产生式 p_1　　图 2　产生式 p_2　　图 3　初始图 G_0

产生式 p_1 表示在 M/S 结构中新增一个 S 标记节点。在产生式 p_1 左部图 L 中, 1　M($w = v$) 表示节点 1 的标记是 M, 属性 *w* 值为 *v*; 2　X 表示节点 2 的标记为 X。右部图 *R* 中1　M($w = v - y$) 表示节点 1 标记为 M, 属性 *w* 值为 $v - y$; 2　X 表示节点 2 的标记为 X; 3　S($c = y$) 表示节点 3 标记为 S, 属性 *c* 值为 *y*。$q: L \rightarrow R$ 是属性图态射(产生式中虚箭头表示图态射,自然数上代数同态显而易见)。*A*(*p*) 是应用条件,肯定否定应用条件均为空,代数表达式约束 $v > y$ 表示节点 1 分配的负载大于新加入节点 3 提供的计算能力。产生式 p_2 的含义与 p_1 类似,它表示删除非终结节点,并加入一个 S 标记节点。在这个产生式集合中, M/S 结构的基本约束表达为: 加入新的 S 标记节点,从 M 标记节点的负载中减去 S 标记节点提供的计算能力,最后不再加入 S 标记节点, M 标记节点的负载值变为 0,则得到的 M/S 结构图满足约束。

初始图 G_0 中1　M($w = v$) 表明 M 标记节点初始时要分配的负载为 *v*, 从初始图出发,使用产生式进行推导,可以得到一系列的 M/S 结构实例属性图。图 4 是从初始图出发利用产生式序列 $p_1 p_2$ 推导得到 M/S 体系结构实例属性图的例子。在由初始图进行推导时,首先要确定属性 *w* 的值,在使用某个产生式时,要给出所加入的 S 标记节点的属性 *c* 的值。在图 4 中,为了讨论方便,假定 $w = 40$, 并且每次加入的 S 标记节点属性 $c = 20$。

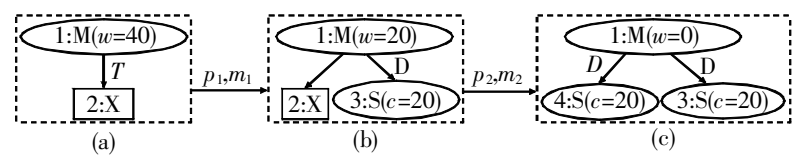

图 4　产生式推导的例子

接下来定义规则集合 $R = \{ r_1, r_2, r_3\}$, M/S 结构属性图基于此规则集进行演化。规则 r_2, r_3 分别与产生式 p_1, p_2 一样。r_1 的定义如图 5 所示,它表示 M 标记节点新增负载,为了能新增 S 标记节点承担新增负载,引入 X 标记节点。其否定应用条件中否定约束是 *n*,其含义是指体系结构图中不能有已经出现 X 标记的节点,即 X 标记节点最多只能出现一次。

图 4(c) 表示的 M/S 结构属性图中节点 1 的负载发生了变化,增加 40。该体系结构实例属性图要进行模拟演化,表现为基于规则集 *R* 的属性图转换,按照规则序列 $r_1 r_2 r_3$ 进行转换(这个基于规则序列的转换与图 4 产生式推导的例子类似),得到模拟执行后体系结构实例属性图 AAG′,如图 6 所示。正如前文所述,要对 AAG 进行约束验证以保证演化将是合法的,产生式 p_1, p_2 满足规定的产生式限制条件,因此可使用前述约束检查算法进行检查,发现可以找到产生式序列 $p_1 p_1 p_1 p_2$。从初始图 G_0($w = 80$) 按此序列进行推导得到的图与 AAG′是属性图同构的。因此演化模拟执行后得到的体系结构属性图 AAG′仍然遵循 M/S 结构风格,该演化合法,可被真正执行。

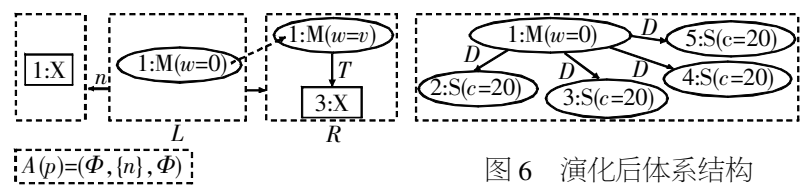

图 5　转换规则 r_1　　图 6　演化后体系结构实例属性图

3　检查器设计与实现

利用 TU Berlin 的属性图文法系统 AGG 的图形解析器设计和实现了体系结构自动检查器。AGG 使用 SPO 的代数方法进行属性图转换,有较强的图形解析能力。该检查器应用到了面向体系结构的服务集成开发平台 Artemis-ARC 系统中,可对属性图描述的体系结构实例进行约束检查,对其他描述方式的体系结构实例,将其转换为属性图描述后,仍可进行检查。

3.1　设计

使用属性图文法方法进行约束验证时,首先要有属性图文法描述的体系结构风格作为验证依据,然后将非属性图描述的体系结构转换为属性图描述形式,最后进行约束验证。因此这个自动检查器主要由体系结构风格描述库、体系结构描述转换器、约束检查器三部分组成,如图 7 所示。体系结构风格描述

库中存储预定义好的体系结构风格属性图文法描述;体系结构描述转换器负责将待验证体系结构图转换为属性图描述形式,约束检查器对其进行检查。

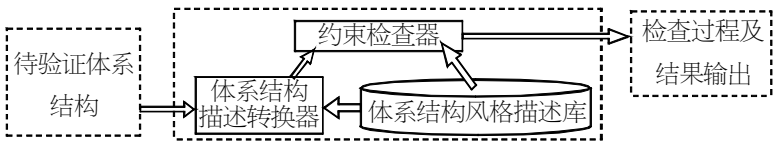

图 7　体系结构检查器结构图

3.2　改进与实现

提出 AGG 图形解析器的初衷是支持图形规约检查,因此其在推导检查上存在缺陷。在使用形如 $p = (A \rightarrow aA)$ 的产生式进行推导时,可能会不停使用该产生式造成检查无法终止,这时可对产生式的前提条件加以限制,但不是所有情况下该方法都有效。前文给出了产生式限制条件,要求规则右部元素不少于左部元素,被重复使用的产生式会不停增加图形元素数,最终元素数目将超过待检查图元素数目。因此第二种方法是修改 AGG 的 API,在检查过程中加入元素数目检查,在推导中检查转换图元素数目是否超过待检查图形元素数目;如超过,则回溯,尝试使用其他产生式进行推导。另外,改进了事件触发机制,使检查过程可以更详细地显示出来。在改进 AGG 的图形解析器基础上,实现了体系结构自动检查器。

(1) 体系结构风格描述库:利用 AGG 提供的可视化编辑环境,预先编辑多种体系结构风格的属性图文法描述,这些体系结构风格描述以 XML 文件存放。要注意的是,AGG 图形解析器中规则的尝试次序是根据 XML 文件中规则存放的顺序进行的,因此在编辑图文法时注意规则的编辑顺序可以减少检查过程回溯次数,在一定程度上提高检查效率。

(2) 体系结构描述转换器:从体系结构风格描述库中得到构件、连接器类型信息,将非属性图描述的体系结构转换为 AGG 中属性图描述的形式。其基本思想是首先将运行时体系结构对象中构件、连接子转换为 AGG 中的节点、边,然后根据体系结构对象的拓扑信息构造得到体系结构实例属性图。在转换过程中进行了初步检查。如果无法完成从体系结构对象到属性图描述的转换,即转换时发现无法找到与构件(连接子)属性 Type 值匹配的节点(边)类型。此时检查器即可判定该体系结构对象不属于预定的体系结构风格,而无须进行下一步检查。

(3) 约束检查器:调用改进的图形解析器进行检查,从体系结构描述转换器中得到属性图描述的待检查体系结构,从体系结构风格描述库中载入体系结构风格图文法描述和初始图,根据这些判断待检查图是否约束满足。

3.3　效果

体系结构需动态演化时演化模拟执行,检查器对模拟执行后的体系结构进行约束验证,如演化合法,使动态演化真正付诸执行。图 8 是对 M/S 结构模拟演化后进行约束验证的例子。

三个体系结构是模拟演化后的体系结构,M, S 是构件类型,D 是连接子类型。图 8(a) 中 $Comp_1$ 属性值 $w = 0$ 时,检查器将该体系结构转换为图 4(c) 属性图的形式,从初始图按产生式序列 $p_1 p_2$ 推导,图 4(c) 可达,演化合法;当 $w \neq 0$ 时,不满足 M/S 结构的基本约束,检查器无法找到合适的产生式序列,演化非法。图 8(b) 在进行体系结构描述转换时,发现类型为 S 的构件找不到匹配节点,检查器报错。图 8(c) 中连接子 Con_1 方向有错,检查器报错。

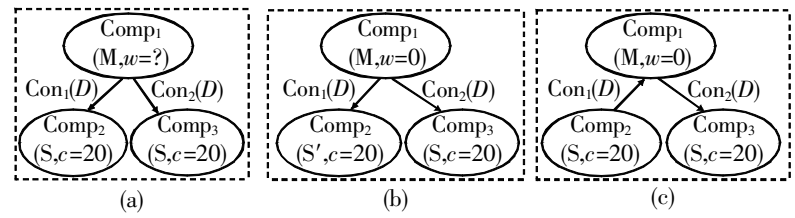

图 8　约束验证

4　结束语

讨论了一种基于属性图文法的体系结构描述和约束验证方法。首先使用属性图文法形式地描述了体系结构风格、体系结构实例和演化,给出约束满足定义和约束检查算法;然后以 M/S 结构为例说明如何基于属性图文法描述体系结构和进行约束验证;最后设计和实现了基于 AGG 的图形解析器体系结构检查器,并将其应用到了 Artemis-ARC 系统中。这种基于属性图文法的体系结构描述和约束验证方法直观易理解,相对于文献[11, 13, 14] 的基于图的描述和验证方法,刻画了体系结构的属性信息,进行了包括属性在内的约束验证。

本文中体系结构演化首先被模拟执行,并对模拟执行后体系结构进行约束验证;如满足约束,则演化被真正付诸执行,该方法效率相对较低。体系结构的演化表现为体系结构实例属性图转换,因此下一步工作是在体系结构实例属性图转换过程中进行检查,判断每一步转换是否会导致最后得到的体系结构实例属性图违反约束。该方法效率相对较高,并且可以在演化过程中尽早地判断最后的结果是否会违反约束。

附录

定义 1　标记图(Labeled Graph)[12]。令 L_V 是节点标记集合,L_E 是边标记集合,L_V 和 L_E 上的一个标记图是 $G = (V, E, s, t, lv, le)$。其中:

(1) V 是有限个节点的集合,E 是有限个边的集合。

(2) $s: E \rightarrow V$ 是源节点函数,$t: E \rightarrow V$ 是目标节点函数。

(3) $lv: V \rightarrow L_V$ 是节点标记函数,$le: E \rightarrow L_E$ 是边标记函数。

一个全图态射(Total Graph Morphism) $f: G \rightarrow G$ 是二元组 $f = (f_V: V_G \rightarrow V_{G'}, f_E: E_G \rightarrow E_{G'})$,$f_V$、$f_E$ 是映射,满足 $f_V \circ t_G = t_{G'} \circ f_E$、$f_V \circ s_G = s_{G'} \circ f_E$、$lv_{G'} \circ f_V = lv_G$、$le_{G'} \circ f_E = le_G$。对于全图态射 $f: G \rightarrow G'$,如果 f_V、f_E 是双射,则称图 G 与 G 是同构的,记做 $G \cong G'$。一个部分图态射(Partial Graph Morphism) $f: G \rightarrow G$ 是一个从 G 的某些子图到 G 的全图态射,显然部分图态射包括全图态射。

定义 2　属性图(Attributed Graph)[15]。L_V 是节点标记集合,L_E 是边标记集合,基调 $Sig = (S, OP)$,则 $AG = (V, E, s, t, lv, le, AG_A, av, ae)$ 是基调为 Sig 的属性图,其中:

(1) V, E, s, t, lv, le 构成了一个在 L_V 和 L_E 上的标记图。

(2) AG_A 是一个 Sig 代数。

(3) $av: V \rightarrow U(AG_A)$ 和 $ae: E \rightarrow U(AG_A)$ 分别是节点和边的属性函数。

属性图态射(Attributed Graph Morphism)是指两个基调为 Sig 的属性图 $G_i = (V_{Gi}, E_{Gi}, s_{Gi}, t_{Gi}, lv_{Gi}, le_{Gi}, AG_{Ai}, av_{Gi}, ae_{Gi})$ $i = 1, 2$ 上的一个元组 $f = (f_G, f_A)$。其中 $f_G = (f_V, f_E)$ 是从 G_1 到 G_2 部分图态射;$f_A: AG_{A1} \rightarrow AG_{A2}$ 是代数同态映射,满足 $\forall v \in dom(f_V)$ $U(f_A((av_{G1}(v)))) = av_{G2}(f_V(v))$ 且 $\forall e \in dom(f_E)$ $U(f_A((ae_{G1}(e)))) = ae_{G2}(f_E(e))$[18]。如果 f_V、f_E、f_A 都是同构映射(即双射),则 G_1 与 G_2 是属性图同构,记为 $G_1 \cong_{AG} G_2$;如果 f_G 是一个全图态射,则称 f 是子属性图匹配态射。

定义 3　属性图产生式(Graph Productions)。属性图产生式是一个带条件的产生式,$p = (L \xrightarrow{q} R, A(p))$。$L$ 是产生式左部,R 是产生

式右部, q 是属性图态射, $A(p) = (AP(p), AN(p), \mathrm{Exp})$ 是产生式应用条件。$AP(p)$ 是肯定应用条件(Positive Application Condition), $AN(p)$ 是否定应用条件(Negative Application Condition), Exp 是属性的代数表达式约束(如 $x > y$)集。$L \xrightarrow{l} L$ 是一个肯定(否定)约束, $L \xrightarrow{m} G$ 是子属性图匹配态射。如果存在一个属性图态射 $L' \xrightarrow{n} G$, 使得 $n \circ l = m$, 则称 m P-满足 l, 记为 $m \models_P l$; 如果 m 不是 P-满足 l, 则称 m N-满足 l, 记为 $m \models_N l$。Exp 中所有表达式约束均满足, 记为 Satisfy(Exp)。对子属性图匹配态射 $L \xrightarrow{m} G$, 如果 $\forall l \in AP(p) \cdot m \models_P l \wedge \mathrm{P}\ k \in AN(p) \cdot m \models_N k \wedge$ Satisfy(Exp), 则 m 满足 $A(p)$, 记为 $m \models_P A(p)$。根据 $L \xrightarrow{m} G$ 对属性图 G 施以规则 $p = (L \xrightarrow{q} R, A(p))$ 得到图 H, 记为 $G \xrightarrow{p, m} H$, 这个推导一般按照 SPO 的代数方法进行[17]。一个形如 $G_0 \xrightarrow{p_i, m_i} G_1 \xrightarrow{p_j, m_j} \cdots \xrightarrow{p_k, m_k} G_k$ 的推导式序列构成了从 G_0 到 G_k 的推导, $p_i, p_j, \cdots, p_k \in P$, $p_i p_j \cdots p_k$ 称为一个产生式序列, 此推导序列记为 $G_0 \rightarrow_p^* G_k$。

定义 4 基调[16]。基调 $\Sigma = \langle S, OP \rangle$ 由下列两部分组成: ①集合 S, 其元素叫做类别(Sort); ②二元组 $\langle op: s_1 \times \cdots \times s_k \rightarrow s \rangle$ 的集合 OP, 其中 $s_1, \cdots, s_k, s \in S$ 并且 op 不出现在两个不同的二元组中。

定义 5 代数[16]。如果 $\Sigma = \langle S, OP \rangle$ 是一个基调, 那么 Σ 代数 A 包含: ①对每个 $s \in S$, 正好有一个载体 A^s; ②一个解释映射 I 把函数 $I(op): A^{s1} \times \cdots \times A^{sk} \rightarrow A^s$ 指派给函数符号 $op: s_1 \times \cdots \times s_k \rightarrow s \in OP$。

参考文献:

[1] 马晓星, 余萍, 陶先平, 等. 一种面向服务的动态协同架构及其支撑平台[J]. 计算机学报, 2005, **28**(4): 467-477.

[2] ANDERSSON J. Issues in dynamic software architectures: proc. of the 4th Int: Software Architecture Workshop (ISAW-4) [C]. [S. l.]: [s. n.], 2000: 111-114.

[3] The attributed graph grammar system: a development environment for attributed graph transformation systems[EB/OL]. [2005-05]. http://tfs.cs.tu-berlin.de/agg/index.html.

[4] GARLAN D, MONROE R T, WILE D. Acme: architectural description of component-based systems[C]//LEAVENS G T, SITARAMAN M. Foundations of component-based systems. Cambridge: Cambridge University Press, 2000: 47-68.

[5] MONROE R T. Capturing software architecture design expertise with armani, CMU-CS-163[R]. Pittsburgh: Carnegie Mellon University, 1998.

[6] The wright architecture description language[EB/OL]. [2005-05]. http://www-2.cs.cmu.edu/~able/wright.

[7] Wright analysis tutorial[EB/OL]. [2005-05]. http://www-2.cs.cmu.edu/afs/cs/project/able/ftp/Wright-tutorial.

[8] A new implementation for the darwin architectural description language[EB/OL]. [2005-05]. http://www.doc.ic.ac.uk/~igeozg/Project/Darwin.

[9] RADESTOCK M, Eisenbach S. What do you get from a Π-calculus semantics: proc. of the 6th International PARLE Conference on Parallel Architectures and Languages Europe[C]. [S. l.]: [s. n.], 1994: 635-647.

[10] RADESTOCK M, Eisenbach S. Formalising system structure: proc. of the 8th ACM/IEEE Int. Workshop on Software Specification and Design (IWSSD) [C]. [S. l.]: [s. n.], 1996.

[11] DANIEL L M. Describing software architecture styles using graph grammars[J]. **IEEE Transactions on Software Engineering**, 1998, **24**(7): 521-533.

[12] CORRADINI A, MONTANARI U, ROSSI F G. Algebraic approach to graph transformation part I: basic concepts and double pushout approach[M]//ROZENBERG G. Handbook of graph grammars and computing by graph transformation: foundations vo1. 1. Singarpore: World Scientific Publishing, 1997: 183.

[13] 马晓星. Internet 软件协同技术研究[D]. 南京: 南京大学, 2003: 86-94.

[14] DAN H, PAOLA I, MONTANARI U. Graph grammars and constraint solving for software architecture styles: proc. of the 3th Int. Workshop on Software Architecture[C]. Orlando: [s. n.], 1998: 69-72.

[15] EHRIG H, HECKEL R, KORFF M, *et al*. Algebraic approach to graph transformation part II: single pushout approach and comparison with double pushout approach[M]//ROZENBERG G. Handbook of graph grammars and computing by graph transformation. Singorpore: World Scientific Publishing, 1997.

[16] MEINKE K, TUCKER J V. Universal algebra[M]//ABRAMSKY S, GABBY D, MAIBAUMT S E. Handbook of logic in computer science (vol. 1): background: mathematical structures. New York: Oxford University Press, 1993: 189-368.

[17] WAGNER A. Attributed graph transformations with controlled application of rules: proc. of Colloquium on Graph Transformation and its Application in Computer Science[C]. [S. l.]: [s. n.], 1994.

[18] REN Zhihong, CAO Jiannong, CHAN A T S, *et al*. Toward a formal approach to composite Web service construction and automation: proc. of the 2003 International Conference on Parallel Processing (ICPP'03) [C]. [S. l.]: [s. n.], 2003: 436-443.

[19] BARESI L, HECKEL R, TH NE S, *et al*. Modeling and analysis of architectural styles based on graph transformation: the 6th ICSE Workshop on Component-based Software Engineering: Automated Reasoning and Prediction[C]. Pittsburgh: Carnegie Mellon University, Australia: Monash University, 2003: 67-72.

[20] 张幸儿. 计算机编译原理[M]. 北京: 科学出版社, 2001.

[21] REKERS J, SCH RR A. A parsing algorithm for context-sensitive graph grammars technical report[R]. Leiden: Leiden University, 1995.

[22] 冉平, 石兵, 马晓星, 等. 上下文相关图文法分析及其应用初探[J]. 计算机科学, 2006, **33**(3): 255-260.

(上接第 159 页)提高。电子邮件的安全也在这个过程中得到了更好的发展和充实。

参考文献:

[1] 龙勤, 李斌, 潘爱民. 一种实用的前向安全 OpenPGP 扩展[J]. 小型微型计算机系统, 2005, **26**(8): 1301-1305.

[2] 王安国, 胡铭曾. OpenPGP 协议在电子邮件系统中的应用[J]. 哈尔滨师范大学学报: 自然科学版, 2005, **21**(4): 51-53.

[3] 陈建奇, 张玉清, 李学农, 等. 安全电子邮件的研究与实现[J]. 计算机工程, 2002, **28**(6): 121-122.

[4] 尤枫, 薛峰, 赵恒永. 基于 S/MIME 协议的电子邮件安全研究与实现[J]. 北京化工大学学报, 2004, **31**(4): 106-108.

[5] BLUM R. 开放源码邮件系统安全[M]. 杜鹏, 译. 北京: 人民邮电出版社, 2002: 253-324.

[6] 周建国, 晏蒲柳, 李凯. 实现双向认证的安全电子邮件系统的研究[J]. 计算机应用研究, 2004, **21**(12): 121-122.

[7] CALLAS J, DONNERHACKE L, FINNEY H, *et al*. RFC 2440-1998 Open-PGP message format[S]. [S. l.]: Network Working Group, 1998.

[8] LINN J. RFC 1421-1993 Privacy enhancement for Internet electronic mail part Ⅰ: massage encryption and authentication procedures[S]. [S. l.]: [s. n.], 1993.

[9] KENT S. RFC 1422-1993 Privacy enhancement for Internet electronic mail part Ⅱ: certificate-based key management[S]. [S. l.]: [s. n.], 1993.

[10] ZIMMERMANN P. Why OpenPGP's PKI is better than an x. 509 PKI[EB/OL]. [2001-02-27]. http://www.Open pgp. Org/technical/whybetter. shtml.