基于 ASP 的 CSP 模型验证性质反例生成技术研究*

王雪松[a], 赵岭忠[b], 张 超[b]
(桂林电子科技大学 a. 电子工程与自动化学院; b. 广西可信软件重点实验室, 广西 桂林 541004)

摘 要: 为了解决当前通信顺序进程(CSP)模型检测不支持在验证工具的一次运行中验证多个性质的问题,建立了基于 ASP 的 CSP 并发模型验证框架。主要研究在该框架下当待验证的系统性质不满足时生成相应性质反例的技术。把 ASP 程序调试中的 ASP 程序支撑原因分析技术应用于该问题的研究,提出了相应的反例生成算法,实例表明了该算法的正确性。

关键词: 通信顺序进程; 回答集编程; 支撑原因
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2013)01-0052-04
doi:10.3969/j.issn.1001-3695.2013.01.012

Counterexample generation in ASP-based CSP model verification

WANG Xue-song[a], ZHAO Ling-zhong[b], ZHANG Chao[b]
(*a. School of Electronic Engineering & Automation, b. Guangxi Provincial Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin Guangxi* 541004, *China*)

Abstract: This paper proposed an ASP based framework for verifying concurrent model described by CSP to solve the problem of verifying multiple properties in one run of a model checker. It mainly discussed the problem of generating counterexamples while the verified property was not satisfied in this framework. The technique of justification of ASP program, which was usually used in the debugging of ASP programs, applied to this study and proposed an algorithm for generating property counterexamples. The effectiveness of the algorithm is shown by examples.

Key words: communicating sequential processes(CSP); answer set program(ASP); justification

0 引言

通信顺序进程(CSP)是一种具有严格数学理论支撑的描述并发进程之间相互作用模式的进程代数语言,已经在网络协议建模、并发系统验证、特别是计算机安全和硬件设计方面得到了广泛应用[1]。CSP 并发进程的验证通常采用模型检测技术。牛津大学计算机计算实验室开发了 CSP 进程模型检测工具 FDR[2]。FDR 验证系统是否满足某性质的基本思想是:利用状态机判断是否存在一个符合该性质的精化迁移系统。Garavel 等人[3]注意到 CSP 进程模型检测中高级语言模型和低层实现模型设计目标的差异性,以及由此造成的直接把高级语言模型转换为低层模型的困难性和复杂性,提出在高级语言与实现模型之间增加必要过渡的中介模型。

目前,并发系统模型检测技术还存在以下问题,即主流的并发系统模型检测算法(如基于 Kripke 结构和 Petri 网的模型检测[4,5])不支持在验证工具的一次运行中验证多个性质,该特点直接限制了系统性质验证效率的提高:a)多次运行验证工具会造成大量的上下文切换的开销;b)由于即使待验证的性质式存在相同的子公式,也必须分开多次处理,从而造成多次处理同一子公式产生的重复计算开销。以上问题在现有的主流模型检测框架(如基于自动机的模型检测、基于图搜索和不动点计算的模型检测)内很难克服。

解决以上问题的出路之一就是建立并发系统验证模型的统一框架。为此,课题组在前期工作中借助于具有声明特征的知识表示和推理语言 ASP[6] 对并发系统验证模型框架进行了统一,提出了一种基于 ASP 的 CSP 模型验证框架。在基于 ASP 的验证框架下,待验证系统性质被统一转换为 ASP 规则,且每个性质都有与之对应的谓词。通过判断与某个性质对应的原子是否属于程序的回答集,可确定该性质是否满足,并实现了在验证工具的一次运行中验证多条系统性质的目标。与该工作类似,Angelis 等人[7]提出了一种利用 ASP 进行并发进程合成的技术,利用该技术可以从多个并发进程生成一个满足所有行为特性和结构特性的新进程。与此不同,本课题组前期工作的重点则在于对多个并发进程是否满足用户期待的性质进行验证。

在前期工作的基础上,本文要解决的问题是:当待验证的性质在模型中不满足时,给出造成性质不满足的反例。ASP 程序的支撑原因(justification)分析技术是一个基于计算过程中产生的回答集,用来生成某原子成立或不成立的依据的技术[8]。本文的主要贡献即引入该技术来解决在待验证的系统

收稿日期: 2012-06-26; **修回日期**: 2012-08-14 **基金项目**: 国家自然科学基金资助项目(61262008,61063002);广西科学基金资助项目(2011GXNSFA018166, 2011GXNSFA018164);广西可信软件重点实验室基金资助项目(kx201113)

作者简介: 王雪松(1981-),女,江苏徐州人,讲师,硕士,主要研究方向为人工智能、形式化方法;赵岭忠(1977-),男(通信作者),河南社旗人,教授,博士,主要研究方向为人工智能、形式化技术(zhaolingzhong163@163.com);张超(1986-),男,硕士,主要研究方向为并发系统验证.

性质不满足时生成反例的问题。

1　基础知识

1.1　ASP

ASP 是一种声明式程序设计框架,其基础是逻辑程序的回答集语义或稳定模型语义。设 A 是一个原子,一个文字可以是 A 也可以是 $\sim A$,其中 A 表示正文字, $\sim A$ 表示负文字。ASP 程序的规则 r 形如 $L_1 \vee \cdots \vee L_k : - L_{k+1},\cdots,L_m,\text{not } L_{m+1},\cdots,\text{not } L_n$,其中 $n \geqslant m \geqslant k \geqslant 0$。每一个 L_i 是一个文字,not 表示失败即否定(negation as failure)。将 $\text{head}(r) = \{L_1,\cdots,L_k\}$ 称为规则头部;$\text{body}^+(r) = \{L_{k+1},\cdots,L_m\}$ 称为规则体的正文字,而 $\text{body}^-(r) = \{L_{m+1},\cdots,L_n\}$ 称为规则体的负文字。如果一个规则 r 没有头部,称之为约束。如果 $\text{body}^+(r)$ 和 $\text{body}^-(r)$ 均为空,则称 r 为事实。如果一个程序的所有规则中的 L_i 均为原子且满足 $k=0$ 且 $m=n$,则称这个程序为限定回答集程序。如果一个程序的每一个规则中的 L_i 均为原子且满足 $k=0$,则这个程序为正规回答集程序。设 P 为不含约束规则的正规逻辑程序,$S \subseteq \text{Lit}(P)$, P 基于 S 的 Gelfond-Lifschitz 规约($GL(P, S)$)可通过如下步骤得到:

a)对 P 中的任意规则 r,如果其规则体中含有公式 not a,且 $a \in S$,则删除此规则[7]。

b)删除剩下规则中所有形如 not a 的公式。

已知一个逻辑程序 $P, S \subseteq \text{Lit}(P)$,$GL(P, S)$ 的最小模型为 X,如果 $X = S$,则 X 称为回答集(answer set)。从回答集的定义可以看出,如果 X 为程序 P 的一个回答集,则 X 满足下面两个条件:

a)对于程序中任一规则 r,如 $\text{pos}(r) \subseteq X$ 且 $\text{neg}(r) \cap X = \varnothing$,则 $\text{head}(r) \in X$。

b)如果 X 中存在一对相反的文字,则 $X = \text{Lit}(P)$。

例如,已知正规逻辑程序:

$$P = \{a : - b,\text{not } c.\quad e : - a,\text{not } d.\quad c : - b,\text{not } a.\quad b : - .\}$$

该程序有两个回答集:{b,a,e}, {b,c}。

基于 ASP 的 CSP 是一种强大的并发系统描述工具,可以方便地描述顺序、循环、选择、并发等结构的进程。LTL 和 CTL 是广泛使用的并发系统性质描述语言。基于 ASP 的 CSP 并发模型验证技术首先将 CSP 程序转换为 ASP 程序,然后将 CSP 进程并发规则和以 LTL/CTL 式表示的待验证性质转换为 ASP 规则,最后通过计算 ASP 程序的回答集来判断 CSP 并发模型是否满足系统性质。基于 ASP 对 CSP 并发进程进行验证的基本流程如下:

a)将 CSP 程序转换为 ASP 程序,记做 C_0。

b)将 CSP 并发进程规则转换为 ASP 规则,记做 Π。

c)结合一种转换算法,将 $C_0 \cup \Pi$ 的回答集中相同步数下并发事件状态转换成析取规则 C_1。

d)将 LTL/CTL 式描述的待验证并发程序性质用 ASP 表示,记做 P。

e)计算 $C_1 \cup P \cup F$ 的回答集,F 是路径判断规则。根据求得的回答集结果确定验证性质是否正确。

按照以上方法就可对 CSP 并发系统进行验证。

1.2　支撑原因(justification)[8,9] 分析

Justification 图是解释一个原子在回答集中真值情况的方法。

定义 1　关于限定程序 P 和原子集合 M 中原子 A 的真值分配 $\tau(M,A)$ 定义为:如果 $A \in M$,则 $\tau(M,A) = \text{true}$;如果 $A \notin M$,则 $\tau(M,A) = \text{false}$。

定义 2　令 P 是一个 ASP 程序,M_P 是 P 的最小的 Herbrand 模型。用符号 $\zeta P(A)$ 表示原子 A 的局部一致性解释(LCE),它是需要满足以下条件原子集合的集合:

a)如果 $\tau(M_P,A) = \text{true}$,则 $\zeta P(A) = \{\text{body}(r) \mid r \in P \wedge A = \text{head}(r) \wedge \forall B \in \text{body}(r).\ \tau(M_P,B) = \text{true}\}$。

b)如果 $\tau(M_P,A) = \text{false}$,则 $\zeta P(A) = \{\alpha_1,\cdots,\alpha_m\}$,其中每一个 α_i 都是最小集合,即满足以下条件:(a) $\forall B \in \alpha_i.\ \tau(M_P, B) = \text{false}$;(b)每一个规则 $r \in P$ 且 $\text{head}(r) = A$,存在 $\exists B \in \text{body}(r).\ B \in \alpha_i$。

定义 3　限定 ASP 程序 P 中原子 A 的 justification 图($J_P(A)$)是一个有向图$\langle V,E\rangle$,其中 $\{A\} \subseteq V \subseteq BP \cup \{T, \perp\}$,且满足:

a)在 $J_P(A)$ 中每一个在 $V \backslash \{A\}$ 中的节点都是从 A 可达的。

b)如果 $\tau(M_P,B) = \text{true} B \in V$,则

(a)$\varnothing \in \zeta P(B)$ 当且仅当$\langle B,T\rangle \in E$ 且 $T \in V$;

(b)有一个唯一的 $\alpha \in \zeta P(B)$ 使得对于每一个 $C \in \alpha$,有 $\alpha \in V$,$\langle B,C\rangle \in E, C \neq B$ 且$\langle C,B\rangle \notin E$;

(c)没有其他从 B 出发的出边。

c)如果 $\tau(M_P,B) = \text{false}$ 且 $B \in V$,则

(a)$\varnothing \in \zeta P(B)$ 当且仅当$\langle B, \perp\rangle \in E$ 且 $\perp \in V$;

(b)有一个唯一的 $\alpha \in \zeta P(B)$ 使得 $\forall C \in \alpha$,有 $\alpha \in V \wedge \langle B, C\rangle \in E$;

(c)没有其他从 B 出发的出边。

正规回答集程序的 justification 图定义如下:

定义 4　已知 ASP 程序 P,M 是它的任意一个回答集,l 是程序语言中的一个文字,如果 l 是原子 A,则 $\tau'(M,l) = \tau(M, A)$;如果 l 是文字 not A,则 $\tau'(M, l) = \neg\ \tau(M,A)$。

定义 5　已知原子 A 和程序 P 的一个回答集 M,A 的关于 M 的 LCE 用 $\zeta P(M,A)$ 表示,它是一个满足以下条件的文字集合:

a)如果 $\tau'(M, A) = \text{true}$,则 $\zeta P(M,A)$ 为$\{\text{body}(r) \mid r \in P \wedge \text{head}(r) = A \wedge \forall B \in \text{pos_body}(r).\ \tau'(M,B) = \text{true} \wedge \forall B \in \text{neg_body}(r).\ \tau'(M,B) = \text{false}\}$。

b)如果 $\tau'(M,A) = \text{false}$,则 $\zeta P(M,A) = \{\alpha_1,\cdots,\alpha_k\}$,其中每个 α_i 是满足以下条件的最小集合:

(a) $\forall B \in \alpha$,有 $\tau'(M,B) = \text{false}$;

(b)对于每个头部原子为 A 的规则 $r \in P$,有 $\exists B \in \alpha_i \cap \text{body}(r)$。

定义 6　Justification 原子 A 关于 ASP 程序 P 和回答集 M 的 justification 用 $J_P(M,A)$ 表示,它是一个带标签的有向图$\langle V, E\rangle$,其中$\{A\} \subseteq V \subseteq BP \cup \{\text{T}, \perp\}$,那么:

a)每一个在 $V \backslash \{A\}$ 中的节点都是从 A 可达的。

b)如果 $\tau'(M, B) = \text{true}$ 且 $B \in V$,则

(a)$\varnothing \in \zeta P(M,B)$ 当且仅当$\langle B,T, +\rangle \in E$ 且 $T \in V$;

(b)有唯一 $\alpha \in \zeta P(M,B)$ 满足,且对于每个 $C \in \alpha$,且有:

①如果 C 是原子,则 $C \in V$,$\langle B,C, +\rangle \in E, C \neq B$ 且$\langle C, B, +\rangle \in E$;

②如果 C 是文字 not D,则 $D \in V$,$\langle B,D, -\rangle \in E$。

(c)没有其他的从 B 出发的出边。

c)如果 $\tau'(M, B) = \text{false}$ 且 $B \in V$,则

(a)$\varnothing \in \zeta P(M,B)$ 当且仅当$\langle B, \perp, +\rangle \in E$ 且 $\perp \in V$;

(b)有唯一 $\alpha \in \zeta P(M,B)$ 满足,且对于每个 $C \in \alpha$,有:

①如果 C 是一个原子,则 $C \in V$, $\langle B,C, +\rangle \in E$;

②如果 C 是一个文字 not D,则 $D \in V$,$\langle B,D, -\rangle \in E$。

例如,一个 ASP 程序 P 为 $\{r_1: d:-a, \text{not } b.\ r_2: a:-f, \text{not } c.\ r_3: c:-g, \text{not } a.\ r_4: g.\ r_5: f:-g.\}$。该程序有两个回答集 $M_1=\{g,d,a,f\}$ 和 $M_2=\{g,f,c\}$。对于 M_1 来说,c 关于 M_1 的 justification 图如图 1 所示。

由图 1 可知,在 M_1 中 c 为 false 的原因是 a 为 true。

可见,justification 图可以很直观地表达出原子在或不在某个回答集中的原因。利用 Justification 图的这一优点,在 CSP 并发系统的 ASP 程序中,可以方便地知道回答集中进程并发的原因。下面将在支撑原因分析技术的基础上给出 CSP 系统反例生成技术。

2 系统模型性质反例生成算法

系统模型反例生成技术要求:对于一个回答集 M,找出造成性质 f 不在回答集中的原因。现有支撑原因分析技术还不能直接用于解决这个问题。原因是支撑原因分析技术针对不含有变量的 ASP 程序,而与 CSP 并发进程对应的 ASP 程序则含有变量。为此,对现有算法进行了适当的修改,给出基于 ASP 的 CSP 并发系统模型性质反例生成算法,步骤如下:

a)将 ASP 表示的 LTL/CTL 式规则用 r_i 表示,用 M 表示求得的 ASP 程序回答集;需要验证的原子集合用 A 表示;已验证的原子集合用 C 表示(集合 B 的初始值为空集);已遍历的规则集合用 R 表示(初始值为空);P 表示 ASP 表示的验证式的规则集合。

b)若 $A=C$,则算法结束;否则,转至步骤 c)。

c)对于 A 中的原子 a,且 $a\notin C$,如果 a 为 ture,转至步骤 d);如果 a 为 false,转至步骤 e)。

d)(a)在 P 中搜索头部为 a 的规则存入集合 N,依次遍历 N 中的规则 r_i。令 $R=R\cup\{r_i\}$,$B_i=\text{Get\_true\_body}(r_i,M)$。若 $R=N$,转至(d);否则,转至(b)。其中,函数 Get_true_body(r_i, M)返回规则 r_i 体部属于 M 的原子。Get_true_body(r_i, M)算法如下:

①令 $S=\varnothing$,T 为规则 r_i 体部集合;

②对于每个原子 $a_i\in T$,如果 $a_i\in M$,则 $S=S\cup\{a_i\}$;

③重复②,直至遍历完所有的原子,返回 S。

(b)对于规则 r_i 体部的原子 $a_i\in B_i$,如果 a_i 中不含 not,则 a_i 为 true;否则 a_i 为 false。

(c)如果 $a_i\notin A$ 且未画出且画出规则 r_i 和原子 a_i,然后令 $A=A\cup\{a_i\}$,重复(b)直至规则 r_i 中所有属于 B_i 的原子遍历完转至(a)。

(d)$C=C\cup\{a\}$,$R=\varnothing$。返回步骤 b)。

e)(a)对 P 中搜索头部为 a 的规则存入集合 N,依次遍历 N 中的规则 r_i。令 $R=R\cup\{r_i\}$,$B_i=\text{Get\_true\_body}(r_i,M)$。若 $R=N$,转至(d);否则,转至(b)。其中,函数 Get_false_body (r_i, M)返回规则 r_i 体部不属于 M 的原子。Get_false_body(r_i, M)算法如下:

①令 $S=\varnothing$,T 为规则 r_i 体部集合;

②对于每个原子 $a_i\in T$,如果 $a_i\notin M$,则 $S=S\cup\{a_i\}$;

③重复②直至遍历完所有的原子,返回 S。

(b)对于规则 r_i 体部的原子 $a_i\in B_i$,如果 a_i 中不含“not”,则 a_i 为 flase;否则 a_i 为 true。

(c)如果 $a_i\notin A$ 且未画出且画出规则 r_i 和原子 a_i,然后,令 $A=A\cup\{a_i\}$,重复(b)直至规则 r_i 中所有属于 B_i 的原子遍历完转至(a)。

(d)$C=C\cup\{a\}$,$R=\varnothing$。返回步骤 b)。

根据上述算法,以系统验证过程中求得的回答集 M、ASP 规则集 P 表示的 LTL 式,以及经验证不满足的性质 f 为输入,即产生相应的性质反例。下面以哲学家进餐问题为例说明以上算法的应用。

3 系统模型性质反例生成的实验分析

哲学家进餐问题:几个哲学家在思考问题,当他们感觉饿的时候会到桌子旁准备进餐。假设有五位哲学家命名为 $PH_0\sim PH_4$。当他们准备进餐时,要先坐下来,然后先拿起左边餐叉,再拿起右边的餐叉。当双手分别有餐叉时就可以进餐了。进完餐之后,离开座位继续去思考问题。图 2 为具体的哲学家进餐座位示意图。

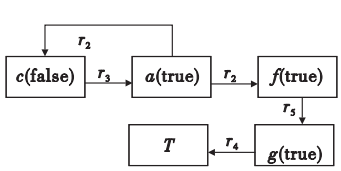

图1 原子c关于M_1的justification图

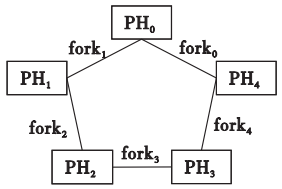

图2 哲学家进餐座位示意图

通常哲学家进餐问题的求解方案需要满足一个性质:一把叉子不能同时被两个哲学家拿起。每一个哲学家的行为可用 CSP 规则描述为 $PH_i=sit_i\rightarrow pick\_fork_i\rightarrow pick\_fork_i\oplus 1\rightarrow eat_i\rightarrow down\_fork_i\rightarrow down\_fork_i\oplus 1\rightarrow up_i\rightarrow Ph_i$,其中“⊕”是模 5 运算。以并发 5 步为例,按照基于 ASP 的 CSP 并发系统验证流程:

a)将并发程序 CSP 模型转换为 ASP 规则:

以邻座(PH_1、PH_2)为例,ASP 描述的 CSP 规则 C_0 为:

{first(sit_1, ph_1), pre(sit_1, pick_fork$_1$, ph_1), pre(pick_fork$_1$, pick_fork$_2$, ph_1), pre(pick_fork$_2$, eat_1, ph_1), pre(eat_1, down_fork$_1$, ph_1), pre(down_fork$_1$, down_fork$_2$, ph_1), pre(down_fork$_2$, up_1, ph_1), last(up_1, ph_1), first(sit_2, ph_2), pre(sit_2, pick_fork$_2$, ph_2), pre(pick_fork$_2$, pick_fork$_3$, ph_2), pre(pick_fork$_3$, eat_2, ph_2), pre(eat_2, down_fork$_2$, ph_2), pre(down_fork$_2$, down_fork$_3$, ph_2), pre(down_fork$_3$, up_2, ph_2), last(up_2, ph_2), aux(sit_1, $sit_{2,0}$).}

b)将 $C_0\cup\Pi$ 的回答集(Π 是 CSP 并发进行规则转换的 ASP 程序)中相同步数下并发事件状态转换成 ASP 的析取规则 C_1。方法是:先将回答集中的形如 aux(X,Y,I) 的原子依次存入一个链表,然后在链表中搜索 I(即并发步骤)相同的原子,以连接符号“v”输出。

c)将 LTL 式转换为 ASP 规则。

性质用 LTL 式表示为 $G\neg$ (pick_fork$_i\wedge$ pick_fork$_j\wedge i=j$),转换为 ASP 规则集 P。

$P=\{f(I):-\text{aux}(X,Y,I), X=Y, X=\text{pick\_fork}_2.$

non_f:- $f(I)$, $I>0$, $I<=5$. f:- not non_f. f?}

d)结合路径判断规则 L:

{:- aux(X,Y,I), aux(M,N,J), $J=I+1$, $X!=M$, $Y!=N$, $X!=Y$.}

验证性质。在 ASP 模型的谨慎推理(cautious reasonning)方式下,对 $C_1\cup P\cup L$ 进行求解,求得的回答集:

$M=\{f$ is cautionsly false, evidenced by {aux(sit_1, $sit_{2,0}$), aux(pick_fork$_1$, $sit_{2,1}$), aux(pick_fork$_2$, $sit_{2,2}$), aux(pick_fork$_2$, pick_fork$_{2,3}$), aux(pick_fork$_3$, $eat_{1,4}$), aux(pick_fork$_3$, down_fork$_{1,5}$), $f(3)$, non_f}}。

e)回答集的结果表明:f 不在回答集中,即一把叉子不能被两个哲学家拿起的性质不满足。

以 $P\cup M$ 以及不满足的性质 f 为输入,则性质 f 不在回答集中的反例生成图如图 3 所示。

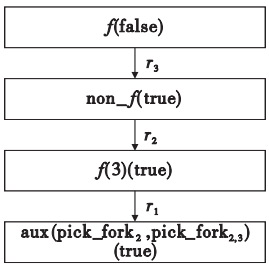

图3　性质 f 关于回答集 M 的反例

由图 3 可知,f 为 false 是因为在 P 的规则 3 中,non_f 为 ture(non_f 在回答集中),依次类推,最终找出 f 不在回答集中的原因是由于原子 aux(pick_fork$_2$,pick_fork$_{2,3}$)在回答集中,即在并发到第三步时,哲学家 1 和 2 同时拿起了 fork$_2$,使得性质不满足。由此,反例找出。

通过上述的例子可以看出,利用图形对反例生成技术研究的一些优点是:a)很直观地给用户指出某原子产生的原因或者不存在的原因;b)本文方法是在原先 justification 图调试方法上的创新,解决了原先 justification 图对含有变量的 ASP 程序无法生成反例的问题,从而完善了基于 ASP 的 CSP 并发验证框架。

4　结束语

本文在基于 ASP 的 CSP 并发系统验证框架下进行了 CSP 系统性质反例生成技术研究。把 ASP 程序支撑原因分析技术引入该框架,给出了一种生成系统性质反例的算法。实验分析结果表明,该方法可以有效地给出性质反例。未来将根据生成的反例对 ASP 程序进行调试,并评估调试过程对已验证性质的影响。

参考文献:

[1] HOARE C A R. Communicating sequential processes[EB/OL]. (2004). http://www.usingcsp.com/cspbooks.

[2] Formal Systems (Europe) Ltd. Failures-Divergence refinement: FDR2 user manual[EB/OL]. (1999). http://www.formal.demon.co.uk.

[3] GARAVEL H, LANG F. NTIF: a general symbolic model for communicating sequential processes with data [C]//Proc of Formal Techniques for Network and Distributed Systems. 2002:276-291.

[4] CLARKE E M, GRUMBERG O, PELED D A. Model checking[M]. Cambridge: MIT Press, 2000.

[5] BOUCHENEB H, GARDEY G, ROUX O H. TCTL model checking of time Petri nets[J]. Journal of Logic and Computation, 2009, 19(6):1509-1540.

[6] GELFOND M, LIFSCHITZ V. The stable model semantics for logic programming[C]//Proc of the 5th International Conference on Logic Programming. 1988:1070-1080.

[7] De ANGELIS E, PETTOROSSI A, PROIETTI M. Synthesizing concurrent programs using answer set programming[C]//Proc of the 26th Italian Conference on Computational Logic. 2011:85-98.

[8] EL-KHATIB O, PONTELLI E, SON T C. Justification and debugging of answer set programs in ASP[C]//Proc of the 6th International Symposium on Automated Analysis-Driven Debugging. 2005:49-58.

[9] PEMMASANI G, GUO Hai-feng, DONG Yi-fei, *et al*. Online justification for tabled logic programs[C]//Proc of the 7th International Symposium on Functional and Logic Programming. 2004:24-38.

(上接第 51 页)

〈Through rdf:resource = "#黄河风景名胜区"/〉
〈Through rdf:resource = "#花园口旅游风景区"/〉
〈Through rdf:resource = "#相国寺"/〉
〈Through rdf:resource = "#龙亭"/〉
〈FeatureID rdf:datatype = "http://www.w3.org/2001/XMLSchema-JHJint"〉
201001〈/FeatureID〉
〈/Railroad〉

d)图形用户界面显示与表达

图形用户界面(图 6)部分实现了图形化分层显示空间关系本体知识库中的具体实例,设置查询条件,点击推理实例查询按钮,可以将推理结果输出并在图中高亮显示,做到了定性空间推理与图形用户界面的交互。

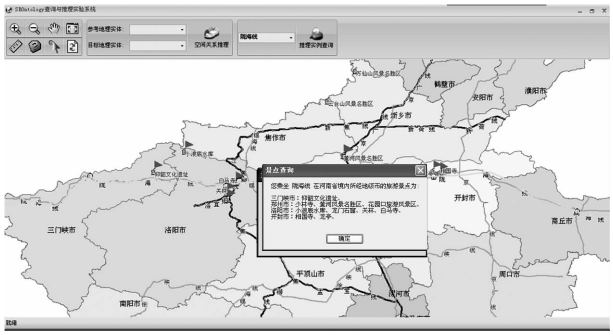

图6　图形用户界面

经过上述实验可知,基于本体的空间关系推理在一定程度上能够解决传统空间关系推理过程中的语义异质问题,并且推理的过程与人的常识性空间认知相符合。

4　结束语

当前 GIS 软件中显式表达的空间关系是有限的,构建一套有效的空间关系推理机制势在必行。空间关系推理本质上还是一种知识推理,但是因其缺乏层次概念,无法充分表现类别关系,只能应用于特定环境,不利于共享和重用。为此,本文引入本体概念,研究地理本体中空间关系的表达和推理机制,通过具体实例进行了验证,最终目的是实现地理空间信息的共享和互操作。

下一步要实现空间关系本体的自动构建,简化空间推理过程;加入时间元素,初步实现时空本体推理;解决推理结果的不确定性问题,排除一些冗余的推理结果,使空间关系本体推理更符合人类的常识空间认知。

参考文献:

[1] 曹菡. 空间关系推理的知识表示与推理机制研究[D]. 武汉:武汉大学,2002:12-13.

[2] 黄茂军. 地理本体空间特征的形式化表达机制研究[J]. 武汉大学学报:自然科学版,2005,30(4):337-340.

[3] 宋佳. 基于 GML 的时空地理本体模型构建及应用研究[J]. 地球信息科学学报,2009,11(4):442-451.

[4] 李宏伟,成毅,李勤超. 地理本体与地理信息服务[M]. 西安:西安地图出版社, 2008:20-28.

[5] 李淑霞,安敏,李宏伟,等. 常识空间认知研究与地名本体设计[J]. 测绘科学技术学报,2011,28(6):450-453.

[6] PAN J Z, STOILOS G, STAMOU G, *et al*. f-SWRL: a fuzzy extension of SWRL[C]//Lecture Notes in Computer Science, Vol 3697. Berlin: Springer, 2005:829-834.