中山大学硕士学位论文

回答集逻辑程序特征环的研究

Research on Proper Loops in Answer Set Program

学位申请人：______________________

指 导 教 师：______________________

专 业 名 称：______________________

答辩委员会主席（签名）：______________________

答辩委员会委员（签名）：______________________

____

____

____

____

二零一五年 五月

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文；可以为建立了馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

保密论文保密期满后，适用本声明。

学位论文作者签名： 导师签名：

日期： 年 月 日 日期： 年 月 日

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程序设计(ASP：Answer Set Programming)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，ASP求解器的效率仍然没能完全满足人们的需求，这也成为影响其推广的一个主要瓶颈。因此，研究与实现更高效的ASP求解器具有非常重要的理论与应用价值。

Lin与Zhao(2002)首次提出了正规逻辑程序(NLP：Normal Logic Program)的环与环公式的概念，将回答集(answer set)的求解归约为求解命题逻辑公式的可满足性(SAT)问题。后来，Lee与Lifschitz(2003)将环公式扩展到析取逻辑程序(DLP：Disjunctive Logic Program)。环与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，环的数目会随着程序规模的增大而指数爆炸。Gebser与Schaub(2005)证明了，并非所有环对于回答集的求解都是有用的，同时，他们提出了正规逻辑程序的基本环(elementary loops)的概念。后来，Gebser等(2011)又将基本环扩展到析取逻辑程序，同时证明了只使用基本环已经可以求解回答集。基本环的出现，减少了回答集的求解所用到的环的数量，推进了基于SAT求解器的ASP求解器的发展。

本文在环与环公式概念的基础上，对基本环进行了深入的研究，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，本文进一步对环的定义加入了限制，提出了特征环(proper loops)的概念。通过把特征环应用到特殊形式的环公式里面，我们发现，只使用特征环也足以求解回答集。总的来说，本文的主要贡献和创新有以下几点：

第一，本文从外部支持(external support)的角度，重新定义的回答集逻辑程序的基本环。对于正规逻辑程序，本文给出了一个基于自顶向下策略的算法。它可以在多项式时间内判断一个环是否为基本环；对于析取逻辑程序，本文给出了多项式时间复杂度的近似算法，用于判断一个环是否属于基本环的超集。

第二，针对正规逻辑程序，本文提出了特征环(proper loops)的概念，并证明了特征环已经足以求解回答集逻辑程序的回答集，同时还给出一个多项式时间复杂度的算法，用于判断一个环是否为特征环。此外，本文还验证了，对于正依赖图符合特定结构的正规逻辑程序，特征环的数量比基本环少很多，同时，特征环

的识别效率也比基本环高很多。

第三，本文将特征环的概念扩展到析取逻辑程序。和正规逻辑程序不一样的是，识别析取逻辑程序的特征环是coNP-complete的。针对这一问题，本文提出了弱基本环(weak elementary loops)和弱特征环(weak proper loops)的概念，并且分别给出了多项式时间复杂度的算法，用于判断一个环是否为弱基本环或弱特征环。

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。另一方面，弱基本环和弱特征环可以在多项式时间内被识别的特点对于提高基于SAT求解器的ASP求解器具有重要的意义。

关键词：正规逻辑程序，析取逻辑程序，环公式，基本环，特征环，回答集求解器

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 not 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 and Zhao (2002), making the computation of answer set reduce to finding models of propositional logic. Later, the notions and the result were extended to Disjunctive Logic Program by Lee and Lifschitz (2003). 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. Gebser and Schaub (2005) showed that not all loops are necessary for selecting the answer sets among the models of a program, they introduced the subclass elementary loops, and later, they (2011) extended it to disjunctive logic programs. Elementary loops decrease the number of loops needed in answer set computation and promote the development of ASP solver.

Basing on the notions of loops and loop formulas, we do a deep research on elementary loops, and find that not all elementary loops are needed in answer set computation. The main contribution and innovation of this paper are as follows:

1. We redefine the notion of elementary loops from the aspect of external support. Basing on this definition, we provide a new algorithm, which follows a top-down strategy, for deciding whether a loop is an elementary loop of NLP in polynomial time. Also, we provide an algorithm running in polynomial time to decide whether a loop is in a superset of all elementary loops of DLP.

2. We introduce a subclass proper loops of elementary loops for NLP, and show that a proper loop can be recognized in polynomial time. For certain programs, the number of proper loops is much smaller than that of elementary loops and identifying all proper loops is more efficient than that of all elementary loops.

3. We extend the notion of proper loops for DLP. Different from NLP, the computational complexities of recognizing proper loops for disjunctive logic programs is coNP-complete. To address this problem, we introduce weaker version of elementary loops and proper loops, providing polynomial time algorithm for identifying them.

Proper loops further reduce the number of loops needed in answer set computation on the basis of elementary loops. On the other hand, the fact that weak elementary loops and weak proper loops can be identified in polynomial time will make great contribution to the development of ASP solver.

Key Words: normal logic programs, disjunctive logic programs, loop formulas, elementary loops, proper loops, ASP solver

目　录

插图

表格

算法

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的背景，然后介绍了当前的各种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(KR：Knowledge Representation and Reasoning)[1]是一个重要的研究方向。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。常见的知识推理方法有归纳推理(inductive reasoning)[4]和演绎推理(deductive reasoning)[5]。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。

图灵奖获得者、人工智能领域先驱McCarthy(1958)考虑实现可以处理常识的人工智能系统。他提到，该系统应该可以接受用户的建议并且能够根据这些建议改进自身的性能和行为[6]。这一系统的构建理论指出了常识推理是人工智能的关键，同时拉开了常识知识表示和推理的研究序幕。McCarthy与Hayes(1968)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[7]。然而，在常识推理中，知识库加入了新的知识后，原有的一些推论往往会被推翻。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)[8]的。而经典逻辑则是单调的，因此，它无法处理非单调的推理问题[9]。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Reiter(1980)的缺省逻辑(default logic)[10]、McDermott(1980)的非单调模态逻辑(non-monotonic modal logic)[8]和McCarthy(1987)的限定理论(circumscription)[11]。

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。Robinson(1965)提出了非常重要的消

除原理(resolution principle)[12]。Green(1967)将逻辑当做一个带有自动推导和构造性逻辑的表示语言[13]。在这些理论的推动下，Colmerauer(1972)等实现了第一个逻辑程序设计语言Prolog[14]。逻辑程序设计的主要思想就是把逻辑和控制分开。Kowalski(1979)提到，算法=逻辑+控制[15]。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(positive programs)[16]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不方便的。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。刻画这一性质的各种全新的语义随之被提出，包括Clark(1978)的完备概念(Clark completion)[17]、Reiter(1978)的闭世界假设(CWA：Closed World Assumption)[18]和Gelder(1991)的良序(well-founded)语义[19]。Lifschitz(1988)等人提出了稳定模型语义(stable models semantics)[20]，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。后来，他们(1991)又将稳定模型语义扩展到析取逻辑程序[21]。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[22]。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，同时这也推进了该语义的逻辑程序设计的发展。这个新的研究领域被研究者们称为回答集程序设计(ASP：Answer Set Programming)[23]。

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ASP求解器(ASP solver)。由于计算回答集逻辑程序的回答集是NP-complete的，大部分求解器都是通过搜索的方式查找回答集。ASP求解器主要分为两大类，一类是基于DPLL算法(Davis-Putnam-Logemann-Loveland procedure)[24]的，主要代表有DLV[25]、smodels[26]和clasp[27]。另一类则是基于SAT求解器[28]的，主要代表

有ASSAT[29]和cmodels[30]。求解器的发展离不开理论的支持。Lin与Zhao(2002)首次提出了正规逻辑程序的环(loops)与环公式(loop formulas)的概念[31]，将逻辑程序回答集的求解归约为命题逻辑公式的可满足性问题，即SAT问题[32]，并使用SAT求解器进行求解。Lin-Zhao规约理论的核心思想在于，通过引入正规逻辑程序的环公式，对程序的正依赖图(positive dependency graph)中的每一个环，添加一个与之相对应的环公式到原逻辑程序的克拉克完备(Clark completion)[17]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(2003)扩展到析取逻辑程序[33]。这些理论的提出，保证了基于SAT求解器的ASP求解器的正确性(correctness)和完备性(completeness)，推进了ASP求解器的发展。

然而，在最坏的情况下，回答集逻辑程序的环的数目会随着程序规模增大而出现指数爆炸[34]。Gebser等(2005)发现，并非所有环对于正规逻辑程序的回答集的求解都是必须的，同时，他们提出了基本环(elementary loops)的概念，并且在仅考虑基本环的情况下，重新定义了Lin-Zhao的规约理论[35]。后来，他们(2011)又把基本环的概念扩展到析取逻辑程序，并且指出，只利用基本环，已经足以完成析取逻辑程序的回答集的求解[36]。同时，他们还提出了头部无基本环的逻辑程序(HEF程序：Head-Elementary-loop-Free Program)[36]，并且指出了，这类析取逻辑程序与头部无环的逻辑程序(HCF程序：Head-Cycle-Free Program)[37]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(2013)通过实验观察到，对于正依赖图符合特定结构的逻辑程序，如果它的所有环的外部支持(external support)都不超过一个，那么使用环公式理论进行转化后，可以显著地提高回答集的计算效率[38]。

1.3 本文的工作

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

第一，本文深入研究了Gebser等提出的基本环的性质及其基于自底向上策略

的识别算法[35]，并从外部支持(external support)的角度，重新定义了回答集逻辑程序的基本环。对于正规逻辑程序，本文给出了一种基于自顶向下策略的识别算法，该算法的时间复杂度和Gebser等(2005)给出的一样；对于析取逻辑程序，本文给出了多项式时间复杂度的近似算法，用于识别基本环的一个超集。

第二，针对正规逻辑程序，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ASP程序的回答集的求解，同时还给出一个多项式时间复杂度的算法，用于识别特征环。此外，本文还验证了，对于正依赖图符合特定结构的正规逻辑程序，特征环的数量比基本环少很多，同时，特征环的识别效率也比基本环高很多。这一结论很好地解释了Ji等的观察结果[38]，即如果回答集逻辑程序的所有环的外部支持都不超过一个，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。

第三，本文将特征环的概念扩展到析取逻辑程序。和正规逻辑程序不一样的是，识别析取逻辑程序的特征环是coNP-complete的。针对这一问题，本文提出了弱基本环和(weak elementary loops)和弱特征环(weak proper loops)的概念，并且分别给出了多项式时间复杂度的算法，用于识别一个环是否为弱基本环或弱特征环。

第四，本文进行了三个实验。实验一对比了正依赖图符合特定结构的正规逻辑程序的基本环与特征环的数量和计算效率，用实际用例说明了特征环的优越性；实验二对比了析取逻辑程序的各种环的数量，并说明了在实际用例中，弱基本环与弱特征环和基本环与特征环在数量上差别不大；实验三针对实际应用中的回答集逻辑程序，统计HWEF*程序所占比例。

本文的研究成果已经整理成两篇学术论文，分别被AAAI-14和AAAI-15收录。其中，AAAI(American Association for Artificial Intelligence)为中国计算机学会(CCF：China Computer Federation)推荐的A类会议。

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、环公式和传统的基本环。

第3章，从外部支持的角度，重新定义了基本环。对于正规逻辑程序，本章给出了多项式时间复杂度的算法，用于判定其环是否为基本环。而对于析取逻辑程序，本章同样给出了多项式时间复杂度的近似算法，用于判定其环是否属于基本环的一个超集。

第4章，针对正规逻辑程序，详细地介绍了特征环的概念和性质，并给出了识别特征环以及计算程序的所有特征环的算法，同时还证明了这些算法的可靠性和完备性。

第5章，把特征环的概念扩展到析取逻辑程序。针对识别析取逻辑程序的特征环是coNP-complete的问题，本章给出多项式时间复杂度的近似算法。基于这些理论基础，本章在最后部分还提出了弱基本环和弱特征环的概念以及他们相对应的HWEF程序和HWPF程序，并总结了各种环以及各种程序类别之间的关系。

第6章，主要介绍了三个实验。实验一针对正依赖图符合特定结构的正规逻辑程序，比较基本环与特征环的数量和计算效率；实验二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；实验三统计在实际应用中，HWEF*程序所占的比例。

第7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节介绍了回答集逻辑程序的语法和语义，从而引出回答集的概念；第3节从回答集逻辑程序的正依赖图出发，介绍了环与环公式的概念及其在回答集求解中的意义；第4节介绍了传统的基本环的概念及其性质。

2.1　命题逻辑

命题逻辑是数理逻辑的一部分，命题逻辑包含一部分的逻辑形式和规律[39]。命题(proposition)是一个非真即假的陈述句，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。

命题逻辑的形式语言$\mathfrak{L}$的符号表包括三类逻辑符号[39]：

1. 命题符号(命题变量)，通常使用小写英文字母表示，比如p和q；
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“(”和“)”。

下面，本文将给出命题逻辑公式各类范式的定义及其相关的定理。

定义 2.1 (否定式) *命题变量的否定称为命题的否定式*[39]。

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)*[39]。

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*[40]。

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[40]。

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(DNF：Disjunctive Normal Form)[40]。

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(CNF：Conjunctive Normal Form)[40]。

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。

命题 2.1 给定命题p和q，如果$p \rightarrow q$为真，p为真，那么q为真。

证明： 由$p \rightarrow q$为真，我们知道$\neg p \vee q$为真。又因为p为真，所以q为真。

2.2 回答集逻辑程序

本节我们将介绍回答集逻辑程序。本文的关注点是被例化(grounding)后的正规逻辑程序和析取逻辑程序。

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(NLP：Normal Logic Program)[41]。一个普通规则具有如下形式：

$$H \leftarrow a_1, ..., a_m, not\ a_{m+1}, ..., not\ a_n. \tag{2.1}$$

其中，$0 \leq m \leq n$，$a_1, ..., a_n$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(contraint)。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：

$$head(r) \leftarrow body(r). \tag{2.2}$$

其中，$head(r) = H$称为规则的头部，$body(r) = body^+(r) \wedge body^-(r)$称为规则的体部，$body^+(r) = a_1 \wedge ... \wedge a_m$，$body^-(r) = \neg a_{m+1} \wedge ... \wedge \neg a_n$，同时我们会将$head(r)$、$body^+(r)$和$body^-(r)$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，$head(R) = \bigcup_{r \in R} head(r)$表示所有在$R$的规则的头部出现过的原子的集合。

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

$$Lit(P) = Atoms(P) \cup \{\neg\, a | a \in Atoms(P)\}. \tag{2.3}$$

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，$\overline{L} = \{\bar{l} | l \in L\}$。

例 2.7 考虑如下的回答集逻辑程序：

$$\begin{aligned}
& a \leftarrow b, not\ c. \\
& b \leftarrow not\ a. \\
& b \leftarrow c. \\
& c \leftarrow . \\
& \leftarrow c, not\ b.
\end{aligned} \tag{2.4}$$

则程序2.4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(DLP：Disjunctive Logic Program)[42]。一个析取规则具有如下形式：

$$a_1 \vee ... \vee a_k \leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n. \tag{2.5}$$

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

例 2.8 考虑如下的回答集逻辑程序:

$$\begin{aligned}
& a \vee b \leftarrow c. \\
& a \leftarrow b, not\ d. \\
& c \leftarrow .
\end{aligned} \tag{2.6}$$

则程序2.6为析取逻辑程序。

2.2.3 回答集逻辑程序的回答集

下面，我们将介绍回答集逻辑程序的回答集[20]。

定义 2.9 (GL规约) 给定一个不含约束的回答集逻辑程序P和原子集合S，P基于S的GL规约(Gelfond-Lifschitz reduction)[20]，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$not\ q$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，对于正规逻辑程序，P^S只有唯一的最小模型(model)；而对于析取逻辑逻辑程序，P^S可能不止一个最小模型，记最小模型的集合为$\Gamma(P^S)$。

例 2.9 程序2.6不含约束，对于$S = \{a, d\}$，其基于S的GL规约结果$P^S = \{a \vee b \leftarrow c.\ c \leftarrow .\}$。注意到，$P^S$的最小模型$\Gamma(P^S) = \{\{a, c\}, \{b, c\}\}$。

定义 2.10 (不含约束的回答集逻辑程序的回答集) 给定一个不含约束的回答集逻辑程序P，原子集合S是P的一个回答集当且仅当$S \in \Gamma(P^S)$[43]。

例 2.10 程序2.6不含约束，对于$S = \{a, c\}$，其基于S的GL规约结果$P^S = \{a \vee b \leftarrow c.\ a \leftarrow b.\ c \leftarrow .\}$。注意到，$P^S$的最小模型$\Gamma(P^S) = \{\{a, c\}\}$，所以$S$是原程序的回答集。

更一般的情况是，逻辑程序P中是含有约束的。

定义 2.11 (回答集逻辑程序的回答集) 给定一个回答集逻辑程序P和原子集合S，记P去掉约束后的程序为P'。S是P的回答集，当且仅当S是P'的回答集且S满足P中的所有约束[22]。

例 2.11 记程序2.4不含约束部分为P'，对于$S = \{b, c\}$，$P'^S = \{b \leftarrow .\ b \leftarrow c.\ c \leftarrow .\}$。注意到，$P'^S$的最小模型$\Gamma(P^S) = \{\{b, c\}\}$，所以$S$是$P'$的回答集。另一方面，$S$满足原程序的约束，所以$S$是原程序的回答集。

2.2.4 回答集逻辑程序的补全

定义 2.12 (回答集逻辑程序的补全) 给定一个回答集逻辑程序P，其补全$Comp(P)$是P的约束和P的克拉克补全(Clark completion)的并集[29]。它包括以下子句：

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

2. 对于约束$\leftarrow G$，则$\neg G$属于$Comp(P)$。

例 2.12 给定程序P：

$$\begin{aligned} & a \leftarrow b, c, not\ d. \\ & a \leftarrow b, not\ c, not\ d. \\ & \leftarrow b, c, not\ d. \end{aligned} \tag{2.7}$$

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

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$sh(P)$[21]。其具体操作每条析取规则替换成如下的形式：

$$\begin{aligned} a_i \leftarrow & not\ a_1, ..., not\ a_{i-1}, not\ a_{i+1}, ..., not\ a_k, \\ & a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n. (1 \leq i \leq k). \end{aligned} \tag{2.8}$$

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，Dechter等(1994)提出了一种头部无环的程序类别，并且证明了这种程序P的回答集和$sh(P)$的回答集一一对应，同时还可以在多项式时间内，转化为非析取逻辑程序[37]。

例 2.13 对于程序2.6，$sh(P)$为：

$$\begin{aligned} &a \leftarrow not\ b, c. \\ &b \leftarrow not\ a, c. \\ &a \leftarrow b, not\ d. \\ &c \leftarrow . \end{aligned} \tag{2.9}$$

定义 2.13 (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$|head(r) \cap L| \leq 1$，那么该程序称为头部无环程序(HCF程序：Head-Cycle-Free Program)[37]。

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。我们甚至可以把回答集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(2004)证明，只要加入环公式(loop formulas)，原回答集逻辑程序的回答集就可以和其对应的命题的模型一一对应[29]。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 2.14 (正依赖图) 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$q \in body^+(r)$[29]。

例 2.14 考虑如下的回答集逻辑程序：

$$
\begin{aligned}
&p \leftarrow . \\
&p \leftarrow r. \\
&q \leftarrow r. \\
&r \leftarrow p. \\
&r \leftarrow q.
\end{aligned}
\tag{2.10}
$$

其正依赖图如图2.1所示。

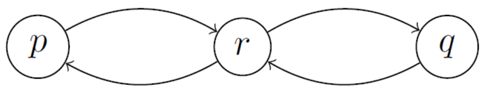

图 2.1: 程序2.10的正依赖图

定义 2.15 (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(subgraph)[44]。

定义 2.16 (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(induced subgraph)[45]。

注意：对于V'，只要在G中有边，那么在G'中同样应该有边。

例 2.15 图2.2是图2.1的子图，但不是其诱导子图，因为缺少边(r, p)。

图 2.2: 图2.1的子图

定义 2.17 (强连通分量) 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[46]。

定义 2.18 (环) 给定原子集合L，如果L中的任意原子p和q在回答集逻辑程序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)[47]。特别地，任意单原子集合都为环。

由定义2.17和定义2.18，我们可以知道，强连通分量是环，但环不一定是强连通分量。

例 2.16 正依赖图2.1有6个环：$\{p\}$，$\{r\}$，$\{q\}$，$\{p, r\}$，$\{q, r\}$，$\{p, r, q\}$，但只有一个强连通分量：$\{p, r, q\}$。

定义 2.19 (子环) 给定环L，环L'是L的子环当且仅当$L' \subset L$且L'对应的诱导子图是一个环[47]。

例 2.17 正依赖图2.1中，环$\{p, r\}$的子环有$\{p\}$和$\{r\}$。

给定回答集逻辑程序P和环L，我们定义如下两种规则的集合：

$$R^{+}(L, P) = \{r | r \in P \ and \ head(r) \cap L \neq \emptyset \ and \ body^{+}(r) \cap L \neq \emptyset\} \tag{2.11}$$

$$R^{-}(L, P) = \{r | r \in P \ and \ head(r) \cap L \neq \emptyset \ and \ body^{+}(r) \cap L = \emptyset\} \tag{2.12}$$

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。下面，我们给出外部支持的概念。

定义 2.20 (外部支持) 给定回答集逻辑程序P，L是P的环。我们称规则$r \in P$是L的外部支持(external support)，如果$r \in R^{-}(L, P)$。

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 2.18 考虑回答集逻辑程序2.10的环$L = \{p, r\}$，有：

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

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

可以观察到，$R^{+}(L)$的回答集为$\emptyset$。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，除非有额外的规则r能推出它，比如$r \in R^{-}(L)$。基于这些观察，Lin等(2004)提出了正规逻辑程序的环公式的概念[29]。对于正规逻辑程序P和环L，其环公式为如下形式：

$$\neg(\bigvee_{r \in R^{-}(L)} body(r)) \supset \bigwedge_{p \in L} \neg p \tag{2.13}$$

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$Comp(P)$，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$Comp(P) \cup LF$的模型。

随着理论的发展，环公式概念已经被扩展到析取逻辑程序，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) 对于回答集逻辑程序P，L是P的环。L对应的析取环公式(DLF：Disjunctive Loop Formulas)，记为$DLF(L, P)$，定义为如下形式：

$$\bigvee_{p \in L} \supset \bigvee_{r \in R^{-}(L)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q) \tag{2.14}$$

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：

$$\bigvee_{p \in L} \supset \bigvee_{r \in R^{-}(L)} body(r) \tag{2.15}$$

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等人提出[33]，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) 对于回答集逻辑程序P，L是P的环。L对应的合取环公式(CLF：Conjunctive Loop Formulas)，记为$CLF(L,P)$，定义为如下形式：

$$\bigwedge_{p\in L} \supset \bigvee_{r\in R^-(L)} (body(r) \wedge \bigwedge_{q\in head(r)\setminus L} \neg q) \tag{2.16}$$

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：

$$\bigwedge_{p\in L} \supset \bigvee_{r\in R^-(L)} body(r) \tag{2.17}$$

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思想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：

$$F_L \supset \bigvee_{r\in R^-(L)} (body(r) \wedge \bigwedge_{q\in head(r)\setminus L} \neg q) \tag{2.18}$$

特别地，对于正规逻辑程序，$LF(L)$为如下形式：

$$F_L \supset \bigvee_{r\in R^-(L)} body(r) \tag{2.19}$$

定理 2.5 给定回答集逻辑程序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(2005)提出了正规逻辑程序的基本环(elementary loops)的概念[35]。之后，他们(2011)又把基本环扩展到析取逻辑程序[36]。

定义 2.23 (向外的) 给定回答集逻辑程序P，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. $head^+(r) \cap (X \backslash Y) \neq \emptyset$

3. $head(r) \cap (X \backslash Y) = \emptyset$

4. $body^+(r) \cap Y = \emptyset$

则称Y在X里是向外的(outbound)[48]。特别地，当P是正规逻辑程序时，条件1则化简为$head(r) \in Y$。

定义 2.24 (基本环) 给定回答集逻辑程序P及其环L。如果L的所有非空真子集在L里都是向外的，那么L是P的基本环(elementary loop)。

例 2.19 程序2.10有6个基本环：$\{p\}$，$\{r\}$，$\{q\}$，$\{p,r\}$，$\{q,r\}$，$\{p,q,r\}$。

实际上，Gesber等(2011)已经证明了，定义2.24的条件换成所有是基本环的子环[36]。

命题 2.2 给定回答集逻辑程序P，X是P中的环。L是基本环，当且仅当L的所有是基本环的子环在L中都是向外的[36]。

显然，把命题2.2的条件换成L的所有子环，也是可以的。

定理 2.6 给定回答集逻辑程序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和Schaub(2005)给出了正规逻辑程序的基本环的识别方法[35]，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：

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

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

$$a = head(r), a \in X,$$

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

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

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

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 subgraph)[48]。

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)[35]。

对于原子数目为n的正规逻辑程序P，其正依赖图的边数最多为$n \times (n-1)$。注意到，在求解$EC_P(X)$的过程中，每次迭代至少添加一条边。所以，在最坏的情况下，最多迭代$n \times (n-1)$次。另一方面，有向图的强连通分量可以在线性时

间内求得[46]。因此，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$O(n^3)$。然而，识别析取逻辑程序的基本环则是coNP-complete的[36]。

定义 2.26 (HEF程序) 给定一个回答集逻辑程序P，对于P的每个基本环L和每条规则r，如果$|head(r) \cap L| \leq 1$，那么该程序称为头部无基本环程序(HEF程序：Head-Elementary-loop-Free Program)[49]。

由于基本环同时也是环，所以HCF程序同时也是HEF程序。不过，反过来则不成立。

例 2.20 考虑如下的程序：

$$
\begin{aligned}
& p \leftarrow r. \\
& q \leftarrow r. \\
& r \leftarrow p,\ q. \\
& p \vee q \leftarrow .
\end{aligned}
\tag{2.20}
$$

该程序有6个环：$\{p\}$，$\{q\}$，$\{r\}$，$\{p, r\}$，$\{q, r\}$，$\{p, q, r\}$。由于$head(p \vee q \leftarrow .) \cap \{p, q, r\} = \{p, q\} > 1$，所以该程序不是HCF程序。另一方面，$\{p, r\}$和$\{q, r\}$在$\{p, q, r\}$里不是向外的，所以$\{p, q, r\}$不是基本环。因此，该程序是HEF程序。

给定HEF程序P的回答集和$sh(P)$的回答集一一对应[36]。和HCF程序相似，HEF程序可以在多项式时间内，转化为非析取逻辑程序[36]。与普通的析取逻辑程序不同，HEF程序的基本环的识别只需要多项式的时间复杂度，这是它的一个很好的性质，然而，判断一个程序是否为HEF程序则是coNP-complete的[50]。

第 3 章　基本环的再研究

本章从外部支持和环公式的角度出发，重新定义了基本环的概念。基于这种定义，本章给出正规逻辑程序的基本环的多项式时间复杂度的识别算法。与Gebser和Schaub提出的自底向上的算法[35]不同，该算法使用的是自顶向下的策略。对于析取逻辑程序，本章同样给出了多项式时间复杂度的近似算法，用于判定其环是否属于基本环的一个超集。

3.1　正规逻辑程序的基本环

本节将针对正规逻辑程序，重新定义基本环的概念，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一个推论。

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：

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

由命题2.1，可得$(\neg B \vee A \vee D) \wedge (\neg C \vee \neg B \vee D)$为真，即$CLF(L_1, P) \supset CLF(L_2, P)$为真。

由推论3.1，可以知道，对于满足$L_1 \subseteq L_2$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) 给定正规逻辑程序P，L是P的环。如果不存在P的其他环$L' \subset L$，满足$R^-(L') \subseteq R^-(L)$，那么称L是未被抑制的(unsubdued)。

直观上看，定义2.23的外向概念想要表达的意思为：原子集合Y在其超集X中是外向的，当且仅当存在$r \in P$，满足$r \in R^-(Y)$且$r \notin R^-(X)$。令L为正规逻辑程序P的环，L的所有子环L'在L里都是向外的，当且仅当$R^-(L') \nsubseteq R^-(L)$，即$L'$是未被抑制的。下面将给出详细的证明。

命题 3.1 给定正规逻辑程序P，L是P的环。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的所有子环L'在L里都是向外的。因此，对于任意子环L'，存在规则$r \in P$，满足$head(r) \in L'$，$body^+(r) \cap (L \backslash L') \neq \emptyset$，$body^+(r) \cap L' = \emptyset$。显然，规则$r$满足$r \in R^-(L')$且$r \notin R^-(L)$，因此，$R^-(L') \nsubseteq R^-(L)$。由此可知，不存其他环$L' \subset L$，满足$R^-(L') \subseteq R^-(L)$。因此$L$是未被抑制的。

$(\Leftarrow)$

由于L在P中是未被抑制的，所以任意环$L' \subset L$，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，规则$r$满足$head(r) \in L'$，$body^+(r) \cap (L \backslash L') \neq \emptyset$，$body^+(r) \cap L' = \emptyset$，即$L'$在$L$中是向外的。因此，$L$是基本环。

基于前面的推论和证明，我们可以从外部支持的角度，重新定义正规逻辑程序的基本环的概念。

定义 3.2 (正规逻辑程序的基本环) 给定正规逻辑程序P，L是P的环。如果不存在P的其他环$L' \subset L$，满足$R^-(L') \subseteq R^-(L)$，那么称L是P的基本环。

例 3.1 程序2.10的6个环的外部支持为：$R^-(\{p\}) = \{p \leftarrow .\ p \leftarrow r.\}$，$R^-(\{q\}) = \{q \leftarrow r.\}$，$R^-(\{r\}) = \{r \leftarrow p.\ r \leftarrow q.\}$，$R^-(\{p, r\}) = \{p \leftarrow .\ r \leftarrow q.\}$，$R^-(\{q, r\}) = \{r \leftarrow p.\}$，$R^-(\{p, q, r\}) = \{p \leftarrow .\}$。因此，它们都是基本环。这和例2.19的结果一样。

3.1.2 基本环的识别

根据定义3.2，下面给出正规逻辑程序基本环的识别算法。算法1采用广度优先搜索(Breadth-First-Search)[47]的方式，从环L出发，考虑其子环的性质。

与Gebser和Schaub[35]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，并且，在搜索过程中，算法1会直接排除

命题 3.2 给定正规逻辑程序P，L为P的环。算法1将在$O(n^3)$时间内返回L，或者环C，其中n是P的原子数目，C满足：

1. $C \subset L$;
2. $R^-(C) \subseteq R^-(L)$。

算法1会返回L，当且仅当L是P的基本环。

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

Input : L a loop of P，P a normal logic program
Output: if returns L，then L is a elementary loop
otherwise，L is not a elementary loop

for *each atom* $a \in L$ **do**
  $G^* :=$ the $L \backslash \{a\}$ induced subgraph of G_P
  $SCC^* :=$ the set of SCCs of G^*
  for *each* $C \in SCC^*$ **do**
    if $R^-(C) \subseteq R^-(L)$ **then**
      return C
    else
      $G_C :=$ the $C \backslash head(R^-(C) \backslash R^-(L))$ induced subgraph of G^*
      $SCC_C :=$ the set of SCC_S of G_C
      append new elements from SCC_C to SCC^*
return L

证明： 首先，如果算法1返回原子集合C，$C \neq L$，那么C是原程序的正依赖图G_P的诱导子图的强连通分量，且满足$R^-(C) \subseteq R^-(L)$。显然，C使得L不是基本环。

下面，我们将证明，如果存在P的环L'，满足$L' \subset L$，$R^-(L') \subseteq R^-(L)$，那么算法1将返回一个原子集合C'，$C' \subset L$。令原子$a \in L \backslash L'$，则L'在G_P关于原子集合$L \backslash \{a\}$的诱导子图的强连通分量中。令L'所在的强连通分量为C，那么会有如下的两种情况：

1. 如果$R^-(C) \subseteq R^-(L)$，那么算法1将会返回C，此时C就是我们在找的环；

2. 如果$R^-(C) \nsubseteq R^-(L)$，那么必定有$head(R^-(C) \backslash R^-(L)) \cap L' = \emptyset$，否则，就会存在$r \in P$，满足$head(r) \in L'$，$head(r) \in R^-(C)$。注意到，$L' \subseteq C$，所以$r \in R^-(L')$，$r \notin R^-(L)$，即$R^-(L') \nsubseteq R^-(L)$，这与我们的假定矛盾。所以，$L'$仍然在$G_P$关于原子集合$C \backslash head(R^-(C) \backslash R^-(L))$的诱导子图的强连通分量中，并且该强连通分量会被加入到SCC^*。在之后的迭代，算法1会继续分析该连通分量。所以，原子集合$C(C \subset L)$最终会被算法1返回。

综合上述，如果返回L，那就说明不存在P的其他环L'，满足$L' \subset L$，$R^-(L') \subseteq R^-(L)$。根据定义3.2，L是基本环。

最后，我们将证明算法1的时间复杂度为$O(n^3)$。令$n = Atoms(L)$，显然算法1的第1行将迭代n次。第2-10行是广度优先搜索的过程。由$R^-(C) \not\subseteq R^-(L)$可知，$head(R^-(C) \backslash R^-(L))$至少会含有一个原子，因此，搜索树的每一层都至少会删除k_i个原子，其中，k_i表示第i层的强连通分量个数。对于第i层，求该层所有节点的强连通分量的时间代价总和为$n-(k_1+...+k_{i-1})-k_i$。在最坏的情况下，搜索树会退化成单向链表的结构，此时，每层只能删除一个原子，搜索树的深度达到最大，$n-1$层。所以，$T = n \times (n + (n-1+n-2+...+1)) = n \times (n + \frac{(n-1+1)\times(n-1)}{2}) = \frac{n^3}{2} + \frac{n^2}{2}$。因此，算法1的时间复杂度为$O(n^3)$。

3.2 析取逻辑程序的基本环

上节，本文从外部支持的角度，重新定义了正规逻辑程序的基本环。本节，本文将把这种基于外部支持的定义扩展到析取逻辑程序。

3.2.1 基本环的定义

在讨论基本环之前，首先给出受限制的外部支持的概念。

定义 3.3 (受限的外部支持) 给定析取逻辑程序P，X和Y为P中的原子集合。规则$r \in P$是Y受X限制的外部支持，如果$r \in R^-_X(Y)$。其中，$R^-_X(Y) = \{r | r \in R^-(Y) \ and \ head(r) \cap (X \backslash Y) = \emptyset\}$。

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$head(r) \cap (L_2 \backslash L_1) = \emptyset$，那么有$head(r) \backslash L_1 = head(r) \backslash L_2$，所以此时可得：

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

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

有：

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

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：

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

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

由推论3.2，可以知道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，我们可以从外部支持的角度，判断一个环是否为基本环。

命题 3.3 给定析取逻辑程序P，L是P的环。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。

(⇒)

使用反证法，假设存在这样的L'，那么对于任意满足$head(r) \cap L' \neq \emptyset$，$head(r) \cap (L \setminus L') = \emptyset$，$body^+(r) \cap L' = \emptyset$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。

(⇐)

使用反证法，假设L不是基本环，那么存在非空真子集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。

基于命题3.3，我们可以重新定义析取逻辑程序的基本环。

定义 3.4 (析取逻辑程序的基本环) 给定析取逻辑程序P，L是P的环。如果不存在L的非空真子集L'，满足$R_L^-(L') \subseteq R^-(L)$，那么称$L$是$P$的基本环。

3.2.2 基本环的识别

Gebser等(2011)证明了识别析取逻辑程序的基本环是coNP-complete的[36]。虽然准确地识别析取逻辑程序的基本环的时间代价太大，但是，如果可以容忍一定的出错率，在识别过程中，可以考虑使用近似处理，那么就能在多项式时间内判断一个环L是否属于基本环的一个超集，记为$EL^*(P)$。

下面，我们给出算法2。算法2的过程和算法1相似，通过自顶向下的策略进行广度优先搜索，仅考虑环L的子环中所有值得检测的部分。

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

Input : L a loop of P，P a disjunctive logic program
Output: if returns L, then L may be a elementary loop
otherwise, L is not a elementary loop

for *each atom* $a \in L$ **do**
　$G^* :=$ the $L \backslash \{a\}$ induced subgraph of G_P
　$SCC^* :=$ the set of SCCs of G^*
　for *each* $C \in SCC^*$ **do**
　　if $R_L^-(C) \subseteq R^-(L)$ **then**
　　　return C
　　else
　　　$G_C :=$ the $C \backslash head(R_L^-(C) \backslash R^-(L))$ induced subgraph of G^*
　　　$SCC_C :=$ the set of SCCs of G_C
　　　append new elements from SCC_C to SCC^*
return L

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[46]，所以算法2的时间复杂度也是$O(n^3)$。

命题 3.4 给定析取逻辑程序P，L为P的环。算法2将在$O(n^3)$时间内返回L或者环C，其中n是P的原子数目，C满足$C \subset L$，$R_L^-(C) \subseteq R^-(L)$。如果算法2返

回C，$C \neq L$，那么L不是基本环；如果算法2返回L，那么L可能是基本环。

证明： 首先，如果算法2返回原子集合C，$C \neq L$，那么C是原程序的正依赖图G_P的诱导子图的强连通分量，且满足$C \subset L$，$R^-_L(C) \subseteq R^-(L)$。显然，$C$使得$L$不是基本环。

下面，我们将证明，如果存在P的环L'，满足$L' \subset L$，$R^-_L(L') \subseteq R^-(L)$，那么算法2可能返回一个原子集合$C'$，$C' \subset L$，也可能返回$L$。

令原子$a \in L \backslash L'$，则L'在G_P关于原子集合$L \backslash \{a\}$的诱导子图的强连通分量中。记L'所在的强连通分量为C，那么会有两种情况：

1. 如果$R^-_L(C) \subseteq R^-(L)$，那么算法2将返回$C$，此时$C$就是我们在找的环；

2. 如果$R^-_L(C) \not\subseteq R^-(L)$，那么对于任意的$r \in R^-_L(C) \backslash R^-(L)$，若$|head(r) \cap C| = 1$，则$head(r) \cap C$不可能属于$L$中的非向外的子环$L'$。否则，就会有$r \in R^-_L(L')$，这与$R^-_L(L') \subseteq R^-(L)$矛盾；若$|head(r) \cap C| > 1$，则可能会有$L' \subseteq C$，满足$r \in R^-(L')$，$head(r) \cap (L \backslash L') \neq \emptyset$，所以有$L' \cap (head(r) \cap C) \neq \emptyset$。因此，如果$|head(r) \cap C| = 1$，那么$L'$仍然在$G_P$关于原子集合$C \backslash head(R^-_L(C) \backslash R^-(L))$的诱导子图的强连通分量中，并且该强连通分量会被加入到SCC^*。在之后的迭代，算法2会继续分析该强连通分量。而如果$|head(r) \cap C| > 1$，那么L'可能不在G_P关于原子集合$C \backslash head(R^-_L(C) \backslash R^-(L))$的诱导子图的强连通分量中，然而，算法2直接把$head(R^-_L(C) \backslash R^-(L))$删除，它只会处理$G_P$关于原子集合$C \backslash head(R^-_L(C) \backslash R^-(L))$的诱导子图的强连通分量，因此，它可能会错过这个$L'$。

综合上述，算法2如果返回C，$C \neq L$，那么L不是基本环；如果返回L，那么L可能是基本环。

算法2的执行过程与算法1类似，由命题3.2可以知道，算法2的时间复杂度为$O(n^3)$。

例 3.2 考虑环$L=\{p,q,r\}$和以下析取逻辑程序P：

$$
\begin{aligned}
& p \vee q \leftarrow r. \\
& p \vee r \leftarrow q. \\
& q \vee r \leftarrow p. \\
& r \leftarrow .
\end{aligned}
\tag{3.1}
$$

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

$$
\begin{aligned}
& R_L^-(L') = \emptyset \\
& R^-(L) = \{r \leftarrow .\}
\end{aligned}
$$

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$EL^*(P)=\{L|L$是程序P的环且$ElementaryLoop^*(L,P)$返回$L\}$。显然，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，但是对于任意的基本环L和规则r，满足$|head(r)\cap L|\leq 1$，所以$EL(P)=EL^*(P)$。

3.3 本章小结

本章从外部支持的角度出发，重新定义了回答集逻辑程序的基本环的概念。基于这种定义，本章给出了两个重要的算法：算法$ElementaryLoop(L,P)$用于识别正规逻辑程序的基本环；算法$ElementaryLoop^*(L,P)$是近似算法，用来识别析取逻辑程序的基本环的超集。

第 4 章 正规逻辑程序的特征环

上一章从外部支持的角度，重新定义了回答集逻辑程序的基本环的概念，并且讨论了相关的识别算法。事实上，并非所有的基本环对于回答集的求解都是必须的。本章将针对正规逻辑程序，提出特征环的概念，同时给出一个多项式时间复杂度的算法，用于识别程序的特征环。

4.1 特征环的定义

对于合取环公式(公式2.17)的体部$\bigwedge_{p\in L} p$，实际上，我们并不需要关注环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，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：

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

特别地，如果$R^{-}(L) = \emptyset$，则：

$$\bigwedge_{p\in L} p \supset \perp \tag{4.2}$$

显然，$RLF(L, P)$是$LF(L, P)$的一个特殊情况。直观上看，$RLF(L, P)$的意思是，如果L内有外部支持支撑的原子都为真，那么至少会有一个的外部支持的体部为真。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。

首先，本文给出与$RLF(L, P)$相关的重要推论。

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：

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

根据命题2.1，有$(\neg C \vee A \vee D) \wedge (\neg C \vee \neg B \vee D)$为真，即$RLF(L_1, P) \supset RLF(L_2, P)$为真。

基于推论4.1，我们可以给出正规逻辑程序的特征环的定义。

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。如果L是基本环，同时，不存在P的其他基本环L'，满足$R^-(L') \neq \emptyset$，$R^-(L') \subset R^-(L)$，那么称L为P的特征环，

注意到，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与定理2.6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 考虑程序2.10，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$R^-(L') \neq \emptyset$，$R^-(L') \subset R^-(L)$。

4.2 特征环的识别

基于定义4.2，下面，本文将介绍正规逻辑程序的特征环的识别算法，记为$ProperLoop(L,P,S)$。

算法3采取自顶向下的策略进行广度优先搜索，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3的时间复杂度为$O(n^3)$。

命题 4.1 给定正规逻辑程序P，L是P的环。算法3，将在$O(n^3)$时间内返回L或者原子集合C，$C \subseteq S$。其中，n为P的原子的数目，C满足如下的条件：

Algorithm 3: 正规逻辑程序特征环的识别算法$ProperLoop(L, P, S)$

Input : L a loop of P，P a normal logic program，S a set of atoms
Output: if returns L, then L is a proper loop
otherwise, L is not a proper loop

$G_P^S :=$ the S induced subgraph of G_P
$SCC :=$ the set of SCCs of G_P^S
for *each* $C \in SCC$ **do**
  if $C \subset L$ *and* $R^-(C) \subseteq R^-(L)$ **then**
    return C
  else if $R^-(C) \neq \emptyset$ *and* $R^-(C) \subset R^-(L)$ **then**
    return C
  else if $R^-(C) = \emptyset$ *or* $R^-(C) = R^-(L)$ **then**
    for *each atom* $a \in C$ **do**
      $G^* :=$ the $C \backslash \{a\}$ induced subgraph of G_P^S
      $SCC^* :=$ the set of SCCs of G^*
      append new elements from SCC^* to SCC
  else
    $G_C :=$ the $C \backslash head(R^-(C) \backslash R^-(L))$ induced subgraph of G_P^S
    $SCC_C :=$ the set of SCCs of G_C
    append new elements from SCC_C to SCC
return L

- $C \subset L$，$R^-(C) \subseteq R^-(L)$，或者
- $R^-(C) \neq \emptyset$，$R^-(C) \subset R^-(L)$。

同时，算法3返回L，当且仅当L是P在S下的特征环。

证明： 首先，如果算法3返回原子集合C，$C \neq L$，$C \subseteq S$，那么C是P的环，且满足：

- $C \subset L$，$R^-(C) \subseteq R^-(L)$，或者
- $R^-(C) \neq \emptyset$，$R^-(C) \subset R^-(L)$。

下面，我们将证明，如果存在环L'，满足上述的条件之一，那么算法3将会返回原子集合C'，$C' \subseteq S$，$C' \neq L$。注意到，L'肯定在G_P^S的强连通分量里。令L'所在的强连通分量为C，那么有如下的4种情况：

1. 如果$C \subset L$，$R^-(C) \subseteq R^-(L)$，那么算法3将返回C，显然，此时，C满足条件1，C就是我们要找的环L'；

2. 如果$R^{-}(C) \neq \emptyset$，$R^{-}(C) \subset R^{-}(L)$，那么算法3将返回$C$，此时，$C$满足条件2，$C$就是我们要找的环$L'$；

3. 如果$R^{-}(C) = \emptyset$或者$R^{-}(C) = R^{-}(L)$，那么L'在G_P^S关于某个原子集合$C\backslash\{a\}$的诱导子图中，其中$a \in C$。注意到，这个强连通分量会被加入到SCC中，并在之后的迭代被分析；

4. 如果$R^{-}C \not\subseteq R^{-}(L)$，那么肯定会有$head(R^{-}(C)\backslash R^{-}(L)) \cap L' = \emptyset$，否则，存在规则$r$，满足$head(r) \in L'$，$head(r) \in R^{-}(C)$，注意到，$L' \subseteq C$，所以$r \in R^{-}(L')$，$r \notin R^{-}(L)$，即$R^{-}(L') \not\subseteq R^{-}(L)$，这与我们的假设矛盾。因此，$L'$仍然会在$G_P^S$关于原子集合$C\backslash head(R^{-}(C)\backslash R^{-}(L))$的诱导子图的强连通分量中。另一方面，这个强连通分量会被加入到SCC，并在之后的迭代被分析。

因此，算法3最终会返回一个原子集合C，$C \neq L$。综合上述，只有在不存在这样的环的情况下，算法3才会返回L，根据定义4.2，此时L为原子集合S下的特征环。

最后，我们将证明算法3的时间复杂度为$O(n^3)$，其中$n = Atoms(S)$。算法3的时间代价包括两部分：1、2行的强连通分量的求解和3-16行都广度优先搜索，其中，1、2行的时间代价为$T_1 = n$。对于3-16行，我们先忽略其中的8-12行，那么整个广度优先搜索过程，在最坏的情况下，每层只有一个强连通分量，每次处理只删除一个原子，那么，此时搜索树的深度达到最深，n层，时间代价为$T_2' = n + (n-1) + ... + 1 = \frac{n \times (n+1)}{2}$。现在我们来考虑8-12行，注意到，这里实际上把$G_P^C$变成$n$个原子个数为$|C| - 1$的子图。由于$C$是强连通分量，所以删除$C$中的一个原子，肯定会引入额外的外部支持。因此，8-12行的处理实际上只会发生一次。在最坏的情况下，对于第一个$C \in SCC$，就有$R^{-}(C) \subseteq R^{-}(L)$，此时，时间代价$T_2 = n \times (n + (n-1) + ... + 1) = \frac{n^2 \times (n+1)}{2}$。因此，整个算法的时间代价$T = T_1 + T_2 = n + \frac{n^3}{2} + \frac{n^2}{2}$，即算法3的时间复杂度为$O(n^3)$。

命题 4.2 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$R^{-}(L') \nsubseteq R^{-}(L)$。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3去对程序的每个环进行过滤。显然，这样做的效率并不高，毕竟环的数目随着程序规模的增长可能会出现指数爆炸。利用命题4.2，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。

Algorithm 4: 正规逻辑程序所有特征环的识别算法$ProperLoops(P, S)$

Input : P a disjunctive logic program，S a set of atoms
Output: a set of all proper loops of P

$Loops := \emptyset$
$G_P^S :=$ the S induced subgraph of G_P
$SCC :=$ the set of SCCs of G_P^S
for *each* $C \in SCC$ **do**
    $C^* := ProperLoop(C, P, S)$
    if $C^* = C$ **then**
        append C to $Loops$
        for *each* $a \in head(R^{-}(C))$ **do**
            $G_C :=$ the $C \backslash \{a\}$ induced subgraph of G_P^S
            $SCC_C :=$ the set of SCCs of G_C
            append new elements from SCC_C to SCC
    else
        for $a \in C$ **do**
            $G_C :=$ the $C \backslash \{a\}$ induced subgraph of G_P^S
            $SCC_C :=$ the set of SCCs of G_C
            append new elements from SCC_C to SCC
return $Loops$

给定正规逻辑程序P，算法4将计算程序P在原子集合S下的所有特征环。与之前的算法类似，算法4也是基于自顶向下的策略，它将会考虑所有值得检测的环。

命题 4.3 给定正规逻辑程序P，S为P中的原子集合。算法4返回P在S下的所有特征环。

证明： 首先，对于算法4返回的集合中的每一个L，由于L通过了$ProperLoop(L,P,S)$的检测，根据命题4.1，L是P在S下的特征环。

下面，我们将证明，如果存在S下的特征环L'，那么算法4返回的集合中将包含有L'。

首先，L'必然在G_P^S的某个强连通分量中。记L'所在强连通分量为C，那么会有以下的情况：

1. 如果$ProperLoop(C,P,S)$返回C，那么C是特征环，C会被加入到结果的集合中。此时，若$L'=C$，那么L'已经在结果集合中；若$L'\neq C$，那么由命题4.2可以知道，$head(R^-(L'))\not\subseteq L$，否则，$L'$不是特征环，矛盾。因此，$L'$仍然在$G_P^S$关于某个原子集合$C\backslash\{a\}$的诱导子图的强连通分量中，其中$a\in head(R^-(C))$。注意到，这个强连通分量会被加入到$SCC$，并在之后的迭代被分析；

2. 如果$ProperLoop(C,P,S)$返回C^*，$C^*\neq C$，那么C不是特征环。此时，L'在G_P^S关于某个原子集合$C\backslash\{a\}$的诱导子图的强连通分量中，其中$a\in C$。注意到，这个强连通分量会被加入到SCC中，并在之后的迭代被分析；

因此，特征环L'最终会在算法4返回的集合中。

综合上述，算法4返回P在S下的所有特征环。

4.3 本章小结

在上一章的基础上，本章从外部支持和环公式的关系出发，提出了正规逻辑程序的特征环的概念，并证明了特征环可以在基本环的基础上，进一步地删除更多的环。本章还讨论了两个重要的算法：算法$ProperLoop(L,P,S)$可以判断某环是否为正规逻辑程序的特征环；算法$ProperLoops(P,S)$可以求出正规逻辑程序的所有特征环。

第 5 章 析取逻辑程序的特征环

上一章就正规逻辑程序，提出了特征环的概念。本章将把特征环的概念扩展到析取逻辑程序上去。考虑到识别析取逻辑程序的基本环和特征环都是coNP-complete的，本章提出了弱基本环和弱特征环的概念。基于这些环的概念和性质，本章还提出了HPF程序、HWEF程序和HWPF程序，并讨论了它们之间的关系。

5.1 特征环

本节，我们将把特征环的概念扩展到析取逻辑程序。

5.1.1 特征环的定义

析取逻辑程序和正规逻辑程序不同，它的头部可能会有多个原子。另一方面，与正规逻辑程序类似，对于环L来说，需要关注的原子应该是L与其外部支持的头部的交集，即$head(R^-(L)) \cap L$部分，因此，对于析取逻辑程序，可以使用$\bigwedge_{p \in head(R^-(L)) \cap L} p$去替换公式2.16中的$\bigwedge_{p \in L} p$，从而得到另一种形式的环公式。

定义 5.1 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：

$$\bigwedge_{p \in head(R^-(L)) \cap L} p \supset \bigvee_{p \in R^-(L)} (body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q) \tag{5.1}$$

特别地，如果$R^-(L) = \emptyset$，则：

$$\bigwedge_{p \in L} p \supset \bot \tag{5.2}$$

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，加入额外的限制，排除更多的环。

下面，先证明与$RLF(L, P)$相关的一个重要推论。该推论是特征环的一个重要基础。

推论 5.1 给定析取逻辑程序P，L_1和L_2为P的环。如果$R^-(L_1) \neq \emptyset$，$R^-(L_2) \neq \emptyset$，$head(R^-(L_1)) \cap (L_1 \cup L_2) \subseteq head(R^-(L_2)) \cap L_2$，$R^-_{L_2}(L_1) \subseteq R^-(L_2)$，那么$RLF(L_1, P) \supset RLF(L_2, P)$。

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

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

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$head(r) \cap L_1 \subseteq L_2$，$head(r) \cap (L_2 \backslash L_1) = \emptyset$，从而有$head(r) \cap L_1 = head(r) \cap L_2$。此时有：

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

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

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

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

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

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

基于推论5.1，我们可以给出析取逻辑程序的特征环的定义。

定义 5.2 (析取逻辑程序的特征环) 给定析取逻辑程序P，L为P的环。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'，满足$R^-(L') \neq \emptyset$，$head(R^-(L')) \cap (L' \cup L) \subseteq head(R^-(L)) \cap L$，$R^-_L(L') \subset R^-(L)$。

那么称L为P的特征环

例 5.1 给定如下的程序：

$$
\begin{aligned}
& p \vee q \leftarrow r. \\
& r \leftarrow p. \\
& r \leftarrow q. \\
& p \leftarrow .
\end{aligned}
\tag{5.3}
$$

程序5.3的正依赖图如图2.1所示，该程序有6个环：$\{p\}$，$\{r\}$，$\{q\}$，$\{p,r\}$，$\{q, r\}$，$\{p,q,r\}$。其中，有5个基本环：$\{p\}$，$\{r\}$，$\{q\}$，$\{p,r\}$，$\{q,r\}$。对于环$\{p,q,r\}$，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。

因此，该程序有4个特征环：$\{p\}$，$\{q\}$，$\{p,r\}$，$\{q,r\}$。

对于所有环的外部支持都不为空的程序，特征环的定义以及识别过程中的处理都会显得更加方便简洁。

定义 5.3 (简化的) 给定析取逻辑程序P。如果P不存在任何外部支持为空的环，那么称P是简化的$(simplified)$。

任意的析取逻辑程序P都可以转化为与其等价的简化析取逻辑程序，记为$simp(P)$，具体操作如下：

1. 删除规则r，其中，r和P的某个环L满足$body^+(r) \cap L \neq \emptyset$，$R^-(L) = \emptyset$；

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$not\ p$，则删除$not\ p$，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$simp(P)$还不是简化的，那么我们需要继续对结果进行化简，即$simp(simp(P))$，直到返回结果为简化的为止。

命题 5.1 给定析取逻辑程序P。P的回答集与$simp(P)$的一一对应。

证明： 对于满足$body^+(r) \cap L \neq \emptyset$，$R^-(L) = \emptyset$的规则$r$和环$L$，由环公式理论可以知道，$L$中的原子是不可能属于$P$的任何回答集的，因此$r$的体部不可能成立，即$r$是多余的。所以$simp(P)$的操作1不会对程序的回答集造成影响；另一方面，由于L中原子不属于P的任何回答集，由失败即否定的语义，我们知道任意$p \in L$都为假，所以$simp(P)$的操作2不会对程序的回答集造成影响。综合上述，P的回答集和$simp(P)$的一一对应。

例 5.2 给定如下的程序P：

$$\begin{aligned} &p \vee q \leftarrow r. \\ &r \leftarrow q. \\ &p \leftarrow . \end{aligned} \tag{5.4}$$

则$simp(P) = \{p \leftarrow .\}$。显然，$P$和$simp(P)$的回答集都为$\{\{p\}\}$。

定义 5.4 给定简化的程序P，L是P的环。如果L满足：

1. 不存在L的子环C，满足$R_L^-(C) \subseteq R^-(L)$，同时，

2. 不存在原子集合C，满足$head(R^-(C)) \cap (C \cup L) \subseteq head(R^-(L)) \cap L$，$R_L^-(C) \subset R^-(L)$。

那么L是特征环。

5.1.2 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。下面将给出多项式时间复杂度的近似算法5，记为$PL^*(L, P)$，它在识别过程中采用了近似处理，可以判断一个环是否属于特征环的超集。

用$PL(P)$表示程序P的所有特征环，$PL^*(P)$表示$PL^*(L,P)$返回为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$PL(P) = PL^*(P)$。

Algorithm 5: 析取逻辑程序特征环的超集的识别算法$PL^*(L,P)$

Input : L a loop of P，P a disjunctive logic program
Output: if returns L, then L is in $PL^*(P)$
otherwise, L is not in $PL^*(P)$

SCC := the SCCs of G_P
for *each* $C \in SCC$ **do**
    if $C \subset L$ *and* $R_L^-(C) \subseteq R^-(L)$ **then**
        return C
    else if $head(R^-(C)) \cap (C \cup L) \subseteq head(R^-(L)) \cap L$ *and* $R_L^-(C) \subset R^-(L)$ **then**
        return C
    else if $R_L^-(C) = R^-(L)$ **then**
        for *each atom* $a \in C$ **do**
            G^* := the $C \backslash \{a\}$ induced subgraph of G_P
            SCC^* := the set of SCCs of G^*
            append new elements from SCC^* to SCC
    else if $head(R^-(C)) \cap (C \cup L) \nsubseteq head(R^-(L)) \cap L$ *and* $C \nsubseteq L$ **then**
        $C' := C \backslash ((head(R^-(C)) \cap C) \backslash (head(R^-(L)) \cap L))$
        G' := the C' induced subgraph of G_P
        SCC' := the set of SCCs of G'
        append new elements from SCC' to SCC
    else
        G_C := the $C \backslash head(R_L^-(C) \backslash R^-(L))$ induced subgraph of G_P
        SCC_C := the set of SCCs of G_C
        append new elements from SCC_C to SCC
return L

命题 5.2 给定简化的析取逻辑程序P，L是P的环。算法5将在$O(n^3)$时间内返回L或者环C，其中n是程序的原子的数目，C满足如下的条件：

- $C \subset L$，$R_L^-(C) \subseteq R^-(L)$，或者
- $head(R^-(C)) \cap (C \cap L) \subseteq head(R^-(L)) \cap L$，$R_L^-(C) \subset R^-(L)$。

如果5返回C，$C \neq L$，那么L不是特征环；如果返回L，那么L可能是特征环，也可能不是。

证明： 首先，如果5返回C，$C \neq L$，那么C是原程序的正依赖图G_P的诱导子图的强连通分量，且满足$C \subset L$，$R^-_L(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap (C \cap L) \subseteq head(R^-(L)) \cap L$，$R^-_L(C) \subset R^-(L)$。显然，$C$使得$L$不是特征环。

下面，我们将证明，如果存在P的环L'，满足$L' \subset L$，$R^-_L(L') \subseteq R^-(L)$，或者$head(R^-(L')) \cap (L' \cap L) \subseteq head(R^-(L)) \cap L$，$R^-_L(L') \subset R^-(L)$，那么算法5可能返回一个原子集合$C'$，$C' \neq L$，也可能返回$L$。

首先，L'必然在G_P的某个强连通分量中。记L'所在强连通分量为C，那么会有以下的情况：

1. 如果$C \subset L$，$R^-_L(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap (C \cap L) \subseteq head(R^-(L)) \cap L$，$R^-_L(C) \subset R^-(L)$，那么算法5将返回$C$，此时$C$就是我们在找的环；

2. 如果$R^-_L(C) \subseteq R^-(L)$，那么$L'$在$G_P$关于某个原子集合$C \backslash \{a\}$的诱导子图中，其中$a \in C$。注意到，这个强连通分量会被加入到$SCC$中，并在之后的迭代被分析；

3. 如果$head(R^-(C)) \cap (C \cup L) \not\subseteq head(R^-(L)) \cap L$且$C \not\subseteq L$，那么有$((head(R^-(C)) \cap C) \backslash (head(R^-(L)) \cap L)) \cap L' = \emptyset$，因此，$L'$仍然会在$G_P$关于原子集合$C \backslash ((head(R^-(C)) \cap C) \backslash (head(R^-(L)) \cap L))$的诱导子图的强连通分量中。另一方面，这个强连通分量会被加入到SCC，并在之后的迭代被分析；

4. 其他情况，此时$R^-_L(C) \not\subseteq R^-(L)$。对于任意的规则$r \in R^-_L(C) \backslash R^-(L)$，若$|head(r) \cap C| = 1$，则必定有$r \notin R^-_L(L')$。否则，与$R^-_L(L') \subseteq R^-(L)$矛盾，此时有$L' \cap (head(r) \cap C) = \emptyset$；若$|head(r) \cap C| > 1$，则可能会有$r \in R^-(L')$，$head(r) \cap (L \backslash L') \neq \emptyset$，所以有$L' \cap (head(r) \cap C) \neq \emptyset$。因此，如果$|head(r) \cap C| = 1$，那么$L'$仍然在$G_P$关于原子集合$C \backslash (head(R^-_L(C) \backslash R^-(L))$的诱导子图的强连通分量中，并且该连通分量会被加入到SCC。在之后的迭代，算法5会继续分析该强连通分量。而如果$|head(r) \cap C| > 1$，那么L'可能不在G_P关于原子集合$C \backslash head(R^-_L(C) \backslash R^-(L))$的诱导子图的强连通分量中，然而，算法5同样把$head(R^-_L(C) \backslash R^-(L))$删除了，在之后的迭代，它只会处

理到G_P关于原子集合$C\backslash head(R_L^-(C)\backslash R^-(L))$的诱导子图的强连通分量，因此，它可能会错过这个$L'$。

综合上述，如果算法5返回C，$C\neq L$，那么L不是特征环；如果返回L，那么L可能是特征环。

算法5的执行过程与算法3类似，因此，它是时间复杂度也为$O(n^3)$。

5.2 弱基本环和弱特征环

在上一章和本章上一节，本文分别讨论过析取逻辑程序的基本环和特征环的概念。针对识别析取逻辑程序的基本环和特征环都是coNP-complete的，本节将提出弱基本环和弱特征环的概念，同时，给出多项式时间复杂度的识别算法。

5.2.1 弱基本环的定义

识别析取逻辑程序的基本环是coNP-complete的，但是，我们可以降低基本环的要求，由此提出限制小一点的弱基本环。下面，本文将提出弱特征环概念，并给出识别弱特征环的算法。

首先，本文给出与弱基本环相关的环公式推论。

推论 5.2 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$R^-(L_1)\subseteq R^-(L_2)$，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5.2定义一个弱化版本的基本环。

定义 5.5 **(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，称L是P的弱基本环(weak elementary loop)。

命题 5.3 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3.3，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5.5，$L$是$P$的弱基本环。

例 5.3 对于程序5.3，环$L = \{p, q, r\}$是弱基本环，但不是基本环。

5.2.2 弱基本环的识别

判断一个环是否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6，记为$WEL(L, P)$。显然，算法6跟算法2一样，时间复杂度为$O(n^3)$。

Algorithm 6: 析取逻辑程序弱基本环的识别算法$WEL(L, P)$

Input : L a loop of P，P a disjunctive logic program
Output: if returns L, then L is a weak elementary loop
otherwise, L is not a weak elementary loop

for *each atom* $a \in L$ **do**
  $G^* :=$ the $L \backslash \{a\}$ induced subgraph of G_P
  $SCC^* :=$ the set of SCCs of G^*
  for *each* $C \in SCC^*$ **do**
    if $R^-(C) \subseteq R^-(L)$ **then**
      return C
    else
      $G_C :=$ the $C \backslash head(R^-(C) \backslash R^-(L))$ induced subgraph of G^*
      $SCC_C :=$ the set of SCCs of G_C
      append new elements from SCC_C to SCC^*
return L

命题 5.4 给定析取逻辑程序P，L是P的环。算法6将在$O(n^3)$时间内返回L或者环C，其中，n为P的原子数目，C满足$C \subset L$，$R^-(C) \subseteq R^-(L)$。当且仅当L是P的弱基本环，算法6才会返回L。

证明： 首先，如果算法6返回原子集合C，$C \neq L$，那么C满足$C \subset L$，$R^{-}(C) \subseteq R^{-}(L)$，同时，$C$是环。

下面，我们将证明，如果存在环L'，满足$L' \subset L$，$R^{-}(L') \subseteq R^{-}(L)$，那么算法6将返回一个原子集合$C'$，$C' \subset L$。令原子$a \in L \backslash L'$，则$L'$在$G_P$关于原子集合$L \backslash \{a\}$的诱导子图的强连通分量中。记$L'$所在强连通分量为$C$，那么会有如下的两种情况：

1. 如果$R^{-}(C) \subseteq R^{-}(L)$，那么算法6将返回$C$，此时$C$就是我们在找的环；

2. 如果$R^{-}(C) \nsubseteq R^{-}(L)$，那么必定有$head(R^{-}(C) \backslash R^{-}(L)) \cap L' = \emptyset$，否则，就会存在$r \in P$，满足$head(r) \cap L' \neq \emptyset$，$head(r) \cap C \neq \emptyset$。另一方面，$L' \subseteq C$，所以$r \in R^{-}(L')$，$r \notin R^{-}(L)$，即$R^{-}(L') \nsubseteq R^{-}(L)$，这与我们的假设矛盾。所以$L'$仍然在$G_P$关于原子集合$C \backslash head(R^{-}(C) \backslash R^{-}(L))$的诱导子图的强连通分量中。注意到，该强连通分量会被加入到SCC^{*}中。在之后的迭代，算法6会继续分析该强连通分量。所以，原子集合$C \subset L$最终会被算法6返回。

综合上述，如果返回L，那就说明不存在P的其他环L'，满足$L' \subset L$，$R^{-}(L') \subseteq R^{-}(L)$。根据定义5.5，$L$是弱基本环。

算法6的执行过程与算法1类似，因此，它的时间复杂度也是$O(n^3)$。

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 5.5 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$head(r) \cap L' \neq \emptyset$，$head(r) \cap (L \backslash L') \neq \emptyset$，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：$EL(P) = EL^{*}(P) = WEL(P)$。

5.2.3 弱特征环的定义

识别析取逻辑程序的特征环是coNP-complete的，但是，我们可以降低特征环的要求，由此提出限制小一点的弱特征环。下面，本文将提出弱特征环概念，并给出多项式时间复杂度的算法，用于识别弱特征环。

首先，本文给出与弱特征环相关的环公式推论。

推论 5.3 给定析取逻辑程序P，L_1和L_2是P的环。如果$R^-(L_1) \neq \emptyset$，$head(R^-(L_1))\cap L_1 \subseteq head(R^-(L_2))\cap L_2$，$R^-(L_1) \subseteq R^-(L_2)$，那么$RLF(L_1, P) \supset RLF(L_1, P)$。

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$head(r)\backslash L_2 \subseteq head(r)\backslash L_1$。因此，$\bigwedge_{q\in head(r)\backslash L_1} \neg q \supset \bigwedge_{q\in head(r)\backslash L_2} \neg q$。所以，我们有：

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

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 5.6 (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 5.4 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$R^-(\{r\}) = \{r \leftarrow p. \ r \leftarrow q.\}$，$R^-(\{r, q\}) = \{r \leftarrow p.\}$，所以它不是弱特征环。

定理 5.1 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和定理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$DLF(L,P)$；

命题 5.6 给定析取逻辑程序P，L是P的环。如果L是P的特征环，那么L也是P的弱特征环。

证明： 因为L是P的特征环，所以由定义5.2可知，L也是基本环。又由命题5.3，L也是弱基本环，条件1成立。另一方面，由于L是P的特征环，所以对任意弱基本环L，有$head(R^-(L')) \cap (L' \cup L) \subseteq head(R^-(L)) \cap L$，$R^-_L(L') \subset R^-(L)$，因此有$head(R^-(L')) \cap L' \subseteq head(R^-(L)) \cap L$，$R^-(L') \subset R^-(L)$。

综合上述，L是弱特征环。

利用简化的的概念，我们可以更简洁地去定义弱特征环。

定义 5.7 (简化的程序的弱特征环) 给定简化的析取逻辑程序P，环L是P的弱特征环，如果不存在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$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$。

5.2.4 弱特征环的识别

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5中的$R^-_L(C)$替换成$R^-(C)$、$head(R^-(C)) \cap (C \cup L)$替换成$head(R^-(C)) \cap C$，就可以得到用于识别弱特征环的算法7。

命题 5.7 给定简化的析取逻辑程序P，L为P的环。算法7将在$O(n^3)$时间内返回L或者环C，其中，n为P的原子数目，C满足：

1. $C \subset L$，$R^-(C) \subseteq R^-(L)$，或者

2. $head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$。

算法7返回L当且仅当L是P的弱特征环。

Algorithm 7: 析取逻辑程序的弱特征环的识别算法$WPL(L,P)$

Input : L a loop of P，P a disjunctive logic program
Output: if returns L, then L is a weak proper loop

$SCC :=$ the set of SCCs of G_P
for *each* $C \in SCC$ **do**
    if $C \subset L$ *and* $R^-(C) \subseteq R^-(L)$ **then**
        return C
    else if $head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$ *and* $R^-(C) \subset R^-(L)$ **then**
        return C
    else if $R^-(C) = R^-(L)$ **then**
        for *each atom* $a \in C$ **do**
            $G^* :=$ the $C\backslash\{a\}$ induced subgraph of G_P
            $SCC^* :=$ the set of SCCs of G^*
            append new elements from SCC^* to SCC
    else if $head(R^-(C)) \cap C \nsubseteq head(R^-(L)) \cap L$ *and* $C \nsubseteq L$ **then**
        $C' := C\backslash((head(R^-(C)) \cap C)\backslash(head(R^-(L)) \cap L))$
        $G' :=$ the C' induced subgraph of G_P
        $SCC' :=$ the set of SCCs of G'
        append new elements from SCC' to SCC
    else
        $G_C :=$ the $C\backslash head(R^-(C)\backslash R^-(L))$ induced subgraph of G_P
        $SCC_C :=$ the set of SCCs of G_C
        append new elements from SCC_C to SCC
return L

证明： 首先，如果算法7返回原子集合C，$C \neq L$，那么C是原程序的正依赖图G_P的某个诱导子图的强连通分量，且满足$C \subset L$，$R^-(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$。显然，$C$是$P$的环，且$C$使得$L$不是弱特征环。

下面，我们将证明，如果存在P的环L'，满足$C \subset L$，$R^-(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$，那么算法7将会返回一个原子集合$C$，$C \neq L$。

首先，L'必然在G_P的某个强连通分量中。记L'所在的强连通分量为C，那么会有以下的情况：

1. 如果$C \subset L$，$R^-(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$，那么算法7将返回$C$，此时$C$就是我们在找的环；

2. 如果$R^-(C) \subseteq R^-(L)$，那么L'在G_P关于某个原子集合$C \setminus \{a\}$的诱导子图的强连通分量中，其中$a \in C$。注意到，这个强连通分量会被加入到SCC中，并在之后的迭代被分析；

3. 如果$head(R^-(C)) \cap C \not\subseteq head(R^-(L)) \cap L$且$C \not\subseteq L$，那么肯定有$((head(R^-(C) \cap C)) \setminus (head(R^-(L) \cap L))) \cap L' = \emptyset$，否则，就会存在规则$r \in P$，满足，$r \in R^-(L')$，$r \notin R^-(L)$，这和我们的假设矛盾。因此，$L'$仍然在$G_P$关于原子集合$C \setminus ((head(R^-(C) \cap C)) \setminus (head(R^-(L) \cap L)))$的诱导子图的强连通分量中。注意到，这个强连通分量会被加入到SCC中，并在之后的迭代被分析；

4. 其他情况，此时有$R^-(C) \not\subseteq R^-(L)$。那么肯定有$head(R^-(C) \setminus R^-(L)) \cap L' = \emptyset$，否则存在规则$r \in R^-(C) \setminus R^-(L)$，满足$r \in R^-(L')$，由此可知$R^-(L') \not\subseteq R^-(L)$，这和我们的假设矛盾。因此，$L'$仍然在$G_P$关于原子集合$C \setminus head(R^-(C) \setminus R^-(L))$的诱导子图的强连通分量中。注意到，这个强连通分量会被加入到SCC中，并在之后的迭代被分析；

因此，算法7最终会返回一个原子集合C，$C \neq L$。

综合上述，如果返回L，那就说明不存在P的其他环L'，满足足$C \subset L$，$R^-(C) \subseteq R^-(L)$，或者$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$，$R^-(C) \subset R^-(L)$。根据定义5.7，$L$是弱特征环。

算法7的执行过程与算法5类似，因此，它的时间复杂度也是$O(n^3)$。

各种环种类之间的关系如图5.1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：

$$\begin{tikzcd} & \mathrm{EL} \arrow[r] & \mathrm{EL}^* \arrow[dr] & \\ \mathrm{PL} \arrow[ur] \arrow[dr] & & & \mathrm{WEL} \\ & \mathrm{PL}^* \arrow[r] & \mathrm{WPL} \arrow[ur] & \end{tikzcd}$$

图 5.1: 环种类之间的关系

5.3 HPF程序、HWEF程序和HWPF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 5.8 (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$|head(r) \cap L| \leq 1$。

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 5.8 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$|head(r) \cap L| > 1$。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程序是coNP-complete的。

同样，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 5.9 (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5.9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[36]，所以HWEF程序也是一样。然而，判断一个程序是否为HWEF程序依然是coNP-complete的。

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 5.9 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$head(r) \cap L' \neq \emptyset$，$body^+(r) \cap L' = \emptyset$，$body^+(r) \cap (E \backslash L') \neq \emptyset$。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 5.10 给定析取逻辑程序P，L是P的环，E是L的非空真子集。如果L是弱基本环，那么存在规则$r \in P$，满足$body^+(r) \cap E \neq \emptyset$，$body^+(r) \cap (L \backslash E) = \emptyset$，$head(r) \cap (L \backslash E) \neq \emptyset$。

证明： 使用反证法，假设不存在这样的规则$r \in P$，那么有$L \backslash E \subset L$，$R^-(L \backslash E) \subseteq R^-(L)$，根据定义5.5，$L$不是弱基本环，矛盾。

根据命题5.10，可以知道，对于非空原子集合E，如果不存在环L和规则$r \in P$，满足$body^+(r) \cap E \neq \emptyset$，$body^+(r) \cap (L \backslash E) = \emptyset$，$head(r) \cap (L \backslash E) \neq \emptyset$，那么$E$的所有非空真超集都不会是弱基本环。基于这一结论，本文给出算法8，用于判断对于非空原子集E，是否不存在弱基本环L，满足$E \subset L$。

命题 5.11 给定的析取逻辑程序P和任意非空原子集E。算法8将在$O(nm)$的时间内返回真或假，其中，m为P的规则数目。当且仅当不存在P的弱基本环L，满足$E \subset L$，算法8才返回假。

Algorithm 8: 判断是否不存在包含某环的弱基本环的算法$EWEL(P, E)$

Input : P a disjunctive logic program，E a loop of P

Output: if returns false, then there is no elementary loop $L \supset E$

$G :=$ the positive dependency graph of P

if *there does not exist a SCC* C *of* G *s.t.* $E \subseteq C$ **then**

  return false

$C :=$ the SCC of G s.t. $E \subseteq C$

$R_E := \{r | r \in P$ and $body^+(r) \cap E \neq \emptyset\}$

for *each* $r \in R_E$ **do**

  $G_r :=$ the $C \backslash (body^+(r) \backslash E)$ induced subgraph of G

  if *there exists a SCC* C_r *of* G_r *s.t.* $E \subseteq C_r$ *and* $head(r) \cap (C_r \backslash E) \neq \emptyset$ **then**

    return true

return false

证明： 首先，如果算法8返回真，那么表示存在G关于某个原子集合的诱导子图的强连通分量C和规则$r \in P$，满足$E \subseteq C$，$body^+(r) \cap E \neq \emptyset$，$body^+(r) \cap (C \backslash E) = \emptyset$，$head(r) \cap (C \backslash E) \neq \emptyset$。

下面，我们将证明，如果存在L和规则$r \in P$，满足$E \subseteq L$，$body^+(r) \cap E \neq \emptyset$，$body^+(r) \cap (L \backslash E) = \emptyset$，$head(r) \cap (L \backslash E) \neq \emptyset$，那么算法8将返回真。注意到，$L$肯定在$G$的某个强连通分量中，记此强连通分量为$C$。那么存在规则$r \in P$，$body^+(r) \cap E \neq \emptyset$。同时，必然有$head(body^+(r) \backslash E) \cap L = \emptyset$。否则就导致$body^+(r) \cap (L \backslash E) \neq \emptyset$，矛盾。因此$L$仍然在$G$关于原子集合$C \backslash (body^+(r) \backslash E)$的诱导子图的强连通分量中。注意到，算法8会继续检测这些强连通分量。并且，如果某个强连通分量C_r满足$E \subseteq C_r$，$head(r) \cap (C_r \backslash E)$，那么算法8就会返回真，此时，$C_r$就是我们在找的环。

综合上述，只有不存在符合条件的环，算法8才会返回假。根据命题5.10，可以知道，不存在P的弱基本环L，满足$E \subset L$。

最后，我们将证明算法8的时间复杂度为$O(nm)$，其中n为程序的原子数目，m为规则数目。算法8主要有两部分的操作：求G的强连通分量和遍历所有规则并求诱导子图的强连通分量。所以，$T = n + m \times n = n \times (m + 1)$。因此，算法8的时间复杂度为$O(nm)$。

利用算法6和算法8，本文在HWEF程序的判断过程中，使用近似处理，那么就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*程序。下面，本文给出算法9。

Algorithm 9: 判断程序是否属于HWEF*程序的算法$HWEF^*(P)$

Input : P a disjunctive logic program
Output: if returns true, then P is a HWEF program
otherwise, P may be a HWEF program

$S := \{\{a,b\} |$there is a rule $r \in P s.t. \{a,b\} \subseteq head(r)\}$
for *each* $E \in S$ **do**
  if E *is a weak elementary loop of* P **then**
    return false
  if $EWEL(P,E)$ *returns true* **then**
    return false
return true

命题 5.12 给定析取逻辑程序P。算法9将在$O(mn^3)$内返回真或者假。如果算法9返回真，那么P是HWEF程序。

证明： 首先，算法9生成$S = \{\{a,b\} |$存在规则$r \in P$，使得$\{a,b\} \subseteq head(r)\}$。令集合$S' = \{E | E = head(r) \cap L$，存在$r \in P$，弱基本环$L$，满足$|head(r) \cap L| > 1\}$。显然，对于任意的$E' \in S'$，存在$E \in S$，满足$E \subseteq E'$。因此，如果不存在弱基本环$L$，满足$E \subseteq L$，那么$P$是HWEF程序。

下面，我们将证明，如果存在弱基本环L，满足$E \subseteq L$，那么算法9，将返回假。首先，对于$E \in S$，算法9进行如下的处理：

1. 使用算法6判断E是否为弱基本环。如果E是弱基本环，那么E就是我们找的弱基本环，此时$L = E$，算法9返回假；

2. 使用算法8判断是否存在弱基本环L，满足$E \subset L$。注意到，算法8只能保证它返回为假时，就不存在这样的L，如果返回为真，那么则可能存在，对于这种情况，算法9直接返回假。

因此，只要存在L，满足$E \subseteq L$，此时P为非HWEF程序，那么算法9最终会返回假。另一方面，如果不存在这样的L，此时P为HWEF程序，那么算法9可能返回真，可能返回假。

综合上述，如果算法9返回真，那么P是HWEF程序。

下面，我们将证明算法9的时间复杂度。算法9包括两部分的操作：生成S和验证S中的元素。其中，最坏情况下，生成S的时间代价$T_1 = n \times (n-1)$，其中n为原子数目。注意到，$|S| = T_1 = n \times (n-1)$。因此，验证$S$中的元素的时间代价$T_2 = |S| \times (|E|^3 + nm) = n \times (n-1) \times (2^3 + nm)$，其中$m$为规则数目。所以，算法9的时间代价为$T = T_1 + T_2 = n \times (n-1) \times (9 + nm)$，即$O(mn^3)$。

对于非HWEF程序，算法9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。

例 5.5 考虑如下的程序P：

$$
\begin{aligned}
&p \vee q \leftarrow r. \\
&r \leftarrow p, q. \\
&p \leftarrow .
\end{aligned}
\tag{5.5}
$$

程序P有6个环：$\{p\}$，$\{q\}$，$\{r\}$，$\{p,r\}$，$\{r,q\}$，$\{p,q,r\}$。由于$R^-(\{p,q,r\}) = \{p \leftarrow .\}$，$R^-(\{r,q\}) = \emptyset$，所以$\{p,q,r\}$不是弱基本环。

因此，弱基本环有：$\{p\}$，$\{q\}$，$\{r\}$，$\{p,r\}$，$\{r,q\}$。

另一方面，P是HWEF程序。但是，对于$E = \{p,q\}$，$EWEL(P,E)$返回真，所以$HWEF^*(P)$返回假。所以，P是HWEF程序，但不是HWEF*程序。

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

定义 5.10 (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$|head(r) \cap L| \leq 1$。

一个HWPF程序同时也会是HPF程序，一个HWEF程序同时也会是HWPF程序。各种程序类别之间的关系如图5.2所示，其中，→表示子集关系：

$$
\begin{tikzcd}
 & & & \mathrm{HEF} \arrow[dr] & \\
\mathrm{HCF} \arrow[r] & \mathrm{HWEF}^{*} \arrow[r] & \mathrm{HWEF} \arrow[ur] \arrow[dr] & & \mathrm{HPF} \\
 & & & \mathrm{HWPF} \arrow[ur] &
\end{tikzcd}
$$

图 5.2: 各种程序类别之间的关系

5.4 本章小结

本章，我们将把特征环的概念扩展到析取逻辑程序上去，同时我们提出了弱基本环和弱特征环的概念。基于这些环的概念和性质，我们讨论了它们的识别算法。最后，我们还提出了HPF、HWEF、HWPF等程序类别，并讨论了它们之间的关系。

第 6 章　实验过程以及结果分析

基于前几章的理论，本章进行了三个实验。实验一针对正规逻辑程序，对比基本环和特征环的数目以及识别效率；实验二针对析取逻辑程序，对比程序的各种环的数目；实验三检测实际用例的程序中，HWEF*程序所占的比例。

6.1　实验工具

词法分析器Flex和语法分析器Bison[1]是本文的实验所采用的预处理工具，它们所生成的程序能够处理结构化的输入。

简单来说，Flex可以把输入分割成一个个有意义的词块，称为记号(token)；而Bison则用来确定这些记号是如何彼此关联的。例如，有如下的C代码片段：

$$alpha = beta + gamma;$$

Flex把这段代码分解为以下记号：“$alpha$”、“$=$”、“$beta$”、“$+$”、“$gamma$”和“;”。接着语法分析器确定了$beta + gamma$是一个表达式，并把这个表达式的计算结果赋值给$alpha$。

本文中所用到的其他工具如表6.1所示。

表 6.1: 编程工具信息

工具	描述
开发语言	c/c++
编译器	g++ 4.8.2
词语分析器	flex 2.5.35
语法分析器	bison 3.0.2

6.2　程序框架

本文的实验的程序都是采用分层的结构，它们的程序框架类似，主要有4个模块：词法分析模块、语法分析模块、正依赖图模块和主任务模块，如图6.1所示。

[1]关于Flex和Bison的更多说明，请参阅《Flex & Bison:Text Processing Tools》

其中，词法分析模块和语法分析模块负责预处理，把输入转化成我们设计的数据结构，正依赖图模块则是利用预处理生成的数据建立程序的正依赖图，最后，主任务模块负责执行实验的任务，对于实验一，该模块负责统计程序的所有基本环和特征环的数目及其识别时间；对于实验二，该模块负责统计程序的各种环的数目；对于实验三，该模块负责判断程序是否属于HWEF*程序。

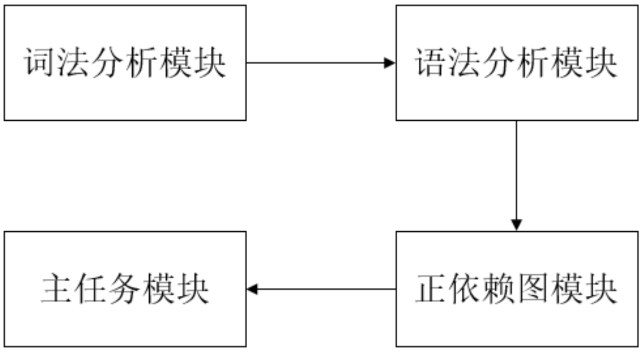

图 6.1: 程序框架

6.3 实验环境

本文中所使用的实验环境如表6.2所示。

表 6.2: 实验环境信息

环境	描述
操作系统	Linux Ubuntu 14.04
处理器	AMD A10-5800K 3.8 GHZ
内存	3.3 GB 1600 MHZ DDR3

6.4 正规逻辑程序的基本环和特征环的对比实验

Lifschitz等(2006)证明了回答集逻辑程序的求解所需要用到的环的数目是很有可能随着程序规模的增大而指数爆炸的[34]，因此，在最坏的情况下，特征环的数目同样也会随着程序规模的增大而指数爆炸。然而，对于某些回答集逻辑程序，特征环的数目和基本环相比，要少很多。同时，使

用$ProperLoops(P, Atoms(P))$求解回答集逻辑程序的所有特征环也比求解所有基本环要快很多。本实验将针对特定结构的正规逻辑程序，对比基本环的特征环的数目以及识别速度。

6.4.1 实验设计

针对哈密顿回路[51]这一问题，本实验考虑这样一种网络图，它由几个部分组成，每个部分之间只有很少的边，而每个部分内部则稠密地连接在一起。这样的网络图实际上是普遍存在的，比如在我们的生活中，大城市群之间往往只由一些主干线相连接，而大城市群内部的城市之间则有很多高速公路和铁路；又比如各种电器所使用的电路，通常都是由多个组件组成，每个组件内部紧密地关联在一起，而组件间则只存在很少的关联。

为了简化问题，本实验使用由多个完全子图组成的图[38]来表示上述的网络图结构。我们把这种图记为M-N-K，其中，M表示该图有M个完全子图，记为$C_1, ..., C_M$，N表示每个完全子图有N个节点，K表示从C_i到C_{i+1}有K条边，$C_{M+1} = C_1$。比如2-3-1表示图6.2所示的结构。

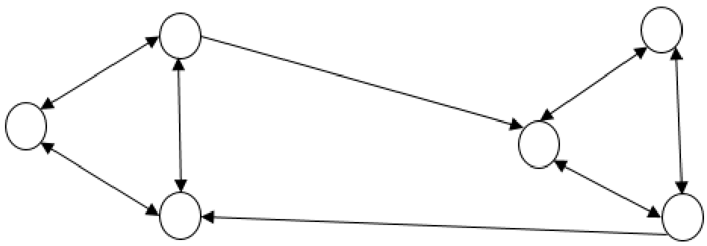

图 6.2: 2-3-1的结构图

我们使用ASP把这种图编码成哈密顿回路问题，然后使用算法1以及算法4分别求解该程序的正依赖图中的所有基本环和特征环。

6.4.2 实验结果与数据分析

实验结果如表6.3所示。表格中的每个测例M-N-K的结果，都是随机生成20个不同的图的实验结果的平均值。

表 6.3: 计算所有基本环和特征环(时间单位：秒)

(a) 2个完全子图

	基本环		特征环	
问题	数量	时间	数量	时间
2-5-1	69	0.13	23	0.04
2-5-2	135	0.12	64	0.06
2-6-1	211	1.95	53	0.18
2-6-2	473	5.22	192	0.75
2-6-3	598	4.45	346	1.29
2-7-1	685	24.88	115	0.91
2-7-2	1734	74.83	616	4.77
2-7-3	2883	46.56	1519	5.92
2-8-1	2399	274.69	241	4.55
2-8-2	6537	162.34	2124	15.74
2-8-3	—	>10min	5628	37.68

(b) 3个完全子图

	基本环		特征环	
问题	数量	时间	数量	时间
3-5-1	95	0.03	35	0.03
3-5-2	161	0.15	76	0.08
3-6-1	268	0.29	80	0.16
3-6-2	532	1.70	219	0.54
3-7-1	—	>10min	173	20.70
3-7-2	—	>10min	7555	162.00
3-7-3	—	>10min	44815	593.16
3-8-1	—	>10min	362	224.26
3-8-2	—	>10min	—	>10min

(c) 4个完全子图

	基本环		特征环	
问题	数量	时间	数量	时间
4-5-1	93	0.08	50	0.05
4-5-2	135	0.25	106	0.13
4-6-1	—	>10min	106	43.30
4-6-2	—	>10min	8364	412.12
4-7-1	—	>10min	—	>10min

实验结果显示，对于本实验的任意测例，基本环的数目都是特征环的2倍多，同时，识别基本环所需要的时间也是特征环的好几倍，对于稍大些的数据量，识别基本环所需要的时间甚至是特征环的10倍以上。由此可见，对于特定的正规逻辑程序，比起基本环，特征环具有很大的优越性。

6.5 析取逻辑程序的各种环的对比实验

本实验对比实际应用中的析取逻辑程序的各种环的数目。

6.5.1 实验设计

为了更好地反映各种环在实际应用中的数量关系，以便于我们能正确并有效地去利用它们，本实验将采用ASP竞赛的测例[2]。这些测例都是针对实际问题

[2]测例的更多详细信息，请访问http://www.cs.uni-potsdam.de/claspD

编码而成的，经常被用于ASP求解器的效率的对比。我们选用其中的一部分测例，共有7种类型，合计63个析取逻辑程序，其中，测例Sokoban和测例SCore-disjunctiveloop是HCF程序，而其他的都不是HCF程序。对于某个测例，如果1小时内不能求解出所有的环，那么我们将停止求解，并只计算当前已经找到的各种环的数目。

6.5.2 实验结果与数据分析

实验结果如表6.4所示。其中，number表示该测例类别的测例数目，timeout表示该测例类别超时的测例数目，Loops表示所有环，EL表示基本环，EL*表示算法2识别出来的环，WEL表示弱基本环，PL表示特征环，PL*表示算法5识别出来的环，WPL表示弱特征环。

表 6.4: 计算所有基本环和特征环

Benchmark	number	timeout	Loops	EL	EL*	WEL	PL	PL*	WPL
Sokoban(HCF)	11	0	2546	2525	2525	2525	642	642	642
SCore-disjunctiveloop(HCF)	2	1	5251	5251	5251	5251	1751	1751	1751
qbf.cgs	23	3	15095	1407	1407	1441	1407	1407	1441
qbf.gw	18	7	10751	4153	4162	5302	4150	4153	5302
SCore-Mutex	2	2	9250	2445	2562	3371	2445	2448	3371
SCore-RandomQBF	6	6	6417	6388	6391	6392	6388	6391	6392
SCore-Strategic	1	1	8000	6954	6983	6959	6983	6985	6992
Average Number	63	20	8188	4165	4180	4463	3396	3398	3698

实验结果显示，对于这些析取逻辑程序P，$|PL(P)|$小于$|EL(P)|$，$|EL^{*}(P)|$和$WEL(P)$接近$EL(P)$，$PL^{*}(P)$和$WPL(P)$接近$PL(P)$。尽管查找所有的环是一项庞大的工程，但是从实验过程中，我们发现，随着数据量的增大，基本环类别的环(EL、EL*和WEL)的增长速率相似，类似地，特征环类别的环(PL、PL*和WPL)的增长速率也相似。注意到，WEL和WPL是可以在多项式时间内被识别的，这意味着，在实际应用中，我们可以使用弱基本环和弱特征环来替换基本环和特征环。

6.6 HWEF*程序的检测实验

在上一章，我们总结了HCF、HWEF*、HWEF、HEF、HWPF和HPF程序之间的关系。在这些程序类别中，只有HWEF*程序可以在多项式时间内被识别。本实验将统计HWEF*程序在实际应用中所占的比例。

6.6.1 实验设计

为了更好地反映HWEF*在实际应用中所占比例，本实验同样采用ASP竞赛的测例。上一章提到，HCF程序肯定HWEF*程序，因此，本实验选择了这些测例中的一部分非HCF程序的测例，共有5个类型，合计237个析取逻辑程序。

6.6.2 实验结果与数据分析

表 6.5: HWEF*程序的比例

Benchmark	number	HWEF*	not HWEF*
qbf.cgs	100	36	64
qbf.gw	100	0	100
SCore-Mutex	7	0	7
SCore-RandomQBF	15	0	15
SCore-StrategicCompanies	15	0	15
Average Number	237	36	201

实验结果如表6.5所示，其中，number表示该类型的测例的数量，HWEF*表示属于HWEF*程序的数量，not HWEF*表示不是HWEF*程序的数量。

实验结果显示，对于测例qbf.cgs，有36%的测例是属于HWEF*。然而，对于所有测例来说，只有15.19%的测例属于HWEF*，由此可见，HWEF*程序所占比例并不高。不过，这在意料之中。在本文上一章所总结的程序类别的关系中，HWEF*程序是HCF程序的直接超集，而符合HCF程序的程序本来就不会很多。另一方面，HWEF*的识别算法可以加速HCF程序的识别。由于HWEF*程序包含所有的HCF程序，所以，在识别HCF程序的过程中，可以先判断其是否为HWEF*程序，如果不是，那么可以直接断定其不会是HCF程序，从而提

升HCF程序的识别效率。

6.7 本章小结

本章进行了三个实验。对于特定的正规逻辑程序，本章通过实验一，说明了和基本环相比，特征环无论在数目上还是识别效率上，都有很大的优势。对于析取逻辑程序，查找所有的基本环或特征环都很难，但是，通过实验二，我们可以看出，弱基本环与弱特征环在数量上和基本环与特征环的差别不大，同时他们的识别效率是比基本环和特征环高很多的，这意味着，我们完全可以在实际应用中，直接使用弱基本环和弱特征环。最后，本章通过实验三，验证了HWEF*程序在实际应用中所占的比例并不高，从而引导出可以使用HWEF*程序的识别算法提升HCF程序的识别效率的想法。

第 7 章　总结与展望

本章是全文的总结，不仅归纳概述本文的主要研究成果与主要工作，还对后续的研究工作进行了展望。

7.1　本文的研究总结

本文致力于提高基于SAT求解器的ASP求解器的效率。这种求解器的技术核心是回答集逻辑程序中的环与环公式。总的来说，本文的主要研究成果有：

本文深入研究了Gebser等提出的基本环的性质及其基于自底向上策略的识别算法，并从外部支持的角度，重新定义了基本环的概念。同时，针对正规逻辑程序，本文给出了一种基于自顶向下策略的识别算法，该算法的时间复杂度和Gebser等给出的一样。针对析取逻辑程序，本文给出了多项式时间复杂度的近似算法，用于识别基本环的一个超集。

对于正规逻辑程序，本文提出了特征环的概念，并证明了特征环已经足以完成ASP程序的回答集的求解，同时还给出一个多项式时间复杂度的算法，用于识别特征环。特征环和基本环的对比实验显示，对于正依赖图符合特定结构的正规逻辑程序，特征环的数量比基本环少很多，同时，特征环的识别效率也比基本环高很多。这一结论很好地解释了Ji等人的观察结果，即如果ASP程序的所有环的外部支持都不超过一个，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。

对于析取逻辑程序，本文同样提出了其特征环的概念。和正规逻辑程序不一样的是，识别析取逻辑程序的特征环是coNP-complete的。针对这一问题，本文提出了弱基本环和弱特征环的概念，并且给出了一个多项式时间复杂度的识别算法，同时，通过实验，本文发现弱基本环与弱特征环和基本环与特征环的数目非常接近。因此，在实际问题中，我们可以使用弱基本环和弱特征环来替换基本环和特征环，以提高求解的效率。

7.2 后续研究工作

环与环公式理论是基于SAT求解器的ASP求解器的基础。本文在理论上证明了特征环、弱基本环和弱特征环的性质，并通过实验，进一步验证了它们的优越性。这些理论基础和实验结果是提高基于SAT求解器的ASP求解器的有效依据。本文的后续研究工作主要有两方面：

第一， 进一步优化各种环的识别算法，提高他们的识别效率；

第二，实现基于特征环、弱特征环和弱基本环，并且能用于工业应用的ASP求解器。

参考文献

[1] Randall Davis, Howard E. Shrobe, and Peter Szolovits. What is a knowledge representation [J]. *AI Magazine*, 14(1):17–33, 1993.

[2] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets [J]. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.

[3] Mike Uschold and Michael Gruninger. Ontologies: principles, methods and applications [J]. *Knowledge Engineering Review*, 11(2):93–136, 1996.

[4] David B. Fogel, Kumar Chellapilla, and Peter J. Angeline. Inductive reasoning and bounded rationality reconsidered [J]. *IEEE Transactions on Evolutionary Computation*, 3(2):142–146, 1999.

[5] Holger Andreas. Deductive reasoning in the structuralist approach [J]. *Studia Logica*, 101(5):1093–1113, 2013.

[6] John McCarthy. Programs with common sense [M]. Defense Technical Information Center, 1963.

[7] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence [M]. Stanford University USA, 1968.

[8] Drew V. McDermott and Jon Doyle. Non-monotonic logic I [J]. *Artificial Intelligence*, 13(1-2):41–72, 1980.

[9] Areski Nait Abdallah. The logic of partial information [M]. Springer Publishing Company, Incorporated, 2012.

[10] Raymond Reiter. A logic for default reasoning [J]. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[11] John McCarthy. Circumscription—a form of nonmonotonic reasoning [M]. 1987.

[12] John Alan Robinson. A machine-oriented logic based on the resolution principle [J]. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

[13] C. Cordell Green. Application of theorem proving to problem solving [A]. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence, Washington*, pages 219–240, 1969.

[14] A Colmeraner, Henri Kanoui, Robert Pasero, and Philippe Roussel. Un systeme de communication homme-machine en francais [C]. Luminy, 1973.

[15] Robert A. Kowalski. Algorithm = logic + control [J]. *Communications of the ACM*, 22(7):424–436, 1979.

[16] Vladimir Lifschitz and Hudson Turner. Splitting a logic program [J]. In *Pro-*

ceedings of the Eleventh International Conference on Logic Programming, pages 23–37, 1994.

[17] Keith L Clark. Negation as failure [M]. Springer, 1978.

[18] Raymond Reiter. On closed world data bases [M]. Springer, 1978.

[19] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs [J]. *Journal of the ACM (JACM)*, 38(3):620–650, 1991.

[20] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming [A]. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, 1988.

[21] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases [J]. *New Generation Computing*, 9(3/4):365–386, 1991.

[22] 吉建民. 提高asp效率的若干途径及服务机器人上应用[D]. 2010.

[23] Vladimir Lifschitz. What is answer set programming [A]. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1594–1597, 2008.

[24] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving [J]. *Communications of the ACM*, 5(7):394–397, 1962.

[25] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning [J]. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.

[26] Ilkka Niemelä, Patrik Simons, and Tommi Syrjänen. Smodels: A system for answer set programming [J]. *Clinical Orthopaedics and Related Research*, cs.AI/0003033, 2000.

[27] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp : A Conflict-Driven Answer Set Solver [M]. 2007.

[28] Niklas Sorensson and Niklas Een. Minisat v1.13-a sat solver with conflict-clause minimization [J]. *SAT Agricultural Research*, 2005:53, 2005.

[29] Fangzhen Lin and Yuting Zhao. ASSAT: computing answer sets of a logic program by SAT solvers [J]. *Artificial Intelligence*, 157(1-2):115–137, 2004.

[30] Yuliya Lierler. cmodels - sat-based disjunctive answer set solver [A]. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 447–451, 2005.

[31] Fangzhen Lin and Yuting Zhao. ASSAT: computing answer sets of a logic

program by SAT solvers [A]. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 112–118, 2002.

[32] Jun Gu, Paul W Purdom, John Franco, and Benjamin W Wah. Algorithms for the satisfiability (sat) problem [M]. Springer, 1999.

[33] Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs [A]. In *Proceedings of 19th International Conference on Logic Programming*, pages 451–465, 2003.

[34] Vladimir Lifschitz and Alexander A. Razborov. Why are there so many loop formulas [J]. *ACM Transactions on Computational Logic*, 7(2):261–268, 2006.

[35] Martin Gebser and Torsten Schaub. Loops: Relevant or redundant [A]. In *Proceedings of 8th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 53–65, 2005.

[36] Martin Gebser, Joohyung Lee, and Yuliya Lierler. On elementary loops of logic programs [J]. *Theory and Practice of Logic Programming*, 11(6):953–988, 2011.

[37] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for disjunctive logic programs [J]. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):53–87, 1994.

[38] Xiaoping Chen, Jianmin Ji, and Fangzhen Lin. Computing loops with at most one external support rule [J]. *ACM Transactions on Computational Logic*, 14(1):3, 2013.

[39] 陆钟万. 面向计算机科学的数理逻辑[M]. 北京大学出版社, 1989.

[40] Kenneth Rosen. Discrete Mathematics and Its Applications 7th edition [M]. McGraw-Hill, 2011.

[41] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web [J]. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.

[42] Piero Bonatti, Francesco Calimeri, Nicola Leone, and Francesco Ricca. Answer set programming [J]. *A 25-Year Perspective on Logic Programming*, pages 159–182, 2010.

[43] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer [M]. Springer, 2009.

[44] Douglas Brent West et al. Introduction to graph theory [M], volume 2. Prentice hall Upper Saddle River, 2001.

[45] Shimon Even. Graph algorithms [M]. Cambridge University Press, 2011.

[46] Robert Endre Tarjan. Depth-first search and linear graph algorithms [J]. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[47] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. Introduction to algorithms [M], volume 2. MIT press Cambridge, 2001.

[48] Martin Gebser, Joohyung Lee, and Yuliya Lierler. Elementary sets of logic programs [A]. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, pages 244–249, 2006.

[49] Martin Gebser, Joohyung Lee, and Yuliya Lierler. Head-elementary-set-free logic programs [A]. In *Proceedings of 9th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 149–161, 2007.

[50] Fabio Fassetti and Luigi Palopoli. On the complexity of identifying head-elementary-set-free programs [J]. *Theory and Practice of Logic Programming*, 10(1):113–123, 2010.

[51] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm [J]. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

在学期间论文发表情况

1. XXXXX(作者名), XXXXX(论文名), In Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI-15). 2015. July 25-31, 2015, Buenos Aires, Argentina.

此论文的内容由本文第五章的工作扩展而来。

2. XXXXX(作者名), XXXXX(论文名), In Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence (AAAI-15). 2015. January 25 – 30, 2015, Austin Texas, USA.

本文第五章中的工作内容为此论文的主要内容。

3. XXXXX(作者名), XXXXX(论文名), In Proceedings of the Twenty-eighth AAAI Conference on Artificial Intelligence (AAAI-14). 2014. July 27 -31, 2014, Québec City, Québec, Canada.

本文第三章和第四章的工作内容为此论文的主要内容。

在学期间参与项目情况

1. 2011年度博士点基金自然科学类课题（新教师类课题），XXXXX(项目编号)，XXXXX(项目名称)，2012/01 –2014/12，4万元，已结题，参与。

2. 广东省自然科学基金（自由申请），XXXXX(项目编号)，XXXXX(项目名称)，2012/10 –2014/10，5万元，已结题，参与。

致 谢