正规逻辑程序回答集存在性研究

方毅立[1] 赵岭忠[1,2] 钱俊彦[1]
(桂林电子科技大学 广西可信软件重点实验室 桂林 541004)[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 判断逻辑程序的回答集是否存在是回答集程序设计的一个重要问题,也是NP完全问题。当前利用否定圈边数的奇偶性来判断回答集存在性的方法还具有一定的局限性,即:对于非分层逻辑程序,现有方法并不能准确判断其回答集存在性。针对该问题,提出了一种新的基于否定圈的判断方法,给出了该判断方法的算法框架,证明了算法的正确性,并以实例分析说明了方法的有效性。

关键词 回答集逻辑程序,回答集,划分集,否定圈
中图法分类号 TP18,TP301.2 **文献标识码** A

Existence of Answer Sets of Normal Logic Programs

FANG Yi-li[1] ZHAO Ling-zhong[1,2] QIAN Jun-yan[1]
(Guangxi Provincial Key Lab of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract Determining the existence of answer sets of logic programs is a key problem in answer set programming, and is also NP-complete. Current methods for determining the existence of answer sets are mainly based on the evenness/oddness of the number of edges in negative circles. The limitation of this kind of methods is that if a normal logic program is not stratified, the existence of answer sets of the program cannot be accurately determined. A novel negative circle based method was proposed to solve this problem. An algorithmic framework and the correctness of the method were presented in this paper, and its effectiveness was illustrated by example analysis.

Keywords Answer set programming, Answer set, Splitting set, Negative cycle

1 引言

回答集程序设计(ASP-Answer Set Programming)的出现是非单调推理领域的突破性成果[1],其理论基础是Gelfond和Lifschitz提出的回答集语义[2,3]。目前,它在规划、诊断、行动推理、智能机器人的规划和决策[4]及航空火箭的决策[5]等领域均已得到广泛应用。为了更有效地利用ASP,国内外众多学者对ASP的性质及应用进行了广泛的研究[1,2,11]。判断逻辑程序回答集的存在性是回答集求解器设计的一个关键技术,判断的准确性直接关系到回答集求解的效率[7],因而成为ASP研究中的一个热点问题[6,7]。国内外学者对该问题进行了深入的研究,产生了一系列代表性的研究成果。

(1) 如果一个逻辑程序的原子推理图中无偶数圈(带偶数条否定边的圈),则该程序至多有一个回答集[5];如果一个程序不包含奇数圈(带奇数条否定边的圈),则必存在回答集[1];

(2) 如果一个逻辑程序的原子推理图中恰有一个奇数圈(可能有多个偶数圈),则其回答集的存在性问题是NP完全问题[1];

(3) 如果一个逻辑程序的原子推理图中有 k 个偶数圈(可能有多个奇数圈),则其至多有 2^k 个回答集,并且所有的回答集可在 $2^{2k}O(n^{k+2})$ 时间内计算出来,其中 n 是逻辑程序中规则的条数[1]。

以上研究成果均是从逻辑程序所含否定圈边数的奇偶性方面来判断该逻辑程序回答集的存在性,下面的例子表明该方法还具有一定的局限性。

例1 程序 $P=\{a:\text{-}not\ b, not\ c. b:\text{-}not\ a, not\ c. c:\text{-}not\ a, not\ b.\}$ 为一个正规逻辑程序,其原子推理图中有2个奇数圈和3个偶数圈,它存在3个回答集,分别为:$\{a\},\{b\},\{c\}$。

显然上述结论(1)和(2)并不适用本例。而按照结论(3)则可根据偶数圈的个数来判断该程序回答集个数的上限为 $2^3=8$,但该结论并不足于判断该程序是否存在回答集。为此,本文提出一种对否定圈中的否定边进行标注的方法,由该方法可准确判断一个逻辑程序是否存在回答集。例如:根据该方

到稿日期:2011-01-29 返修日期:2011-04-28 本文受国家自然基金(60803033,61063002),广西自然基金(2011GXNSFA018166,2011GXNSFA018164),武汉大学软件工程国家重点实验室开放基金项目(SKLSE2010-08-06)和广西研究生教育创新计划资助项目(2010105950812M28)资助。

方毅立(1982-),男,硕士,主要研究方向为知识表示与推理、逻辑程序与非单调推理,E-mail:fangyl20051@gmail.com;赵岭忠(1977-),男,博士,副教授,硕士生导师,主要研究方向为逻辑程序验证测试、形式化技术。

法，可以判断例 1 中逻辑程序 P 必然存在回答集。

本文第 2 节介绍逻辑程序的基本概念，给出否定圈的定义及相关概念；第 3 节给出本文方法的主要思想；第 4 节利用与逻辑程序回答集判断无关的正规则及环的概念对逻辑程序进行约减，使约减后逻辑程序回答集的存在性更容易判定；第 5 节利用否定圈的性质把正规逻辑程序分为三大类，给出了分类算法，并对前两类逻辑程序回答集的存在性进行了分析判定；第 6 节对第三类逻辑程序否定圈进行分析，给出了否定圈回答集存在性的判断算法；第 7 节以实例说明了本文方法的有效性；最后给出结论。

2 背景知识

本节介绍逻辑程序回答集、原子推理图、程序划分及否定圈的概念，并给出否定圈相关规则集合的定义。

2.1 逻辑程序和回答集语义

正规逻辑程序是一种特殊的逻辑程序[4]，是形如

$r\equiv l_0 :\!- l_1,\cdots,l_m,not\ l_{m+1},\cdots,not\ l_n.$

的规则的集合，其中 $l_i(0\leqslant i\leqslant n)$ 是一个基本命题语言文字（文字由原子或原子的否定组成）。对任意规则 r，定义：$head(r)=\{l_0\}$，$Pos(r)=\{l_1,\cdots,l_m\}$，$Neg(r)=\{l_{m+1},\cdots,l_n\}$，$body(r)=Pos(r)\cup Neg(r)$，已知正规逻辑程序 P，$Lit(P)$ 表示 P 中出现的所有文字的集合。如果 P 中只有一条规则 r，则 $Lit(\{r\})=head(r)\cup Pos(r)\cup Neg(r)$。为了方便，$Lit(\{r\})$ 简记为 $Lit(r)$；规则 r 可表示为：$head(r) :\!- Pos(r),not\ Neg(r)$ 或者 $head(r) :\!- body(r)$。如果 $head(r)$ 为空，则称 r 为约束规则；如果 $Neg(r)$ 为空，则称 r 为正规则；如果 $Pos(r)$ 和 $Neg(r)$ 均为空，则称 r 为事实。

设 P 为不含约束规则的正规逻辑程序，$S\subseteq Lit(P)$，P 基于 S 的 Gelfond-Lifschitz 规约（$GL(P,S)$）可通过如下步骤得到[2]：

(1) 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；

(2) 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，$S\subseteq Lit(P)$，$GL(P,S)$ 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：

1)对于程序中任一规则 r，若 $Pos(r)\subseteq X$ 且 $Neg(r)\cap X=\varnothing$，则 $head(r)\in X$；

2)若 X 中存在一对相反的文字（即不一致），则 $X=Lit(P)$。

例 2　已知正规逻辑程序

$P=\{g :\!- b,not\ f,not\ h.\ f :\!- a,not\ h,not\ g.$

$h :\!- e,not\ f,not\ g.\ a :\!- b,e.\ g :\!- p,not\ j.\ e :\!- d.\ d.\ b.\}$

该程序有 3 个回答集：$\{d,b,h,a,e\}$，$\{d,b,f,a,e\}$，$\{d,b,g,a,e\}$。

定义 1（原子推理图）　已知正规逻辑程序 P，P 的原子推理图 $DG_P=(V_P,E_P)$，$V_P=\{a\mid a\in Lit(P)\}$，$E_P=NE_P\cup PE_P$，其中 $NE_P=\{\langle x,y\rangle\mid \exists r\in P.(y\in head(r)\wedge x\in Neg(r))\}$，是 P 中规则产生的所有否定边的集合，$PE_P=\{\langle x,y\rangle\mid \exists r\in P.(y\in head(r)\wedge x\in Pos(r))\}$ 是 P 中规则产生的所有正边的集合。在 DG_P 中，如果边 $e=\langle x,y\rangle$ 由规则 r 产生，则称 r 是 e 的关联规则，e 是 r 的关联边；DG_P 中节点 N 所属的所有否定圈上边的集合记作 $CirE(N)$。

例 3　已知逻辑程序 $P=\{b :\!- a,d :\!- c.\ b :\!- not\ d.\ d :\!- f.\ a :\!- f.\ f :\!-.\}$，则 P 的原子推理图如图 1 所示。

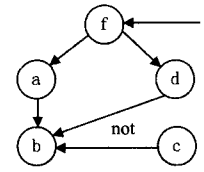

图 1　例 3 中逻辑程序 P 的原子推理图

其中，$V_P=\{a,b,c,d,f\}$，$E_P=\{\langle f,a\rangle,\langle a,b\rangle,\langle d,b\rangle,\langle b,c\rangle,\langle f,d\rangle\}$，$NE_P=\{\langle d,b\rangle\}$，$PE_P=\{\langle f,a\rangle,\langle a,b\rangle,\langle b,c\rangle,\langle f,d\rangle\}$。

2.2 程序的划分

划分集[3,10]的目的是将 ASP 程序划分为若干子程序，从而可由子程序的回答集计算原程序的回答集。由于 ASP 具有非单调性，因此划分集必须满足一定条件。

定义 2（划分集，Splitting Set）　已知逻辑程序 P，$U\subseteq Lit(P)$，对于任意 $r\in P$，如果 $head(r)\cap U\neq\varnothing$ 蕴涵 $Lit(r)\subseteq U$，则 U 称为 P 的一个划分集。令 $b_U(P)=\{r\mid r\in P$ 且 $Lit(r)\in U\}$，$b_U(P)$ 称为 P 相对于划分集 U 的底，$P\backslash b_U(P)$ 称为 P 相对于 U 的顶。

由定义可知，$\varnothing$ 和 $Lit(P)$ 是任何程序 P 的划分集。利用逻辑程序 P 基于划分集 U 的底 $b_U(P)$，可计算 P 的回答集。已知正规逻辑程序 P，原子集 U 和 X，其中 U 为逻辑程序 P 的一个划分集，定义子程序：

$$e_U(P,X)=\{r'\mid \text{存在 } r\in P,\text{使得}(Pos(r)\cap U)\subset X,Neg(r)\cap U)\cap X=\varnothing,head(r')=head(r),Pos(P')=Pos(r)\backslash U,Neg(r')=Neg(r)\backslash U\}\quad(1)$$

$$e_U(P,X)=e_U(b_U(P),X)\cup e_U(P\backslash b_U(P),X)\quad(2)$$

定义 3（Solution，解）[10]　已知正规逻辑程序 P，U 为 P 的一个划分集合。P 基于 U 的解为满足如下 3 个条件的二元组 $\langle X,Y\rangle$：

1)X 为 $b_U(P)$ 的一个回答集；

2)Y 为 $e_U(P\backslash bu(P),X)$ 的一个回答集；

3)$X\cup Y$ 满足一致性。

定理 1（Splitting Set Theorem，划分集定理）[10]　已知正规逻辑程序 P，U 为 P 的一个划分集，原子集 S 是 P 的一个回答集，当且仅当存在 P 基于 U 的一个解 $\langle X,Y\rangle$ 使得 $S=X\cup Y$。

例 4　已知正规逻辑程序

$P=\{p :\!- not\ q.\ p :\!- not\ p.\ q :\!- not\ r.\ r :\!- not\ q.\}$

显然 $U=\{q,r\}$ 为 P 的一个划分集。$b_U(P)=\{q :\!- not\ r.\ r :\!- not\ q.\}$ 的回答集为 $\{q\}$ 和 $\{r\}$；由于 $e_U(P\backslash b_U(P),\{q\})=\{p :\!- not\ p.\}$ 没有回答集，而 $e_U(P\setminus b_U(P),\{r\})=\{p :\!-.\ p :\!- not\ p.\}$ 的回答集为 $\{p\}$，根据定义 3，P 基于 U 的解为 $\langle\{r\},\{p\}\rangle$。由定理 1，程序 P 只有一个回答集：$\{r,p\}$。

2.3 环和否定圈

已知正规逻辑程序 P，a 为 DG_P 的一个节点，将 a 可达节点集记作 $ReaN(a)$，并且约定：

1)$ReaRn(a,P)=\{r\mid r\in P,head(r)\in ReaN(a),\exists p.(p\in$

$body(r) \wedge p \in ReaN(a))$，且 $\forall b.(b \in ReaN(a) \wedge a \notin ReaN(b))\}$，其中 $head(r) \in ReaN(a)$，$\exists p.(p \in body(r) \wedge p \in ReaN(a))$表明规则 r 至少有一条关联边的两个节点包含在 $ReaN(a)$中，$\forall b.(b \in ReaN(a) \wedge a \neq b \wedge a \notin ReaN(b))$表示 $ReaN(a)$是 a 的可达节点的集合，且 a 对于该集合中的其他节点均不可达。显然 $ReaRn(a,P)$中的节点不构成回路。

2) $EP(P,B)=\{r' \mid \exists r.(r \in B), r' \in P$，且 $head(r)=head(r'), body(r) \subseteq body(r')\}$，其中 P 和 B 为正规逻辑程序。显然，$EP(P,B)$是 P 的一个子集，并可由 B 中的规则扩展得到。

定义 4(环)[1] 已知 P 为正规逻辑程序，$L \subseteq Lit(P)$，对 L 中的任意两个文字 p 和 q，在 DG_P 中，如果至少存在一条从 p 到 q 长度大于 0 的路径，该路径所经过的节点均为 L 中的元素且所经过的边均不为否定边，则称 L 为逻辑程序 P 的一个环[9]。

已知正规逻辑程序 P，L 为与其相关联的环，定义：

$$R^+(L,P)=\{r \mid r \in P, head(r) \in L, \exists q.\ q \in body(r) \wedge q \in L)\}$$

$$R^-(L,P)=\{r \mid r \in P, head(r) \in L, \neg \exists q.\ \in body(r) \wedge q \in L)\}$$

在原子推理图中，该定义中的 $R^+(L,P)$表示环中的边的所有关联规则组成的集合；

$R^-(L,P)$表示在原子推理图中，入环的边的所有关联规则组成的集合。

例 5 已知逻辑程序

$P=\{a:\text{-}\ b.\ b:\text{-}\ a.\ h:\text{-}not\ c.\ c:\text{-}\ not\ h.\ a:\text{-}\ not\ f.\ f:\text{-}\ not\ g.\}$

则 P 程序存在一个环 $L=\{a,b\}$。于是：

$R^+(L,P)=\{a:\text{-}\ b.\ b:\text{-}\ a.\}$，$R^-(L_1)=\{a:\text{-}\ not\ f.\}$

根据定义，如果 P 中两个环 L_1 和 L_2 的交集不为空，则这两个环公式的并集仍然为环。

定义 5(环公式)[1] 令 P 是一个逻辑程序，L 为该程序的一个环，$R^-(L,P)=\{r_1,\cdots,r_n\}$，则程序 P 关于 L 的环公式 $LF(L,P)$定义为：

$$LF(L,P)=\neg\Big(\bigvee_{r_i \in R^-(L,P)} body(r_i)\Big) \supset \bigwedge_{r_i \in R^-(L,P)} \neg head(r_i).$$

定义 6(否定圈) 已知逻辑程序 P，如果该程序的 DG_P 中存在长度大于 0 的否定回路，则将该回路称为否定圈。

下文中，利用一个否定圈中所有边的关联规则的集合来表示该否定圈。

已知正规逻辑程序 P 和 P 的一个否定圈 NL，约定下面 4 个相关规则集：

$$NR^+(NL,P)=\{c \in ReaRn(a,P) \mid a \in Lit(NL)\}$$

$$NR^0(NL,P)=\{r \mid r \in P, head(r) \in NL, (\forall p).\ p \in body(r) \wedge p \in Lit(NL)\}$$

$$NR^{-1}(NL,P)=\{r \mid r \in P, head(r) \in NL, \neg(\exists p).\ p \in body(r) \wedge p \in Lit(NL)$，且 $head(r) \in (Ans(P \backslash (NL(NR^+(NL,P))))\}$

$$NR^{-2}(NL,P)=\{r \mid r \in P, head(r) \in NL, body(r) \cap Lit(NL) \neq \emptyset, (\exists p)\ p \in body(r) \wedge p \notin Lit(NL)$且 $p \notin Ans(P/(NL \cup NR^+(NL,P)))\}$

其中，$Ans(P)$表示 P 的回答集。在此定义中，$NR^+(NL,P)$表示满足如下性质的规则 r：r 不在否定圈 NL 中，且 $Lit(r)$中的原子与否定圈 NL 中的节点之间存在通路；$NR^0(NL,P)$表示满足如下规则 r：r 的均为否定圈 NL 中规则；$NR^{-1}(NL,P)$表示满足如下性质的规则 r：$head(r)$属于否定圈中的原子，且 $head(r)$为否定圈外的规则集能被推出；$NR^{-2}(NL,P)$表示为如下的规则 r：$body(r) \backslash Lit(NL)$在圈外的规则无法推出。

例 6 已知逻辑程序

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

该程序存在两个否定圈：$NL_1=\{a:\text{-}not\ b.\ b:\text{-}not\ c, not\ a.\}$，$NL_2=\{a:\text{-}not\ b.\ b:\text{-}not\ a.\}$，$NR^0(NL_1,P)=\{a:\text{-}not\ b.\ b:\text{-}not\ a.\}$，$NR^+(NL_2,P)=\{a:\text{-}not\ b.\ b:\text{-}not\ a.\}$，$NR^{-1}(NL_1,P)=\{a:\text{-}not\ f.\}$，$NR^{-2}(NL_1,P)=\{b:\text{-}not\ c, not\ a.\}$和 $NR^{-1}(NL_2,P)=\{a:\text{-}not\ f.\}$。

由定义 6 可知，如果两个否定圈 NL_1 和 NL_2 相交，即 $NL_1 \cap NL_2 \neq \emptyset$，则其并集仍为否定圈，于是：

$$NR^0(NL_1 \cup NL_2,P)=NR^0(NL_1,P) \cup NR^0(NL_2,P)$$

3 方法概述

本节介绍本文回答集存在性判断方法的基本思想。

已知 ASP 程序 P，其回答集存在性判断过程如下。

首先，对该逻辑程序进行约减，方法如下：给出逻辑程序 P 的原子推理图[10]，在该图中，如果正规则的关联边与原子推理图中的否定边不连通，则该规则与逻辑程序回答集存在性的判断无关，可直接删除；在原子推理图中，如果在圈中无否定边，则为环[9]，可以根据环公式[9]删除该圈中所有边的关联规则，这并不影响其回答集的存在性；对于规则 r，如果原子 A 属于 $Pos(r)$，并且不是任意规则的头部，则删除该规则；按照以上方法对程序 P 进行约减，直至无法进一步减少规则，此时得到程序 P'。

然后，对程序 P'进行分析。如果 P'不存在否定圈，则该程序为分层逻辑程序，存在回答集，定义为第一类程序。如果 P'的原子推理图中存在否定圈，则根据圈的性质分为两种情形：a)存在一条规则 r，$Pos(r)$为空集，且 $Neg(r)$中出现的文字在程序 P'中均无定义，本文称为第二类程序。可以证明该类程序存在回答集；b)如果不符合情形 a)，则称其为第三类程序。对此类程序回答集的存在性可进行如下分析：

假设某个第三类逻辑程序存在一个回答集 M，则对于圈中的任意一条规则 r，要么 r 被用于推出 $head(r)$，要么未被运用。

1. 如果 r 被用于推出 $head(r)$，则 $head(r) \in M, body(r)$中的文字均成立。为了表示这一情形，可把该程序原子推理图中由 r 产生的边标注为 1，并记 $S=\{head(r)\}$；

2. 程序 P 的约简：利用 S 对 P 约减得 $P^S=\{r \mid r \in P, Neg(r) \cap S=\emptyset\}$，并把 $P \backslash P^S$ 中规则所产生的边均标注为 0。对 P^S 中出现的所有规则 r'，令 $A=Pos(r') \cap S$，如果 A 不为空，则在规则 r'中删除 A；如果 $Pos(r')$中存在 P^S 未定义的原子，则该原子在模型 M 中不可能为真，即规则 r'不可能被用于推导出其他结论，因此可把由 r'所产生的边均标注为 0；如

果在 $Neg(r')$ 中存在 P^S 未定义的原子，则删除该原子，约减后的程序为 P''。假设 P'' 中的事实集合为 F，令 $S=S\cup F$，用相同的方法利用 S 对程序 P'' 进行约减，直到无法约减为止。根据本文定理 4 和推论 1，以上约简过程均不会改变程序 P 的回答集 M 的存在性。

3. 最后将未被标住的边均标为 1。如果在以上标记过程中，规则 r 产生的所有边的标记数值均未被改动（即全部为 1），则程序 P 存在回答集，即为 S；否则如果存在规则 r 所产生的一条边，其标记由 1 修改为 0，则表明“规则 r 被用于推出 $head(r)$”是不合理的，于是需要考虑“r 未被用于推出 $head(r)$”的情况，即 $head(r)\notin M$（如果该假设是合理的，则 P 存在回答集；否则 P 不存在回答集）。为此，在程序 P 中将规则 r 所产生的边标为 0（表示 r 未被用于推出 $head(r)$），令 $S=\emptyset$；按照相同的方法对程序 P 重新进行标注；如果标注过程中由 r 产生的边只被标注为 0，则逻辑程序 P 存在回答集，即为 S；否则 P 不存在回答集。

在上面的例 1 中，逻辑程序的原子推理图如图 2(a)所示。

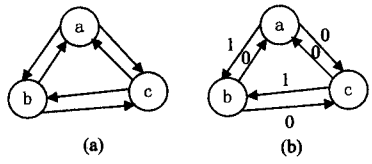

图 2　例 1 中原子推理图和标注后的原子推理图

根据以上讨论，该程序为第三类程序。其回答集存在性判断过程如下：假设存在 P 的一个回答集 M，b :-*not a*, *not c*. 可被用于推导；即 $b\in M$，则边 (a,b)，(c,b) 均被标为 1，$S=\{b\}$。接着对逻辑程序 P 进行约简：P^S 为 $\{b$:-*not a*, *not c*. $\}$，而 $P\backslash P^S$ 为 $\{c$:- *not a*, *not b*. *a* :-*not b*, *not c*$\}$，$P\backslash P^S$ 的关联边 (c,a)，(b,a)，(a,c)，(b,c) 均被标为 0；由于 a 和 c 在 P^S 中均无定义，P^S 进一步约减为 $\{b$:-$\}$，至此无法再约减。最后规则 b :-对应原程序中的规则为 b :-*not a*, *not c*。该规则的关联边为 (a,b)，(c,b)，标为 1，规则 b :-*not a*, *not c* 在图 1 中的边被标注的值未被改变。所以该逻辑程序存在一个回答集 $S=\{b\}$。

4　逻辑程序的约简

本节主要介绍逻辑程序的约简，以提高正规逻辑程序回答集存在性判断的效率。任意程序 P 的约简均可通过以下两个步骤实现。

步骤 1　已知 ASP 程序 P，

规则 1)　对 P 中的任意正规则 r，如果 r 的关联边与原子推理图中的否定边均不连通，则该规则与逻辑程序回答集存在性的判断无关，可直接删除；

规则 2)　对 P 中的任意规则 r，如果 $Pos(r)$ 中包含有在 P 中未定义的原子，则删除该规则。

显然，规则 2)不会改变回答集的存在性。而根据下面的定理 2，规则 1)也不会改变程序回答集的存在性，因此利用步骤 1 对程序 P 进行约简并不改变其回答集的存在性。

例 7　已知正规逻辑程序

$P=\{c$:-*c*, *b*. *a* :- *b*, *c*. *b* :-*c*, *a*. *e* :-*f*, *not d*. *m* :-*not e*. *e* :-*not m*. $\}$

由规则 1)，得 $P'=\{e$:-*f*, *not d*, *m* :-*not e*. *e* :-*not m*. $\}$；由规则 2)得 $P''=\{m$:-*not e*. *e* :-*not m*. $\}$，于是程序 P 回答集的存在性等价于 P'' 回答集的存在性。

定理 2　已知正规逻辑程序 P，$P'=\{r\mid r\in P, r$ 为正规则，且 r 的关联边与 P 的原子推理图中的否定边不连通$\}$，P 的回答集存在性与 P' 无关，只与 $P\backslash P'$ 有关。

证明：在逻辑程序 P 中，因为 P' 为无否定的正规逻辑程序，所以 P 存在唯一的最小 *herbrand* 模型[12] M，并且 M 为 P' 的回答集[2]。又 P' 中规则的关联边与原子推理图中的否定边不连通，所以可知 $Lit(P')\cap Lit(P\backslash P')=\emptyset$，设 $U=Lit(P')$，则 $b_U(P)=P'$，$e_U(P\backslash P', M)=P\backslash P'$。根据定理 1，得 $Ans(P)=M\cup Ans(P\backslash P')$，所以 P 的回答集存在性与 P' 无关，只与 $P\backslash P'$ 有关。

下面通过环的概念对逻辑程序进行约简。

步骤 2　已知逻辑程序 P，L 为 P 的一个环，

规则 1)　如果对于任意 $r\in R^-(L,P)$，$body(r)$ 中的原子均无法在 P 中推导出，则删除 $R^+(L,P)$；

规则 2)　如果存在 $r\in R^-(L,P)$，$body(r)$ 中的原子均可在 P 中推导出，则

a)删除 $R^+(L,P)$；

b)对任意规则 $r\in P$，如果 $Pos(r)\cap L\neq\emptyset$，则在规则 r 中删除 $Pos(r)\cap L$；

c)对任意规则 $r\in P$，如果 $Nes(r)\cap L\neq\emptyset$，则在程序中删除 r。

根据环公式的定义可知，L 中的原子的真值由 $R^-(L,P)$ 决定，由文献[9]可知：如果 $\forall r\in R^-(L,P)$ 且 $body(r)$ 中的原子均无法在 P 中推出，则 L 在该程序的回答集中均为假，于是可从 P 中删除 $R^+(L,P)$ 中的规则，而不改变其回答集的存在性。同时如果 $\exists r.(r\in R^-(L,P))$，且 $body(r)$ 中的原子均可在 P 中推导出，则 L 在该程序回答集中均为真，于是可对程序 P 进行 a)、b)、c) 3 种形式的约简，该过程不改变该程序回答集存在性，且在约简后的程序中不再存在环。下面给出一个实例说明该方法。

以例 5 中程序 P 为例，$R^-(L_1)=\{a$:- *not f*. $\}$，且原子 f 在回答集中为真，根据步骤 2，可在 P 中删除 $R^+(L,P)$，于是 P 的回答集存在性由 $P''=\{h$:-*not c*. *c* :- *not h*. *a* :- *not f*. *f* :- *not g*. $\}$ 决定。下文中，利用步骤 1 和步骤 2 对程序 P 进行约简后所得的程序记作 $Reduct(P)$。

5　正规逻辑程序的分类

本节根据否定圈性质将正规逻辑程序分为 3 类，然后利用否定圈的性质及划分集的概念，分别对各类正规逻辑程序回答集的存在性进行讨论。

定义 7(第一类正规逻辑程序)　设 P 为一个正规逻辑程序，如果该程序的原子推理图中不存在否定圈，则该程序为第一类正规逻辑程序。

此类逻辑程序中由于不存在否定圈，因此为分层逻辑程序，存在回答集[10]。例如，逻辑程序 $P=\{a$:-*not b*, *b* :- *not c*$\}$ 不存在否定圈，为第一类程序，存在回答集。

定义 8(第二类正规逻辑程序)　设 P 为一个正规逻辑程序，如果该程序原子推理图中的任意否定圈 NL 至少满足以下性质之一，则该程序为第二类正规逻辑程序：

1. $NR^{-1}(NL,P)\neq\emptyset$;

2. $NR^{-2}(NL,P)\neq\emptyset$。

定理 3 如果 P 为第二类正规逻辑程序，则 P 存在回答集。

证明：任取否定圈 NL，

1. 如果 $NR^{-1}(NL,P)\neq\emptyset$；任取规则 $r\in NR^{-1}(NL,P)$，于是 $head(r)\in Ans(P\backslash(NL\cup(NR^{+}(NL,P)))$。设 $U=Lit(P\backslash(NL\cup NR^{+}(NL,P)))$，$X$ 为子程序 $P\backslash(NL\cup NR^{+}(NL,P))$的回答集，则 $e_U(NL\cup NR^{+}(NL,P),X)$中不包含头为 $head(r)$的规则集。而 $head(r)\in Lit(NL)$，因此 $e_U(NL\cup NR^{+}(NL,P),X)$不存在否定圈，即 $e_U(NL\cup NR^{+}(NL,P),X)$为分层逻辑程序。且由于 $X\cup Ans(e_U(NL\cup NR^{+}(NL,P),X))$一致，因此 P 存在回答集。

2. 如果 $NR^{-2}(NL,P)\neq\emptyset$，任取规则 $r\in NR^{-2}(NL,P)$，于是 $body(r)$中的原子与 $Ans(P\backslash(NL\cup NR^{+}(NL,P)))$。设 $U=Lit(P\backslash(NL\cup NR^{+}(NL,P)))$，则 X 为子程序 $P\backslash(NL\cup NR^{+}(NL,P))$的回答集与 $body(r)$的原子相矛盾，因此 $r\notin e_U(NL\cup NR^{+}(NL,P),X)$，则该否定圈 NL 被消除，即 $e_U(NL\cup NR^{+}(NL,P)$不存在否定圈。由于该否定圈为任意性，则可知该程序最后均可转化为分层逻辑程序，于是 P 存在回答集。

由以上证明过程可知，第二类逻辑程序存在的否定圈均可通过 $NR^{-1}(NL,P)$中的规则进行消除，且该过程不影响 P 回答集的存在性。

例 8 已知逻辑程序 $P=\{a:\text{-}not\ b, b:\text{-}not\ c.\ b:\text{-}not\ e.\ e:\text{-}not\ a, not\ f.\ f:\text{-}not\ c.\}$，$NR^{-1}(NL,P)=\{b:\text{-}not\ c\}$，$NR^{-2}(NL,P)=\{e:\text{-}not\ a, not\ f\}$，通过消除 $NR^{-1}(NL,P)$和 $NR^{-2}(NL,P)$，P 被约减为$\{a:\text{-}not\ b.\ b:\text{-}not\ c.\ f:\text{-}not\ c.\}$。从而 P 被转化为分层逻辑程序，因此原程序 P 存在回答集。

定义 9(第三类正规逻辑程序) 设 P 为正规逻辑程序，如果该程序的原子推理图满足以下条件，则该程序为第三类正规逻辑程序，即该程序原子推理图中存在一个否定圈 NL，并且满足以下两个条件：

1. $NR^{-1}(NL,P)=\emptyset$；

2. $NR^{-2}(NL,P)=\emptyset$。

第三类程序为带否定圈的逻辑程序，且属于该否定圈的原子无法从不属于该圈定理的规则推出，对此类正规逻辑程序只需判断该程序的否定圈是否存在回答集。如果该程序所有的否定圈均存在回答集，则该程序存在回答集，否则便不存在回答集。否定圈回答集的存在性判断见第 6 节。

例 9 已知逻辑程序 $P=\{a:\text{-}not\ b, b:\text{-}not\ c.\ c:\text{-}not\ a.\ c:\text{-}not\ e.\ e:\text{-}not\ f\}$，其中 $NL=\{a:\text{-}not\ b, b:\text{-}not\ c.\ c:\text{-}not\ a.\}$为 P 的否定圈，$NR^{-1}(NL,P)=\emptyset$，$NR^{-2}(NL,P)=\emptyset$。根据定义 9 知 P 为第三类正规逻辑程序。

5.1 逻辑程序分类算法

已知逻辑程序 P，图 4 给出了一种判断 P 的类型的算法。其基本思想是：首先对 P 进行约简，然后判断约简后的程序是否存在否定圈，如果不存在否定圈，则 P 属于第一类程序；如果存在否定圈，则为第二类或者第三类程序。对于后者，如果其任意否定圈中的所有原子都可以由圈外的规则推出，则 P 为第二类程序；否则为第三类程序。其中判断程序是否存在否定圈通过 Step2、Step3 实现。显然，如果 P 不存在否定圈，则在 Step4 中，G 的节点数必然为 0，于是可得 P 为第一类程序。判断否定圈内的原子是否可由圈外规则推出，由 Step5 和 Step6 实现，具体过程如下：1) 利用 S 对 P 进行约简：设 r 为 P^S 中出现的规则，如果 $Pos(r)\cap S$ 不为空，则 $Pos(r)\cap S$ 在 P^S 的回答集中为真，于是可从 r 的规则体中删除 $Pos(r)\cap S$；如果 $Pos(r)$中有 P^S 未定义的原子，则该原子在 P^S 回答集中为假，于是可删除规则 r；如果在 $Neg(r)$中存在 P^S 中未定义的原子 A，则 A 无法在 P^S 中推出，$not\ A$ 为真，于是可在 r 中删除 $not\ A$(该过程见图 3 中的 $TRS()$算法)。约简后的程序为 P'。令 P'中的事实为 F，$S:=S\cup F$。利用相同的方法对 P'进行约简，直到无法约简为止，得到程序 P''；2) 最后如果 P''中都为事实，则知 P 中的否定圈可由圈外规则推出，为第二类程序；否则为第三类程序。定理 4 表明 $TRS()$算法不会改变该程序的回答集。为了证明该定理，首先引入定义 10 及定理 4。

函数：$TRS(P,S)$
输入：正规逻辑程序 P 和 S；
输出：经过转换后的程序 P'。
begin
Step1 对任意规则 $r\in P$，如果 $a\in Pos(r)$且 $a\in S$，则在规则 r 中删除 a，得到规则 r'，$P:=(P\backslash\{r\})\cup\{r'\}$；
Step2 对任意规则 $r\in P$，如果 $a\in Neg(r)$且 $a\in S$，则 $P:=P\backslash\{r\}$；
Step3 对任意规则 $r\in P$，如果 $a\in Pos(r)$且 a 在 P'中未定义，则 $P:=P\backslash\{r\}$；
Step4 对任意规则 $r\in P$，如果 $a\in Neg(r)$且 a 在 P 中未定义，则在 r 中删除 $not\ a$，得到 r'，$P:=(P\backslash\{r\})\cup\{r'\}$；
Step5 返回 P。
end

图 3 $TRS(P,S)$算法

函数：$Asst(P,GD_P)$
输入：正规逻辑程序 P 和 P 的原子推理图 GD_P；
输出：如果为第一类程序，则输出 1；
如果为第二类程序，则输出 2；
如果为第三类程序，则输出 3。
begin
Step1 令 $P:=Reduct(P)$，$G:=GD_P$；
Step2 在 G 中，删去所有入度为 0 的节点，及以这些节点为起点的边；
Step3 重复 Step 2 直至 G 的节点数不再减少；
Step4 如果 G 的节点集为空，则返回 1；否则令 $S:=\emptyset$，转 Step 5；
Step5 $P:=TRS(P,S)$，P 中的事实记作 F；
Step6 如果 $S=S\cup F$，则转 Step 7，否则 $S:=S\cup F$，转 Step 5；
Step7 如果 P 中均为事实，则返回 2，否则返回 3。
end

图 4 $Asst(P,DG_P)$ 算法

定义 10($PP(S,P)$) 设 P 为一逻辑程序，$S\subset Lit(P)$，$PP(S,P)=Lit(r|r\in P, head(r)\in S)$。

定理 4 逻辑程序 P 存在回答集 X，如果 $S\subseteq X$，$U=PP(S,P)$，则逻辑程序 $e_U(P,S)$的回答集为 M。

证明：设 $S\subseteq X$，$U=PP(X,P)$，根据定理 1，X 为 P 的回

答集，并且 X 是一致的，Y 为 $e_U(P\backslash b_U(P),S)$ 的回答集，则知 $X=S\cup Y$。而根据 2.1 节式子可得 $e_U(P,S)=e_U(P\backslash b_U(P),S)\cup e_U(b_U(P),S)$，并且 S 为 $b_U(P)$ 的回答集，同理知 $e_U(b_U(P),S)$ 的回答集为 S，此处 X 为 P 的回答集，且 $S\subseteq X$，所以 $e_U(P\backslash b_U(P),S)$ 不存在与 S 不一致的规则，于是 $e_U(P\backslash b_U(P),S)\cup e_U(b_U(P),S)$ 的回答集为 $S\cup e_U(P\backslash b_U(P),S)$ 的回答集，即 $M=S\cup Y$，因此 $X=M$。

推论 1 如果 M 为逻辑程序 P 的回答集，$S\subseteq M$，则 M 为逻辑程序 $TRS(P,S)$ 的回答集。

证明：设 M 为 P 的回答集，$U=PP(S,P)$，P 经过 $TRS()$ 算法的 Step1 和 Step2，得 $e_U(P,S)$，根据定理 4 得，模型 M 即等于逻辑程序 $e_U(P,S)$ 的回答集，而在逻辑程序 $e_U(P,S)$ 中，经过 Step4，对于任意 $r\in e_U(P,S)$：如果 $Pos(r)$ 存在 $e_U(P,S)$ 中未定义的原子，则该原子无法被推出，因此该规则不能用于该回答集的推导，于是删除该规则 r，得程序 P'。如果规则 r 中的 $Neg(r)$ 存在 a，并且 a 在 P' 中未定义，即 $a\notin M$，则 $TRS(P,S)$ 的回答集即为 M。

根据图 4 的算法可将正规逻辑程序分为 3 类。如果返回 1，则为第一程序；返回 2，则为第二类程序；返回 3，则为第三类程序。下面定理 5 给出了算法 $Asst(P,DG_P)$ 的正确性证明。

定理 5 已知 P 为正规逻辑程序，P 的原子推理图为 DG_P，

1）如果 $Asst(P,DG_P)=1$，则 P 为第一类正规逻辑程序；

2）如果 $Asst(P,DG_P)=2$，则 P 为第二类正规逻辑程序；

3）如果 $Asst(P,DG_P)=3$，则 P 为第三类正规逻辑程序。

利用反证法证明，根据第一类正规逻辑程序的定义及定理 4 和推论 1 容易验证 1）为真，同理，2）和 3）同样为真，限于篇幅，证明过程在此不再给出。

6 否定圈回答集存在性判断

本节讨论第三类逻辑程序中否定圈回答集的存在性判断方法。第三类程序的否定圈 NL 可分两类：

a) $NR^{-1}(NL,P)=\emptyset$ 和 $NR^{-2}(NL,P)=\emptyset$，此时称 NL 为第一类否定圈；

b) 如果 $(NR^{-1}(NL,P)\neq\emptyset)$ 或 $(NR^{-2}(NL,P)\neq\emptyset)$，此时称 NL 为第二类否定圈。

下文针对这两类否定圈，分别给出了两个存在性判定算法（图 5 的 $DN()$ 算法和图 6 的 $DNE()$ 算法）。算法 $DN(P)$ 判断第一类否定圈回答集的存在性：如果 P 存在一个回答集 M，则对于圈中的任意一条规则 r，要么 r 被用于推出 $head(r)$，要么未被运用。具体实现如下：

1）如果 r 被用于推出 $head(r)$，则 $head(r)\in M$，$body(r)$ 中的文字均成立。为了表示这一情形，可把该程序原子推理图中由 r 产生的边标注为 1，并记 $S=\{head(r)\}$，该过程由算法的 Step1 实现。

2）P 的约简：假设 $TRS(P,S)$ 中的事实集合为 F，令 $S=S\cup F$，用相同的方法对 $TRS(P,S)$ 进行约减，直到无法约减为止，令 $A:=EP(P_1,P)$，$B:=B\cup P_1/A$，这里 A 中规则即为在概论叙述中关联边被标注为 1 的规则，B 中规则为其关联边均被标注为 0 的规则，该过程由算法的 Step2、Step3 和 Step4 实现。

3）如果 $r\in A$，则 S 满足该否定圈，于是 P 存在回答集 S；如果在 P 中存在一条规则 r'，使得 $head(r)\in Neg(r')$，则通过以上方法判断；否则令 $r=r'$，并在 P 中取出规则 r''，直到 $head(r)\in Neg(r'')$ 为止，令 $r=r''$，令 $S=head(r)$，利用 1）、2）迭代计算。该过程由算法 $DN(P)$ 的 Step7 实现。

函数：$DN(P)$；
输入：否定圈 P；
输出：如果不存在回答集，输出 0；否则输出一个回答集 S；
begin
Step1 在程序 P 中取出组成该逻辑程序的规则 r，$P_1:=P$；且 $A:=\{r\}$，$B=\emptyset$，$S:=\{head(r)\}$，$T:=1$；
Step2 $P:=TRS(P,S)$，令 F 为 P 中的事实；
Step3 如果 $S\neq S\cup F$，则 $S:=S\cup F$，转 Step2，否则转 Step4；
Step4 如果 $P\neq F$，取出 P 中的圈 P'，转 Step5；
否则 $A:=EP(P_1,P)$，$B:=B\cup P_1/A$，转 Step6；
Step5 如果 $DN(P')=0$，如果 $T=2$，则返回 0；如果 $T=1$，转 Step7；
否则 $S:=S\cup DN(P')$，转 Step2；
Step6 如果 $r\notin B$ 且 $T=1$，则返回 S；如果 $r\in B$ 且 $T=1$，则转 Step7；如果 $r\notin B$ 且 $T=2$，则返回 S；否则返回 0；
Step7 令 $S=\emptyset$，并在 P 中取出规则 r''；如果 $head(r)\cap Pos(r'')\neq\emptyset$，则 $r:=r''$，转到 Step7；如果 $Neg(r'')\cap head(r)\neq\emptyset$，则 $r:=r''$，转为 Step2。
end

图 5 $DN(P)$ 算法

函数：$DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))$；
输入：逻辑程序的否定圈 NL，$NR^{-1}(NL,P)$ 和 $NR^{-2}(NL,P)$；
输出：存在回答集输出 1，不存在回答集输出 0；
begin
Step1 如果 $NR^{-1}(NL,P)\neq\emptyset$，则取出所有的规则 r，$A:=A\cup head(r)$，$U:=PP(NR^{-1}(NL,P))$，令 $P:=e_U(NL,A)$，否则 $P=NL$；
Step2 如果 $NR^{-2}(NL,P)\neq\emptyset$，$P:=P/NR^{-2}(NL,P)$；
Step3 如果 P 不存在否定圈 NL，则该程序存在回答集，返回 1；否则转 Step4；
Step4 取出否定圈 NL'，如果 $NR^{-1}(NL',P)\neq\emptyset$ 或者 $NR^{-2}(NL,P)\neq\emptyset$，则返回 $DNE(NL,NR^{-1}(NL',P),NR^{-2}(NL',P))$，否则转 Step5；
Step5 如果 $DN(NL')\neq0$，则返回 1；否则返回 0。
end

图 6 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))$ 算法

例 10 $DN()$ 算法的应用。已知程序 $P=\{a:\text{-}not\ b.\ b:\text{-}not\ c,not\ d.\ c:\text{-}not\ d.\ d:\text{-}not\ a.\}$ 的原子推理图如图 7 所示（图中的边均为否定边）。

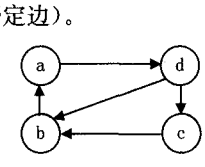

图 7 例 10 中的原子推理图

由图 7 可知 P 中存在第一类否定圈，根据算法，首先取出规则 $a:\text{-}not\ b.$ 得 $S=\{a\}$，执行 Step1 可得 $P=TRS(P,S)=\{a:\text{-}\ not\ b.\ b:\text{-}not\ c.\ c:\text{-}.\}$，$S=\{a,c\}$，经过 Step2 得 $P=TRS(P,S)=\{a:\text{-}\ not\ b.\ b:\text{-}not\ c.\ c:\text{-}.\}$，$S=\{a,c\}$，此时

$A=\{a:\text{-}not\ b.\ c:\text{-}\ not\ d.\}$，$B=\{b:\text{-}not\ c, not\ d.\ d:\text{-}\ not\ a.\}$，由于 $a:\text{-}not\ b\notin B$，则该程序存在回答集 S。

下面定理 6 给出了算法 $DN()$ 的正确性。

定理 6 已知 P 为第一类否定圈，

1) 如果 $DN(P)\neq 0$，则 P 存在回答集为 $DN(P)$；

2) 如果 $DN(P)=0$，则 P 不存在回答集。

根据定理 6，如果 P 为一个正规逻辑程序，且 $NR^{-1}(NL,P)$ 或者 $NR^{-2}(NL,P)$ 不为空集，则该否定圈受 P 中的其他规则影响；如果是一个单独否定圈，通过删除 $NR^{-2}(NL,P)$ 中的规则或者删除 $Lit(NR^{-1}(NL,P))$ 中的原子，使该否定圈被破坏，变成了分层逻辑程序，则存在回答集；如果是多个否定圈的情况，则还需判断其余的否定圈；如果所有的否定圈均为可被破坏，则该逻辑程序存在回答集，图 6 中给出了相关的算法。下面利用 $DNL(P,NR^0(NL,P),Ans(P/(NL\cup NR^+(NL,P)))$ 操作来判断否定圈是否存在回答集；对于一个否定圈，如果 $NR^{-1}(NL,P)$ 和 $NR^{-2}(NL,P)$ 至少一个集合不为空，则可根据图 6 的算法来判断，在该算法中，可利用划分的概念验证该算法的有效性，为此引入定理 7。

算法 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))$ 用于判定第二类否定圈回答集的存在性：1) 如果 $NR^{-1}(NL,P)\neq\emptyset$，则存在 $r\in NR^{-1}(NL,P)$。令 $S_1:=\{head(r)\}$，$P_1:=e_U(NL,S)$，F 为 P 中的事实，则 $S_1:=S\cup F$。通过以上方法再次求 $TRS(P,S)$，直到 $S=S\cup F$，最后得到程序 P'。如果 P' 的规则均为事实，则否定圈 NL 存在回答集，否则 NL 中存在第一类否定圈，可调用 $DN(P)$ 算法进一步判断。2) 如果 $NR^{-2}(NL,P)\neq\emptyset$，且 $NL\backslash NR^{-2}(NL,P)$ 中不存在否定圈，则该否定圈存在回答集；如果仍然存在第二类否定圈，则调用 $DNE()$ 进一步判断，否则存在第一类否定圈，则调用 $DN()$ 再次判断。

例 11 $DNE()$ 算法的应用。已知程序 $P=\{b:\text{-}not\ a.\ b:\text{-}\ not\ h.\ h:\text{-}not\ b.\ g:\text{-}not\ h.\ c:\text{-}not\ b.\ c:\text{-}not\ d.\ d:\text{-}not\ e.\ e:\text{-}not\ f.\ f:\text{-}not\ c.\}$ 的原子推理图如图 8 所示（图中的边均为否定边）。

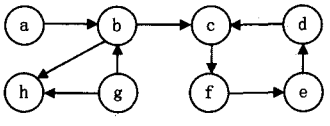

图 8 例 11 中的原子推理图

根据定义 9，P 为第三类程序，且从图 8 可得 P 存在两个否定圈，分别为：$NL_1=\{b:\text{-}\ not\ h.\ h:\text{-}not\ b.\ g:\text{-}not\ h.\}$，$NL_2=\{c:\text{-}not\ d.\ d:\text{-}not\ e.\ e:\text{-}not\ f.\ f:\text{-}not\ c.\}$，并且 $NR^{-1}(NL_1,P)=\{b:\text{-}not\ a.\}$ 和 $NR^{-2}(NL_1,P)=\emptyset$，因此否定圈 NL_1 为第二类否定圈，于是在 NL_1 对应的原子推理图中删除节点 b，得 $TRS(NL_1,\{a\})=\{g:\text{-}not\ h.\}$，且该程序不存在否定圈，根据第一类正规逻辑程序知 $TRS(NL_1,\{a\})$ 存在回答集，因此 NL_1 存在回答集。同时由于 $NR^{-1}(NL_2,P)=\emptyset$ 和 $NR^{-2}(NL_2,P)=\emptyset$，故 NL_2 为第一类否定圈，调用 $DN()$ 可得 NL_2 存在回答集。因此程序 P 存在回答集 S。下面由定理 7 证明算法 $DNE()$ 的正确性。

定理 7 P 为一个正规逻辑程序，设 NL 为一个第二类否定圈，

1)如果 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=1$，则 NL 存在回答集；

2)如果 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=0$，则 NL 不存在回答集。

证明：

1)利用反证法证明。假设 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=1$，则 NL 不存在回答集。已知 NL 为一个第二类否定圈，则 $NR^{-1}(NL,P)\neq\emptyset$ 或者 $NR^{-2}(NL,P)\neq\emptyset$。设 A 为程序 $P\backslash(NL\cup NL^+(NL,P)$ 的回答集，根据定理 1，则 P 的回答集由 $P'=e_U(NL,A)$ 决定，因 NL 不存在回答集，可知 P' 不存在回答集，则 P' 为第三类正规逻辑程序且 P' 存在否定圈 NL'，并且 $DN(NL')=0$，根据算法 $DNE()$ 得 $DNE()=0$，与假设矛盾。假设不成立，因此如果 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=1$，则 NL 存在回答集。

2)如果 $DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=0$，已知 NL 为一个第二类否定圈，则 $NR^{-1}(NL,P)\neq\emptyset$ 或者 $NR^{-2}(NL,P)\neq\emptyset$。设 A 为程序 $P\backslash(NL\cup NL^+(NL,P)$ 的回答集，根据定理 1，则 P 的回答集由 $P'=e_U(NL,A)$ 决定，$DNE(NL,NR^{-1}(NL,P),NR^{-2}(NL,P))=0$，则在 P' 中必存在一个第一类否定圈 NL'，使 $DN(NL')=0$，且 P' 为第三类正规逻辑程序，于是 P' 的稳定模型由 NL' 决定，NL' 不存在回答集，则 P' 不存在回答集，于是 NL 不存在回答集。

由以上定理可知，如果正规逻辑程序 P 存在否定圈 NL，且对于 P 中所有的否定圈 NL 均存有 $NR^{-1}(NL,P)=\emptyset$ 和 $NR^{-2}(NL,P)=\emptyset$，则可将该逻辑程序回答集存在性的判断转化为 NL 回答集存在性的判断。如果 P 不存在否定圈或者对所有否定圈 NL 均有 $NR^{-1}(NL,P)\neq\emptyset$ 或者 $NR^{-2}(NL,P)\neq\emptyset$，则 P 存在回答集。

7 实例分析

本节利用实例来说明逻辑程序回答集存在性的判断方法，并将该结果与回答集求解器的求解结果相比较，从而说明该方法的有效性。

程序 $P=\{e:\text{-}d, a.\ a:\text{-}e.\ h:\text{-}e.\ g:\text{-}\ b.\ f:\text{-}a.\ g:\text{-}\ not\ f, not\ h.\ p:\text{-}not\ n.\ f:\text{-}\ not\ h, not\ g.\ h:\text{-}\ not\ f, not\ g.\ g:\text{-}p, not\ j.\ n:\text{-}not\ m\}$ 的原子推理图如图 9 所示，其中实线表示肯定边，虚线表示否定边。首先，对程序 P 进行约简，因为该程序既无独力的正规则，也不存在环，根据第 4 节步骤 1 的规则 2)，$\{e:\text{-}d, a.\ a:\text{-}e.\ h:\text{-}e.\ g:\text{-}\ b.\ f:\text{-}a\}$ 均被删除，故 $Reduct(P)=\{a:\text{-}not\ b, not\ c.\ b:\text{-}not\ a, not\ c.\ c:\text{-}\ not\ a, not\ b.\ p:\text{-}not\ n.\ n:\text{-}not\ m\}$，$Reduct(P)$ 的原子推理图如图 9(b)所示。在图 9(b)中，根据定义 6 得否定圈 $NL=\{a:\text{-}not\ b, not\ c.\ b:\text{-}not\ a, not\ c.\ c:\text{-}\ not\ a, not\ b.\ g:\text{-}not\ i.\}$，$NL$ 的原子推理图如图 9(c)所示，且有：$NR^{-1}(NL,Reduct(P)=\emptyset, NR^{-2}(NL,Reduct(P))=\emptyset$。根据定义 9，$P$ 为第三类正规逻辑程序。现只需判断组成图 9(c)逻辑程序回答集的存在性即可。在图 9(c)中，NL 为第一类否定圈，根据 $DN()$ 算法，假设存在 NL 的一个回答集 M，$b:\text{-}not\ a, not\ c.$ 可被用于推导，即 $b\in M$，则边 (a,b)，(c,b) 均被标为 1，$S=\{b\}$。接着对逻辑程序 P 进行约简：$TRS(NL,S)$ 为 $\{b:\text{-}.\}$，而 $A=\{b:\text{-}not\ a, not\ c.\}$，规则 $b:\text{-}not\ a, not\ c.$ 的关联边 (c,b)，(a,b) 均被标为 1；$B=\{a:\text{-}not\ b, not\ c.\ c:\text{-}\ not\ a, not\ b.\}$ 中规则的关联边分别为 (c,a)，(b,a)，(a,c)，(b,c)，且均被标注为 0，由于 $(b:\text{-}not\ a, not\ c)\notin$

B,故 $DN(NL)=\{a\}$,因此 NL 存在回答集$\{a\}$,因而 P 也存在回答集。

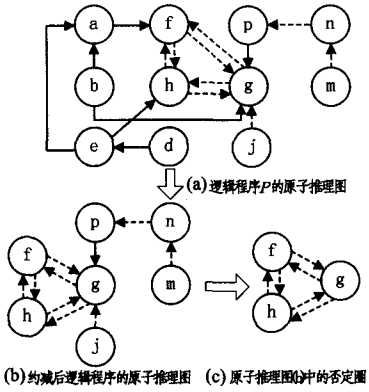

图 9

由 DLV[13] 求解知逻辑程序 P 的回答集为$\{a,n\}$,$\{b,n\}$,$\{c,n\}$。即该程序存在回答集,算法结论正确。

结束语 本文给出了一种判定正规逻辑程序存在性的方法框架。该方法首先对正规逻辑程序进行约简,然后将约简后的正规逻辑程序分为 3 类,从而将正规逻辑程序的判定问题转化为对正规逻辑程序类别的判定。给出了逻辑程序的分类算法并证明了算法的正确性。未来的工作包括:本文方法的时空效率和其他性能分析,利用本文方法构建回答集求解器,以及把该方法应用于判断正规逻辑程序回答集个数的下限。

参 考 文 献

[1] Lin F Z, Zhao X S. On Odd and Even Cycles in Normal Logic Programs[C]// Proceedings of the 19th National Conference on Artificial Intelligence (AAAI2004). 2004:80-85

[2] Gelfond M, Lifschitz V. The stable model semantics for logic programming[C]// Proceedings of 5th International Conference on Logic Programming (ICLP'1988). 1988:1070-1080

[3] Lifschitz V, Turner H. Splitting a Logic Program [C]// Proceedings of 11th International Conference on Logic Programming (ICLP'1994). 1994:23-37

[4] 钟勇,郭伟刚,钟昌乐,等. Datalog 逻辑程序调用语义及其应用研究[J]. 计算机科学,2010,37(1):170-175

[5] You J H, Yuan L. A Three-Valued Semantics for Deductive Database and Logic Programs[J]. Computer and System Science, 1999,49(2):334-361

[6] Kunen K. Signed Dada Dependencies in Logic Programs [J]. Logic Programming,1987,7(3):231-245

[7] Lin F Z, Zhao Y T. ASSAT: computing answer sets of a logic program by SAT solvers[C]// Proceedings of the 18th National Conference on Artificial Intelligence (AAAI2002). 2002,157(1/2):115-137

[8] Pontelli E. Answer Set Programming in 2010: A Personal Perspective[C]// Proceedings of 12th International Symposium on Practical Aspects of Declarative Languages (PADL' 2010). 2010:1-3

[9] Chen Xiao-ping, Ji Jian-min, Lin Fang-zhen. Computing Loops with at Most One External Support Rule for Disjunctive Logic Programs[C]// Proceedings of 25th International Conference on Logic Programming (ICLP'2009). 2009:130-144

[10] Baral C. Knowledge representation, reasoning, and declarative problem solving[M]. New York: Cambridge University Press, 2003

[11] 沈榆平,赵希顺. 一个带破圈启发方法的回答集编程系统[J]. 软件学报,2008,19(4):69-78

[12] Gelfond M, Przymusinska H. Teodor Przymusinski On the Relationship Between CWA, Minimal Model and Minimal Herbrand Model Semantics[J]. International Journal of Intelligent Systems,1990,5(5):549-564

[13] Leone N, Pfeifer G, et al. The DLV system for knowledge representation and reasoning[J]. ACM TOCL,2006,7(3): 499-562

(上接第 208 页)

[6] Velho L. Quasi 4-8 subdivision[J]. Computer Aided Geometric Design,2001,5(18):345-357

[7] Kobbelt L. $\sqrt{3}$-subdivision [C]// Computer Graphics Processdings. Annual Conference Series. ACM SIGGRAPH,2000:103-112

[8] Oswald P, Schroder P. Composite primal/dual-subdivision schemes[J]. Comput. Aided Geom. Design,2003,20(3):135-164

[9] 吴剑煌,刘伟军,王天然.$\sqrt{3}$细分曲面的误差分析[J]. 机械工程学报,2007,43(2)

[10] Jiang Qing-tang. Orthogonal and Biorthogonal $\sqrt{3}$-refinement Wavelets for Hexagonal Data Processing[J]. IEEE Transactions on Signal Processing,2009,57(11)

[11] Ghulam M, Sadiq H, Faheem K.$\sqrt{3}$细分曲面的误差界[J]. 中国科学技术大学学报,2009,39(6)

[12] Labsik U, Greiner G. Interpolatory $\sqrt{3}$-subdivision[J]. EUROGRAPHICS 2000/M. Gross F. R. A. Hopgood,2000,19

[13] 彭育辉,高诚辉. 基于形状修正的三角网格模型顶点法失估算方法[J]. 中国图象图形学报,2010,15(1)

[14] Zhou Kun, Huang Jin, Snyde J, et al. Large mesh deformation using the volumetric graph Laplacian[J]. ACM SIGGRAPH, ACM Transactions on Graphics(TOG),2005,24(3):496-503

[15] Chung F R K. Spectral Graph Theory[C]// Volume 92 of Regional Conference Series in Mathematics. AMS, Prov,1997

[16] Desbrun M, Meyer M. Peter Schroder etal. Implicit fairing of irregular meshes using diffusion and curvature flow[C]// SIGGRAPH 99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques. New York, NY, USA, ACM Press/Addison-Wesley Publishing Co,1999:317-324

[17] Sorkine O, Lipman Y, Daniel C-O, et al. Laplacian surface editing [C]// Proceedings of the Eurographics symposium on Geometry processing. Eurographics Association,2004:179-188

[18] The digital michelangelo project, the standford university[EB]. http://graphics. stanford. edu/data/3Dscanrep/

[19] 龚勋. 基于单张二维图片的三维人脸建模[D]. 成都:西南交通大学,2008