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 Logic Program

学位申请人：袁镇锋

指 导 教 师：万海 高级讲师

专 业 名 称：软件工程

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

论文题目： 回答集逻辑程序特征环的研究
专　　业： 软件工程
硕 士 生： 袁镇锋
指导教师： 万海　高级讲师

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

Title： Research on Proper Loops in Answer Set Logic Program

Major： Software Engineering

Name： Zhenfeng Yuan

Supervisor： Hai Wan

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 III

目　录 V

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 3

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　一阶限定理论 10

2.4　一阶稳定理论 15

2.5　回答集程序 19

参考文献 24

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，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)、规则(rules)和本体(ontologies)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[2]。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[3]。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)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)[4],Reiter的缺省逻辑(default logic)[5]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)[7]。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[8]。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[9]。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[10]。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)[11]概念、Reiter的闭世界假设(Closed World Assumption, CMA)[12]和Van Gelder的良序(well-founded)语义[13]。1988年，Lifschitz等人提出了稳定模型语义(stable models semantics)[14]，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[15]。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)[16]。

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)的，包括DLV，smodels和clasp等。另一类则是基于SAT求解器的，包括ASSAT和cmodels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[17]，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)[11]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[18]。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[19]。2005年，Gebser与Schaub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)[20]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[21]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[22]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[23]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计算算法[20]，本文提出了基本环的另一种定义，并给出一种自顶向下的计算算法，该算法的计算复杂性和Gebser-Schaub的算法一样。

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[23]，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、现阶段国内为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。

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，然后把特征环拓展到析取逻辑程序，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化版本特征环。

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；第二个实验室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[24]。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：命题符号，通常使用小写英文字母表示；联结符号，包括?(否定)、∧(合取)、∨(析取)、→(蕴含)和?(等价于)；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[24, 25]。

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)： 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：

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

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：

$$head(r) \leftarrow body(r).$$

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，$body^+(r) = a_1, ..., a_m$，$body^-(r) = not\ a_{m+1} \wedge not\ a_n$，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$R$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,$Atoms(P)$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：

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

给定文字l，它的补(complement)记为$\overline{l}$。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，$\overline{L} = \{\overline{l} | l \in L\}$。

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：

1. 如果X是一个n元谓词变元，$t_0, ..., t_{n-1}$是v-二阶项，则$X(t_0, ..., t_{n-1})$ 是L^v_{II}-公式；
2. 如果φ是一个L^v_{II}-公式且X是一个谓词变元，则$\exists X\varphi$和$\forall X\varphi$均是L^v_{II}-公式；
3. 如果φ是一个L^v_{II}-公式且h是一个谓词变元，则$\exists h\varphi$和$\forall h\varphi$均是L^v_{II}-公式。

其中，所有作用于谓词变元和函词变元的量词均被称为二阶量词，而作用于个体变元的量词将被称为一阶量词。与一阶语言类似，我们定义二阶量词$Q\xi$(其中$Q \in \{\forall, \exists\}$，$\xi$为谓词变元或函词变元)在公式$Q\xi\varphi$中的辖域为$\varphi$。若某一谓词变元(或函词变元)在某一公式中存在不在任何二阶量词辖域中的出现，则称该变元为自由谓词变元(或自由函词变元)。所有自由谓词变元、函词变元及个体变元均统称为自由变元。称无自由变元的L^v_{II}-公式为L^v_{II}-语句。

有时为了叙述简单，我们不严格区分常元与变元，统称谓词常元与谓词变元为谓词，统称函词常元与函词变元为函词。设$\sigma = \{\xi_1, \xi_2, ..., \xi_n\}$是由谓词变元、函词变元和个体变元组成的集合，为方便起见，我们通常将$Q\xi_1Q\xi_2...Q\xi_n\varphi$简写为$Q\sigma\varphi$。由后面即将给出的语义我们不难看出$\sigma$中变元的顺序完全不影响公式的语义，因此这样的简写不会产生任何本质上的歧义。

类似于一阶逻辑，下面定义二阶逻辑的语义。结构μ上的一个二阶赋值α是从所有变元(谓词、函词、个体)到其论域A上相应对象(关系、函数、元素)的一个映

射，其中被映射的变元元数与映射到的对象元素应当匹配。在无歧义的情况下，我们不区分二阶赋值和一阶赋值，统称为赋值(Assignment)。在一阶逻辑的基础上，增加下述规则来定义二阶逻辑$L-II$的满足关系：

- $(\mu,\alpha)\models X(t_1,t_2,...,t_n)$ iff $(\alpha(t_1),\alpha(t_2),...,\alpha(t_n))\in\alpha(X)$
- $(\mu,\alpha)\models\forall X\varphi$ iff 对A上所有n元关系R均有$(\mu,\alpha^X_R)\models\varphi$
- $(\mu,\alpha)\models\exists X\varphi$ 存在A上某个n元关系R满足$(\mu,\alpha^X_R)\models\varphi$
- $(\mu,\alpha)\models\forall h\varphi$ iff 对A上所有m元全函数f均有$(\mu,\alpha^h_f)\models\varphi$
- $(\mu,\alpha)\models\exists h\varphi$ 存在A上某个m元全函数f满足$(\mu,\alpha^h_f)\models\varphi$

其中，n为任意正整数，X为任意n元谓词变元，$t_1,t_2,...,t_n$为任意v-二阶项，$\alpha(t_i)$ $(i=1,2,...,n)$表示v-二阶项t_i在赋值α下的新v-二阶项，$\alpha(X)$表示谓词变元X在赋值α下的n元关系；m为任意非负整数，h为任意n元谓词变元；α^X_R(或α^h_f)表示将赋值α中谓词变元X(或函词变元h)重新赋值为关系R(或全函数f)后得到的新赋值。

二阶逻辑的语义定义建立在一阶逻辑语义的基础上。此处不作详细介绍，具体参阅文献[?]。

2.3 一阶限定理论

限定理论[? ?](Circumscription)最早由John McCarthy在上世纪八十年代提出，用于形式化常识推理。在过去几十年中，限定理论一直是非单调逻辑研究的一个重要分支。

限定理论实际上是一种二阶逻辑，同时也可以看作是一种语义。本节将介绍两种限定理论：并行限定理论以及其扩充带优先级的限定理论。当并行限定理论或带优先级的限定理论作为一种语义应用在一个一阶理论中，我们统称为一阶限定理论语义下的理论，在无歧义的情况下，简称为一阶限定理论。

2.3.1 一阶并行限定理论

Vladimir Lifschitz在1994年对John McCarthy提出的限定理论提供了一个精确的数学定义[?]，称之为并行限定理论，本文就沿用他的定义。设φ是任意

一个一阶语句，我们将它的符号集划分为三个互不相交的集合：极小化谓词常元σ_m(Minimized predicates)，可变常元σ_v(Varied constants)，不变常元(Fixed constants)。其中，φ称为被限定的语句，符号集的划分称为限定策略(Policy)。一阶限定理论是二阶逻辑的一个片段，需要引入谓词变元和函词变元。对每一个极小化谓词常元$P \in \sigma_m$，我们都相应的引入一个与其具有相同元数的谓词变元P^*，用σ_m^*表示这些谓词变元组成的集合；对每一个在σ_v中的可变谓词、函词、个体常元，我们都相应的引入一个与其具有相同元数的谓词、函词、个体变元，用σ_v^*表示这些变元组成的集合。在给出一阶限定理论的定义前，我们需要用到某些简写：

- $P^* = P$表示$\forall \bar{x}(P(\bar{x}) \leftrightarrow P^*(\bar{x}))$
- $P^* \leq P$表示$\forall \bar{x}(P^*(\bar{x}) \rightarrow P(\bar{x}))$
- $P^* < P$表示$(P^* \leq P) \wedge \neg(P^* = P)$

其中，$\bar{x}$是个体变元元组，P是任意一个谓词常元，P^*是与P具有相同元数的谓词变元。

这种简写实际上说明了谓词间的比较关系，比较的是谓词外延(Extension)。谓词外延，指谓词被解释或赋值为论域上的关系时所包含的元素。具体地说，n元谓词P的外延就是那些使$P(\bar{x})$为真的n元个体变元元组$\bar{x}$组成的集合。$=$表示两谓词外延相等，$P^* \leq P$表示P^*的外延是P的子集，$P^* < P$表示P^*的外延是P的真子集。限定理论的直观含义是，在保持原一阶语句成立的情况下，使极小化谓词的外延在集合包含关系下最小。

进一步地，我们可以将这种简写扩展到谓词集合间的比较：

- $\sigma^* = \sigma$表示对于任意的$P \in \sigma$及其对应的$P^* \in \sigma^*$，满足$P^* = P$
- $\sigma^* \leq \sigma$表示对于任意的$P \in \sigma$及其对应的$P^* \in \sigma^*$，满足$P^* \leq P$
- $\sigma^* < \sigma$表示$(\sigma^* \leq \sigma) \wedge \neg(\sigma^* = \sigma)$

其中，σ和σ^*是具有相同谓词个数的两个谓词常元集，对σ中每个谓词P，在σ^*中都存在与之对应的具有相同元数的谓词P^*。

接下来，我们用一个二阶公式来定义并行限定理论：

定义 2.9： 设φ是任意一阶语句，σ_m是φ的符号集$v(\varphi)$中的一集谓词常元，σ_v是$v(\varphi)$中的一集常元，并且σ_m与σ_v互不相交，我们定义在并行限定理论语义下的一阶语句为：

$$\text{CIRC}[\varphi;\sigma_m;\sigma_v] = \varphi \wedge \forall\sigma_m^*\sigma_v^*(\sigma_m^* < \sigma_m \rightarrow \neg\varphi(\sigma_m^*,\sigma_v^*)) \tag{2.1}$$

其中，σ_m^*和σ_v^*分别是与σ_m和σ_v中每一常元相对应的具有相同元数的变元组成的集合；$\varphi(\sigma_m^*,\sigma_v^*)$表示在一阶语句$\varphi$中将所有$\sigma_m$(或$\sigma_v$)中的常元的出现替换为相应的$\sigma_m^*$(或$\sigma_v^*$)中的变元后得到的公式。

直观地，并行限定理论对多个谓词整体极小化，在保持原一阶语句成立的情况下，可以随意改变可变常元的解释，使得每个极小化谓词的外延在集合包含关系下最小，换句话说，不能使至少一个极小化谓词的外延更小，除非扩展其它极小化谓词的外延。

下面我们来定义并行限定理论的语义。在公式(**??**)中，令σ表示φ的符号集。设μ和ν为任意的两个σ-结构，定义$\mu \preceq_{\sigma_m,\sigma_v} \nu$当且仅当以下条件成立：

1. μ与ν具有相同的论域；

2. 对于σ中所有的极小谓词常元P，均有$\mu(P) \subseteq \nu(P)$；

3. 对于σ中所有不变常元C，均有$\mu(C) = \nu(C)$。

容易看出，$\preceq_{\sigma_m,\sigma_v}$是所有$\sigma$-结构上的一个偏序关系，具有自反性和传递性。特别地，如果$\mu \preceq_{\sigma_m,\sigma_v} \nu$成立但是$\nu \preceq_{\sigma_m,\sigma_v} \mu$不成立，我们用$\mu \prec_{\sigma_m,\sigma_v} \nu$表示。

定义 2.10： 设φ是任意一阶语句，若μ是φ的一个模型，并且不存在φ的任何模型满足$\nu \prec_{\sigma_m,\sigma_v} \mu$，则称$\mu$为$\varphi$在关系$\prec_{\sigma_m,\sigma_v}$下的极小模型(Minimal model)。

并行限定理论实际上就是描述了极小模型，用下面的命题来说明。

命题 2.11： **(文献[?])** 设φ是任意一阶语句，μ是一个$v(\varphi)$-结构，则$\mu \models \text{CIRC}[\varphi;\sigma_m;\sigma_v]$当且仅当$\mu$是语句$\varphi$在关系$\prec_{\sigma_m,\sigma_v}$下的极小模型。

为简单起见，我们将在并行限定理论语义下的一阶理论称为一阶并行限定理论。此外，我们通常将CIRC$[\varphi;\sigma_m\cup\tau_m;\sigma_v\cup\tau_v]$简写成CIRC$[\varphi;\sigma_m,\tau_m;\sigma_v,\tau_v]$。特别地，当$\tau_m=\{P\}$和$\tau_v=\{Q\}$时，我们简写成CIRC$[\varphi;\sigma_m,P;\sigma_v,Q]$。

在实际生活中，我们对事物的理解都是带着常识知识进行推理的，实际上是用了将“异常”的事物“极小化”的思想。用上一章的例子：在正常情况下鸟会飞。我们可以用一个一阶语句φ简单对其形式化，$\forall xBird(x)\wedge\neg Ab(x)\rightarrow fly(x)$。如果我们只知道Tweety是一只鸟，即$Bird(Tweety)$，我们一般会尽可能的认为鸟都是正常的，也就是极小化“异常”的因素，通过计算CIRC$[\varphi\wedge Bird(Tweety);\{Ab,fly\};\emptyset]$得到结论$fly(Tweety)$，即Tweety会飞。但当我们还知道Tweety的翅膀断了，它不是一只正常的鸟，即$Ab(Tweety)$，用同样的限定策略对$\varphi\wedge Bird(Tweety)\wedge Ab(Tweety)$进行限定，我们会推理到Tweety不会飞，而这结论与我们日常的推理是一致的。通过对“异常”因素的极小化，我们用限定理论形式化了常识推理。

我们将$\forall\bar{x}(P(\bar{x})\vee\neg P(\bar{x}))$称为谓词$P$的选择规则(Choice Rule)，记为$choice(P)$。

定理 2.4： **(文献[?])** 设φ是任意一阶语句，令σ_c表示$v(\varphi)$中的不变谓词常元之集，则有CIRC$[\varphi;\sigma_m;\sigma_v]$逻辑等价于CIRC$[\varphi\wedge\bigwedge_{P\in\sigma_c}choice(P);\sigma_m,\sigma_c;\sigma_v]$。

该定理说明了一阶并行限定理论中的不变谓词常元可转换为极小化谓词，因此我们在计算一阶并行理论时，一般不讨论不变谓词常元，更关注的是极小化谓词常元和可变常元。

2.3.2 一阶带优先级的限定理论

Vladimir Lifschitz在[?]中除了为并行限定理论提供了一个数学定义，还引入“优先级”的思想，提出了带优先级的限定理论(Prioritized circumscription)。

我们将极小化谓词集合σ_m划分为k个互不相交的子集合$\sigma_1,\sigma_2,...,\sigma_k$，将这样的划分称为优先级限定策略。我们认为$\sigma_i$中的谓词比$\sigma_j(i\leq j)$中的谓词具有更高的优先级来极小化。类似地，在定义带优先级的限定理论前，我们先定义一个简

写：设$\sigma_1, \sigma_2, ..., \sigma_k$是$k$个互不相交的谓词常元集合，$\sigma_1^*, \sigma_2^*, ..., \sigma_k^*$是按照上一子节的方法引入谓词变元组成的并且互不相交的集合。令σ_m表示$\sigma_1 \cup \sigma_2 \cup ... \cup \sigma_k$，$\sigma_m^*$表示$\sigma_1^* \cup \sigma_2^* \cup ... \cup \sigma_k^*$，则我们用$\sigma_m^* \preceq \sigma_m$表示

$$\bigwedge_{i=1}^{k} \Big[(\bigwedge_{j=1}^{i-1} (\sigma_j^* = \sigma_j)) \rightarrow (\sigma_i^* \leq \sigma_i) \Big]$$

其中，$=$、$\leq$沿用上一字节的简写。

实际上，$\preceq$描述了两组谓词集合在一定优先级序列下的比较关系，在高优先级的谓词外延相同的情况下，再比较那些具有相同优先级的谓词。例如，当$k = 1$时，$\sigma_m^* \preceq \sigma_m$就等价于$\sigma_m^* \leq \sigma_m$，或者$\sigma_1^* \leq \sigma_1$；当$k = 2$时，$\sigma_m^* \preceq \sigma_m$就等价于$(\sigma_1^* \leq \sigma_1) \wedge ((\sigma_1^* = \sigma_1) \rightarrow (\sigma_2^* \leq \sigma_2))$。

特别地，当$\sigma_m^* \preceq \sigma_m$成立但是$\sigma_m \preceq \sigma_m^*$不成立，我们简写成$\sigma_m^* \prec \sigma_m$。我们同样可以用一个二阶公式来定义带优先级的限定理论。

定义 2.12： 设φ是任意一阶语句，σ_m是φ的符号集$v(\varphi)$中的一集谓词常元，σ_v是 $v(\varphi)$中的一集常元，令σ_m划分为k个互不相交的谓词集合$\sigma_1, \sigma_2, ..., \sigma_k$，并且$\sigma_i (1 \leq i \leq k)$与$\sigma_v$互不相交，我们定义在带优先级的限定理论语义下的一阶语句为：

$$\text{CIRC}[\varphi; \sigma_1 > \sigma_2 > \cdots > \sigma_k; \sigma_v] = \varphi \wedge \forall \sigma_m^* \sigma_v^* (\sigma_m^* \prec \sigma_m \rightarrow \neg \varphi(\sigma_m^*, \sigma_v^*)) \quad (2.2)$$

其中，$\sigma_i^* (1 \leq i \leq k)$和$\sigma_v^*$分别是与$\sigma_i (1 \leq i \leq k)$和$\sigma_v$中每一常元相对应的具有相同元数的变元组成的集合，$\sigma_m^*$表示这些$\sigma_i (1 \leq i \leq k)$的并集，$\varphi(\sigma_m^*, \sigma_v^*)$表示在一阶语句$\varphi$中将所有$\sigma_m$(或$\sigma_v$)中的常元的出现替换为相应的$\sigma_m^*$(或$\sigma_v^*$)中的变元后得到的公式。

为简单起见，我们将在带优先级的限定理论语义下的一阶理论简称为一阶带优先级的限定理论。实际上，当$k = 1$时，一阶带优先级的限定理论就变成了一阶并行理论，因此后者是前者的一个特例，前者是后者的扩充或者泛化。

直观地，带优先级的限定理论优先让那些具有高优先级的极小化谓词的外延最小，再考虑极小化那些低优先级的极小化谓词的外延。根据优先级的直观思

想，我们可以逐级逐级地讨论极小化谓词，在讨论当前优先级的时候，在保证原一阶语句成立和不改变较高优先级的极小化谓词的外延的情况下，可以随意改变较低优先级的极小化谓词的外延和可变常元的解释，使得当前优先级的极小化谓词的外延最小。下面的命题就叙述了这一思路：

命题 2.13： **(文献[?] 命题15)**带优先级的限定理论$\mathbf{CIRC}[\varphi;\sigma_1 > \sigma_2 > \cdots > \sigma_k;\sigma_v]$ 等价于k个并行限定的合取：

$$\bigwedge_{1\leq j\leq k} \mathrm{CIRC}[\varphi;\sigma_j;\sigma_{j+1},\sigma_{j+2},...,\sigma_k,\sigma_v]. \tag{2.3}$$

下面给出一个具体的例子介绍一阶带优先级的限定理论与一阶并行限定理论：

例 2.8： **(文献[?] 第7节)**设φ表示如下一阶公式的合取的全称闭包：

$$Quaker(x) \wedge \neg Ab_1(x) \rightarrow Pacifist(x) \tag{2.4}$$

$$Republican(x) \wedge \neg Ab_2(x) \rightarrow \neg Pacifist(x) \tag{2.5}$$

其中$Quaker(x)$表示x是教友派信徒，$Republican(x)$表示x是共和党人，$Pacifist(x)$表示x是和平主义者，$Ab_1(x)$表示x是不普通的教友派信徒，$Ab_2(x)$表示x是不普通的共和党人。整个理论φ表示每个普通的教友派信徒都是和平主义者，每个普通的共和党人都不是和平主义者。如果一个人$Nixon$既是教友派信徒又是共和党人，当我们尽可能排除那些“异常”的因素，即同时极小化Ab_1和Ab_2两个谓词，那么我们会得到两个解释：一个认为$Nixon$是和平主义者，另一个认为他不是和平主义者。但是，当我们认为共和党人的异常几率会高于教友派信徒的异常几率时，我们通过计算$\mathrm{CIRC}[\varphi;Ab_1 > Ab_2;Pacifist]$得到结论：$Nixon$是一个和平主义者。

2.4 一阶稳定理论

本节我们将介绍另一种二阶逻辑－－－一阶稳定模型语义。Paolo Ferraris等人在2005年在[?]提出了一阶稳定模型语义的思想，随后在[?]以二阶公式的形式给出它的精确的数学定义。

设ψ是任意一个一阶语句，我们将它的符号集划分为两部分：内涵谓词常元σ_i(Intensional predicates)和外延常元(Extensional constants)。在不引起歧义的情况下，我们将一阶稳定模型语义下的一阶理论，简称为一阶稳定理论。需要注意的是，这里的稳定理论与经典模型论中的稳定理论是两个不同的概念，不能混淆。一阶稳定理论实际上是二阶逻辑的一个片段，也需要引入谓词变元和函词变元。类似于一阶限定理论，对每一个内涵谓词常元$P \in \sigma_i$，我们都相应的引入一个与其具有相同元数的谓词变元P^*，用σ_i^*表示这些谓词变元组成的集合。下面我们给出一阶稳定理论的定义：

定义 2.14： 设ψ是任意一阶语句，σ_i是φ的符号集$v(\varphi)$中的一集谓词常元，我们定义一阶稳定理论为：

$$\mathrm{SM}[\psi;\sigma_i] = \psi \wedge \forall\sigma_i^*(\sigma_i^* < \sigma_i \rightarrow \neg\mathrm{St}(\psi;\sigma_i)) \tag{2.6}$$

其中σ_i^*是与σ_i中每一常元相对应的具有相同元数的变元组成的集合，$St(\psi;\sigma_i)$递归地定义如下：

- 若$P \in \sigma_i$，则$\mathrm{St}(P(\bar{x});\sigma_i) = P^*(\bar{x})$；
- 若谓词常元Q不在σ_i中，则$\mathrm{St}(Q(\bar{x});\sigma_i) = Q(\bar{x})$；
- $\mathrm{St}(\psi_1 \circ \psi_2;\sigma_i) = \mathrm{St}(\psi_1;\sigma_i) \circ \mathrm{St}(\psi_2;\sigma_i)$，其中$\circ \in \{\wedge,\vee\}$；
- $\mathrm{St}(\psi_1 \rightarrow \psi_2;\sigma_i) = (\mathrm{St}(\psi_1;\sigma_i) \rightarrow \mathrm{St}(\psi_2;\sigma_i)) \wedge (\psi_1 \rightarrow \psi_2)$；
- $\mathrm{St}(Qx\psi;\sigma_i) = Qx\mathrm{St}(\psi;\sigma_i)$，其中$Q \in \{\forall,\exists\}$。

在St的定义中，我们没有对¬进行讨论，因为我们将$\neg\psi$看作$\psi \rightarrow \bot$，而$\bot$可看作是一个特别的谓词常元符号，也就是说，$\mathrm{St}(\neg\psi;\sigma_i) = \neg(\mathrm{St}(\psi;\sigma_i) \wedge \neg\psi$。此外，关于否定词¬，操作St还具有一下性质：

命题 2.15： **(文献[?] 命题2)**设ψ为任意一个一阶公式，设σ_i表示内涵谓词集。若$\sigma_i^* < \sigma_i$，则有$\mathrm{St}(\neg\psi;\sigma_i) \equiv \neg\psi$。

另外，在一阶稳定理论的定义中，我们没有对个体变元和函词变元作任何的变换。实际上，国外已有不少工作关注于内涵函词，但这不在本文的讨论范围中，详细的可参考**[? ?]**。

为简便起见，若σ_i包含ψ中出现的所有谓词常元，我们将SM$[\psi;\sigma_i]$简写为SM$[\psi]$，将St$(\psi;\sigma_i)$简写为St(ψ)。类似于一阶限定理论，我们通常将SM$[\psi;\sigma_i \cup \tau_i]$简写成SM$[\psi;\sigma_i,\tau_i]$；若$\tau_i = P$，则简写成SM$[\psi;\sigma_i,P]$。

设ψ是任意一个一阶语句，如果$v(\psi)$-结构μ是SM$[\psi;\sigma_i]$的一个模型，则μ被称为ψ的一个σ_i-稳定模型(Stable model)。特别地，μ是ψ的稳定模型当且仅当μ是SM$[\psi]$的模型。

设(φ,σ_i)是一个稳定模型语义下的一阶逻辑，v为φ的符号集，则称一个v- 结构μ是φ关于σ_i的稳定模型(Stable Model)当且仅当μ是公式SM$[\varphi,\sigma_i]$的模型。

接下来我们介绍稳定模型语义中的强等价关系[?]。设φ和ψ为任意两个一阶公式，σ_i是任意一集谓词常元。如果对任意的一阶公式θ，将φ在θ 中的任意一个或多个出现替换为ψ后得到的公式θ'，均有SM$[\theta;\sigma_i]$逻辑等价于SM$[\theta';\sigma_i]$，则称φ和ψ强等价，记为$\varphi \equiv^s \psi$。

命题 2.16： **(文献[?] 定理6.4)**设φ和ψ为任意的一阶公式，则下列强等价关系成立：

(1) $\forall x\varphi(x) \wedge \psi \equiv \forall x(\varphi(x) \wedge \psi)$　　(2) $\exists x\varphi(x) \wedge \psi \equiv \exists x(\varphi(x) \wedge \psi)$

(3) $\forall x\varphi(x) \vee \psi \equiv \forall x(\varphi(x) \vee \psi)$　　(4) $\exists x\varphi(x) \vee \psi \equiv \exists x(\varphi(x) \vee \psi)$

(5) $\exists x\varphi(x) \rightarrow \psi \equiv \forall x(\varphi(x) \rightarrow \psi)$　　(6) $\forall x\varphi(x) \rightarrow \psi \equiv \exists x(\varphi(x) \rightarrow \psi)$

(7) $\psi \rightarrow \forall x\varphi(x) \equiv \forall x(\psi \rightarrow \varphi(x))$　　(8) $\psi \rightarrow \exists x\varphi(x) \equiv \exists x(\psi \rightarrow \varphi(x))$

(9) $\neg\exists x\varphi(x) \equiv \forall x\neg\varphi(x)$　　(10) $\neg\forall x\varphi(x) \equiv \exists x\neg\varphi(x)$

其中个体变元x不在公式ψ中自由出现。

通过对个体变元进行适当的改名，然后应用命题2.16，我们可得到下面的定理：

定理 2.5： 对任意一个一阶理论都存在一个前缀范式与之强等价。

类似于一阶限定理论，一阶稳定理论同样可以在不引入新的谓词常元的情况下，使用上一节定义的选择规则，来消去外延谓词常元。下面我们通过一定理来说明。

定理 2.6： **(文献[?] 定理2)** 设ψ是任意一阶语句，令σ_i和σ_e分别表示$v(\psi)$中的内涵谓词常元之集合和外延谓词常元之集合，则有SM$[\psi;\sigma_i]$逻辑等价于SM$[\psi \wedge \bigwedge_{P\in\sigma_e} choice(P);\sigma_i,\sigma_e]$。

一阶稳定理论的另一重要性质是分割定理**[?]**。首先，我们需要定义若干概念。我们称表达式ϵ在一阶公式φ中的负度为n当且仅当ϵ出现在φ中的子公式ψ恰好有n个满足后述条件：子公式ψ是否定式；或者ψ是蕴含式且ϵ出现在ψ的前件中。若表达式ϵ在一阶公式φ中的负度为偶数，则称ϵ在φ中的出现是正的；进一步地，若负度为零，则称ϵ在φ中的出现是严格正的。若表达式ϵ出现在在φ中的某个子公式是否定式，则称ϵ在φ中的出现是否定的，否则是非否定的。给定一个一阶稳定理论SM$[\psi;\sigma_i]$，我们定义ψ关于谓词集σ_i的谓词依赖图DG$[\psi;\sigma_i]$为如下有向图：

- 图DG$[\psi;\sigma_i]$的顶点集是内涵谓词集σ_i；
- 对σ_i中任何两个内涵谓词P和Q，图DG$[\psi;\sigma_i]$中存在从P到Q的一条边，当且仅当存在一个ψ的子公式$\psi_1 \to \psi_2$满足下述条件：
  - 该子公式$\psi_1 \to \psi_2$在ψ中的出现是严格正的；
  - 谓词P在ψ_2中的出现是严格正的；
  - 谓词Q在ψ_1中的出现是正的并且非否定的。

定理 2.7： **(文献[?])** 给定一个一阶稳定理论SM$[\psi;\sigma_i]$，若将σ_i划分为两个子集σ_1和σ_2，并且谓词依赖图DG$[\psi;\sigma_i]$的任一强连通分量或者是σ_1的子集，或者是σ_2的子集，则有SM$[\psi;\sigma_i]$逻辑等价于SM$[\psi;\sigma_1] \wedge$ SM$[\psi;\sigma_2]$。

定理 2.8： **(文献[?])** 给定一个一阶稳定理论SM$[\psi_1 \wedge \psi_2;\sigma_i]$，若将$\sigma_i$划分为两个子集$\sigma_1$和$\sigma_2$，并且谓词依赖图DG$[\psi_1 \wedge \psi_2;\sigma_i]$的任一强连通分量或者是$\sigma_1$的子集，或者是$\sigma_2$的子集；且$\sigma_1$中任一谓词在公式$\psi_2$中没有严格正的出现；

且σ_2中任一谓词在公式ψ_1中没有严格正的出现，则有SM$[\psi_1 \wedge \psi_2; \sigma_i]$逻辑等价于SM$[\psi_1; \sigma_1] \wedge$ SM$[\psi_2; \sigma_2]$。

上述两个定理被称为分割定理，根据这两个定理，我们可以采用分而治之的思想计算稳定模型；反过来说，我们也可以通过计算一次稳定理论来得到若干个一阶稳定理论的结果。

Heng Zhang等人在[?]中将一阶限定理论的计算嵌入到一阶稳定理论的计算中，但他们的方法不允许可变谓词的存在。

定义 2.17： 设φ为任意具有否定范式形式的一阶语句，σ_m表示$v(\varphi)$中的谓词常元子集，$Tr(\varphi; \sigma_m)$为如下公式的合取：

$$\varphi^{\neg\neg} \wedge (\tilde{\varphi} \vee \bigwedge_{P \in \sigma_m} choice(P)) \tag{2.7}$$

其中，$choice(P)$表示谓词P的选择规则$\forall \bar{x}(P(\bar{x}) \vee \neg P(\bar{x}))$；$\varphi^{\neg\neg}$是将$\varphi$中所有形如$P(\bar{x})(P \in \sigma_m)$的正文字替换为$\neg\neg P(\bar{x})$后所得到的公式；$\tilde{\varphi}$是将$\varphi$中所有形如$\neg P(\bar{x})(P \in \sigma_m)$的负文字替换为后$(P(\bar{x} \rightarrow \bigwedge_{P \in \sigma_m} choice(P))$得到的公式。

定理 2.9： 设φ为任意具有否定范式形式的一阶语句，σ_m表示$v(\varphi)$中的谓词常元子集，则$\mathbf{SM}[Tr(\varphi; \sigma_m); \sigma_i]$与$\mathbf{CIRC}[\varphi; \sigma_i; \sigma_v]$逻辑等价。

Heng Zhang等人提出的将不带可变谓词常元的一阶并行限定理论转换为一阶稳定理论的翻译，为两者的计算建起了一座桥梁。如果我们能够实现一阶稳定理论的计算方法，那么我们也就实现了不带可变谓词常元的一阶并行限定理论的计算方法了。幸运的是，我们能够将一阶稳定理论转换为逻辑程序，通过求解逻辑程序的模型来求得一阶理论的稳定模型。详细的转换方法将在下一子节介绍。

2.5 回答集程序

在这一节中，我们介绍一种较为成熟的知识表示语言——逻辑程序。一个逻辑程序是由有限个如下形式的规则组成的集合：

$$A_1 \vee A_2 \vee ... \vee A_m \leftarrow B_1 \wedge B_2 \wedge ... \wedge B_n. \tag{2.8}$$

其中 $m,n \in \mathbb{N}$，每一个 $A_i, B_j (1 \leq i \leq m, 1 \leq j \leq n)$都被称为文字，它们或为原子公式，称为正文字；它们或为原子公式的否定式，称为负文字。在蕴含符“$\leftarrow$”左边部分称为规则的头部，右边部分称为规则的体部。

若限制规则的头部不允许有负文字的出现，我们称这类逻辑程序为析取逻辑程序(Disjunctive Logic Program)，简记为DLP。若对规则没有任何限制，我们称这类逻辑程序为扩充的析取逻辑程序(Extended Disjunctive Logic Program)，简记为EDLP。

接下来我们介绍逻辑程序的语义。在1988年Michael Gelfond和Vladimir Lifschitz在[?]首次提出了不允许析取的逻辑程序的稳定模型语义，后来又扩展到扩充的析取逻辑程序[?]，并称之为回答集语义(Answer Set Semantics)。对于DLP，Michael Gelfond和Vladimir Lifschitz提出了一个逻辑程序在一个原子公式集下的归约，定义如下：

定义 2.18： 令Π为一个DLP逻辑程序，L表示逻辑程序Π中出现过的原子公式的集合，X是L的一个子集。如果对于 Π 中的每一条规则 r：

1. 如果X中的原子公式出现在规则 r 中体部的负文字中，则删除规则 r；
2. 否则，删除规则r中体部的所有负文字。最终得到的规则组成的集合称为 Π 关于文字集 X 的归约程序，记为 Π^X。

若原子公式集X满足逻辑程序Π的所有规则，则称X是Π的模型。给定一个DLP逻辑程序Π，X是原子公式集合，若不存在一个Π在X下的归约程序Π^X的模型Y且$Y \subset X$，则称X是Π的回答集(Answer Set)。

接下介绍逻辑程序的稳定模型语义。对于任意形如式2.8的规则，我们称如下公式为该规则的一阶表示：

$$\tilde{\forall}(A_1 \vee A_2 \vee ... \vee A_m \leftarrow B_1 \wedge B_2 \wedge ... \wedge B_n.) \tag{2.9}$$

一个逻辑程序Π的一阶表示，记为$\hat{\Pi}$，是该程序中所有规则的一阶表示的集合。实际上，逻辑程序的一阶表示就是一个一阶理论。为了叙述方便，在无歧义的情况

下，我们往往不对逻辑程序与它的一阶表示进行严格区分。此外，在稳定模型语义下的逻辑程序中，所有在规则的体部出现过的谓词均视作内涵谓词。

实际上，对于一个DLP逻辑程序，一阶稳定模型语义与回答集语义是一致的。根据文献[?]，我们可得到：

命题 2.19： 对于任意一个DLP逻辑程序Π，一个结构μ是$\hat{\Pi}$的稳定模型当且仅当在μ的解释下为真的原子公式组成的集合X是Π的回答集。

进一步地，一阶稳定理论与逻辑程序之间的等价关系在[? ? ?]中得到阐述。Heng Zhang等人在[?]提出了一个在稳定模型语义下消除存在量词的翻译，将任意一个一阶稳定理论转换为一个稳定模型语义下的Π_0-语句。

定义 2.20： 任意给定形如$\forall\bar{x}\exists\bar{y}\vartheta(\bar{x},\bar{y})$ 的一个一阶语句φ，定义$Tr_{QE}(\varphi)$ 为下述一阶公式的合取式的全称闭包：

$$\neg\neg S(\bar{x},\overline{\min}) \tag{2.10}$$

$$(succ(\bar{y},\bar{z})\ \wedge\ S(\bar{x},\bar{z}))\ \vee\ \vartheta^{\neg\neg}(\bar{x},\bar{y})\ \rightarrow\ S(\bar{x},\bar{y}) \tag{2.11}$$

$$T(\bar{x},\overline{\min})\ \vee\ \vartheta(\bar{x},\overline{\min}) \tag{2.12}$$

$$\begin{aligned}[(\overline{\max}=\bar{y})\vee(succ(\bar{y},\bar{z})\wedge\neg S(\bar{x},\bar{z}))]\wedge S(\bar{x},\bar{y})\\ \rightarrow(T(\bar{x},\overline{\max})\leftrightarrow\vartheta(\bar{x},\bar{y}))\end{aligned} \tag{2.13}$$

$$succ(\bar{y},\bar{z})\ \rightarrow\ (T(\bar{x},\bar{y})\ \leftrightarrow\ \vartheta(\bar{x},\bar{z})\ \vee\ T(\bar{x},\bar{z})) \tag{2.14}$$

令n为序列$\bar{x}$与$\bar{y}$的长度之和，其中：

1. $\vartheta^{\neg\neg}$是将ϑ中所有形如$P(\bar{x})$的正文字替换为$\neg\neg P(\bar{x})$后所得到的公式；
2. 谓词$succ$是未在φ中出现过的谓词常元，它是建立在有穷论域A上的$2\times|\bar{y}|$元关系，它刻画了$A^{|\bar{y}|}$中元素的一个线序，其中$\overline{\min}$和$\overline{\max}$是未在φ中出现过的个体常元，分别表示在这个线序下的最小元和最大元；
3. S和T是不在φ中出现的两个$|\bar{x}|+|\bar{y}|$元新的谓词常元。

定理 2.10： 任意给定一个一阶稳定理论$\mathbf{SM}[\varphi;\sigma_i]$，若$\varphi$是形如$\forall\bar{x}\exists\bar{y}\vartheta(\bar{x},\bar{y})$的一阶语句，则在不管$S,T,succ,\overline{\min}$和$\overline{\max}$的解释的情况下，$\mathbf{SM}[Tr_{QE}(\varphi);\sigma_i,S,T]$与$\mathbf{SM}[\varphi;\sigma_i]$在有穷结构上是逻辑等价的。

观察到Tr_{QE}[1]实际上是在给定的一个有穷结构上消去了一个存在量词块，使得整个一阶稳定理论只出现全称量词。根据命题2.5，每个一阶语句都存在一个前缀范式与之强等价。因此，只要将任意一个一阶稳定理论转换为一个稳定模型语义下的一阶前缀范式，就可以使用Tr_{QE}来消去存在量词。实际上，定义2.20中的ϑ中可能存在全称量词和存在量词，只要将翻译后的一阶语句$Tr_{QE}(\varphi)$再转换成前缀范式，再多次使用Tr_{QE}，就可以将所有存在量词都消去，最终得到一个稳定模型语义下的Π_1^0-语句。

将所有的存在量词消去后，我们可以将一个稳定模型语义下的仅含全称量词的一阶公式转换为一个与之等价的逻辑程序。Pedro Cabalar等人在[?]中提出了将稳定模型语义下的命题公式转换为一个EDLP逻辑程序的归约翻译。给定一个有穷论域，我们可以轻易地将一个全称一阶语句常例化，解释成一个命题公式。

定理 2.11： **(文献[?] 定理1)** 每一个命题理论都强等价于一个EDLP逻辑程序，并且存在一个翻译算法，将任意一个命题理论在有穷结构上在有穷时间内，转换成与之强等价的EDLP逻辑程序。

根据定理2.10和2.11，我们可以将任意一个一阶稳定理论转换成一个逻辑程序，通过一个逻辑程序的求解器，来计算一阶稳定理论。

实际上，我们可以将一个EDLP转化为与之等价的DLP，具体地，我们可以将每一条形如式2.8的规则转化为如下形式：

$$A_1 \vee A_2 \vee \ldots \vee A_m \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n \wedge \neg C_1 \wedge \ldots \wedge \neg C_l \wedge \neg\neg D_1 \wedge \ldots \wedge \neg\neg D_k \quad (2.15)$$

其中，A_i, B_j, C_s, D_t均为原子公式。变换的方式是将头部中的每一负文字移至体中，并在该负文字前再添上一个否定词。根据文献文献[?]中的规则(L3)，该变换

[1]原文中的公式(2.13)是$\neg(succ(\bar{y},\bar{z}) \wedge S(\bar{x},\bar{z})) \wedge S(\bar{x},\bar{y}) \rightarrow (T(\bar{x},\overline{\max}) \leftrightarrow \vartheta(\bar{x},\bar{y}))$，但笔者发现此处存在笔误，因为原公式的定义与它要表达的意义不同，没有表达出$\bar{z}$为$\bar{y}$的后继的意思，在征得原作者的同意下，这里将之改正。

将保持强等价。为了方便起见，我们有时也将扩充逻辑程序看作上述形式规则的一个集合。

在章衡的博士论文[?]中，他提出了将一个扩充的析取逻辑程序转换为一个与之具有相同回答集的析取逻辑程序的翻译，具体定义如下：

定义 2.21： 给定一个扩充的析取逻辑程序Π，设其内涵谓词的集合为σ，定义$Tr_{\neg\neg}(\Pi)$为按如下方式得到的析取逻辑程序：

1. 为σ中的每个谓词常元P引入一个辅助谓词P'，将Π中的每一个形如$\neg\neg P(\bar{x})$ $(P \in \sigma)$的双否定文字替换为$\neg P'(\bar{x})$；
2. 为σ中的每个谓词常元P引入规则$P'(\bar{x}) \leftarrow \neg P(\bar{x})$。

上述翻译$Tr_{\neg\neg}$中所使用的方法，在实际的逻辑程序设计中是一种相当常用的编码技巧，章衡将它以命题的形式说明了翻译前后的逻辑程序的等价性，并在其博士论文中证明了翻译的正确性。

命题 2.22： **(文献[?] 命题5.5)** 给定一个扩充的析取逻辑程序Π，设其内涵谓词的集合为σ，在不管辅助谓词集σ'的解释的情况下，$Tr_{\neg\neg}(\Pi)$与Π逻辑等价。

我们将在回答集语义下的析取逻辑程序称为回答集程序(Answer Set Programming, 简称ASP)。最近几年，随着布尔可满足性求解器(SAT求解器)的高速发展，基于SAT技术的回答集求解器也在不断发展，为找出回答集程序的回答集提供了一个高效的实现。目前较为流行的回答集求解器有ClaspD[?]、DLV[?]、GnT[?] 和Smodels[?]。

实际上，最近十几年来ASP一直是国内外学者的研究热点，这使得回答集程序的语法和语义都得到不同程度的扩展，包括$\#sum$约束、优化等等概念的引入，使得ASP能够应用于很多Σ_2^p的组合优化问题上。但这些扩展不在本文的讨论范围内，请参阅文献[? ? ?]。

参考文献

[1] Davis R, Shrobe H, Szolovits P. What is a knowledge representation? [J]. AI magazine, 1993, 14(1):17.

[2] McCarthy J. Programs with common sense [M]. Defense Technical Information Center, 1963.

[3] McCarthy J, Hayes P. Some philosophical problems from the standpoint of artificial intelligence [M]. Stanford University USA, 1968.

[4] McCarthy J. Circumscription—a form of nonmonotonic reasoning [J]. & &, 1987.

[5] Reiter R. A logic for default reasoning [J]. Artificial intelligence, 1980, 13(1):81–132.

[6] McDermott D, Doyle J. Non-monotonic logic I [J]. Artificial intelligence, 1980, 13(1):41–72.

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

[8] Green C. Application of theorem proving to problem solving [R]. DTIC Document, 1969.

[9] Colmeraner A, Kanoui H, Pasero R, et al. Un systeme de communication homme-machine en francais [C].//. Luminy. 1973.

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

[11] Clark K L. Negation as failure [G].// Logic and data bases. Springer, 1978: 293–322.

[12] Reiter R. On closed world data bases [M]. Springer, 1978.

[13] 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.

[14] Gelfond M, Lifschitz V. The stable model semantics for logic programming. [C].// ICLP/SLP. vol 88. 1988: 1070–1080.

[15] Gelfond M, Lifschitz V. Classical negation in logic programs and disjunctive databases [J]. New generation computing, 1991, 9(3-4):365–385.

[16] 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.

[17] Lin F, Zhao Y. ASSAT: Computing answer sets of a logic program by SAT solvers [J]. Artificial Intelligence, 2004, 157(1):115–137.

[18] Lee J, Lifschitz V. Loop formulas for disjunctive logic programs [G].// Logic Programming. Springer, 2003: 451–465.

[19] Lifschitz V, Razborov A. Why are there so many loop formulas? [J]. ACM Transactions on Computational Logic (TOCL), 2006, 7(2):261–268.

[20] Gebser M, Schaub T. Loops: relevant or redundant? [G].// Logic Programming and Nonmonotonic Reasoning. Springer, 2005: 53–65.

[21] Gebser M, Lee J, Lierler Y. On elementary loops of logic programs [J]. Theory and Practice of Logic Programming, 2011, 11(06):953–988.

[22] Ben-Eliyahu R, Dechter R. Propositional semantics for disjunctive logic programs [J]. Annals of Mathematics and Artificial intelligence, 1994, 12(1-2):53–87.

[23] 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.

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

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