软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(8):2096−2112 [doi: 10.13328/j.cnki.jos.005125]

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

EPCCL 理论的求交知识编译算法*

牛当当[1], 刘 磊[1], 吕 帅[1,2,3]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](吉林大学 数学学院,吉林 长春 130012)

[3](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)

通讯作者: 吕帅, E-mail: lus@jlu.edu.cn

摘 要: 超扩展规则是对扩展规则的扩充,基于超扩展规则能够求得任意两个非互补且不相互蕴含的子句所能扩展出极大项集的交集、差集和并集,并将所得结果以 EPCCL(each pair of clauses contains complementary literals)理论的形式保存.基于超扩展规则的性质,提出一种 EPCCL 理论编译算法:求交知识编译算法 IKCHER(intersection approach to knowledge compilation based on hyper extension rule).该算法适合难解类 SAT 问题的知识编译,也是一种可并行的知识编译算法.研究了如何实现多个 EPCCL 理论的求交操作,证明了 EPCCL 理论的求交过程是可并行执行的,并设计了相应的并行求交算法 PIAE(parellel intersection of any number of EPCCL).通过对输入 EPCCL 理论对应普通子句集的利用,设计了一种高效的并行求交算法 imp-PIAE(improvement of PIAE).基于上述算法,还设计了两种并行知识编译算法 P-IKCHER(IKCHER with PIAE)和 impP-IKCHER(IKCHER with imp-PIAE),分别采用 PIAE 并行合并算法和 imp-PUAE 并行合并算法.最后,通过实验验证了,大部分情况下,IKCHER 算法的编译质量是目前为止所有 EPCCL 理论编译器中最优的,P-IKCHER 算法所使用的合并策略并没有起到加速的效果,反而使得编译效率和编译质量有所下降;impP-IKCHER 算法提高了 IKCHER 算法的编译效率,CPU 四核环境下最高可提高 2 倍.

关键词: 知识编译;扩展规则;超扩展规则;EPCCL(each pair of clauses contains complementary literals)理论;并行知识编译

中图法分类号: TP182

中文引用格式: 牛当当,刘磊,吕帅.EPCCL 理论的求交知识编译算法.软件学报,2017,28(8):2096−2112. http://www.jos.org.cn/1000-9825/5125.htm

英文引用格式: Niu DD, Liu L, Lü S. Knowledge compilation algorithm based on computing the intersection for EPCCL theories. Ruan Jian Xue Bao/Journal of Software, 2017,28(8):2096−2112 (in Chinese). http://www.jos.org.cn/1000-9825/5125.htm

Knowledge Compilation Algorithm Based on Computing the Intersection for EPCCL Theories

NIU Dang-Dang[1], LIU Lei[1], LÜ Shuai[1,2,3]

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

[2](College of Mathematics, Jilin University, Changchun 130012, China)

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

* 基金项目: 国家自然科学基金(61300049, 61502197, 61503044); 教育部高等学校博士学科点专项科研基金(20120061120059); 吉林省重点科技攻关项目(20130206052GX); 吉林省自然科学基金(20140520069JH, 20150101054JC, 20150520058JH)

Foundation item: National Natural Science Foundation of China (61300049, 61502197, 61503044); Specialized Research Fund for the Doctoral Program of Higher Education of China (20120061120059); Key Program for Science and Technology Development of Jilin Province of China (20130206052GX); Natural Science Research Foundation of Jilin Province of China (20140520069JH, 20150101054JC, 20150520058JH)

收稿时间: 2015-11-24; 修改时间: 2016-04-04; 采用时间: 2016-07-20; jos 在线出版时间: 2016-10-11

CNKI 网络优先出版: 2016-10-12 16:26:51, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.017.html

Abstract: HER (hyper extension rule) is an expansion of ER (extension rule). The results of computing the union and difference set of two sets of maximum terms which are extended by two clauses respectively based on the hyper extension rule can be saved as EPCCL (each pair of clauses contains complementary literals) theories. In this paper, a parallelizable knowledge compilation algorithm for EPCCL theories, IKCHER, is proposed based on computing the intersection of maximum terms sets. The focus is placed on the research of parallel computing the intersection sets of multiple EPCCL theories based on HER, resulting in the design of the corresponding algorithm, PIAE (parallel intersection of any number of EPCCL). Through using the origin CNF formulae of EPCCL theories, another efficient merging algorithm imp-PIAE is proposed. Based on above methods two parallel knowledge compilation algorithms P-IKCHER and impP-IKCHER are constructed for EPCCL theories, utilizing the PIAE algorithm and imp-PIAE algorithm to merge multiple EPCCL theories respectively. Experimental results show that IKCHER algorithm has the best compilation quality of all EPCCL compilation algorithms. P-IKCHER algorithm does not improve the compilation quality and compilation efficiency of IKCHER while impP-IKCHER algorithms can maintain the compilation quality of IKCHER, and improve the compilation efficiency of IKCHER. When the number of CPU cores is 4, the compilation efficiency of IKCHER can be improved twice as much in the best cases.

Key words: knowledge compilation; extension rule; hyper extension rule; EPCCL (each pair of clauses contains complementary literals) theory; parallel knowledge compilation

可满足性问题(SAT 问题)是 NP 完全问题的核心[1],相变现象在不同形式的 SAT 问题中是普遍存在的[2,3],相变点附近的命题可满足性判定通常需要巨大的时间开销.目前,主要通过改善局部搜索或设计新型启发式等手段使 SAT 求解得以高效实现[4−8].SAT 求解并行化是提高 SAT 求解效率的重要手段之一,同时也是一个热门领域[9−11].知识库的查询操作通常是通过调用 SAT 求解器完成的,然而 SAT 求解始终需要指数级的时间,因此,通过多次调用 SAT 求解器完成对同一个知识库的多次查询显然是不合适的.

知识编译的主要目的是为了提高重复性任务的计算效率,主要思想为,将问题的求解分为两个基本阶段:离线编译和在线推理.Val 给出了可控制类的概念,可以作为目标编译语言的判断标准[12].Van den Broeck 等人研究了约简的 SDD(sentential decision diagram)语言上多项式时间内的推理方法[13].Marquis 和 Fargier 等人研究了几种不同目标编译语言中存在的闭包[14,15].Darwiche 提出了一种将子句形式的命题公式编译为可分解否定范式(DNNF)的完备知识编译方法[16,17],并给出了知识编译图谱[18].Fargier 基于 Krom,Horn 和 Affine 这 3 种较有影响力的命题片段对 Darwiche 提出的知识编译图谱做了扩充[19],并设计了针对变种表示语言的知识编译图谱[20]和有序真值决策图的知识编译图谱[21].Koriche 等人研究了基于 Affine 决策树的模型计数方法,并弥补了知识编译图谱中 Affine 族的空缺[22].Lai 等人设计了带有蕴含文字的有序二元决策图,是一种新的知识编译语言[23].目前,国际上针对知识编译的研究大多集中于对各种知识编译语言的比较以及如何提高目标语言的查询能力上.知识编译方法与 SAT 求解存在着紧密的联系,例如,可满足判定是知识编译图谱中 8 种查询操作的一种[18];Huang 等人基于先进的 SAT 求解器实现了多种知识编译语言的编译[24].因此,如何借鉴并行 SAT 求解的方法实现知识编译方法的并行化,也是一个值得探索的新领域.

2003 年,Lin 等人提出了扩展规则推理方法(extension rule,简称 ER)[25],被著名人工智能专家 Davis 称为与归结方法“互补”的方法.与归结方法相反,扩展规则推理方法通过扩展出所有极大项组成的集合判定子句集的可满足性.Lin 等人基于扩展规则提出了一种新的命题逻辑理论证明方法 IER[25].殷明浩等人提出了 CER 算法.该算法基于容斥原理解决了 ER 算法在求解#SAT 问题的空间复杂性问题[26].赖永等人提出了一种命题扩展规则方法 ER 的高效实现和#ER 算法,并结合#DPLL 算法和#ER 算法提出了#CDE 算法,具有较高的命题推理效率和#SAT 求解效率[27].李莹等人在 ER 算法的基础上提出了分别基于 IMOM 和 IBOHM 启发式策略的 IMOMH_IER 和 IBOHMH_IER 算法,提高了扩展规则推理算法的效率[28].此外,李莹等人基于扩展规则还提出了一种新的定理证明技术 NER[29].在扩展规则的基础上,Lin 等人提出了一种基于扩展规则的知识编译方法 KCER,可以将子句集编译为 EPCCL 理论[30].EPCCL 理论是一种高效的目标编译语言,相对于现有知识编译语言具有较强的竞争力,基于 EPCCL 理论能够在线性时间内实现知识编译图谱中的全部查询操作.因此,深入研究 EPCCL 理论的特性,通过挖掘 EPCCL 编译器编译过程中的有效信息来提高编译效率及编译质量,对于 EPCCL 理论的发展具有重要意义.Yin 等人基于扩展规则和知识编译设计了求解模型计数问题的 KCCER 算法.该算法具有较高

的效率[26,31].谷文祥等人设计了 MCN 和 MO 两种启发式策略,提高了 KCER 算法的编译效率,并降低了编译后的子句集规模[32].刘大有等人基于扩展规则提出了一种新的 EPCCL 理论编译器 C2E.该编译器有较高的编译效率[33].我们基于超扩展规则[34]提出了扩展反驳的概念,在扩展反驳与知识编译之间建立了联系,并提出了两种知识编译算法:DKCHER 和 UKCHER[35].目前为止,与其他 EPCCL 理论编译算法相比,DKCHER 对于相变点附近的难解问题有非常优秀的编译效率和编译质量,且对于子句数和变量数的比值较大的子句集同样能够取得最优的编译效率和编译质量,是目前为止最优秀的 EPCCL 理论编译算法,然而它并不能直接并行化.通过分析算法 UKCHER 的执行过程,证明了它是目前为止唯一一个能够并行知识编译的 EPCCL 理论编译算法,然而其编译质量和编译效率较差.因此,对于 EPCCL 理论编译算法的提升,存在两种主要途径:(1) 提高串行编译算法的编译质量和编译效率;(2) 将优秀的 EPCCL 理论编译算法并行化.

本文基于我们提出的超扩展规则[34],设计了一种新的 EPCCL 理论编译算法 IKCHER.同时,本文还设计了两种并行求交策略 PIAE 和 imp-PIAE,均能求得任意两种 EPCCL 理论所能扩展出的极大项集的交集.基于上述两种并行求交策略,本文设计了两种并行 EPCCL 理论编译算法:P-IKCHER 和 impP-IKCHER.实验结果表明:大部分情况下,IKCHER 算法的编译质量是目前为止所有 EPCCL 理论编译器中最优的,其求解效率与目前最好的 EPCCL 理论编译器 DKCHER 算法相当;P-IKCHER 算法所使用的合并策略并没有起到加速的效果,反而使得编译效率和编译质量有所下降;impP-IKCHER 算法提高了 IKCHER 算法的编译效率,四核并行下最高可提高 2 倍.目前,尚未出现关于并行知识编译的成果,本文的研究工作能够为其他 EPCCL 理论编译算法的并行化及其他目标语言编译算法的并行化提供借鉴.

本文第 1 节介绍超扩展规则的基本概念和性质.第 2 节介绍求交知识编译算法 IKCHER.第 3 节对本文提出的 IKCHER 算法与 DKCHER、C2E 等优秀 EPCCL 理论编译算法做对比测试.第 4 节设计两种并行求交策略 PIAE 和 imp-PIAE,并提出两种对应的 EPCCL 理论并行知识编译算法 P-IKCHER 和 impP-IKCHER.第 5 节测试两种 EPCCL 理论并行知识编译算法的编译质量和编译效率.最后为本文总结.

1 超扩展规则

扩展规则能够对单个子句和单个原子进行扩展操作.在求解推理问题的过程中,直接对多个子句操作能够提高处理效率,并且尽可能多地利用潜在的问题结构.超扩展规则可应用于两个子句,是经典扩展规则的扩展形式.

定义 **1**(扩展规则(**extension rule**))[25]. 给定一个子句 C 和一个变量集 M,$D=\{C\vee a, C\vee\neg a\}$,其中 $a\in M$ 并且 a 和 $\neg a$ 都不在 C 中出现,将 C 到 D 的推导过程称为扩展规则,D 中的子句称为 C 扩展得到的结果,并且 $C\equiv D$.

由定义 1 可以看出,扩展规则与归结规则是完全相反的规则.同时,定义 1 应用扩展规则后得到的子句集与原子句集等价.因此,扩展规则可以被看作是一条新的推理规则.应用扩展规则后,D 中子句间存在互补文字对.应用扩展规则推理方法过程中,期望得到扩展结果中子句间最好存在更多的互补文字对,以利用扩展规则的推理特性:依赖扩展规则的推理方法适用于互补因子较高的子句集可满足性判定[15].

根据文献[25],称任意一个子句 C 为变量集 M 上的极大项,当且仅当 C 包含 $|M|$ 个 M 中互不相同的变量.扩展规则要求一个子句针对一个变量进行扩展,超扩展规则允许一个子句针对另一个子句进行扩展.为了表示方便,定义子句 C 的(原子)变量集合为 $V(C)$,子句 C 关于变量集 M 所能扩展出的极大项集合为 $J_M(C)$,子句集 F 的(原子)变量集合为 $V(F)$.本文所研究的子句集中不包含重言式,同时符号“≡”表示式子两边语义等价.对于单个子句 C,$C\equiv J_M(C)$.

定义 **2**(超扩展规则(**hyper extension rule**)). 给定两个子句 C 和 A,$D=\{C\vee A, C\vee\neg A\}$,其中,$V(C)\cap V(A)=\varnothing$,将 C 到 D 的推导过程称为超扩展规则,D 中的元素为 C 应用超扩展规则的结果.

本文为了表示更直观,在定义 2 中规定 $V(C)\cap V(A)=\varnothing$,然而该条件并非必要条件.由定义 2 可以推出:如果 $A\not\vDash C$,且 A 和 C 不包含互补文字对,则利用 A 对 C 扩展所得到的结果为 $D=\{C\vee A, C\vee\neg(V(A)-V(C))\}$,其中,$C\vee A$ 为标准子句,即不包含相同的文字.超扩展规则中,子句 C 与其应用超扩展规则的扩展结果 D 是等价的.

与经典扩展规则不同,超扩展规则用子句 A 对子句 C 进行扩展,但这种扩展方法产生的 $C\vee\neg A$ 为非子句形式($C\vee A$ 为标准子句形式),因此需要对 $C\vee\neg A$ 以适当形式展开.假设 $A=\{l_1,\ldots,l_n\}$,则运用 De Morgan 律.

$$E=\{C\vee\neg A\}=\{C\vee\neg(l_1\vee\ldots\vee l_n)\}=\{C\vee\neg l_1,\ldots,C\vee\neg l_n\}.$$

上述展开过程是语义等价的,但由于每个 $l_i(i=1,\ldots,n)$两两不同且均不在 C 中出现,所以 E 的互补因子较低,难以利用扩展规则的推理特性.为了保证扩展结果子句间的互补性,采用如下方法展开:

$$\frac{\neg(l_1\vee\ldots\vee l_n)}{\begin{array}{c}(\neg l_1)\\(l_1\vee\neg l_2)\\\ldots\\(l_1\vee\ldots\vee l_{n-1}\vee\neg l_n)\end{array}} \tag{1}$$

公式(1)的展开过程称为互补展开,则 $E=\{C\vee\neg A\}=\{C\vee\neg l_1,C\vee l_1\vee\neg l_2,\ldots,C\vee l_1\vee\ldots\vee l_{n-1}\vee\neg l_n\}$,进而,

$$D=\{C\vee l_1\vee\ldots\vee l_n,C\vee\neg l_1,C\vee l_1\vee\neg l_2,\ldots,C\vee l_1\vee\ldots\vee l_{n-1}\vee\neg l_n\}.$$

依赖公式(1)的互补展开保证了超扩展规则的等价性,并且展开结果中子句之间存在互补文字对.同样,也可以采用公式(2)的形式描述互补展开:

$$CNF_{linear}(A\vee\neg(l\vee B))=\begin{cases}A\vee\neg l, & |B|=0\\ \{A\vee B\vee\neg l\}\cup CNF_{linear}(A\vee\neg B), & |B|>0\end{cases} \tag{2}$$

定义 **3(EPCCL 理论)**[30]. 若子句集 F 是一个 EPCCL 理论,则 F 中任意两个子句间均含有互补文字对.

超扩展规则能够利用一个子句扩展另一个子句,并且利用互补展开在保证可满足性等价的前提下保证了扩展之后的子句间存在互补文字对,这使得扩展过程能够高效进行并得到能够发挥扩展规则推理方法特性的理论.

超扩展规则还具有以下特殊性质[35].

性质 **1**[35]. 对两个子句使用超扩展规则得到的结果是一个 EPCCL 理论,由于互补展开使得超扩展规则的展开结果中所有子句之间存在互补文字对,因此使用超扩展规则得到的结果是一个 EPCCL 理论.

性质 **2**[35]. 基于超扩展规则可以计算任意两个子句所能扩展出的极大项的交集.

对于两个子句 C 和 A,如果 $A\not\models C$,且 A 和 C 不包含互补文字对,则利用子句 A 扩展子句 C 可以得到结果 $D=\{C\vee A,C\vee\neg(V(A)-V(C))\}$,其中,$C\vee A$ 代表了 $J_M(C)\cap J_M(A)$.

性质 **3**[35]. 基于超扩展规则能够利用 EPCCL 理论保存两个子句所能扩展的极大项集的差集.

对于两个子句 C 和 A,如果 $A\not\models C$,且 A 和 C 不包含互补文字对,则利用子句 A 扩展子句 C 可以得到结果 $D=\{C\vee A,C\vee\neg(V(A)-V(C))\}$,其中,$\{C\vee\neg(V(A)-V(C))\}$代表了 $J_M(C)-J_M(A)$.

2 基于超扩展规则的求交知识编译算法

在文献[35]中,我们利用超扩展规则的性质 1 和性质 3 设计了求并知识编译算法 UKCHER 和求差知识编译算法 DKCHER,其中,DKCHER 算法是目前为止最优秀的 EPCCL 理论编译算法,UKCHER 算法是目前为止唯一可并行的 EPCCL 理论编译算法.本节我们将利用超扩展规则的性质 2 设计一种新的可并行知识编译算法 IKCHER.IKCHER 算法采用了与 DKCHER 算法类似但不同的求解结构.该算法继承了 DKCHER 算法的优秀特性,并且还具备了可并行的新特性.

根据文献[35],子句□在变量集 $M(|M|=2^m)$上所能扩展出的极大项集为 M 上所有极大项的全集,令 H 表示等价于子句集 F 在 M 上所不能扩展出的极大项集的子句集,则 $J_M(H)=J_M(\square)-J_M(F)$且 $J_M(F)=J_M(\square)-J_M(H)$.若利用超扩展规则能够求得 $J_M(H)$,且将 $J_M(H)$保存为一个 EPCCL 理论,就能利用 $J_M(H)$求得 $J_M(F)$,且所得结果仍然是一个 EPCCL 理论.基于上述过程,能够将任意子句集编译为与之等价的 EPCCL 理论.为了求得任意子句集所不能扩展出的极大项集,本文利用定理 1 对求解过程进行具体描述.

定理 **1**. 给定子句集 $F=\{C_1,\ldots,C_n\}$,M 为在 F 中出现的所有变量的集合,令 H 为等价于 F 所不能扩展出的极

大项集的子句集,则 $J_M(H)=J_M(\square)-J_M(F)=J_M(\square)-J_M(C_1)-\ldots-J_M(C_n)=\bigcap_{1\le i\le n}(J_M(\square)-J_M(C_i))$.

证明:由于 $J_M(F)=\bigcup_{1\le i\le n}J_M(C_i)$,因此定理 1 显然成立. □

定理 1 中,根据超扩展规则性质 3 可求得任意 $J_M(\square)-J_M(C_i)(1\le i\le n)$,且所得结果为一个 EPCCL 理论.假设 $C_i=l_1\vee\ldots\vee l_k$,则 $J_M(\square)-J_M(C_i)\equiv\{\neg l_1,l_1\vee\neg l_2,\ldots,l_1\vee\ldots\vee l_{k-1}\vee\neg l_k\}$,显然,等式右边为一个 EPCCL 理论.

将所有的 $J_M(\square)-J_M(C_i)$求解为 EPCCL 理论后,再求 $\bigcap_{1\le i\le n}(J_M(\square)-J_M(C_i))$,并需要保证所得结果仍然是一个 EPCCL 理论.为此,我们首先提出了定理 2.

定理 **2**. 给定两个 EPCCL 理论 $E_1=\{R_1,\ldots,R_p\}$和 $E_2=\{T_1,\ldots,T_q\}$,则 E_1 和 E_2 在 M 上所能扩展出的极大项集的交集 $S=J_M(E_1)\cap J_M(E_2)\equiv\{I_1,\ldots,I_{p\times q}\}$,其中,任意 $I_i(1\le i\le p\times q)$为 $J_M(R_{(i-1)/q+1})\cap J_M(T_{(i-1)\bmod q+1})$的计算结果,且 S 为一个 EPCCL 理论.

证明:由于 $J_M(E_1)=\bigcup_{1\le i\le p}J_M(R_i)$且$J_M(E_2)=\bigcup_{1\le j\le q}J_M(T_j)$,因此,

$$J_M(E_1)\cap J_M(E_2)=\bigcup_{1\le i\le p}J_M(R_i)\cap\bigcup_{1\le j\le q}J_M(T_j)=\bigcup_{1\le i\le p,1\le j\le q}(J_M(R_i)\cap J_M(T_j));$$

由于任意 $I_i\equiv J_M(R_{(i-1)/q+1})\cap J_M(T_{(i-1)\bmod q+1})(1\le i\le p\times q)$,因此,$\{I_1,\ldots,I_{p\times q}\}\equiv\bigcup_{1\le i\le p,1\le j\le q}(J_M(R_i)\cap J_M(T_j))$.

定理 2 成立. □

定理 2 中需要计算任意两个子句所能扩展出的极大项,利用超扩展规则性质 2 能够求得两个非互补且不存在蕴含关系的子句的所能扩展出的极大项集的交集.对于两个互补或存在蕴含关系的子句,需要单独考虑其所能扩展出极大项集交集的求解.

引理 **1**[35]. 如果两个子句 C_i 和 C_j 包含互补文字对,则其在 M 上所能扩展出的极大项集的交集 $J_M(C_i)\cap J_M(C_j)=\varnothing$.

根据引理 1,能够计算互补子句的所能扩展出的极大项集的交集,对于非互补的子句,本文采用定理 3 的方式进行计算.

定理 **3**. 若两个子句 $C_i=l_1\vee\ldots\vee l_k$ 和 $C_j=g_1\vee\ldots\vee g_h$ 不包含互补文字对,则其在 M 上所能扩展出的极大项集的交集 $J_M(C_i)\cap J_M(C_j)=b_1\vee\ldots\vee b_s$,其中,$\{b_1,\ldots,b_s\}=\{l_1,\ldots,l_k\}\cup\{g_1,\ldots,g_h\}$.

证明:子句 C_i 所能扩展出的极大项集中,所有极大项必然包含文字集$\{l_1,\ldots,l_k\}$;子句 C_j 所能扩展出的极大项集中,所有极大项必然包含文字集 $\{g_1,\ldots,g_h\}$.因此,$J_M(C_i)\cap J_M(C_j)$中所有极大项必然包含文字集 $\{l_1,\ldots,l_k\}\cup\{g_1,\ldots,g_h\}$,从而定理 3 成立. □

根据引理 1 和定理 3,能够求得任意两个子句所能扩展出的极大项集的交集,因此能够将定理 2 中任意 $I_i(1\le i\le p\times q)$用子句的形式表示.

定理 **4**. 利用引理 1 和定理 3 对定理 2 中任意 $I_i(1\le i\le p\times q)$进行求解,所得结果$\{I_1,\ldots,I_{p\times q}\}$仍为一个 EPCCL 理论.

证明: 根据引理 1 和定理 3,所得结果$\{I_1,\ldots,I_{p\times q}\}$中会存在若干恒真子句,可以直接忽略.剩余子句中,任取两个子句 I_x 和 I_y,假设 I_x 是由 C_i 和 C_j 求极大项交集而得$(C_i\in E_1\&C_j\in E_2)$,I_y 是由 C_g 和 C_h 求极大项交集而得$(C_g\in E_1\&C_h\in E_2)$,显然,C_i 与 C_g 之间存在互补文字对,C_j 与 C_h 之间存在互补文字对.根据定理 3,I_x 子句由 C_i 和 C_j 中所包含文字的并集组成,I_y 由 C_g 和 C_h 中所包含文字的并集组成,因此,I_x 和 I_y 之间必然存在互补文字对.综上,$\{I_1,\ldots,I_{p\times q}\}$是一个 EPCCL 理论,定理 4 成立. □

根据上述定理,能够求得任意两个 EPCCL 理论所能扩展出的极大项集的交集,所得结果仍为 EPCCL 理论.因此,对于任意子句集 $F=\{C_1,\ldots,C_n\}$,首先求出每个子句 C_i 所不能扩展出的极大项集,并用 EPCCL 理论 E_i 予以保存,然后利用定理 2 逐步求出 $H=E_1\cap\ldots\cap E_n$,所得结果 H 等价于 F 所不能扩展出的极大项集.再用相同过程求得 H 所不能扩展出的极大项集,并用 EPCCL 理论保存,该 EPCCL 理论等价于 F.本文将上述过程具体化为算法形式,并将该算法称为 IKCHER,描述如下.

算法 **1**. IKCHER.

1. 输入:令子句集 $F_1=\{C_1,\ldots,C_n\}$,M 包含了 F 中出现的所有变量.
2. h=1
3. 初始化:$F_2=\{\neg C_1\}$,$F_3=\varnothing$,i=2,j=k=1
4. **While** $i\leq|F_1|$
5. S_i={$\neg C_i$ 的互补展开} //$J_M(\square)-J_M(C_i)$,变量集 M 包含了 F 中出现的所有变量
6. **While** $j\leq|F_2|$
7. **If** C_i 与 C_j 互补 **Then skip**
8. **Else**
9. **While** $k\leq|C_i|$
10. **If** $S_i[k]$与 C_j 互补 **Then skip**
11. **Else if** $S_i[k]\models C_j$ **Then** $F_3=F_3\cup\{C_j\}$
12. **Else if** $C_j\models S_i[k]$ **Then** $F_3=F_3\cup\{S_i[k]\}$
13. **Else** $F_3=F_3\cup\{C_j\vee(S_i[k]-C_j)\}$
14. k++
15. j++
16. k=1
17. $F_2=F_3$
18. $F_3=\varnothing$
19. i++
20. j=1
21. **If** h=1 **Then**
22. $F_1=F_2$
23. h−−
24. **Goto** 3
25. **Else return** F_2

定理 **5**. IKCHER 算法是正确完备的,且输出结果是一个等价的 EPCCL 理论.

证明:利用超扩展规则的性质 3,IKCHER 算法中的第 5 行实现了□与 F 中第 i 个子句 C_i 所能扩展出的极大项集的差集,计算结果为一个 EPCCL 理论.第 4 行中的循环保证了能够求得所有 $J_M(\square)-J_M(C_i)(1\leq i\leq n)$,第 6 行的循环用于计算前 i−1 个子句所不能扩展出的极大项集与第 i 个子句所不能扩展出的极大项集的交集,所得结果即为前 i 个子句所不能扩展出的极大项集.根据定理 4,F_2 始终是 EPCCL 理论,因此,当第 4 行的循环结束后就能求得与 F_1 所不能扩展出的极大项集等价的 EPCCL 理论 F_2.算法在第 21 行控制利用 h 继续求解 F_2 所不能扩展出的极大项集,再次求解所得结果即为与 F_1 等价的 EPCCL 理论.因此,IKCHER 算法是正确而完备的. □

通常情况下,SAT 问题的难解程度与其解的个数具有直接关系,难度越大则解越少.将问题编译为等价的 EPCCL 理论之后,编译结果所能扩展出的极大项数与弄假原子句集解的个数是相等的,因此直观上看,对于难解类 SAT 问题,直接将其编译为等价的 EPCCL 理论,其结果的规模将是非常庞大的.在编译难解类 SAT 问题时,与 DKCHER 算法类似,IKCHER 算法采用两阶段式的编译:第 1 阶段所得结果为原子句集所不能扩展出的极大项集,该集合的模与原子句集的模型数是等价的,且该结果为一个 EPCCL 理论,因此,第 1 阶段所得结果的规模将会是非常小的;第 2 阶段将求得第 1 阶段所得结果所不能扩展出的极大项集,并将结果保存为一个 EPCCL 理论,由于第 1 阶段所得结果的规模较小以及互补展开的特性,编译结果中会存在较多短子句,这些短子句甚至短于原子句集中任意子句,因此,第 2 阶段所得结果的规模将远远小于直接编译所得结果的规模.因此理论上,IKCHER 算法对于求解难度较大的 SAT 问题具有较好的编译效果.

另外,IKCHER 算法求解过程中会利用互补展开求解每个子句所不能扩展出的极大项集,并在展开后的 EPCCL 理论上进行求交操作,而 DKCHER 算法每次对一个子句进行求差操作,因此,IKCHER 算法求解过程比 DKCHER 算法求解过程所能获得的启发式信息更多,依据这些启发式信息所设计的启发式策略也将更加直观、高效.

3 IKCCER 算法编译效果的对比测试

本文提出了一种新的 EPCCL 理论编译算法:IKCHER 算法.KCER 算法[30]、C2E 算法[33]、UKCHER 算法[35]和 DKCHER 算法[35]均能将任意子句集编译为等价的 EPCCL 理论,其中,DKCHER 算法是目前为止表现最好的 EPCCL 理论编译器.本文在随机问题和国际上通用的测试用例上对比测试了 IKCHER 算法的编译效率和编译质量(使用编译结果中子句数量进行衡量).本文实验平台如下:

- CPU:Intel Core(TM) i7-3770 CPU@3.40GHZ 3.40GHZ;
- 内存:8GB;
- 操作系统:Windows 10.

3.1 对于随机子句长度的子句集上IKCHER算法的测试

我们用随机产生器生成了子句长度不固定的测试样例,随机产生器的结果为包含 3 个参数⟨*m*,*n*,*k*⟩的子句集,其中,*m* 为变量个数,*n* 为子句个数,*k* 为每个子句的最大长度.本文提出的知识编译算法和现有的 EPCCL 理论编译算法均不支持大规模的实例,表 1~表 3 中分别给出了⟨20,*n*,10⟩,⟨25,*n*,10⟩和⟨30,*n*,10⟩这 3 种变量数固定、子句数不同的随机测试样例的编译结果,实验结果为 50 次实验的平均值,size 表示子句个数,time 表示运行时间(单位为 s),□表示相应算法编译质量最优.下表类似,不再赘述.

Table 1 Experimental results on random instances ⟨20,*n*,10⟩

表 1 随机实例⟨20,*n*,10⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨20,30,10⟩	2 007	0.168	1 790	0.109	1 772	0.018	928	0.077	1 893	0.027
⟨20,40,10⟩	1 510	0.093	1 519	0.090	2 896	0.033	1 221	0.075	2 817	0.040
⟨20,50,10⟩	818	0.070	1 111	0.069	3 279	0.045	1 780	0.086	3 432	0.052
⟨20,60,10⟩	715	0.039	597	0.027	3 471	0.063	1 768	0.084	4 009	0.071
⟨20,70,10⟩	435	0.027	280	0.018	3 619	0.078	2 134	0.084	4 393	0.097
⟨20,80,10⟩	246	0.023	116	0.013	2 891	0.086	2 197	0.084	4 312	0.120
⟨20,90,10⟩	59	0.018	81	0.016	2 701	0.100	2 019	0.082	4 196	0.131
⟨20,100,10⟩	21	0.019	16	0.013	3 010	0.101	2 250	0.083	4 064	0.163

Table 2 Experimental results on random instances ⟨25,*n*,10⟩

表 2 随机实例⟨25,*n*,10⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨25,40,10⟩	8 149	3.278	12 502	3.334	14 887	0.137	6 975	0.170	15 796	0.256
⟨25,50,10⟩	4 713	1.038	7 858	1.556	18 495	0.300	7 306	0.166	18 895	0.472
⟨25,60,10⟩	3 333	0.325	3 686	0.791	27 011	0.412	7 142	0.151	27 227	0.803
⟨25,70,10⟩	2 228	0.278	1 987	0.264	15 268	0.447	9 264	0.164	18 482	0.668
⟨25,80,10⟩	1 482	0.241	1 367	0.263	24 434	0.601	9 360	0.179	26 216	1.040
⟨25,90,10⟩	1 479	0.281	731	0.221	18 227	0.813	12 302	0.282	25 749	1.197
⟨25,100,10⟩	237	0.091	329	0.213	18 691	0.780	9 639	0.181	26 868	1.596
⟨25,110,10⟩	178	0.142	202	0.161	16 661	0.818	9 193	0.181	24 048	1.386

Table 3 Experimental results on random instances ⟨30,*n*,10⟩

表 **3** 随机实例⟨30,*n*,10⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨30,50,10⟩	29 675	19.385	33 778	21.246	102 847	1.372	45 363	0.693	80 336	14.699
⟨30,60,10⟩	12 114	9.696	18 752	10.500	142 109	2.149	46 133	0.671	150 186	13.362
⟨30,70,10⟩	6 096	3.553	7 082	3.220	123 242	2.840	41 903	0.562	123 717	17.251
⟨30,80,10⟩	6 260	3.553	4 853	2.301	90 857	3.228	50 169	0.690	136 676	19.738
⟨30,90,10⟩	2 032	1.115	2 231	1.088	91 499	3.361	55 943	0.684	169 235	19.233
⟨30,100,10⟩	1 381	1.038	1 052	0.609	80 883	3.573	52 454	0.680	170 060	22.176
⟨30,110,10⟩	750	0.861	1 053	0.748	55 830	2.978	58 068	0.720	101 370	19.556
⟨30,120,10⟩	335	0.810	204	0.850	89 962	3.742	48 422	0.611	130 408	16.250

基于表 1~表 3 能够观察出以下现象:(1) 当变量数固定时,IKCHER 算法和 DKCHER 算法的编译结果的规模和编译所需时间会随着子句数的增加而减少;(2) 当子句集规模较小时,UKCHER 算法、C2E 算法和 KCER 算法的编译结果的规模和编译所需时间随着子句数的增加而增加;(3) 当子句集规模大于某一临界值时,UKCHER 算法、C2E 算法和 KCER 算法的编译结果的规模会维持不变,甚至减少;(4) 由于样例随机生成,因此可能会存在少量的波动.例如表 2 中,当 n<90 时,UKCHER 算法、C2E 算法和 KCER 算法的编译结果的规模和编译所需时间随着子句数的增加而逐渐增加(现象(2));当 n≥90 时,UKCHER 算法、C2E 算法和 KCER 算法的编译所需时间仍然随着子句数的增加而逐渐增加,其编译结果的规模却逐渐稳定下来(现象(3));当 n≥90 时,UKCHER 算法、C2E 算法和 KCER 算法编译结果的规模均上下无规律波动(现象(4)).类似的现象同样出现在表 1 和表 3 中.

从编译规模来看,表 1~表 3 中,45.8%的实例下,IKCHER 算法的编译质量是最优的;41.7%的实例下,DKCHER 算法的编译质量是最优的;12.5%的实例下,C2E 的编译质量是最优的.由于 DKCHER 算法和 IKCHER 算法的编译质量的差值较小,因此可以认定 IKCHER 算法和 DKCHER 算法的编译质量相当,二者在大多数情况下是最优的.表 1 中,当 n<90 时,C2E 算法的编译规模会随着子句数的增加而增加;当 n≥90 时,C2E 算法的编译规模会稳定在 2 100 左右.同样的现象出现在 KCER 算法中,不过整体来看,KCER 算法的编译规模大于 C2E 算法的编译规模.表 1 中,当 n<50 时,IKCHER 算法和 DKCHER 算法的编译规模大于 C2E 算法的编译规模;然而随着子句数的增加,当 n≥50 时,IKCHER 算法和 DKCHER 算法的编译规模远远小于 C2E 和 KCER 的编译规模.甚至当 n=100 时,C2E 算法的编译规模是 IKCHER 算法的 107 倍,是 DKCHER 算法编译规模的 128 倍;KCER 算法编译规模是 IKCHER 算法的 193 倍,是 DKCHER 算法编译规模的 254 倍.由此可知,求交知识编译算法 IKCHER 对于变量数固定、子句数规模较大的子句集能够取得很好的编译效果.UKCHER 算法的编译规模始终大于 C2E 算法的编译规模,而和 KCER 算法的编译规模接近,但是小于后者.从表 2 和表 3 可以得到和表 1 相同的结论.

从编译效率来看,表 1 中,当 n≤50 时,UKCHER 算法的效率是最高的,C2E 算法居于第 3 位,而 IKCHER 算法和 DKCHER 算法的效率最差.这是由于 IKCHER 算法和 DKCHER 算法的扩展过程中都需要保存较多的中间结果,其中很多无效的中间结果会参与到算法扩展过程中,影响了整体执行效率.当 n>50 时,DKCHER 算法的编译效率最高,IKCHER 算法的编译效率排在第 2 位,但是与 DKCHER 算法的编译效率相差不大;同时,随着子句数的增加,UKCHER 算法和 KCER 算法所需编译时间也随之大幅度增加,二者的编译效率最终均差于 C2E 算法.表 3 中,IKCHER 算法的编译效率始终差于 C2E 算法的编译效率.整体来看,对于随机长度的 SAT 问题,IKCHER 算法适用于变量数较小、子句数较大的子句集.

3.2 对于标准3-SAT子句集上IKCHER算法的测试

为了更全面地展示本文提出的知识编译算法的特性,本文还随机生成了变量数固定、子句数改变的标准 3-SAT 子句集,表 4~表 6 分别给出了⟨20,*n*⟩,⟨25,*n*⟩和⟨30,*n*⟩这 3 种不同 3-SAT 子句集实例的样例进行测试,测试结

果为 50 次实验的平均值.

Table 4 Experimental results on random 3-SAT instances ⟨20,*n*⟩

表 4 随机 3-SAT 实例⟨20,*n*⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨20,46⟩	1 493	0.049	1 601	0.047	6 199	0.105	2 972	0.093	6 199	0.114
⟨20,56⟩	745	0.028	857	0.032	6 726	0.153	3 101	0.091	6 726	0.160
⟨20,66⟩	329	0.026	299	0.029	6 945	0.205	3 250	0.087	6 945	0.203
⟨20,76⟩	132	0.025	129	0.028	6 108	0.251	3 241	0.092	6 118	0.245
⟨20,86⟩	54	0.025	63	0.026	6 008	0.261	3 010	0.101	5 859	0.282
⟨20,96⟩	17	0.027	22	0.026	6 556	0.337	3 457	0.093	6 812	0.351
⟨20,106⟩	6	0.028	5	0.025	6 512	0.337	3 303	0.097	6 943	0.376
⟨20,116⟩	3	0.025	3	0.024	6 961	0.343	3 075	0.091	6 709	0.414

Table 5 Experimental results on random 3-SAT instances ⟨25,*n*⟩

表 5 随机 3-SAT 实例⟨25,*n*⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨25,57⟩	8 012	0.846	8 031	1.009	41 680	0.859	17 861	0.230	41 680	1.612
⟨25,67⟩	4 386	0.453	4 237	0.473	43 660	1.182	21 320	0.268	43 660	1.940
⟨25,77⟩	1 967	0.234	1 849	0.274	42 422	1.537	18 908	0.244	42 422	2.160
⟨25,87⟩	613	0.175	645	0.209	43 344	1.933	17 566	0.249	43 344	2.688
⟨25,97⟩	258	0.198	347	0.190	44 898	2.394	18 826	0.259	45 470	3.101
⟨25,107⟩	92	.0189	145	0.206	43 106	2.711	19 763	0.251	43 360	3.285
⟨25,117⟩	36	0.212	34	0.216	42 009	2.568	21 979	0.273	40 233	3.745
⟨25,127⟩	8	0.180	16	0.218	44 833	2.801	18 243	0.243	43 504	3.933

Table 6 Experimental results on random 3-SAT instances ⟨30,*n*⟩

表 6 随机 3-SAT 实例⟨30,*n*⟩的实验结果

Instances	IKCHER		DKCHER		UKCHER		C2E		KCER	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨30,69⟩	26 436	6.774	20 548	5.854	230 468	6.160	112 513	1.272	235 802	28.009
⟨30,79⟩	12 453	3.014	13 999	3.401	219 648	6.114	107 060	1.145	–	–
⟨30,89⟩	6 697	2.342	7 721	2.074	–	–	103 951	1.181	–	–
⟨30,99⟩	2 949	1.535	4 116	1.947	–	–	112 619	1.198	–	–
⟨30,109⟩	1 152	1.536	1 671	1.623	–	–	109 678	1.129	–	–
⟨30,119⟩	425	1.796	345	1.501	–	–	122 524	1.303	–	–
⟨30,129⟩	163	1.835	107	1.664	–	–	115 610	1.244	–	–
⟨30,139⟩	53	1.646	58	1.567	–	–	106 545	1.146	–	–

基于表 4~表 6 能够得出:(1) 对于 3-SAT 子句集,当变量数固定、子句数改变时,UKCHER、C2E 和 KCER 这 3 种知识编译算法的编译规模均随着变量数的增加而增加;(2) IKCHER 算法和 DKCHER 算法的编译规模会随着子句数的增加而减少,而二者的编译质量相差不大.然而,其中 62.5%的实例下,IKCHER 算法的编译质量是最优的;37.5%的实例下,DKCHER 算法的编译质量是最优的,因此从编译质量角度考虑,在编译 3-SAT 子句集时,应尽可能地选择 IKCHER 算法.从编译效率上来看,DKCHER 算法的编译效率在大多数情况下优于 IKCHER 算法的编译效率,但是二者相差不大.

表 6 中,由于变量数过大,UKCHER 算法和 KCER 算法会经常内存溢出,因此,对于表 6 中的很多实例,UKCHER 和 KCER 是无法解决的.显然,二者不适用于变量数较多的 3-SAT 问题.

为了验证 IKCHER 算法并行化的可行性,本文进一步研究了如何实现 IKCHER 算法的并行化.

4 求交知识编译算法的并行化

本文将求解多个 EPCCL 理论所能扩展出的极大项集简称为 EPCCL 理论求交,EPCCL 理论求交对于实现 IKCHER 算法的并行化是一个关键问题.

4.1 EPCCL理论的求交操作

事实上,IKCHER 算法中用到了两个 EPCCL 理论的求交操作,算法执行过程中需要不断地对 F_2 和 S_i 进行求交操作.定理 2 给出了两个 EPCCL 理论求交的具体过程,任意两个 EPCCL 理论 $E_1=\{R_1,\ldots,R_p\}$ 和 $E_2=\{T_1,\ldots,T_q\}$ 求交之后,所得结果为一个 EPCCL 理论:$S=J_M(E_1)\cap J_M(E_2)\equiv\{I_1,\ldots,I_{p\times q}\}$,其中,任意 $I_i(1\leq i\leq p\times q)$为 $J_M(R_{(i-1)/q+1})\cap J_M(T_{(i-1) \bmod q+1})$的计算结果.显然,上述过程可以并行分解,由于任意 I_i 和 I_j 的求解过程之间互不影响,因此可以并行求解$\{I_1,\ldots,I_{p\times q}\}$.我们可以将 E_1 拆分成 k 个子集 $E_1^1,\ldots,E_1^k$,然后并行求得 $\bigcup_{1\leq i\leq k}(E_1^i\cap E_2)$,所得结果即为 E_1 和 E_2 所能扩展出的极大项集的交集.

由于两个 EPCCL 理论求交之后仍然是一个 EPCCL 理论,因此可以采用增量式的思想并行求得任意多个 EPCCL 理论所能扩展出的极大项集的交集.根据上述过程,我们设计了多个 EPCCL 理论的并行求交算法,并称其为 PIAE.该算法描述如下.

算法 **2**. PIAE.

1. 输入:s 个 EPCCL 理论$\{E_1,\ldots,E_s\}$
2. $E=E_1$, $h=2$
3. **While** $h\leq s$
4. **Divide** E_h **to** $\{E_h^1,\ldots,E_h^k\}$
5. **Parallel**
6. **Foreach** E_h^v $(1\leq v\leq k)$
7. $d=1$
8. **While** $d\leq|E_h^v|$
9. $T_v=ICE(E,C_d)$ $(C_d\in E_h^v)$
10. d++
11. **End parallel**
12. $E=T_1\cup\ldots\cup T_k$
13. h++
14. **Return** E

Sub-procedure ICE (CNF $F=\{C_1,\ldots,C_n\}$, clause C)

1. 令 $F_1=\varnothing$, $i=j=1$
2. **While** $i\leq|F|$
3. **If** C 与 C_i 互补 **Then skip**
4. **Else if** $C\vDash C_i$ **Then** $F_1=F_1\cup\{C_i\}$
5. **Else if** $C_i\vDash C$ **Then** $F_1=F_1\cup\{C\}$
6. **Else** $F_1=F_1\cup\{C_i\vee(C-C_i)\}$
7. j++
8. $j=1$
9. i++

10. **Return** F_1

算法 2 实现了 k 个 EPCCL 理论进行的并行求交操作,并将结果保存为 EPCCL 理论.该算法利用第 3 行的 while 循环实现 EPCCL 理论的增量式求交操作;在第 4 行,将已合并的 EPCCL 理论分为 k 份,然后和下一个未操作的 EPCCL 理论进行并行求交操作;第 5 行~第 11 行是算法的并行操作部分;第 12 行将求交结果合并.该算法在第 9 行调用了子程序 ICE.ICE 的功能是求解一个子句与一个子句集所能扩展出的极大项集的交集,若输入子句集是一个 EPCCL 理论,则输出的子句集为 EPCCL 理论,否则为普通子句集.

定理 **6**. 给定一个 EPCCL 理论 $E=\{R_1,\ldots,R_p\}$和一个子句集 $F=\{T_1,\ldots,T_q\}$,则 E_1 和 F 在 M 上所能扩展出的极大项集的差集 $S=J_M(E)-J_M(F)\equiv\bigcup_{1\le i\le p}(J_M(R_i)\cap J_M(\neg T_1)\cap\ldots\cap J_M(\neg T_q))$,且 S 为一个 EPCCL 理论.

证明:根据定理 1,$J_M(\neg T_1)\cap\ldots\cap J_M(\neg T_q)$能够表示 F 所不能扩展出的极大项集,

则任意 $J_M(R_i)\cap J_M(\neg T_1)\cap\ldots\cap J_M(\neg T_q)$能够表示 R_i 与 F 在 M 上所能扩展出的极大项集的差集,

因此, $S=J_M(E)-J_M(F)\equiv\bigcup_{1\le i\le p}(J_M(R_i)\cap J_M(\neg T_1)\cap\ldots\cap J_M(\neg T_q))$ 成立.

又由于 $J_M(R_i)\cap J_M(\neg T_1)\cap\ldots\cap J_M(\neg T_q)\subseteq J_M(R_i)$且 $E=\{R_1,\ldots,R_p\}$是一个 EPCCL 理论,

因此,S 是一个 EPCCL 理论. □

根据定理 6,PIAE 算法中,若能知道输入 EPCCL 理论对应与其所不能扩展出的极大项集等价的普通子句集,则可以直接对普通子句集操作.通常,当普通子句集的子句数与变量数的比值较小时,求解其所不能扩展出的极大项集所得 EPCCL 理论的规模要远远大于输入子句集,因此,直接对普通子句集操作能够加快合并过程.基于此,我们设计了新的并行 EPCCL 理论求并算法,称其为 imp-PIAE,描述如下.

算法 **3**. imp-PIAE.

1. 输入:s 个 EPCCL 理论$\{E_1,\ldots,E_s\}$,$F_1,\ldots,F_s$ 是分别与 $E_1,\ldots,E_s$ 所不能扩展出的极大项集等价的普通子句集
2. $E=E_1$, $h=2$
3. **While** $h\le s$
4. **Divide** E **to** $\{E_h^1,\ldots,E_h^k\}$
5. **Parallel**
6. **Foreach** E_h^k $(1\le v\le k)$
7. $d=1$
8. **While** $d\le|E_h^k|$
9. $T_v=CIF(F_h,C_d)$ $(C_d\in E_h^v)$
10. d++
11. **End parallel**
12. $E=T_1\cup\ldots\cup T_k$
13. h++
14. **Return** E

Sub-procedure CIF (CNF $F=\{C_1,\ldots,C_n\}$, clause C)

1. 令 $F_1=\{C\}$, $F_2=\varnothing$, $i=j=k=1$
2. **While** $i\le|F|$
3. $S_i=\{\neg C_i$ 的互补展开$\}$ //$J_M(\square)-J_M(C_i)$,变量集 M 包含了 F 中出现的所有变量
4. **While** $j\le|F_1|$
5. **If** C_i 与 C_j 互补 **Then skip**
6. **Else**
7. **While** $k\le|C_i|$

8. **If** $S_i[k]$与 C_j 互补 **Then skip**
9. **Else if** $S_i[k]\models C_j$ **Then** $F_2=F_2\cup\{C_j\}$
10. **Else if** $C_j\models S_i[k]$ **Then** $F_2=F_2\cup\{S_i[k]\}$
11. **Else** $F_2=F_2\cup\{C_j\vee(S_i[k]-C_j)\}$
12. k++
13. j++
14. k=1
15. $F_1=F_2$
16. $F_2=\varnothing$
17. i++
18. j=1
19. **Return** F_1

算法 3 的执行过程与算法 2 大致相同.不同之处在于:算法 3 第 9 行利用了子程序 CIF,而算法 2 调用了子程序 ICE.子程序 CIF 的功能是计算任意子句与任意子句集所不能扩展出的极大项集的交集,输出结果为一个 EPCCL 理论.CIF 算法的过程是 IKCHER 算法的子过程,其正确性可参照定理 5 的证明.

4.2 两种并行的IKCHER算法

从 IKCHER 算法的求解过程中不难看出,可以对求交知识编译过程进行拆分以实现并行执行.对于输入子句集 $F=\{C_1,\ldots,C_n\}$,有两种并行方式.

1) 利用互补展开求得 F 中 n 个子句分别所不能扩展出的极大项集$\{\neg C_1\},\ldots,\{\neg C_n\}$,然后利用 PIAE 并行合并这 n 个 EPCCL 理论,所得结果为 F 所不能扩展出的极大项集.再次利用上述过程即可求得与 F 等价的 EPCCL 理论,这种方式至少需要 n 次并行化,因此,并行编译过程拆分和合并需要较大开销.
2) 对于任意子句集 $F=\{C_1,\ldots,C_n\}$,将 F 划分为 k 个子集 $F_1,\ldots,F_k$,并行求出任意 F 子集 F_i 所不能扩展出的极大项集,所得结果用 EPCC 理论 E_i 保存,然后利用 PIAE 并行合并 F 子集对应的 k 个 EPCCL 理论.再次利用上述过程即可求得与 F 等价的 EPCCL 理论,这种方式只需要 4 次并行化,因此,本文结合 PIAE 采用该方式进行 IKCHER 算法的并行化.

算法 **4**. P-IKCHER.

1. 输入:子句集 $F=\{C_1,\ldots,C_n\}$
2. t=1
3. **Divide** F **to** $\{F_1,\ldots,F_k\}$
4. **Parallel**
5. **Foreach** $F_h(1\leq h\leq k)$
6. $E_h=CKCHER(F_h)$
7. **End parallel**
8. $E=PIAE(E_1,\ldots,E_k)$
9. **If** t=1 **Then**
10. t−−
11. $F=E$
12. **Goto** 3
13. **Else return** E

Sub-procedure CKCHER (CNF $F=\{C_1,\ldots,C_n\}$)

1. h=1
2. 初始化:$F_2=\{\neg C_1\}$, $F_3=\varnothing$, i=2, j=k=1

3. **While** $i \le |F_1|$
4. S_i={¬C_i 的互补展开} //$J_M(\square)-J_M(C_i)$,变量集 M 包含了 F 中出现的所有变量
5. **While** $j \le |F_2|$
6. **If** C_i 与 C_j 互补 **Then skip**
7. **Else**
8. **While** $k \le |C_i|$
9. **If** $S_i[k]$与 C_j 互补 **Then skip**
10. **Else if** $S_i[k] \models C_j$ **Then** $\Sigma_3=\Sigma_3\cup\{C_j\}$
11. **Else if** $C_j \models S_i[k]$ **Then** $\Sigma_3=\Sigma_3\cup\{S_i[k]\}$
12. **Else** $\Sigma_3=\Sigma_3\cup\{C_j\vee(S_i[k]-C_j)\}$
13. k++
14. j++
15. k=1
16. $F_2=F_3$
17. $F_3=\varnothing$
18. i++
19. j=1
20. **Return** F_2

算法 4 采用 PIAE 策略进行 EPCCL 理论的并行求交知识编译.该算法中,第 6 行的 CKCHER 子程序用来计算任意子句集所不能扩展出的极大项集,所得结果为 EPCCL 理论,该过程为 IKCHER 算法的子过程;第 8 行的 PIAE 子程序的具体过程参照第 3 节中的算法 2.P-IKCHER 算法采用的策略是:先将输入子句集分割成 k 个子集,体现在算法 4 的第 3 行;然后并行求得各个子集所不能扩展出的极大项集,所得结果为 EPCCL 理论,体现在算法 4 的第 4 行~第 7 行;最后,利用 PIAE 算法将所有的 EPCCL 理论求交,所得结果就是与输入子句集整体所不能扩展出的极大项集等价的 EPCCL 理论.另外,该算法中用 t 控制求解出与原子句集等价的 EPCCL 理论.

在利用扩展规则对子句集进行编译的过程中,子句的长度会随着编译的过程不断增加,子句的长度增加后,其所能扩展出的极大项数会减少,扩展能力会降低.P-IKCHER 算法在对输入子句集的子集编译后,采用 PIAE 求交策略对子任务所得的 EPCCL 理论进行合并.求交过程中,保留了大量子任务输出 EPCCL 理论中的长子句.这些长子句扩展能力低,因此会需要更多的长子句来得到与输入子句集等价的 EPCCL 理论.而且并行算法使用的核数越多,得到的长子句就越多,进而编译效果和编译效率会越差.因而可预见的是:利用 P-IKCHER 算法并行编译所得结果规模会增加,编译效率相对于串行程序会有所降低.

算法 **5**. impP-IKCHER.

1. 输入:子句集 $F=\{C_1,\ldots,C_n\}$
2. t=1
3. **Divide** F **to** $\{F_1,F_2\}$
4. $E=CKCHER(F_1)$
5. $F=imp\text{-}PIAE(E,F_2)$
6. **If** t=1 **Then**
7. t––
8. $F=E$
9. **Goto** 3
10. **Else return** F

算法 5 采用 imp-PIAE 策略进行 EPCCL 理论的并行求交知识编译.该算法中,CKCHER 子程序与算法 4 中

的 CKCHER 子程序执行过程相同,第 5 行的 imp-PIAE 子程序的具体过程参照第 3 节中的算法 3.在算法 5 中,首先求得输入子句集中部分子句所不能扩展出的极大项集,并将结果保存为 EPCCL 理论;然后对利用 imp-PIAE 算法对所得到的 EPCCL 理论和剩余子句集进行并行求交操作,得到输入子句集所不能扩展出的极大项对应的 EPCCL 理论;然后对该 EPCCL 理论进行相同的操作,得到与输入子句集等价的 EPCCL 理论.同样地,该算法中,用 t 控制求解出与原子句集等价的 EPCCL 理论.

impP-IKCHER 算法利用 imp-PIAE 策略对所得的 EPCCL 理论进行求交操作,根据定理 6 以及 IKCHER 算法的执行过程不难看出,impP-IKCHER 算法是将 IKCHER 算法中不影响最终结果的可以并行的操作并行化执行.因此可以预见的是,imp-IKCHER 算法不会改变 IKCHER 算法的编译结果的规模,而且还能提升 IKCHER 算法的编译效率.

5 IKCHER 算法并行效果的测试

UKCHER 算法是另一种可并行的知识编译算法,然而其并行化的工作尚未实现,并且其编译效率和编译质量远差于 IKCHER 算法(实验结果见第 3 节),因此本文仅对比测试了 impP-IKCHER 算法、P-IKCHER 算法和 IKCHER 算法的编译效果,以此验证 EPCCL 理论并行编译的可行性.本文按照子句数/变量数≈4.3 的比例生成了若干在相变点附近的 3-SAT 实例.并行算法用多核算法设计实现,用 k 表示任务的划分粒度.

由于 P-IKCHER 算法是本文设计的一种值得借鉴的有瑕疵的并行算法,该算法所使用的核数越多,效果越差,因此本节对该算法只测试 k=2 的并行效果.实验结果为 50 次实验的平均值,结果见表 7.

Table 7 Experimental results on random 3-SAT instances with n/m≈4.3

表 7 n/m≈4.3 的随机 3-SAT 实例测试

Instances	impP-IKCHER (k=4)		impP-IKCHER (k=2)		IKCHER		DKCHER		P-IKCHER (k=2)	
	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)	Size	Time (s)
⟨26,111⟩	149	0.192	149	0.223	149	0.274	157	0.268	161	0.396
⟨27,116⟩	141	0.150	141	0.298	141	0.412	141	0.402	158	0.532
⟨28,120⟩	113	0.349	113	0.554	113	0.718	111	0.711	133	0.832
⟨29,124⟩	198	0.503	198	0.782	198	0.998	202	0.990	222	1.224
⟨30,128⟩	96	0.538	96	1.068	96	1.311	97	1.303	108	1.678
⟨31,133⟩	223	1.116	223	1.882	223	2.079	220	2.068	245	2.326
⟨32,137⟩	207	1.623	207	2.112	207	2.914	205	2.896	237	3.995
⟨33,141⟩	130	2.003	130	3.169	130	3.931	132	3.924	144	5.465
⟨34,146⟩	159	2.798	159	4.582	159	5.185	162	5.173	170	6.365
⟨35,150⟩	168	4.362	168	6.004	168	7.216	170	7.209	181	9.332
uf20-01	79	0.014	79	0.024	79	0.029	78	0.020	92	0.041
uf20-02	103	0.011	103	0.015	103	0.018	100	0.014	114	0.032
uf20-03	18	0.009	18	0.013	18	0.015	20	0.011	22	0.018
blockworld-anomaly	47	0.037	47	0.149	47	0.157	48	0.153	56	0.256
pigeon-hole-6	1	3.068	1	4.328	1	5.533	1	5.528	3	7.362
par8-1-c	62	0.026	62	0.033	62	0.034	64	0.033	72	0.051

从编译质量来看,表 7 中,对于所有实例,impP-IKCHER(k=4),impP-IKCHER(k=2)的编译质量与 IKCHER 算法均一样;P-IKCHER 算法相对于 IKCHER 算法,编译质量有所下降.这是由于 impP-IKCHER 算法先将部分子句集编译为 EPCCL 理论,然后在合并过程采用 imp-PIAE 对已编译的 EPCCL 理论以及剩余未编译的原始子句集进行并行求交,这一系列过程与 IKCHER 算法的执行流程类似,不同之处在于,impP-IKCHER 算法是将 IKCHER 算法中不影响最终结果的可以并行的操作并行化执行;P-IKCHER 算法先将原始子句集分为若干子集,然后求得这些子集所不能扩展出的 EPCCL 理论,再将这些 EPCCL 理论并行求交,再求得子集所不能扩展出的极大项后会改变原始子句集的结构,因此在合并之后很难得到与 IKCHER 算法相同的结果.而且由于 EPCCL 理论中子句的长度普遍要长于原始子句集中子句的长度,因此合并之后,P-IKCHER 算法包含了较多的长子句,这些长子句所能扩展出的极大项集的规模较小,因此需要更多的长子句来表示原有子句集,从而 P-IKCHER 算法的编译

规模会大于 impP-IKCHER 算法和 IKCHER 算法的编译规模.

从编译效率上看,表 7 中,impP-IKCHER(k=4)和 impP-IKCHER(k=2)均起到了一定的加速效果,然而加速效果并不明显,当 k=4 时,最大加速比仅为 2 倍(如表 1 中,⟨27,116⟩,⟨30,128⟩,⟨34,146⟩和 blockworld-anomaly 等实例),这是由于本文分配任务的策略采用的是按子句数划分,这种划分策略并未考虑到子句集本身的内在结构,因此负载均衡较差;P-IKCHER(k=2)未起到加速效果,反而降低了编译效率,其原因正在于它采用 PIAE 算法直接合并多个 EPCCL 理论,极大地增加了编译过程所需时间开销.

整体来看,impP-IKCHER 能够提高 P-IKCHER 算法的编译效率.原始 IKCHER 算法的编译效率稍弱于 DKCHER 算法的编译效率,但由于 impP-IKCHER 算法采用了合适的并行合并策略,使求交知识编译方法的编译效率得到了提高(当 k=2 时,impP-IKCHER 算法的编译效率已经明显高于 DKCHER 算法的编译效率).因此,采用合适的并行合并策略之后,IKCHER 算法的并行化是可行的,这一点能够为其他目标编译语言的并行编译提供借鉴.

6 结论与展望

本文提出了一种新的 EPCCL 理论编译算法——求交知识编译算法 IKCHER,并设计了两种 EPCCL 理论的并行求交算法 PIAE 和 imp-PIAE,其中,imp-PIAE 利用了输入 EPCCL 理论对于所不能扩展出极大项集的原始子句集.基于 PIAE 算法和 imp-PIAE 算法,提出了两种并行求交知识编译算法 P-IKCHER 和 impP-IKCHER.实验结果表明,IKCHER 算法有最优的编译质量以及接近最优的编译效率,而 impP-IKCHER 能够提高 IKCHER 算法的编译效率,P-IKCHER 算法的编译效率相对于 IKCHER 算法反而有所降低,说明了合适的合并策略对于知识编译算法的并行化至关重要.本文的研究成果验证了并行知识编译是可行的,该成果同时能够为其他目标语言编译算法的并行化提供借鉴.

未来,我们将研究如何进一步提高 EPCCL 理论合并算法的效率,并使 impP-IKCHER 算法有更广的应用范围.本文中,对输入子句集的分割采用了等子句数分割,而分割结果会影响负载均衡及并行算法的整体执行效率,因此,任务分割的启发式选择也将是下一步的研究重点.

References:

[1] Cook SA. The complexity of theorem-proving procedures. In: Harrison MA, Banerji RB, Ullman JD, eds. Proc. of the 3rd Annual ACM Symp. on the Theory of Computing (STOC'71). Shaker Heights: ACM Press, 1971. 151−158. [doi: 10.1145/800157.805047]

[2] Mitchell DG, Selman B, Levesque HJ. Hard and easy distributions of SAT problems. In: Swartout WR, ed. Proc. of the 10th National Conf. on Artificial Intelligence (AAAI'92). San Jose: AAAI Press, The MIT Press, 1992. 459−465.

[3] Bai S, Bu DB. Analysis for phase transition of the 2-3-SAT problem. Ruan Jian Xue Bao/Journal of Software, 1998,9(11):828−832 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/9/828.htm [doi: 10.13328/j.cnki.jos.1998.11.006]

[4] Cai SW, Su KL. Local search for Boolean satisfiability with configuration checking and subscore. Artificial Intelligence, 2013,204: 75−98. [doi: 10.1016/j.artint.2013.09.001]

[5] Luo C, Su KL, Cai SW. More efficient two-mode stochastic local search for random 3-satisfiability. Applied Intelligence, 2014, 41(3):665−680. [doi: 10.1007/s10489-014-0556-7]

[6] Luo C, Cai SW, Wu W, Su KL. Double configuration checking in stochastic local search for satisfiability. In: Brodley CE, Stone P, eds. Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI 2014). Québec: AAAI Press, 2014. 2703−2709.

[7] Cai SW, Su KL. Comprehensive score: Towards efficient local search for SAT with long clauses. In: Rossi F, ed. Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI Press, 2013. 489−495.

[8] Jeavons P, Petke J. Local consistency and SAT-solvers. Journal of Artificial Intelligence Research, 2012,43:329−351. [doi: 10.1613/jair.3531]

[9] Katsirelos G, Sabharwal A, Samulowitz H, Simon L. Resolution and parallelizability: Barriers to the efficient parallelization of SAT solvers. In: desJardins M, Littman ML, eds. Proc. of the 27th AAAI Conf. on Artificial Intelligence (AAAI 2013). Washington: AAAI Press, 2013.

[10] Audemard G, Simon L. Lazy clause exchange policy for parallel SAT solvers. In Sinz C, Egly U, eds. Proc. of the 17th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS 8561, Vienna: Springer-Verlag, 2014. 197−205. [doi: 10.1007/978-3-319-09284-3_15]

[11] Sonobe T, Kondoh S, Inaba M. Community branching for parallel portfolio SAT solvers. In Sinz C, Egly U, eds. Proc. of the 17th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS 8561, Vienna: Springer-Verlag, 2014. 188−196. [doi: 10.1007/978-3-319-09284-3_14]

[12] Val DA. On some tractable classes in deduction and abduction. Artificial Intelligence, 2000,116(1-2):297−313. [doi: 10.1016/S0004-3702(99)00088-0]

[13] Broeck GV, Darwiche A. On the role of canonicity in knowledge compilation. In: Bonet B, Koenig S, eds. Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI 2015). Austin: AAAI Press, 2015. 1641−1648.

[14] Marquis P. Existential closures for knowledge compilation. In: Walsh T, ed. Proc. of the 22nd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2011). Barcelona: IJCAI/AAAI Press, 2011. 996−1001. [doi: 10.5591/978-1-57735-516-8/IJCAI11-171]

[15] Fargier H, Marquis P. Disjunctive closures for knowledge compilation. Artificial Intelligence, 2014,216:129−162. [doi: 10.1016/j.artint.2014.07.004]

[16] Darwiche A. Compiling knowledge into decomposable negation normal form. In: Dean T, ed. Proc. of the 16th Int'l Joint Conf. on Artificial Intelligence (IJCAI'99). Stockholm: Morgan Kaufmann Publishers, 1999. 284−289.

[17] Darwiche A. Decomposable negation normal form. Journal of the ACM, 2001,48(4):608−647. [doi: 10.1145/502090.502091]

[18] Darwiche A, Marquis P. A knowledge compilation map. Journal of Artificial Intelligence Research, 2002,17:229−264. [doi: 10.1613/jair.989]

[19] Fargier H, Marquis P. Extending the knowledge compilation map: Krom, Horn, Affine and beyond. In: Fox D, Gomes CP, eds. Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008). Chicago: AAAI Press, 2008. 442−447.

[20] Fargier H, Marquis P, Niveau A. Towards a knowledge compilation map for heterogeneous representation languages. In: Rossi F, ed. Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI Press, 2013. 877−883.

[21] Fargier H, Marquis P, Niveau A, Schmidt N. A knowledge compilation map for ordered real-valued decision diagrams. In: Brodley CE, Stone P, eds. Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI 2014). Québec: AAAI Press, 2014. 1049−1055.

[22] Koriche F, Lagniez JM, Marquis P, Thomas S. Knowledge compilation for model counting: Affine decision trees. In: Rossi F, ed. Proc. of the 23rd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI Press, 2013. 947−953.

[23] Lai Y, Liu DY, Wang SS. Reduced ordered binary decision diagram with implied literals: A new knowledge compilation approach. Knowledge and Information Systems, 2013,35(3):665−712. [doi: 10.1007/s10115-012-0525-6]

[24] Huang JB, Darwiche A. The language of search. Journal of Artificial Intelligence Research, 2007,29:191−219. [doi: 10.1613/jair.2097]

[25] Lin H, Sun JG, Zhang YM. Theorem proving based on the extension rule. Journal of Automated Reasoning, 2003,31(1):11−21. [doi: 10.1023/A:1027339205632]

[26] Yin MH, Lin H, Sun JG. Solving #SAT using extension rules. Ruan Jian Xue Bao/Journal of Software, 2009,20(7):1714−1725 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]

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

[28] Li Y, Sun JG, Wu X, Zhu XJ. Extension rule algorithms based on IMOM and IBOHM heuristics strategies. Ruan Jian Xue Bao/Journal of Software, 2009,20(6):1521−1527 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3420.htm [doi: 10.3724/SP.J.1001.2009.03420]

[29] Sun JG, Li Y, Zhu XJ, Lü S. A novel theorem proving algorithm based on extension rule. Journal of Computer Research and Development, 2009,46(1):9−14 (in Chinese with English abstract).

[30] Lin H, Sun JG. Knowledge compilation using the extension rule. Journal of Automated Reasoning, 2004,32(2):93−102. [doi: 10.1023/B:JARS.0000029959.45572.44]

[31] Yin MH, Lin H, Sun JG. Counting models using extension rules. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007). Vancouver: AAAI Press, 2007. 1916−1917.

[32] 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 (in Chinese with English abstract).

[33] Liu DY, Lai Y, Lin H. C2E: An EPCCL compiler with good performance. Chinese Journal of Computers, 2013,36(6):1254−1260 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2013.01254]

[34] Liu L, Niu DD, Li Z, Lü S. Dynamic online reasoning algorithm based on the hyper extension rule. Journal of Harbin Engineering University, 2015,36(12):1614−1619 (in Chinese with English abstract). [doi: 10.11990/jheu.201404055]

[35] Liu L, Niu DD, Lü S. Knowledge compilation methods based on the hyper extension rule. Chinese Journal of Computers, 2016, 39(8):1681−1696 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2016.01681]

附中文参考文献:

[3] 白硕,卜东波.2-3-SAT 问题相变现象剖析及其应用.软件学报,1998,9(11):828−832. http://www.jos.org.cn/1000-9825/9/828.htm [doi: 10.13328/j.cnki.jos.1998.11.006]

[26] 殷明浩,林海,孙吉贵.一种基于扩展规则的#SAT 求解系统.软件学报,2009,20(7):1714−1725. http://www.jos.org.cn/1000-9825/3320.htm [doi: 10.3724/SP.J.1001.2009.03320]

[27] 赖永,欧阳丹彤,蔡敦波,吕帅.基于扩展规则的模型计数与智能规划方法.计算机研究与发展,2009,46(3):459−469.

[28] 李莹,孙吉贵,吴瑕,朱兴军.基于 IMOM 和 IBOHM 启发式策略的扩展规则算法.软件学报,2009,20(6):1521−1527. http://www.jos.org.cn/1000-9825/3420.htm [doi: 10.3724/SP.J.1001.2009.03420]

[29] 孙吉贵,李莹,朱兴军,吕帅.一种新的基于扩展规则的定理证明方法.计算机研究与发展,2009,46(1):9−14.

[32] 谷文祥,王金艳,殷明浩.基于 MCN 和 MO 启发式策略的扩展规则知识编译方法.计算机研究与发展,2011,48(11):2064−2073.

[33] 刘大有,赖永,林海.C2E:一个高性能的 EPCCL 理论编译器.计算机学报,2013,36(6):1254−1260. [doi: 10.3724/SP.J.1016.2013.01254]

[34] 刘磊,牛当当,李壮,吕帅.基于超扩展规则的动态在线推理算法.哈尔滨工程大学学报,2015,36(12):1614−1619. [doi: 10.11990/jheu.201404055]

[35] 刘磊,牛当当,吕帅.基于超扩展规则的知识编译方法.计算机学报,2016,39(8):1681−1696. [doi: 10.11897/SP.J.1016.2016.01681]

牛当当(1990－),男,陕西西安人,博士生,CCF 学生会员,主要研究领域为智能规划,自动推理.

刘磊(1960－),男,教授,博士生导师,CCF 专业会员,主要研究领域为软件理论与技术.

吕帅(1981－),男,博士,副教授,CCF 高级会员,主要研究领域为智能规划,自动推理.