软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(9):2250−2261 [doi: 10.13328/j.cnki.jos.004734]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

半扩展规则下分解的定理证明方法*

张立明[1,2,3], 欧阳丹彤[1,2], 赵 毅[3]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

[3](吉林大学 电子科学与工程学院,吉林 长春 130012)

通讯作者: 欧阳丹彤, E-mail: ouyangdantong@163.com

摘 要: 基于扩展规则的定理证明方法在一定意义上是与归结原理对偶的方法,通过子句集能否推导出所有极大项来判定可满足性.IER(improved extension rule)算法是不完备的算法,在判定子句集子空间不可满足时,并不能判定子句集的满足性,算法还需重新调用 ER(extension rule)算法,降低了算法的求解效率.通过对子句集的极大项空间的研究,给出了子句集的极大项空间分解后子空间的求解方法.通过对扩展规则的研究,给出了极大项部分空间可满足性判定方法 PSER(partial semi-extension rule).在 IER 算法判定子空间不可满足时,可以调用 PSER 算法判定子空间对应的补空间的可满足性,从而得到子句集的可满足性,避免了不能判定极大项子空间可满足性时需重新调用 ER 算法的缺点,使得 IER 算法更完备.在此基础上,还提出 DPSER(degree partial semi-extension rule)定理证明方法.实验结果表明:所提出的 DPSER 和 IPSER 的执行效率较基于归结的有向归结算法 DR、IER 及 NER 算法有明显的提高.

关键词: 定理证明;命题逻辑;扩展规则;可满足性问题

中图法分类号: TP181

中文引用格式: 张立明,欧阳丹彤,赵毅.半扩展规则下分解的定理证明方法.软件学报,2015,26(9):2250−2261. http://www.jos.org.cn/1000-9825/4734.htm

英文引用格式: Zhang LM, Ouyang DT, Zhao Y. Theorem proving decomposition algorithm based on semi-extension rule. Ruan Jian Xue Bao/Journal of Software, 2015,26(9):2250−2261 (in Chinese). http://www.jos.org.cn/1000-9825/4734.htm

Theorem Proving Decomposition Algorithm Based on Semi-Extension Rule

ZHANG Li-Ming[1,2,3], OUYANG Dan-Tong[1,2], Zhao Yi[3]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

[3](College of Electronic Science & Engineering, Jilin University, Changchun 130012, China)

Abstract: The extension rule based theorem proving methods are inverse methods to resolution in a sense that they check the satisfiability by determining whether all the maximum terms of the clause set can be deduced. IER (improved extension rule) algorithm is incomplete as it cannot determine the satisfiability of the clause set when the subspace of the clause set is unsatisfiable. In this condition, calling ER (extension rule) algorithms is still needed. After a thorough investigation on the maximum terms space of the clause set, this paper develops a decomposition method for decomposing the maximum terms space of the clause set. The study on extension rule also results in the PSER (partial semi-extension rule) algorithm for determining the satisfiability of a partial space of the maximum terms. When the IER determines the subspace is unsatisfiable, PSER can be used to determine the satisfiability of the complementary space, thereby, the satisfiability of the clause set can be obtained. Based on the above progress, this paper further introduces DPSER (degree

* 基金项目: 国家自然科学基金(61133011, 61272208, 61402196, 61003101, 61170092); 吉林省科技发展计划(20101501, 20140520067JH); 中国博士后科学基金(2013M541302)

收稿时间: 2014-04-12; 定稿时间: 2014-09-19

partial semi-extension rule) theorem proving method. Results show that the proposed DPSER and IPSER outperform both the directional resolution algorithm DR and the extension rule based algorithms IER and NER.

Key words: theorem proving; propositional logic; extension rule; satisfiability problem

20 世纪 50 年代以来,定理证明研究的兴起,拉开了人工智能研究的序幕,并且一直是人工智能的核心技术和最具挑战性的研究方向之一,被公认为硕果累累[1−3]的研究分支之一.自动推理的发展,深刻地影响着人工智能以及计算机科学的其他研究方向,无论是在硬件校验还是在软件验证中,自动推理都是最重要的方法.IBM, Intel,AMD 等国际知名硬件厂商普遍采用基于自动推理的硬件校验技术.在模型诊断领域,Grastien 等人[4,5]也将离散事件系统的基于模型的诊断问题转换为定理证明问题进行求解.在智能规划研究领域中, SATPLAN 和 BLACKBOX 等智能规划系统通过将规划问题转换为自动推理问题,在近年来的国际智能规划竞赛中获得了多项冠军[6,7].在验证领域中,PVS[8]验证系统已经应用到许多领域中,包括宇宙飞船部分控制系统规格说明的验证和建筑的调度及诊断算法的验证.Bruno 等人[9]开发的 Proverif 使用基于归结方法进行逆向搜索的安全协议模型检测器,并且不限制并行会话的数量;David 等人开发的 OFMC 和 Yannick 等人开发的 CL-Atse[10]将归结方法和约束求解方法相结合,用来对工业级安全协议进行高效验证;Alessandro 等人开发的 SATMC[11]将 SAT 方法与规划图相结合用来验证工业级安全协议,同时可以处理规模较大的问题.NASA 也使用定理证明保证由高级规格说明和自动生成实现的航天软件的功能正确性和安全性.定理证明问题研究领域的成功,直接或间接促进了其他研究领域的发展.对于很多 NP 完全问题,一般都存在定理证明问题的多项式归约,所以基于定理证明的求解方法都会有显著效果.

定理证明中常用的方法主要有基于归结的方法[12−14]、自然演绎法[15]、基于表的方法[16]等.基于归结原理的方法以空子句不可满足为基础,如果一个子句集能够推导出空子句,就说该子句集不可满足;基于扩展规则[17]的方法则以所有的极大项组成的子句集不可满足为基础,如果一个子句集能够推导出所有极大项组成的子句集,就说该子句集不可满足.因此,基于扩展规则的定理证明方法在一定意义上是与归结原理对偶的方法.基于扩展规则的定理证明方法已经得到了许多相关研究人员的重视,并取得了较好的成果:将扩展规则改进应用到模态逻辑的自动定理证明中[18];将扩展规则方法用于模型计数(model counting)问题[19],结合扩展规则的知识编译方法[20];对基于扩展规则方法进行改进,提出 NER,MCN-MO-KCER 和 IBOHMH_IER 等系列算法[21−23]; Murray[24]将扩展规则方法用于基于知识编译的目标语言生成中,并且取得了较好的结果.

文献[17]中,基于扩展规则算法的思想是:通过子句集能否推导出所有极大项来判定可满足性,并采用容斥原理计算子句集可扩展的极大项的个数,并将其与所有极大项的个数进行比较来判断子句集是否可满足,这样做解决了指数级的空间需求问题.算法中虽然利用容斥原理计算子句集扩展出的极大项个数,减小了算法的空间需求,但是其时间复杂性仍然是指数级的,严重影响该类算法的执行效率.为了提高效率,文献[17]中同时给出了改进的扩展规则 IER 方法,思想是:先运行一个高效但并不完备的算法,希望多数问题能被这个高效的算法所解决;对于解决不了的问题,再重新调用原来完备的算法.高效的方法先搜索该空间的一个子空间,若在这个子空间上存在极大项不能由子句集扩展得到,则显然在整个空间上也存在极大项不能被扩展出来,于是判定子句集可满足;否则,如果子空间所有极大项都被扩展出来,此时不能判定子句集的满足性.因为在子空间的补空间的极大项是否能都被扩展出来还无法判定,所以这时再调用原来的 ER 算法.

为了避免 IER 算法在不能判定子句集的满足性时需重新调用原来 ER 算法的缺点,本文给出了子句集的极大项空间分解后子空间的求解方法,并给出了极大项部分空间可满足性的判定方法 PSER.当使用 IER 算法判定得到子空间上所有极大项都被扩展出来,即,不能判定子句集的可满足性时,使用 PSER 算法判定子空间对应的补空间的可满足性.若补空间中有极大项不能由子句集扩展得到,则子句集可满足;若补空间的所有极大项都能由子句集扩展得到,则子句集不可满足.本文提出的 PSER 算法使 IER 算法完备,满足了先在子空间判定满足性,如得不到判定结果,然后在补空间上判定,最后得到子句集的可满足性.

在此基础上,我们还提出了 DPSER 算法,将 MOM 启发式策略用于限定搜索子空间的子句 C 的选择,利用子句集的信息选择子句 C,可以得到更适合该问题的子空间.使用 PSER 方法对选定的子空间进行判定,对可满足

的问题可以直接返回结果“SAT”,不可满足的问题再调用 PSER 算法对补空间进行判断,从而减少了问题的判定时间.

1 基于扩展规则的方法

首先给出扩展规则的相关定义.

定义 1[17]. 对于一个子句集 Y,$M(|M|=m)$是其中所有原子的集合.对于一个子句 C,$C\in Y$,$D=\{C\vee A, C\vee\neg A|A\in M$,且 A 和$\neg A$ 都不在 C 中出现$\}$.把从 C 到 D 的推导过程叫做扩展规则,把 D 称做 C 应用扩展规则的结果.

定理 1[17]. 子句 C 和它扩展后的结果 D 是等价的.

该定理保证了应用扩展规则后的子句集和原子句集等价,因此,扩展规则可以被看作是一条推理规则.

定义 2[25]. 一个非重言式子句是集合 M 上的极大项当且仅当它包含集合 M 上的所有原子或其否定.

定理 2[17]. 给定一个子句集 Y,它其中所有原子的集合是 $M(|M|=m)$,若 Y 中的子句都是 M 上的极大项,则子句集 Y 不可满足当且仅当 Y 含有 2^m 个互不相同的子句.

如果要判定一个子句集的可满足性,首先用扩展规则把原来的子句集扩展成和它等价的极大项组成的子句集合,再由定理 2:若扩展后的子句集里有 2^m 个互不相同的子句,则子句集不可满足;否则,子句集可满足.这个过程叫做基于扩展规则的定理证明方法.

例 1:为了证明子句集 $Y=\{A\vee B,\neg B,\neg A\vee B\}$是不可满足的,下面分别给出基于归结和扩展规则方法求解过程:

(1) 归结方法.$A\vee B$ 和$\neg A\vee B$ 归结出 B,然后再和$\neg B$ 归结出空子句,得子句集不可满足.

(2) 扩展规则方法.用$\neg B$ 扩展出 $A\vee\neg B$ 和$\neg A\vee\neg B$,与原有的 $A\vee B$ 和$\neg A\vee B$ 共同构成由 A 和 B 两个原子组成的所有极大项的集合,从而子句集不可满足.

显然,由 m 个原子组成的所有极大项的集合一定包含 2^m 个子句,所以只需计算一个子句集能够扩展出的极大项的数目就能判定它的可满足性,而计算一个子句集能够扩展出的极大项数目可以使用如下包含排斥原理:

定理 3[25]**(包含排斥原理)**. 集合 $A_1,A_2,\ldots,A_n$ 并集的元素个数可以用如下公式计算:

$$|A_1\cup A_2\cup\ldots\cup A_n|=\sum_{i=1}^{n}|A_i|-\sum_{1\le i<j\le n}|A_i\cap A_j|+\ldots+(-1)^{n+1}|A_1\cap A_2\cap\ldots\cap A_n|.$$

定理 4[17]. 两个子句扩展出的极大项集合不含交集当且仅当这两个子句含有互补对.

给定子句集 $Y=\{C_1,C_2,\ldots,C_n\}$,M 是其原子集合,且$|M|=m$,记 P_i 为 C_i 扩展出的所有极大项的集合,$i=1,\ldots,n$.令 S 为 Y 能扩展出的所有极大项集合的元素个数,则:

$$S=|P_1\cup P_2\cup\ldots\cup P_n|.$$

由包含排斥原理易得:

$$S=\sum_{i=1}^{n}|P_i|-\sum_{1\le i<j\le n}|P_i\cap P_j|+\sum_{1\le i<j<l\le n}|P_i\cap P_j\cap P_l|+\ldots+(-1)^{n+1}|P_1\cap P_2\cap\ldots\cap P_n| \tag{1}$$

其中,

$$|P_i|=2^{m-|C_i|},|P_i\cap P_j|=\begin{cases}0, & \text{在}C_i\cup C_j\text{中含有互补对}\\ 2^{m-|C_i\cup C_j|}, & \text{在}C_i\cup C_j\text{中不含有互补对}\end{cases}.$$

C_i 中已含有$|C_i|$个原子,在扩展极大项时,剩余的 $m-|C_i|$个原子可以为正或为负,所以$|P_i|$扩展出来的极大项数目是 $2^{m-|C_i|}$ 个.根据定理 4,两个子句扩展出的极大项不含交集当且仅当它们含有互补对.因此当两个子句 C_i,C_j 含互补对时,$|P_i\cap P_j|$扩展出来的极大项数目是 0;否则为 $2^{m-|C_i\cup C_j|}$.其余项的计算方法依此类推.

例 2:利用公式(1)判定子句集 $X=\{A\vee\neg B\vee C,A\vee\neg C,\neg A\}$的可满足性.

计算 X 能扩展出的极大项子句数目:$S=2^0+2^1+2^2-0-0-0+0=7$,由于 $7<2^3$,子句集 X 可满足.

一般情况下,ER 算法中使用包含排斥原理计算极大项个数,算法的复杂性是指数级的,效率比较低.如果子句集中任意两个子句都含互补对,即,从第 $n+1$ 项开始所有的项都为 0,则只需要计算公式(1)的前 n 项就可得到子句集的可满足性,此时,ER 算法的效率会很高;当子句集中任意两个子句都含互补对时,使用基于归结的方法

效率比较低,因为有很多归结需要做.因此,基于扩展规则的方法在一定程度上是与基于归结原理对偶的方法.当一个子句集中的任意两个子句都不含互补对,ER 算法需要计算所有的 2^n-1 项才能判定子句集的可满足性,此时,ER 算法效率低下;然而,基于归结的方法不需要任何归结就可以判定它的可满足性.

文献[17]中还给出一种更高效的算法 IER,在完备的算法前先运行一种高效的但并不完备的算法,希望多数的问题能够被这种高效的算法解决,如果解决不了,再重新调用原来完备的算法.在 IER 算法中搜索子句集空间的一个子空间,如果在这个子空间上就有极大项不能被扩展出来,则在整个空间上必有极大项不能被扩展出来,也就是说子句集可满足;否则,在子空间上不能判定该子句集的可满足性,我们再重新调用原来的 ER 算法.下面给出 IER 算法.

令子句集 $Y=\{C_1,C_2,\ldots,C_n\}$,其原子的集合为 M 且$|M|=m$,C 是任意一个 M 上的子句.

Function IER().

1. BEGIN
2. $Y'\leftarrow Y$
3. For each $P\in Y'$
4. If *Complement*(C,P) Then $Y'\leftarrow Y'-P$
5. If $ER(Y')$==SAT
6. Return SAT
7. Else
8. Return $ER(Y)$
9. END

输入的子句 C 相当于一个过滤器,用于限定极大项的子空间,所有与 C 含互补对的子句都被删除.因此,实际的搜索空间也被限定.子句 C 使得算法 IER 非常灵活,如果输入的子句 C 为单元子句 A,那么实际上是在所有 A 为正的极大项组成的空间中进行搜索;如果输入的子句 C 为$\neg A\vee\neg B$,则实际上是在所有 A 为负、B 为负的极大项组成的空间中进行搜索.

上面给出的 IER 算法在不能判定子句集的满足性时,需重新调用原来的 ER 算法,因此影响了算法的效率.此时,我们应对子句集的子空间对应的补空间进行可满足性判定,从而得到子句集的可满足性.如果子句集的补空间是可满足的,则子句集是可满足的;否则,子句集是不可满足的.下面给出了子句集的极大项空间分解后子空间的求解方法和极大项部分空间可满足性的判定方法(PSER).

2 求解部分空间的可满足性

2.1 极大项部分空间求解方法

首先给出子句集的极大项空间分解后子空间的求解方法的相关定义和算法.

定义 3. 对于 $M=\{V_1,V_2,\ldots,V_m\}$,则 M 对应的 2^m 个极大项为$\{\neg V_1\vee\neg V_2\vee\ldots\vee\neg V_{m-1}\vee\neg V_m,\neg V_1\vee\neg V_2\vee\ldots\vee\neg V_{m-1}\vee V_m,\ldots,V_1\vee V_2\vee\ldots\vee V_{m-1}\vee\neg V_m,V_1\vee V_2\vee\ldots\vee V_m\}$,并记每个极大项对应编号为 $mt(1),mt(2),\ldots,mt(2^m-1),mt(2^m)$.

定义 4. 给定子句集 $Y=\{C_1,C_2,\ldots,C_n\}$,M 是其原子集合且$|M|=m$,称 M 的极大项空间为 $MT(M)$,称子句 C 对应的极大项的子空间为 $MTS(C)$,称 $MT(M)-MTS(C)$为子句 C 对应的极大项子空间的补空间,并记为 $MTR(C)$.

为了方便求解极大项子空间,我们调用 *Reorder*(C,Y)算法对子句集的每个子句中原子出现的顺序根据 C 中原子的顺序进行调整,使得在子句 C 中出现的原子全部排在其他原子的左端.

Function Reorder(***C***,***Y***).

1. BEGIN
2. For each $P\in Y$
3. For each $L_i\in C$
4. If $L_i\notin P$ Then $L_i\leftarrow Not(L_i)$

5. $P\leftarrow P-L_i$

6. $P\leftarrow L_i\cup P$

7. END

根据定义 1,对子句 C 应用扩展规则,C 可扩展出 $2^{m-|C|}$个极大项.进而 $MTS(C)=\{mt(si)|si\in\{f+1,f+2,\ldots,f+2^{m-|C|}\}\}$,其中,$f$ 为 $MTS(C)$的第 1 个极大项.进而我们可以得到:

$$MTR(C)=\{mt(ri)|ri\in\{1,2,\ldots,f,f+2^{m-|C|}+1,f+2^{m-|C|}+2,\ldots,2^m\}\},MT(M)=\{mt(i)|i\in\{1,2,\ldots,2^m\}\}.$$

例 3:设 $M=\{V_1,V_2,V_3\},C=V_2\vee\neg V_3$,则 $MTS(C)=\{V_2\vee\neg V_3\vee\neg V_1,V_2\vee\neg V_3\vee V_1\},MTR(C)=\{\neg V_2\vee\neg V_3\vee\neg V_1,\neg V_2\vee\neg V_3\vee V_1,\neg V_2\vee V_3\vee\neg V_1,\neg V_2\vee V_3\vee V_1,V_2\vee V_3\vee\neg V_1,V_2\vee V_3\vee V_1\}$.我们也可以记为 $MTS(C)=\{mt(si)|si\in\{5,6\}\},MTR(C)=\{mt(si)|si\in\{1,2,3,4,7,8\}\}$.

下面给出计算子句 C 对应的第 1 个极大项 $mt(f)$中 f 的算法.

***Function FirstMTS*(*C*,*M*)**.

1. BEGIN

2. $f\leftarrow 1$;

3. For each $C_i\in C$

4. If (C_i 为正文字) $f\leftarrow f+2^{(m-i)}$

5. Return f

6. END

算法 FirstMTS 求出 $MTS(C)$的第 1 个极大项的编号 f,则可以得到 $MTS(C)$和 $MTR(C)$.

2.2 求解部分空间可满足性的方法

为了避开 ER 算法中容斥原理的复杂的求解过程,文献[21]给出了一种基于扩展规则的定理证明算法 NER.该算法按照一定次序判断极大项是否可由子句集中子句扩展得到:若存在极大项不可被扩展,则子句集可满足;否则,若所有极大项均可被扩展,则子句集不可满足.这样无需求出子句集可扩展的极大项个数,而是通过判断极大项是否可由子句集中子句扩展,进而判断子句集的可满足性.在此基础上,我们提出 PSER 算法,通过判断极大项子空间中所有极大项是否可由子句集中子句扩展,进而判断极大项子空间的可满足性.

定义 5. $M=\{V_1,V_2,\ldots,V_m\},m=|M|$.设子句 $C=L_i\vee\ldots\vee L_j\ldots\vee L_d,L_i$ 为 V_i 对应的文字,$1\leqslant i\leqslant j\leqslant d\leqslant m$,称 d 为子句 C 的度.$Y=\{C\vee V_k,C\vee\neg V_k|d\leqslant k\leqslant m\}$,把从 C 到 Y 中元素的推导过程叫做半扩展规则,D 中的元素叫做应用半扩展规则的结果.

命题 1. 根据定义 5,对 C 应用半扩展规则时,剩余的 $m-d$ 个文字可以为正或者为负,所以 C 可半扩展出 2^{m-d}个子句.

例 4:设 $M=\{V_1,V_2,V_3\},C=V_2,m=|M|$,则对子句 C 应用半扩展规则得到的结果为$\{V_2\vee\neg V_3,V_2\vee V_3\}$;对子句 C 应用扩展规则得到的结果为$\{\neg V_1\vee V_2\vee\neg V_3,\neg V_1\vee V_2\vee V_3,V_1\vee V_2\vee\neg V_3,V_1\vee V_2\vee V_3\}$.也就是说:对子句 C 应用半扩展规则,可半扩展出 2^{m-d} 个子句;而对子句 C 应用扩展规则,可扩展出 $2^{m-|C|}$个子句.其中,d 为子句 C 的度,$|C|$为子句的文字的个数.

PSER 算法思想:按照一定次序判断极大项部分空间中极大项是否可由子句集中子句扩展得到:若存在一个极大项不可由子句集中子句扩展得到,则极大项部分空间可满足;若所有极大项都可由子句集中子句扩展得到,则极大项部分空间不可满足.并且在算法判断某个极大项可由子句扩展时,则基于此子句半扩展出的极大项都不必再判断是否可由子句集中子句扩展,因此有效地缩小了判断可被扩展的极大项个数.

命题 2. 设子句 C_1 和 C_2 的度分别为 d_1 和 $d_2,d_1<d_2$.根据命题 1,C_1 和 C_2 可半扩展出的子句是由 $m-d_1$ 和 $m-d_2$ 个原子为正或者为负得到的,所以 C_2 可半扩展出的子句是 C_1 可扩展出子句的子集.

根据命题 2,在判断极大项子句是否可以由子句扩展得到时,要先判断是否可由度较小的子句扩展得到,从而不产生不同子句半扩展出的极大项子句集的交集,同时避免了使用包含排斥原理去计算子句可半扩展子句的个数.为了先判断子句集中度较小子句,我们先对子句集按子句的度由小到大进行排序.对子句集按度的大小

排序后,不仅避免了包含排斥原理计算过程,同时加速了极大项子句的遍历速度,因为度较小的子句可半扩展出的子句个数较多.

***Function PSER*(*CNF*:*Y*,*starti*,*endi*).**

```
1.  BEGIN
2.     i←starti; Y←DegreeSort(Y);
3.     While (i<endi)
4.        BEGIN
5.           T←getMaxTerm(i);
6.     If Expand(Y,T)==False
7.        Then Return SAT;
8.        Else If (d<m) i←i+2^(m−d)−1
9.     i++;
10.       END
11.    Return UNSAT;
12. END
```

在算法 PSER 中,首先,i 从 *strati* 开始循环递增,每循环一次调用函数 *get*Max*Term*,该函数按照一定次序得到对应 Y 的原子集合 M 的第 i 个极大项 T,然后调用函数 *Expand* 判断 T 是否可以由子句集 Y 扩展.

- 如果函数返回 False,则说明当前极大项不可被 Y 扩展,即,极大项部分空间可满足,算法 PSER 返回 SAT;
- 如果第 i 个极大项 T 可被子句扩展,根据命题 1,则从第 i+1 到 $i+2^{m-d}-1$ 个极大项都可由此子句半扩展.继续下一次循环时,从第 $i+2^{m-d}$ 个极大项开始.如果对于所有极大项函数 *Expand* 均返回 True,即,Y 可扩展所有极大项,则极大项部分空间不可满足,算法 PSER 返回 UNSAT.

定理 5[21]. 给定一个子句集 $Y=\{C_1,C_2,\ldots,C_n\}$,M 是其原子的集合且$|M|=m$.关于 M 的一个极大项 $T=L_1\vee L_2\vee\ldots\vee L_m$ 可由子句 $C=L_i\vee\ldots L_j\vee\ldots\vee L_k$,$1\leqslant i\leqslant j\leqslant k\leqslant m$ 扩展当且仅当$\{L_i\ldots,L_j\ldots,L_k\}\subseteq\{L_1,L_2,\ldots,L_m\}$.

根据定理 5,我们给出判断极大项 T 是否可由子句集 Y 扩展的算法 Expand.算法的输入为子句集 Y 和极大项 T.若极大项 T 可以被子句集 Y 扩展,算法返回 True;否则,返回 False.

***Function Expand*(*CNF*:*Y*,*MaxTerm*:*T*).**

```
1.  BEGIN
2.     For each clause C∈Y do
3.     BEGIN
4.        If C⊆T Then
5.           d←max(C);
6.           return True;
9.     END
10.    Return False;
11. END
```

算法的具体流程是:对于给定的极大项 T,遍历子句集 Y 中子句 C,并判断 C 是否包含于 T:若包含关系成立,根据定理 5 知 T 可以由子句 C 扩展,用 d 记录子句的度(d 用来在 PSER 算法中计算 C 可半扩展的极大项个数),算法返回 True;否则,对于任何一个子句 C,上面所说的包含关系不成立,说明 T 不可被 Y 中的任何一个子句扩展得到,即,T 不可由 Y 扩展,算法返回 False.

定理 6. 算法 PSER 是正确的并且是完备的.

证明:

- 正确性.

在算法 PSER 中,若算法返回 SAT,则算法结束前必执行第 7 行,即,函数 *Expand* 返回 False,说明当前的 T 不可由子句集中子句扩展得到.根据上面的分析可知,即,极大项部分空间可满足;若算法返回 UNSAT,则算法必然从第 11 行返回,即,此时已遍历极大项部分空间所有的极大项,且 Expand 均返回 True,说明每个极大项都可由该子句集扩展,即,极大项部分空间不可满足.正确性得证.

- 完备性.

在算法 PSER 中,对于可满足的情况,至少存在一个极大项不可由该子句集中子句扩展得到,则算法 PSER 中变量 i 在增加到 *endi* 前必有一次调用函数 *Expand* 时返回 False,即,PSER 必在第 7 行结束,并返回 SAT;对于不可满足的情况,对于极大项部分空间中所有的子句均可由该子句集扩展,函数 *Expand* 一直返回 True,则算法遍历完极大项后在第 11 行返回 UNSAT.综上所述,算法是完备的. □

3 基于分解的定理证明方法

以上给出了极大部分空间的求解方法及其可满足性判定方法,如果存在极大项部分空间可满足,即,存在极大项不可由子句集扩展,则子句集是可满足的;若所有极大项部分空间都不可满足,即,所有极大项都可由子句集扩展,则子句集不可满足.下面给出基于分解的定理证明方法 IPSER 和 DPSER.

3.1 IPSER定理证明方法

在 IER 算法中,不能判定子句集的满足性时,需重新调用原来的 ER 算法,因此影响了算法的效率.此时,我们可以使用算法 FirstMTS 求出 $MTR(C)$,然后用 PSER 算法判定 $MTR(C)$的可满足性.此时,如果 $MTR(C)$是可满足的,则子句集可满足;否则,子句集不可满足.下面给出基于 IER 算法改进的 IPSER 算法.

令子句集 $Y=\{C_1,C_2,\ldots,C_n\}$,其文字的集合为 M 且$|M|=m$,C 是任意一个 M 上的子句.

Function IPSER.

```
1.  BEGIN
2.    Y'←Y
3.    For each P∈Y'
4.    If Complement(C,P) Then Y'←Y'−P
5.    If ER(Y')==SAT
6.      Return SAT
7.    Else
8.      Reorder(C,Y)
9.      FirstMTS(C,M)
10.     PSER(Y,pstarti,pendi)
11. END
```

IPSER 算法解决了 IER 算法在不能判定子句集的满足性时,需重新调用原来的 ER 算法的问题,使 IER 算法完备.当使用 IER 算法判定得到子空间上所有极大项都被扩展出来,即,不能判定子句集的可满足性时,调用 *Reorder*(C,Y)算法对子句集中原子的顺序根据 C 中原子的顺序进行调整,然后使用 PSER 算法判定子空间对应的补空间的可满足性:若补空间中有极大项不能被扩展,则子句集可满足;若补空间的所有极大项都能由子句集扩展得到,则子句集不可满足.本文提出的 PSER 算法满足了先在子空间判定满足性,若得不到判定结果,再在补空间上判定,最后得到子句集的可满足性.

3.2 DPSER定理证明方法

DPSER 算法基本思想:将 MOM 启发式策略应用在限定搜索部分空间的子句 C 的选择,然后在子空间上调用 PSER 算法判定子空间的可满足性:如果可满足,则子句集可满足;如果子空间不可满足,再调用 PSER 判定子空间对应的补空间的可满足性,从而得到子句集的可满足性.

在 IER 算法中,尽可能通过在子空间上判定出问题是否可满足,从而达到提高效率的目的.通过选取子句 C 来限定搜索空间,所有与 C 含互补对的子句都被删除,而实际的搜索空间因此也被限定,所以 C 的选取十分重要.如果输入的子句 C 为单元子句 A,那么实际上是在所有 A 为正的极大项组成的空间中进行搜索.基于此,我们结合 DPLL 算法中动态选择下一个分裂变量的启发式策略 MOM[26]限定搜索空间的子句的选择,下面介绍启发式策略 DMOM.

DMOM 启发式策略:在子句集中,优先选择度较大子句中出现频率高的文字的否定加入子句 C.

Function DMOM().

```
1.  BEGIN
2.     S ←MaxDegree(Y);
3.     For each C_i∈S
4.        For each L∈literal(C_i)
5.           Count_L←Count_L+1;
6.     Num←KthCount();
7.     For each L∈literal(S)
8.        If Count_L>Num Then C←C∪{¬L};
9.     Return C;
10. END
```

算法的基本流程是:首先遍历子句集 Y,得到其中度较大的子句的集合 S,该功能由函数 *MaxDegree* 完成;然后遍历集合 S,记录每个文字在 S 中出现的次数,用数组 *Count* 计数;统计完每个文字出现的次数后,调用函数 *KthCount* 得到 *Count* 数组中第 K 大的值 *Num*;最后遍历 S 中出现的所有文字 L,判断其出现次数 $Count_L$ 是否大于 *Num*,若大于,则将其否定$\neg L$ 加入子句 C.

该启发式策略中,基于两个优先原则选择文字:首先,度较大的子句中,文字否定的优先级比较高,度较小的子句可半扩展出的极大项个数较多,因而将度较大的子句从子句集中删掉后,可加快极大项的遍历速度,从而较早地得出不可被扩展的极大项或所有极大项都可被扩展;其次,出现次数较多的文字的否定优先级别比较高,因为我们将所有与 C 含有互补对的子句删掉,这样选取文字使得删除子句的个数最多,得到的问题规模最小,由于规模的缩小,判断极大项是否可被扩展时可以很快地返回结果.

通过以上的分析可知,该启发式策略很适合用于限定搜索空间的子句 C 的选取.

Function DPSER(***CNF***:***Y***).

```
1.  BEGIN
2.     C←DMOM(Y);
3.     Y'←Y
4.     For each C_i∈Y'
5.        If (Complement(C,C_i)) Then Y'←Y'/{C_i};
6.     Reorder(C,Y)
7.     f←FirstMTS(C,M)
8.     If PSER(Y',f+1,f+2^(m−c))==SAT
9.        Return SAT
10.    Else
11.       If PSER(Y,1,f)==SAT return SAT
12.       If PSER(Y,f+2^(m−c)+1,2^m)==SAT return SAT
13.    Return UNSAT.
14. END
```

在 DPSER 算法中,首先调用函数 *DMOM*,该函数根据 DMOM 启发式策略生成子句 *C*.然后,对于子句集 *Y*′中的每个子句 C_i,判断是否与 *C* 含有互补对,若含有,将 C_i 删除.调用 *Reorder*(*C*,*Y*)算对子句集中原子的顺序根据 *C* 中原子的顺序进行调整,然后调用 PSER 算法判定子句 *C* 限定的部分空间 *MTS*(*C*)的可满足性.此时,如有极大项不能由子句集中子句扩展得到,则子句集可满足.如 *MTS*(*C*)上所有极大项都可被扩展,则调用 PSER 算法判定子句 *C* 限定的子空间的补空间 *MTR*(*C*).在 *MTR*(*C*)上若有极大项不能由子句集中子句扩展得到,则子句集可满足;如 *MTR*(*C*)上所有极大项都可被扩展,则子句集不可满足.

定理 7. 在命题逻辑中,算法 DPSER 是正确且完备的.

证明:由于 DMOM 启发式策略根据文字在子句集中出现的频率对文字进行选取,所以选取的子句 *C* 对应的极大项子空间必然包含于极大项空间中.如果在 *C* 限定的子空间中存在不可扩展的极大项,则子句集的所有极大项中存在极大项不可扩展,那么子句集是可满足的;否则,调用 PSER 算法判定子句集的补空间的可满足性,如有极大项不能被扩展,则子句集是可满足的.如果所有极大项都可被扩展,则子句集是不可满足的.由此可见,算法 DPSER 是正确的并且是完备的. □

4 实验结果

在文献[23]基础上,将我们提出的 IPSER 算法与原算法 IER,IBOHMH_IER 和 IMOMH_IER 以及 Dechter 和 Rish 提出的命题逻辑中效率较高的归结方法——有向归结算法(directional resolution,简称 DR)[27]进行了比较.测试环境为:硬件 DELL Intel PIV 2.8GHz CPU/512MB RAM;软件 Windows XP Professional SP2/Visual C++6.0.测试用例选用处于相变区域的 Uniform Random-3-SAT 问题和 AIM 类问题(http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html 2007)进行测试,实验结果见表 1.表中"—"表示时间超时,超时时间为 600s.通过测试可以看出:我们提出的 IPSER 算法较原算法在执行效率上有明显的提高,比较快的 IBOHMH_IER 和 IMOMH_ IER 算法效率提高 2~5 倍.

Table 1 Experimental results of benchmark instances

表 1 标准用例测试结果

Instances	Algorithms				
	DR(s)	IER(s)	IBOHMH_IER(s)	IMIMH_IER(s)	IPSER(s)
uf20_07	0.781	2.875	0.015	0.015	0.009
uf20_08	0.710	0.265	0.015	0.014	0.006
uf20_09	3.671	6.171	0.140	0.141	0.054
uf20_010	3.703	3.600	0.031	0.312	0.082
uf20_013	3.828	1.437	0.234	0.515	0.105
uf20_016	1.140	0.218	0.015	0.010	0.003
uf20_018	1.312	15.218	0.014	0.015	0.008
aim-50-1_6-yes1-1	–	315.570	103.520	47.406	9.358
aim-50-1_6-yes1-2	3.031	20.462	6.297	2.750	0.620

此外,我们将算法 DPSER 分别与 NER 算法、IER 算法和 Dechter 等人提出的有向归结算法进行了比较.本实验在文献[21]实验数据基础上,对 DPSER 算法使用随机 SAT 问题进行测试.测试使用的随机 SAT 问题样例是由一个随机产生器产生,随机产生器中有 3 个输入参数⟨*N*,*M*,*K*⟩,其中,*N* 代表子句集中子句的个数,*M* 代表子句集中变量的个数,*K* 代表子句集中子句的最大长度.以下对不同规模的问题进行了测试和比较.

首先,我们选取参数为⟨*N*,20,10⟩和⟨*N*,30,10⟩的问题进行测试,其中,60≤*N*≤160.对于每类难度的问题,随机生成 10 个样例进行求解,均值作为最后结果,实验结果如图 1 和图 2 所示.此外,我们还对参数为⟨80,*M*,10⟩和⟨120,*M*,10⟩(10≤*M*≤30 之间)的随机 SAT 问题进行了测试,实验结果如图 3 和图 4 所示.

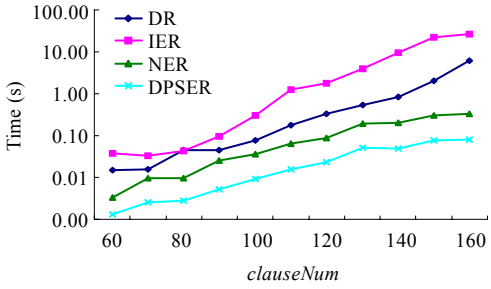

Fig.1 ⟨*N*,20,10⟩

图 1 ⟨*N*,20,10⟩

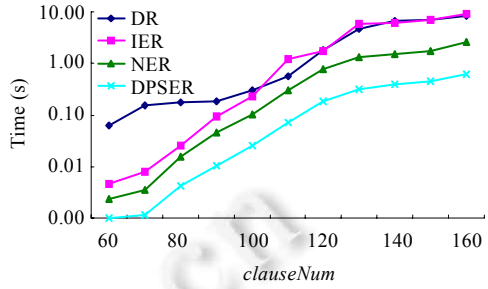

Fig.2 ⟨*N*,30,10⟩

图 2 ⟨*N*,30,10⟩

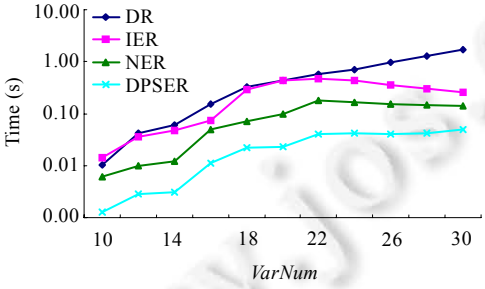

Fig.3 ⟨80,*M*,10⟩

图 3 ⟨80,*M*,10⟩

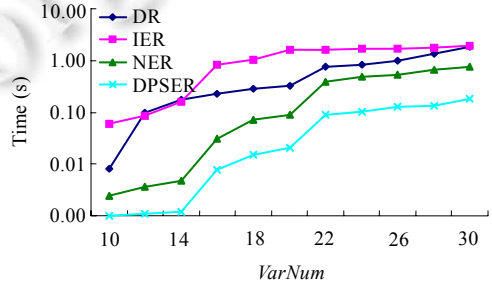

Fig.4 ⟨120,*M*,10⟩

图 4 ⟨120,*M*,10⟩

通过对参数为⟨*N*,20,10⟩和⟨*N*,30,10⟩的随机 SAT 问题进行测试,从图 1 和图 2 中可以看出:DPSER 算法在效率上有明显的优势,比较快的 NER 算法效率提高 2~3 倍.并且当子句个数增加到 130 以上时,我们提出的 DPSER 算法的计算时间增加平缓.

通过对参数为⟨80,*M*,10⟩和⟨120,*M*,10⟩的随机 SAT 问题的测试,可以看出:我们的 DPSER 算法在效率上有明显的优势,比较快的 NER 算法效率提高 2~4 倍.

5 结 论

定理机器证明是人工智能重要的研究方向之一,并已成功地应用于很多领域.许多学者对扩展规则方法进行了研究.Murray 已将扩展规则方法用于知识编译的目标语言生成过程,并取得了较好的结果.在 IER 算法中不能判定子句集的满足性时,需重新调用原来的 ER 算法,因此影响了算法的效率.为了避免重新调用原来的 ER 算法的缺点,本文给出了子句集的极大项空间分解后子空间的求解方法及其可满足性的判定方法 PSER.当使用 IER 算法判定得到子空间上所有极大项都被扩展出来,即,不能判定子句集的可满足性时,使用 PSER 算法判定子空间对应的补空间的可满足性.若补空间中有极大项不能由子句集扩展得到,则子句集可满足;若补空间的所有极大项都能由子句集扩展得到,则子句集不可满足.因此使 IER 算法完备,满足了先在子空间判定满足性,若不能判定可满足性,再在补空间上判定,最后得到子句集的可满足性.最后,我们给出了 DPSER 算法,通过 DMOM 启发式策略来限定搜索部分空间的子句 C 的选择,然后在子空间上调用 PSER 算法判定子空间的可满足性;如果可满足,则子句集可满足.如果子空间不可满足,再调用 PSER 判定子空间对应的补空间的可满足性,从而得到子句集的可满足性.实验结果表明:我们给出的 IPSER 算法和 DPSER 算法的具有较好的执行效率,优于 NER,DR 和 IER 等算法.

致谢 本文作者对所有匿名审稿人的辛勤劳动表示真诚的感谢.

References:

[1] Zhou JP, Yin MH, Zhou CG. New worst-case upper bound for #2-sat and #3-sat with the number of clauses as parameter. In: Proc. of the AAAI 2010. Atlanta: AAAI Press, 2010. 217−222. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1672

[2] Giampaolo B, Paulson LC. Accountability protocols: Formalized and verified. ACM Trans. on Information and System Security, 2006,9(2):138−161. [doi:10.1145/1151414.1151416]

[3] Rintanen J, Heljanko K, Niemel I. Parallel encodings of classical planning as satisfiability. In: Proc. of the 9th European Conf. on Logics in Artificial Intelligence. Lisbon: Springer-Verlag, 2004. 27−30. [doi: 10.1007/978-3-540-30227-8_27]

[4] Grastien A, Anbulagan N, Rintanen J, Kelareva E. Diagnosis of discrete-event systems using satisfiability algorithms. In: Proc. of the 22nd Conf. on Artificial Intelligence (AAAI 2007). Vancouver: AAAI Press, 2007. 305−310.

[5] Rintanen J, Grastien A. Diagnosability testing with satisfiability algorithms. In: Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007). Hyderabad: Morgan Kaufmann Publishers, 2007. 532−537. [doi:10.1016/j.tcs.2006.08.002]

[6] Henry K, Bart S. Unifying SAT-based and graph-based planning. In: Proc. of the 16th Int'l Joint Conf. on Artificial Intelligence (IJCAI'99). Morgan Kaufmann Publishers, 1999. 318−325. http://dl.acm.org/citation.cfm?id=1624218.1624265

[7] Henry AK. Deconstructing planning as satisfiability. In: Proc. of the 21th National Conf. on Artificial Intelligence (AAAI 2006). Boston: AAAI Press, 2006. 1524−1526. http://dl.acm.org/citation.cfm?id=1597348.1597432

[8] Myla A. TAME: Using PVS strategies for special-purpose theorem proving. Annals Mathematics and Artificial Intelligence, 2000, 29(1-4):139−181. [doi:10.1023/A:1018913028597]

[9] Bruno B. An efficient cryptographic protocol verifier based on prolog rules. In: Proc. of the 14th IEEE Computer Security Foundations Workshop (CSFW 2014). Cape Breton: IEEE Computer Society Press, 2001. 82−96. [doi:10.1109/CSFW.2001. 930138]

[10] Yannick C, Laurent V. Automated unbounded verification of security protocols. In: Proc. of the Computer-Aided Verification Conf. (CAV 2002). Paris: Kluwer Academic, 2002. 324−337. http://dl.acm.org/citation.cfm?id=647771.734411

[11] Alessandro A, Luca C. SAT-Based model-checking for security protocols analysis. Int'l Journal of Information Security, 2008,7(1): 3−32. [doi:10.1007/s10207-007-0041-y]

[12] Wang XH, Liu XH. Generalized resolution. Chinese Journal of Computers, 1982,5(2):81−92 (in Chinese with English abstract).

[13] 孙吉贵,刘叙华.广义线性半锁归结.科学通报,1992,37(19):1812−1814.

[14] Fitting M. First-Order Logic and Automated Theorem Proving. New York: Springer-Verlag, 1990. http://dl.acm.org/citation. cfm?id=78167

[15] Liu Q, Sun JG. A Boolean pruning method for improving Tableau reasoning efficiency in first-order many-valued logic. Chinese Journal of Computers, 2003,26(9):1165−1170 (in Chinese with English abstract).

[16] Liu Q, Sun JG, Cui ZM. A method of simplifying many-valued generalized quantifiers tableau rules based on boolean pruning. Chinese Journal of Computers, 2005,28(9):1514−1518 (in Chinese with English abstract).

[17] Lin H, Sun JG, Zhang YM. Theorem proving based on the extension rule. Journal of Automated Reasoning, 2003,31:11−21. [doi:10.1023/A:1027339205632]

[18] Wu X, Sun JG, Lin H, Feng SS. Modal extension rule. Progress in Natural Science, 2005,15(6):550−558.

[19] Lai Y, Ouyang DT, Cai DB, Lü S. Model counting and planning using extension rule model counting and planning using extension rule. Journal of Computer Research and Development, 2009,46(3):459−469 (in Chinese with English abstract).

[20] Yin MH, Sun JG, Wu X. Possibilistic extension rules for reasoning and knowledge compilation. Ruan Jian Xue Bao/Journal of Software, 2010,21(11):2826−2837 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3690.htm [doi: 10.3724/SP. J.1001.2010.03690]

[21] Sun JG, Li Y, Zhu XJ, Lü S. A novel theorem proving algorithm based on extension rule. Journal of Computer Research and Development, 2009,14(1):9−14 (in Chinese with English abstract).

[22] Gu WX, Wang JY, Yin MH. Knowledge compilation using extension rule based on MCN and MO heuristic strategies. Journal of Computer Research and Development, 2011,48(11):2064−2073.