中山大学硕士学位论文

一种改进的回答集逻辑程序分割方法及程序化简研究

An improved Answer Set Program Splitting Method and Research on Program Simplification

学位申请人：XXX

指 导 教 师：XXXX XXXXX

专 业 名 称：XXXX

答辩委员会主席（签名）：

答辩委员会委员（签名）：

年　　月　　日

学位论文原创性声明

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。

学位论文作者签名：

导师签名：

日期：　　　　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刻画知识和理解逻辑，以计算机为载体实现人类智能为目的的学科。非单调逻辑是进行知识表示与推理的主要手段。本文研究的回答集编程(ASP: Answer Set Programming)则是非单调逻辑的研究热点。随着回答集编程领域多年以来的发展，其求解器已比较成熟。然而，回答集编程求解器的效率问题依旧是该领域的研究重点。

在ASP求解器提速的发展过程中，Lifschitz和Turner在1994年提出了分割集(Splitting Set)和程序分割(Program Splitting)的概念，并从理论上证明了一个ASP逻辑程序可以通过分割集被划分为底部(bottom)和顶部(top)两部分，并根据这两部分的回答集可以计算得到原程序的回答集。分割集和程序分割的提出，为ASP求解器的提速带来了新思路。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当初定义的分割集需要满足较为苛刻的条件，在实际情况中往往会出现一个ASP逻辑程序的分割集就只有空集和程序中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。

本文对Lifschitz和Turner提出的分割集和程序分割方法进行了探讨和研究，所获得的主要成果具体如下：

首先，把Lifschitz和Turner原来定义的分割集扩展为任意原子集，同时为正规逻辑程序提出了新程序分割方法，及把新程序分割方法扩展到了析取逻辑程序，和提出强程序分割方法。在分割集可以为任意原子集的情况下，程序分割方法的适用范围得到了扩展。

其次，本文分析了任意原子集构成的分割集对新程序分割的性能影响，找出主要性能瓶颈所在并给出了改进的方案。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求解一个ASP逻辑程序的回答集比直接求解更快，提速效果大致维持在2到3倍。

最后，在分析新分割集对新程序分割方法的性能影响之时，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有效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扩展到了程序化简当中，即通过程序结论(Consequence)去化简ASP逻辑程序。程序化简的目的依旧是为了让回答集程序的求解提速。

本文提出的新分割集与新程序分割给求解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思想到程序化简这样的实际应用当中，为回答集程序的求解提速带来了新思路。

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. Non-monotonic logic is the mainstream tool of knowledge representation and reasoning. In this dissertation, 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. However, the main development focus in answer set program is still 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 tough 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.

In this dissertation, I will discuss and research about Lifschitz and Turner’s splitting set and program splitting theorem. The main achievements obtained in this dissertation 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 for NLP. After this, I extend this result to DLP and propose the strong splitting method. As the splitting set can be an arbitrary set of atoms, the applicability of program splitting can be extended greatly.

Secondly, this dissertation figures out the main performance bottlenecks through analyzing properties of the new splitting method and puts forward the improved scheme. Moreover, the data coming from experiment in this dissertation 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.

Thirdly, 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 dissertation 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 purpose of program simplification is still to speed up solving answer sets of ASP program.

This dissertation 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 bring new ideas to improve solving ASP program.

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

目　录

第 1 章　引言

本章将介绍本文的研究背景和当前国内外在回答集编程领域的研究成果，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

知识表示与推理是人工智能领域中相当重要的一个研究热点[1]。经过多年来的研究和发展，知识表示这个领域已经发展得比较成熟，而其最主要的描述工具则是逻辑程序[2]。在其发展过程中，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成为了主流的工具[3]。

1951年，Horn提出了霍恩(Horn)子句，这是一种带有不超过一个正文字的析取范式[4]。霍恩子句是逻辑程序的重要构成基础。上世纪70年代末，Kowalski提出逻辑可以作为程序语言的基础的观点，并重新定义了算法的概念[5]。不久后，逻辑程序正式诞生。逻辑程序只需要设置待求解问题需要满足的规则和加入问题所有的前置事实（类似知识库）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”等步骤来描述解决过程[1]。故可以简单地理解为，逻辑程序就是：已有事实联合推理规则得到新知识[6]。基于Kowalski的逻辑程序思想，Colmerauer于1979年发明了世界上第一种用于逻辑程序设计的语言——PROLOG (PROgram in LOGic)[7]。PROLOG对知识表示和推理的发展起了十分深远的影响[8]，这种语言事实上为编程语言的发展提供了新的方向，即：不基于过程控制去演算一个问题的解，而是允许程序员从逻辑出发，刻画出一个问题是什么，并由推理系统去确定怎么解决该问题，然后问题便可以被求解[2]。由于只需描述问题“是什么”，所以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”（描述过程）到“做什么”（描述问题）的转变[9]。

然而，早期的逻辑推理系统中，以命题逻辑和一阶逻辑为基础[10]，都是通过

已知事实推出知识，故并不会因为已知事实的增加而使得之前的知识被修正或否定，只会随着事实的增加而使知识增加，因此是单调的，被称为单调逻辑[11]。但人类在正常认知的过程中，当前的认知并不一定是真理，所以一旦得到新的修正认知，之前的知识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知识增多，所以这样的逻辑称为非单调逻辑[12]。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，即：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，Clark提出了失败即否定和克拉克完备。此理论补全了逻辑程序中的否定问题的表达[13]。但那时还没有能够立刻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，使得人们可以使用非单调逻辑领域的知识解释失败即否定[14]。稳定模型语义和失败即否定的提出，和其后续的发展和扩充，回答集编程作为一种新的非单调逻辑描述工具应运而生。

1.2 国内外研究现状

在ASP的发展过程中，其语义和应用场景不断地被扩展，并且拥有了一系列高效的求解器。这些ASP求解器主要都是使用DPLL算法[15]或者SAT求解方法[16]实现的，具体分类如下[2]：

- DPLL算法：DLV[17]、Smodels[18]、clasp及其扩展claspD、clingo及其扩展iclingo；
- SAT求解方法：ASSAT[19]和Cmodels[20]。

由于这些高效ASP求解器的发明，ASP得以被广泛地应用于实际项目当中。2001年，NASA在航天决策系统中使用了ASP[21]；2006年时，Tu和Son把ASP应用在带感知行为的推理和规划问题中[22]；Confalonieri和Prade于2007年在决策问题中使用了ASP技术[23]；中科大的吉建民老师及其团队长久以来都把ASP技术运用到智能机器人上[2]。

虽然ASP求解器已经发展相对成熟，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年提出了分割集的概念，以及相应的程序分割方法，即通过分割集把原逻辑程序划分为bottom和top两部分，并证明了原程序的回答集可以通过这两部分的回答集求解得到[24]。

分割集理论自被提出以来就被认为是研究回答集语义的一个良好工具[25]。Gebser(2008)以分割集理论作为基础实现了增量式ASP求解器——iclingo[26]。此外，Oikarinen和Janhunen(2008)把分割集的思想拓展到了带嵌套表达式的逻辑程序中[27]，Ferraris(2009)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[28]。分割集的思想对ASP领域有着重要的影响。

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逻辑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。本文基于这个局限，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。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、环与环公式等。同时，详细介绍了Lifschitz和Turner的分割理论，并分析其定义的分割集的局限所在。

2. 在对Lifschitz和Turner的理论进行研究后，本文对原来的分割集理论进行扩展，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，本文为正规逻辑程序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重新定义了bottom和top这两部分。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. 接着，本文把新程序分割方法扩展到析取逻辑程序中，并提出了强程序分割方法。

5. 同时，本文对新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性能瓶颈所在，并给出相关改进方案。同时通过实验验证了使用新程序分割方法求解ASP逻辑程序的回答集的正确性和有效性。

6. 本文在分析新程序分割方法的计算复杂性时，得出了如果分割集为程序结论的话，可以有效减少耗时。由此为思路，提出了通过程序结论作为分割集以进行程序化简，把分割理论应用到程序化简当中，并通过实验得到理想的效果。

本文的研究成果已组织成学术论文，并被2015年的AAAI收录。AAAI是被中国计算机学会认证的人工智能领域A类会议。

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。在分析了Lifschitz和Turner的分割集的局限性后，提出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。第一个为对比使用新程序分割方法求解ASP逻辑程序的回答集和直接求解的效率，结果为使用新程序分割求解的效率更好；第二个为在程序化简中引入分割集的概念，使用程序结论进行程序化简，并比较求解化简后的程序跟原程序的效率，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，指出了本文的工作内容对回答集编程发展的意义。

第二章介绍了逻辑程序和回答集编程的基础知识，主要讲及失败即否定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方法的概念和设计思路，并给出了相关命题和定理的证明。同时会对程序分割过程的计算复杂性进行分析，指出其中的主要性能瓶颈，并给出改进方案。

第四章把新分割集理论应用在程序化简中，提出了新的程序化简命题，并给出相关证明和例子。

第五章根据第三章和第四章的内容设计相关对比实验，并对得到的实验数据进行了分析。两个实验的结果都给本文提出的理论提供了正面的支持。

第六章给出了对全文的总结和对分割理论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

本章先介绍命题逻辑，然后引入失败即否定，并给出ASP逻辑程序中规则的定义，及规则中各个部分涉及的公式命名和集合符号。然后给出“满足”的概念，继而引入ASP逻辑程序的回答集语义，并列出一些ASP逻辑程序的主要性质，如：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

命题是非真即假的陈述句。我们一般使用字母代表一个命题[29]。

定义 2.1 命题逻辑的符号包括[30]：

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

定义 2.2 (文字(Literal) [31]) 文字就是一个命题符号本身或者一个命题符号的否定。其中一个命题符号本身为正文字，一个命题符号的否定为负文字。

在逻辑中，否定也称为逻辑补[32]。

定义 2.3 (补(Complement) [32]) 对于一个文字l，其补记为$\bar{l}$。其中，若l为a形式的正文字，则$\bar{l}$为$\neg a$，若l为$\neg a$形式的负文字，则$\bar{l}$为a。对于一个文字集L，其补定义为：$\overline{L} = \{\bar{l} \mid l \in L\}$。

命题的真假性被称为其真值(Truth Value)，真值只有真和假两个。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[33]。通常地，在求解逻辑程序中，我们所得到的结果就是指逻辑程序中能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。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 2.4 [29] 合取范式(CNF: Conjunctive Normal Form)：一系列析取式的合取形式；析取范式(DNF: Disjunctive Normal Form)：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$\vee$($\wedge$)进行连接。

2.2 回答集编程

本节介绍回答集编程的基础知识和本文后续将会用到的相关概念。

2.2.1 回答集编程基础

经典逻辑程序只能推出正文字的真值，而实际情况下，负文字的真值也需要被推出。Reiter提出的封闭世界假定(CWA: Close World Assumption)[34]和Clark的失败即否定(NAF: Negation as Failure)为逻辑程序可以推出负文字形式的结论带来了支持。ASP逻辑程序的形式主要便是在经典命题公式中引入失败即否定，其中出现的not即表示失败即否定。

失败即否定的基本思想就是：无法证明一个命题为真，则判定其为假。这也是最为自然的一种逻辑假定。在ASP逻辑程序中一些规则的体部会出现“$not\ A$”这样的文字，而对于这样的文字，可以直观地把它看作是命题：“不能确定A为真”，所以如果ASP逻辑程序中不能推出A为真，则可以推出$not\ A$为真。实际上，失败即否定描述的是一个文字在逻辑程序中的一致性[13]。封闭世界假定也是把当前无法确定的事件假定为假。在失败即否定中，这种推理过程其实是一种倾向于否定的假定逻辑。即使当前推出$not\ A$为真，那是基于暂时无法推出A为真。一旦加入可以推出A为真的新规则，那么原来的$not\ A$则被认为是假[35]。

一个ASP逻辑程序一般分为两部分：事实集和规则集。使用事实集对规则集进行例化后得到ASP逻辑程序。

ASP逻辑程序的求解过程分为以下两步[36]：

1. 先使用例化工具通过事实集例化规则集；

2. 对例化后得到的ASP逻辑程序，调用求解器进行求解。

从实际效果来看，例化后的ASP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规则数量有限的ASP逻辑程序。本文所探讨的ASP逻辑程序中的规则(rule)形式如下：

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

其中称a_i为原子(atom)。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[19]：

- $k = 0$，$m > 0$时，规则被称为约束(Constraint)；
- $k = 1$时，规则被称为正规规则(Normal Rule)；
- $k > 0$，$n = m = 0$时，规则被称为事实(Fact)；
- $n = 0$时，即规则中没有带not的原子时，规则称其为正规则(Positive Rule)。

例 2.1 下面为四种形式的规则：

$$\leftarrow \ p, \ q. \tag{2.2}$$

$$r \ \leftarrow \ s, \ not\ t. \tag{2.3}$$

$$s. \tag{2.4}$$

$$p \ \leftarrow \ r, \ t. \tag{2.5}$$

$$s \ \vee \ p \ \leftarrow \ r, \ t, \ not\ p. \tag{2.6}$$

其中的(2.2)为限制，(2.3)为正规规则，(2.4)为事实，(2.5)为正规则，(2.6)为一般规则。

定义 2.5 [37] 正规逻辑程序(NLP: Normal Logic Program)是由有限条正规规则组成的逻辑程序，其中可以包含有限个事实及约束；析取逻辑程序(DLP: Disjunctive Logic Program)是由有限条形如形式(2.1)的规则组成的逻辑程序，其中可以包含有限个事实及约束。

对于形式(2.1)中的规则，还可以将其等价于以下形式：

$$head(r) \leftarrow body(r). \tag{2.7}$$

其中r代表形式(2.1)中的整个规则，$head(r)$称为规则r的头部，$body(r)$称为规则r的体部。具体有$head(r)$为$a_1 \vee a_2 \vee ... \vee a_k$，$body(r)$为$a_{k+1}, ..., a_m, not\ a_{m+1}, ..., not\ a_n$。其中规则的头部中的原子连接关系只为析取，体部中的原子连接关系只为合取。更进一步的划分有$body(r) = body^+(r) \wedge body^-(r)$，其中$body^+(r)$为$a_{k+1}, ..., a_m$，即原规则中不带$not$的文字集合，$body^-(r)$为$not\ a_{m+1}, ..., not\ a_n$，即原规则中带有$not$的文字集合。不失一般性地，上述定义的命题公式可以被用作原子集(a set of atoms)进行讨论，并引入以下两个常用原子集：

定义 2.6 [38] 给定r为符合形式(2.1)的一个规则，P为一个ASP逻辑程序，则有：

- $Atoms(r) = head(r) \cup body^+(r) \cup body^-(r)$；
- $Atoms(P) = \bigcup_{r \in P} Atoms(r)$。

例 2.2 把例子2.1中的规则集合看作是一个ASP逻辑程序P，对于式(2.6)中的规则r则有：

$$head(r) = \{s, q\} \tag{2.8}$$

$$body(r) = \{r, t, p\} \tag{2.9}$$

$$body^+(r) = \{r, t\} \tag{2.10}$$

$$body^-(r) = \{p\} \tag{2.11}$$

$$Atoms(r) = \{s, q, r, t, p\} \tag{2.12}$$

同时，$Atoms(P) = \{s, q, r, t, p\}$。

接着，本节将通过满足(satisfy)性来定义ASP逻辑程序的模型。

定义 2.7 [19] 给定一个原子集S，一个正合取范式C和一个正析取范式D，S满足C，当且仅当$Atoms(R) \subseteq S$；S满足D，当且仅当$Atoms(R) \cap S \neq \emptyset$。记满足符号为$\models$。

相应地，一个原子集S满足一个规则r的$body(r)$当且仅当$body^+(r) \subseteq S$且$body^-(r) \cap S = \emptyset$，记为$S \models body(r)$。一个原子集$S$满足一个规则$r$的$head(r)$当且仅当$head(r) \cap S \neq \emptyset$，记为$S \models head(r)$[39]。

例 2.3 给定一个原子集$S = \{a, b, c\}$，规则$r1$为$a \lor b \leftarrow c, not\ d.$，规则$r2$为$d \lor e \leftarrow c, not\ b.$，则有：$S \models head(r1)$，$S \models body(r1)$，$S \not\models head(r2)$，$S \not\models body(r2)$。

接下来，本节给出如何判断一个原子集是否满足一个规则，以及一个原子集是否满足一个ASP逻辑程序。

定义 2.8 [14] 一个原子集S满足一个规则r，当且仅当$S \models body(r)$蕴涵$S \models head(r)$，并记为$S \models r$。一个原子集S满足一个ASP逻辑程序P，当且仅当S满足P中的所有规则。

基于原子集对ASP逻辑程序的满足性定义，本节给出ASP逻辑程序的模型定义如下：

定义 2.9 [14] 给定原子集S和ASP逻辑程序P，若S满足P，我们将S称为P的一个模型(model)。若一个原子集I是P的模型，且不存在另一个原子集J符合$J \subseteq I$且J是P的模型，则称I是P的极小模型(minimal model)。

2.2.2 回答集编程的语义

Gelfond和Lifschitz(1988)提出了稳定模型语义，稳定模型语义解决了非单调推理无法解释失败即否定的问题，此外他们还给出了一个规约方法(G-L规约)，以化简一个ASP逻辑程序中的失败即否定。

定义 2.10 (G-L规约[14]) 给定原子集S和不含约束的ASP逻辑程序P，P基于S的G-L规约结果记为P^S。P通过以下两个化简规则得到P^S：

- 若一个规则的体部中有$not\ p$，且$p \in S$，则删掉该规则；
- 对剩下的所有规则，删除体部中的$not\ p$，p为$Atoms(P)$中任意一个原子。

例 2.4 已知原子集$S=\{a,b,c\}$，且ASP逻辑程序P如下：

$$a \leftarrow b,\ c. \tag{2.13}$$

$$e \leftarrow b,\ not\ a. \tag{2.14}$$

$$f \leftarrow not\ e. \tag{2.15}$$

根据G-L规约的规则，P中第二条规则的负文字中包含S里的原子，所以直接删掉；P中第三条规则的负文字中包含S以外的原子，所以只把负文字删掉。

所以P^S为：

$$a \leftarrow b,\ c. \tag{2.16}$$

$$f. \tag{2.17}$$

显然，通过G-L规约进行化简后得到的P^S是一个不包含任何失败即否定的ASP逻辑程序。这样的ASP逻辑程序只有一个唯一的极小模型，这个模型被称为稳定模型，并记为$Cons(P^S)$[14]。

对于包含有约束的ASP逻辑程序，我们可以通过G-L规约来定义其回答集，通过对程序中的约束和一般规则分离判断即可。

定理 2.1 [19] 给定一个包含约束的正规逻辑程序P和原子集S，S是P的一个回答集(Answer Set)当且仅当$S=Cons(PD^S)$，且S满足P中的所有约束。其中PD是P去掉所有约束后所得到的ASP逻辑程序。

本文记一个ASP逻辑程序P的回答集为$\Gamma(P)$。

Gelfond和Lifschitz(1991)补充了析取逻辑程序的回答集定义[40]。关于析取逻辑程序的回答集，依旧可以通过G-L规约得到。然而不同于正规逻辑程序，该逻辑程序将有一系列极小模型，这里记为$\Psi(P^S)$。

定理 2.2 [40] 给定析取逻辑程序P和原子集S，S是P的一个回答集，当前仅当$S\in\Psi(P^S)$。

定义 2.11 (程序结论(Consequence) [41]) 给定ASP逻辑程序P和文字集L，如果L能被P的每个回答集所满足，则称L为P的一个程序结论。

ASP逻辑程序的程序结论并不需要求解出所有回答集后才能得到。Chen和Ji等(2013)给出了一个计算程序结论的算法[41]。在引入程序结论的计算方法前，先介绍其需要用到的算子和概念。

定义 2.12 (子句(Clause)) [42] 在逻辑中，子句即若干个文字的析取。若一个子句中只包含一个文字，则称为单位子句(Unit Clause)。

把ASP逻辑程序中的规则r转换成子句就是通过蕴涵式等价为析取式的转换：$p \rightarrow q \equiv \neg p \vee q$。

定义 2.13 (单位传播(Unit Propagation)) [43] 给定一个子句集C，令其中的单位子句集合为A，单位传播即对$C \backslash A$中的子句执行以下两步：

- 删去包含A中的文字的子句；
- 删去子句中出现的$\neg l$，其中$l \in A$。

Chen和Ji等(2013)给出了一个计算单位传播的算法[41]，即$Algorithm$ 1。

其中的$Lit(P) = Atoms(P) \cup \{\neg a \mid a \in Atoms(P)\}$[41]，$unit_clause(\Gamma)$为子句集$\Gamma$中的单元子句集合，$assign(A, \Gamma) = \{c \mid c' \in \Gamma,\ c' \cap A = \emptyset,\ 且\ c = c' \backslash \overline{A}\}$[41]。

Gelder和Ross等(1991)提出了ASP逻辑程序的最大无理集(Greatest Unfounded Set)[44]。

定义 2.14 (无理集(Unfounded Set)[44]) 给定ASP逻辑程序P和文字集I。称原子集A为P关于I的无理集，若对P中满足$head(r) \in A$的规则r，以下其中一个条件成立：

- 存在原子$q \in body(r)$在I中为假；
- 存在原子$q \in body(r)$属于A。

Algorithm 1: 计算单位传播的算法$UP(P)$

输入: 一个逻辑程序P

输出: 文字集lit

$\Gamma\ :=\ P$对应的子句;
if $\emptyset \in \Gamma$ **then**
    return $Lit(P)$;
end
$A\ :=\ unit\_clause(\Gamma)$;
if A是不一致的 **then**
    return $Lit(P)$;
end
if $A \neq \emptyset$ **then**
    return $A \cup UP(assign(A, \Gamma))$;
end
else
    return $\emptyset$;
end

Gelder和Ross等(1991)证明了两个无理集的并集仍然是一个无理集[44]，因此ASP逻辑程序P基于文字集I存在一个最大无理集(Greatest Unfounded Set)，记为$GUS(P, L)$。定义以下算子以计算$GUS(P, L)$：

定义 2.15 [41] 给ASP逻辑程序P，文字集L和原子集X，定义算子：

$$\begin{aligned}\varPhi_L(X)\ =\ &\{a \mid \exists r \in P,\ a \in head(r)\ \text{且}\ a \notin \{p \mid \neg p \in L\},\\ &body(r) \cap (\{p \mid \neg p \in L\} \cup \{\neg p \mid p \in L\}) = \emptyset,\\ &body^+(r) \subseteq (X \backslash \{p \mid \neg p \in L\}),\ head(r) \cap L = \emptyset\}\end{aligned} \tag{2.18}$$

则有P的最大无理集为[41]：

$$GUS(P, L) = Atoms(P) \setminus lfp(\varPhi_L(X) \cup \{p \mid p \in L\}) \tag{2.19}$$

其中$lfp(\phi)$为计算极小不动点。

记ASP逻辑程序P的程序结论为$conq(P)$，其计算公式为[41]：

$$conq(P)\ =\ lfp(UP(L, P) \cup \neg GUS(P, L)) \tag{2.20}$$

其中$UP(L,P) = L\cup UP(assign(L,P))$。接下来给出一个计算程序结论的例子。

例 2.5 给定逻辑程序P：

$$a \leftarrow c. \tag{2.21}$$

$$c \leftarrow e. \tag{2.22}$$

$$e \leftarrow not\ f. \tag{2.23}$$

$$f \leftarrow not\ e. \tag{2.24}$$

$$\leftarrow not\ c. \tag{2.25}$$

其对应的子句集Γ为：

$$\neg c \vee a. \tag{2.26}$$

$$\neg e \vee c. \tag{2.27}$$

$$e \vee f. \tag{2.28}$$

$$c. \tag{2.29}$$

计算$W_P(L)$的极小不动点，故令$L=\emptyset$开始，有：

$$UP(\{\},P) = \{\}\cup UP(assign(\{\},P)) = \{a,c\} \tag{2.30}$$

$$GUS(P,\{\}) = Atoms(P)\backslash lfp(\Phi_{\{\}}(\{\})) = \emptyset \tag{2.31}$$

$$W_P(\{\}) = UP(\{\},P)\cup\neg GUS(P,\{\}) = \{a,c\}\cup\{\} = \{a,c\} \tag{2.32}$$

继续计算，令$L = \{a,c\}$，有：

$$UP(\{a,c\},P) = \{a,c\}\cup UP(assign(\{a,c\},P)) = \{a,c\} \tag{2.33}$$

$$GUS(P,\{a,c\}) = Atoms(P)\backslash lfp(\Phi_{\{a,c\}}(\{\})) = \emptyset \tag{2.34}$$

$$\begin{aligned} W_P(\{a,c\}) &= UP(\{a,c\},P)\cup\neg GUS(P,\{a,c\}) \\ &= \{a,c\}\cup\{\} = \{a,c\} \end{aligned} \tag{2.35}$$

故计算得到极小不动点为$\{a,c\}$，所以原程序P的一个程序结论为$\{a,c\}$。

接下来，本节将说明ASP逻辑程序中的一个重要性质，即ASP逻辑程序中的环以及对应的环公式。

2.2.3 回答集逻辑程序中的环与环公式

Lin和Zhao(2002)提出了正规逻辑程序中的环的概念，同时定义了其环公式[19]。紧接着，Lee和Lifschitz(2003)给出了析取逻辑程序中的环和环公式的定义[45]。

在引入具体的环及环公式定义前，需要先给出一个ASP逻辑程序的正依赖图及强连通分量的定义。

定义 2.16 (正依赖图(Positive Dependency Graph)[19]) 给定ASP逻辑程序P，以P中的原子作为顶点，规则作为构成边的依据，可以构造出一个有向连通图，称其为P的正依赖图，记为G_P。其中，当存在P中的一个规则有$p \in head(r)$且$q \in body^+(r)$，则正依赖图中存在一条从原子p指向原子q的有向边。

定义 2.17 (强连通分量(SCC: Strongly Conntected Component)[46]) 一个有向图G中的一个强联通分量SCC满足对于任意两个节点s_1，$s_2 \in SCC$均存在一条路径从s_1到达s_2，且该路径中的所有节点均属于该SCC。

例 2.6 给定ASP逻辑程序P如下：

$$a \leftarrow not\ d. \tag{2.36}$$

$$d \leftarrow not\ c. \tag{2.37}$$

$$a \leftarrow c,\ d. \tag{2.38}$$

$$c \leftarrow a. \tag{2.39}$$

根据定义，关于P的正依赖图如图2.1所示，原子集$\{a, c\}$是P中的一个强连通分量。

接着，本节引入基于正依赖图定义的环。

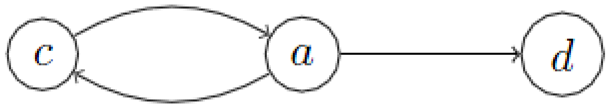

图 2.1: P的正依赖图

定义 2.18 (环(Loop) [19]) 给定一个ASP逻辑程序P，对于$Atoms(P)$的任意非空子集L，如果对于L中的任意两个原子p、q，G_P中都有至少一条长度大于0的路径使得p可达q，则称L是P的一个环。此外，任意单个原子也是一个环。

ASP逻辑程序的环实质上就是其对应的正依赖图中的强连通分量。环还有一个重要性质用于定义环公式，那就是环的外部支持规则(External Support Rules)。环L在ASP逻辑程序P中的外部支持规则记为$R^-(L,P)$，定义如下：

$$R^-(L,P) = \{r \in P \mid head(r) \cap L \neq \emptyset,\ body^+(r) \cap L = \emptyset\} \tag{2.40}$$

还可以更进一步地定义环L在P中基于原子集X的外部支持规则，记为$R^-(L,P,X)$，定义如下：

$$R^-(L,P,X) = \{r \in R^-(L,P) \mid X \models body(r) \wedge \bigwedge_{q \in head(r) L} \neg q\} \tag{2.41}$$

然后，本节引入环公式的定义。

定义 2.19 (环公式(Loop Formula) [47]) 给定ASP逻辑程序P，对于其中的一个环L所对应的环公式记为$LF(L,P)$，定义如下：

$$\bigwedge_{p \in L} p \supset \bigvee_{r \in R^-(L,P)} (body(r) \wedge \bigwedge_{q \in head(r) L} \neg q) \tag{2.42}$$

对于一个原子集S和环L，若有$L \subseteq S$推出存在规则$r \in R^-(L,P)$满足$S \models body(r)$及$(head(r) \backslash L) \cap S = \emptyset$，则称$S$满足$L$，记为$S \models L$。

命题 2.1 [45] 给定ASP逻辑程序P和原子集合S。如果S满足P，则以下说法是等价的：

- S是P的一个回答集；

- S满足P中所有环L的环公式$LF(L,P)$；
- S满足$Atoms(P)$所有非空子集E的环公式$LF(E,P)$。

使用SAT求解器得到的模型不完全是ASP逻辑程序的回答集，其根本原因就在于环的存在。如果ASP逻辑程序中存在环，即存在环内所有原子相互推导的行为，即环内的原子相互推导对方为真，这样就会产生大量模型。然而，从回答集语义出发，这其中如果没有事实或者失败即否定的支持，则推导不成立，故这些模型不是回答集。而环公式就是提取正体部原子全在环外的规则以支持环内原子的成立，这也是外部支持规则的名字由来。基于这样的事实，Lin和Zhao(2004)提出了使用环公式求解ASP逻辑程序回答集的方法，并依此实现了相关的ASP求解器——ASSAT[19]。

Lin和Zhao(2004)扩展了完备的定义，并给出如下定义：

定义 2.20 (完备(Completion) [19]**)** 给定ASP逻辑程序P，其完备记为$Comp(P)$。$Comp(P)$是以下规则的集合：

- 对于任意$p \in Atoms(P)$，P中所有以p作为头部的规则形如$p \leftarrow G_k$，则$p \equiv G_1 \vee G_2 \vee ... \vee G_n$为$Comp(P)$中的元素，特别地，如果一个原子$p$没有作为头部出现过，那么把$\neg p$加入到$Comp(P)$中；
- 对于P中的所有限制，形如$\leftarrow G_k$，把$\neg G_k$加入到$Comp(P)$中。

其中第一部分为克拉克完备(Clark Completion)[13]。一个ASP逻辑程序的完备具体计算如下例子所示：

例 2.7 [19] 给定ASP逻辑程序P：

$$a \leftarrow b,\ c,\ not\ d. \tag{2.43}$$

$$a \leftarrow b,\ not\ c,\ not\ d. \tag{2.44}$$

$$\leftarrow b,\ c,\ not\ d. \tag{2.45}$$

则其完备$Comp(P)$为：

$$\{a \equiv (b \wedge c \wedge \neg d) \vee (b \wedge \neg c \wedge \neg d),\ \neg b,\ \neg c,\ \neg d,\ \neg(b \wedge c \wedge \neg d)\} \tag{2.46}$$

Lin和Zhao(2004)基于环公式和完备的概念提出了一个新的回答集求解理论。

定理 2.3 [19] 给定ASP逻辑程序P，LF表示P中所有环的环公式集合。一个原子集S是P的回答集，当且仅当它是$Comp(P) \cup LF$的一个模型。

Lin和Zhao的环理论自提出以来就被不断地引用。Gebser等(2010)提出了基本环(Elementary Loop)[48]，其是一系列可以代表其他环的环，Gebser等还将基本环应用到clasp求解器的实现中。Ji和Wan等(2014)在Gebser等提出的基本环的基础上更进一步地在正规逻辑程序中提出了更具有代表性的特征环(Proper Loop)[49]，并在不久后将特征环的定义扩展到析取逻辑程序[50]。

2.3 分割集与程序分割

Lifschitz和Turner(1994)提出了分割集(Splitting Set)的概念[24]。其本意是为了把逻辑程序划分成若干个较小规模的程序，然后通过这些小规模的程序的回答集去求解原程序的回答集。由于逻辑程序的规模对求解效率有很大的影响，通过把逻辑程序的规模降低，把指数性的关系降为加性关系，将有助于提高逻辑程序的求解效率。然而，要达到分割程序后得到的回答集是原程序的回答集，需要满足一定的条件[2]。

Lifschitz和Turner在给出分割集的概念后，也基于分割集提出了相应的程序分割(Program Splitting)方法。程序分割方法具体就是根据定义找出分割集，通过分割集把原程序分割为bottom和top两部分，并且证明原程序的回答集可以通过bottom和top这两部分的回答集所求得。

定义 2.21 (分割集(Splitting Set)[24]) 给定一个ASP逻辑程序P，其分割集是一个原子集，记为U，U需要满足：对于任意的规则$r \in P$有$head(r) \cap U \neq \emptyset$蕴涵$Atoms(r) \subseteq U$。

从ASP逻辑程序P的正依赖图拓扑结构出发，分割集的原子存在一个直观的性质。

命题 2.2 [2] 分割集的原子构成ASP逻辑程序P的正依赖图中出度为零的一个子图。

基于分割集划分而成的bottom和top两部分的具体定义如下：

定义 2.22 (底部和顶部(bottom and top) [24]**)** 给定一个ASP逻辑程序P，及其分割集U，标记P的底部为$b_U(P)$，顶部为$t_U(P)$，并定义为：

$$b_U(P) = \{r \in P \mid head(r) \cap U \neq \emptyset\} \tag{2.47}$$

$$t_U(P) = P \setminus b_U(P) \tag{2.48}$$

显然，ASP逻辑程序的bottom是把分割集U中的原子作为头部的规则都抽取出来，而top则是头部与分割集U无关的规则集合，且$\emptyset$和$Atoms(P)$是任意一个ASP逻辑程序P的分割集。

为了能通过bottom和top求解原程序的回答集，需要引入化简操作。Lifschitz和Turner定义了操作$e_U(P, X)$来联合bottom的回答集和top求出另一部分的回答集。

定义 2.23 给定ASP逻辑程序P，和原子集X、U，定义$e_U(P, X)$如下：

- 删除符合以下条件的规则r：$head(r) \cap X \neq \emptyset$且$body^+(r) \cap U \not\subseteq X$，或者$(body^-(r) \cap U) \cap X \neq \emptyset$；
- 在剩下的规则的体部中把所有形如a或$not\ a$的文字删掉，其中$a \in U$。

定义 2.24 给定ASP逻辑程序P及其分割集U，P关于U的一个方案(Solution)，是一个原子集组合$\langle X, Y\rangle$，具体地：

- X是$b_U(P)$的回答集；
- Y是$e_U(P \backslash b_U(P), X)$的回答集。

例 2.8 给定ASP逻辑程序P：

$$a \leftarrow not\ d. \tag{2.49}$$

$$d \leftarrow not\ c. \tag{2.50}$$

$$a \leftarrow c,\ d. \tag{2.51}$$

$$c \leftarrow . \tag{2.52}$$

根据定义可知$U = \{c, d\}$是P的一个分割集，并且可以计算得到：

$$b_U(P) = \{d \leftarrow not\ c.\ c \leftarrow .\} \tag{2.53}$$

$$P \backslash b_U(P) = \{a \leftarrow not\ d.\ a \leftarrow c,\ d.\} \tag{2.54}$$

且有$\{c\} \in \Gamma(b_U(P))$，故令$X = \{c\}$，有：

$$e_U(P \backslash b_U(P), X) = \{a \leftarrow .\} \tag{2.55}$$

其回答集为$\{a\}$。$\langle\{c\}, \{a\}\rangle$则是$P$关于$U$的一个方案。

关于原程序的回答集，可以通过上述定义的逻辑程序关于分割集的方案得到，Lifschitz和Turner给出的分割集理论就是指出如何从方案得到回答集。

定理 2.4 (分割集理论(Splitting Theorem)[24]) 已知ASP逻辑程序P和其分割集U，则一个原子集S是P的回答集，当且仅当$S = X \cup Y$，其中$\langle X, Y\rangle$是P关于U的一个方案。

2.4 本章小结

本章介绍了命题逻辑和回答集编程的基础知识，并对Lifschitz和Turner所提出的分割集理论进行了说明，接下来，本文提出新分割集和新程序分割方法，以及通过具体的应用场景来体现新程序分割方法的意义。

第 3 章　改进的程序分割方法

本章首先对Lifschitz和Turner提出的分割集和程序分割理论进行分析，给出计算其定义的分割集的算法，然后分析了其分割集的局限性。接着，本章提出了对分割集的扩展和新程序分割方法。本章将分别给出正规逻辑程序和析取逻辑程序的新程序分割方法，然后提出一个强程序分割方法。最后以正规逻辑程序的程序分割方法为例，对新程序分割方法的计算复杂性进行分析，并指出主要性能瓶颈所在及改进思路。

3.1　分割集的计算与扩展

本节将对Lifschitz和Turner提出的分割集理论进行分析，然后基于他们对分割集的定义提出一个计算分割集的算法，并分析其分割集在一般ASP逻辑程序中的局限性。随后给出本文定义的新分割集。

3.1.1　分割集的计算

Lifschitz和Turner给出的分割集定义的直观含义是若分割集包含一个规则头部的某些原子，则包含该规则中的所有原子。这样的性质保证了分割集U可以把原程序从结构上划分成独立的两部分。而Lifschitz和Turner给出的分割集的定义实际上是一个验证型定义。本节根据Lifschitz和Turner对分割集的定义和Ji(2010)指出的分割集性质2.2给出了一个计算ASP逻辑程序的分割集的算法，为$Alogrithm\ 2$。

根据$Alogrithm\ 2$，给出以下命题：

命题 3.1　*给定ASP逻辑程序P，记函数$Us(P)$的返回结果为Us，则有以下结论：*

- *Us中的元素都是P的分割集；*
- *P中的任意分割集U，满足$U \in Us$。*

Algorithm 2: 计算所有分割集的算法$Us(P)$

输入: 一个ASP逻辑程序P

输出: 所有分割集Us

$Us\ :=\ \emptyset$
for $rule \in P$ **do**
　$U\ :=\ Atoms(rule)$
　$P'\ :=\ P\backslash\{rule\}$
　for $r \in P'$ **do**
　　if $head(r) \cap U \neq \emptyset$ **then**
　　　$U\ :=\ U \cup Atoms(r)$
　　end
　end
　$Us\ :=\ Us \cup \{U\}$
end
return Us

证明： 对于命题3.1中的结论一，给定任意$U \in Us$，根据$Alogrithm$ 2，用于初始化U的规则r_1，有$Atoms(r_1) \subseteq U$，故符合$head(r_1) \cap U \neq \emptyset$蕴涵$Atoms(r_1) \subseteq U$。对于规则$r_2 \in P'$，其中$P' = P \setminus \{r_1\}$，也符合若$head(r_2) \cap U \neq \emptyset$，则$Atoms(r_2) \subseteq U$。则对于规则$r \in P$均有$head(r) \cap U \neq \emptyset$蕴涵$Atoms(r) \subseteq U$。故任意$U \in Us$都是$P$的分割集。

对于命题3.1中的结论二，令U为P的一个分割集，故必存在规则$r_U \in P$，有$Atoms(r_U) \subseteq U$。而函数$Us(P)$会遍历P中的所有规则，所以必存在一个原子集$U' \in Us$满足$Atoms(r_U) \subseteq U'$。而且，对于任意规则$r' \in P\backslash\{r_U\}$满足：若$head(r')\ capU \neq \emptyset$，则$Atoms(r') \subseteq U'$。故有$U = U'$，所以必有$U \in Us$。 ■

下面给出一个调用函数$Us(P)$的例子：

例 3.1 给定ASP逻辑程序P：

$$a \vee b\ \leftarrow\ c,\ d. \tag{3.1}$$

$$a\ \leftarrow\ e,\ b. \tag{3.2}$$

$$e\ \leftarrow\ f. \tag{3.3}$$

$$g \leftarrow h, i. \tag{3.4}$$

$$g \vee h \leftarrow j, k. \tag{3.5}$$

$$k \leftarrow i. \tag{3.6}$$

通过$Alogrithm$ 2可以得到：

$$Us(P) = \{\{a, b, c, d, e, f\}, \{g, h, i, j, k\}\} \tag{3.7}$$

根据定义2.21可以判断得到$Us(P)$中的元素均为P的分割集，且$Us(P)$是P中所有分割集构成的集合。

3.1.2 分割集的扩展

明显地，$\emptyset$和$Atoms(P)$都是任何逻辑程序的分割集。而对于国际ASP竞赛[1]中的大部分逻辑程序，其分割集都是$Atoms(P)$。这样的事实表明，Lifschitz和Turner所定义的分割集于实际的ASP逻辑程序是存在局限性的。分割集是从原程序中抽取出来结构独立的一个子图，根据命题2.2，这样的子图对应的规则集合没有外部支持，所以在求解回答集上也是独立的。

然而从ASP竞赛的逻辑程序的分割集情况可以知道，实际中的程序并非都具有如此性能优良的子图。为了能让分割集的思想能应用到一般的情景下，本文对其Lifschitz和Turner的分割集进行了扩展。事实上，本文并没有定义一个新的分割集，而是定义了一个新程序分割方法，并确保这个新的程序分割方法能对任意原子集构成的分割集都有效。即新的分割集是任意原子集，摆脱了原有分割集对ASP逻辑程序的结构的依赖。

3.2 新程序分割方法

本节中将分别给出正规逻辑程序和析取逻辑程序的新程序分割方法，这两者间的主要不同在于对top的定义，这是由正规逻辑程序和析取逻辑程序的回答集语义所决定。在介绍正规逻辑程序的新程序分割方法过程中，本节会在定义操作符

[1]https://www.mat.unical.it/aspcomp2013

的同时给出其直观上的含义，并通过相关的命题证明新程序分割方法能使用任意原子集构成的分割集对ASP逻辑程序进行分割，并可根据分割后两部分的回答集求解得到原程序的回答集。最后，本节会提出一个强程序分割方法，该方法是为了扩展把程序分割方法在结合律下的有效性。

3.2.1 正规逻辑的新程序分割方法

首先，本节将继续使用定义2.22中的$b_U(P)$和定义2.23中的操作$e_U(P, X)$，记ASP逻辑程序的分割集为U。而此时分割集U是任意的原子集，并不存在结构上的独立性，即：

$$Atoms(b_U(P)) \subseteq U \tag{3.8}$$

式(3.8)不一定成立。可以更为直接地说，大部分情况下都不再成立。为了保证新程序分割方法的普遍适用性，我们考虑以下关系是成立的：

$$Atoms(b_U(P)) \nsubseteq U \tag{3.9}$$

事实上，对于$Atoms(b_U(P)) \nsubseteq U$情况下有效的程序分割方法在$Atoms(b_U(P)) \subseteq U$情况下一样有效，因为附加操作都是为了保证最坏情况的。

对于Lifschitz和Turner定义的分割集，G_P关于$Atoms(b_U(P))$的子图保证了自身在结构上的独立性，即有：

$$Atoms(b_U(P)) \ \cap \ head(P \setminus b_U(P)) = \emptyset \tag{3.10}$$

即若一个原子$a \in Atoms(b_U(P))$且$a \in \Gamma(P)$，那么a只会在$b_U(P)$中被推出。而对于U为任意原子集的情况下，有：

$$Atoms(b_U(P)) \setminus U \ \neq \ \emptyset \tag{3.11}$$

所以需要考虑$Atoms(b_U(P)) \setminus U$这部分原子的真值可能性问题。显然，这部分原子的真值仅靠$b_U(P)$是无法确定的，因为它们可能是$P \setminus b_U(P)$中某个规则的头部，即可能会在$P \setminus b_U(P)$部分被推出为真。所以最直接的方法就是

在$\Gamma(b_U(P))$中加入这部分原子的真值可能性的全排列组合。然而，这部分原子的真值也并非可以任意猜测，需要配合$b_U(P)$中的规则，所以本节定义了以下规则集合，来确保这些原子的真值在合理推导下被遍历。

定义 3.1 规则集合$EC_U(P)$为$Atoms(b_U(P)) \setminus U$中的原子的真值可能性提供补充规则，定义如下：

$$EC_U(P) \ = \ \{p \ \leftarrow \ not\, p'.\, p' \ \leftarrow \ not\, p. \mid p \in Atoms(b_U(P)) \backslash U\} \tag{3.12}$$

$EC_U(P)$为$Atoms(b_U(P)) \backslash U$中的原子引入一组规则：

$$p \ \leftarrow \ not\, p'.\, p' \ \leftarrow \ not\, p. \tag{3.13}$$

其中的p'其实代表的就是$\neg p$，上面一组规则在ASP逻辑程序中的含义是$p \vee not\, p.$，换句话说即，如果一个ASP逻辑程序只有这两个规则，那么它的回答集就是$\{\{p\}, \{p'\}\}$，实际就是$\neg p$或p。接着，通过一个简单的例子来展现$EC_U(P)$的效果。

例 3.2 给定ASP逻辑程序P_1：

$$a \ \leftarrow \ not\, d. \tag{3.14}$$

$$d \ \leftarrow \ not\, c. \tag{3.15}$$

$$c \ \leftarrow \ a. \tag{3.16}$$

$$a \ \leftarrow \ c,\ d. \tag{3.17}$$

令P_1的分割集$U = \{a\}$，则有：

$$b_U(P_1) \ = \ \{a \ \leftarrow \ not\, d.\, a \ \leftarrow \ c,\ d.\} \tag{3.18}$$

并求解得到$\Gamma(b_U(P_1)) = \{\{a\}\}$。而$Atoms(b_U(P_1)) \backslash U = \{c, d\}$，所以根据定义有：

$$EC_U(P_1) \ = \ \{d \ \leftarrow \ not\, d'.\, d' \ \leftarrow \ not\, d.\, c \ \leftarrow \ not\, c'.\, c' \ \leftarrow \ not\, c.\} \tag{3.19}$$

求解$b_U(P_1) \cup EC_U(P_1)$，得到：

$$\Gamma(b_U(P_1) \cup EC_U(P_1)) = \{\{a, c, d\}, \{a, c, d'\}, \{a, c', d'\}, \{c', d\}\} \tag{3.20}$$

让$b_U(P)$并上$EC_U(P)$能使得$Atoms(b_U(P)) \setminus U$中原子的真值的有效可能性被加入到$b_U(P)$的回答集中。如$\Gamma(b_U(P) \cup EC_U(P))$中只有$\{c', d\}$而没有$\{a, c', d\}$，这是因为在$\{c', d\}$为真的情况下，原程序无法推出$a$为真。这就是本节定义$EC_U(P)$而没有直接往$b_U(P)$的回答集中遍历插入剩下原子的可能真值组合的原因。

定义新的$bottom$为$b_U(P) \cup EC_U(P)$，记$bottom$的一个回答集为X，$X \in \Gamma(b_U(P) \cup EC_U(P))$，那么这时就有$X \setminus U \neq \emptyset$。

定义2.23中的操作$e_U(P, X)$只会去掉有形如a或$not\ a$的文字，其中$a \in U$。而实际上，在$Atoms(b_U(P)) \setminus U \neq \emptyset$且$X \setminus U \neq \emptyset$的情况下，需要在$P \setminus b_U(P)$中保证$Atoms(b_U(P)) \setminus U$中的原子的真值确定性，所以本节定义了一个新的规则集合$ECC_U(P, X)$以达到这个目的。

定义 3.2 规则集合$ECC_U(P, X)$为$P \setminus b_U(P)$部分提供$Atoms(b_U(P)) \setminus U$中原子的真值确定性，具体有：

$$\begin{aligned} ECC_U(P) = \{\leftarrow\ not\ p. \mid p \in Atoms(b_U(P)) \setminus U \text{ 且 } p \in X\} \\ \cup \{\leftarrow\ p. \mid p \in Atoms(b_U(P)) \setminus U \text{ 且 } p \notin X\} \end{aligned} \tag{3.21}$$

对于$ECC_U(P, X)$直观上的功能就是保证在$P \setminus b_U(P)$中的每个原子p，若$p \in X$，则要确保p为真，故往ASP逻辑程序中加入“$\leftarrow\ not\ p.$”；若$p \notin X$，则要确保p为假，故往ASP逻辑程序中加入“$\leftarrow\ p.$”。基于这样的分析，可以得到以下命题。

命题 3.2 已知P是一个正规逻辑程序，U是一个原子集。那么一个原子集$S \subseteq Atoms(P)$能满足P，当且仅当$S = (X \cup Y) \cap Atoms(P)$，其中的原子集$X$和$Y$分别有：

- $X \models b_U(P) \cup EC_U(P)$

- $Y \models e_U(P \backslash b_U(P), X) \cup ECC_U(P, X)$

证明： ($\Rightarrow$) 若有$X \models b_U(P) \cup EC_U(P)$及$Y \models e_U(P \backslash b_U(P), X) \cup ECC_U(P, X)$，则有：

$$X \models b_U(P) \tag{3.22}$$

而$ECC_U(P, X)$保证了以下两点：

- 若原子$p \in (Atoms(b_U(P)) \setminus U) \cap X$，则$p$为真；
- 若原子$p \in (Atoms(b_U(P)) \setminus U) \setminus X$，则$p$为假。

$ECC_U(P, X)$确保了Y和X的一致性。且$e_U(P \setminus b_U(P), X)$仅对$P \setminus b_U(P)$中不会因$X$中的原子真值确定性而失去意义的规则操作，并仅会删去由U中原子导致为真的文字。所以有：

$$(X \cup Y) \cap Atoms(P) \models P \setminus b_U(P) \tag{3.23}$$

联合式(3.22)和式(3.23)可以得到$(X \cup Y) \cap Atoms(P) \models P$。故充分性得证。

($\Leftarrow$) 若$S \models P$，令$X = (S \cap U) \cup \{a \mid a \in S \cap (Atoms(b_U(P)) \setminus U)\} \cup \{a' \mid a' \in Atoms(b_U(P) \setminus (U \cup S)\}$，$Y = S \setminus U$。显然有$(X \cup Y) \cap Atoms(P) = S$。

由于$S \models P$，所以$\{a \mid a \in S \cap (Atoms(b_U(P)) \setminus U)\}$为$Atoms(b_U(P)) \setminus U$中满足$P$的原子部分；$Atoms(b_U(P) \setminus (U \cup S)$则为$Atoms(b_U(P)) \setminus U$中不满足$P$的原子部分。故有：

$$X \models b_U(P) \cup EC_U(P) \tag{3.24}$$

另，$S \models P$，故$S \models P \backslash b_U(P)$。由前面的分析可知$e_U(P \backslash b_U(P), X)$仅删去受$U$中原子真值影响的文字，所以$Y \models e_U(P \backslash b_U(P), X)$。又$ECC_U(P, X)$保证的是若一个原子$p \in X \setminus U$，则$p$为真。$S$为$P$的模型，故$S \setminus U$能满足$ECC_U(P, X)$。故有：

$$Y \models e_U(P \backslash b_U(P), X) \cup ECC_U(P, X) \tag{3.25}$$

由式(3.24)和式(3.25)可知必要性成立。 ■

接下来，本节给出一个验证命题3.2的例子。

例 3.3 令P_1的分割集为$U=\{a\}$，那么可以计算得到：

$$b_U(P_1) \ = \ \{a \ \leftarrow \ not\, d.\ a \ \leftarrow \ c,\ d.\} \tag{3.26}$$

及：

$$EC_U(P_1) \ = \ \{d \ \leftarrow \ not\, d'.\ d' \ \leftarrow \ not\, d.\ c \ \leftarrow \ not\, c'.\ c' \ \leftarrow \ not\, c.\} \tag{3.27}$$

且求解得到：

$$\Gamma(b_U(P_1) \cup EC_U(P_1)) = \{\{a,c,d\},\{a,c,d'\},\{a,c',d'\},\{c',d\}\} \tag{3.28}$$

令$X=\{a,c,d'\}$，显然，$X \models b_U(P_1) \cup EC_U(P_1)$。另，可以计算得到：

$$\begin{aligned} e_U(P_1 \backslash b_U(P_1), X) &= \{d \ \leftarrow \ not\, c.\ c \ \leftarrow .\} \\ ECC_U(P_1, X) &= \{\leftarrow \ not\, c.\ \leftarrow \ d.\} \end{aligned} \tag{3.29}$$

并求解得到：

$$\Gamma(e_U(P_1 \backslash b_U(P_1), X)) = \{\{c\}\} \tag{3.30}$$

令$Y=\{c\}$，显然，$Y \models e_U(P_1 \backslash b_U(P_1), X) \cup ECC_U(P_1, X)$。$X \cup Y = \{a,c,d'\}$，明显也有$(X \cup Y) \cap Atoms(P) \models P_1$。

在引入了$EC_U(P)$和$ECC_U(P,X)$后，依据命题3.2，可以知道原程序的模型可以通过$bottom$和top两部分的模型计算得到。然而，直接使用$P \setminus b_U(P)$作为top并不能求解出原程序剩下的回答集部分。这是因为分割集U把逻辑程序P划分为$bottom$和top两部分的同时也可能会破坏了P中的某些环L。

接下来，本节将基于Lin和Zhao(2002)的环与环公式的思想，来定义可以用于求解原程序剩余回答集部分的top。首先，这里定义两个规则集合。

定义 3.3 对一个ASP逻辑程序P基于一个原子集U，定义以下两个规则集合：

$$in_U(P) = \{r \in P \mid head(r) \cap U \neg \emptyset,\ (body^+(r) \cup head(r)) \subseteq U\} \tag{3.31}$$

$$out_U(P) = \{head(r) \subseteq U,\ (body^+(r) \cup head(r)) \cap U \neq \emptyset\} \tag{3.32}$$

$in_U(P)$和$out_U(P)$是与ASP逻辑程序正依赖图结构性相关的规则集合。在正规逻辑程序中直观的含义是：$in_U(P)$代表了头部属于分割集U，而体部正原子存在与U中原子不同的原子的规则集合；$out_U(P)$代表了头部不属于分割集U，而体部正原子存在与U中原子相同的原子的规则集合。根据两者的定义，显然可以得到以下集合关系：

$$in_U(P) \subseteq b_U(P) \tag{3.33}$$

$$out_U(P) \subseteq P \setminus b_U(P) \tag{3.34}$$

在有了$in_U(P)$和$out_U(P)$的概念后，本节给出以下定义。

定义 3.4 (半环(semi-loop)) 称一个非空原子集E为ASP逻辑程序P基于原子集U的半环，当且仅当在P中存在一个环L使得$E = L \cap U$且$E \subset L$。

定义 3.5 给定正规逻辑程序P和原子集X、U和E，且E为P基于U的半环，定义一个半环集合如下：

$$SL_U(P, X) = \{E \mid E \subseteq X,\ R^-(E, P, X) \subseteq in_U(P)\} \tag{3.35}$$

根据semi-loop和$SL_U(P, X)$的定义，可以得到以下引理。

引理 3.1 给定正规逻辑程序P和其分割集U，原子集$X \in \Gamma(b_U(P) \cup EC_U(P))$，一个半环$E$属于$SL_U(P, X)$，则$P$中存在一个环$L$满足$E \subset L$且$X \nvDash LF(L, P)$。

证明： 使用反证法。

分割集U在把ASP逻辑程序P划分为两部分的同时可能会破坏P中的某些环L。这些被破坏的环可以分为两类：

- 该环的环公式已经被$bottom$部分的回答集X所满足；
- 该环的环公式不能被$bottom$部分的回答集X所满足。

对于ASP逻辑程序P中一个环L，其与分割集U的关系可以分为以下的情况：

1）$L \subseteq U$。则对于规则$r \in R^-(L,P)$有$head(r) \cap L \neq \emptyset$，故$head(r) \cap U \neq \emptyset$，所以$r \in b_U(P)$，因此可以得到：

$$R^-(L,P) = R^-(L, b_U(P) \cup EC_U(P)) \tag{3.36}$$

故有：

$$LF(L,P) = LF(L, b_U(P) \cup EC_U(P)) \tag{3.37}$$

而$X \in \Gamma(b_U(P) \cup EC_U(P))$，所以有：

$$X \models LF(L, b_U(P) \cup EC_U(P)) \tag{3.38}$$

进而有：

$$X \models LF(L,P) \tag{3.39}$$

2）$L \not\subseteq U$且$L \cap U = \emptyset$，即$L \subseteq Atoms(P) \setminus U$。这样的环$L$与$b_U(P)$无关，完全在$P \setminus b_U(P)$中。因为$L \cap U = \emptyset$，则$L \not\subseteq X$。根据环公式被一个原子集满足的定义可以知道$L \not\subseteq X$使得前键为假，故定义中的蕴涵式为真，即有$X \models LF(L,P)$。

当环L完全在$b_U(P)$中时，或者环L完全在$P \backslash b_U(P)$中时，X能满足其环公式。实际上这样的环不属于被U破坏的环。以下情况的环为被U破坏的环。

3）$L \not\subseteq U$且$L \cap U \neq \emptyset$。

当$(L \cap U) \not\subseteq X$时，$L \not\subseteq X$，$X \models LF(L,P)$。

当$(L \cap U) \subseteq X$时，若$L \not\subseteq X$，则也有$X \models LF(L,P)$。这两种即前面说到的环被破坏了，但X能满足其环公式的情况。

若$L \subseteq X$时，若$R^-(L \cap U, P, X) \backslash in_U(P) \neq \emptyset$，即存在规则$r \in R^-(L \cap U, P, X) \backslash in_U(P)$，且有$R^-(L \cap U, P, X) \in b_U(P)$。

对于$r \notin in_U(P)$，所以有$body^+(r) \subseteq U$。

对于$r \in R^-(L \cap U, P, X)$，所以有$X \models body(r)$和$body^+(r) \cap (L \cap U) = \emptyset$，且已有$L \cap U \neq \emptyset$和$body^+(r) \subseteq U$，故$body^+(r) \cap L = \emptyset$。同时，$head(r) \cap (L \cap U) \neq \emptyset$，所以$head(r) \cap L \neq \emptyset$。故可以得到存在一个规则$r \in R^-(L,P)$且$X \models body(r)$，所以有$X \models LF(L,P)$。

故令$E = L \cap U$，$E \subseteq X$，且有$R^{-}(E, P, X) \subseteq in_U(P)$时，就存在一个环$L$有$X \nvDash LF(L, P)$。 ■

被分割集U破坏的环的环公式可能不能为$bottom$的回答集X所满足，所以需要剩下部分的回答集联合X一起去满足。定义$SL_U(P, X)$就是为了找出ASP逻辑程序P中被U破坏且环公式不能被X所满足的环。

根据上述的分析，引用Lin和Zhao(2002)提出的环理论，本节定义新的top的关键部分$t_U(P, X)$以保证top部分的回答集联合X能满足这部分环的环公式，进而保证ASP逻辑程序P中所有环的环公式都能被满足。

定义 3.6 给定P为一个正规逻辑程序，X和U为原子集，P基于U通过X得到的top记为$t_U(P, X)$，其由以下三部分组成：

- $P \backslash (b_U(P) \cup out_U(P))$，
- 对每个$E \in SL_U(P, X)$构造$\{x_E \leftarrow body(r) \mid r \in in_U(P), r \in R^{-}(E, P, X)\}$，
- $\{head(r) \leftarrow x_{E_1}, x_{E_2}, ..., x_{E_t}, body(r) \mid r \in out_U(P), E_i \in SL_U(P, X)(1 \leq i \leq t)$ 且 $body^{+}(r) \cap E_i \neq \emptyset\}$。

其中x_{E_i}为基于$SL_U(P, X)$中的semi-loop引入的新原子，对于这些新原子，最后通过逻辑交$Atoms(P)$即可消去。$t_U(P, X)$由三部分组成，其中第一部分从P中删去了$b_U(P)$和$out_U(P)$，事实上，$b_U(P)$和$out_U(P)$不是被单纯地删去，而是分别在$t_U(P, X)$的第二和第三部分根据$SL_U(P, X)$中的元素被进行重构，以此来补全那些被分割集U破坏了而$bottom$的回答集X不能满足其环公式的环。

在明确了$t_U(P, X)$的直观作用后，本节将对定义2.24进行重定义，给出新的ASP逻辑程序P基于分割集U的方案。

定义 3.7 (方案(Solution)) 给定正规逻辑程序P和其分割集U，P基于U的方案，是一组原子集$\langle X, Y \rangle$，其中：

- X是$b_U(P) \cup EC_U(P)$的一个回答集；
- Y是$e_U(t_U(P, X), X) \cup ECC_U(P, X)$的一个回答集。

称$e_U(t_U(P,X),X)\cup ECC_U(P,X)$为新的$top$。下面给出一个计算ASP逻辑程序$P$基于其分割集$U$的方案的例子。

例 3.4 继续使用ASP逻辑程序P_1，并令分割集$U=\{a\}$，根据定义3.3可以计算得到：

$$in_U(P_1)=\{a\ \leftarrow\ c,\ d.\} \tag{3.40}$$

$$out_U(P_1)=\{c\ \leftarrow\ a.\} \tag{3.41}$$

且可以从P_1中取环$L=\{a,c\}$，则有$E=L\cap U=\{a\}$，满足$E\subset L$。并由例3.3已经算得：

$$\Gamma(b_U(P_1)\cup EC_U(P_1))=\{\{a,c,d\},\{a,c,d'\},\{a,c',d'\},\{c',d\}\} \tag{3.42}$$

这里，取$X=\{a,c,d'\}$和$X'=\{a,c,d\}$，那么可以计算得到：

$$R^{-}(E,P_1,X)=\{a\ \leftarrow\ not\ d.\} \tag{3.43}$$

$$R^{-}(E,P_1,X')=\{a\ \leftarrow\ c,\ d.\} \tag{3.44}$$

则有：

$$SL_U(P_1,X)=\emptyset \tag{3.45}$$

$$SL_U(P_1,X')=\{\{a\}\} \tag{3.46}$$

并基于定义3.6计算得到：

$$t_U(P_1,X)=\{d\ \leftarrow\ not\ c.\ c\ \leftarrow\ a.\} \tag{3.47}$$

$$t_U(P_1,X')=\{d\ \leftarrow\ not\ c.\ x_{\{a\}}\ \leftarrow\ c,\ d.\ c\ \leftarrow\ x_{\{a\}},\ a.\} \tag{3.48}$$

另外，有：

$$ECC_U(P_1,X)=\{\leftarrow\ not\ c.\ \leftarrow\ d.\} \tag{3.49}$$

$$ECC_U(P_1,X')=\{\leftarrow\ not\ c.\ \leftarrow\ not\ d.\} \tag{3.50}$$

通过求解器计算得到：

$$\Gamma(e_U(t_U(P_1, X), X) \cup ECC_U(P_1, X)) = \{\{c\}\} \tag{3.51}$$

$$\Gamma(e_U(t_U(P_1, X'), X') \cup ECC_U(P_1, X')) = \emptyset \tag{3.52}$$

其中$\langle X, \{c\}\rangle$为P基于U的一个方案。

根据引理3.1的证明可以知道X满足了大部分情况下的环公式，对于X所不能满足的环公式的情况是：环$L \subseteq X$，但不存在规则$r \in R^-(L, P)$且$X \models body(r)$。若有另一个原子集Y对这样的环都符合存在规则$r \in R^-(L, P)$且$Y \models body(r)$，则可以得到$(X \cup Y) \cap Atoms(P)$能满足P中的所有环的环公式。

令Y是top部分的回答集，所以Y只可能满足$r \in R^-(L, P) \setminus b_U(P)$中的某些$body(r)$。但无法直接判定$Y$能满足哪些$body(r)$。若能在$P \setminus b_U(P)$部分中构造关于$L \setminus U$的环，则有$R^-(L, P) \backslash b_U(P) \subseteq R^-(L \backslash U, t_U(P, X))$，而$Y$为$top$的回答集，构造关于$L \backslash U$的环在$top$内，必然存在规则$r \in R^-(L, P) \setminus b_U(P)$且$Y \models body(r)$。而事实上，$t_U(P, X)$达到了构造这样关于$L \setminus U$的环的目的。

根据上述的分析，本节可以得到以下命题。

命题 3.3 给定ASP逻辑程序P及其分割集U，一组原子集$\langle X, Y\rangle$是P基于U的一组方案，则$(X \cup Y) \cap Atoms(P)$能满足P中的所有环的环公式。

证明： 这里考虑P中一个环L，其被U划分为两部分，构成环L的规则分别存在于$b_U(P)$和$P \setminus b_U(P)$两部分中，$L \subseteq X$但$X \not\models LF(L, P)$。

首先证明$t_U(P, X)$中的第二和第三部分能够在$P \setminus b_U(P)$中构造出关于$L \setminus U$的环。特别地，记$r_1 \in out_U(P)$和$r_2 \in in_U(P)$。

对于第二部分中构造的规则“$x_E \leftarrow body(r_2)$”，其中$r_2 \in in_U(P)$且$r_2 \in R^-(E, P, X)$。由于$r_2 \in in_U(P)$，则有$head(r_2) \cap U \neq \emptyset$，故$r_2 \in b_U(P)$，而且$r_2 \in R^-(E, P, X)$，故有：

$$head(r_2) \cap (L \cap U) \neq \emptyset \tag{3.53}$$

$$body^+(r_2) \cap (L \cap U) = \emptyset \tag{3.54}$$

因为L是P中的一个环，在构成L的规则r中，满足$head(r) \cap U \neq \emptyset$的规则$r$都在$b_U(P)$中，且在这些规则中必然存在规则$r_2$满足：

$$body^+(r_2) \cap (L \setminus U) \neq \emptyset \tag{3.55}$$

否则P中不会存在L这个环。对于每个满足$body^+(r_2) \cap (L \setminus U) \neq \emptyset$的规则$r_2$都会使用$E \in SL_U(P, X)$来构成一个形如$x_E \setminus body(r_2)$的新规则。此外，这些规则都是从$b_U(P)$中提取出来的，且保留了体部。而$X \in \Gamma(b_U(P) \cup ECC_U(P))$，故对于$r_2 \in b_U(P)$也必然符合$X \models body(r_2)$，不会与$r_2 \in R^-(E, P, X)$产生矛盾。

对于第三部分中构造的规则“$head(r_1) \leftarrow x_{E_1}, x_{E_2}, ..., x_{E_t}, body(r_1)$”，其中$r_1 \in out_U(P)$，所以有：

$$body^+(r_1) \cap (L \cap U) \neq \emptyset \tag{3.56}$$

而为了保证能构成L这个环，在$P \backslash b_U(P)$的规则中，对于满足$body^+(r_1) \cap (L \cap U) \neq \emptyset$的规则，必存在一个规则满足：

$$head(r_1) \cap (L \backslash U) \neq \emptyset \tag{3.57}$$

根据式子3.53和3.56，以及式子3.57和3.55，可以知道$(L \setminus U) \cup \{x_E\}$是一个环。即在$t_U(P, X)$中必能为被$U$破坏的环$L$构造出一个包含$L \setminus U$的环，该环就是$(L \setminus U) \cup \{x_E\}$。

另，必然存在规则$r_s \in R^-((L \setminus U) \cup \{x_E\}, P)$满足$body^+(r_s) \cap L = \emptyset$，即$r_s \in R^-(L, P)$，而$Y$作为$top$的回答集，必然有$Y \models body(r_s)$，此时基于$L \subseteq X$进行讨论，故有$L \subseteq (X \cup Y) \cap Atoms(P)$，所以$(X \cup Y) \cap Atoms(P) \models LF(L, P)$。

此外，$e_U(P, X)$只会删去U相关的原子，$ECC_U(P, X)$只包含约束，约束不会是构成环的规则，故这两个操作都不会影响环$(L \setminus U) \cup \{x_E\}$的某个外部支持的体部被$Y$所满足这个事实。

综合上面的证明，可以得到$(X \cup Y) \cap Atoms(P)$能满足那些被U破坏且X无法满足的环公式，故$(X \cup Y) \cap Atoms(P)$能满足P中的所有环公式。 ■

明确命题3.3的正确性后，本节提出一个引理，具体如下：

引理 3.2 给定一个正规逻辑程序P和其分割集U，如果$\langle X, Y\rangle$是P基于U的一组方案，且$SL_U(P, X) = \emptyset$，那么$(X \cup Y) \cap Atoms(P)$是P的一个回答集。

证明： 根据前面已提及的$SL_U(P, X)$的属性可知，如果$SL_U(P, X) = \emptyset$时，可以得到：在P中不存在环L同时满足以下三点：

- $L \cap U = \emptyset$，
- $L \cap (Atoms(P)\backslash U) \neq \emptyset$，
- $X \cup Y \nvDash LF(L, P)$。

所以可以进一步得到$(X \cup Y) \cap Atoms(P)$满足P中的所有环公式。而命题3.2中指出$(X \cup Y) \cap Atoms(P)$是P的模型，联合Lin和Zhao(2002)的环理论，即定理2.3，可以知道P的模型若能满足P中所有环公式，则该模型是P的一个回答集，所以可以得到$S = (X \cup Y) \cap Atoms(P)$是$P$的一个回答集。 ■

更进一步地，本节依据新程序分割方法，得到以下定理。

定理 3.1 (新分割理论) 已知正规逻辑程序P和任意原子集U，一个原子集S是P的回答集，当且仅当$S = (X \cup Y) \cap Atoms(P)$，其中$\langle X, Y\rangle$是$P$基于$U$的某组方案。

证明： $(\Rightarrow)$ 即证明当X和Y是P基于U的一组方案时，$(X \cup Y) \cap Atoms(P)$是P的回答集。而根据命题3.3可以知道有$(X \cup Y) \cap Atoms(P)$能满足P中的所有环公式。故只需证明$(X \cup Y) \cap Atoms(P)$满足P即可。

1）已知X是$b_U(P) \cup EC_U(P)$的回答集，$EC_U(P)$会为X引入$Atoms(P)$以外的新原子，所以有$X \cap Atoms(P)$满足$b_U(P)$中的所有规则。

2）已知Y是$e_U(t_U(P, X), X) \cup ECC_U(P, X)$的回答集。并且在$e_U(t_U(P, X), X) \cup ECC_U(P, X)$中引入的新原子$x_E$满足$\{x_E \mid E \in SL_U(P, X)\} = Y \setminus Atoms(P)$，则有$Y$满足$e_U(P\backslash b_U(P), X)$中的所有规则。而$e_U(P, X)$只对$U$中的原子进行删减操

作，且$ECC_U(P,X)$确保了在X中的原子均为真。所以有$(X \cup Y) \cap Atoms(P)$满足$P \backslash b_U(P)$。

综合上面两个情况，可以有$(X \cup Y) \cap Atoms(P)$满足P。所以有$(X \cup Y) \cap Atoms(P)$是P的回答集。故充分性成立。

$(\Leftarrow)$ 即证明当原子集S是P的回答集时，存在X和Y是P基于U的一组方案，同时有$S = (X \cup Y) \cap Atoms(P)$。令：

$$\begin{aligned} X = (S \cap U) \cup \{a \mid a \in S \cap Atoms(b_U(P) \setminus U)\} \\ \cup\{a' \mid a' \in Atoms(b_U(P)) \setminus (U \setminus S)\} \end{aligned} \tag{3.58}$$

$$Y = (S \setminus U) \cup \{x_E \mid E \in SL_U(P,X)\} \tag{3.59}$$

显然有$(X \cup Y) \cap Atoms(P) = S$，接着证明$\langle X, Y \rangle$是$P$基于$U$的一组方案。

首先证明X是$b_U(P) \cup EC_U(P)$的回答集。因为S是P的回答集，所以X中的$(S \cap U) \cup \{a \mid a \in S \cap Atoms(b_U(P) \backslash U)\}$部分就是$Atoms(b_U(P))$中属于$P$某个回答集的原子部分，$\{a' \mid a' \in Atoms(b_U(P)) \backslash (U \cup S)\}$部分就是$Atoms(b_U(P))$中不属于$P$任意回答集的原子部分，故可以有$X$满足$b_U(P) \cup EC_U(P)$。而对于任意$b_U(P) \cup EC_U(P)$中的环$L$，由于任意规则$r \in EC_U(P)$都有$body^+(r) = \emptyset$，所以这部分不会产生环，故$Atoms(L) \subseteq Atoms(b_U(P))$，所以有$R^-(L, b_U(P) \cup EC_U(P)) = R^-(L,P)$。又由于$S$满足$P$中的所有环公式$LF(L,P)$，所以$X$满足$b_U(P) \cup EC_U(P)$中的所有环公式$LF(L, b_U(P) \cup EC_U(P))$。故$X$是$b_U(P) \cup EC_U(P)$的回答集。

接着证明Y是$e_U(t_U(P,X),X) \cup ECC_U(P,X)$的回答集。$\{x_E \mid E \in SL_U(P,X)\}$是用于在$t_U(P,X)$中构造环的，而且这些环$(L \setminus U) \cup \{x_E\}$和满足$Atoms(L \setminus U) \subseteq Atoms(P \setminus b_U(P))$的环$L \backslash U$都需要被满足，故$x_E$必在$top$部分的回答集中，而$S$为$P$的回答集，所以能够得到$Y$满足$e_U(t_U(P,X),X) \cup ECC_U(P,X)$中的所有规则。

接着证明Y满足$e_U(t_U(P,X),X) \cup ECC_U(P,X)$中的所有环公式。根据前面的分析可知，$e_U(t_U(P,X),X) \cup ECC_U(P,X)$中的环分为两种情况。

1）$L \subseteq Atoms(P)\backslash U$，即原程序$P$中本来就在$P\backslash b_U(P)$中的环。即不存在$x_E \in L$。这种情况下的环，由于$L \cap U = \emptyset$，故$e_U(t_U(P,X),X)$只删去$U$中相关原子，同时，$ECC_U(P,X)$只有约束规则，不会产生环。所以两种情况都不会影响到L。而$t_U(P,X)$中的第二部分只对满足$head(r) \cap U \neq \emptyset$的规则进行修改，故也不对$L$造成影响。而$S$是原程序$P$的回答集，$S \setminus U$为$P \setminus b_U(P)$部分的回答集，且$L \cap U = \emptyset$，故$S\backslash U$可以满足这些环的环公式。那么显然地，$Y$也能满足这些环的环公式。

2）当存在$x_E \in L$时，根据前面的结论可知，含有x_E的环L都为了原程序中被U分割开且环公式不为X所满足的环L'而构造的。且构造的新环为$L = (L'\backslash U) \cup \{x_E\}$。若$Atoms((L'\backslash U) \cup \{x_E\}) \not\subseteq (S\backslash U) \cup \{x_E\}$（即$Y$），则根据环公式被满足的定义有$Y$满足该环公式。若$Atoms((L'\backslash U) \cup \{x_E\}) \subseteq (S\backslash U) \cup \{x_E\}$，而$S$为原程序$P$的回答集，所以$S$满足$P$中所有的环公式。且$(L'\backslash U) \cup \{x_E\}$在$P\backslash b_U(P)$的外部支持也是$L'$的外部支持。故$(S\backslash U)$也满足$(L'\backslash U) \cup \{x_E\}$的某个外部支持。所以$(S\backslash U) \cup \{x_E\}$，即$Y$满足$e_U(t_U(P,X),X) \cup ECC_U(P,X)$中的所有环公式。

所以Y是$e_U(t_U(P,X),X) \cup ECC_U(P,X)$的回答集。故必要性成立。 ■

在定理3.1中给出的新分割理论在内容上跟Lifschitz和Turner当初提出的分割理论是一样的。本质上的不同在于定义来计算P基于U的方案$\langle X, Y \rangle$的运算符的不同，及此时的分割集U是一个任意的原子集。

半环集$SL_U(P,X)$根据Lin和Zhao(2002)的环理论，定义了让$bottom$的回答集X所不能满足的环公式的环与分割集的交集所构成的半环集。定义这个半环集的目的就是为了依据它们构造出能保证X为真的规则，同时补全好被X破坏了的环。更进一步地，基于Ji和Wan等(2014)的特征环的思想，可以想象得到并非所有的半环都是必要的，本文在此根据Ji和Wan等(2014)中的思想，提取出必要的半环，定义一个$SL_U(P,X)$的子集以指代必要的半环集合。

定义 3.8 (关键半环(Dominated Semi-loop)) 给定正规逻辑程序P和原子集U，称一个半环E能代表另一个半环E'，当且仅当$E' \subseteq E$，且有$E' \cap head(in_U(P)) =$

$E \cap head(in_U(P))$，以及$E' \cap body^+(out_U(P)) = E \cap body^+(out_U(P))$。若一个半环$E$，不存在其他半环能代表它，则称$E$为关键半环。

根据关键半环的特性，对于关键半环E和任意一个E所能代表的半环E'，记L为top中与E相关的环，L'为top中与E'相关的环，那么$LF(L, t_U(P, X))$能推出$LF(L', t_U(P, X))$。

由于关键半环能够代替其他半环，所以这里定义一个关键半环的集合。

定义 3.9 (关键半环集) 给定析取逻辑程序P和原子集U，P关于U的关键半环集记为$DSL_U(P, X)$，并定义为以下集合：

$$\begin{aligned} DSL_U(P, X) = \{E \mid E \in SL_U(P, X) \text{ 且不存在另外一个} \\ E' \in SL_U(P, X) \text{ 满足 } E\text{能代表}E'\} \end{aligned} \tag{3.60}$$

显然，$DSL_U(P, X) \subseteq SL_U(P, X)$。同时，本节在此提供一个计算$DSL_U(P, X)$的算法$Algorithm$ 3，并有以下命题。

命题 3.4 给定一个ASP逻辑程序P和一个原子集X，记$Algorithm$ 3中的函数$dsl_U(P, X)$返回的结果为dsl，则有：

- dsl中的元素都是P基于X的关键半环；
- P基于X的任意关键半环E，满足$E \in dsl$。

证明： 对于命题3.4中的结论一，根据$DSL_U(P, X)$的定义可知，只需关注$head(in_U(P))$和$body^+(out_U(P))$。故$dsl_U(P, X)$在最外层对$head(in_U(P))$和$body^+(out_U(P))$的所有非空子集进行遍历。$dsl_U(P, X)$把原程序P中的原子分为两部分：

- $AP_1 = Atoms(P) \backslash (head(in_U(P)) \cup body^+(out_U(P))$
- $AP_2 = head(in_U(P)) \cup body^+(out_U(P)$

然后，$dsl_U(P, X)$基于$AP_1 \cup S_1 \cup S_2$从G_P中提取子图G_P^S，并查找G_P^S中包含$S_1 \cup S_2$的强连通分量L。强连通分量保证其是一系列环中在集合意义上是最大

Algorithm 3: 计算关键半环集的算法$dsl_U(P,X)$

输入: 一个逻辑程序P和一个原子集X

输出: 关键半环集dsl

$dsl := \emptyset$;
for 非空子集$S_1 \subseteq head(in_U(P))$和$S_2 \subseteq body^+(out_U(P))$ **do**
    $G_P^S := G_P$关于$S_1 \cup S_2 \cup (Atoms(P) \backslash (head(in_U(P)) \cup body^+(out_U(P))))$的子图;
    $L := G_P^S$的一个强连通分量，并满足$S_1 \cup S_2 \subseteq L$;
    if 不存在这样的L **then**
        break;
    end
    if $L \cap U \subseteq X$且$R^-(L \cap U, P, X) \subseteq in_U(P)$ **then**
        把$L \cap U$加入到dsl中;
    end
    else
        $S := head(R^-(L \cap U, P, X) \backslash in_U(P)) \cup ((L \cap U) \backslash X)$;
        $G_P^S := G_P$关于$L \backslash S$的子图;
        goto 4;
    end
end
return dsl

的一个，同时根据其满足$S_1 \cup S_2 \subseteq L$，所以L的任意非空子集L'都满足：

$$L' \cap head(in_U(P)) \neq \emptyset \text{ 且 } L' \cap head(in_U(P)) = L \cap head(in_U(P)) \tag{3.61}$$

$$L' \cap body^+(out_U(P)) \neq \emptyset \text{ 且 } L' \cap body^+(out_U(P)) = L \cap body^+(out_U(P)) \tag{3.62}$$

故这样的强连通分量L若不存在，则可以跳过当前的G_P^S子图，继续尝试下个子图。如果存在强连通分量L，则进一步判断其是否满足以下条件，以确定当前的L是否$SL_U(P,X)$中的元素。

$$L \cap U \subseteq X\ ,\ R^-(L \cap U, P, X) \subseteq in_U(P) \tag{3.63}$$

如果满足条件，则$L \cap U$是$DLS_U(P,X)$中的元素，且前面有L是SCC，且L满足式3.61和3.62，则L是一个关键半环。同时$dsl_U(P,X)$把L加入到dsl中，故dsl中的元素都是关键半环。

对于命题3.4中的结论二，在$dsl_U(P,X)$中，如果L不符合条件3.63，则$L\notin SL_U(P,X)$，而$dsl_U(P,X)$会继续找出导致不满足条件3.63的原子集S，如下：

$$S = head(R^-(L\cap U,P,X)\backslash in_U(P))\cup((L\backslash U)\backslash X) \tag{3.64}$$

令G_P^S是G_P关于$L\setminus S$的子图，即可能满足条件3.63的原子子图。并从该子图中重复寻找强连通分量L及判断其是否满足条件的过程。故所有可能构成关键半环的SCC的环都会被$dsl_U(P,X)$遍历，并且判断其符合关键半环条件时，加入到dsl中。故P基于X的任意关键半环E，都满足$E\in dsl$。 ■

基于$DSL_U(P,X)$，本节定义关键顶部和关键方案。

定义 3.10 (关键顶部(Dominated Top)) 给定P为一个正规逻辑程序，X和U为原子集，P基于U通过X得到的关键顶部记为$dt_U(P,X)$，其由以下三部分组成：

- $P\backslash(b_U(P)\cup out_U(P))$，
- 对每个$E\in DSL_U(P,X)$构造$\{x_E \leftarrow body(r)\,|\,r\in in_U(P),\ r\in R^-(E,P,X)\}$，
- $\{head(r) \leftarrow x_{E_1},x_{E_2},...,x_{E_t},body(r)\mid r\in out_U(P),\ E_i\in DSL_U(P,X)(1\leq i\leq t)$ 且 $body^+(r)\cap E_i\neq\emptyset\}$。

定义 3.11 (关键方案(Dominated Solution)) 给定正规逻辑程序P和其分割集U，P基于U的关键方案，是一组原子集$\langle X,Y\rangle$，其中：

- X是$b_U(P)\cup EC_U(P)$的一个回答集；
- Y是$e_U(dt_U(P,X),X)\cup ECC_U(P,X)$的一个回答集。

显然，它们的具体内容就是通过把之前定义的顶部和方案中的$SL_U(P,X)$替换为$DSL_U(P,X)$。最后本节扩展新的分割集理论，得到以下定理。

定理 3.2 给定一个正规逻辑程序P和原子集U，一个原子集S是P的一个回答集，当且仅当$S=(X\cup Y)\cap Atoms(P)$，其中$\langle X,Y\rangle$是P基于U的某组关键方案。

证明： 由于关键半环在top中对应的环的环公式能推出其代表的半环对应的环的环公式。联合定理3.1及Lin和Zhao(2002)的环理论，可以得到定理3.2成立。 ■

新程序分割方法对于$bottom$部分，通过定义$EC_U(P)$来补充$Atoms(b_U(P))\backslash U$中原子的真值的可能性，以保证这部分原子不会被忽略。

对于top部分，定义了$ECC_U(P,X)$以确保$Atoms(top) \cap \Gamma(bottom)$这部分原子必为真。同时使用Lin和Zhao(2002)的环理论，引入semi-loop的集合$SL_U(P,X)$，其代表的是环公式不能被底部回答集所满足的环所相关的semi-loop集合。基于$SL_U(P,X)$和原来的$P\backslash b_U(P)$来构建出新的top，即定义3.6中的$t_U(P,X)$。

在上述的定义和分析证明过程中可以知道新程序分割方法能对任意原子集构成的分割集都有效。

3.2.2 析取逻辑程序分割方法

本节将把新程序分割方法从正规逻辑程序扩展到析取逻辑程序。由于之前所定义下的规则集合都是基于普通规则所定义的，所以对于析取逻辑程序，这些规则集合的操作符依旧有用。

具体地，$b_U(P)$，$e_U(P,X)$，$EC_U(P)$，$ECC_U(P,X)$，$in_U(P)$，$out_U(P)$，$SL_U(P,X)$和$DSL_U(P,X)$都保持原来的定义不变。跟正规逻辑程序有所不同的是，在析取逻辑程序中，$in_U(P) \cap out_U(P) \neq \emptyset$。在明确了需要用到的基本概念后，本节将重新定义析取逻辑程序的top的关键部分。

定义 3.12 (析取逻辑程序的顶部(DLP top)) 给定一个析取逻辑程序P和原子集X、U，P基于U关于X的top记为$dlpt_U(P,X)$，并由以下三部分组成：

- $P\backslash(b_U(P) \cup out_U(P)$，
- 对于每个$E \in SL_U(P,X)$构造$\{\{x_E\} \cup head(r)\backslash E \leftarrow body(r) \mid r \in in_U(P)$ 且 $r \in R^-(E,P,X)\}$，
- $\{head(r) \leftarrow x_{E_1}, x_{E_2}, ..., x_{E_t}, body(r) \mid r \in out_U(P),\ E_i \in SL_U(P,X)(1 \leq i \leq t)$ 且 $body^+(r) \cap E_i \neq \emptyset\}$。

根据析取逻辑程序的top的定义，可以得到以下命题。

命题 3.5 给定一个析取逻辑程序P，及其分割集U和其$bottom$部分的回答集X，环L有$L \subseteq X$且$X \nvDash LF(L,P)$，则$dlpt_U(P,X)$中的能构成环$(L \setminus U) \cup \{x_E\}$，其中$E \in SL_U(P,X)$。

证明： 显然，$dlpt_U(P,X)$与正规逻辑程序的$t_U(P,X)$的差别只在第二部分。在正规逻辑程序中，对于$head(r) \cap E \neq \emptyset$的情况下，由于$|head(r)| = 1$，所以实际为$head(r) \in E$。故在正规逻辑程序的$t_U(P,X)$的第二部分直接用$x_E$替换原来的$hear(r)$即可。而在析取逻辑程序中$|head(r)| \geq 1$，所以不能直接用$x_E$替换原来的$hear(r)$。而是直接把$head(r) \cap E$部分替换为$x_E$，故头部修正为$\{x_E\} \cup (head(r) \backslash E)$。

同时，由于x_E的存在，且第三部分中规则的$head(r)$本来就不需要修改，故依旧能保持环$(L \setminus U) \cup \{x_E\}$的构成。此外，构成$(L \backslash U) \cup \{x_E\}$环后的有效性跟正规逻辑程序一致。 ■

例 3.5 给定原子集$U = \{a\}$，及析取逻辑程序P_2：

$$a \vee d \leftarrow \ . \tag{3.65}$$

$$d \leftarrow not\ c. \tag{3.66}$$

$$a \leftarrow c,\ d. \tag{3.67}$$

$$c \leftarrow a. \tag{3.68}$$

可以计算得到：

$$b_U(P_2) = \{a \vee d \leftarrow \ . \ a \leftarrow c,\ d.\} \tag{3.69}$$

$$EC_U(P_2) = \{c \leftarrow not\ c'.\ c' \leftarrow not\ c.\ d \leftarrow not\ d'.\ d' \leftarrow not\ d.\} \tag{3.70}$$

求解得到：$X = \{a, c, d\} \in \Gamma(b_U(P_2) \cup EC_U(P_2))$。$L = \{a, c\}$是$P_2$中的环，故有semi-loop $E = \{a\}$，并计算得到：

$$ECC_U(P_2, X) = \{\leftarrow not\ c.\ \leftarrow not\ d.\} \tag{3.71}$$

$$R^{-}(E, P_2, X) = \{a \leftarrow c, d.\} \tag{3.72}$$

$$SL_U(P_2, X) = \{\{a\}\} \tag{3.73}$$

进而得到：

$$dlpt_U(P_2, X) = \{d \leftarrow not\, c.\ x_{\{a\}} \leftarrow c, d.\ c \leftarrow x_{\{a\}}, a.\} \tag{3.74}$$

根据析取逻辑程序的top，同样可以定义析取逻辑程序的方案，如下：

定义 3.13 (析取逻辑程序的方案(DLP Solution)) 给定析取逻辑程序P和其分割集U，P基于U的方案，是一组原子集$\langle X, Y\rangle$，其中：

- X是$b_U(P) \cup EC_U(P)$的一个回答集；
- Y是$e_U(dlpt_U(P, X), X) \cup ECC_U(P, X)$的一个回答集。

基于上述的分析，本节给出关于析取逻辑程序在程序分割理论下的回答集定理。

定理 3.3 给定析取逻辑程序P和原子集U，一个原子集S是P的一个回答集，当且仅当$S = (X \cup Y) \cap Atoms(P)$，其中$\langle X, Y\rangle$是$P$基于$U$的某组方案。

证明： 联合命题3.5的证明和定理3.1的证明可以直接地得到定理3.3的正确性。 ■

此外，对于析取逻辑程序，同样可以在$dlpt_U(P, X)$中使用$DSL_U(P, X)$代替$SL_U(P, X)$，可以取得与正规逻辑程序类似的效果。

3.2.3 强程序分割方法

在Lifschitz和Turner的分割理论中，分割集U把逻辑程序P分割为$b_U(P)$和$P \backslash b_U(P)$两部分。这个特点除了可以得到定理2.4之外，还能引申出来其他的结论。对此，本节总结出如下命题。

命题 3.6 给定逻辑程序P，原子集U是P在Lifschitz和Turner的分割理论定义下的分割集。另有逻辑程序P'满足$Atoms(P') \cap U = \emptyset$，则原子集$S$是逻辑程序$P \cup P'$的一个回答集，当且仅当$S = X \cup Y$，其中$X$为$b_U(P)$的某个回答集，$Y$为$e_U(P \backslash b_U(P), X) \cup P'$的某个回答集。

证明： 由于$Atoms(P') \cap U = \emptyset$，且$\Gamma(b_U(P)) \subseteq U$，所以$\Gamma(P') \cap \Gamma(b_U(P)) = \emptyset$，同时有$\Gamma(P') \cap U = \emptyset$。所以$P \cup P'$中划分出的$b_U(P)$依旧是一个独立的结构，其回答集依旧不受$(P \cup P')\backslash b_U(P)$影响。

又在$Atoms(P') \cap U = \emptyset$及$Atoms(b_U(P)) \subseteq U$情况下，可以得到：

$$(P \cup P')\backslash b_U(P) \equiv (P \backslash b_U(P)) \cup P' \tag{3.75}$$

而$e_U(P, X)$操作中只会依旧条件删去与U有关的文字以保证X中的原子的真值确定性。所以在$Atoms(P') \cap U = \emptyset$情况下，$e_U(P, X)$不需要对$P'$操作，而$P'$所能推出的结论也不会干扰$P \backslash b_U(P)$或受其干扰，所以$e_U(P \backslash b_U(P), X) \cup P'$可以求解出$P \cup P'$的另一部分回答集。■

然而，对于使用任意原子集作为分割集时，命题3.6并不成立。但基于这个思想，为了继续扩展分割集理论对逻辑程序的结合律有效性，本节提出了强程序分割方法(Strong Splitting Method)。为此，本节对$SL_U(P, X)$进行了扩展。

定义 3.14 给定析取逻辑程序P以及原子集X和U，标记以下原子集的集合：

$$\begin{aligned} SS_U(P, X) \ = \ \{E \mid E \text{ 为 } U \text{ 的任意非空子集，} \\ E \subseteq X \text{ 且 } R^{-}(E, P, X) \subseteq in_U(P)\} \end{aligned} \tag{3.76}$$

很直观地，$SS_U(P, X)$把$SL_U(P, X)$从P关于U的smei-loop扩展到U的任意非空子集。同时，对任何满足$Atoms(P') \cap U = \emptyset$的析取逻辑程序$P'$，则$P' \cap in_U(P) = \emptyset$，故不会影响判断$E \subseteq X$和$R^{-}(E, P, X) \subseteq in_U(P)$等条件是否成立，故其对$SL_U(P, X)$中的元素没有附加限制，而且$SS_U(P, X)$中的$E$扩展为$U$的任意非空子集，则可以得到$b_U(P \cup P') = b_U(P)$，且对任意的原子集$X$有$SL_U(P \cup P', X) \subseteq$

$SS_U(P,X)$。此外，若U是Lifschitz和Turner的分割理论定义下的分割集，则对任意原子集X都有$SS_U(P,X)=\emptyset$。基于$SS_U(P,X)$，本节提出了析取逻辑程序P基于原子集U关于原子集X的强顶部。

定义 3.15 (强顶部(Strong Top)) 给定析取逻辑程序P以及原子集X和U，记P基于U关于X的强顶部为$st_U(P,X)$，其由以下三部分组成：

- $P\backslash(b_U(P)\cup out_U(P)$，
- 对于每个$E\in SS_U(P,X)$构造$\{\{x_E\}\cup head(r)\backslash E\leftarrow body(r)\mid r\in in_U(P)$ 且 $r\in R^-(E,P,X)\}$，
- $\{head(r)\leftarrow x_{E_1},x_{E_2},...,x_{E_t},body(r)\mid r\in out_U(P),\ E_i\in SS_U(P,X)(1\le i\le t)$ 且 $body^+(r)\cap E_i\neq\emptyset\}$。

在给出$st_U(P,X)$后，本节定义一个析取逻辑程序P关于原子集U的强方案。

定义 3.16 (强方案(Strong Solution)) 给定析取逻辑程序P和原子集U，P关于U的强方案为一个原子集组合$\langle X,Y\rangle$，其中X和Y满足：

- X是$b_U(P)\cup EC_U(P)$的一个回答集；
- Y是$e_U(st_U(P,X),X)\cup ECC_U(P,X)$的一个回答集。

在有了强方案的定义后，依据$SS_U(P,X)$对$SL_U(P,X)$性质的扩展，可以得到以下两个定理。

定理 3.4 给定析取逻辑程序P和原子集U。一个原子集S是P的一个回答集，当且仅当$S=(X\cup Y)\cap Atoms(P)$，其中$\langle X,Y\rangle$是P基于U的某组强方案。

定理3.4的正确性可以直接从定理3.2的证明中得到。

定理 3.5 给定析取逻辑程序P和原子集U。另有逻辑程序P'满足$Atoms(P')\cap U=\emptyset$。则原子集$S$是$P\cup P'$的一个回答集，当且仅当$S=(X\cup Y)\cap Atoms(P)$，其中$X$为$b_U(P)\cup EC_U(P)$的某个回答集，$Y$为$e_U(st_U(P,X),X)\cup ECC_U(P,X)\cup P'$的某个回答集。

证明： 由于$Atoms(P') \cap U = \emptyset$，所以$b_U(P \cup P') = b_U(P)$，故一样可以把$P \cup P'$通过$U$来分割，而不造成对$bottom$的影响，$P$只影响$top$部分。而$P'$的影响就是会造成原来在$P \backslash b_U(P)$中不是环的原子集变成了环，所以直接修改$SS_U(P, X)$为$U$中任意被破坏的非空子集。这些非空子集是可能因为$P'$而被构成环的。

因为$Atoms(P') \cap U = \emptyset$，所以对$(P \cup P') \backslash b_U(P)$进行$st_U(P, X)$和$e_U(P, X)$操作的话，实际上$P'$中的规则不会被操作，这是因为$st_U(P, X)$只对与$E \subseteq U$有关的规则操作。且$Atoms(P') \cap U = \emptyset$，所以$Atoms(P') \cap E = \emptyset$，而$e_U(P, X)$只对与$U$相关的规则操作，所以也不会涉及$P'$中的规则。故只需对$P$执行$e_U(P, X)$操作，并在最后加上$P'$。

所以对于$e_U(st_U(P, X), X) \cup ECC_U(P, X)$这部分，它保证了所有不被$X$满足的环公式的子集都被构造出相应的环，而且这些环会被满足，此外，去掉了X带来的原子真值确定性影响。在$e_U(st_U(P, X), X) \cup ECC_U(P, X)$中加入$P'$后求得的回答集$Y$与$X$结合就是这部分所能满足原程序所有环公式的回答集。 ■

定理3.5把程序分割方法的结合性扩展到以任意原子集作为分割集的情况。而这种结合性的本质是一个二次分割，这种扩展有助于新程序分割方法的递归使用问题研究。

3.3 计算复杂性分析

本节将分析分割集如何影响程序分割过程的计算复杂性，并指出符合哪些条件的原子集作为分割集才会让程序分割更高效。本节以正规逻辑程序在程序分割方法上的计算复杂性为例进行分析。

对于一个正规逻辑程序P通过一个原子集U进行程序分割的过程中主要耗时的有两个地方，分别为bottom的$EC_U(P)$和top的$DSL_U(P, X)$所造成。具体体现为以下两个问题：

1. 在bottom中，$EC_U(P)$中的规则在最坏情况下会为$b_U(P) \cup EC_U(P)$带来$2^{|Atoms(b_U(P) \backslash U|}$个回答集；

2. 在top中，$DSL_U(P,X)$中的半环为$dt_U(P)$引入的新原子x_E的数量会发生指数爆炸。

接下来的小节将分别对上述两个问题进行具体的分析和给出改善的方案。

3.3.1 底部计算复杂性

本节对第一个问题进行分析。bottom中的$EC_U(P)$是为了补充$Atoms(b_U(P))\setminus U$中的原子在回答集中的真值可能性而定义的，就是这种真值可能性的遍历行为导致了回答集的指数增多。对于$EC_U(P)$带来的耗时，本节给出以下两个降低计算复杂性的措施：

- 保证$|Atoms(b_U(P))\setminus U|$足够小，最好是在某个常数值以内；
- 让分割集U中的原子能被逻辑程序P的每个回答集所满足。

第一个措施中，在$|Atoms(b_U(P))\setminus U|$足够小的情况下，特别地，是保持在某个常数值以内的话，$2^{|Atoms(b_U(P))\setminus U|}$的大小就会被限制，这样就能防止bottom部分的回答集数量因为引入的新原子而造成指数爆炸。此外，由于分割集能够是任意原子集，所以限制$|Atoms(b_U(P))\setminus U|$的大小是具备可行性的。

第二个措施中，若能使得分割集U中的原子能被ASP逻辑程序P的每个回答集所满足，那么这在后续计算top部分的$dt_U(P)$时只需考虑$b_U(P)\cup EC_U(P)$中满足$U\subseteq X$的回答集X。这样就能剪枝掉一系列X，达到降低$dt_U(P)$的复杂性。

需要明确的一点是，能被逻辑程序P的每个回答集所满足的原子集就是一个程序结论，其有独立的定义和算法，不需要求解出逻辑程序的所有回答集才能得到。所以找出这样的原子集去作为分割集是可行的。而这样的原子集也将给本文带来分割集应用的新思路。具体的应用场景将在第四章给出。

3.3.2 顶部计算复杂性

本小节主要对第二个问题进行分析。关于top部分中的$DSL_U(P,X)$，依据其定义可以知道其中的关键半环E与其所能代表的半环E'必须满足：

$$E\cap head(in_U(P)) = E'\cap head(in_U(P)) \tag{3.77}$$

$$E \cap body^+(out_U(P)) = E' \cap body^+(out_U(P)) \tag{3.78}$$

而关键半环是可以代表其他一般半环的，此外，半环本来是由一般的环交分割集而来，所以保证了不存在两个不同的关键半环可以同时代表一个一般半环的情况。故在关键半环的互斥性之下，关键半环的数量的理论最大值将由$head(in_U(P))$和$body^+(out_U(P))$的基所决定。

更具体地来说就是由$E \cap head(in_U(P))$和$E \cap body^+(out_U(P))$的可能情况数量所决定。设$n = |head(in_U(P))|$和$m = |body^+(out_U(P))|$，那么有：

$$|E \cap head(in_U(P))| \ \leq \ \Sigma_{0 \leq i \leq n} C_n^i = 2^n \tag{3.79}$$

同理，还有：

$$|E \cap body^+(out_U(P))| \leq 2^m \tag{3.80}$$

故可以得到：

$$|DSL_U(P,X)| \leq |E \cap head(in_U(P))| * |E \cap body^+(out_U(P))| \leq 2^n * 2^m \tag{3.81}$$

从式3.81可以明显地知道通过限制$|head(in_U(P))|$和$|body^+(out_U(P))|$的大小就可以限制$|DSL_U(P,X)|$的大小。而根据$in_U(P)$和$out_U(P)$的定义，$|head(in_U(P))|$和$|body^+(out_U(P))|$的大小依旧可以通过分割集U的基来限制。在限制$|DSL_U(P,X)|$的大小后，在$dt_U(P)$中的第二和第三部分中将会大量减少新原子x_E的引入。进而降低top部分的程序规模，加快求解速度。本质上，这是人为地减少了被分割集U破坏的环的数量。

上述的计算复杂性都是基于正规逻辑程序而进行的。而事实上，正规逻辑程序的程序分割的计算复杂性的相关结论可以应用于析取逻辑程序上。因为两者仅在规则头部的基上有所不同，且头部的原子关系仅为析取。

3.4 本章小结

本章首先给出了一个计算Lifschitz和Turner所定义的分割集的算法。并分别为正规逻辑程序和析取逻辑程序定义了top中的关键部分：$t_U(P,X)$和$dlpt_U(P,X)$，

以及补充原子真值可能性的$EC_U(P)$和确保X中原子的真值确定性的$ECC_U(P,X)$，以此构造出对任意原子集作为分割集都有效的新程序分割方法。同时，基于环理论，提出了一个具有代表性的关键半环集的定义，并给出了具有工程意义的求解半环集的方法。最后给出了强程序分割方法。

此外，本章还对新程序分割方法的计算复杂性进行了分析，并根据主要耗时的点，逐一给出了改进的方案。

第 4 章　程序化简

本章将把分割集理论应用到程序化简中。实际上是把程序结论作为分割集对原程序进行分割，再构造相应的顶部以达到化简的效果。本章将定义两个化简算子和程序结论顶部来完成分割集理论在程序化简上的应用，并给出相关的证明。此外，程序结论并不需要在求解出逻辑程序的所有回答集后才能得到，第二章已经给出了相关计算方法－－式2.20。这样也保证了本章使用程序结论的可行性。

4.1　正负文字化简算子

程序结论中的文字的真值在对应的逻辑程序中的所有回答集里都为真。基于这个确定性，可以把原程序中的一些文字及其所影响的规则删掉，得到一个简化的逻辑程序，并对化简后的逻辑程序进行求解，得到的回答集联合程序结论即可得到原程序的回答集。这就是对ASP逻辑程序使用程序化简的基本思路。

4.1.1　正负文字化简算子的定义

基于程序结论中的文字的真值确定性，本节定义两个基于程序结论去简化逻辑程序的化简操作符。它们为原逻辑程序删去受程序结论影响的文字和规则。下面将给出这两个化简操作符的具体定义。

定义 4.1 (负文字化简) 给定ASP逻辑程序P和它的一个程序结论L，P基于L的负文字化简记为$tr_n(P, L)$。$tr_n(P, L)$对P执行以下两步操作：

- 若P中的一个规则r中存在原子$p \in body^+(r)$，且有$\neg p \in L$，则从P中删去r；
- 若P中的一个规则r中存在原子$p \in head(r)$或$p \in body^-(r)$，且有$\neg p \in L$，则把该规则r替换为以下形式：

$$head(r)\backslash\overline{L} \leftarrow body^+(r),\ body^-(r)\backslash\overline{L} \tag{4.1}$$

定义 4.2 (正文字化简) 给定ASP逻辑程序P和它的一个程序结论L，P基于L的正文字化简记为$tr_p(P, L)$。$tr_p(P, L)$对P执行以下两步操作：

- 若P中的一个规则r中存在原子$p \in head(r)$或$p \in body^-(r)$，且有$p \in L$，则从P中删去r；
- 若P中的一个规则r中存在原子$p \in body^+(r)$，且有$p \in L$，则把该规则r替换为以下形式：

$$head(r) \ \leftarrow \ body^+(r)\backslash L, \, body^-(r) \tag{4.2}$$

根据正、负文字化简算子的定义，可以得到以下命题：

命题 4.1 给定一个ASP逻辑程序P及其一个程序结论L，则有以下两个结论成立：

- 若原子集S是$tr_n(P,L)$的模型，则S也是P的模型；
- 若原子集S是$tr_p(P,L)$的模型，且S与L一致，则$S \cup L$是P的模型。

证明： 对于结论一，S为$tr_n(P,L)$的模型，则必有S满足$P \cap tr_n(P,L)$中的所有规则。而负文字化简$tr_n(P,L)$只考虑程序结论L中的负文字。在$\neg p \in L$时，基于程序结论的定义性质，可以得到推不出p为真，所以得到p为假，$not\ p$为真。根据$tr_n(P,L)$的定义，对于$P \setminus tr_n(P,L)$部分的规则，可分为以下两种情况：

- 第一种情况下的规则r满足$p \in body^+(r)$。在确定p为假及体部原子仅为合取关系的情况下，可以得到$body(r)$为假，对于这种规则，前键为假，整体为真，所以S必会满足这种规则。
- 第二种情况下的规则r形如"$head(r') \cup \overline{L} \ \leftarrow \ body^+(r'), body^-(r) \cup \overline{L}$"，其中规则$r' \in tr_n(P,L)$。$S$作为$tr_n(P,L)$的模型，$S$满足$r'$，而对于原子$p' \in \overline{L}$有$p'$为假，$not\ p'$为真。根据头部仅有析取关系，体部仅有合取关系，故也有S满足r。

故有S也满足$P \setminus tr_n(P,L)$中的所有规则，所以S为P的模型。

对于结论二，S为$tr_p(P,L)$的模型，则必有S满足$P \cap tr_p(P,L)$中的所有规则。而正文字化简$tr_p(P,L)$只考虑程序结论L中的正文字。在$p \in L$时，基于程序结论的定义性质，可以推出p为真，$not\ p$为假。根据$tr_p(P,L)$的定义，对于$P \setminus tr_p(P,L)$部分的规则，可分为以下两种情况：

- 第一种情况下的规则r满足$p \in head(r)$或$p \in body^-(r)$。在确定p为真及头部原子仅为析取关系的情况下，可以确定$head(r)$为真，后键为真，故r必为真，所以有S满足r。同样地，在明确$not\ p$为假及体部原子仅为合取关系的情况下，可以确定$body(r)$为假，前键为假，故r必为真，所以有S满足r。故S满足这种情况下的规则。
- 第二种情况下的规则r形如$head(r') \leftarrow body^+(r') \cup L,\ body^-(r')$，其中$r' \in tr_p(P, L)$。已有$S$满足$r'$，所以$S \cup L$满足$body^+(r') \cup L$，故也有$S$满足$r$。

故有$S \cup L$满足$P \setminus tr_p(P, L)$中的所有规则，所以$S \cup L$为P的模型。 ■

以下给出一个计算$tr_n(P, L)$和$tr_p(P, L)$的例子：

例 4.1 给定ASP逻辑程序P_3如下，及其程序结论$L = \{a,\ c,\ \neg b\}$：

$$a \leftarrow not\ b. \tag{4.3}$$

$$b \leftarrow not\ a. \tag{4.4}$$

$$\leftarrow not\ a. \tag{4.5}$$

$$c \leftarrow a. \tag{4.6}$$

$$d \leftarrow c. \tag{4.7}$$

$$a \leftarrow d. \tag{4.8}$$

$$f \leftarrow b. \tag{4.9}$$

$$b \leftarrow f. \tag{4.10}$$

则有：

$$\begin{aligned} tr_n(P_3, L) = \{a \leftarrow .\ b \leftarrow not\ a.\ \leftarrow not\ a. \\ c \leftarrow a.\ d \leftarrow c.\ a \leftarrow d.\ \leftarrow f.\} \end{aligned} \tag{4.11}$$

及：

$$tr_p(P_3, L) = \{d \leftarrow .\ f \leftarrow b.\ b \leftarrow f.\} \tag{4.12}$$

根据正文字化简和负文字化简算子的定义可以知道给定一个ASP逻辑程序P和它的一个程序结论L，$tr_n(P, L)$得到的ASP逻辑程序中将不含任何满

足$\neg p \in L$的原子p。同理，$tr_p(P, L)$得到的ASP逻辑程序中将不含任何满足$p \in L$的原子p。那么$tr_p(tr_n(P, L), L)$得到的逻辑程序中将不含任何满足$p \in Atoms(L)$的原子p。即对于一个ASP逻辑程序P，$tr_p(tr_n(P, L), L)$使得P基于它的程序结论L把已经可以确定真值的文字及这些文字对P的影响移除。

这个操作的实质效果便是：在已知一部分回答集（即程序结论）的情况下，去掉这部分回答集对P的影响，在剩下的部分里求解出余下的回答集部分。这个过程跟程序分割方法通过bottom求解出X，再代X入top求解出剩下的回答集部分Y的思路是一致的。

Van和Gelder等(1991)提出的正规逻辑程序的良序模型(Well-founded Model)[44]也是正规逻辑程序的一个程序结论。当前的主要ASP求解器都内置地使用了良序模型去化简程序。但存在其他比良序模型的基要大的程序结论，而使用这些程序结论将比良序模型更有效地化简程序[41]。明显地，程序结论的基越大，能消去的确定性的的文字所带来的影响越多，化简后的程序规模越小，求解越快。Chen，Ji和Lin(2013)提出了通过使用最多只有一个外部支持的环的环公式去计算程序结论的方法，能有效地得到比良序模型的基更大的程序结论[41]。

4.1.2 正负文字化简算子的适用性

$tr_n(P, L)$和$tr_p(P, L)$两者虽然只是根据程序结论中的不同类型的文字（负/正）去化简逻辑程序，但两者的适用性并不完全一致。从本质上来说，在逻辑程序中删去正负文字的影响是不一样的。具体的原因可以使用Lin和Zhao(2002)提出的环理论进行分析。首先，本节给出以下命题：

命题 4.2 给定ASP逻辑程序P，文字集L是P的一个程序结论，那么有以下两个结论：

- 一个原子集S是P的回答集当且仅当S是$tr_n(P, L)$的一个回答集；
- 一个原子集S是P的回答集可以推出$S \backslash L$是$tr_p(P, L)$的一个回答集，但反之不然。

证明： 对于命题4.2中的第一个结论：

($\Rightarrow$) 令E是P中的一个环，但E不一定能保持也是$tr_n(P,L)$中的一个环，因为$tr_n(P,L)$中可能删掉某些规则导致环E被破坏。然而$tr_n(P,L)$的回答集仍然能满足环E的环公式。这是因为$tr_n(P,L)$删去的是程序结论中的负文字及其相关的影响。负文字代表推不出为真的原子，故这些原子p满足：

$$p \in Atoms(r)\text{，且 } r \notin R^-(E,P) \tag{4.13}$$

所以去掉这些相关的文字并不会影响环公式被满足的基本要求。故保证了$tr_n(P,L)$的回答集仍然能满足逻辑程序中所有环的环公式。且由命题4.1可知$tr_n(P,L)$的回答集也必满足P，同时负文字必然不会作为回答集的部分，所以$tr_n(P,L)$的回答集也是P的回答集。

($\Leftarrow$) 根据定义，$tr_n(P,L)$操作一中删去的原子是$p \in body^+(r)$且$\neg p \in L$，故有p为假，删去这样的p不会影响模型和环公式被满足，而$tr_n(P,L)$的操作二中，仅删去$\overline{L}$中的原子，且$S \cap \overline{L} = \emptyset$。所以，$S$也是$tr_n(P,L)$的模型，同时满足其所有环公式。所以$S$也是$tr_n(P,L)$的回答集。所以得到第一个结论的正确性。

对于命题4.2中的第二个结论，E也不一定能保持也是$tr_p(P,L)$中的一个环，因为$tr_p(P,L)$中同样可能会删去某些正体部的原子导致环E被破坏。但由于$tr_p(P,L)$是基于程序结论的正文字进行化简的，会删去这些正文及其影响。而L中的正文字代表能推出为真的原子p，所以存在规则r满足：

$$r \in R^-(E,P)\text{ 且 } p \in Atoms(r) \tag{4.14}$$

即这些正文字可能作为环E的外部支持，故当这些正文字被删除后，求解出来的回答集将不满足环E的环公式。同时，这些正文字是会出现在原程序的回答集中的，故$S \backslash L$作为$tr_p(P,L)$的回答集时并不一定能保证S是原程序的一个回答集，它可能不能满足原程序的所有环公式。 ■

接着，本节给出一个验证命题4.2的例子：

例 4.2 给定ASP逻辑程序P_4如下：

$$a \leftarrow b. \tag{4.15}$$

$$c \leftarrow a. \tag{4.16}$$

$$b \leftarrow c. \tag{4.17}$$

$$c \leftarrow d. \tag{4.18}$$

$$a \leftarrow f. \tag{4.19}$$

$$d \leftarrow not\, e. \tag{4.20}$$

$$e \leftarrow not\, d. \tag{4.21}$$

$$\leftarrow not\, a. \tag{4.22}$$

$$f \leftarrow a. \tag{4.23}$$

使用本节提供的算法可以算得程序结论$L = \{a, f\}$，并可以计算得到：

$$tr_p(P_4, L) = \{c.\, b \leftarrow c.\, c \leftarrow d.\, d \leftarrow not\, e.\, e \leftarrow not\, d.\} \tag{4.24}$$

且有程序P_4的回答集为：

$$\Gamma(P_4) = \{a, b, c, d, f\} \tag{4.25}$$

及$tr_p(P_4, L)$的回答集为：

$$\Gamma(tr_p(P_4, L)) = \{\{b, c, d\},\ \{b, c, e\}\} \tag{4.26}$$

其中$\{a, b, c, d, f\} \setminus L = \{b, c, d\} \neq \{b, c, e\}$。

基于命题4.2的两个结论，可以知道$tr_n(P, L)$比$tr_p(P, L)$的泛用性更好。然而$tr_p(P, L)$使用的程序结论中的正文字才更有意义，因为这部分将是原程序回答集中的内容。所以本文后续将顺着需要满足所有环公式的思路去设计能让$tr_p(P, L)$泛化的操作符。

4.2 基于程序结论的程序化简

本节将给出使用程序结论去分割ASP逻辑程序的方法，提出分割后的结论顶部。并对结论顶部使用$tr_p(P, L)$进行化简。同时，给出一个关于$tr_p(P, L)$具有泛用性的命题结论。

对于一个ASP逻辑程序P及其一个程序结论U，本节将使用U去分割P以得到一个不含U中任何原子的新逻辑程序，再使用$tr_p(P,L)$对新的逻辑程序进行化简，求解回答集，以得到原程序的实际回答集。根据前面的分析可知，$tr_p(P,U)$会删去满足$head(r)\cap U\neq\emptyset$及$body^-(r)\cap U\neq\emptyset$的规则$r$。同时，在剩下的规则中删去出现在正体部中的$U$中原子。由于正体部被破坏，这可能会导致原程序中的某些环被破坏。故可以引用分割集中构造$t_U(P,X)$的思想构造新的环以保证化简后的程序的回答集联合程序结论可以满足原程序的所有环公式。首先，本节给出后续需要用到的相关定义。

定义 4.3 给定ASP逻辑程序P及其程序结论U，定义规则集合$in'_U(P)$如下：

$$in'_U(P)=\{r\in P\mid head(r)\cap U\neq\emptyset,\ Atoms(r)\not\subseteq U\} \tag{4.27}$$

基于$in'_U(P)$定义用于构造新环的结论半环集。

定义 4.4 (结论半环集(Consequence Semi-Loop)) 给ASP定逻辑程序P及其程序结论U，标记P基于U的结论半环集为$CS_U(P)$，且有：

$$CS_U(P)=\{E\mid E\subseteq U,\ E\neq\emptyset,\ R^-(E,P)\subseteq in'_U(P)\} \tag{4.28}$$

有了$CS_U(P)$后，本节给出结论顶部的定义如下：

定义 4.5 (结论顶部(Consequence Top)) 给定ASP逻辑程序P及其程序结论U，标记P基于U的结论顶部为$ct_U(P)$，并由以下四部分组成：

- $P\setminus(b_U(P)\cup out_U(P))$，
- 对于每个$E\in CS_U(P,X)$构造$\{\{x_E\}\cup head(r)\setminus E\leftarrow body(r)\mid r\in in'_U(P)$ 且 $r\in R^-(E,P,X)\}$，
- $\{head(r)\leftarrow x_{E_1},x_{E_2},...,x_{E_t},body(r)\mid r\in out_U(P),\ E_i\in CS_U(P,X)(1\leq i\leq t)$ 且 $body^+(r)\cap E_i\neq\emptyset\}$，
- 对于每个$E\in CS_U(P,X)$构造$\{\leftarrow not\ x_E\}$。

根据$ct_U(P)$的定义，其使用$CS_U(P)$把$b_U(P)$和$out_U(P)$中的符合条件的规则重新构造。为了构造所有可能被破坏的环，$CS_U(P)$中跟强程序分割方法中的$SS_U(P, X)$类似，也考虑U的所有非空子集。因为这时已经没有$bottom$的回答集X去满足某些环了。$CS_U(P)$中只剔除了$R^-(E, P)$中存在规则r满足$Atoms(r) \subseteq U$的E，删去该规则不会影响环公式被满足。因为U中的原子的真值都为真。若这些原子属于某个环L，那么必有L中的所有原子均为真，则必有$Atoms(L) \subseteq U$。故删去这样的规则r不会破坏环，这也是$in'_U(P)$定义的原因。

接着，本节给出使$tr_p(P, L)$具有泛用性的命题结论，并给出相关的证明和例子。注意，这里只考虑只包含正文字的程序结论，所以下面也将程序结论看做一个原子集。

命题 4.3 给定正规逻辑程序P和其程序结论U，一个原子集S是P的一个回答集，当且仅当存在$tr_p(ct_U(P), U)$的一个回答集S^*满足$S \backslash U = S^* \cap Atoms(P)$。

证明： $(\Rightarrow)$ 即证明若S^*为$tr_p(ct_U(P), U)$的回答集，且$S \backslash U = S^* \cap Atoms(P)$，则原子集$S$为$P$的回答集。

根据定理3.1的证明可以知道，在$ct_U(P)$中的第二和第三部分中依旧保证为所有可能被破坏的环构造新的环，并加入第四部分$\{\leftarrow not\ x_E\}$，这也是为了保证被替换成x_E的那些原来的原子的真值都为真，因为它们是属于程序结论U的。

定义P_U如下：

$$P_U = \{r \in P \mid Atoms(r) \subseteq U\} \tag{4.29}$$

则可以知道U是P_U的回答集。此外，令S^*为$tr_p(ct_U(P), U)$的回答集，根据上述对$ct_U(P)$的分析，显然，$S^* \cap Atoms(P)$为$P \backslash P_U$的回答集，所以有$(S^* \cap Atoms(P)) \cup U$为$P$的回答集，所以有$S \backslash U = S^* \cap Atoms(P)$，$S$为$P$的回答集。故充分性得证。

$(\Leftarrow)$ 即证明若S为P的回答集，则存在S^*为$tr_p(ct_U(P), U)$的回答集，且满足$S \backslash U = S^* \cap Atoms(P)$。

$ct_U(P)$的定义可知，$\{x_E \mid E \in CS_U(P)\} = S^* \backslash Atoms(P)$。且$ct_U(P)$把$Atoms(r) \subseteq U$的规则排除掉，并以$x_E$替代$U$中的原子，所以可以得到$S^* \cap Atoms(P)$为$P \backslash P_U$的回答集。又$U$为$P_U$的回答集，故有$S^* \cap Atoms(P) \cup U$为$P$的回答集。故$S = S^* \cap Atoms(P) \cup U$，所以满足$S \backslash U = S^* \cap Atoms(P)$。故必要性得证。 ■

在计算复杂性方面，$tr_p(ct_U(P), U)$同样会因为$CS_U(P)$而引入新原子，这里也存在指数爆炸的最坏情况。但类似分割理论中的分析，只要通过限制$|in'_U(P)|$的大小来限制$|CS_U(P)|$的大小，则可以有效地把$tr_p(ct_U(P), U)$的计算变得可以接受。

以下通过一个例子直观地展示命题4.3的效果。

例 4.3 继续使用例4.2中的逻辑程序，令$U = \{a, f\}$，则根据定义可以计算得到$CS_U(P) = \{\{a, f\}\}$，并有：

$$\begin{aligned} ct_U(P) = \{b \leftarrow c.\ c \leftarrow d.\ d \leftarrow not\ e.\ e \leftarrow not\ d.\ \leftarrow a. \\ x_{\{a,f\}} \leftarrow b.\ c \leftarrow x_{\{a,f\}},\ a.\ \leftarrow not\ x_{\{a,f\}}.\} \end{aligned} \tag{4.30}$$

则有：

$$\begin{aligned} tr_p(ct_U(P), U) = \{b \leftarrow c.\ c \leftarrow d.\ d \leftarrow not\ e.\ e \leftarrow not\ d. \\ x_{\{a,f\}} \leftarrow b.\ c \leftarrow x_{\{a,f\}}.\ \leftarrow not\ x_{\{a,f\}}.\} \end{aligned} \tag{4.31}$$

通过ASP求解器可以计算得到：

$$\Gamma(tr_p(ct_U(P), U) = \{b, c, d, x_{\{a,f\}}\} \tag{4.32}$$

则有：

$$(\Gamma(tr_p(ct_U(P), U)) \cap Atoms(P)) \cap U = \{a, b, c, d, f\} \tag{4.33}$$

结果等于原程序的回答集。

4.3 本章小结

本章提出了使用程序结论作为分割集去划分ASP逻辑程序，进而达到化简的效果。实际上是从ASP逻辑程序中划分出已经确定真值的部分，并对剩下的部分构造相应的环，以确保最终计算得到的模型能满足原程序中的所有环公式，进而确定为原程序的回答集。

程序化简是ASP逻辑程序中对分割集的一个典型应用，提取了“分割出确定部分造成的影响”的核心思想。

第 5 章　实验与分析

本章涉及两个实验：程序分割实验和程序化简实验。本章将介绍这两个实验的实验环境和实验设计，最后给出实验数据，并对其进行分析。实验数据的结果验证了程序分割方法和程序化简都能为求解ASP逻辑程序带来提速的效果。

5.1　实验环境

本节给出程序分割实验和程序化简实验所用到的实验工具和测例内容。两个实验的实验工具和使用的测例都是一致的。

5.1.1　实验工具

两个实验在词法分析和语法分析上都分别使用flex和bison[2]，在例化和求解方面都使用gringo和clasp[3]。而本文所提出的新程序分割方法和程序化简的操作过程均由C/C++代码实现。具体的实验工具内容和环境如表5.1：

表 5.1: 实验环境

项目	内容
处理器	AMD A10-5800K 3.8GHz
内存	3.3GB
操作系统	Linux Ubuntu 13.10LTS 64bit
词法分析工具	flex 2.5.35
语法分析工具	bison 2.4.1
编程语言	C/C++
例化工具	gringo 3.0.5
ASP求解器	clasp 2.1.3

其中程序分割实验和程序化简实验的代码执行流程图分别如图5.1和图5.2所示，这里不加入原程序直接使用ASP求解器进行求解的步骤。

[2]更多请参考John Levine 所著《Flex与Bison》，O’Reilly出版
[3]更多请参考Gebser等撰写的《A User’s Guide to gringo, clasp, clingo, and iclingo》

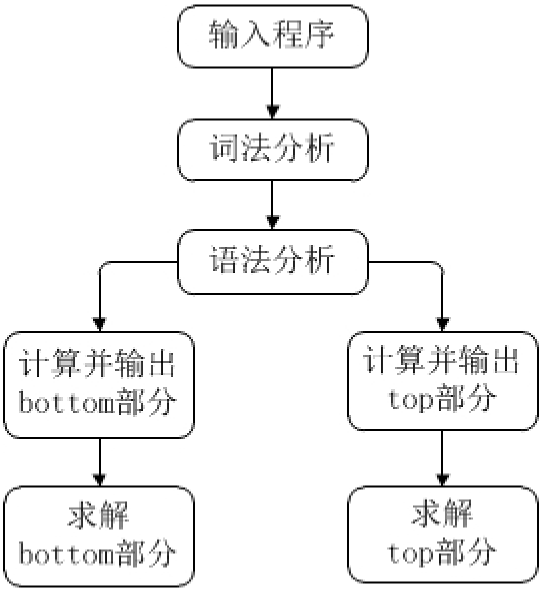

图 5.1: 程序分割实验流程图

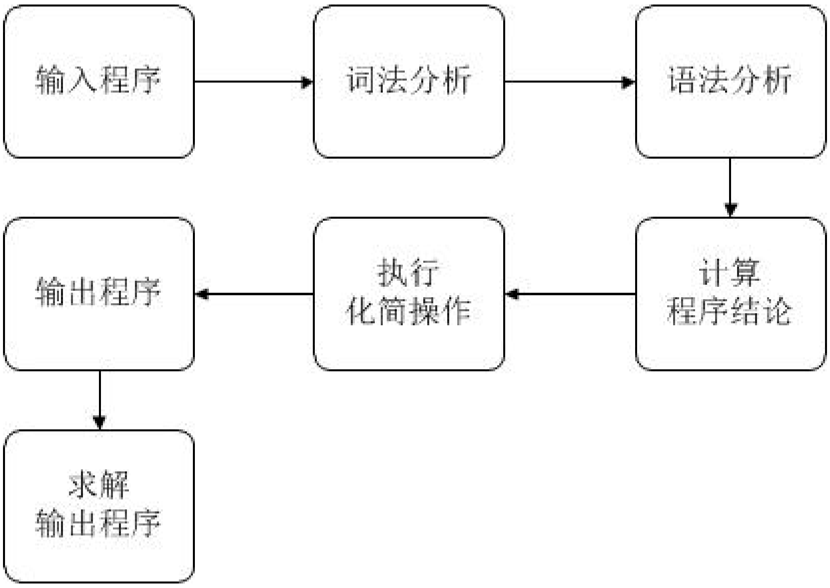

图 5.2: 程序化简实验流程图

5.1.2 实验测例

本章的两个实验均使用哈密顿回路问题(HC: Hamiltonian Circuit/Cycle)作为测例问题，并使用Niemelä在1999年对该问题的编码[51]。

Niemelä给出的求解哈密顿回路问题ASP逻辑程序如下[51]：

$$hc(V1, V2) :- arc(V1, V2), not\ otherroute(V1, V2). \tag{5.1}$$

$$otherroute(V1, V2) : -arc(V1, V2), arc(V1, V3), hc(V1, V3), V2! = V3 \tag{5.2}$$

$$otherroute(V1, V2) : -arc(V1, V2), arc(V3, V2), hc(V3, V2), V1! = V3 \tag{5.3}$$

$$reached(V2) : -arc(V1, V2), hc(V1, V2), reached(V1), not\ initial(V1). \tag{5.4}$$

$$reached(V2) : -arc(V1, V2), hc(V1, V2), initial(V1). \tag{5.5}$$

$$initial(0). \tag{5.6}$$

$$: -vertex(V), not\ reached(V). \tag{5.7}$$

哈密顿回路问题的ASP逻辑程序所对应的事实文件为相应图结构中的点集和边集。即以下两个形式的集合：

$$vertex(0). \tag{5.8}$$

$$arc(0, 1). \tag{5.9}$$

其中$vertex(0)$代表点，$arc(0,1)$代表$vertex(0)$到$vertex(1)$的边。本章的实验选用哈密顿回路问题及Niemelä的编码作为实验例子的原因在于其事实文件中的图结构与例化后的ASP逻辑程序的正依赖图是同构的。这样的性质有利于分析ASP逻辑程序的求解结果和新程序分割方法及程序化简的效果。

具体地，本实验中使用到的事实文件对应的图结构是两个全连通子图A和B通过一条从A中某个节点指向B中某个节点的边，以及一条从B中某个节点指向A中某个节点的边连接而成的，这两条连接边涉及的4个节点都互不相同。本实验称该图为“珍珠图”(Peal Graph)，每个全连通子图称为一颗“珍珠”。同时，珍珠图例化后的ASP逻辑程序符合高内聚低耦合的程序结构特性，这对于大部分的ASP逻辑程序来说，是具有代表性的。此外，每颗“珍珠”都是全连通的，故能体现最坏情况，若实验结果理想，则本文提出的理论对最坏情况也有效。图5.3是2颗基为5的珍珠构成的珍珠图：

5.2 程序分割实验

有了新程序分割方法及新的分割理论后，可以知道有两种方法能求解一个ASP逻辑程序P：

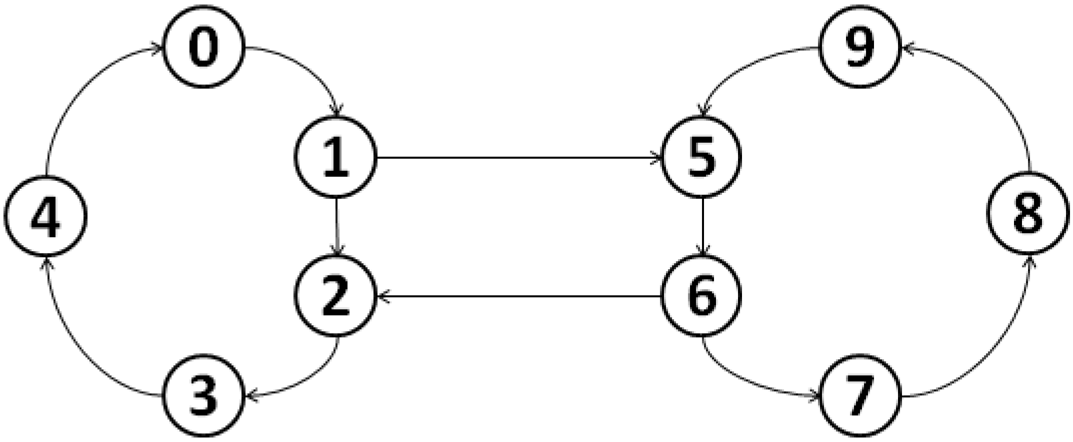

图 5.3: 2颗基为5的珍珠构成的珍珠图

- 通过分割集U来把P划分及构造为两部分：$b_U(P) \cup EC_U(P)$和$e_U(t_U(P,X),X) \cup ECC_U(P,X)$，并通过这两部分的回答集$X$和$Y$来求解得到$P$的回答集；
- 直接求P。

本实验的目的是为了验证本文第3章所提出的新程序分割方法在求解ASP逻辑程序中的正确性和有效性。在此，本节提出两个实验假设：

1. 使用新程序分割方法求解ASP逻辑程序与直接求解所得到的回答集内容是一致的；

2. 使用新程序分割方法求解ASP逻辑程序比直接求解的效率更高。

本实验的内容围绕验证这两个假设而展开。实验结果表明，两种方法求解得到的回答集内容一致，同时，使用新程序分割方法求解出一组方案$\langle X,Y\rangle$进而得到原程序的回答集比直接求解原程序更高效。

5.2.1 程序分割实验设计

实验的测例和结构图已给出，而对于分割集U的选择方面，本实验使用整个右子图中的节点对应的原子组成分割集U，基于哈密顿回路要求经过图中的所有节点，故这部分原子其实都能被ASP逻辑程序P的每个回答集所满足。此外，这样的分割集U能保证$|Atoms(b_U(P))\backslash U| = 3$。这两点性质跟第3章中分析底部计算复杂性时提出的提速方案一致。

本实验的对每个测例的具体执行过程如下：

1. 通过程序自动生成当前珍珠图对应的事实文件；
2. 使用$gringo$通过事实文件例化ASP逻辑程序；
3. 使用$clasp$求解出例化后的ASP逻辑程序的回答集，并统计平均求解时间为t_1；
4. 计算出分割集U；
5. 基于分割集U对例化后的ASP逻辑程序使用新程序分割方法进行划分，得到$b_U(P) \cup EC_U(P)$和$e_U(dt_U(P,X),X) \cup ECC_U(P,X)$两部分逻辑程序；
6. 使用$clasp$分别对$b_U(P) \cup EC_U(P)$和$e_U(dt_U(P,X),X) \cup ECC_U(P,X)$求解回答集，并分别统计两者的平均求解时间，记为$t_b$和$t_t$。

对于每个实验测例，本实验首先验证两种方法求解出来的回答集是否相同，以检验新程序分割方法求解ASP逻辑程序的正确性，然后比较t_1和$t_b + t_t$的大小，以判断哪种ASP逻辑程序求解方式更为高效。其中t_1等代表的统计平均时间是通过求解指定数目的回答集（这个可以通过$clasp$命令参数设置）除以求解这些回答集的总耗时所得到的。

5.2.2 程序分割实验数据分析

本实验中使用的珍珠图包含2颗珍珠，每个珍珠的基的范围是10到50，同一个珍珠图里的每颗珍珠的基是相等的。对于每个测例，根据其对应珍珠图中包含的珍珠数量k和每个珍珠的基N，将该测例记为“k-N”，如：一个例化后的ASP逻辑程序对应的珍珠图里包含2颗珍珠，每颗珍珠的基为10，则将此测例记为“2-10”。

本实验对每个测例执行上述的操作过程，并将统计得到的求解耗时数据整理如表5.2所示。

表 5.2: 两种求解方法的求解耗时对比（单位为：秒）

Benchmark	bottom	top	bottom+top	whole	match
2-10	0.01	0.02	0.03	0.03	true
2-15	0.03	0.16	0.19	0.53	true
2-20	0.03	0.68	0.71	1.84	true
2-25	0.07	2.25	2.32	5.56	true
2-30	0.15	5.66	5.81	14.07	true
2-35	0.23	12.61	12.84	27.16	true
2-40	0.32	23.61	23.93	52.53	true
2-45	0.59	48.73	49.32	106.93	true
2-50	0.75	80.95	81.70	171.94	true

表5.2中的“Benchmark”一列记录的是各个测例的记号，“bottom”一列记录的是求解$b_U(P) \cup EC_U(P)$部分的耗时，“top”一列记录的是求解$e_U(dt_U(P,X),X) \cup ECC_U(P,X)$部分的耗时，“bottom+top”一列记录的是“bottom”和“top”两部分的耗时总和，“whole”一列记录的是直接求解原ASP逻辑程序的耗时，“match”一列记录的是两种方法求解得到的回答集的内容是否一致。

根据表5.2中的数据及两种方法的回答集求解结果对比可以得到以下观察结果：

- 使用新程序分割方法对ASP逻辑程序进行划分后求解得到的回答集与直接求解原程序得到的回答集的内容一致；
- 使用新程序分割方法对ASP逻辑程序进行划分后求解的耗时比直接求解原程序要少；
- 随着ASP逻辑程序的规模的增大，使用新程序分割方法求解ASP逻辑程序的速度稳定在直接求解的速度的2-3倍之间。

本实验通过比较两种方法求解得到的回答集内容的一致性，验证了使用新程序分割方法求解ASP逻辑程序的正确性。此外，本实验通过表5.2中的数据验证了使用新程序分割方法求解ASP逻辑成的有效性。这些结果验证了本节开始时所给

出的两个假设的正确性。更进一步地，可以知道新程序分割方法能更为高效地求解这类程序结构上高内聚低耦合的问题。

5.3 程序化简实验

本实验的目的是为了验证本文第4章基于分割集思想而提出的程序化简方法的正确性和有效性。跟上一节的程序分割实验类似，本实验基于程序化简方法，使用程序结论对原ASP逻辑程序进行化简，再求解化简后的逻辑程序，将这个过程的耗时与直接求解ASP逻辑程序的耗时进行对比，并比较两种方法下求解的回答集内容是否一致。

这里，本实验根据实验目的，提出两个假设：

1. 对ASP逻辑程序使用程序结论进行程序化简后得到的新程序的回答集联合程序结论可以得到原程序的回答集；
2. 求解使用程序结论进行程序化简后的新逻辑程序比直接求解原程序要更为高效。

本实验基于上述两个假设开展验证实验。最终的实验结果验证了这两个假设的正确性。

5.3.1 程序化简实验设计

本实验对每个测例具体的实验步骤为：

1. 直接使用clasp求解例化后的ASP逻辑程序，统计其平均求解时间为t_1；
2. 通过公式2.20计算出例化后的ASP逻辑程序的程序结论U；
3. 使用程序结论U对例化后的ASP逻辑程序执行程序化简操作，得到新的ASP逻辑程序；
4. 使用clasp新的ASP逻辑程序进行求解，并统计其平均求解时间为t_2。

本实验首先对每个测例判断程序化简后的ASP逻辑程序的回答集联合程序结论U得到的结果是否与原程序的回答集内容一致，以验证命题4.3的正确性。同时，对比上述步骤中统计的t_1和t_2的大小，以确定哪种方法能更高效地求解出ASP逻辑程序的回答集。

5.3.2 程序化简实验数据分析

本实验使用的珍珠图与程序分割实验中的一致，一共统计了9个测例的实验数据。对每个测例执行5.3.1中的实验步骤，并统计进行程序化简前后的其平均求解时间，同时对比使用程序化简方法得到的回答集与原程序的回答集的内容是否一致。具体的数据整理如表5.3。

表 5.3: 程序化简前后的求解耗时对比（单位为：秒）

Benchmark	original	simplifying	match
2-10	0.03	0.03	true
2-15	0.53	0.38	true
2-20	1.84	1.40	true
2-25	5.56	4.76	true
2-30	14.07	13.12	true
2-35	27.16	22.55	true
2-40	52.53	46.03	true
2-45	106.93	98.42	true
2-50	171.94	164.35	true

表5.3中的“Benchmark”一列记录的是各个测例的记号，“original”一列记录的是各个测例直接使用clasp求解回答集的平均耗时，“simplifying”一列记录的是进行程序化简后的ASP逻辑程序的求解回答集的平均耗时，“match”一列记录的是两种方法求解得到的回答集的内容是否一致。

对于表5.3提供的数据，本节可以归纳得出以下的观察结果：

- 求解程序化简后的ASP逻辑程序得到的回答集联合程序结论U得到的回答集与直接求解原程序的回答集内容一致；

- 基于程序结论使用程序化简方法求解ASP逻辑程序的耗时比直接求解ASP逻辑程序要少，但未算得上本质上的提速。

通过上述的观察结果可以知道，本实验开始时给出的两个假设成立。观察结果中的第一点验证了使用程序结论对ASP逻辑程序进行程序化简的正确性。观察结果中的第二点从实验层面给予了基于程序结论的程序化简方法的有效性一个正面的支持。更进一步地，可以知道程序化简方法能更高效地求解这类程序结构上高内聚低耦合的问题。

5.4 本章小结

本章通过程序分割实验和程序化简实验的结果分别验证了第3章和第4章所提出的新程序分割方法和基于分割集思想的程序化简方法的正确性和有效性。为新的分割理论的有效性及其在程序化简方面的可用性提供了正面的支持。

第 6 章　总结与展望

本章将对本文提出的新分割理论及其在程序化简上的应用效果进行总结，并基于ASP领域当前的发展情况给出了新分割理论在后续发展中的若干可尝试方向，指出新分割理论对ASP领域的意义。

6.1　研究总结

首先，本文提出了新分割理论，即可以使用任意原子集构成分割集，通过分割集把逻辑程序划分为bottom和top两部分。同时，根据Lin和Zhao的环理论重新定义top部分以保证最终求解的回答集能满足原程序的所有环公式，继而通过证明原程序的回答集可以通过求解bottom和top两部分的回答集得到，此即为本文提出的新程序分割方法及新分割理论的内涵。新的分割理论扩展了Lifschitz和Turner的分割理论，使得分割集不再有定义上的限制，而是可以由任意原子集构成。此外，本文还分析了如何选择分割集能有效地提高新程序分割方法的效率。

然后，本文基于分割集的思想提出了使用程序结论来化简程序的方法。同样地，程序化简的方法是通过程序结论把原程序分割为两部分，类似于新程序分割方法中重新定义的top，程序化简方法基于程序结论为原程序定义了一个程序结论顶部，以确保去掉与程序结论有关的规则后所求解得到的回答集联合程序结论能满足原程序中的所有环公式。

最后，本文进行了两个实验，分别是程序分割实验和程序化简实验，这两个实验都使用了哈密顿回路问题作为测例。程序分割实验的结果验证了新程序分割方法的正确性，同时，还可以知道新程序分割方法能更为高效地求解ASP逻辑程序。而程序化简实验，也验证了基于程序结论的程序化简方法的正确性，并通过实验结果正面支持了程序化简方法比直接求解原程序更为高效。

6.2 研究展望

本节将给出本文的不足以及相关的后续工作方向，同时指出新分割理论在ASP领域的其他方面的应用。

6.2.1 不足及后续工作

本文存在不足的地方主要是没有给出通用的分割集计算方法。这个不足可以作为后续相关工作的重点研究问题。而目前来看，分割集的确定跟程序结构有关，但并非无法找出共性，可以从良序模型语义和环理论出发进行研究。

6.2.2 ASP领域的其他方向应用

本文只在程序化简中应用了新分割理论。事实上，ASP领域还有很多方面可以应用新分割理论，主要可以考虑以下几个方面：

- 增量式ASP求解器；
- 遗忘处理(Forgetting)[52]；
- 嵌套式表达。

上述方向中的增量式ASP求解和嵌套式表达都引入过Lifschitz和Turner的分割集思想，故本文的新分割集理论也同样地可以应用在其上面。而遗忘处理的基本思想主要也与分割集理论吻合，故也是一个可尝试的研究方向。

6.3 本章小结

本章总结了本文研究工作的成果和实验结果，同时指出了本文一些不足及基于不足的后续相关研究工作方向，最后给出了新分割理论在ASP领域的其他可能应用场景。

参考文献

[1] Michael Gelfond Chitta Baral. Logic programming and knowledge representation. *Journal of Logic Programming*, 19(94):73–148, 1994.

[2] 吉建民. 提高*ASP* 效率的若干途径及服务机器人上应用. PhD thesis, 合肥: 中国科学技术大学, 2010.

[3] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*, volume 1. Elsevier, 2008.

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

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

[6] Vladimir Lifschitz. Foundations of logic programming. *Principles of knowledge representation*, 3:69–127, 1996.

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

[8] Ehud Shapiro. Concurrent prolog: A progress report. *Fundamentals of Artificial Intelligence*, 19(8):44 – 58, 1986.

[9] Keith L Clark and Sten-Ake Tärnlund. *Logic programming*. Number 16. Academic Press New York, 1982.

[10] Luís Moniz Pereira Carlos Viegas Damásio. Monotonic and residuated logic programs. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 748–759, 2001.

[11] David J Israel. What's wrong with non-monotonic logic? In *AAAI*, volume 80, pages 99–101, 1980.

[12] John McCarthy. Circumscription—a form of nonmonotonic reasoning. *& &*, 1987.

[13] Keith L Clark. Negation as failure. In *Logic and data bases*, pages 293–322. Springer, 1978.

[14] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

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

[16] Robert G Jeroslow and Jinchang Wang. Solving propositional satisfiability problems. *Annals of mathematics and Artificial Intelligence*, 1(1-4):167–187, 1990.

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

[18] Ilkka Niemela, Patrik Simons, and Tommi Syrjanen. Smodels: a system for answer set programming. *arXiv preprint cs/0003033*, 2000.

[19] Fangzhen Lin and Yuting Zhao. Assat: Computing answer sets of a logic program by sat solvers. *Artificial Intelligence*, 157(1):115–137, 2004.

[20] Yuliya Lierler Yuliya Lierler. cmodels - sat-based disjunctive answer set solver. *of Lecture Notes in Computer Science*, pages 447–451, 2005.

[21] Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An a-prolog decision support system for the space shuttle. In *Practical Aspects of Declarative Languages*, pages 169–183. Springer, 2001.

[22] Chitta Baral Phan Huy Tu, Tran Cao Son. Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming. *Theory and Practice of Logic Programming*, 7(7):377–450, 2006.

[23] Henri Prade Roberto Confalonieri. Using possibilistic logic for modeling qualitative decision: Answer set programming algorithms. *International Journal of Approximate Reasoning*, 55(2):711 – 738, 2014.

[24] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *ICLP*, volume 94, pages 23–37, 1994.

[25] Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner. Modular nonmonotonic logic programming revisited. In *Logic Programming*, pages 145–159. Springer, 2009.

[26] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. Engineering an incremental asp solver. In *Logic Programming*, pages 190–205. Springer, 2008.

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

[28] Paolo Ferraris, Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. Symmetric splitting in the general theory of stable models. In *IJCAI*, volume 9, pages 797–803, 2009.

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

[30] George F Luger. *Artificial Intelligence:Structures and Strategies for Complex*

Problem Solving. Pearson Education, 2003.

[31] Michael Gelfond. Logic programming. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1990.

[32] Dov M Gabbay and Heinrich Wansing. *What is negation?* Springer, 1999.

[33] Gabriel Nuchelmans. Theories of the proposition. 1973.

[34] Raymond Reiter. *On closed world data bases*. Springer, 1978.

[35] Allen Van Gelder. Negation as failure using tight derivations for general logic programs. *The Journal of Logic Programming*, 6(1):109–133, 1989.

[36] Vladimir Lifschitz. What is answer set programming?. In *AAAI*, volume 8, pages 1594–1597, 2008.

[37] Piero Bonatti, Francesco Calimeri, Nicola Leone, and Francesco Ricca. Answer set programming. In *A 25-year perspective on logic programming*, pages 159–182. Springer-Verlag, 2010.

[38] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1):39–54, 2002.

[39] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. ios press, 2009.

[40] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.

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

[42] Chin-Liang Chang and Richard C. T. Lee. *Symbolic logic and mechanical theorem proving*. Computer science classics. Academic Press, 1973.

[43] Hantao Zhang and Mark E Stickel. An efficient algorithm for unit propagation. In *In Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics (AI-MATH’96), Fort Lauderdale (Florida USA*. Citeseer, 1996.

[44] Allen Van Gelder, Kenneth A Ross, and John S Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3):619–649, 1991.

[45] Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs. In *Logic Programming*, pages 451–465. Springer, 2003.

[46] Bengt Aspvall, Michael F Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information*

Processing Letters, 8(3):121–123, 1979.

[47] Fangzhen Lin and Yuting Zhao. Assat: Computing answer sets of a logic program by sat solvers. In *Sixteenth AAAI Conference on Artificial Intelligence*, 2002.

[48] Martin Gebser, Joohyung Lee, and Yuliya Lierler. On elementary loops of logic programs. *Theory and Practice of Logic Programming*, 11(06):953–988, 2011.

[49] Jianmin Ji, Hai Wan, Peng Xiao, Ziwei Huo, and Zhanhao Xiao. Elementary loops revisited. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[50] Jianmin Ji, Hai Wan, and Peng Xiao. On elementary loops and proper loops for disjunctive logic programs. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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

[52] Yisong Wang, Kewen Wang, and Mingyi Zhang. Forgetting for answer set programs revisited. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1162–1168. AAAI Press, 2013.

在学期间论文发表情况

致　谢

首先要感谢我的导师XXXX老师。自大四与XXXX老师接触以来，他在

首先感谢我的指导老师XXXX老师，对于本文的选题和撰写方面，万海老师都给出建设性的意见，在万老师的悉心指导下，让我在知识表示和限定理论学习中受益匪浅，在此，对老师无私的指导和不厌其烦的帮助表示最衷心的感谢！

其次，我要感谢实验室的萧展豪同学，他给出了翻译理论的证明，奠定了翻译算法的基础，一起共同学习讨论的日子，对我来说是一段美好的经历。我还要感谢林尚武同学，感谢他对于本文格式的检查与帮助。同时感谢硕士期间帮助过我的老师和同学们，你们丰富了我的硕士生涯。

最后，我要感谢我的父母，感谢你们一直以来在我背后的支持！