doi:10.3969/j.issn.1006-7043.2010.12.016

产生式知识库的不动点计算建模方法

黄宏涛,黄少滨,张 涛
(哈尔滨工程大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

摘 要:提出了利用产生式规则的过程性特征对产生式知识库进行建模的方法.该方法在给出条件迁移定义的基础上引入不动点计算构建条件迁移系统,将模型构建过程转换为求解状态集合构造函数最小不动点的过程,并给出了求解算法的正确性与可终止性证明;该方法的时间复杂度相对静态建模方法降低了一个量级,构造的条件迁移系统包含了状态迁移过程的完整信息,同时解决了静态建模方式构建的迁移系统信息丢失问题,提高了错误诊断的效率.

关键词:产生式知识库;模型检测;条件迁移系统;建模;最小不动点
中图分类号:TP311.5 **文献标志码**:A **文章编号**:1006-7043(2010)12-1655-07

A modeling method based on fixed point computation for a production knowledge base

HUANG Hong-tao, HUANG Shao-bin, ZHANG Tao
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract: A modeling method based on the procedural nature of production rules was proposed to build a system model. The definition of the conditional transition system(CTS) was given by the concept of conditional transitions. Small fix point computing was introduced to build the conditional transition system. The system model represented by the conditional transition system was built by solving the small fix point of the constructing function of the states set, and the correctness and termination of this method was proven. The proposed dynamic modeling method improves the efficiency of the production knowledge base modeling process. The time complexity reduced one order of magnitude compared with the static modeling method. The conditional transition system built by this method contains all information about the state transition process, and solves the information loss problem of the transition system constructed by the static modeling method, increasing the efficiency of error diagnosis.

Keywords: production knowledge base; model checking; conditional transition system (CTS); modeling; small fix point

随着各种产生式知识库的广泛应用以及知识库规模的不断膨胀,各种知识错误逐渐成为影响智能系统运行效率的重要原因.大量验证技术也随着产生式知识库的发展不断出现,如基于二维判定表[1]、有向图[2]、Petri 网[3-5]、知识网[6]、DNA 操作[7]以及自适应推理[8]的方法,这些方法能够有效地发现各种知识错误.然而,这些方法中多数将验证过程局限在知识库中规则集本身.难以发现在知识库运行过程中才能暴露出来的更为隐蔽的深层错误.目前,解决这一问题常用的方法是测试[9-10],但是测试方法不仅效率低,而且只能覆盖系统运行路径的一小部分.研究人员提出了使用模型检测[11]技术调试产生式知识库[12-13]以及检验规则不一致性与循环规则[14]的方法.对知识库进行建模是该方法中最基础、最耗时的步骤.现有基于模型检测技术的的建模方法普遍忽略了规则的动态过程性特征,是一种静态建模方法(static modeling method,SMM).

SMM 把迁移关系看作状态与状态的一个静态动作的建模方法思路清晰,算法简单,适用于对小规模的知识库进行建模.但将其用于规模较大的知识库时,它存在如下不足之处:1)建模过程忽略了状

收稿日期:2009-11-06.
基金项目:国家自然科学基金资助项目(60873038).
作者简介:黄宏涛(1980-),男,博士研究生,E-mail:horntau@gmail.com;
黄少滨(1965-),男,教授,博士生导师.
通信作者:黄宏涛.

态迁移是由规则条件触发并由规则动作决定迁移目标的动态过程性特征这一基本事实,其直接后果是建模效率不高;2)SMM 方法构建的静态迁移系统(static transition system,STS)中迁移关系为状态与状态笛卡尔积的子集,虽然在这样的标准迁移系统上可以完成模型检测的所有工作,但是,SMM 方法建立的 STS 模型不包含引起状态迁移的错误诊断信息,这是 SMM 方法忽略迁移过程的过程性特征导致的另一个局限性.

本文提出一种利用不动点计算对对产生式知识库建模的动态建模方法(dynamic modeling method,DMM)方法,该方法能够有效提高产生式知识库的建模和错误诊断效率.

1 条件迁移系统

1.1 相关定义

设产生式知识库中全部知识共涉及 m 个变量,$V=\{v_1,v_2,\cdots,v_m\}$ 为这些变量的有穷集合,其中 $m\geqslant 1$,每个变量都有一个确定的数据类型(如布尔型、整形、字符串形等),每个数据类型都对应一个数据域,令 $D(v)$ 表示变量 v 的数据域,其中 $v\in V$.

定义 1 令 $E(V)$ 为定义在变量集合 V 上的集合,$E(V)$ 中的元素由 V 中变量的任意取值组合构成.

定义 2 定义 $f_a: E(V)\rightarrow E(V)$ 为变量集合 V 上的动作函数,令 F 表示动作函数的集合.

定义 3 令 $C(V)$ 为定义在变量集合 V 上的布尔条件集合. 此处布尔条件为具有 $\bar{v}\in\bar{d}$ 形式的命题逻辑公式,$\bar{v}=(v_1,v_2,\cdots,v_n)$ 是一个 n 元组,其中 $v_i\in V, 1\leqslant i\leqslant n, 1\leqslant n\leqslant m$,且如果 $i\neq j$,则 $v_i\neq v_j$,$1\leqslant j\leqslant n$;$\bar{D}=D(v_1)\times D(v_2)\times\cdots\times D(v_n)$.

例如命题 $(v_1,v_2)\in\{(x,y)\in \mathrm{IN}^2 \mid y-x\leqslant 2\}$,为描述方便,将其简写为 $v_2-v_1\leqslant 2$. 则命题 $(v_1>2)\wedge(v_2-v_1\leqslant 2)\vee(v_3=\mathrm{true})$ 为一个合法的命题条件,其中,v_1、v_2 为整形变量,v_3 为布尔变量.

知识库中的规则可以看作一个二元组:

$$r:(c,a).$$

式中:r 为规则标号,c 为规则条件,a 为规则动作或结论. 根据定义 3,c 即为定义在 V 上的布尔条件,于是有 $c\in C(V)$;根据定义 2,a 为定义在 V 上的动作函数,即 $r.a: E(V)\rightarrow E(V)$,即 $a\in F$. 令 R 为知识库中所有规则构成的二元组集合,则有 $R\subseteq C(V)\times F$. R 为有穷集合.

1.2 条件迁移

定义 4 定义表达式 $s\xrightarrow{r.c:\ r.a}s'$ 表示由知识库中规则条件触发的条件迁移过程. 其中 r 为规则标号,$r.c$ 和 $r.a$ 分别表示规则 r 的条件和动作.

条件迁移的操作语义为:当系统运行至某个状态 s 时,根据状态 s 上变量的当前值进行规则匹配,当状态 s 满足规则 r 的条件 $r.c$ 时,触发规则 r 的动作(结论)$r.a$,$r.a$ 对状态 s 上的变量进行操作,改变后的变量值构成一个新的状态 s',s' 可以和 s 相同. 这种情况是因为 $r.a$ 虽然对 s 上的变量进行了操作,但操作的结果没有改变变量值. 简而言之,状态 s 满足规则 $r.c$ 时,规则 r 在状态 s 上执行 $r.a$ 到达状态 s'.

过程性规则对状态迁移的影响可以直观地描述为上述过程. 对于陈述性规则,可把规则结论部分的事实陈述当做变量赋值动作,即把陈述性规则当做过程性规则处理.

1.3 条件迁移系统

由于 SMM 构建的系统模型不包含引起状态迁移原因的相关信息,而这部分信息对于错误调试又是非常重要的. 因此,一个完备的迁移系统应该包含条件迁移信息,下面给出条件迁移系统的定义.

定义 5 定义在变量集合 V 上的条件迁移系统(conditional transition system,CTS)是一个六元组 (S,A,T,S_0,AP,L).

1)S 为状态集合,$S\subseteq E(V)$;

2)A 为动作集合,$A\subseteq F$;

3)$T\subseteq S\times C(V)\times F\times S$ 为迁移集合;

4)$S_0\subseteq S$ 为初始状态集合;

5)AP 为原子命题集合,$AP=C(V)$;

6)$L: S\rightarrow 2^{AP}$ 为标签映射函数.

定义 6 令 $\tilde{S}$ 为定义在 CTS 上的一个状态集合,且有 $\tilde{S}\subseteq S$,则有

1)$\tilde{S}$ 的前趋状态集合为

$$\mathrm{Pre}(\tilde{S})=\{s\in S\mid \exists s', s\xrightarrow{r.c:\ r.a}s'\wedge s'\in\tilde{S}\};$$

2)$\tilde{S}$ 的后继状态集合为

$$\mathrm{Post}(\tilde{S})=\{s\in S\mid \exists s', s'\xrightarrow{r.c:\ r.a}s\wedge s'\in\tilde{S}\}.$$

定义 7 称状态序列 $\pi=s_1,s_2,\cdots,s_n$ 为一条从 s_1 到 s_n 的路径,当 $s_2\in\mathrm{Post}(s_1)$,$s_3\in\mathrm{Post}(s_2)$,$\cdots$,$s_n\in\mathrm{Post}(s_{n-1})$ 成立,其中 $s_1\in S, s_2\in S,\cdots, S_n\in S$,定

义路径 π 的长度 $|\pi|$ 为 n. 若 s_1 到 s_n 存在路径,则称从 s_1 可达 s_n.

定理 1　设 $s_1, s_n \in S$,如果存在从 s_1 到 s_n 的路径,则必存在一条路径 π 使得从 s_1 可达 s_n,且有 $|\pi| \leqslant |S|$ 成立.

证明　假设 $\pi_{\min} = s_1, s_2, \cdots, s_n$ 是一条从 s_1 可达 s_n 的最短路径,且 $|\pi_{\min}| > |S|$. 因为集合 S 的基数为 n,所以至少有一个 S 的元素重复出现在路径 $\pi_{\min}$ 中,令 $s_i = s_j$,为 $\pi_{\min}$ 中的重复元素,其中 $1 \leqslant i < j \leqslant n$,于是可得另外一条从 s_1 可达 s_n 的路径 $\pi = s_1, s_2, \cdots, s_i, s_j, \cdots, s_{n-1}, s_n$,并且有 $|\pi| < |\pi_{\min}|$,这与假设 $\pi_{\min}$ 是从 s_1 可达 s_n 的最短路径矛盾.

2　CTS 构造函数

根据条件迁移和 CTS 的定义,可以把知识库中的规则集转换为 CTS. 基本思想是由初始状态出发,模拟条件迁移的动态迁移过程,从由条件迁移关系生成的可达关系出发,寻找所有能够由条件迁移到达的新目标状态,将其加入状态集合,同时生成迁移关系,然后在扩展后的状态集合上进行迭代,直到没有新状态产生为止. 这个模型构造过程中最核心的步骤是系统状态集合与条件迁移集合的构建,其中 CTS 状态集合的构建过程本质上同最小不动点计算过程是一致的[15]. 与 SMM 方法相比,这种根据条件迁移的可达关系逐步动态扩展构建系统模型方法的一个显著特征是状态集与条件迁移关系集合是在同一过程中同时动态生成的,而不是先构造状态集合,再根据状态集合构建迁移关系集合.

定义 8　定义函数 $G: 2^{E(V)} \to 2^{E(V)}$,称 G 是单调的,当且仅当对 $E(V)$ 的任意子集 X 和 Y,若有 $X \subseteq Y$ 成立,则有 $G(X) \subseteq G(Y)$ 成立.

定义 9　称 X 为 G 的一个不动点,当且仅当 $G(X) = X$,称 X 为 G 的最小不动点,当且仅当对 G 的任意不动点 Y,有 $X \subseteq Y$ 成立.

根据条件迁移的特点以及 CTS 的定义,可以采用模拟条件迁移的动态迁移过程的方法构建 CTS,CTS 状态集合的形式化定义如下.

定义 10　在 CTS 上定义 S 为系统状态集合,则集合 S 中元素为由初始状态集合 S_0 出发可达的所有状态,可描述为

1) $S_0 \subseteq S$;

2) 对任何集合 $X \subseteq S$, $\mathrm{Post}(X) \subseteq S$;

3) 只有由有限次使用 1) 和 2) 生成的集合 $X \subseteq S$.

根据这个递归定义,可以得出 CTS 状态集合的构造函数.

定义 11　定义函数 $G(X) = S_0 \cup X \cup \mathrm{Post}(X)$,则集合 CTS 状态集合 S 可以通过对函数 $G(X)$ 进行迭代计算获得.

定理 2　函数 $G(X)$ 对任意集合 $X \in 2^{E(V)}$ 是单调的.

证明　令 X、Y 为集合 $E(V)$ 的任意子集,且有 $X \subseteq Y$ 成立,问题的证明归结于 $\mathrm{Post}(X) \subseteq \mathrm{Post}(Y)$ 是否成立. 当 $X = Y$ 时,由定义 5 知,$\mathrm{Post}(X) \subseteq \mathrm{Post}(Y)$ 成立;当 $X \neq Y$ 时,设 X 有 j 个元素,Y 有 k 个元素,其中 $0 \leqslant j \leqslant k$,集合 Y 可表示为

$$Y = X \cup \{s_{j+1}, s_{j+2}, \cdots, s_k\},$$

于是

$$\mathrm{Post}(Y) = \mathrm{Post}(X) \cup \mathrm{Post}(\{s_{j+1}, s_{j+2}, \cdots, s_k\}).$$

综上所述, $\mathrm{Post}(X) \subseteq \mathrm{Post}(Y)$ 成立,所以 $(S_0 \cup X \cup \mathrm{Post}(X)) \subseteq (X_0 \cup Y \cup \mathrm{Post}(Y))$ 成立,也即 $G(X) \subseteq G(Y)$ 成立,由定义 8 可知 $G(X)$ 对任意集合 $X \subseteq 2^{E(V)}$ 是单调的.

由于“$\subseteq$”构成集合 2^S 上的偏序关系,2^S 是在该偏序关系下的完全格,且对任意 $a \in 2^S$ 都有唯一确定的上确界与下确界,而定理 2 确保了 $G(X)$ 是次序保持函数,这些条件符合 Knaster-Tarski 定理[16-17]的要求. 于是由 Knaster-Tarski 定理可得函数 G 的最小不动点是 $G^{\alpha}(\varnothing)$ 的固定极限,把表达式

$$\underbrace{G(G(\cdots G(X)\cdots))}_{\alpha\text{次}}$$

记为 $G^{\alpha}(X)$,表示 $G(X)$ 在 X 上作用 α 次,其中 $\alpha \geqslant 0$,这里的 $G^{\alpha}(\varnothing)$ 采用超限归纳法定义,即 $G^{\alpha+1}(\varnothing) = G(G^{\alpha}(\varnothing))$,并且对于极限序数 γ, $G^{\gamma}(\varnothing)$ 是 $G^{\beta}(\varnothing)$ 对于所有小于 γ 的基数 β 的上确界. 于是,由 Knaster-Tarski 定理可得以下基本结论.

定理 3　设 $E(V)$ 是有 n 个元素的有限状态集合 $\{s_1, s_2, \cdots, s_n\}$, $G: 2^{E(V)} \to 2^{E(V)}$ 是定义在 $E(V)$ 上的单调函数,则:

1) $G^n(\varnothing)$ 是函数 G 的不动点;

2) $G^n(\varnothing)$ 是函数 G 的最小不动点.

证明

1) $G^n(\varnothing)$ 是函数 G 的不动点.

因为 $\varnothing \subseteq G^1(\varnothing)$,且 G 为单调函数,由定义 6 知 $G^1(\varnothing) \subseteq G^1(G^1(\varnothing))$ 成立,即有 $G^1(\varnothing) \subseteq G^2(\varnothing)$. 由数学归纳法可以证明对所有 $1 \leqslant m \leqslant n$:

$$G^1(\varnothing) \subseteq G^2(\varnothing) \subseteq G^3(\varnothing) \subseteq \cdots \subseteq G^m(\varnothing)$$

成立,即 $G^m(\varnothing) \subseteq G^{m+1}(\varnothing)$ 成立.

令 $m=n$,对于 $1 \leqslant i \leqslant m$,假设不存在状态集合 $G^1(\varnothing)$,使得 $G^1(\varnothing)$ 是 G 的不动点,此时 $G^1(\varnothing)$ 至少包含 1 个状态,因为 $\varnothing \neq G^1(\varnothing)$,同理,$G^2(\varnothing)$ 至少包含 2 个状态,因为 $G(\varnothing) \neq G^2(\varnothing)$,依次类推,当 $i=n$ 时,$G^n(\varnothing)$ 至少包含 n 个状态,此时由于 $G^m(\varnothing)$ 也不是 G 的不动点,这蕴含着 $G^{n+1}(\varnothing)$ 至少包含 $n+1$ 个状态,这与 $E(V)$ 只有 n 个元素矛盾,因此,至少存在一个 i 使 $G^i(\varnothing)$ 是 G 的不动点,此时有 $G^i(\varnothing) = G^{i+1}(\varnothing) = \cdots = G^n(\varnothing)$ 成立,所以 $G^n(\varnothing)$ 是 G 的不动点.

2)$G^n(\varnothing)$ 是函数 G 的最小不动点.

假设存在任意状态集合 $Y \in 2^{E(V)}$,且 Y 为 G 的一个不动点. 因为 $\varnothing \subseteq Y$,且 G 为单调函数,所以 $G(\varnothing) \subseteq G(Y)$,由归纳法可得 $G^n(\varnothing) \subseteq G^n(Y)$,又因 Y 为 G 的不动点,所以有 $G^n(Y) = Y$,故 $G^n(\varnothing) \subseteq Y$ 成立,因此 $G^n(\varnothing)$ 为 G 的最小不动点.

定理4 设 $E(V)$ 有 n 个元素,S 为 CTS 的状态集合,X 为 $E(V)$ 的子集,则 S 是函数 $G(X) = S_0 \cup X \cup \mathrm{Post}(X)$ 的最小不动点,即 $S = G^n(\varnothing)$.

证明

1)证明 $G^n(\varnothing) \subseteq S$. 因为 $G^1(\varnothing) = S_0 \cup \varnothing \cup \mathrm{Post}(\varnothing) = S_0$,由定义 10 有 $G^1(\varnothing) \subseteq S$,设 $G^i(\varnothing) \subseteq S$,其中 $1 \leqslant i < n$,$G^{i+1}(\varnothing) \subseteq S_0 \cup G^i(\varnothing) \cup \mathrm{Post}(G^i(\varnothing))$,由于 $G^i(\varnothing) \subseteq S$,根据定义 10 可得 $\mathrm{Post}(G^i(\varnothing)) \subseteq S$,所以 $S_0 \cup G^i(\varnothing) \cup \mathrm{Post}(G^i(\varnothing)) \subseteq S$,即 $G^{i+1}(\varnothing) \subseteq S$,由纳法可得 $G^n(\varnothing) \subseteq S$ 成立;

2)证明 $S \subseteq G^n(\varnothing)$. 令 $s \in S$,若 $s \in S_0$,根据定义 11 有:

$$s \in G^1(\varnothing) \subseteq G^2(\varnothing) \subseteq \cdots \subseteq G^n(\varnothing).$$

若 $s \notin S_0$,由 S 的定义与定理 1 知必存在一条长度小于等于 n 路径 $\pi = s_1 s_2 s_3 \cdots s_i$,其中 $s_1 \in S_0$,$s_i = s$,使得 $s_2 \in \mathrm{Post}(s_1)$,$s_3 \in \mathrm{Post}(s_2)$,$\cdots$,$s_i \in \mathrm{Post}(s_{i-1})$ 成立,$1 < i \leqslant n$,由于

$$s_1 \in S_0 = S_0 \cup \varnothing \cup \mathrm{Post}(\varnothing) = G^1(\varnothing),$$

且

$$s_2 \in \mathrm{Post}(s_1) \subseteq \mathrm{Post}(S_0) \subseteq S_0 \cup S_0 \cup \mathrm{Post}(S_0) = G^1(G^1(\varnothing)) = G^2(\varnothing).$$

假设 $i>1$ 时有 $s_i \in G^i(\varnothing)$ 成立,因为 $s_{i+1} \in \mathrm{Post}(s_i)$ 且 $s_i \in G^i(\varnothing)$,于是 $s_{i+1} \in \mathrm{Post}(s_i) \subseteq \mathrm{Post}(G^i(\varnothing))$ 成立,又因为

$$\mathrm{Post}(G^i(\varnothing)) \subseteq S_0 \cup G^i(\varnothing) \cup \mathrm{Post}(G^i(\varnothing)) = G^{i+1}(\varnothing)$$

成立,所以 $s_{i+1} \in G^{i+1}(\varnothing)$ 成立,由数学归纳法知 $s = s_i \in G^i(\varnothing) \subseteq G^{i+1}(\varnothing) \subseteq \cdots \subseteq G^n(\varnothing)$ 成立.

综上所述,对任意 $s \in S$,有 $s \in G^n(\varnothing)$ 成立,所以有 $S \subseteq G^n(\varnothing)$ 成立.

3 最小不动点计算

定理 3 断言了函数 $G(X)$ 的最小不定点的存在性,同时也给出了求解函数 $G(X)$ 最小不动点的基本步骤,定理 4 进一步证明了状态集合 S 就是函数 $G(X)$ 的最小不动点. 于是,CTS 状态集合 S 的构建可以通过计算 $G(X)$ 的最小不定点实现. 由定理 3 可得函数 $G(X)$ 最小不动点的求解思想为:求函数 $G(X)$ 在输入 $\varnothing$ 时的结果状态集合,再将此结果输入 $G(X)$,求得新的结果状态集合,依次迭代,直到结果在 $G(X)$ 的作用下不再变化为止. 令 S_i 表示第 i 次迭代结果,则求解最小不动点的具体过程为:$S_1 = G(\varnothing)$,$S_2 = G(S_1)$,$\cdots$,$S_i = G(S_{i-1})$,$S_{i+1} = G(S_i)$,由于 $G(X)$ 为单调函数,故对 $1 \leqslant i \leqslant n$ 有下式成立:

$$S_1 \subset S_2 \subset \cdots \subset S_i = S_{i+1} = \cdots = S.$$

上述通过计算 $G(X)$ 最小不动点求解状态集合 S 的过程没有考虑迁移关系的生成,实际上新状态的产生过程是通过匹配规则集确定条件迁移关系的过程,即求解当前状态集的后继状态集合的过程,这个求解过程的前提条件是条件迁移关系的确定. 因此,迁移关系集合 S 的生成可以通过生成状态集合的附加操作实现. 下面以图 1 所示的条件迁移系统为例说明通过不动点计算构造 CTS 的过程.

图 1 所示 CTS 有 8 个状态,初始状态集合 $S_0 = \{s_1, s_4\}$,每个迁移关系上的标签表示触发条件迁移的规则标号,通过最小不动点计算构建该 CTS 的过程如图 2 所示.

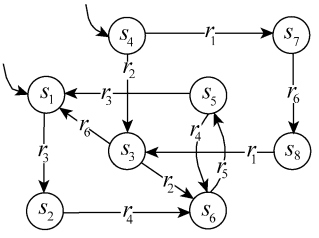
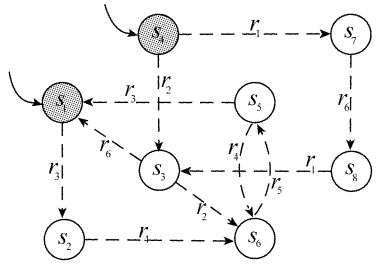
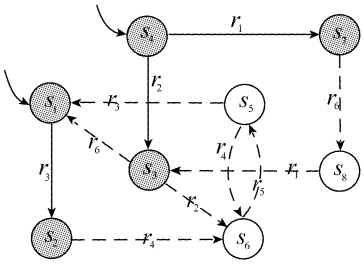


图 1 CTS 实例

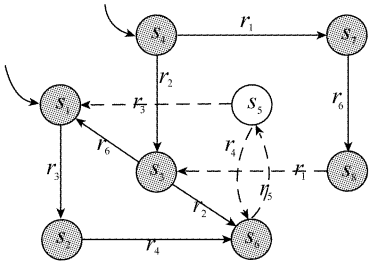
Fig. 1 An example of CTS



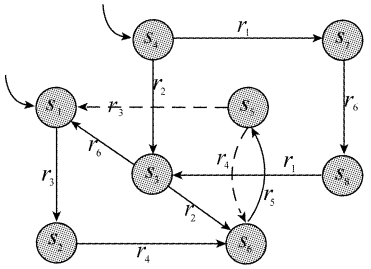
(a)第1次迭代结果



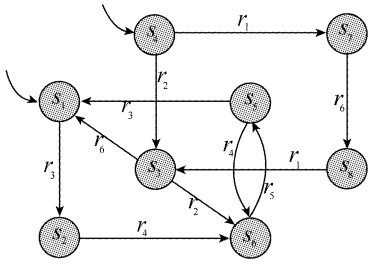
(b)第2次迭代结果



(c)第3次迭代结果



(d)第4次迭代结果



(e)第5次迭代结果

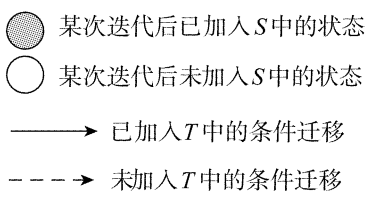


图 2　计算最小不动点求解 CTS 过程

Fig. 2　Solving process of CTS by small fix point computation

第一次迭代时输入为 $\varnothing$,计算结果将集合 $\{s_1,s_4\}$ 加入状态集合 S,由于没有进行规则匹配,所以没有条件迁移关系生成,如图 2(a)所示;第二次迭代时,由于状态 s_1 触发 $r_3.c$,经过 $r_3.a$ 迁移到状态 s_3、s_4 分别在规则 r_1、r_2 作用下迁移到状态 s_3、s_7,此次迭代结果将状态集合 $\{s_2,s_3,s_7\}$ 加入 S,把条件迁移 $(s_1,r_3.c,r_3.a,s_2)$、$(s_4,r_1.c,r_1.a,s_7)$、$(s_4,r_2.c,r_2.a.s_3)$ 加入 T,迭代结果如图 2(b)所示;第 3 次迭代将 $\{s_6,s_8\}$ 加入 S,条件迁移 $(s_2,r_4.c,r_4.a,s_6)$、$s_3,r_2.c,r_2.a,s_6)$、$(s_7,r_6.c,r_6.a,s_8)$ 加入 T,迭代结果如图 2(c)所示;第 4 次迭代把状态 s_5 加入 S,条件迁移 $(s_6,r_5.c,r_5.a,s_5)$、$(s_8,r_1.c,r_1.a,s_3)$ 加入 T,结果如图2(d)所示;第 5 次迭代把条件迁移 $(s_5,r_3.c,r_3.a,s_1)$、$(s_5,r_4.c,r_4.a,s_6)$ 加入 T,结果如图 2(e)所示,由于没有新状态生成,第 5 次迭代后计算过程结束.

下面给出通过最小不动点计算构造 CTS 的实现算法,其中 S_0 为初始状态集合,S 为状态集合,N 为新产生的状态集合,A 为动作集合,T 条件迁移关系集合.

算法:

1)初始化操作,将 S_0 加入状态集合 S,同时把 S_0 加入新状态集合 N,动作集合 A 和条件迁移集合 T 初始为空;

2)从集合 N 选择一个新产生的状态 s,并从 N 中把 s 删除;

3)从规则集 R 中取出一条未和状态 s 匹配过的规则 r,判断 s 是否满足 $r.c$,如果不满足转 7);

4)判断 $r.a(s)$ 是否已经在状态集合 S 中,如果不存在,否则转 6);

5)将 s 加入状态集合 S 和新状态集合 N;

6)将 $(s,r.c,r,a,r.a(s))$ 加入条件迁移集合 T,将 $r.a$ 加入动作集合 A;

7)如果 R 中有未和状态 s 匹配过的规则,转 3);

8)如果新状态集合 N 为空,返回状态集合 S、条件迁移集合 T 和动作集合 A,否则转 2).

4　DMM 算法分析

4.1　正确性与可终止性

定理 3 保证了算法的可终止性,求解最小不动点的计算过程最多在 n 次迭代后结束,n 为 CTS 状态集合 S 的规模,即计算 $G^n(\varnothing)$ 的过程最多在 n 次

迭代后结束;除此之外,定理3断言了函数 G 最小不动点的存在性,在此基础上,定理4进一步断言 S 就是函数 G 的最小不动点,保证了将求解状态集合 S 的过程转换为求解函数 G 的最小不动点的过程的正确性.

4.2 建模时间开销

SMM需要首先确定状态集合,然后用任意两个状态的组合匹配规则集以生成迁移关系,其建模算法时间复杂度为 $O(N^2 \times K)$,而DMM算法通过动态扩展生成系统状态和条件迁移,可保证每个生成的状态只遍历一次规则集,其时间复杂度为 $O(N \times K)$,其中 N 为系统状态规模,K 为知识库中规则规模.

4.3 错误诊断时间开销

设模型检测器返回的错误路径为 π,则 π 是一个有限状态序列 $s_0 s_1 \cdots s_n$,其中 $n \geqslant 0$,π 的长度为 n,记为 $|\pi| = n$. 使用SMM方法构造的迁移系统从错误路径 π 计算规则推理路径的过程为:由 n 个迁移关系 $s_0 \times s_1, s_1 \times s_2, \cdots, s_{n-1} \times s_n$ 分别匹配规则集,计算开销为 $O(|\pi| \times K)$;而从CTS出发可以直接由迁移关系定位错误,其计算开销为 $O(|\pi|)$.

5 实验结果与分析

本文在相同环境下对SMM与DMM的建模时间性能与错误诊断时间性能进行了对比实验. 程序运行环境为Pentim(R) Dual-Core CPU 2.50GHz,2.00GB DDR2 RAM,操作系统为Windows 7 7100;在Netbeans集成开发环境下实现SMM算法与DMM算法;实验数据为社保审计知识库. 系统模型及数据来源均存放在数据库中,数据库版本为DB2 9.1,运行环境为IBM x3950服务器. 实验第一部分在不同规模的规则集上分别用SMM算法与DMM算法进行建模. 实验结果如表1所示,其中RN为规则数量,AL为算法名称,SN为状态数量,TN为迁移数量,CN为比较次数,BT为建模时间.

从表1中数据可以看出,在相同情况下DMM算法较SMM算法的建模效率有较大的提高. 除此之外,在规则集规模相同的情况下,DMM构造的CTS规模比SMM算法构建的迁移系统略小,这是由于DMM算法在构建CTS过程中按需动态扩展,可以保证不引入冗余状态.

实验第二部分在第一部分基础上测试在2种算法构造的迁移系统上进行错误诊断的效率,实验结果如表2所示,其中RN与AL含义同表1,其余4列分别为 $|\pi|$ 取不同值时的错误定位时间(ms). 从表2可以看出,在SMM算法构建的迁移系统上进行错误诊断,其错误定位时间与规则规模与错误路径长度成正比,而在DMM算法构造的CTS上,错误定位时间仅与错误路径长度有关.

表1 SMM与DMM建模性能

Table 1 Modeling performance of SMM and DMM

RN	AL	SN	TN	CN/10^3	BT/s
13	SMM	33	85	14.15	4.193
22	SMM	81	299	144.34	31.097
30	SMM	162	874	787.32	154.927
44	SMM	382	3 323	6 420.65	1 196.200
51	SMM	693	9 355	24 492.69	4 505.320
13	DMM	33	85	2.81	1.736
22	DMM	76	293	22.27	8.191
30	DMM	152	839	127.53	34.609
44	DMM	348	3 127	1 088.20	283.163
51	DMM	613	8 628	5 289.06	1 064.375

表2 STS与CTS错误定位性能

Table 2 Error locating performance of STS and CTS

RN	AL	$\|\pi\|=20$	$\|\pi\|=30$	$\|\pi\|=40$	$\|\pi\|=50$
13	STS	46.312	69.468	92.625	115.781
22	STS	78.375	117.562	156.750	195.937
30	STS	106.875	160.312	213.750	267.187
44	STS	156.750	235.125	313.500	391.875
51	STS	181.687	272.531	363.375	454.218
13	CTS	3.557 87	5.336	7.115	8.894
22	CTS	3.566	5.350	7.133	8.917
30	CTS	1.799	3.598	5.397	7.196
44	CTS	3.490	5.235	6.981	8.726
51	CTS	3.519	5.279	7.038	8.798

6 结 论

本文提出的DMM建模方法从初始状态出发,由规则生成新状态和迁移关系的建模过程转换为求解最小不动点的计算过程. 得出结论如下:

1)DMM方法不仅避免了冗余状态的引入,而且提高了建模效率,算法的正确性和可终止性也得到了证明;

2)由规则条件出发构建的CTS同静态建模方

法构建的系统模型相比,融入了触发状态迁移的过程信息,能够解决静态建模方法造成的信息丢失问题,并且减少了错误诊断所需的计算开销,提高了错误诊断的效率;

这种针对产生式知识库的动态建模方法具有一定的通用性,能够为模型检测技术在验证产生式知识库中的应用提供有力的支持.

参考文献:

[1]CRAGUN B J, STEUDEL H J. A decision-table-based processor for checking completeness and consistency in rule-based expert systems[J]. International Journal of Man-Machine Studies, 1987, 26(5): 633-648.

[2]RAMASWAMY M, SARKAR S, YE-SHO C. Using directed hypergraphs to verify rule-based expert systems [J]. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(2): 221-237.

[3] NAZARETH D L. Investigating the applicability of Petri nets for rule-based system verification[J]. IEEE Transactions on Knowledge and Data Engineering, 1993, 5(3): 402-415.

[4]YANG S J H, LEE A S, CHU W C, HONGJI Y. Rule base verification using Petri nets[C]// Computer Software and Applications Conference. Vienna, Austria, 1998: 476-481.

[5] BOSTAN-KORPEOGLU B, YAZICI A. Using fuzzy Petri nets for static analysis of rule-bases[C]//19th International Symposium on Computer and Information Sciences (ISCIS 2004). Kemer Antalya, Turkey, 2004: 72-81.

[6]BEFLEFEUILLE S, CAPUS L, TOURIGNY N, WELCOMME J B. An approach using a knowledge network to verify consistency of rule bases[C]//7th World Multiconference on Systemics, Cybernetics and Informatics. Orlando, 2003: 63-67.

[7]CHUNG-WEI Y, CHIH-PING C. Molecular verification of rule-based systems based on DNA computation[J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(7): 965-975.

[8]阎礼祥,覃征. 自适应推理的知识求精[J]. 浙江大学学报(理学版),2004, 31(1):48-50.
YAN Lixiang, QIN Zheng. Knowledge refinement of self-adaptive reasoning[J]. Journal of Zhejiang University: Science Edition, 2004, 31(1): 48-50.

[9] HAMLET D, MARVIN Z. Software quality, software process, and software testing[J]. Advances in Computers, 1995, 41: 191-229.

[10]FRANKL P G, WEYUKER E J. Testing software to detect and reduce risk[J]. Journal of Systems and Software, 2000, 53(3): 275-286.

[11]CLARKE E M, PELED D. Model checking[M]. Cambridge: MIT Press, 1999:26-30.

[12]HUI L, JINGLEI Z, RUZHAN L. Model checking a rule-based parser [C]//Natural Language Processing and Knowledge Engineering. Beijing, 2007: 221-228.

[13]HUI L, JINGLEI Z, RUZHAN L. Toward the formal verification of a unification system[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(4): 399-408.

[14] XU Desheng, XIA Kejian, ZHANG Dezheng, ZHANG Huangsheng. Model checking the inconsistency and circularity in rule-based expert systems[J]. International Journal of Computer and Information Science, 2009, 2(1): 12-17.

[15]ANDRZEJ G, JAMES D. Fixed point theory[M]. New York: Springer-Verlag, 2003: 16-58.

[16]KNASTER B. Un théorème sur les fonctions d'ensembles [J]. Pacific Journal of Mathematics, 1928, 6: 133-134.

[17]TARSKI. A lattice-theoretical fixed point theorem and its applications[J]. Pacific Journal of Mathematics , 1955, 5(2): 285-309.

[责任编辑:王亚秋]