中山大学硕士学位论文

一种回答集逻辑程序改进分割计算方法及程序化简的研究

Research on Program Splitting of ASP Program and Its Application in Program Simplification

学位申请人：霍子伟

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）：_____

_____

_____

_____

年　月　日

摘 要

人工智能是计算机科学中的一个重要分支，其中的知识表示与推理则是一门通过理解智能和认知本质，以最终通过计算机实现并体现人类智能为目的的学科。人工智能创始人之一的John McCarthy早在上世纪50年代末就开始推动知识表示的发展，他第一次提出了使用符号推理形式来刻画知识的认知和推理。

早年的知识表示是通过单调逻辑进行推理的，但很快人们就意识到人类常识的认知和推理是非单调的，所以非单调逻辑在产生至今都是知识表示的研究热点。本文研究的回答集编程（Answer Set Programming, ASP）则是非单调逻辑的重要内容。随着回答集编程问题的多年发展，其理论和求解器都已比较成熟。然而经过证明，回答集编程中判断正规逻辑程序和析取逻辑程序是否存在稳定模型的计算复杂性分别是**NP**-$complete$和Π_2^P-$complete$，所以回答集编程的主要发展瓶颈是求解器的效率问题。

在求解器提速的发展过程中，Lifschitz 和Turner在1994年时提出了分割集（splitting set）和程序分割（program splitting）的概念，并从理论上证明了ASP程序可以通过分割集被分割为bottom和top两部分，并通过这两部分的回答集计算原程序的回答集。分割集和程序分割的提出，为求解ASP回答集的提速发展带来了新思路。在后续的时间里，分割集和程序分割得到了不断的推广。然而，Lifschitz和Turner当初定义的分割集是基于强条件的，在实际程序中往往会出现一个ASP程序的分割集就只有空集和程序全部原子构成的集合，而这样的分割集对于分割程序是没有任何意义的。

基于这样的应用背景之下，本文对Lifschitz 和Turner提出的分割集和程序分割算法进行了深入的探讨和研究，对原来的分割集和程序分割方法进行了扩展，并把分割的概念引入到程序化简当中。本文所获得的主要成果具体如下：

首先，把原来Lifschitz 和Turner定义的强条件下的分割集扩展为可以是任意原子集，同时定义了新的程序分割方法。在分割集可以为任意原子集的情况下，程序分割的适用性可以大大地得到扩展。此外，本文通过分析得到了新分割集对新程序分割的性能影响，找出了主要性能瓶颈所在。同时，通过实验数据验证了使用新分割集和新程序分割方法计算一个ASP程序的回答集对比直接求解要快，提速效果大致维持在2到3倍。

其次，在分析新分割集对新程序分割的性能影响之时，得到结论：如果分割集中原子都被原程序的每个回答集所满足，那么使用这样的分割集来分割程序可

以大大降低原程序回答集的计算复杂性。基于这个发现，本文把分割集的应用拓展到了程序化简当中，即通过程序结论去化简ASP程序。这里的程序结论就是一个能被程序所有回答集所满足的文字集，本文只取正文字，以达到原子集的效果。程序化简的目的依旧是为了让回答集程序求解提速。

本文提出的新分割集与新程序分割给回答集程序的提速带来了实质性的效果，并由此推广分割集的概念到程序化简等实际应用当中，让回答集程序的求解效率有了长足的提升和新思路。

关键词：非单调逻辑，回答集编程，分割集，程序分割，程序化简

Abstract

Artificial intelligence is a very important subject in computer science. The knowledge representation and reasoning in artificial intelligence aims at completing and reflecting human intelligence in computer through the understanding of intelligence and cognitive nature. John McCarthy, one of the founders of artificial intelligence, has begun to promote the development of knowledge representation in the early 1950s. He first proposed the use of symbolic reasoning form to depict the cognitive and reasoning of knowledge.

Knowledge representation is through monotonic logic for reasoning at its early age. However, people came to realize that human cognition of common sense and reasoning is non-monotonic soon. In this case, non-monotonic logic became the mainstream tool of knowledge representation from then on. In this thesis, the answer set programming (ASP) which we studied is an import content of non-monotonic logic. With many years development of answer set programming problems, its theory and solver has been relatively mature. Nevertheless, the computational complexity of normal logic program (NLP) and disjunctive logic program (DLP) in ASP program under stab le model semantic are $\mathbf{NP}$-$complete$ and Π_2^P-$complete$ respectively after strict proven. Therefore, the main development bottleneck problem in answer set program is solver efficiency.

In the process of ASP solver speeding up development, Lifschitz and Turner proposed the concept of splitting set and program splitting in 1994. Moreover, they provided a method to divide a logic program into two parts which was named as bottom and top, and showed that the task of computing the answer sets of the program can be converted into the tasks of computing the answer sets of these parts. The concept of splitting set and program splitting brought new ideas to speed up solving answer sets of ASP program. In the subsequent period, splitting set and program splitting have been promoted and recognized continuously. However, the notion of splitting set which proposed by Lifschitz and Turner was based on strong conditions. Because of this, the empty set and the set of all atoms are the only two splitting sets for many ASP programs in the actual situation. These two splitting sets made no sense to speed up the answer sets

solving in ASP programs, because these splitting sets cannot divide programs by the splitting methods.

Based on this background, in this thesis, I carried on the deep discussion and research about Lifschitz and Turner's splitting set and program splitting theorem and extend them. Along this train of thought, I introduce the concept of splitting set to program simplification. The main achievements obtained in this thesis are shown as following details.

Firstly, I extend Lifschitz and Turner's splitting set and program splitting theorem to allow the program to be split by an arbitrary set of atoms and introduce a new program splitting method. As the splitting set can be an arbitrary set of atoms, the applicability of program splitting can be extended greatly. Besides, this thesis figure out the main performance bottlenecks through analyzing properties of the new splitting method. At the same time, the data coming from experiment in this thesis support the fact that using arbitrary set of atoms as splitting set and the new splitting method to divide ASP program and solve its answer sets is quicker than solve directly. More precisely, it is two to three times faster than the original according the experiment.

Secondly, during analyzing how the new splitting set effect on the performance of new splitting method, I found out that if atoms in the splitting set are satisfied by every answer set of the program, it could release the computational complexity of answer set solving. According to this, this thesis extends the usage of splitting set to program simplification. In the same words, we can use consequence of the ASP programs to simplify themselves. The consequence of an ASP program is a set of literals that are satisfied by every answer set of the program. I took only positive literals to make them as atoms. The purpose of program simplification is still to speed up solving answer sets of ASP program.

This thesis proposes the new splitting set and new splitting method to make contribution to speed up solving answer sets of ASP program which brings a substantial progress, and extends the concept of splitting set to practical application such as program simplification. All these make important improvement to solving ASP program.

Key Words: non-monotonic logic, answer set programming, splitting set,

program splitting, program simplification

目　录

第 1 章　引言

本章将以人工智能的发展过程为主线，层层递进地来介绍本文的研究背景和当前国内外在回答集编程方面的研究成果和发展前景。并描述本文主要的研究问题和工作成果及其对回答集编程领域的发展意义，最后简明扼要地给出本文后续章节的安排结构。

1.1　研究背景与现状

自计算机诞生以来，人们便致力于让计算机拥有智能，希望计算机能够模拟人类的大脑去思考和推理。如何让计算机认知和理解知识，并使用规则进行推理以求解问题是人工智能领域中最为困难但具有重要意义的一类问题，而这类问题被称为知识表示与推理（Knowledge Representation and Reasoning）[1]。经过多年来的研究和发展，知识表示这个领域已经得到了长足的发展，其最主要的描述和求解工具则是逻辑程序[2]。在发展过程中，更加符合人类常识推理模式的非单调逻辑替代了单调逻辑称为了主流的工具。

1955年，人工智能奠基人之一的John McCarthy发起了达特茅斯会议，正式在会议上提出了“人工智能”这个概念，并在1959年发明了LISP语言，这是第一个广泛流行于人工智能研究工作中的高级语言。LISP最大的特点是它所计算和处理的是符号表达式而不是数。这为逻辑程序的诞生和发展提供了基础。1951年，Aflred Horn 提出了霍恩(Horn)子句，这是带有最多一个肯定文字的析取范式。霍恩子句是逻辑程序的重要构成基础[3]。上世纪70年代，Kowalski率先提出了逻辑可以作为程序设计语言的基本思想。不久逻辑程序设计正式诞生[4]。逻辑程序只需要设置求解问题需要符合的规则和加入问题所有的前置事实即可求解问题，而非传统的高级语言那样使用“顺序、控制、循环”等步骤来解决问题。逻辑程序就是：事实+ 规则= 结果。基于Kowalski提出的逻辑程序思想，法国的Colmerauer在1979年研发出世界上第一种用于逻辑程序设计的语言－－PROLOG（PROgram in LOGic）[5]。LSIP和PROLOG对知识表示的

发展起了十分深远的影响，这两种语言事实上为计算机科学中编程语言发展提供了新的方向：即与当下流行的高级语言不同，并不是基于过程控制去演算一个问题的解，而是通过程序员从逻辑出发，刻画出一个问题是什么即可，至于怎么实现则由系统完成，然后问题便可以被求解。很遗憾，这个目标至今也没有被实现[2]。但可以知道逻辑程序是一种更为贴合人类日常语言模式的编码方式。同时逻辑程序可以帮助我们在编程上从编写“怎么做”到“做什么”的转变[6]。

然而，早期的经典逻辑中，以命题逻辑和一阶逻辑为主，都是通过已知的事实推出结论，并不会因为已知事实的增加而使得之前的结论丧失其有效性，因此是单调的(monotonic)，知道的事实越多，结论就越多。但人类在正常认知的过程中，当前的认知并不一定是事实或真理，所以一旦得到新的修正认知，之前的结论就存在被否定的可能性和需要，由于新认知的加入不一定会让结论增多，所以这样的逻辑称为非单调逻辑(non-monotonic logic)[7]。就如这样一个例子：一个人相信树的叶子在冬天都会掉光，他看到的树也都是这样子；当他看到柏树的树叶在冬天没有掉光时，他不会认为柏树不是树，而是会否定以前的结论：并非所有树的叶子都会在冬天掉光。所以说非单调逻辑才更符合人类的日常认知模式。

1978年，Keith Clark提出了失败即否定（Negation as Failure）。此理论补全了逻辑程序中的否定问题的表达困难[8]。但彼时尚未能马上得到失败即否定的模型语义。Gelfond和Lifschitz在1988年提出稳定模型语义（stable model semantics）后，填补了非单调逻辑领域对失败即否定的解释[9]。基于Gelfond和Lifschitz的稳定模型语义，及其后的发展，在上世纪90年代前后，形成了一种新的逻辑程序设计方法，即回答集编程（Answer Set Programming，ASP）[10]。

1.2 国内外研究现状

ASP在发展过程中不断地被扩展，并且拥有了一系列高效的求解器。主要分为两大类。一类是基于DPLL算法的：DLV、smodels、clasp及其扩展claspD、clingo及其扩展iclingo；另一类是基于SAT求解方法的：ASSAT和cmodels。ASP领

域在过去20年里可谓发展迅速，求解器的速度有了很大的提升。由于求解器的提速，ASP得以被广泛应用在实际项目中。2001年，NASA决定在航天决策系统中使用ASP[11]；2003年时，Eiter和Lenoe把ASP应用在了行为决策中[12]；Soininen和Niemela则尝试了在产品配置上运用ASP，并取得了不错效果[13]。而国内也有不少关于ASP的应用尝试。2010年，华南理工的李鑫为了克服E-R模型不具有自动推理能力的缺陷，提出了一种利用ASP来表示E-R模型的方法[14]；2012年，华南师范大学的赖河蒲在Banks选举问题上使用了ASP取代启发式算法进行求解，得到了良好的结果[15]；中科大的吉建民老师及其团队长久以来都把ASP技术运用到机器人控制上[2]。

虽然ASP已经发展相对成熟，但近几年来，国内外对ASP的研究重心依旧围绕着ASP求解器的提速问题。Lifschitz和Turner在1994年提出了分割集（splitting set）的概念，以及相应的程序分割（program splitting）方法，即通过分割集把原逻辑程序分割为bottom和top两部分，并证明了原程序的回答集可以通过该两部分的回答集求解得到[16]。分割集的思想很快被应用到ASP领域的多个方向，同时，分割集和程序分割能为ASP求解器的提速带来新思路。

1.3 本文的工作及意义

Lifschitz和Turner提出的分割集和程序分割方法为ASP的理论扩展和ASP求解器提速都带来了帮助。然而Lifschitz和Turner所定义的分割集对于很多ASP程序而言只有空集和全部原子构造成的集合（全集）两种情况。分割集为空集或全集都无法进行程序分割，这样便会导致分割集变得毫无意义。本文基于这个缺陷，对Lifschitz和Turner的理论展开了研究，并对其进行了扩展和应用，具体的工作如下：

1. 首先对ASP领域的基础知识进行梳理总结。首先详细介绍说明ASP中的重要理论概念，如：失败即否定、稳定模型语义、极小模型等。同时，详细分析了Lifschitz和Turner的分割理论，并剖析其分割集的缺陷所在。

2. 在深入研究Lifschitz和Turner的理论后，对原来的分割集进行扩展，具体是提出了新的分割集，新分割集可以为关于原程序的任意原子集。

3. 此外，本文提出了一种可以基于分割集为任意原子集的新程序分割方法。这种新的程序分割方法把bottom和top从之前的简单互补关系进行了扩展。在计算top部分时会引入新原子辅助，但不会影响最终结果。

4. 同时，本文对新的程序分割方法进行了计算复杂性的分析，指出整个分割和求解过程中的主要耗时部分，并给出如何设计分割集可以帮助求解提速的结论，同时通过实验支持了使用新程序分割方法求解原程序的回答集可以带来显著的提速效果。

5. 在分析新的程序分割的计算复杂性时，得出了如果分割集为程序结论的话，可以大大降低复杂性。由此为思路，提出了通过程序结论去进行程序化简，为程序化简问题引入分割集的概念，并得到理想的效果。

分割集思想和理论自提出以来就被认为是研究回答集语义的一个良好工具[17]。Gebser在2008年以分割集理论作为基础实现了增量式ASP求解器——iclingo[18]。此外，Oikarinen和Janhunen把分割集的思想拓展到了带嵌套表达式的逻辑程序中[19]，Ferraris则在稳定模型语义下的任意一阶逻辑中引入了分割集的使用[20]。分割集的思想对ASP领域有着重要的影响，所以本文将分割集扩展到任意原子集，并提出相应的新程序分割方法，让原来分割集的局限性得意打破，同时本文把新的分割集思想延伸到实际应用中——程序化简。对ASP求解提速有莫大的意义。

1.4 本文的安排结构

本文首先介绍了人工智能的发展过程及回答集编程产生的背景，同时说明了回答集编程当前的发展情况，然后引出分割集和程序分割的概念，详细分析Lifschitz和Turner的分割集的局限性后，提出自己的新分割集和新程序分割方法，然后将其应用到程序化简中。此外，本文进行了两个实验。第一个为对比使

用程序分割方法求解原程序回答集和直接求解的效率，结果为使用程序分割的效率更好；第二个为在程序化简中引入分割集的概念，并比较求解化简后的程序跟原程序的效率对比，结果为化简后的求解更快。具体的章节安排如下：

第一章给出了本文的研究背景和现状，简要地阐述了从人工智能到回答集编程这个领域诞生的整个过程，指出了当前回答集编程的发展情况和发展方向——求解器提速。同时，指出了本文的主要工作和工作内容对回答集编程发展的意义。

第二章介绍了回答集编程的基础知识，主要讲及失败即否定和稳定模型语义，以及回答集编程自身的语义。此外，还会引入说明回答集编程的特性，如：环与环公式，分割集和程序分割的基本概念。

第三章详细地讲述本文提出的新分割集和新程序分割的概念和思想。同时会对程序化简过程的计算复杂性进行剖析，指出其中的主要耗时点，并给出提升的办法。

第四章展现了新分割集和新程序分割在回答集编程中的应用，并给出了这些应用的具体细节。

第五章根据第四章的应用设计相关实验，并通过实验数据分析新分割集和新程序分割的使用情况。

第六章是本文的收尾部分，给出了对全文的总结和对分割集思想的后续工作进行展望，列举出一些具体的可尝试方向。

第 2 章　预备知识

本章给出了本文将涉及的基础知识，先从命题逻辑进行介绍，引入失败即否定，并定义ASP中的普通规则，及规则中各个部分涉及的命名符号。然后给出“满足”的概念。完成基础介绍后，进入回答集语义，即如何定义ASP程序的回答集。然后列出一些ASP程序的主要性质，主要是环与环公式，及本文的核心内容分割集与程序分割。

2.1　命题逻辑

命题（**proposition**）是非真即假的陈述句。我们一般使用字母代表一个命题。命题逻辑由命题公式和一套证明规则所组成，其中命题公式和证明规则就是命题逻辑的运算本体和运算规则[21]。

定义 2.1： 命题逻辑的符号包括[22]：

- 命题符号：A，B，C，...，统称为$\aleph$
- 真值符号：$true$，$false$；
- 连接词：$\wedge$，$\vee$，$\neg$，$\rightarrow$，$\equiv$。

在ASP逻辑程序中，我们也把命题符号称为一个原子。

定义 2.2 (文字)： 一个命题符号或者一个命题符号的否定。其中一个命题符号为正文字，一个命题符号的否定为负文字。

命题的真假性被称为其真值（truth value），真值只有真（true）和假（false）两个。每个命题只能为真或者为假，不能既是真又是假，或者既不是真又不是假。通常地，在求解逻辑程序中，我们所得到的结果就是指能够被确定真值为真的命题集合。

在定义2.1中的连接词的作用是把若干个文字连接成命题公式。每个连接词都有自己的真值表，可以概括如下[22]：

1. 非（¬），真假性与其操作的命题相反，当A为真时，$\neg A$为假，当A为假时，$\neg A$为真；

2. 合取（∧），$A \wedge B$为真，当且仅当命题A和B同时为真时；

3. 析取（∨），$A \vee B$为真，当且仅当命题A和B中至少有一个为真；

4. 蕴涵（→），$A \rightarrow B$为假，当且仅当命题A为真且B为假。

命题公式由一个文字或者多个文字通过连接词组成。命题公式是命题逻辑的推理基础。在有需要的情况下，命题公式可以通过对合律、德?摩根定律、结合律和分配律等运算性质得到逻辑等价的范式，本文涉及的范式有合取范式和析取范式。

定义 2.3 (范式)： 合取范式（$ConjunctiveNormalForm, CNF$）：一系列析取式的合取形式；析取范式（$DisjunctiveNormalForm, DNF$）：一系列合取式的析取形式；其中析取式（合取式）为若干文字只通过连接词∨（）∧)进行连接。

2.2 逻辑程序

逻辑程序是知识表示的基础。早期研究人工智能的学者都详细可以找到一种以符号刻画知识和推理过程的方法以达到模拟人类大脑推理的过程[2]。而事实上，逻辑程序便是这样的一种手段。逻辑程序的形式有很多。而在ASP逻辑程序中，主要用及的便是引入失败即否定和经典否定的命题公式。

ASP程序一般分为两部分：事实集（Facts）和ASP逻辑程序。要求解一个ASP程序的回答集，需要结合使用事实集和ASP逻辑程序。

ASP程序的求解过程是：

1. 使用例化工具（常用的有gringo和lparse）通过事实集例化ASP逻辑程序；

2. 对例化后的逻辑程序，调用求解器进行求解。

从实际效果来看，例化后的逻辑程序就是不含变量的命题公式集合。本文中只考虑完全例化后的长度有限的逻辑程序。本文所探讨的ASP逻辑程序由有限个规则（rule）所组成，其中规则的形式如下：

$$a_1 \vee a_2 \vee ... \vee a_k \leftarrow a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n. \tag{2.1}$$

其中$n \geq m \geq k \geq 1$，a_i为原子（atom），not代表失败即否定。在k，m，n取不同范围的值时，形式(2.1)所表示的规则有不同的意义，具体如下：

1. k = 0时，规则被称为限制（Constraint）；
2. k = 1时，规则被称为正规规则（Normal Rule）；
3. n = m = 0时，规则被称为事实（Fact）；
4. n = m时，即规则中没有带not的原子时，规则称其为正规则（Positive Rule）。

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

其中$\bar{x}$是变元元组，$P^* < P$表示P^*的外延是P的外延的子集。

进一步，求解限定理论时，通常会有谓词集合的比较，设σ和σ^*都是谓词常元集，且$|\sigma| = |\sigma^*|$，

$$\sigma^* = \sigma\text{表示对于任意的}P \in \sigma\text{和}P^* \in \sigma^*\text{，满足}P^* = P,$$
$$\sigma^* \leq \sigma\text{表示对于任意的}P \in \sigma\text{和}P^* \in \sigma^*\text{，满足}P^* \leq P,$$
$$\sigma^* < \sigma\text{表示}(\sigma^* \leq \sigma) \wedge \neg(\sigma^* = \sigma),$$

其中P^*与P是一一对应的，且具有相同的元数。

设φ是一个一阶语句，σ_i是极小（minimized）谓词常元集，可变（varied）常元集σ_v是包含个体、函词或不在σ_i中出现的谓词常元集，表示谓词常元不能同

时是极小的和可变的，φ的符号集中剩余的符号被称作固定（fixed）常元。一个限定理论是一个三元组$(\varphi;\sigma_i;\sigma_v)$，记为$\mathbf{CIRC}[\varphi;\sigma_i;\sigma_v]$，如果$\varphi$是命题公式，则称为命题限定理论；如果$\varphi$是一阶语句，则称为一阶限定理论；如果$\sigma_v$为空，则称为标准限定理论，可以简写为$\mathbf{CIRC}[\varphi;\sigma_i]$。

定义 2.4： 并行限定定义为如下一个二阶公式：

$$\mathbf{CIRC}[\varphi;\sigma_i;\sigma_v] = \varphi \wedge \forall\sigma_i^*\sigma_v^*(\sigma_i^* < \sigma_i \rightarrow \neg\varphi(\sigma_i^*,\sigma_v^*)) \tag{2.2}$$

其中，σ_i^*是一个谓词变元集，这些谓词变元分别对应于σ_i中的谓词且常元具有相同的元数；类似地，σ_v^*是变元集，分别对应于σ_v中的常元，且具有相同的元数；$\varphi(\sigma_i^*,\sigma_v^*)$是将公式$\varphi$中在$\sigma_i$（或$\sigma_v$）中出现的常元替换成对应的$\sigma_i^*$（或$\sigma_v^*$）中的变元。

公式(2.2)的直观含义是，满足公式φ正确的条件下，使得σ_i中谓词的外延最小，称这样的公式φ的模型为$\mathbf{CIRC}[\varphi;\sigma_i;\sigma_v]$的模型。

令v为上述语句φ的符号集，设$\mathfrak{A}$和$\mathfrak{B}$是任意两个v-结构，定义$\mathfrak{A} \preceq_{\sigma_i,\sigma_v} \mathfrak{B}$当且仅当以下条件成立：

1. $\mathfrak{A}$与$\mathfrak{B}$具有相同的论域；
2. 对于v中所有的极小谓词常元P，均有$\mathfrak{A}(P) \subseteq \mathfrak{B}(P)$；
3. 对于v中所有固定的常元C，均有$\mathfrak{A}(C) = \mathfrak{B}(C)$。

换句话说，$\mathfrak{A} \preceq_{\sigma_i,\sigma_v} \mathfrak{B}$意味着$\mathfrak{A}$和$\mathfrak{B}$仅在解释$\sigma_i$和$\sigma_v$上不同。

关系$\preceq_{\sigma_i,\sigma_v}$具有自反律和传递律，令$\mathfrak{A} \prec_{\sigma_i,\sigma_v} \mathfrak{B}$表示$\mathfrak{A} \preceq_{\sigma_i,\sigma_v} \mathfrak{B}$成立但是$\mathfrak{B} \preceq_{\sigma_i,\sigma_v} \mathfrak{A}$不成立，如果结构$\mathfrak{A}$是语句$\varphi$的模型，且$\varphi$不存在模型$\mathfrak{B}$满足$\mathfrak{B} \prec_{\sigma_i,\sigma_v} \mathfrak{A}$。

设σ_i为一个谓词常量集合，将σ_i分割成若干不相交的段$\sigma_i^1,...,\sigma_i^k$。想使谓词$\sigma_i^1$极小化的优先级高于谓词$\sigma_i^2$，谓词$\sigma_i^2$极小化的优先级高于$\sigma_i^3$，$\sigma_i^m$的优先级

高于σ_i^n（$1 \leq n < m \leq k$）。对于公式φ，σ_v为变化的谓词集，$\sigma_i^1 > ... > \sigma_i^k$为极小化谓词集的优先级限定记为$\mathbf{CIRC}[\varphi; \sigma_i^1 > ... > \sigma_i^k; \sigma_v]$，同样可以定义成与公式(2.2)相似的二阶公式。优先级限定与并行限定有如下的关系性质：

命题 2.5： **(文献[24]命题15)**优先级限定$\mathbf{CIRC}[\varphi; \sigma_i^1 > ... > \sigma_i^k; \sigma_v]$ 等价于并行限定

$$\bigwedge_{j=1}^{k} \mathbf{CIRC}[\varphi; \sigma_i^j; \sigma_i^{j+1}, ..., \sigma_i^k, \sigma_v]. \tag{2.3}$$

下面给出一个具体地例子介绍限定理论的相关概念。

例 2.1： (Reiter的例子，文献[25] 第7节)设φ为如下一阶公式的全称闭包：

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

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

其中谓词$Quaker(x)$表示x是教徒，谓词$Republican(x)$表示x是共和党人，谓词$Pacifist(x)$ 表示x是和平主义者，谓词Ab_1表示一种不正常因素，谓词Ab_2表述另一种不正常因素。如果一个人$Nixon$同时是教徒和共和党人，对Ab_1和Ab_2限定，不能得到他是不是一个和平主义者，但是，如果使用优先级限定，则说明$Nixon$更像一个教徒而非共和党人，可以通过计算$\mathbf{CIRC}[\varphi; Ab_1 > Ab_2; Pacifist]$得到$Nixon$是一个和平主义者。

2.3 稳定模型语义

本小节介绍稳定模型语义（Stable Model Semantics）下的一阶与命题理论。定义任意稳定模型语义是一个二元组$(\varphi; \sigma_i)$，其中φ是一阶语句或者命题公式，σ_i是在φ中出现的谓词常元集合，这些谓词称为内涵谓词常元，除此之外的常元称为外延常元。

定义 2.6： $SM[\varphi; \sigma_i]$定义为如下的二阶公式：

$$\varphi \wedge \forall \sigma_i^* (\sigma_i^* < \sigma_i \rightarrow \neg St(\varphi; \sigma_i)) \tag{2.6}$$

其中$St(\varphi; \sigma_i)$递归地定义如下：

- 如果$P \in \sigma_i$，则$\mathrm{St}(P(\overline{x});\sigma_i) = P^*(\overline{x})$；
- 如果谓词F不在σ_i中出现，则$\mathrm{St}(F(\overline{x});\sigma_i) = F(\overline{x})$；
- 如果$\circ \in \{\wedge, \vee\}$，则$\mathrm{St}(\psi \circ \chi;\sigma_i) = \mathrm{St}(\psi;\sigma_i) \circ \mathrm{St}(\chi;\sigma_i)$；
- $\mathrm{St}(\psi \rightarrow \chi;\sigma_i) = (\mathrm{St}(\psi;\sigma_i) \rightarrow \mathrm{St}(\chi;\sigma_i)) \wedge (\psi \rightarrow \chi)$；
- 如果$Q \in \{\forall, \exists\}$，则$\mathrm{St}(Qx\psi;\sigma_i) = Qx\mathrm{St}(\psi;\sigma_i)$；

如果结构μ是$\mathbf{SM}[\varphi;\sigma_i]$的一个模型，则被称为$\varphi$的一个$\sigma_i$-稳定模型。令SM[FO]表示所有一阶稳定理论对应的二阶公式。

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$\mathfrak{A}$是φ关于σ_i的稳定模型（Stable Model）当且仅当$\mathfrak{A}$是公式$\mathbf{SM}[\varphi,\sigma_i]$的模型。

稳定模型语义下前束范式的获取与经典语义等价关系有所不同，根据文献[26]，如果两个一阶公式φ和ψ是强等价的，当且仅当对于任意的公式γ以及任意一个谓词常元集σ_i，将公式φ在γ中的任意出现都替换成ψ后得到公式γ'，满足$\mathbf{SM}[\gamma;\gamma_i]$等价于$\mathbf{SM}[\gamma';\gamma_i]$。结合强等价的定义，有如下稳定稳定模型语义下的性质。

命题 2.7： **(文献[27]定理6.4)**设φ和ψ为任意的一阶公式，则下列等价关系成立：

(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。

章衡[28]在近几年的研究中提出了将稳定模型语义下一阶语句翻译成稳定模型语义下的全称一阶语句的方法，即消去了存在量词。该方法的主要思想如下定义：

定义 2.8： **(文献[28]定义5.3)** 任意给定形如$\forall\bar{x}\exists\bar{y}\theta(\bar{x},\bar{y})$的一个一阶语句$\varphi$，定义$tr_{QE}(\varphi)$为下述公式的合取式的一阶全称闭包：

$$\neg\neg S(\bar{x},\overline{min}) \tag{2.7}$$

$$(\overline{succ}(\bar{y},\bar{z})\ \wedge\ S(\bar{x},\bar{z}))\ \vee\ \theta^{\neg\neg}(\bar{x},\bar{y})\ \rightarrow\ S(\bar{x},\bar{y}) \tag{2.8}$$

$$T(\bar{x},\overline{min})\ \vee\ \theta(\bar{x},\overline{min}) \tag{2.9}$$

$$\neg(\overline{succ}(\bar{y},\bar{z})\ \wedge\ S(\bar{x},\bar{z}))\ \wedge\ S(\bar{x},\bar{y})\ \rightarrow\ (T(\bar{x},\overline{max})\ \leftrightarrow\ \theta(\bar{x},\bar{y})) \tag{2.10}$$

$$\overline{succ}(\bar{y},\bar{z})\ \rightarrow\ (T(\bar{x},\bar{y})\ \leftrightarrow\ \theta(\bar{x},\bar{z})\ \vee\ T(\bar{x},\bar{z})) \tag{2.11}$$

令n为序列$\bar{x}$与$\bar{y}$的长度之和，其中：

1. $\theta^{\neg\neg}$是由θ将所有原子公式$P(\bar{t})$替换为$\neg\neg P(\bar{t})$后得到的公式；
2. $\overline{min}$和$\overline{max}$分别表示n元组$(min,...,min)$和$(max,...,max)$，$\overline{succ}$描述$succ$所定义序被推广到n元组后得到的后继关系；
3. S和T是不在φ中出现的两个n元新谓词常元。

对任意一个一阶稳定理论$(\varphi;\sigma)$，若φ是一个形如$\forall\bar{x}\exists\bar{y}\theta$的一阶语句，定义$Tr_{QE}(\mathbf{SM}[\varphi;\sigma])\ =\ \exists S\exists T\mathbf{SM}[tr_{QE}(\varphi);\sigma,S,T]$，其中$S,T$是由$tr_{QE}$引入的辅助谓词。

Tr_{QE}被证明在后继结构上保持等价关系。在这个存在量词消去的定义2.8中，引入了三个具有后继结构的常元符号，分别为min、max和$succ$，min和max是个体常元，$succ$是谓词常元。参考文献[29]给出后继结构的定义：

定义 2.9： 称一个结构$\mathfrak{A}$是一个后继结构当且仅当满足下面两个条件同时成立：

1. $\mathfrak{A}$的符号集是一个后继符号集；
2. 在关系$\mathfrak{A}(succ)$，论域中每一个元素至多有一个直接前继和一个直接后继，且无前继当且仅当该元素为$\mathfrak{A}(min)$，无后继当且仅当该元素为$\mathfrak{A}(max)$。

后继结构可以类比一个正整数结构$\mathbb{Z}_n$，其中$n = |\mathbb{Z}_n|$，例如集合$\{0, ..., n-1\}$，$\mathfrak{A}(succ)$对应该集合上的后继关系，$\mathfrak{A}(min)$和$\mathfrak{A}(max)$分别对应0和$n-1$。在后续的章节中，会使用到这一类后继结构，假定一般的公式符号集中不存在这三个常元符号。

章衡[29]同样给出了一阶限定理论到一阶稳定理论的翻译，此翻译没有处理可变谓词的情形，但是给本文的研究提供了重要的翻译思路。

定义 2.10： **(文献[28]定义6.1)** 设v为谓词常元的一个有穷集，分情况进行如下定义：若φ为无蕴含否定范式，令F_φ为不在φ中出现的一个0元辅助谓词，令$\hat{tr}^v_{c2sm}(\varphi)$为如下公式：

$$\varphi^{\neg\neg} \wedge \varphi^{-} \wedge F_\varphi \rightarrow \bigwedge_{Q \in v} \forall \bar{x}(Q(\bar{x}) \vee \neg Q(\bar{x})) \tag{2.12}$$

其中：

1. $\varphi^{\neg\neg}$为将φ中所有形如$P(\bar{t})$（$P \in v$）的公式替换为$\neg\neg P(\bar{t})$后所得的公式；
2. φ^{-}为将φ中所有形如$\neg P(\bar{t})$（$P \in v$）的公式替换为$(P(\bar{t}) \rightarrow F_\varphi)$后所得的公式。

定理 2.1： $\hat{Tr}_{c2sm}$是从CIRC$[FO]$到SM$[FO]$保持一般结构上投影等价的翻译。

2.4 析取逻辑程序

逻辑程序是一种知识表示语言，一个逻辑程序是由多个规则组成，一个规则分为两部分：头部（$Head$）和体部（$Body$），如果体部不为空，使用符号$\leftarrow$将其区分：

$$Head \leftarrow Body.$$

如果体部为空，则可以省略符号$\leftarrow$。

一个正规逻辑程序（Normal Logic Program，记为LP[¬]）由多个子句构成，子句形式为

$$C \leftarrow L_1, ..., L_k. \tag{2.13}$$

其中头部C是正文字，体部$L_1, ..., L_k (k \geq 0)$是文字。一个文字要么是一个原子（即正文字），要么是形如**not** A的公式，其中A 是原子。

一个标准析取逻辑程序（Standard Disjunction Logic Program，记为DLP）由多个子句构成，子句形式为

$$A_1 \vee \cdots \vee A_k \leftarrow B_1, ..., B_m. \tag{2.14}$$

其中$k \geq 1, m \geq 0$，$A_i (1 \leq i \leq k)$、$B_i (1 \leq i \leq m)$都是原子。

一个正规析取逻辑程序（Normal Disjunction Logic Program，记为DLP[¬]）由多个子句构成，子句形式为

$$A_1 \vee \cdots \vee A_k \leftarrow B_1, ..., B_m, \textbf{not } C_1, ..., \textbf{not } C_n. \tag{2.15}$$

其中$k \geq 1, m \geq 0, n \geq 0$，$A_i (1 \leq i \leq k)$、$B_i (1 \leq i \leq m)$、$C_i (1 \leq i \leq n)$都是原子。

一个扩展析取逻辑程序（Extended Disjunction Logic Program，记为DLP[¬¬]）由多个子句构成，子句中同时包含了经典的¬和“否定即失败”的**not**，子句形式为

$$A_1 \vee \cdots \vee A_k \leftarrow B_1, ..., B_m, \textbf{not } C_1, ..., \textbf{not } C_n. \tag{2.16}$$

其中$k \geq 1, m \geq 0, n \geq 0$，$A_i (1 \leq i \leq k)$、$B_i (1 \leq i \leq m)$、$C_i (1 \leq i \leq n)$都是文字。

对于形如式(2.16)的规则，等价于如下一阶语句：

$$\forall \bar{x}(B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n \rightarrow A_1 \vee \cdots \vee A_k) \tag{2.17}$$

其中$\bar{x}$是$A_i (1 \leq i \leq k)$、$B_i (1 \leq i \leq m)$、$C_i (1 \leq i \leq n)$中出现的所有变元元组。这样一个逻辑程序等价一阶语句就是该程序所有规则对应的一阶语句的合取（因为

逻辑程序中所有的语句都应成立，所以所有的这些一阶表示语句应该是合取的关系）。

下面介绍逻辑程序的稳定模型语义，首先先介绍$Herbrand$模型。设Π为一个逻辑程序，包含的Π中所有常量的集合称为Π的$Herbrand$域（Herbrand universe），常项（ground term）是没有变量出现的项，因此$Herbrand$域包含了所有常项。一个常原子（ground atom）是一个谓词中仅出现常项的原子。$Herbrand$基（Herbrand base）是包含所有常原子的集合。设Π的一个$Herbrand$解释为$\mathfrak{I}$，该解释是取Π的$Herbrand$域为论域，将Π中的常项和常原子指派为它们自己，将Π中的谓词指派成$Herbrand$域中相同元数的关系，如果$\mathfrak{I} \models \Pi$，则$\mathfrak{I}$是Π的一个$Herbrand$模型（Herbrand model）。例如，逻辑程序$\Pi = \{p(1), q(2), q(x) \leftarrow p(x)\}$有两个$Herbrand$模型：

$$\{p(1), q(1), q(2)\} \tag{2.18}$$

和

$$\{p(1), p(2), q(1), q(2)\} \tag{2.19}$$

Π的一个$Herbrand$模型是极小的，当且仅当该模型的真子集都不是Π的$Herbrand$模型。上述例子中，(2.18)是极小模型，而(2.19)不是极小模型。具体关于Herbrand理论可以参考文献[30]。

定理 2.2： （**文献[9]定理1**）设Π为一个逻辑程序，Π的任意一个稳定模型都是Π的一个极小$Herbrand$模型。

定理2.2给出了逻辑程序的稳定语义，Π的符号集为v，一个结构$\mathfrak{A}$是Π关于v的一个稳定模型当且仅当$\mathfrak{A}$是$\mathbf{SM}[\Pi; v]$的模型。

在文献[31]中，Cabalar等人给出了从任意全称稳定理论转化为析取逻辑程序的方法，主要包括两个转换规则集，为了不失一般性，在规则集中所有的蕴含式都先被转换成稳定语义下的否定范式。有如下将否定前缀向内移动，直到变成NNF的规则：

$$\neg\top \iff \bot \quad \text{(N1)} \qquad \neg\bot \iff \top \quad \text{(N2)}$$

$$\neg\neg\neg\varphi \iff \neg\varphi \quad \text{(N3)} \qquad \neg(\varphi \wedge \psi) \iff \neg\varphi \vee \neg\psi \quad \text{(N4)}$$

$$\neg(\varphi \vee \psi) \iff \neg\varphi \wedge \neg\psi \quad \text{(N5)} \qquad \neg(\varphi \to \psi) \iff \neg\neg\varphi \wedge \neg\psi \quad \text{(N6)}$$

下面列出两个主要的规则集，在下面的规则表示中，将α看成合取式，β看成析取式（如果不是，就把他们当成$\alpha \wedge \top$，$\beta \vee \bot$）。

左规则：

$$\top \wedge \alpha \to \beta \iff \alpha \to \beta \quad \text{(L1)}$$

$$\bot \wedge \alpha \to \beta \iff \emptyset \quad \text{(L2)}$$

$$\neg\neg\varphi \wedge \alpha \to \beta \iff \alpha \to \neg\varphi \vee \beta \quad \text{(L3)}$$

$$(\varphi \vee \psi) \wedge \alpha \to \beta \iff \left\{ \begin{array}{l} \varphi \wedge \alpha \to \beta \\ \psi \wedge \alpha \to \beta \end{array} \right\} \quad \text{(L4)}$$

$$(\varphi \to \psi) \wedge \alpha \to \beta \iff \left\{ \begin{array}{l} \neg\varphi \wedge \alpha \to \beta \\ \psi \wedge \alpha \to \beta \\ \alpha \to \varphi \vee \neg\psi \vee \beta \end{array} \right\} \quad \text{(L5)}$$

右规则：

$$\alpha \to \bot \vee \beta \iff \alpha \to \beta \quad \text{(R1)}$$

$$\alpha \to \top \vee \beta \iff \emptyset \quad \text{(R2)}$$

$$\alpha \to \neg\neg\varphi \vee \beta \iff \neg\varphi \wedge \alpha \to \beta \quad \text{(R3)}$$

$$\alpha \to (\varphi \wedge \psi) \vee \beta \iff \left\{ \begin{array}{l} \alpha \to \varphi \vee \beta \\ \alpha \to \psi \vee \beta \end{array} \right\} \quad \text{(R4)}$$

$$\alpha \to (\varphi \to \psi) \vee \beta \iff \left\{ \begin{array}{l} \varphi \wedge \alpha \to \psi \vee \beta \\ \neg\psi \wedge \alpha \to \neg\varphi \vee \beta \end{array} \right\} \quad \text{(R5)}$$

对于规则(R5)，当$\beta = \bot$时，会发现第二条规则遵从第一条规则，因此，可以有一个特殊的版本如下：

$$\alpha \to (\varphi \to \psi) \iff \varphi \wedge \alpha \to \psi \quad \text{(R5′)}$$

回答集程序（Answer Set Programming，简写为ASP）是基于稳定模型语义的逻辑程序，使用陈述语句编写的形式，用于求解较难（主要是NP-难问题）的搜索问题。最近几年，随着回答集求解器的出现，答集程序在知识表示方面发展成为了一个可行方案，目前较为流行的回答集求解器有ClaspD[32]、DLV[33]、GnT[34] 和Smodels[35]。

参考文献

[1] Van Harmelen F, Lifschitz V, Porter B. Handbook of knowledge representation [M]. Elsevier, 2008.

[2] 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.

[3] Horn A. On sentences which are true of direct unions of algebras [J]. The Journal of Symbolic Logic, 1951, 16(01):14–21.

[4] 刘富春. 关于逻辑程序不动点语义的讨论[J]. 广东工业大学学报, 2005, 22(2):120–124.

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

[6] Clark K L, Tärnlund S A. Logic programming [M]. Academic Press New York, 1982.

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

[8] Clark K L. Negation as failure [J]. Logic and data bases, 1978, 1:293–322.

[9] Gelfond M, Lifschitz V. The stable model semantics for logic programming [C].// Proceedings of the 5th International Conference on Logic programming. vol 161. 1988.

[10] 翟仲毅, 程渤. 回答集程序设计: 理论, 方法, 应用与研究[J]. 2014.

[11] Nogueira M, Balduccini M, Gelfond M, et al. An A-Prolog decision support system for the Space Shuttle [C].// Practical Aspects of Declarative Languages. Springer, 2001: 169–183.

[12] Eiter T, Faber W, Leone N, et al. Answer set planning under action costs [J]. Journal of Artificial Intelligence Research, 2003, 25–71.

[13] Soininen T, Niemelä I, Tiihonen J, et al. Representing Configuration Knowledge With Weight Constraint Rules. [J]. Answer Set Programming, 2001, 1.

[14] Xin L, Fan L, Xingbin B, et al. Answer Set Programming Representation for ER Model [J]. Journal of Computer Research and Development, 2010, 1:025.

[15] 赖河蒨, 陈红英, 赖博先, et al. 基于回答集编程的Banks 选举求解方法[J]. 计算机工程, 2013, 39(8):266–269.

[16] Lifschitz V, Turner H. Splitting a Logic Program. [C].// ICLP. vol 94. 1994: 23–37.

[17] Dao-Tran M, Eiter T, Fink M, et al. Modular nonmonotonic logic programming revisited [C].// Logic Programming. Springer, 2009: 145–159.

[18] Gebser M, Kaminski R, Kaufmann B, et al. Engineering an incremental ASP solver [C].// Logic Programming. Springer, 2008: 190–205.

[19] Oikarinen E, Janhunen T. Achieving compositionality of the stable model semantics for smodels programs [J]. Theory and Practice of Logic Programming, 2008, 8(5-6):717–761.

[20] Ferraris P, Lee J, Lifschitz V, et al. Symmetric Splitting in the General Theory of Stable Models. [C].// IJCAI. vol 9. 2009: 797–803.

[21] 李未, 黄雄. 命题逻辑可满足性问题的算法分析[J]. 计算机科学, 1999, 26(3):1–9.

[22] Luger G F. 人工智能: 复杂问题求解的结构和策略: 英文版· [M]. 机械工业出版社, 2003.

[23] Enderton H, Enderton H B. A mathematical introduction to logic [M]. Academic press, 2001.

[24] Lifschitz V. Circumscription [C].// Handbook of logic in artificial intelligence and logic programming. vol 3. Oxford University Press, Inc. 1994: 297–352.

[25] McCarthy J. Applications of circumscription to formalizing common-sense knowledge [J]. Artificial Intelligence, 1986, 28(1):89–116.

[26] Ferraris P, Lee J, Lifschitz V. Stable models and circumscription [J]. Artificial Intelligence, 2011, 175(1):236–263.

[27] Pearce D, Valverde A. A first order nonmonotonic extension of constructive logic [J]. Studia Logica, 2005, 80(2):321–346.

[28] 章衡. 非单调逻辑的可判定性与可译性[D]. 北京: 清华大学, 2011.

[29] Zhang H, Zhang Y, Ying M, et al. Translating first-order theories into logic programs [C].// Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two. AAAI Press. 2011: 1126–1131.

[30] Ben-Ari M. Mathematical logic for computer science [M]. Springer Verlag, 2001.

[31] Cabalar P, Pearce D, Valverde A. Reducing propositional theories in equilibrium logic to logic programs [J]. Progress in Artificial Intelligence, 2005, 3808:4–17.

[32] Gebser M, Kaufmann B, Neumann A, et al. Conflict-driven answer set solving [C].// Proceedings of the 20th international joint conference on Artifical intelligence. 2007: 386–392.

[33] Leone N, Pfeifer G, Faber W, et al. The DLV system for knowledge representation and reasoning [J]. ACM Transactions on Computational Logic (TOCL), 2006, 7(3):499–562.

[34] Janhunen T, Niemelä I, Seipel D, et al. Unfolding partiality and disjunctions in stable model semantics [J]. ACM Transactions on Computational Logic (TOCL), 2006, 7(1):1–37.

[35] Niemelä I, Simons P. Smodels—an implementation of the stable model and well-founded semantics for normal logic programs [C].// Logic Programming and Nonmonotonic Reasoning. vol 1265. Springer, 1997: 420–429.