

摘要

在人工智能领域，如何让计算机运用已有的知识库进行逻辑推理和求解问题是一个很重要的研究方向。非单调逻辑被认为是该研究方向的一类重要的知识表示语言。随着理论研究的成熟与相对高效求解器的出现，越来越多的研究者将回答集编程(ASP)作为具有非单调推理能力的知识表示与形式推理的一个工具，同时将其应用到诸多实际领域。然而，求解器的效率仍然没能完全满足人们的需求，这也成为是影响其推广的一个主要瓶颈。因此，研究与实现更高效的逻辑程序的求解器具有非常重要的理论与应用价值。

Lin在2002年首次提出了正规逻辑程序的环和环公式的概念，将回答集求解归约为求解命题逻辑公式的模型。环和环公式在回答集的求解中扮演着很重要的角色。然而，在最坏的情况下，环的数目是会指数爆炸的。2005年，Gebser定义了基本环(elementary loops)，并且指出，使用基本环已经足以完成回答集的求解。基本环的出现，极大地减少了回答集求解所用到的环的数量，推进了求解器的发展。

本文在环和环公式概念的基础上，对基本环进行深入研究，发现并非所有的基本环对于回答集的求解都是必须的，以此为基础，进一步对环的定义加入了限制，提出了特征环(proper loops)。特征环是基本环的子集。通过把特征环应用到特殊形式的环公式里面，我们发现，特征环同样也足以完成回答集的求解。本文的主要贡献和创新有以下几点：

第一，对于正规逻辑程序，提出了一个多项式时间复杂度的算法，用于识别逻辑程序的特征环。对于大部分的逻辑程序，识别所有的特征环比基本环要高效得多，并且，对于依赖图符合特定结构特点的逻辑程序，我们只需要提取小部分的特征环，即可完成回答集的求解。

第二，将特征环拓展到析取逻辑程序。和正规逻辑程序不一样的是，识别析取逻辑程序的特征环的时间复杂是coNP-complete，这是当今计算机无法接受的。针对这一问题，本文引入了一个弱化版本的特征环，并且给出了一个多项式时间复杂度的识别算法。

特征环在基本环的基础上，进一步减少了回答集求解所用到的环的数量，对于求解器效率的提升有着重要的意义。

关键词：回答集编程，正规逻辑程序，析取逻辑程序，环公式

Abstract

In the field of artificial intelligence, making computers to use an existing knowledge base in reasoning and problem solving is one of the most important research area. Non-monotonic logic is considered as an important class of knowledge representation languages targeting on this problem. With the development of the theory and the presence of efficient solvers, more and more researchers consider Answer Set Programming(ASP) as a general knowledge representation and reasoning tool with non-monotonic reasoning ability, and apply it to many practical area. However, the efficiency of these ASP solvers still can't meet people's needs, which is the bottleneck for more applications of ASP. As a result, research and implementation of more efficient ASP solvers for logic programs is of great theoretical and practical value.

The notions of loops and loop formulas for normal logic programs were first proposed by Lin in 2002, making the computation of answer set reduce to finding models of propositional logic. Loops and loop formulas play an important role in answer set computation. However, there will be an exponential number of loops in the worst case. In 2005, Gebser and Schaub showed that not all loops are necessary for selecting the answer sets among the models of a program, they introduced the subclass elementary loops, which greatly decrease the number of loops needed in answer set computation and promote the development of ASP solver.

The main contribution and innovation of this paper are as follows:

1. We introduce a subclass proper loops of elementary loops for normal logic programs, and show that a proper loop can be recognized in polynomial time. For certain programs, identifying all proper loops is more efficient than that of all elementary loops.

2. We extend the notion of proper loops for disjunctive logic programs. Different from normal logic programs, the computational complexities of recognizing proper loops for disjunctive logic programs is coNP-complete. To address this problem, we introduce weaker version of proper loops and provide polynomial time algorithm for identifying it.

Proper loops further reduce the number of loops needed in answer set computation, which will make great contribution to the development of ASP solver.

Key Words: ASP, normal logic programs, disjunctive logic programs, loop formulas

目 录

摘 要	I
Abstract	II
目 录.....	IV
第一章 引言	1
1.1 研究背景	1
1.2 研究现状	2
1.3 本文的工作	4
1.4 本文的安排	4
第二章 预备知识.....	6
2.1 命题逻辑	6
2.2 回答集逻辑程序	7
2.3 环与环公式	11
2.4 传统的基本环	16
第三章 全新的基本环(Elementary Loops).....	19
3.1 正规逻辑程序的基本环	19
3.2 析取逻辑程序的基本环	22
3.3 弱基本环	26
3.4 <i>HWEF</i> 程序	28
3.5 本章小结	31
第四章 特征环(Proper Loops).....	32
4.1 正规逻辑程序的特征环	32
4.2 析取逻辑程序的特征环	37
4.3 <i>HPF</i> 程序	42
4.4 弱特征环	42
4.5 <i>HWPF</i> 程序	44
4.6 本章小结	44

第五章 实验过程与结果分析 46

5.1 实验概述 46

第六章 总结与展望..... 47

6.1 本文的研究工作总结 47

参考文献 48

第1章 引言

本章分为四个小节：首先介绍非单调逻辑、逻辑程序和ASP背景，然后介绍了目前的各种ASP求解器的原理与国内外的研究现状，从而引出了本文的研究重点和意义，最后对本文的组织结构安排进行了概述。

1.1 研究背景

在人工智能领域，知识表示与推理(knowledge representation and reasoning, KR) [1]是一个重要的研究方向。知识表示与推理的主要目标为存储知识，让计算机能够处理，以达到人类的智慧。常见的知识表示方法有语义网(semantic nets)[2]、规则(rules)和本体(ontologies)[3]等。这里的知识包括常识(common sense)，所谓的常识，指的是人生活在社会中所应该具备的基本知识，特别指总所周知的知识。

早在1958年，图灵奖获得者、“人工智能之父”之一的John McCarthy就在考虑可以处理常识的人工智能系统，他在“具有常识的程序”一文中提到，该系统应该可以接受用户的建议并且能根据这些建议来改进自身的性能和行为[4]。这一系统的构建理论指出了常识推理是人工智能的关键，标志着他向“常识推理”的难题开始宣战，同时也拉开了常识知识表示和推理的研究序幕。1959年，McCarthy与Hayes提出，知识表示与常识推理应该要分离开来，即分为认识论与启发式两部分[5]。然而，在常识推理中，知识库加入了新的知识后，原有的推论往往会被推翻。换句话说，知识库的推论不会随着知识的增长而增长，是非单调(non-monotonic)[6]的。而经典逻辑则是单调的，无法处理非单调的推理问题。因此，研究者便开始了新的逻辑形式的研究，伴随着而诞生的比较著名的非单调逻辑有McCarthy的限定理论(circumscription)[7]、Reiter的缺省逻辑(default logic)[8]和McDermott的非单调模态逻辑(non-monotonic modal logic)[6]。

另一方面，随着人工智能各领域知识理论的发展，六十年代末到七十年代

初，逻辑程序的概念慢慢地形成。1965年，Robinson提出了非常重要的消除原理(resolution principle)[9]。1967年，Green将逻辑当做一个带有自动推导和构造性逻辑的表示语言[10]。在这些理论的推动下，1972年，Colmerauer等人实现了第一个逻辑程序设计语言Prolog[11]。对传统的程序设计来说，算法的逻辑意义往往被程序复杂的控制成分所掩盖，使得程序的正确性难以得到证明。逻辑程序设计的主要思想就是把逻辑和控制分开。Kowalski提到，算法=逻辑+控制[12]。其中，逻辑部分刻画了算法要实现的功能，控制部分刻画了如何实现这些功能。作为程序员，只需要关心算法的逻辑部分，而算法的控制部分则留给逻辑程序解释系统去完成。传统的逻辑程序是基于正程序的，即程序的规则中不会出现任何形式的否定(negation)。然而，不使用否定去描述实际问题是很困难的。为了解决这一难题，失败即否定(negation as failure)的概念就被研究者提出来了。为了刻画这一性质，各种语义先后被提出，包括Clark完备(Clark completion)[13]概念、Reiter的闭世界假设(Closed World Assumption, CMA)[14]和Van Gelder的良序(well-founded)语义[15]。1988年，Lifschitz等人提出了稳定模型语义(stable models semantics)[16]，首次利用非单调推理领域的成果成功解释了失败即否定，并将其推广到正规逻辑程序中。1991年，他们又将稳定模型语义拓展到析取逻辑程序[17]。稳定模型语义不仅仅可以解释逻辑程序中的失败即否定，还与非单调推理中的很多工作密切联系，从而被认可为一个实用的非单调推理工具和可以表达常识知识的知识表示语言。正因为稳定模型语义有着这些良好的性质，越来越多的研究者关注这个方向，同时也推进了该语义的逻辑程序设计的发展。这一全新的研究领域被研究者们称为回答集程序设计(Answer Set Programming, ASP)[18]。

1.2 研究现状

近十几年来，随着ASP的快速发展，先后出现了很多求解器(ASP solver)。由于计算ASP程序的回答集的复杂性为NP-complete，大部分求解器都是通过搜索的方式查找回答集。针对普通逻辑程序的ASP求解器主要分为两大类，

一类是基于DPLL算法(Davis-Putnam-Logemann-Loveland procedure)[19]的, 包括DLV[20]、smodels[21]和clasp[22]等。另一类则是基于SAT求解器的, 包括ASSAT[23]和cmodels[24]等。

求解器的发展离不开理论的支持。2002年, Lin与Zhao首次提出了正规逻辑程序的环(loops)和环公式(loop formulas)的概念[23], 将回答集的求解归约为求解命题逻辑公式的模型, 即SAT问题, 并使用SAT求解器进行回答集的求解。Lin-Zhao规约理论的核心在于, 通过引入逻辑程序的环公式, 对于逻辑程序的正依赖图(positive dependency graph)中的每一个环, 添加一个相应的环公式到原逻辑程序对应的克拉克完备(Clark completion)[13]中, 从而得到模型与原逻辑程序的回答集一一对应的命题逻辑公式集。不久之后, Lin-Zhao规约理论被Lifschitz等人拓展到析取逻辑程序[25]。这些理论的提出, 保证了基于SAT求解器的回答集求解器的正确性(correctness)和完备性(completeness), 极大地推进了求解器的发展。

然而, 通常情况下, ASP程序的环的数目可能出现指数爆炸[26]。2005年, Gebser等人发现, 并非所有环对于从正规逻辑程序的模型中挑选回答集都是必须的。他们提出了基本环(elementary loops)[27]的概念, 并且在仅考虑基本环的情况下, 重新定义了Lin-Zhao的环公式理论。2011年, Gebser等人[28]把基本环的概念拓展到析取逻辑程序, 并指出, 只利用这些基本环, 已经足以完成从析取逻辑程序的模型中选取回答集这一操作。他们还提出了一种名为HEF(Head-Elementary-loop-Free)的逻辑程序类别, 并指出了, 这类析取逻辑程序与HCF(Head-Cycle-Free)逻辑程序[29]一样, 可以通过把规则头部的原子移动到规则体部, 在多项式时间内转换为与其等价的正规逻辑程序。Ji等人[30]在2013年通过实验观察到, 对于特定的逻辑程序, 如果其环都是最多只有一个外部支持(external support), 那么使用环公式理论进行转化后, 可以显著地提升回答集的计算效率。

1.3 本文的工作

影响ASP的推广的最大问题是其求解器的效率。本文的主要关注点的是提升求解器的效率。总的来说，本文的主要工作包括：

第一，深入研究了基本环及其自低向上的识别算法[27]，本文提出了基本环的另一种定义，并给出一种自顶向下的识别算法，该算法的时间复杂性和Gebser-Schaub算法是一样的。

第二，对于正规逻辑程序，本文提出了特征环(proper loops)的概念，并证明了特征环已经足以完成回答集的求解，同时还给出一个多项式时间复杂度的算法，用于识别逻辑程序的特征环。此外，本文还证明了，对于依赖图符合特定结构特点的逻辑程序，我们只需要提取小部分的特征环，即可完成回答集的求解。这一结论，不仅仅解释了Ji等人的观察结果[30]，还引导我们想出了求解所有特征环的算法。

第三，本文将特征环的概念拓展到析取逻辑程序。和正规逻辑程序不一样的是，识别析取逻辑程序的特征环的时间复杂是coNP-complete，这是当今计算机所无法接受的。针对这一问题，本文介绍了一个弱化版本的特征环，并且给出了一个多项式时间复杂度的识别算法。

第四，通过实验，对比了正规逻辑程序的基本环与特征环的数量和计算效率以及析取逻辑程序的各种环的数量，进一步说明了特征环的优越性。

1.4 本文的安排

本文的章节安排如下：

第1章，主要介绍了本文的研究背景、现阶段国内外的研究状况以及本文的主要工作。

第2章，详细介绍了与本文相关的预备知识，包括命题逻辑、回答集编程、环公式和基本环。

第3章，提出了基本环的另一种定义，并给出了自顶向下的识别算法。针对析

取逻辑程序的基本环的识别效率低的问题，本文提出了弱基本环的概念，并基于这些概念提出了 *HWEF* 程序。

第4章，首先针对正规逻辑程序，详细地介绍了特征环及其性质，并给出了识别特征环以及计算程序的所有特征环的算法，然后把特征环扩展到析取逻辑程序，针对计算复杂度太高的难题，提出了适用于大部分逻辑程序但时间复杂度较低的弱特征环。

第5章，主要介绍了两个对比实验。第一个实验比较正规逻辑程序下，基本环与特征环的数量和计算效率；第二个实验比较析取逻辑程序下，各种环的数量，包括基本环、特征环及其弱化版本。

第6章，主要对本文的工作进行了总结，指出本文未完成的工作，并对未来下一步的研究工作进行了展望。

第2章 预备知识

本章主要介绍本文工作的理论基础，并给出后续章节将使用的一些性质和已有的结果。第1节介绍了经典命题逻辑的相关知识，这是后续章节的基础；第2节介绍了逻辑程序的语法和语义，从而引出回答集的概念；第3节从逻辑程序的依赖图出发，介绍了环和环公式的概念及其在回答集求解中的意义；第4节介绍了基本环(elementary loops)的概念及其性质。

2.1 命题逻辑

命题逻辑是数理逻辑的一部分，命题逻辑只包含一部分的逻辑形式和规律[31]。命题(proposition)是非真即假的陈述句，比如2是质数。简单命题(或原子命题)为简单陈述句，它不能分解成更简单的句子，一般我们用英文字母 p, q, r 等表示。使用联结词，简单命题可以联结成复合命题。命题逻辑主要就是研究复合命题。

命题逻辑的形式语言的符号表通常包括三类逻辑符号：

1. 命题符号，通常使用小写英文字母表示；
2. 联结符号，包括 \neg (否定)、 \wedge (合取)、 \vee (析取)、 \rightarrow (蕴含)和 \leftrightarrow (等价于)；
3. 标点符号，包括“(”、“)”。

下面，我们将给出命题逻辑公式各类范式的定义和以及相关的定理。这些知识点主要来源于文献[31, 32]。

定义 2.1 (否定式)： 命题变量的否定称为命题的否定式。

例 2.1： $\neg p$ 为 p 的否定式。

定义 2.2 (文字)： 命题变量及其否定称为文字(literal)。

例 2.2： $p, \neg p, q, \neg q$ 都是文字，而 $p \vee q, p \wedge q, p \rightarrow q$ 都不是文字。

定义 2.3 (简单析取式): 仅由有限个文字构成的析取式称为简单析取式。

例 2.3: $p, p \vee q, \neg p \vee q \vee r$ 都是简单析取式, 而 $p \vee q, \neg(p \vee q), p \wedge q \vee r$ 都不是简单析取式。

定义 2.4 (简单合取式): 仅由有限个文字构成的合取式称为简单合取式。

例 2.4: $p, p \wedge q, \neg p \wedge q \wedge \neg r$ 都是简单合取式, 而 $p \vee q, \neg(p \vee q), p \wedge q \vee r$ 都不是简单合取式。

定理 2.1: 简单析取式是重言式, 当且仅当它同时含有一个命题变量及其否定; 简单合取式是矛盾式, 当且仅当它同时含有一个命题变量及其否定。

定义 2.5 (析取范式): 仅由有限个简单合取式构成的析取式称为析取范式(disjunctive normal form, DNF)。

例 2.5: $p, p \vee q, (p \wedge q) \vee r$ 都是析取范式, 而 $(p \vee q) \wedge r, p \wedge q, p \rightarrow q$ 都不是析取范式。

定义 2.6 (合取范式): 仅由有限个简单析取式构成的合取式称为合取范式(conjunctive normal form, CNF)。

例 2.6: $p, p \wedge q, (p \vee q) \wedge (r \vee q)$ 都是合取范式, 而 $(p \wedge q) \vee r, p \vee q, p \rightarrow q$ 都不是合取范式。

定理 2.2: 析取范式为矛盾式, 当且仅当构成它的每一个简单合取式都是矛盾式; 合取范式为重言式, 当且仅当构成它的每一个简单析取式都是重言式。

定理 2.3: 任何命题都存在着与之等值的析取范式和合取范式。

2.2 回答集逻辑程序

本节我们将介绍逻辑程序。本文的关注点是完全被例化(fully grounded)的回答集逻辑程序。

2.2.1 正规逻辑程序

定义 2.7 (正规逻辑程序): 普通规则(normal rule)的有限集合称为正规逻辑程序(normal logic program, NLP)。一个普通规则具有如下形式:

$$H \leftarrow a_1, \dots, a_m, \text{not } a_{m+1}, \dots, \text{not } a_n.$$

其中, $0 \leq m \leq n$, a_1, \dots, a_n 是原子(atom), not 表示失败即否定, H 为一个原子或者空。若 H 为一个原子, 则此规则为一般规则(proper rule); 若 H 为空, 则此规则为约束(constraint)。如果 $m = n = 0$, 则此规则为事实(fact)。

普通规则常常也会被写成如下形式:

$$\text{head}(r) \leftarrow \text{body}(r).$$

其中, $\text{head}(r) = H$ 称为规则的头部, $\text{body}(r) = \text{body}^+(r) \wedge \text{body}^-(r)$ 称为规则的体部, $\text{body}^+(r) = a_1, \dots, a_m$, $\text{body}^-(r) = \text{not } a_{m+1} \wedge \text{not } a_n$, 同时我们会将 $\text{head}(r)$ 、 $\text{body}^+(r)$ 和 $\text{body}^-(r)$ 看作是它们各自对应的原子的集合。

给定一个规则的集合 R , $\text{head}(R) = \bigcup_{r \in R} \text{head}(r)$ 表示 R 中所有规则的头部出现的原子的集合。

给定一个正规逻辑程序 P , $\text{Atoms}(P)$ 表示 P 中出现的所有原子的集合。 $\text{Lit}(P)$ 表示有 $\text{Atoms}(P)$ 构成的文字的集合, 即:

$$\text{Lit}(P) = \text{Atoms}(P) \cup \{\neg a \mid a \in \text{Atoms}(P)\}$$

给定文字 l , 它的补(complement)记为 \bar{l} 。若 l 为原子 a , 则 l 的补为 $\neg a$; 若 l 为 $\neg a$, 则 l 的补为 a 。对于任意文字集合 L , $\bar{L} = \{\bar{l} \mid l \in L\}$ 。

例 2.7: 考虑以下的程序 P_1 :

$$p \rightarrow .$$

$$p \rightarrow r.$$

$$q \rightarrow r.$$

$$r \rightarrow p.$$

$$r \rightarrow q.$$

则 P_1 为正规逻辑程序。

2.2.2 析取逻辑程序

定义 2.8 (析取逻辑程序): 析取规则(disjunctive rule)的有限集合称为析取逻辑程序(disjunctive logic program, DLP)。一个析取规则具有如下形式:

$$a_1 \vee \dots \vee a_k \leftarrow a_{k+1}, \dots, a_m, \text{not } a_{m+1}, \dots, \text{not } a_n.$$

其中, $1 \leq k \leq m \leq n$, a_1, \dots, a_n 为原子, 即正文字。若 $k = 1$, 则为普通规则。类似地, 我们定义 $\text{head}(r) = \{a_1, \dots, a_k\}$, $\text{body}^+(r) = \{a_{k+1}, \dots, a_m\}$, $\text{body}^-(r) = \{a_{m+1}, \dots, a_n\}$ 。

例 2.8: 考虑以下的程序 P_2 :

$$p \vee q \leftarrow r.$$

$$r \leftarrow p.$$

$$r \leftarrow q.$$

$$p \leftarrow .$$

则 P_2 为析取逻辑程序。

2.2.3 逻辑程序的回答集

现在, 我们介绍逻辑程序的回答集[16]。

定义 2.9 (GL规约[16]): 给定一个不含约束的正规逻辑程序 P 和原子集合 S , P 基于 S 的 GL 规约(Gelfond-Lifschitz reduction), 记为 P^S , 是对 P 做以下操作所得到的程序:

1. 删除所有体部存在 $\text{not } q$ 的规则, 其中, $q \in S$;
2. 删除剩下的规则中的所有负文字;

对于任意的原子集合 S , 其对应的 P^S 不含任何形式的负文字。所以, P^S 只有唯一的最小模型(model), 记为 $\Gamma(P^S)$ 。

定义 2.10 (不含约束的程序的回答集): 给定一个不含约束的正规逻辑程序 P , 原子集合 S 是 P 的一个回答集当且仅当 $S = \Gamma(P^S)$ 。

更一般的情况是, 逻辑程序 P 中是含有约束的。

定义 2.11 (正规逻辑程序的回答集): 给定一个正规逻辑程序 P , 记其去掉约束后的程序为 P' , 原子集合 S 是 P 的一个回答集, 当且仅当, S 是 P' 的回答集且 S 满足 P 中的所有约束。

与正规逻辑程序不同, 由于析取逻辑程序的头部的原子数可以大于1个, 所以经过 GL 规约后的程序的最小模型可能有多个。

定义 2.12 (析取逻辑程序的回答集): 给定一个析取逻辑程序 P 和原子集合 S , 记 P 去掉约束后的程序为 P' , P' 的最小模型的集合为 $\Gamma(P^S)$ 。 S 是 P 的回答集, 当且仅当 $S \in \Gamma(P^S)$ 且 S 满足 P 中的所有约束。

2.2.4 逻辑程序的补全(completion)

Clark[13]在1978年通过将逻辑程序翻译为经典逻辑的公式来给出其语义。

定义 2.13 (补全): 给定一个逻辑程序 P , 其补全 $Comp(P)$ 是 P 的约束和 P 的克拉克补全(Clark completion)的并集[23]。它包括以下子句:

1. 对于 $p \in Atoms(P)$, 令 $p \leftarrow G_1, \dots, p \leftarrow G_n$ 为 P 中与 p 相关的规则, 则 $p \equiv G_1 \vee \dots \vee G_n$ 属于 $Comp(P)$ 。特别地, 如果 $n = 0$, 则 $p \equiv \text{false}$, 等价于 $\neg p$ 。
2. 对于约束 $\leftarrow G$, 则 $\neg G$ 属于 $Comp(P)$ 。

例 2.9: 给定程序 P_3 :

$$a \leftarrow b, c, \text{not } d.$$

$$a \leftarrow b, \text{not } c, \text{not } d.$$

$$\leftarrow b, c, \text{not } d.$$

该程序的补全为: $\text{Comp}(P_3) = \{a \equiv (b \wedge c \wedge \neg d) \vee (b \wedge \neg c \wedge \neg d), \neg b, \neg c, \neg d, \neg(b \wedge c \wedge \neg d)\}$ 。

2.2.5 析取逻辑程序到正规逻辑程序的转换

析取逻辑程序和正规逻辑程序的区别在于头部原子的个数，一个自然的问题是，是否存在一种转换使得析取逻辑程序可以转化为正规逻辑程序。Gelfond等人[17]提出了一个转换，通过把析取逻辑程序 P 头部的原子移动(shifting)到体部，把析取逻辑程序转化为正规逻辑程序，记为 $sh(P)$ 。其具体操作每条析取规则替换成如下的形式：

$$a_i \leftarrow \text{not } a_1, \dots, \text{not } a_{i-1}, \text{not } a_{i+1}, \dots, \text{not } a_k, a_{k+1}, \dots, a_m, \text{not } a_{m+1}, \dots, \text{not } a_n. (1 \leq i \leq k)$$

直观上，我们可以看出， $sh(P)$ 的每个回答集同时也是 P 的回答集。但是，反过来就不一定成立了。后来，Ben-Eliyahu和Dechter提出了一种名为Head-Cycle-Free(HCF)[29]的析取逻辑程序类型，并且证明了HCF程序 P 的回答集和其 $sh(P)$ 的回答集一一对应。

定义 2.14 (HCF程序): 给定一个析取逻辑程序 P ，对于 P 的每一个环 L 和每条规则 r ，如果 $|\text{head}(r) \cap L| \leq 1$ ，那么该程序称为Head-Cycle-Free(HCF)程序。

环的概念将在下一节中给出。

2.3 环与环公式

回答集逻辑程序和命题逻辑的关系是很密切的。我们甚至可以把逻辑程序中的每一条规则看成是命题逻辑中的一个子句。Lin和Zhao[23]证明，只要加入环

公式(loop formulas), 原逻辑程序的回答集就可以和对应的命题的模型一一对应。下面我们给出环和环公式的定义。

环和环公式的概念是基于正依赖图的, 首先我们给出正依赖图的定义, 本节的所有定义都是针对析取逻辑程序的, 正规逻辑程序可以看成是析取逻辑程序的特例。

定义 2.15 (正依赖图[23]): 给定一个析取逻辑程序 P , 其正依赖图(positive dependency graph), 记为 G_P , 是以 P 中原子为顶点的有向图。其中, 两原子之间存在从 p 到 q 的有向边, 当且仅当, 存在 P 中的规则 r , 使得 $p \in head(r)$ 且 $q \in body^+(r)$ 。

例 2.10: 程序 P_1 和 P_2 的正依赖图是一样的, 如图2.1所示。

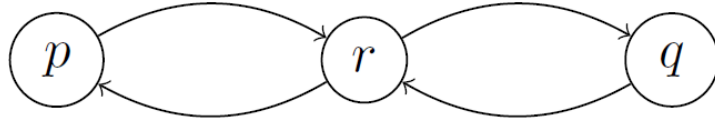


图 2.1: P_1 和 P_2 的正依赖图

定义 2.16 (子图): 记有向图 $G = (V, E)$, 其中, V 为顶点的集合, E 为有向边的集合。令 $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E$, 则我们称 G' 为 G 的子图。

定义 2.17 (诱导子图): 记有向图 $G = (V, E)$, 其中, V 为顶点的集合, E 为有向边的集合。令 $G' = (V', E')$, $V' \subseteq V$, $E' = \{(u, v) | u, v \in V', (u, v) \in E\}$, 则我们称 G' 为 G 的诱导子图。

注意: 对于 V' , 只要在 G 中有边, 那么在 G' 中同样应该有边。

定义 2.18 (强连通分量[33]): 记有向图 $G = (V, E)$, 其中, V 为顶点的集合, E 为有向边的集合。 G 的一个强连通分量就是一个最大的顶点集合 $C \subseteq V$, 对于 C 中的每一对顶点 u 和 v , 存在从 u 到 v 的路径以及从 v 到 u 的路径, 即顶点 u 和 v 相互可达。

定义 2.19 (环): 给定原子集合 L , 如果 L 中的任意原子 p 和 q 在程序 P 的正依赖图 G_P 中存在一条路径, 并且路径上的所有顶点 $r \in L$, 那么我们称 L 为程序 P 的环(loop)。特别地, 任意单原子集合都为环。

由定义2.18和定义2.19, 我们可以知道, 强连通分量和环是等价的。

例 2.11: 程序 P_2 有6个环: $\{p\}, \{r\}, \{q\}, \{p, r\}, \{q, r\}, \{p, q, r\}$ 。

定义 2.20 (子环): 给定环 L , 环 L' 是 L 的子环当且仅当 $L' \subset L$ 且 L' 对应的子图是一个强连通分量。

例 2.12: 程序 P_2 中, 环 $\{p, r\}$ 的子环有 $\{p\}$ 和 $\{r\}$ 。

给定逻辑程序 P 和环 L , 我们定义如下两种规则的集合:

$$R^+(L, P) = \{r | r \in P \text{ and } \text{head}(r) \cap L \neq \emptyset \text{ and } \text{body}^+(r) \cap L \neq \emptyset\} \quad (2.1)$$

$$R^-(L, P) = \{r | r \in P \text{ and } \text{head}(r) \cap L \neq \emptyset \text{ and } \text{body}^+(r) \cap L = \emptyset\} \quad (2.2)$$

一般地, 我们会把 $R^+(L, P)$ 简写成 $R^+(L)$, 把 $R^-(L, P)$ 简写成 $R^-(L)$ 。显然, 这两个集合是没有交集的。直观上看, $R^+(L)$ 表示环里面的公式, $R^-(L)$ 表示可以推出环中原子的公式。所以, 我们把 $R^+(L)$ 称为内部支持(internal support)的集合, 把 $R^-(L)$ 称为外部支持(external support)的集合。需要注意的是, 外部支持的概念并不针对环, 我们可以把环替换成任意原子集合。

例 2.13: 考虑如下正规逻辑程序 P_4 :

$a \leftarrow b.$

$b \leftarrow a.$

$a \leftarrow \text{not } c.$

$c \leftarrow d.$

$d \leftarrow c.$

$$c \leftarrow \text{not } d.$$

则，该程序有两个环： $L_1 = \{a, b\}$ 和 $L_2 = \{c, d\}$ 。对于这两个环，我们有：

$$R^+(L_1) = \{a \leftarrow b. b \leftarrow a.\}$$

$$R^-(L_1) = \{a \leftarrow \text{not } c.\}$$

$$R^+(L_2) = \{c \leftarrow d. d \leftarrow c.\}$$

$$R^-(L_2) = \{c \leftarrow \text{not } a.\}$$

可以观察到， $R^+(L_1)$ 和 $R^+(L_2)$ 的回答集都为 \emptyset 。事实上，对于任意逻辑程序 P 和环 L ， \emptyset 是 $R^+(L)$ 的唯一回答集。因此，环里面的原子不可能属于任何回答集，除非有额外的公式能推出它，比如 $R^-(L)$ 。基于这些观察，Lin等人[23]提出了正规逻辑程序的环公式的概念。对于正规逻辑程序 P 和环 L ，其环公式为如下形式：

$$\neg\left(\bigvee_{r \in R^-(L)} \text{body}(r)\right) \supset \bigwedge_{p \in L} \neg p \quad (2.3)$$

该环公式的直观意思是，如果环 L 的所有外部支持的体部都为假，那么就不能推出环的任何原子，即环中原子都为假。

定理 2.4： 给定逻辑程序 P ，其补全为 $\text{Comp}(P)$ ，记 LF 为 P 的所有环公式的集合。原子集合 S 是 P 的回答集，当且仅当它是 $\text{Comp}(P) \cup LF$ 的模型。

定理2.4的证明比较复杂，详细过程可以查阅文献[23, 25]。

随着理论的发展，环公式概念已经被拓展到析取逻辑程序，下面我们将介绍析取逻辑程序的环公式。正规逻辑程序的环公式可以看成是其特殊情况。

定义 2.21 (析取环公式)： 对于逻辑程序 P 及其环 L ，对应的析取环公式(disjunctive loop formulas)，记为 $DLF(L, P)$ ，定义为如下形式：

$$\bigvee_{p \in L} \supset \bigvee_{r \in R^-(L)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L} \neg q) \quad (2.4)$$

通常, $DLF(L, P)$ 可以简写为 $DLF(L)$ 。特别地, 对于正规逻辑程序, $DLF(L)$ 为如下形式:

$$\bigvee_{p \in L} \supset \bigvee_{r \in R^-(L)} body(r) \quad (2.5)$$

直观上, 析取环公式的思想是如果环 L 中存在某些原子为真, 那么必然存在某些外部支持的体部为真。

环公式的另一种定义是由 Lifschitz 等人[25]提出, 由于他把 $DLF(L, P)$ 左边的 $\bigvee_{p \in L} p$ 换成 $\bigwedge_{p \in L} p$, 所以我们一般也将其称为合取环公式。

定义 2.22 (合取环公式): 对于逻辑程序 P 及其环 L , 对应的合取环公式(conjunctive loop formulas), 记为 $CLF(L, P)$, 定义为如下形式:

$$\bigwedge_{p \in L} \supset \bigvee_{r \in R^-(L)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q) \quad (2.6)$$

通常, $CLF(L, P)$ 可以简写为 $CLF(L)$ 。特别地, 对于正规逻辑程序, $CLF(L)$ 为如下形式:

$$\bigwedge_{p \in L} \supset \bigvee_{r \in R^-(L)} body(r) \quad (2.7)$$

直观上, 合取环公式的思想是如果环 L 中的所有原子都为真, 那么必然存在某些外部支持的体部为真。

此外, 我们还可以使用蕴含 $\bigvee_{p \in L} p$ 且被 $\bigwedge_{p \in L} p$ 蕴含的命题公式来替换它们, 此时环公式所表达的和析取环公式以及合取环公式都是类似的。比如, 对于任意环 L , 记 F_L 为由环中原子使用合取或者析取组成的公式, 那么环公式又可以定义为 $LF(L, P)$, 它是如下的形式:

$$F_L \supset \bigvee_{r \in R^-(L)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q) \quad (2.8)$$

特别地, 对于正规逻辑程序, $LF(L)$ 为如下形式:

$$F_L \supset \bigvee_{r \in R^-(L)} body(r) \quad (2.9)$$

定理 2.5: 给定程序 P 和原子集合 S , 如果 S 满足 P , 那么以下结论是等价的:

1. S 是 P 的回答集;
2. 对于 P 中的所有环 L , S 满足 $DLF(L, P)$;
3. 对于 P 中的所有环 L , S 满足 $CLF(L, P)$;
4. 对于 P 中的所有环 L , S 满足 $LF(L, P)$;

2.4 传统的基本环

Gebser和Schaub在2005年首次提出了正规逻辑程序的基本环(elementary loops)的概念[27]。2011年, 他们又把基本环拓展到析取逻辑程序[28]。

定义 2.23 (向外的): 给定一个原子集合 X 及其子集 Y , 若存在 $r \in P$, 满足以下的条件:

1. $head(r) \cap Y \neq \emptyset$
2. $head^+(r) \cap (X \setminus Y) \neq \emptyset$
3. $head(r) \cap (X \setminus Y) = \emptyset$
4. $body^+(r) \cap Y = \emptyset$

则称 Y 在 X 里是向外(outbound)的。

定义 2.24 (基本环): 给定环 L 和程序 P , 若 L 的所有非空真子集在 L 里都是向外的, 那么我们称 L 是基本环(elementary loop)。

例 2.14: 例2.8的程序 P_2 有5个基本环: $\{p\}, \{r\}, \{q\}, \{p, r\}, \{q, r\}$ 。

定理 2.6: 给定程序 P 和原子集合 S , 如果 S 满足 P , 那么以下结论与定理2.5的都是等价的:

1. 对于 P 中的所有基本环 L , S 满足 $CLF(L, P)$;
2. 对于 P 中的所有基本环 L , S 满足 $DLF(L, P)$;
3. 对于 P 中的所有基本环 L , S 满足 $LF(L, P)$;

Gebser等人[27]在2005年给出了基本环的识别方法, 该方法基于基本子图的概念。

定义 2.25 (基本子图): 记有向图为 (V, E) , 其中, V 表示节点的集合, E 表示有向边的集合。对于正规逻辑程序 P 和原子集合 X , 我们定义如下计算:

$$EC_P^0(X) = \emptyset$$

$$EC_P^{i+1}(X) = \{(a, b) \mid \text{若存在 } r \in P,$$

$$a = \text{head}(r), a \in X,$$

$$b \in \text{body}^+(r) \cap X,$$

且 $\text{body}^+(r) \cap X$ 的所有原子都属于有向图 (X, EC_P^i) 中的同一个强连通分量\}

$$EC_P(X) = \bigcup_{i \geq 0} EC_P^i(X)$$

有向图 $(X, EC_P(X))$ 称为原子集合 X 关于程序 P 的基本子图(elementary sub-graph)。

定理 2.7: 给定正规逻辑程序 P 和非空原子集合 X , X 是 P 的基本环, 当且仅当 X 关于 P 的基本子图是强连通(strongly connected)。

使用定理2.7的方法识别正规逻辑程序的基本环的时间复杂度为 $O(n^2)$ 。然而, 识别析取逻辑程序的基本环可要复杂很多, 时间复杂度达到coNP-complete[28], 这是当今计算机无法承受的。

定义 2.26 (HEF程序): 给定一个析取逻辑程序 P , 对于 P 的每个基本环 L 和每条规则 r , 如果 $|\text{head}(r) \cap L| \leq 1$, 那么该程序称为HEF(Head-Elementary-loop-Free)程序。

命题 2.1: 给定一个 HEF 程序 P , P 的回答集和 $sh(P)$ 的回答集相同。

相比于普通程序, 识别 HEF 程序的一个基本环要快很多, 这是它的一个很好的性质, 然而, 判断一个程序是否为 HEF 程序的时间复杂度为 $coNP$ -complete[34]。

第3章 全新的基本环(Elementary Loops)

本文在对Gebser等人[27]提出的基本环进行深入研究后，提出了基本环的另一种更为直观的定义。基于这种全新的定义，我们提出一个新的识别算法。该算法的时间复杂性和Gebser等人[27]提出的一样，然而，本文的算法使用的是自顶向下的策略，而Gebser等人[27]的则是自底向上。算法的核心思想来源于环的外部支持和环公式。本章先针对正规逻辑程序讨论基本环的概念，然后，我们把基本环拓展到析取逻辑程序。

3.1 正规逻辑程序的基本环

本节，我们将针对正规逻辑程序，提出基本环的另一种定义，并基于这种定义，给出对应的识别算法。

3.1.1 基本环的定义

在定义基本环之前，我们先给出与合取环公式相关的推论。

推论 3.1: 给定正规逻辑程序 P ， L_1 和 L_2 为 P 的环。如果 $L_1 \subseteq L_2$ 且 $R^-(L_1) \subseteq R^-(L_2)$ ，那么有 $CLF(L_1, P) \supset CLF(L_2, P)$ 。

证明: 记 $A = \bigwedge_{p \in L_1} p$ ， $B = \bigwedge_{p \in L_2} p$ ， $C = \bigvee_{r \in R^-(L_1)} body(r)$ ， $D = \bigvee_{r \in R^-(L_2)} body(r)$ ，则

$$\begin{aligned} CLF(L_1, P) &\supset CLF(L_2, P) \\ \Leftrightarrow (A \rightarrow C) &\rightarrow (B \rightarrow D) \\ \Leftrightarrow (\neg A \vee C) &\rightarrow (\neg B \vee D) \\ \Leftrightarrow (A \wedge \neg C) \vee &(\neg B \vee D) \\ \Leftrightarrow (A \vee \neg B \vee D) &\wedge (\neg C \vee \neg B \vee D) \end{aligned}$$

另一方面，对于 $L_1 \subseteq L_2$ ，有： $\bigwedge_{p \in L_1} p \leftarrow \bigwedge_{p \in L_2} p$ ，即 $B \rightarrow A$ 为真。对于 $R^-(L_1) \subseteq R^-(L_2)$ ，有： $\bigvee_{r \in R^-(L_1)} body(r) \rightarrow \bigvee_{r \in R^-(L_2)} body(r)$ ，即 $C \rightarrow D$ 为

真。所以有：

$$\begin{aligned}
 & true \wedge true \\
 & \Leftrightarrow (B \rightarrow A) \wedge (C \rightarrow D) \\
 & \Leftrightarrow (\neg B \vee A) \wedge (\neg C \vee D) \\
 & \Rightarrow (\neg B \vee A \vee D) \wedge (\neg C \vee \neg B \vee D)
 \end{aligned}$$

由于对于 $p \rightarrow q$ 为真，若 p 为真，则 q 为真，所以得出 $CLF(L_1, P) \supset CLF(L_2, P)$ 为真。 ■

由推论3.1，我们可以知道，对于符合 $L_1 \subseteq L_2$ 且 $R^-(L_1) \subseteq R^-(L_2)$ 的两个环 L_1 和 L_2 ，因为 L_1 的环公式已经可以蕴含 L_2 的环公式了，所以 L_2 的环公式就没有存在的必要。

定义 3.1 (未被抑制的)： 给定程序 P 和环 L ，如果不存在 P 中的其他环 L' ，满足 $L' \subset L$ 和 $R^-(L') \subseteq R^-(L)$ ，那么我们称 L 是未被抑制的(unsubdued)。

定理 3.1： 给定程序 P 和原子集合 S ，如果 S 满足 P ，那么以下结论与定理2.5和定理2.6的结论都是等价的：

1. 对于 P 中的所有未被抑制的环 L ， S 满足 $CLF(L, P)$ 。

命题 3.1： 给定正规逻辑程序 P 和环 L ， L 是 P 的基本环，当且仅当 L 在 P 中是未被抑制的。

证明： 证明过程分为两步，如下所示。

(\Rightarrow)

由于 L 是 P 的基本环，所以 L 的所有非空真子集 L' 都是向外的。因此，必然存在规则 $r \in P$ ，使得 $head(r) \in L'$ ， $body^+(r) \cap (L \setminus L') \neq \emptyset$ ， $body^+(r) \cap L' = \emptyset$ 。显然，此 r 满足 $r \in R^-(L')$ 且 $r \notin R^-(L)$ 。由此可知，不存在其他环 L' ，使得 $L' \subset L$ ， $R^-(L') \subseteq R^-(L)$ 。因此， L 是未被抑制的。

(\Leftarrow)

由于 L 在 P 中是未被抑制的, 所以对于 L 的所有非空真子集 L' , 有 $R^-(L') \not\subseteq R^-(L)$ 。因此, 存在规则 r , 使得 $r \in R^-(L')$ 且 $r \notin R^-(L)$ 。显然, 此 r 满足 $head(r) \in L'$, $body^+(r) \cap (L \setminus L') \neq \emptyset$, $body^+(r) \cap L' = \emptyset$, 即 L' 在 L 中是向外的。因此, L 是基本环。

直观地, 定义2.23的外向概念想要表达的意思是: 原子集合 Y 在 X 中是外向的, 当且仅当存在 $r \in P$, 使得 $r \in R^-(Y)$ 且 $r \notin R^-(X)$ 。令 L 为正规逻辑程序 P 的环, L 的任意非空真子集 L' 是外向的, 当且仅当 $R^-(L') \not\subseteq R^-(L)$ 。

3.1.2 基本环的识别

根据命题3.1, 下面给出正规逻辑程序基本环的识别算法1。该算法从环 L 出发, 考虑其子环的性质。为了得到 L 的子环, 我们的方法是先通过剔除 L 中的一个原子破坏 L 的连通性, 得到其子图, 然后通过求子图的强连通分量得到子环的集合, 对于每个子环 C :

Algorithm 1: 基本环的识别算法 $ElementaryLoop(L, P)$

输入: 正依赖图中的环 L , 正规逻辑程序 P

输出: 若返回 L , 则 L 是基本环; 若返回环 C , 则 L 不是基本环

```

1 for 原子  $a \in L$  do
2    $G^* :=$  原子集合  $L \setminus \{a\}$  在  $G_P$  的诱导子图;
3    $SCC^* := G^*$  的强连通分量的集合;
4   for 强连通分量  $C \in SCC^*$  do
5     if  $R^-(C) \subseteq R^-(L)$  then
6       return  $C$ 
7     else
8        $G_C :=$  原子集合  $C \setminus head(R^-(C) \setminus R^-(L))$  在  $G^*$  的诱导子图;
9        $SCC_C := G_C$  的强连通分量的集合;
10      把所有  $SCC_C$  中所有新的强连通分量加入  $SCC^*$ ;
11 return  $L$ 

```

1. 如果 $R^-(C) \subseteq R^-(L)$, 那么子环 C 不是未被抑制的, 根据命题3.1, L 不是基本环;
2. 如果 $R^-(C) \not\subseteq R^-(L)$, 那么对于任意的 $r \in R^-(C) \setminus R^-(L)$, 有 $r \notin R^-(L)$, 同时 $\text{head}(r)$ 肯定不属于任意非向外的子环 L' 中, 因为如果 $\text{head}(r)$ 属于 L' , 则有 $r \in R^-(L')$ 。因为 L' 是非向外的, 所以有 $R^-(L') \subseteq R^-(L)$, 这样就会得出 $r \in R^-(L)$, 导致矛盾。所以在这种情况下, 我们可以把 $\text{head}(R^-(C) \setminus R^-(L))$ 从当前子图中删除, 从而得到更小的子图和子环, 然后继续进行检测。

对于给定的正规逻辑程序 P 及其环 L , 若 L 是 P 的基本环, 那么算法1将会返回 L ; 否则, 算法1将返回环 C , 其中, C 满足 $C \subset L$ 和 $R^-(C) \subseteq R^-(L)$ 。

与Gebser等人[27]提出的基本环识别算法不同的是, 算法1采取自顶向下的策略, 先大环开始, 逐步深入到小环。注意到, 该算法每次都至少删除子图中的一个原子。所以, 在最坏的情况下, 算法1将迭代 n^2 次, 其中, n 为环 L 中的原子数目。另一方面, 由于图的强连通分量可以在线性时间里面计算出, 所以整个算法的时间复杂性为 $O(n^2)$ 。

3.2 析取逻辑程序的基本环

上一节, 我们从外部支持的角度, 给出了正规逻辑程序的基本环的另一种定义。本节, 我们将把这种基于外部支持的定义拓展到析取逻辑程序。

3.2.1 基本环的定义

令 X, Y 为析取逻辑程序 P 中的原子集合, 我们记 $R_X^-(Y) = \{r | r \in R^-(Y) \text{ and } \text{head}(r) \cap (X \setminus Y) = \emptyset\}$, 特别地, 对于正规逻辑程序, $R_X^-(Y) = R^-(Y)$ 。

推论 3.2: 给定析取逻辑程序 P , L_1 和 L_2 为 P 中的环。如果 $L_1 \subseteq L_2$ 且 $R_{L_2}^-(L_1) \subseteq R^-(L_2)$, 那么有 $CLF(L_1, P) \supset CLF(L_2, P)$ 。

证明： 由 $L_1 \subseteq L_2$ ，得 $\bigwedge_{p \in L_2} p \supset \bigwedge_{p \in L_1} p$ 。对于任意规则 $r \in R^-(L_1)$ ，如果 $r \in R_{L_2}^-(L_1)$ ，此时 $\text{head}(r) \cap (L_2 \setminus L_1) = \emptyset$ ，那么有 $\text{head}(r) \setminus L_1 = \text{head}(r) \setminus L_2$ ，所以此时可得：

$$\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q \supset \text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q$$

如果 $r \notin R_{L_2}^-(L_1)$ ，此时 $\text{head}(r) \cap (L_2 \setminus L_1) \neq \emptyset$ ，那么存在 q ，满足 $q \in \text{head}(r) \setminus L_1$ ， $q \in L_2$ 。注意到，在 $\bigwedge_{p \in L_2}$ 为真的条件下， $\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q$ 为假。所以此时也有：

$$\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q \supset \text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q$$

另一方面，注意到 $R_{L_2}^-(L_1) \subseteq R^-(L_2)$ ，所以在 $\bigwedge_{p \in L_2} p$ 为真的情况下，有：

$$\bigvee_{r \in R^-(L_1)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q) \supset \bigvee_{r \in R^-(L_2)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q)$$

综合上述， $CLF(L_1, P) \supset CLF(L_2, P)$ 成立。 ■

由推论3.2，我们知道 L_1 的合取环公式已经可以蕴含 L_2 的合取环公式，因此 L_2 的合取环公式在回答集的计算中是多余的。

根据基本环的定义，如果环 L 的所有非空真子集在 L 中都是向外的， L 才是程序的基本环。而原子集合 C 在 L 中是向外的，当且仅当存在规则 $r \in P$ ，使得 $r \in R_L^-(C)$ ， $r \notin R^-(L)$ ，当且仅当 $R_L^-(C) \not\subseteq R^-(L)$ 。利用这些已有结论，我们从外部支持的角度，重新给出析取逻辑程序的基本环的概念。

命题 3.2： 给定析取逻辑程序 P ，环 L 是 P 的基本环，当且仅当不存在 L 的非空真子集 L' ，使得 $R_L^-(L') \subseteq R^-(L)$ 。

证明： 证明过程分为两步，如下所示。

(\Rightarrow)

使用反证法，假设存在这样的环 L' ，那么对于任意满足 $\text{head}(r) \cap L' \neq \emptyset$ ， $\text{head}(r) \cap (L \setminus L') = \emptyset$ ， $\text{body}^+(r) \cap L' = \emptyset$ 的规则 r ，即 $r \in R_L^-(L')$ ，由于 $R_L^-(L') \subseteq$

$R^-(L)$, 所以有 $r \in R^-(L)$ 。此时可得: $body^+(r) \cap L = \emptyset$, 因此, $body^+(r) \cap (L \setminus L') = \emptyset$, 即不存在规则 r , 使得 L' 是向外的, 所以 L 不是基本环, 矛盾。

(\Leftarrow)

使用反证法, 假设 L 不是基本环, 那么存在环 L' 不是向外的, 所以存在规则 r , 使得 $head(r) \cap L' \neq \emptyset$, $head(r) \cap (L \setminus L') = \emptyset$, $body^+(r) \cap L' = \emptyset$, $body^+(r) \cap (L \setminus L') \neq \emptyset$, 即 $r \in R_L^-(L')$, $head(r) \cap (L \setminus L') = \emptyset$, $body^+(r) \cap (L \setminus L') \neq \emptyset$ 。由于 $body^+(r) \cap (L \setminus L') \neq \emptyset$, 所以有 $body^+(r) \cap L \neq \emptyset$, 即 $r \notin R^-(L)$, 这和 $R_L^-(L') \subseteq R^-(L)$ 矛盾。 ■

3.2.2 基本环的识别

Gebser等人[28]在2011年证明了识别析取逻辑程序是coNP-complete的, 这是计算机无法承受的。基于命题3.2, 我们给出一个近似算法。该算法能在多项式时间内判断一个环 L 是否属于析取逻辑程序 P 的基本环的一个超集, 记为 $EL^*(P)$ 。

Algorithm 2: 析取逻辑程序基本环的近似识别算法 $ElementaryLoop^*(L, P)$

输入: 正依赖图中的环 L , 析取逻辑程序 P

输出: 若返回环 L , 则可能是基本环; 若返回环 C , 则不是基本环

```

1 for 原子  $a \in L$  do
2    $G^* :=$  原子集合  $L \setminus \{a\}$  在  $G_P$  的诱导子图;
3    $SCC^* := G^*$  的强连通分量的集合;
4   for 强连通分量  $C \in SCC^*$  do
5     if  $R_L^-(C) \subseteq R^-(L)$  then
6       return  $C$ 
7     else
8        $G_C :=$  原子集合  $C \setminus head(R_L^-(C) \setminus R^-(L))$  在  $G^*$  的诱导子图;
9        $SCC_C := G_C$  的强连通分量的集合;
10      把所有  $SCC_C$  中所有新的强连通分量加入  $SCC^*$ ;
11 return  $L$ 

```

算法2的过程和算法1相似, 通过自顶向下的策略, 考虑了环 L 的所有子环。对于每一个子环 C , 有两种情况:

1. 如果 $R_L^-(C) \subseteq R^-(L)$, 那么根据命题3.2, 环 L 不是基本环;
2. 如果 $R_L^-(C) \not\subseteq R^-(L)$, 那么对于任意规则 $r \in R_L^-(C) \setminus R^-(L)$:
 - (a) 若 $|head(r) \cap C| = 1$, 则 $head(r) \cap C$ 不可能在非向外的环 L' 里, 其中, $L' \subset C$ 。否则, 就会有 $r \in R_L^-(L')$, 而由 L' 是非向外的, 有 $R_L^-(L') \subseteq R^-(L)$, 从而得出 $r \in R_L^-(L)$, 矛盾。此时, 可以删除 r ;
 - (b) 若 $|head(r) \cap C| > 1$, 则可能会有 $L' \subseteq C$, 使得 $r \in R^-(L')$, $head(r) \cap (L \setminus L') \neq \emptyset$ 。此时, $L' \cap (head(r) \cap C) \neq \emptyset$, 根据基本环的定义, 不能删除 r 。然而, 为了优化识别时间, 该算法采取了近似的操作, 在这里同样对 r 进行删除。

给定任意析取逻辑程序 P 及其环 L , 算法2要么返回环 L , 要么返回环 C , 其中, $C \subset L$, $R_L^-(C) \subseteq R^-(L)$ 。若返回环 C , 则 L 肯定不是基本环; 若返回 L , 则有可能是基本环, 也有可能不是。

算法2是算法1的一样, 每次迭代至少删除一个原子。在最坏的情况下, 整个算法迭代 n^2 次, n 为环的原子的数目。由于强连通分量可以在线性时间内求得, 所以算法2的时间复杂度为 $O(n^2)$ 。

例 3.1: 考虑环 $L = \{p, q, r\}$ 和析取逻辑程序 P_5 :

$$p \vee q \leftarrow r.$$

$$p \vee r \leftarrow q.$$

$$q \vee r \leftarrow p.$$

$$r \leftarrow .$$

$ElementaryLoop^*(L, P_5)$ 返回 L , 事实上, L 并不是 P 的基本环。因为对于 $L' = \{p, q\}$:

$$R_L^-(L') = \emptyset$$

$$R^-(L) = \{r \leftarrow .\}$$

所以 $R_L^-(L') \subseteq R^-(L)$ 。

记 $EL(P)$ 为程序 P 的基本环的集合, $EL^*(P)$ 为使用算法2求得的环的集合, 即 $EL^*(P) = \{L | L \text{ 是程序 } P \text{ 的环且 } ElementaryLoop^*(L, P) \text{ 返回 } L\}$ 。那么, 对于任意析取逻辑程序 P , 有 $EL(P) \subseteq EL^*(P)$ 。特别地, 如果 P 为正规逻辑程序, 那么 $EL(P) = EL^*(P)$ 。另外, HEF 程序虽然属于析取逻辑程序, 但是由于它有性质 $|head(r) \cap L| \leq 1$, 所以 $EL(P) = EL^*(P)$ 。

3.3 弱基本环

尽管基本环和 HEF 程序的特性在回答集计算中很有用, 但是识别它们的时间复杂度太高了。本节, 我们将提出基本环的一个超集, 同时, 给出多项式时间复杂度的识别算法。

推论 3.3: 给定析取逻辑程序 P , L_1 和 L_2 为 P 中的环。如果 $L_1 \subseteq L_2$ 且 $R^-(L_1) \subseteq R^-(L_2)$, 那么有 $CLF(L_1, P) \supset CLF(L_2, P)$ 。

证明: 对于环 L_1 , 显然我们有 $R_{L_2}^-(L_1) \subseteq R^-(L_1)$ 。又因为 $R^-(L_1) \subseteq R^-(L_2)$, 所以有 $R_{L_2}^-(L_1) \subseteq R^-(L_1) \subseteq R^-(L_2)$, 因此可得 $R_{L_2}^-(L_1) \subseteq R^-(L_2)$ 。根据推论3.2, 则有 $CLF(L_1, P) \supset CLF(L_2, P)$ 。 ■

现在, 我们可以基于推论3.3定义一个弱化版本的基本环, 如下:

定义 3.2 (弱基本环): 给定析取逻辑程序 P , L 为 P 的环。如果不存在 L 的非空真子集 C , 使得 $R^-(C) \subseteq R^-(L)$, 那么, 我们称 L 是 P 的弱基本环(weak elementary loop)。

推论 3.4: 给定析取逻辑程序 P , L 是 P 的环。如果 L 是 P 的基本环, 那么 L 也是 P 的弱基本环。

证明： 因为 L 是 P 的基本环，根据命题3.2，对于 L 的任意非空真子集 C ， $R_L^-(C) \not\subseteq R^-(L)$ ，即存在规则 $r \in R_L^-(C)$ 且 $r \notin R^-(L)$ 。由于 $R_L^-(C) \subseteq R^-(C)$ ，所以 $r \in R^-(C)$ ，由此可得 $R^-(C) \not\subseteq R^-(L)$ 。根据定义3.2， L 是 P 的弱基本环。 ■

例 3.2： 对于例2.8的程序 P_2 ，环 $L = \{p, q, r\}$ 是弱基本环，但不是基本环。

判断一个环是否是程序的弱基本环是很容易的。我们可以通过把算法2中的所有 $R_L^-(C)$ 替换成 $R^-(C)$ ，得到算法3，记为 $WEL(L, P)$ 。对于给定的程序 P 及其环 L ，若 L 是 P 的弱基本环，那么算法3将会返回 L ；否则，算法3将返回环 C ，其中， C 满足 $C \subset L$ 且 $R^-(C) \subseteq R^-(L)$ 。显然，算法3跟算法2一样，时间复杂度为 $O(n^2)$ 。

Algorithm 3: 析取逻辑程序弱基本环的识别算法 $WEL(L, P)$

输入: 正依赖图中的环 L ，析取逻辑程序 P

输出: 若返回环 L ，则 L 是弱基本环；若返回环 C ，则 L 不是弱基本环

```

1 for 原子 $a \in L$  do
2    $G^* :=$  原子集合 $L \setminus \{a\}$ 在 $G_P$ 的诱导子图;
3    $SCC^* := G^*$ 的强连通分量的集合;
4   for 强连通分量 $C \in SCC^*$  do
5     if  $R^-(C) \subseteq R^-(L)$  then
6       return  $C$ 
7     else
8        $G_C :=$  原子集合 $C \setminus head(R^-(C) \setminus R^-(L))$ 在 $G^*$ 的诱导子图;
9        $SCC_C := G_C$ 的强连通分量的集合;
10      把所有 $SCC_C$ 中所有新的强连通分量加入 $SCC^*$ ;
11 return  $L$ 

```

记算程序 P 的所有弱基本环为 $WEL(P)$ 。对于任意的析取逻辑程序 P ，我们有 $EL(P) \subseteq EL^*(P) \subseteq WEL(P)$ ，特别地，如果 P 是正规逻辑程序，那么 $EL(P) = EL^*(P) = WEL(P)$ 。

命题 3.3： 如果 P 是HCF程序，那么 $EL(P) = EL^*(P) = WEL(P)$ 。

证明： 因为 P 是 HCF 程序，所以不存在任意的规则 r ，环 L 及其子环 L' ，使得 $head(r) \cap L' \neq \emptyset$ ， $head(r) \cap (L \setminus L') \neq \emptyset$ ，否则就会有 $|head(r) \cap L| > 1$ 。所以对于任意的环 L 及其子环 L' ，都会有 $R_L^-(L') = R^-(L')$ 。由此可以得出： $EL(P) = EL^*(P) = WEL(P)$ 。 ■

3.4 HWEF程序

基于上节给出的弱基本环的概念，我们可以定义 $HWEF$ (Head-Weak-Elementary-loop-Free)程序。

定义 3.3 ($HWEF$ 程序)： 给定析取逻辑程序 P ，如果对于任意规则 $r \in P$ 和任意弱基本环 L ，有 $|head(r) \cap L| \leq 1$ ，那么我们称 P 为 $HWEF$ 程序。

根据定义3.3和定义2.26，我们可以知道 $HWEF$ 程序同时也是 HEF 程序，由于 HEF 程序的回答集和 $sh(P)$ 一一对应[28]，所以 $HWEF$ 程序也是一样。然而，判断一个程序是否为 $HWEF$ 程序的时间复杂性依然是 $coNP$ -complete。

虽然 $HWEF$ 程序的识别很困难，但是弱基本环的一些性质可以帮助我们构造一个多项式的算法，用于判断程序是否属于 $HWEF$ 程序的一个子类，记为 $HWEF^*$ 程序。

命题 3.4： 给定析取逻辑程序 P ， L 都是 P 的环， E 是 L 的非空真子集。如果 $R^-(E) \subseteq R^-(L)$ ，那么存在环 L' ，使得 $R^-(L') \subseteq R^-(L)$ ， $L' \subset L$ 。

证明： 如果 E 是环，那么 $L' = E$ 满足要求。如果 E 不是环，那么对于 E 在程序的正依赖图的诱导子图，把该诱导子图的在同一强连通分量的原子看成一个整体，则形成有向无环图，而有向无环图是肯定存在入度或出度为0的节点的。因此，总会存在强连通分量 L' ，使得不存在规则 r ，满足 $head(r) \cap L' \neq \emptyset$ ， $body^+(r) \cap L' = \emptyset$ ， $body^+(r) \cap (E \setminus L') \neq \emptyset$ 。换句话说，对于所有 $r \in R^-(L')$ ，有 $body^+(r) \cap E = \emptyset$ ，由此可得 $R^-(L') \subseteq R^-(E)$ ，所以存在 $R^-(L') \subseteq R^-(L)$ ， $L' \subset L$ 。 ■

命题 3.5: 给定析取逻辑程序 P , L 都是 P 的环, E 是 L 的非空真子集。如果不存在规则 $r \in P$, 使得 $body^+(r) \cap E \neq \emptyset$, $body^+(r) \cap (L \setminus E) = \emptyset$, $head(r) \cap (L \setminus E) \neq \emptyset$, 那么 L 不是弱基本环。

证明: 由于不存在这样的规则 r , 所以有:

$$R^-(L \setminus E) \subseteq R^-(L)$$

由命题3.4可知, 存在环 $L' \subseteq L$, 满足:

$$R^-(L') \subseteq R^-(L)$$

所以 L 不符合弱基本环的定义。 ■

基于命题3.5, 我们给出了算法4, 用来判断对于非空原子集 E , 是否不存在弱基本环 L , $E \subset L$ 。算法4从包含 E 的强连通分量($E \subseteq C$)出发, 对每条体部正文字与 E 有交集的公式, 删除 C 中与 $body^+(r) \setminus E$ 相交的部分并求得残留图的强连通分量 C_r ; 判断 C_r 是否满足命题3.5的性质, 若满足, 则返回真。遍历所有公式后, 若找不到符合要求的, 则返回假。

Algorithm 4: 判断是否不存在包含某环的弱基本环的算法 $EWEL(P, E)$

输入: 析取逻辑程序 P , 正依赖图中的环 E

输出: 若返回假, 则不存在弱基本环 $L(L \supset E)$; 若返回真, 则可能存在

```

1  $G := P$ 的正依赖图;
2 if 不存在 $G$ 的强连通分量 $C$ , 使得 $E \subseteq C$  then
3   | return false
4  $C :=$ 满足 $E \subseteq C$ 的强连通分量
5  $R_E := \{r | r \in P \text{ and } body^+(r) \cap E \neq \emptyset\}$ 
6 for 规则 $r \in R_E$  do
7   |  $G_r :=$ 原子集合 $C \setminus (body^+(r) \setminus E)$ 在 $G$ 的诱导子图;
8   | if 存在 $G_r$ 中的强连通分量 $C_r$ , 使得 $E \subseteq C_r$ ,  $head(r) \cap (C_r \setminus E) \neq \emptyset$  then
9     | | return true
10 return false

```

对于给定的析取逻辑程序 P 和任意非空原子集 E ，算法4将在 $O(m)$ 的时间内返回真或假，其中， m 为 P 的规则数目。若返回假，那么表示不存在 P 的弱基本环 L ，使得 $E \subset L$ 。

基于算法3和算法4，我们给出算法5。算法5可以在多项式时间内判断给定程序是否属于 $HWEF$ 程序的子类，记该子类为 $HWEF^*$ 。对于给定的析取逻辑程序 P ，首先利用程序中的规则，生成基为2的原子集合 $E = \{\{a, b\} | \text{there is a rule } r \in P \text{ s.t. } \{a, b\} \subseteq \text{head}(r)\}$ 。然后对于每个这样的原子集合 E ，使用算法3判断其是否为弱基本环，如果 E 是弱基本环，那么就会有存在 $r \in P$ ，使得 $|\text{head}(r) \cap E| = 2$ ，根据 $HWEF$ 程序的定义，该程序不符合要求，此时，返回假；如果 E 不是弱基本环，那么我们就使用算法4判断是否存在弱基本环 L ，使得 $E \subset L$ 。若算法4返回真，即可能存在弱基本环 L ，使得对某条规则 r ，有 $|\text{head}(r) \cap E| \geq 2$ ，不符合 $HWEF$ 程序的定义。这种情况下，尽管只是有可能不符合，但是我们的处理是也返回假。

Algorithm 5: 判断程序是否属于 $HWEF^*$ 程序的算法 $HWEF^*(P)$

输入: 析取逻辑程序 P

输出: 若返回真，则是 $HWEF$ 程序；若返回假，则可能是

```

1  $\varepsilon := \{\{a, b\} | \text{存在规则 } r \in P, \text{ 使得 } \{a, b\} \subseteq \text{head}(r)\};$ 
2 for 原子集  $E \in \varepsilon$  do
3   if  $E$  是  $P$  的基本环 then
4     return false
5   if  $EWEL(P, E)$  返回 true then
6     return false
7 return true

```

所以对于非 $HWEF$ 程序，算法5肯定返回假。然而，对于 $HWEF$ 程序，则可能返回真，也可能返回假。由此可见， $HWEF^*$ 是 $HWEF$ 的一个子类。另一方面，算法5只有一层循环，里面所调用的算法3和算法4的时间复杂度分别为 $O(n^2)$ 和 $O(m)$ ，所以算法5的时间复杂度为 $O(mn^2)$ ，其中， m 为规则的个数， n 为原子的个数。

例 3.3: 考虑程序 P_5 :

$$p \vee q \leftarrow r.$$

$$r \leftarrow p, q.$$

$$p \leftarrow .$$

程序 P_5 有 6 个环: $\{p\}, \{q\}, \{r\}, \{p, r\}, \{r, q\}, \{p, q, r\}$ 。由于 $R^-(\{p, q, r\}) = \{p \leftarrow .\}$, $R^-(\{q\}) = \emptyset$ 。 $\{p, q, r\}$ 不是弱基本环, 即弱基本环有: $\{p\}, \{q\}, \{r\}, \{p, r\}, \{r, q\}$ 。同时, P_5 是 $HWEF$ 程序。但是, 对于 $E = \{p, q\}$, $EWEL(P_5, E)$ 返回真, 所以 $HWEF^*(P_5)$ 返回假。所以, P_5 是 $HWEF$ 程序, 但不是 $HWEF^*$ 程序。

3.5 本章小结

本章从外部支持的角度出发, 介绍了基本环的另一种更直观的定义。基于这种定义, 本章给出了正规逻辑程序的基本环的识别算法和析取逻辑程序的近似识别算法。针对析取逻辑程序的基本环的识别效率低这一难题, 本章还提出了弱基本环的概念, 同时, 还给出了弱基本环的识别算法。在本章的最后部分, 我们还提出了 $HWEF$ 程序的概念, 随后, 我们还讨论了一个可以识别 $HWEF$ 程序的一个子集 $HWEF^*$ 程序的算法。

第4章 特征环(Proper Loops)

上一章，我们介绍了基本环的概念，并且指出，基本环已经足以完成回答集的求解。本章，我们将进一步指出，并非所有的基本环对于回答集的求解都是必须的，同时，提出了特征环的概念。本章分别从正规逻辑程序和析取逻辑程序的角度介绍特征环及其识别算法，基于特征环的概念，我们还提出了 HPF (Head-Proper-loop-Free)程序。针对识别特征环的时间代价高的问题，我们提出了弱特征环和 $HWPF$ (Head-Weak-Proper-loop-Free)程序的概念和识别算法。

4.1 正规逻辑程序的特征环

本节，我们将针对正规逻辑程序，提出特征环的概念，并给出对应的识别算法。

4.1.1 特征环的定义

对于合取环公式2.7的体部 $\bigwedge_{p \in L} p$ ，实际上，我们并不关注环 L 中的所有原子。对于 $p \in L$ ，如果不存在规则 $r \in R^-(L)$ ，使得 $head(r) = p$ ，那么原子 p 的真假性是由环中的其他原子决定的。所以，我们可以考虑使用 $\bigwedge_{p \in head(R^-(L))} p$ 替换 $\bigwedge_{p \in L} p$ ，得到另一种形式的环公式。

给定正规逻辑程序 P ， L 为 P 中的环。用 $RLF(L, P)$ 表示以下的蕴含式：

$$\bigwedge_{p \in head(R^-(L))} p \supset \bigvee_{r \in R^-(L)} body(r) \quad (4.1)$$

特别地，如果 $R^-(L) = \emptyset$ ，则：

$$\bigwedge_{p \in L} p \supset \perp \quad (4.2)$$

显然， $RLF(L, P)$ 是 $LF(L, P)$ 的一个特殊情况。利用 $RLF(L, P)$ ，我们可以在基本环的基础上，加入更多的限制，从而进一步减少回答集的求解所需要的环的数量。

推论 4.1: 给定正规逻辑程序 P , L_1 和 L_2 为 P 的环。如果 $R^-(L_1) \neq \emptyset$, $R^-(L_1) \subseteq R^-(L_2)$, 那么 $RLF(L_1, P) \supset RLF(L_2, P)$ 。

证明: 记 $A = \bigwedge_{p \in head(R^-(L_1))} p$, $B = \bigvee_{p \in R^-(r)} body(r)$, $C = \bigwedge_{p \in head(R^-(L_2))} p$, $D = \bigvee_{p \in R^-(L_2)} body(r)$, 则:

$$\begin{aligned}
 & RLF(L_1, P) \supset RLF(L_2, P) \\
 \Leftrightarrow & (A \rightarrow B) \rightarrow (C \rightarrow D) \\
 \Leftrightarrow & (\neg A \vee B) \rightarrow (\neg C \vee D) \\
 \Leftrightarrow & (A \wedge \neg B) \vee (\neg C \vee D) \\
 \Leftrightarrow & (A \vee \neg C \vee D) \wedge (\neg B \vee \neg C \vee D)
 \end{aligned}$$

另一方面, 由 $R^-(L_1) \subseteq R^-(L_2)$, 得: $\bigwedge_{p \in head(R^-(L_1))} p \leftarrow \bigwedge_{p \in head(R^-(L_2))} p$, 即 $C \rightarrow A$ 为真。由 $R^-(L_1) \subseteq R^-(L_2)$, 得: $\bigvee_{p \in R^-(r)} body(r) \rightarrow \bigvee_{p \in R^-(L_2)} body(r)$, 即 $B \rightarrow D$ 为真。所以有:

$$\begin{aligned}
 & true \wedge true \\
 \Leftrightarrow & (C \rightarrow A) \wedge (B \rightarrow D) \\
 \Leftrightarrow & (\neg C \vee A) \wedge (\neg B \vee D) \\
 \Rightarrow & (\neg C \vee A \vee D) \wedge (\neg C \vee \neg B \vee D)
 \end{aligned}$$

由于对于 $p \rightarrow q$ 为真, 若 p 为真, 则 q 为真, 所以得出 $RLF(L_1, P) \supset RLF(L_2, P)$ 为真。 ■

基于推论4.1, 我们可以给出一种新的环(特征环)的定义。

定义 4.1 (正规逻辑程序的特征环): 给定正规逻辑程序 P , L 为 P 的环。我们称 L 为 P 的特征环, 如果不存在 P 其他环 L' , 使得 $L' \subset L$, $R^-(L') \subseteq R^-(L)$ 或者 $R^-(L') \neq \emptyset$, $R^-(L') \subset R^-(L)$ 。

注意到, 特征环的第一个条件实际上就是要求环 L 是未被抑制的, 根据命题3.1, 我们可以知道, 其实就是要求 L 是基本环。因此, 特征环是在基本环的基础上, 再加入了额外的限制, 所以特征环是基本环的子集。

定理 4.1: 给定正规逻辑程序 P 和原子集合 S , 如果 S 满足 P , 那么以下结论和定理2.5、定理2.6以及定理3.1都是等价的:

1. 对于 P 中任意特征环 L , S 满足 $RLF(L, P)$;
2. 对于 P 中任意特征环 L , S 满足 $DLF(L, P)$;

例 4.1: 考虑例2.7的程序 P_1 , 由于 $R^-(\{p, r\}) = \{p \leftarrow ., r \leftarrow q.\}$, $R^-(\{p\}) = \{p \leftarrow ., p \leftarrow r.\}$, $R^-(\{p, q, r\}) = \{p \leftarrow .\}$, 所以 $\{p, r\}$ 和 $\{p\}$ 都不是特征环。

由于 $R^-(\{r\}) = \{r \leftarrow p. r \leftarrow q.\}$, $R^-(\{q, r\}) = \{r \leftarrow p.\}$, 所以 $\{r\}$ 也不是特征环。

因此, P_1 的特征环只有 $\{q\}, \{r, q\}, \{p, r, q\}$ 。

由命题3.1和定义4.1, 我们可以知道, 给定正规逻辑程序 P , L 为 P 的环。如果 L 是特征环, 那么 L 同时也是基本环, 反过来则不成立。对于基本环 L , 如果不存在其他基本环 L' , 满足 $R^-(L') \neq \emptyset$, $R^-(L') \subset R^-(L)$, 那么基本环 L 才会是特征环。注意到, 这里并没有规定 L' 是 L 的子集, 所以需要检查的 L' 的可能非常多。不过, 我们可以加入额外的条件, 限制 L' 的范围。

定义 4.2 (正规逻辑程序在原子集合下的特征环): 给定正规逻辑程序 P 和原子集合 S , L 是 P 的环。我们称 L 是 P 在 S 下的特征环, 如果 $L \subseteq S$, 且不存在其他环 $L' \subseteq S$, 使得 $L' \subset L$, $R^-(L') \subseteq R^-(L)$ 或者 $R^-(L') \neq \emptyset$, $R^-(L') \subset R^-(L)$ 。

4.1.2 特征环的识别

下面, 基于定义4.2, 我们将介绍特征环的识别算法。

给定正规逻辑程序 P , 原子集合 S , L 为 P 中的环。记 P 的正依赖图为 G_P , 原子集合 S 在 P 中的诱导子图为 G_P^S 。首先, 算法6找出 G_P^S 的所有强连通分量的集合 SCC 。对于每个强连通分量 $C \in SCC$:

Algorithm 6: 正规逻辑程序特征环的识别算法 $ProperLoop(L, P, S)$ **输入:** 正依赖图中的环 L , 析取逻辑程序 P , 原子集合 S **输出:** 若返回 L , 则 L 是特征环; 否则, L 不是特征环

```

1  $G_P^S :=$  原子集合  $S$  在  $G_P$  中的诱导子图;
2  $SCC := G_P^S$  的强连通分量的集合;
3 for  $C \in SCC$  do
4   if  $C \subset L$  且  $R^-(C) \subseteq R^-(L)$  then
5     return  $C$ 
6   else if  $R^-(C) \neq \emptyset$  且  $R^-(C) \subset R^-(L)$  then
7     return  $C$ 
8   else if  $R^-(C) = \emptyset$  或  $R^-(C) = R^-(L)$  then
9     for  $a \in C$  do
10       $G^* :=$  原子集合  $C \setminus \{a\}$  在  $G_P^S$  的诱导子图;
11       $SCC^* := G^*$  的强连通分量的集合;
12      把  $SCC^*$  里面的新元素加入到  $SCC$ ;
13   else
14      $G_C :=$  原子集合  $C \setminus head(R^-(C) \setminus R^-(L))$  在  $G_P^S$  的诱导子图;
15      $SCC_C := G_C$  的强连通分量的集合;
16     把  $SCC_C$  里面的新元素加入到  $SCC$ ;
17 return  $L$ 

```

1. 若 $C \subset L$ 且 $R^-(C) \subseteq R^-(L)$, 那么此时符合定义4.2的第一种情况, 返回 C , 表示 C 使得 L 不是特征环;
2. 若 $R^-(C) \neq \emptyset$ 且 $R^-(C) \subset R^-(L)$, 那么此时符合定义4.2的第二种情况, 返回 C , 表示 C 使得 L 不是特征环;
3. 若 $R^-(C) = \emptyset$, 那么 C 对 L 没有影响, 但是, C 的子环则有可能使得 L 不是特征环。所以这种情况, 我们将删除 C 的一个原子, 获得其残留图, 并把残留图的新的强连通分量加入到 SCC 中, 然后重复次整个判断过程;
4. 若 $R^-(C) = R^-(L)$, 那么意味着 C 和 L 的环公式是一样的, 从环公式的角度来看, 只需要保留一个即可。由于现在我们是在判断 L 是否为特征环, 而 C 并不能否决 L , 所以这种情况的处理实际和情况3是一样的: 获取 C 的所

有子环，并对其重复整个判断过程；

5. 其他情况，即 C 和 L 或者 $R^-(C)$ 和 $R^-(L)$ 都没有直接的包含关系。此时的处理和情况3、4是一样的，不过，我们可以做一些优化：考虑到在 $R^-(C)$ 中，实际上我们只需要关注的是在 $R^-(C)$ 和 $R^-(L)$ 的交集，所以，可以直接删掉 C 中的原子集 $head(R^-(C) \setminus R^-(L))$ 得到残留图 G_C ，并获得 G_C 的强连通分量，最后对其重复整个判断过程。

若没有找到任何 C ，使得 L 不是特征环，那么算法6将返回 L ，表示 L 是特征环。

算法6采取自顶向下的策略，先大环开始，逐步深入到小环。需要注意的是，该算法每次都至少删除子图中的一个原子。所以，在最坏的情况下，算法将迭代 n^2 次，其中， n 为环 L 中的原子数目，即算法6的时间复杂度为 $O(n^2)$ 。

命题 4.1: 给定正规逻辑程序 P ， L 是 P 的特征环，且满足 $R^-(L) \neq \emptyset$ 。如果 L' 是 P 的一个环，且满足 $L' \subset L$ ， $head(R^-(L)) \subseteq L'$ ，那么 L' 不是特征环。

证明: 因为 L 是特征环，且满足 $L' \subset L$ ，由定义4.1，有 $R^-(L') \not\subseteq R^-(L)$ 。另一方面，由 $head(R^-(L)) \subseteq L'$ ，得 $R^-(L) \subseteq R^-(L')$ ，所以， $R^-(L) \subset R^-(L')$ 。根据定义4.1， L 使得 L' 不是特征环。 ■

想要求得程序的所有特征环，一个直接的方法是，使用算法6去对程序的每个环进行过滤。显然，这样做的效率并不高，毕竟程序的环的数目是可能指数爆炸的。利用命题4.1，我们可以直接忽略掉部分不可能是特征环的环，从而在一定的程度上提高算法的效率。

给定正规逻辑程序 P ，算法7将计算程序 P 在原子集合 S 下的所有特征环。与之前的算法一样，算法7也是采用自顶向下的方式考虑所有环。记 G_P 为程序的正依赖图， G_P^S 为 G_P 关于原子集合 S 的诱导子图。首先，算法7求得 G_P^S 的强连通分量 SCC ，然后，对 SCC 里面的每个强连通分量 C 使用算法6判断其是否为特征环。如果 C 是特征环，那么根据推论4.1，我们并不需要遍历 C 的所有子环，所以

Algorithm 7: 正规逻辑程序所有特征环的识别算法 $ProperLoops(P, S)$ **输入:** 析取逻辑程序 P , 原子集合 S **输出:** 返回程序的所有特征环

```

1  $Loops := \emptyset$ ;
2  $G_P^S :=$  原子集合  $S$  在  $G_P$  中的诱导子图;
3  $SCC := G_P^S$  的强连通分量的集合;
4 for  $C \in SCC$  do
5    $C^* := ProperLoop(C, P, S)$ ;
6   if  $C^* = C$  then
7     将  $C$  加入到  $Loops$ ;
8     for  $a \in head(R^-(C))$  do
9        $G_C :=$  原子集合  $C \setminus \{a\}$  在  $G_P^S$  的诱导子图;
10       $SCC_C := G_C$  的强连通分量的集合;
11      把  $SCC_C$  里面的新元素加入到  $SCC$ ;
12   else
13     for  $a \in C$  do
14        $G_C :=$  原子集合  $C \setminus \{a\}$  在  $G_P^S$  的诱导子图;
15        $SCC_C := G_C$  的强连通分量的集合;
16       把  $SCC_C$  里面的新元素加入到  $SCC$ ;
17 return  $Loops$ 

```

此时, 每次删除 C 中的一个原子 a , 其中 $a \in head(R^-(C))$, 然后把残留图的强连通分量加入 SCC 中, 并重复整个判断过程; 如果 C 不是特征环, 那么我们需要遍历其 C 的所有子环, 此时每次删除 C 中的一个原子, 破坏其连通性, 然后把残留图的强连通分量加入 SCC 中, 并重复整个判断过程。

4.2 析取逻辑程序的特征环

上节, 我们介绍了正规逻辑程序的特征环, 本节, 我们将继续把特征环的概念拓展到析取逻辑程序。

4.2.1 特征环的定义

析取逻辑程序的头部可能会有多个原子, 对于环 L 来说, 我们需要关注的是 $head(R^-(L)) \cap L$ 部分, 因此对于析取逻辑程序, 我们使用 $\bigwedge_{p \in head(R^-(L)) \cap L} p$ 去

替换公式2.6中的 $\bigwedge_{p \in L} p$, 从而得到另一种形式的环公式。

定义 4.3: 给定析取逻辑程序 P , L 为 P 中的环。用 $RLF(L, P)$ 表示以下的蕴含式:

$$\bigwedge_{p \in \text{head}(R^-(L)) \cap L} p \supset \bigvee_{p \in R^-(L)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L} \neg q) \quad (4.3)$$

特别地, 如果 $R^-(L) = \emptyset$, 则:

$$\bigwedge_{p \in L} p \supset \perp \quad (4.4)$$

显然, $RLF(L, P)$ 是 $LF(L, P)$ 的一个特殊形式。当使用这种形式的环公式时, 我们可以在基本环的基础上, 排除更多的环。

推论 4.2: 给定析取逻辑程序 P , L_1 和 L_2 为 P 的环。如果 $R^-(L_1) \neq \emptyset$, $R^-(L_2) \neq \emptyset$, $\text{head}(R^-(L_1)) \cap (L_1 \cup L_2) \subseteq \text{head}(R^-(L_2)) \cap L_2$, $R_{L_2}^-(L_1) \subseteq R^-(L_2)$, 那么 $RLF(L_1, P) \supset RLF(L_2, P)$ 。

证明: 由 $\text{head}(R^-(L_1)) \cap (L_1 \cup L_2) \subseteq \text{head}(R^-(L_2)) \cap L_2$, 得 $\text{head}(R^-(L_1)) \cap L_1 \subseteq \text{head}(R^-(L_2)) \cap L_2$, 所以有:

$$\bigwedge_{p \in \text{head}(R^-(L_2)) \cap L_2} p \supset \bigwedge_{p \in \text{head}(R^-(L_1)) \cap L_1} p$$

对于任意的 $r \in R^-(L_1)$, 如果 $r \in R_{L_2}^-(L_1)$, 那么由 $R_{L_2}^-(L_1) \subseteq R^-(L_2)$ 可知, $r \in R^-(L_2)$, 因此 $\text{head}(r) \cap L_1 \subseteq L_2$, $\text{head}(r) \cap (L_2 \setminus L_1) = \emptyset$, 从而有 $\text{head}(r) \cap L_1 = \text{head}(r) \cap L_2$ 。此时有:

$$\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q \supset \text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q$$

如果 $r \notin R_{L_2}^-(L_1)$, 那么 $\text{head}(r) \cap (L_2 \setminus L_1) \neq \emptyset$ 。由于 $\text{head}(r) \cap L_2 \subseteq \text{head}(R^-(L_2)) \cap L_2$, 所以, 存在 $q \in \text{head}(r) \setminus L_1$ 且 $q \in \text{head}(R^-(L_2)) \cap L_2$ 。在 $\bigwedge_{p \in \text{head}(R^-(L_2)) \cap L_2} p$ 为真的条件下, $\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q$ 为假。因此, 在 $\bigwedge_{p \in \text{head}(R^-(L_2)) \cap L_2} p$ 为真的条件下, 也有:

$$\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q \supset \text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q$$

所以, 在 $\bigwedge_{p \in \text{head}(R^-(L_2)) \cap L_2} p$ 为真的条件下:

$$\bigwedge_{r \in R^-(L_1)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_1} \neg q) \supset \bigwedge_{r \in R^-(L_2)} (\text{body}(r) \wedge \bigwedge_{q \in \text{head}(r) \setminus L_2} \neg q)$$

综合上述, $RLF(L_1, P) \supset RLF(L_2, P)$ 成立。 ■

基于推论4.2, 我们给出析取逻辑程序的特征环的定义。

定义 4.4 (析取逻辑程序的特征环): 给定析取逻辑程序 P , L 为 P 的环。我们称 L 为 P 的特征环, 如果 L 满足:

1. L 是 P 的基本环;
2. 不存在 P 中的其他基本环 L' , 使得 $R^-(L') \neq \emptyset$, $\text{head}(R^-(L')) \cap (L' \cup L) \subseteq \text{head}(R^-(L)) \cap L$, $R_L^-(L') \subset R^-(L)$ 。

例 4.2: 例2.8的程序 P_2 有4个特征环: $\{p\}, \{q\}, \{p, r\}, \{r, q\}$ 。对于环 $\{r, q, p\}$, 因为它不是基本环, 所以它也不可能是特征环; 对于环 $\{r\}$, 因为 $R_{\{r\}}^-(\{r, q\}) = \{r \leftarrow q.\}$, $R^-(\{r\}) = \{r \leftarrow p. r \leftarrow q.\}$, 所以它不是特征环。

对于所有环的外部支持都不为空的程序, 特征环的定义可以更为简洁。

定义 4.5 (简化的): 给定析取逻辑程序 P , 我们称 P 是简化的(*simplified*), 如果 P 不存在任何外部支持为空的环。

任意的析取逻辑程序 P 都可以转化为化简的析取逻辑程序, 记为 $\text{simp}(P)$, 具体操作如下:

1. 删除规则 r , 其中, r 和 P 的某个环 L 满足 $\text{body}^+(r) \cap L \neq \emptyset$, $R^-(L) = \emptyset$;
2. 对于剩下的公式, 若规则头部有 p , 则删除 p ; 若规则体部有 $\text{not } p$, 则删除规则, 其中, $p \in L$, L 为 P 的环且满足 $R^-(L) = \emptyset$ 。

该操作是个递归调用的过程, 若 $\text{simp}(P)$ 还不是简化的, 那么我们需要继续对结果进行化简, 即 $\text{simp}(\text{simp}(P))$, 直到返回结果为简化的为止。

命题 4.2: 给定简化的程序 P , 环 L 是特征环当且仅当:

1. 不存在 L 的非空真子集 C , 使得 $R_L^-(C) \subseteq R^-(L)$, 同时,
2. 不存在非空原子集 C , 使得 $\text{head}(R^-(C)) \cap (C \cup L) \subseteq \text{head}(R^-(L)) \cap L$, $R_L^-(C) \subset R^-(L)$ 。

4.2.2 特征环的识别

和析取逻辑程序的基本环类似, 识别析取逻辑程序的特征环是coNP-complete的。下面, 我们将给出多项式时间复杂度的近似算法 $PL^*(L, P)$, 它可以判断一个环是否属于特征环的超集, 记为 $PL^*(P)$ 。

在下文中, 我们将使用 $PL(P)$ 表示程序 P 的所有特征环, 使用 $PL^*(P)$ 表示 $PL^*(L, P)$ 返回 L 的所有环。对于简化的析取逻辑程序 P , $PL(P) \subseteq PL^*(P)$ 。特别地, 如果 P 是正规逻辑程序, 则 $PL(P) = PL^*(P)$ 。

给定析取逻辑程序 P , L 为 P 中的环。记 P 的正依赖图为 G_P 。首先, 算法8找出 G_P 的所有强连通分量的集合 SCC 。对于每个强连通分量 $C \in SCC$:

1. 若 $C \subset L$ 且 $R_L^-(C) \subseteq R^-(L)$, 则返回 C , 表示 L 不属于 $PL^*(L, P)$;
2. 若 $\text{head}(R^-(C)) \cap (C \cup L) \subseteq \text{head}(R^-(L)) \cap L$ 且 $R_L^-(C) \subset R^-(L)$, 则返回 C , 表示 L 不属于 $PL^*(L, P)$;
3. 若 $R_L^-(C) = R^-(L)$, 此时需要考虑 C 的子环, 我们将每次删除 C 的一个原子, 并把对应的诱导子图的新强连通分量加入到 SCC 中去, 然后对其重复整个判断过程;
4. 若 $\text{head}(R^-(C)) \cap (C \cup L) \not\subseteq \text{head}(R^-(L)) \cap L$ 且 $C \not\subseteq L$, 此时, 把 $C \setminus ((\text{head}(R^-(C)) \cap C) \setminus \text{head}(R^-(L)) \cap L)$ 对应的诱导子图的强连通分量加入到 SCC 中去, 并重复整个判断过程;
5. 其他情况, 则把 $C \setminus \text{head}(R_L^-(C) \setminus R^-(L))$ 对应的诱导子图的强连通分量加入到 SCC 中去, 并重复整个判断过程;

Algorithm 8: 析取逻辑程序特征环的超集识别算法 $PL^*(L, P)$ **输入:** 正依赖图中的环 L , 正规逻辑程序 P **输出:** 若返回 L , 则 L 在特征环超集里; 若返回环 C , 则 L 不在特征环的超集里

```

1  $SCC := G_P$  的强连通分量的集合;
2 for  $C \in SCC$  do
3   if  $C \subset L$  且  $R_L^-(C) \subseteq R^-(L)$  then
4     return  $C$ 
5   else if  $head(R^-(C)) \cap (C \cup L) \subseteq head(R^-(L)) \cap L$  且  $R_L^-(C) \subset R^-(L)$  then
6     return  $C$ 
7   else if  $R_L^-(C) = R^-(L)$  then
8     for  $a \in C$  do
9        $G^* :=$  原子集合  $C \setminus \{a\}$  在  $G_P$  的诱导子图;
10       $SCC^* := G^*$  的强连通分量的集合;
11      把  $SCC^*$  的所有新元素加入到  $SCC$  中;
12   else if  $head(R^-(C)) \cap (C \cup L) \not\subseteq head(R^-(L)) \cap L$  且  $C \not\subseteq L$  then
13      $C' := C \setminus ((head(R^-(C)) \cap C) \setminus (head(R^-(L)) \cap L))$ ;
14      $G' :=$  原子集合  $C'$  在  $G_P$  的诱导子图;
15      $SCC' := G'$  的强连通分量的集合;
16     把  $SCC'$  的所有新元素加入到  $SCC$  中;
17   else
18      $G_C :=$  原子集合  $C \setminus head(R_L^-(C) \cap R^-(L))$  在  $G_P$  的诱导子图;
19      $SCC_C := G_C$  的强连通分量的集合;
20     把  $SCC_C$  的所有新元素加入到  $SCC$  中;
21 return  $L$ 

```

对于简化的析取逻辑程序 P 及其环 L , $PL^*(L, P)$ 将返回环 L 或者环 C , 其中, 环 C 满足 $C \subset L$, $R_L^-(C) \subseteq R^-(L)$, 或者 $\text{head}(R^-(C)) \cap (C \cup L) \subseteq \text{head}(R^-(L)) \cap L$, $R_L^-(C) \subset R^-(L)$ 。

4.3 HPF程序

基于特征环的概念, 我们可以定义一种叫做 HPF (Head-Proper-loop-Free)程序的析取逻辑程序类别。

定义 4.6 (HPF程序): 给定析取逻辑程序 P , 我们称 P 是 HPF 程序, 如果对于任何规则 r 和特征环 L , 有 $|\text{head}(r) \cap L| \leq 1$ 。

与 HEF 程序类似, 对于任意的 HPF 程序 P , 原子集合 S 是 P 的回答集当且仅当 S 是 $sh(P)$ 的回答集。

命题 4.3: 给定析取逻辑程序 P , 如果 P 是 HEF 程序, 那么 P 同时也是 HPF 程序。

证明: 使用反证法, 假设 P 不是 HPF 程序, 那么存在特征环 L 和规则 r , 满足 $|\text{head}(r) \cap L| > 1$ 。由于特征环同时也是基本环, 所以 P 不是 HEF 程序, 矛盾, 因此命题成立。 ■

与 HEF 程序一样, 判断析取逻辑程序是否为 HPF 程序的时间复杂性是 coNP-complete 的。

4.4 弱特征环

针对识别析取逻辑程序的特征环和 HPF 程序的时间复杂性都为 coNP-complete 的问题, 本节, 我们提出了弱特征环和 $HWPF$ 程序的概念, 其中, 弱特征环是特征环的超集, $HWPF$ 程序是 HPF 程序的子集。

推论 4.3: 给定析取逻辑程序 P , L_1 和 L_2 是 P 的环。如果 $R^-(L_1) \neq \emptyset$, $\text{head}(R^-(L_1)) \cap L_1 \subseteq \text{head}(R^-(L_2)) \cap L_2$, $R^-(L_1) \subseteq R^-(L_2)$, 那么 $RLF(L_1, P) \supset RLF(L_2, P)$ 。

证明： 由 $R^-(L_1) \neq \emptyset$, $head(R^-(L_1)) \cap L_1 \subseteq head(R^-(L_2)) \cap L_2$, 有:

$$\bigwedge_{p \in head(R^-(L_2)) \cap L_2} p \supset \bigwedge_{p \in head(R^-(L_1)) \cap L_1} p$$

对于规则 $r \in R^-(L_1)$, 由 $head(r) \cap L_1 \subseteq head(r) \cap L_2$, 有 $head(r) \setminus L_2 \subseteq head(r) \setminus L_1$ 。因此, $\bigwedge_{q \in head(r) \setminus L_1} \neg q \supset \bigwedge_{q \in head(r) \setminus L_2} \neg q$ 。所以, 我们有:

$$\bigvee_{r \in R^-(L_1)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L_1} \neg q) \supset \bigvee_{r \in R^-(L_2)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L_2} \neg q)$$

综合上述, $RLF(L_1, P) \supset RLF(L_2, P)$ 成立。 ■

下面, 我们给出弱特征环的定义。

定义 4.7 (弱特征环): 给定析取逻辑程序 P , L 为 P 的环。我们称 L 是弱特征环, 如果 L 是弱基本环, 同时不存在其他弱基本环 L' , 满足 $R^-(L') \neq \emptyset$, $head(R^-(L')) \cap L' \subseteq head(R^-(L)) \cap L$, $R^-(L') \subset R^-(L)$ 。

例 4.3: 程序 P_2 有 5 个弱特征环: $\{p\}, \{q\}, \{p, r\}, \{q, r\}, \{p, q, r\}$ 。对于环 $\{r\}$, 由于 $R^-(\{r\}) = \{r \leftarrow p. r \leftarrow q.\}$, $R^-(\{r, q\}) = \{r \leftarrow p.\}$, 所以它不是弱基本环。

定理 4.2: 给定析取逻辑程序 P 和原子集合 S , 如果 S 满足 P , 那么以下结论和定理 2.5、定理 2.6、定理 3.1 以及定理 4.1 都是等价的:

1. 对于 P 中任意弱特征环 L , S 满足 $RLF(L, P)$;
2. 对于 P 中任意弱特征环 L , S 满足 $DLF(L, P)$;

显然, 如果环 L 是程序的特征环, 那么它也会是程序的弱特征环, 反过来则不一定成立。

各种环种类之间的关系如图 4.1 所示, 其中, PL 表示特征环, PL^* 表示算法 $PL^*(L, P)$ 识别出来的环, WPL 表示弱特征环, EL 表示基本环, EL^* 表示算法 $EL^*(L, P)$ 识别出来的环, WEL 表示弱基本环, \rightarrow 表示子集关系:

利用简化的的概念, 我们可以更简洁地去定义弱特征环。

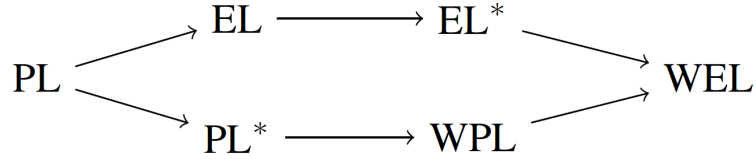


图 4.1: 环种类之间的关系

命题 4.4 (简化程序的弱特征环): 给定简化的析取逻辑程序 P , 环 L 是 P 的弱特征环, 当且仅当不存在 L 的非空真子集 C , 使得 $R^-(C) \subseteq R^-(L)$, 同时, 也不存在非空原子集 C , 使得 $\text{head}(R^-(C)) \cap C \subseteq \text{head}(R^-(L)) \cap L$, $R^-(C) \subset R^-(L)$ 。

与特征环的识别不同, 简化的程序的弱特征环的识别可以在多项式时间内完成。考虑到弱特征环和特征环在定义上的相似性, 我们只需要把算法8中的 $R_L^-(C)$ 替换成 $R^-(C)$ 、 $\text{head}(R^-(C)) \cap (C \cup L)$ 替换成 $\text{head}(R^-(C)) \cap C$, 就可以得到用于识别弱特征环的算法。

4.5 HWPF程序

同样, 我们可以使用弱基本环的概念, 定义一种叫 *HWPF* (Head-Weak-Proper-loop-Free) 程序的类别。

定义 4.8 (HWPF程序): 给定简化的析取逻辑程序 P , 我们称 P 是 *HWPF* 程序, 如果对于 P 的任意规则 r 和任意弱特征环 L , 有 $|\text{head}(r) \cap L| \leq 1$ 。

各种程序类别之间的关系如图4.2所示, 其中, \rightarrow 表示子集关系:

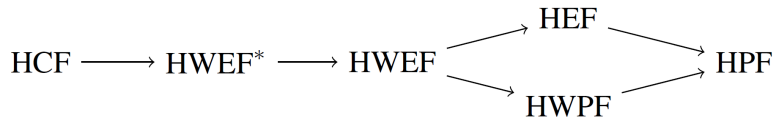


图 4.2: 各种程序类别之间的关系

4.6 本章小结

在上一章的基础上, 本章进一步提出了特征环的概念, 并证明了特征环已经足以完成逻辑程序的回答集的求解。基于这些概念, 本章给出了正规逻辑程序的

基本环的识别算法和析取逻辑程序的近似识别算法。针对析取逻辑程序的特征环的识别效率低这一难题，本章还提出了弱特征环的概念，同时，还给出了弱基本环的识别算。在本章的最后部分，我们还提出了 HPF 程序和 $HWPF$ 程序的概念，同时，还总结了各种程序类别之间的关系。

第 5 章 实验过程与结果分析

5.1 实验概述

第 6 章 总结与展望

本章是全文的总结。本章首先概述了本文的主要工作和研究成果，然后对后续的研究工作做出了展望。

6.1 本文的研究工作总结

参考文献

- [1] Davis R, Shrobe H, Szolovits P. What is a knowledge representation? [J]. AI magazine, 1993, 14(1):17.
- [2] Rada R, Mili H, Bicknell E, et al. Development and application of a metric on semantic nets [J]. Systems, Man and Cybernetics, IEEE Transactions on, 1989, 19(1):17–30.
- [3] Uschold M, Gruninger M. Ontologies: Principles, Methods and Applications [J]. To appear in Knowledge Engineering Review, 1996, 11(2).
- [4] McCarthy J. Programs with common sense [M]. Defense Technical Information Center, 1963.
- [5] McCarthy J, Hayes P. Some philosophical problems from the standpoint of artificial intelligence [M]. Stanford University USA, 1968.
- [6] McDermott D, Doyle J. Non-monotonic logic I [J]. Artificial intelligence, 1980, 13(1):41–72.
- [7] McCarthy J. Circumscription—a form of nonmonotonic reasoning [J]. & &, 1987.
- [8] Reiter R. A logic for default reasoning [J]. Artificial intelligence, 1980, 13(1):81–132.
- [9] Robinson J A. A machine-oriented logic based on the resolution principle [J]. Journal of the ACM (JACM), 1965, 12(1):23–41.
- [10] Green C. Application of theorem proving to problem solving [R]. DTIC Document, 1969.

- [11] Colmeraner A, Kanoui H, Pasero R, et al. Un systeme de communication homme-machine en francais [C].// Luminy. 1973.
- [12] Kowalski R. Algorithm= logic+ control [J]. Communications of the ACM, 1979, 22(7):424–436.
- [13] Clark K L. Negation as failure [G].// Logic and data bases. Springer, 1978: 293–322.
- [14] Reiter R. On closed world data bases [M]. Springer, 1978.
- [15] Van Gelder A, Ross K A, Schlipf J S. The well-founded semantics for general logic programs [J]. Journal of the ACM (JACM), 1991, 38(3):619–649.
- [16] Gelfond M, Lifschitz V. The stable model semantics for logic programming. [C].// ICLP/SLP. vol 88. 1988: 1070–1080.
- [17] Gelfond M, Lifschitz V. Classical negation in logic programs and disjunctive databases [J]. New generation computing, 1991, 9(3-4):365–385.
- [18] 吉建民. 提高ASP 效率的若干途径及服务机器人上应用[D]. 合肥: 中国科学技术大学, 2010.
- [19] Davis M, Logemann G, Loveland D. A machine program for theorem-proving [J]. Communications of the ACM, 1962, 5(7):394–397.
- [20] Leone N, Pfeifer G, Faber W, et al. The DLV system for knowledge representation and reasoning [J]. ACM Transactions on Computational Logic (TOCL), 2006, 7(3):499–562.
- [21] Niemela I, Simons P, Syrjanen T. Smodels: a system for answer set programming [J]. arXiv preprint cs/0003033, 2000.

- [22] Gebser M, Kaufmann B, Neumann A, et al. clasp: A conflict-driven answer set solver [G].// Logic Programming and Nonmonotonic Reasoning. Springer, 2007: 260–265.
- [23] Lin F, Zhao Y. ASSAT: Computing answer sets of a logic program by SAT solvers [J]. Artificial Intelligence, 2004, 157(1):115–137.
- [24] Lierler Y. cmodels–SAT-based disjunctive answer set solver [G].// Logic Programming and Nonmonotonic Reasoning. Springer, 2005: 447–451.
- [25] Lee J, Lifschitz V. Loop formulas for disjunctive logic programs [G].// Logic Programming. Springer, 2003: 451–465.
- [26] Lifschitz V, Razborov A. Why are there so many loop formulas? [J]. ACM Transactions on Computational Logic (TOCL), 2006, 7(2):261–268.
- [27] Gebser M, Schaub T. Loops: relevant or redundant? [G].// Logic Programming and Nonmonotonic Reasoning. Springer, 2005: 53–65.
- [28] Gebser M, Lee J, Lierler Y. On elementary loops of logic programs [J]. Theory and Practice of Logic Programming, 2011, 11(06):953–988.
- [29] Ben-Eliyahu R, Dechter R. Propositional semantics for disjunctive logic programs [J]. Annals of Mathematics and Artificial intelligence, 1994, 12(1-2):53–87.
- [30] Chen X, Ji J, Lin F. Computing loops with at most one external support rule [J]. ACM Transactions on Computational Logic (TOCL), 2013, 14(1):3.
- [31] 陆钟万. 面向计算机科学的数理逻辑[M]. 北京大学出版社, 1989.
- [32] Rosen K. Discrete Mathematics and Its Applications 7th edition [M]. McGraw-Hill, 2011.

-
- [33] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to algorithms [M]. MIT press Cambridge, 2001.
- [34] Fassetti F, Palopoli L. On the complexity of identifying head-elementary-set-free programs [J]. Theory and Practice of Logic Programming, 2010, 10(01):113–123.