基于超扩展规则的知识编译方法

刘 磊[1] 牛当当[1] 吕 帅[1),2),3)]

[1)](吉林大学计算机科学与技术学院 长春 130012)

[2)](吉林大学数学学院 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) 长春 130012)

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与差集,并将所得结果以 EPCCL 理论的形式保存.该文首次提出了扩展反驳方法,是一种新型推理方法,并在该推理方法与知识编译之间建立了联系.基于超扩展规则的性质,该文还提出了两种知识编译算法:求并知识编译算法 UKCHER 和求差知识编译算法 DKCHER,是两种新的知识编译算法.算法 UKCHER 是目前为止唯一一个可并行的 EPCCL 理论编译算法,算法 DKCHER 对于相变点附近的难解问题具有较高的编译效率和编译质量.实验结果表明:UKCHER 算法的编译效率和编译质量均优于 Lin 等人提出的 KCER 算法;当子句数和变量数的比值较大时,DKCHER 算法的编译效率和编译质量是最优的,相比于现有 EPCCL 理论编译算法,该算法具有较强的竞争力.

关键词 知识编译;扩展规则;超扩展规则;EPCCL 理论;扩展反驳

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2016.01681

Knowledge Compilation Methods Based on the Hyper Extension Rule

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

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

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

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

Abstract Hyper extension rule is the expansion of extension rule. We can compute the union and difference set of two sets of maximum terms which are extended by two clauses respectively based on the hyper extension rule, and the results are saved as EPCCL theories. In this paper, we propose extension refutation, which is a novel reasoning method. A link between extension refutation and knowledge compilation is also established. We also introduce two knowledge compilation methods based on hyper extension rule: the algorithm UKCHER of computing the union set and the algorithm DKCHER of calculating the difference set, which are two novel knowledge compilation methods. UKCHER is the only one of the EPCCL compilers that can be parallelized, and the efficiency and quality of DKCHER are excellent for the difficult problems around phase transition point. Experimental results show that, the efficiency and quality of UKCHER are all superior to the algorithm KCER proposed by Lin et al.; when the ratio of the clause number and the variable number is bigger, the efficiency and quality of DKCHER is better.

收稿日期:2014-11-06;在线出版日期:2015-07-01.本课题得到国家自然科学基金(61300049)、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)、吉林省青年科研基金项目(20140520069JH,20150520058JH,20150520060JH)和吉林省自然科学基金项目(20150101054JC)资助.**刘 磊**,男,1960 年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为软件理论与技术.E-mail:liulei@jlu.edu.cn.**牛当当**,男,1990 年生,博士研究生,主要研究方向为智能规划与自动推理.**吕 帅**(通信作者),男,1981 年生,博士,副教授,中国计算机学会(CCF)高级会员,CCF 理论计算机科学专委会委员,主要研究方向为智能规划与自动推理.E-mail:lus@jlu.edu.cn.

DKCHER has strong competitiveness comparing with other existing knowledge compilation algorithms of EPCCL theory.

Keywords knowledge compilation; extension rule; hyper extension rule; EPCCL theory; extension refutation

1 引 言

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得 SAT 求解得以高效实现[4-8].然而 SAT 求解始终需要指数级的时间,而且通过多次调用 SAT 求解器对一个知识库进行多次查询显然也是不合适的.同时许多约束问题仅判定可满足性是远远不够的,如将概率推理问题[9]、一致性概率规划问题[10]转换为命题公式集后,除了判定可满足性之外,还需解决该公式集的模型计数问题,即计算所有模型(可满足的真值指派)个数,模型计数问题也称为 #SAT 问题. #SAT 问题是概率推理中的重要问题,它能在多项式时间内与贝叶斯推理进行相互转换,在扩展规则推理方法提出之前,绝大多数 #SAT 问题的求解算法都是基于 #DPLL 算法设计的[11].

知识编译的主要目的是为了提高某些任务的计算效率,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.知识编译方法能够使得命题推理和模型计数等问题得以高效解决,原因在于:在线推理通常能够在多项式时间内完成,而对编译结果进行多次查询能够补偿编译过程所消耗的时间. del Val[12]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. Marquis[13]提出了一种基于本原蕴含式的知识编译方法,并研究了几种不同目标编译语言中存在的闭包[14-15]. Darwiche[16-17]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,并依据编译语言的简洁性及编译语言在多项式时间内所支持的查询类型两个基本关注点,对不同编译方法进行分析,给出了知识编译的发展图谱[18]. Fargier 等人[19]基于 Krom、Horn 和 Affine 等 3 种较有影响力的命题片段对 Darwiche 提出的知识编译发展图谱做了扩充,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[22]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. Lai 等人[23]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.目前,国际上针对知识编译的研究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.

2003 年,Lin 等人[24]提出了扩展规则推理方法(Extension Rule,ER),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[24]基于扩展规则提出了一种新的命题逻辑理论证明方法 IER.殷明浩等人[25]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.赖永等人[26]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.许有军等人[27]提出了间接应用扩展规则的 MCEHST 算法,适应子句长度短,子句集规模大的问题.李莹等人[28]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.此外,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[30]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.此外,Yin 等人[25,31]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.谷文祥等人[32]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.刘大有等人[33]证明了 EPCCL 理论能够满足文献[18]所提出的所有查询标准,并基于扩展规则提出了一种新的 EPCCL 理论编译器 C2E,该编译器具有较高的编译效率.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.该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，因此提高 EPCCL 编译器的编译效率及编译质量具有重要意义.

本文基于自身提出的超扩展规则①[34]，提出了扩展反驳的概念，利用超扩展规则能够实现输入子句集的扩展反驳过程，并在扩展反驳过程与基于知识编译之间建立了联系. 由于现有知识编译算法存在一些缺陷，本文基于超扩展规则提出了两种知识编译算法：UKCHER 算法和 DKCHER 算法，这两种算法具有不同的适用范围，且相比于现有 EPCCL 理论编译算法具有较大的优势.

本文第 2 节介绍超扩展规则的基本概念和性质；第 3 节提出了扩展反驳方法；第 4 节阐述求并知识编译的基本原理并设计知识编译算法 UKCHER；第 5 节阐明反向求差知识编译的基本原理并设计知识编译算法 DKCHER；第 6 节为实验部分，对本文提出的两种知识编译算法与现有 EPCCL 理论知识编译算法进行了比较；最后为本文总结.

2　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，是一种应用于两个子句之间的经典扩展规则的扩展形式.

定义 1(扩展规则，Extension Rule)[24]. 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].

扩展规则要求一个子句结合一个变量进行扩展，超扩展规则允许一个子句结合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 $J(C)$，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，$C\equiv J(C)$.

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 $A\not\models C$，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.

定理 1. 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.

证明. 可以通过真值表证明子句 C 及其扩展结果语义上是等价. 证毕.

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 $A=\{b_1,\cdots,b_n\}$，则运用德摩根律，$E=\{C\vee\neg A\}=\{C\vee\neg(b_1\vee\cdots\vee b_n)\}=\{C\vee\neg b_1,\cdots,C\vee\neg b_n\}$. 上述展开过程是语义等价的，但由于每个 $b_i(i=1,\cdots,n)$ 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：

$$\frac{\neg(b_1\vee\cdots\vee b_n)}{(\neg b_1)}$$
$$(b_1\vee\neg b_2)\qquad(1)$$
$$\cdots$$
$$(b_1\vee\cdots\vee b_{n-1}\vee\neg b_n)$$

称式(1)的展开过程为互补展开，则 $E=\{C\vee\neg A\}=\{C\vee\neg b_1, C\vee b_1\vee\neg b_2,\cdots, C\vee b_1\vee\cdots\vee b_{n-1}\vee\neg b_n\}$，进而 $D=\{C\vee b_1\vee\cdots\vee b_n, C\vee\neg b_1, C\vee b_1\vee\neg b_2,\cdots, C\vee b_1\vee\cdots\vee b_{n-1}\vee\neg b_n\}$.

定理 2. 依赖式(1)的互补展开保证了超扩展规则的等价性，并且展开结果子句之间存在互补文字对.

证明. 假设 $A=\{b_1,\cdots,b_n\}$，由于 C 本身对扩展

① 本文涉及的"超扩展规则"概念是我们的前期工作，用于构造新型的扩展规则推理方法，相关论文中已经详细描述，详见文献[34]，故不作为本文核心工作阐述.

结果性质没有影响，只需要证明：(1) $\neg A=\neg(b_1\vee\cdots\vee b_n)$ 与 $Z=\{\{\neg b_1\},\{b_1\vee\neg b_2\},\cdots,\{b_1\vee\cdots\vee b_{n-1}\vee\neg b_n\}\}$ 等价；和(2) $\{\neg b_1, b_1\vee\neg b_2,\cdots, b_1\vee\cdots\vee b_{n-1}\vee\neg b_n\}$ 子句之间存在互补文字对. 其中(2)显然成立. 下面证明(1).

证明. 若 $\neg A$ 为真，则 A 中所有文字均被弄假，而 Z 中每个子句均包含 A 中一个文字的否定，因此 Z 为真.

若 $\neg A$ 为假，假设 Z 为真，则 Z 中每个子句均为真. 为使第 1 个子句 $\neg b_1$ 为真，则 b_1 必须被弄假；同样，为使第 2 个子句 $b_1\vee\neg b_2$ 为真，而 b_1 为假，则 b_2 必须被弄假；依次传播到 $b_1\vee\cdots\vee b_{n-1}\vee\neg b_n$，则 b_n 必须被弄假. 因此，若 Z 为真，则所有 $b_i(i=1,\cdots,n)$ 均为假，进而 A 为假，$\neg A$ 为真，与假设矛盾. 因此，若 $\neg A$ 为假，则 Z 为假.

综上，结论(1)得证，进而定理 2 成立. 证毕.

也可以采用式(2)的形式描述互补展开.

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

定义 3(EPCCL 理论)[30]. 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. 通过结合 EPCCL 理论，本文进一步挖掘了超扩展规则的其他特殊性质：

性质 1. 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. 基于超扩展规则可以计算任意两个子句所能扩展出极大项的交集.

对于两个子句 C 和 A，如果 $A\models C$，且 A 和 C 不包含互补文字对，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 $J(C)\cap J(A)$.

性质 3. 基于超扩展规则能够利用 EPCCL 理论保存两个子句所能扩展极大项集合的差集.

对于两个子句 C 和 A，如果 $A\models C$，且 A 和 C 不包含互补文字对，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 $J(C)-J(A)$.

3 扩展反驳过程

归结反驳的基本含义为：若判断 $F\models C$，只需判断对于子句集 $F\cup\{\neg C\}$ 是否存在一个归结过程，该过程能够得到一个空子句. 然而对于互补因子高的问题，归结反驳过程需要做大量归结，其求解效率较低. 扩展规则能够用于求解互补因子较高的问题，然而利用扩展规则进行反驳需要定义新的反驳过程.

引理 1[24]. 给定子句集 F 及其原子集 M ($|M|=m$)，若 F 中所有子句均为 M 上的极大项，则 F 不可满足当且仅当 $|F|=2^m$.

基于引理 1，本文提出了扩展反驳过程的基本定理.

定理 3. 给定子句集 F，C 是待查询的子句，F_1 是通过对 $F\cup\{\neg C\}$ 等价变换得到的子句集，且 F_1 中所有子句均为 $M=V(F)\cup V(C)$ 上的极大项 ($|M|=m$)，则 $F\models C$ 当且仅当 $|F_1|=2^m$.

证明. 若要 $F\models C$ 判断是否成立，只需判断 $F\cup\{\neg C\}$ 是否可满足. 若 $F\cup\{\neg C\}$ 不可满足，则 $F\models C$ 成立，否则不成立. 若 $|F_1|=2^m$，由于 F_1 是由 $F\cup\{\neg C\}$ 等价变换所得，根据引理 1，有 $F\cup\{\neg C\}$ 是不可满足的，因此 $F\models C$ 成立. 若 $F\models C$ 成立，则 $F\cup\{\neg C\}$ 不可满足，由于 F_1 与 $F\cup\{\neg C\}$ 等价且 F_1 中所有子句均为 M 上的极大项，因此必然有 $|F_1|=2^m$. 证毕.

由定理 3 可以得到一个基于扩展规则 $F\models C$ 的扩展反驳扩展：首先将子句 C 取否定之后加入 F 中，然后将 $F\cup\{\neg C\}$ 所有子句利用扩展规则扩展为 $V(F\cup\{\neg C\})$ 上的极大项，并将这些极大项保存在集合 F_1 中，接着去掉 F_1 中所有重复的极大项，最后通过判断 F_1 的模来决策 $F\models C$ 是否成立.

上述过程是一个基本的扩展反驳过程，其终止条件为将 $F\cup\{\neg C\}$ 扩展为由极大项组成的子句集. 然而将 $F\cup\{\neg C\}$ 中所有子句扩展为极大项形式进行计算是不现实的，因为这会导致指数级的时间复杂度和空间复杂度. Lin 等人[20] 基于容斥原理提出了 ER 算法解决了这一问题，然而 ER 算法并没有真正地对子句进行扩展，而仅仅通过容斥原理对子句集所能扩展出的极大项集的个数进行了计算. 本文将利用超扩展规则对子句集扩展，并利用

EPCCL 理论的特性定义扩展反驳过程的终止条件.

推论 1.　给定子句集 F 及任意子句 C，则 $F\models C$ 当且仅当 $|J(F)\cup J(\{\neg C\})|=2^m$，其中 $J(F)$ 和 $J(\{\neg C\})$ 均为 $V(F)\cup V(C)$ 上的极大项，且 $|V(F)\cup V(C)|=m$.

由推论 1 可知，扩展反驳过程并不需要将所有子句扩展为极大项的形式. 对扩展过程加以控制，如果得到某个中间结果使得计算子句集所能扩展出的极大项数变得易处理，则可以停止扩展过程，这种中间结果的最佳形式为 EPCCL 理论，因此，本文对扩展反驳过程做了改进.

引理 2[24].　给定任意两个子句 C_i 和 C_j，变量集 M 满足 $(V(C_i)\cup V(C_j))\subseteq M$，$C_i$ 和 C_j 在 M 上所能扩展出的极大项集分别为 $J(C_i)$ 和 $J(C_j)$，C_i 与 C_j 互补当且仅当 $J(C_i)\cap J(C_j)=\varnothing$.

推论 2.　给定任意两个子句 C_i 和 C_j，变量集 M 满足 $(V(C_i)\cup V(C_j))\subseteq M$，$C_i$ 和 C_j 在 M 上所能扩展出的极大项集分别为 $J(C_i)$ 和 $J(C_j)$，若 C_i 与 C_j 互补，则 $|J(C_i)\cup J(C_j)|=|J(C_i)|+|J(C_j)|$ 且 $J(C_i)-J(C_j)=J(C_i)$.

由推论 2 可以得出：对于两个互补子句 C_i 和 C_j，显然 $\{C_i,C_j\}$ 就是一个与 $J(C_i)\cup J(C_j)$ 等价的 EPCCL 理论.

推论 3.　给定子句集 F 和子句 C，$J(F)$ 和 $J(C)$ 均为 $V(F)\cup V(C)$ 上的极大项集. 若 $J(C)\cap J(F)=\varnothing$，则 C 与 F 中所有子句互补.

证明.　利用反证法，假设存在一个子句 $D\in F$，且 D 与 C 非互补，则根据引理 2 有 $J(D)\cap J(C)\neq\varnothing$. 又因为 $D\in F$，则 $J(D)\subseteq J(F)$，因此必然有 $J(C)\cap J(F)\neq\varnothing$，与条件不符，推论 3 成立. 证毕.

定理 4.　给定子句集 F，C 是待查询的子句，$|V(F)\cup V(C)|=m$，$F_1=\{C_1,\cdots,C_n\}$ 是通过对 $F\cup\{\neg C\}$ 等价变换得到的子句集，且 F_1 中是一个 EPCCL 理论，则 $F\models C$ 当且仅当 $\sum\limits_{1\leq i\leq n}2^{m-|C_i|}=2^m$.

证明.　根据引理 2 有 F_1 所能扩展出的极大项数为 $\sum\limits_{1\leq i\leq n}2^{m-|C_i|}$，则根据推论 1，定理 4 显然成立. 证毕.

事实上，基于知识编译能够快速实现扩展反驳，将子句集 F 编译为等价的 EPCCL 理论之后，就能够在多项式时间内将 $F\cup\{\neg C\}$ 等价变换为 EPCCL 理论，KCER 算法[26] 和 C2E 算法[28] 是两种已有的 EPCCL 理论编译算法，然而二者都不能并行，且很多情况下编译过程需要较长时间. 本文基于超扩展规则设计了求并知识编译算法 UKCHER 和求差知识编译算法 DKCHER，其中算法 UKCHER 可以并行实现.

4　基于超扩展规则的求并知识编译

定理 5.　给定任意两个子句 C_i 和 C_j，变量集 M 满足 $(V(C_i)\cup V(C_j))\subseteq M$，$C_i$ 和 C_j 在 M 上所能扩展出的极大项集分别为 $J(C_i)$ 和 $J(C_j)$，若 $C_i\models C_j$，则 $J(C_i)\cup J(C_j)=J(C_i)$ 且 $J(C_j)-J(C_i)=\varnothing$.

证明.　因为 $C_i\models C_j$，则 $V(C_i)\subseteq V(C_j)$，且 C_i 和 C_j 之间不存在互补文字，则有 $J(C_j)\subseteq J(C_i)$ 成立，因此 $J(C_i)\cup J(C_j)=J(C_i)$ 成立且 $J(C_j)-J(C_i)=\varnothing$ 成立. 证毕.

定理 6.　给定任意两个子句 C_i 和 C_j，变量集 M 满足 $(V(C_i)\cup V(C_j))\subseteq M$，$C_i$ 和 C_j 在 M 上所能扩展出的极大项集分别为 $J(C_i)$ 和 $J(C_j)$，若 $C_i\not\models C_j$，$C_j\not\models C_i$，且 C_i 和 C_j 之间不存在互补文字，则利用超扩展规则可以将 $J(C_i)\cup J(C_j)$ 等价表示为 $S_1=\{C_i,C_j\vee\neg(V(C_i)-V(C_j))\}$ 或 $S_2=\{C_j,C_i\vee\neg(V(C_j)-V(C_i))\}$ 或 $S_3=\{C_i\vee\neg(V(C_j)-V(C_i)),C_j\vee\neg(V(C_i)-V(C_j)),C_i\vee(V(C_j)-V(C_i))\}$ 或 $S_4=\{C_i\vee\neg(V(C_j)-V(C_i)),C_j\vee\neg(V(C_i)-V(C_j)),C_j\vee(V(C_i)-V(C_j))\}$，其中，$S_1\sim S_4$ 均为 EPCCL 理论，且 S_3 和 S_4 为相同公式的不同抽象形式.

证明.　首先 $J(C_i)\cup J(C_j)=J(C_i)\cup(J(C_j)-J(C_i))=J(C_j)\cup(J(C_i)-J(C_j))=(J(C_i)-J(C_j))\cup(J(C_j)-J(C_i))\cup(J(C_j)\cap J(C_i))$.

因为 $C_i\not\models C_j$，且 C_i 和 C_j 之间不存在互补文字，根据超扩展规则，$J(C_j)-J(C_i)\equiv C_j\vee\neg(V(C_i)-V(C_j))$，则 $J(C_i)\cup J(C_j)=J(C_i)\cup(J(C_j)-J(C_i))\equiv\{C_i,C_j\vee\neg(V(C_i)-V(C_j))\}$ 成立，即 $J(C_i)\cup J(C_j)\equiv S_1$. 同理 $J(C_i)-J(C_j)\equiv C_i\vee\neg(V(C_j)-V(C_i))$，则 $J(C_i)\cup J(C_j)=J(C_j)\cup(J(C_i)-J(C_j))\equiv\{C_j,C_i\vee\neg(V(C_j)-V(C_i))\}$ 成立，即 $J(C_i)\cup J(C_j)\equiv S_2$. 由于 $C_i\vee(V(C_j)-V(C_i))\equiv C_j\vee(V(C_i)-V(C_j))\equiv J(C_j)\cap J(C_i)$，且 $C_i\vee(V(C_j)-V(C_i))$ 和 $C_j\vee(V(C_i)-V(C_j))$ 均为单个子句，则 $C_i\vee(V(C_j)-V(C_i))=C_j\vee(V(C_i)-V(C_j))$. 因此有 $\{C_i\vee\neg(V(C_j)-V(C_i)),C_j\vee\neg(V(C_i)-V(C_j)),C_i\vee(V(C_j)-V(C_i))\}=\{C_i\vee\neg(V(C_j)-V(C_i)),C_j\vee\neg(V(C_i)-V(C_j)),C_j\vee(V(C_i)-V(C_j))\}\equiv J(C_i)\cup J(C_j)$ 成立，即 $J(C_i)\cup J(C_j)\equiv S_3=S_4$.

根据超扩展规则的性质 3，$C_j \vee \neg(V(C_i)-V(C_j))$展开后为一个 EPCCL 理论，又因为 $J(C_i)\cap(J(C_j)-J(C_i))=\varnothing$，因此根据推论 3，$C_i$与 $C_j \vee \neg(V(C_i)-V(C_j))$中所有子句均互补，因此 $S_1=\{C_i, C_j \vee \neg(V(C_i)-V(C_j))\}$中所有子句均互补，$S_1$是一个 EPCCL 理论，同理可证得 $S_2=\{C_j, C_i \vee \neg(V(C_j)-V(C_i))\}$是一个 EPCCL 理论. 显然 $(J(C_i)-J(C_j))$，$(J(C_j)-J(C_i))$和 $(J(C_j)\cap J(C_i))$这 3 个集合中两两不相交，因此 $\{C_i \vee \neg(V(C_j)-V(C_i)), C_j \vee \neg(V(C_i)-V(C_j)), C_i \vee (V(C_j)-V(C_i))\}$中所有子句之间均互补，即 S_3为一个 EPCCL 理论，由于 $S_3=S_4$，因此 S_4同样是一个 EPCCL 理论.

综上，定理 6 成立. 证毕.

由于 S_3和 S_4的计算结果一样，对于非互补且不存在蕴含关系的两个子句，定理 6 实际上给出了求解两个子句所能扩展出极大项集并集的 3 种基本方法，且所得结果均为 EPCCL 理论. 根据超扩展规则，显然 $|S_1|=1+|V(C_i)-V(C_j)|$，$|S_2|=1+|V(C_j)-V(C_i)|$，$|S_3|=1+|V(C_i)-V(C_j)|+|V(C_j)-V(C_i)|$. 在计算 EPCCL 理论所能扩展出的极大项集时，计算过程所需时间与 EPCCL 理论的规模成正比，因此，使用 S_3的计算方式是不合适的. 对于非互补且不存在蕴含关系的两个子句，在计算它们所能扩展出的极大项集的并集时，可以首先求解 $\min\{|V(C_i)-V(C_j)|, |V(C_j)-V(C_i)|\}$，然后再决策使用 S_1和 S_2中哪种求解方法来使结果的规模最小.

推论 2、定理 5 和定理 6 给出了任意两个子句所能扩展出极大项集的并集的求解方法，然而，在将任意两个子句所能扩展出的极大项集的并集表示为 EPCCL 理论之后，需要对 EPCCL 理论和一个子句进行求并操作，即求解 EPCCL 理论所能扩展出的极大项集和任意子句所能扩展出的极大项集的并集.

定理 7. 给定 EPCCL 理论 $E=\{C_1,\cdots,C_e\}$和子句 C，则 $J(E)\cup J(C)$可以表示为 $E_1=\{C\}\cup\bigcup_{1\le i\le e}(J(C_i)-J(C))$或 $E_2=E\cup(J(C)-J(C_1)-\cdots-J(C_e))$. E_1和 E_2为使用推论 2、定理 5 和超扩展规则计算所得结果，则 E_1和 E_2均为 EPCCL 理论.

证明. 首先 $J(E)\cup J(C)=J(C)\cup(J(E)-J(C))=J(E)\cup(J(C)-J(E))=(J(E)-J(C))\cup(J(C)-J(E))\cup(J(E)\cap J(C))$，其中，最后一项由于计算结果规模较大，因此定理 7 中不予考虑该种转换形式.

(1) $J(E)=\bigcup_{1\le i\le e}J(C_i)$，$J(E)-J(C)=\bigcup_{1\le i\le e}(J(C_i)-J(C))$，因此 $J(E)\cup J(C)$可以表示为 $E_1=\{C\}\cup\bigcup_{1\le i\le e}(J(C_i)-J(C))$，同理可得 $J(E)\cup J(C)$可以表示为 $E_2=E\cup(J(C)-J(C_1)-\cdots-J(C_e))$.

(2) E为EPCCL理论，则$J(C_i)\cap J(C_j)=\varnothing(1\le i,j\le e)$，进而$(J(C_i)-J(C))\cap(J(C_j)-J(C))=\varnothing$. 又由于对于任意 $J(C_i)-J(C)$使用推论 2、定理 5 和超扩展规则所得结果为 EPCCL 理论，因此 $\bigcup_{1\le i\le e}(J(C_i)-J(C))$的计算结果为一个 EPCCL 理论，显然 C与 $\bigcup_{1\le i\le e}(J(C_i)-J(C))$极大项的交集为空，因此根据推论 3，$\{C\}\cup\bigcup_{1\le i\le e}(J(C_i)-J(C))$是一个 EPCCL 理论.

(3) 利用数学归纳法可证得 $J(C)-J(C_1)-\cdots-J(C_e)$为一个 EPCCL 理论. 首先 $J(C)-J(C_1)$的计算结果是一个 EPCCL 理论，假设 $1\le i<e$，且 $J(C)-J(C_1)-\cdots-J(C_i)$是一个 EPCCL 理论，则依据(2)可以合理推断出$(J(C)-J(C_1)-\cdots-J(C_i))-J(C_{i+1})$是一个EPCCL理论. 因此$J(C)-J(C_1)-\cdots-J(C_e)$的计算结果为一个 EPCCL 理论. 由于 E 是一个 EPCCL 理论，且 $J(E)\cap(J(C)-J(C_1)-\cdots-J(C_e))=\varnothing$，因此 $E\cup(J(C)-J(C_1)-\cdots-J(C_e))$为一个 EPCCL 理论.

综上，定理 7 成立. 证毕.

利用定理 7 中可以求得任意一个 EPCCL 理论 E 和一个子句 C 所能扩展出极大项集的并集，E_1和 E_2是两种语义等价的结果. 显然大多数情况下，E_1的规模小于 E_2的规模，只有在 E 中存在一个子句 $C_i\models C$ 的时候，$|E_1|=|E_2|+|V(C)-V(C_i)|$，因此在扩展过程中，本文选择 E_1作为扩展策略.

推论 2、定理 5 以及定理 6 给出了任意两个子句所能扩展出极大项集的并集求解方法，定理 7 则给出了任意 EPCCL 理论与任意子句所能扩展出极大项集的并集的求解方法. 基于上述思想，本文设计了基于超扩展规则的求并知识编译算法 UKCHER，该算法描述如算法 1.

算法 1. UKCHER.

1. 输入：子句集 $F=\{C_1,\cdots,C_n\}$
2. 初始化：令 $F_1=\varnothing$，$i=j=1$
3. WHILE $i\le|F|$

(a) WHILE $j\leq|F_1|$

i. IF C_i 与 C_j 互补 THEN *skip*

ii. ELSE IF $C_i\models C_j$ THEN $F_1=F_1-\{C_j\}$

iii. ELSE $C_j=\{C_j\vee\neg(C_i-C_j)\}$

iv. j++

(b) $F_1=\{C_i\}\cup F_1$

(c) $j=1$

(d) i++

4. RETURN F_1

根据定理 6 和定理 7，UKCHER 算法显然是正确的. 算法结束后，F_1 是一个与 F 等价的 EPCCL 理论. KCER 算法采用桶删除原理，而 UKCHER 算法采用增量式求解方式. 因此，如果输入的子句集是不可满足的，则 UKCHER 算法能够尽早地判断出子句集的不可满足性，从而提高了知识编译的效率.

例 1.　给定一个子句集 $F=\{C_1:\neg x_1\vee x_2\vee\neg x_3, C_2:x_2\vee\neg x_3\vee x_5, C_3:x_3\vee\neg x_4, C_4:x_1\}$，图 1 给出了对 F 进行求并知识编译 UKCHER 算法的过程，显然 F_1 自始至终都为一个 EPCCL 理论.

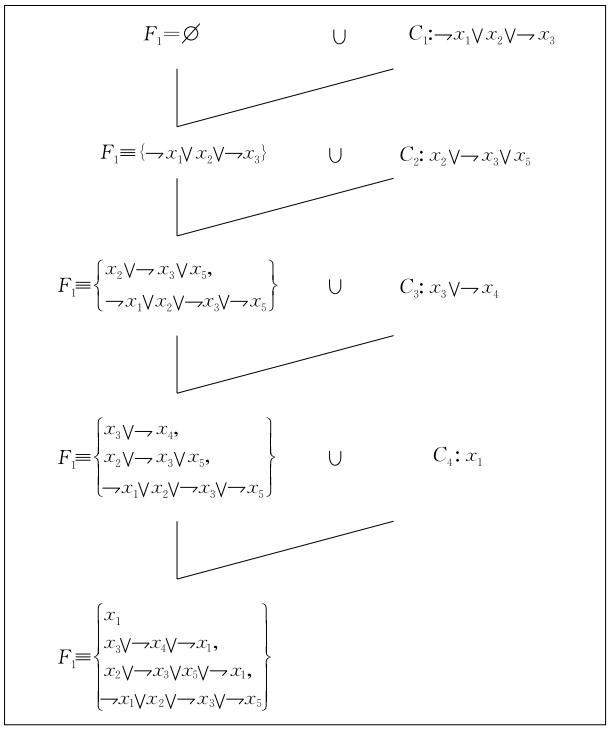

图 1　基于超扩展规则的求并扩展树

由图 1 可以看出，基于超扩展规则的求并知识编译算法 UKCHER 是可以并行实现的. 例 1 中，采用分治法的思想能够实现并行求解，因为 $J(F)=J(C_1)\cup J(C_2)\cup J(C_3)\cup J(C_4)$，则 $J(F)=(J(C_1)\cup J(C_2))\cup(J(C_3)\cup J(C_4))$，即：可以分别求出 $J(C_1)\cup J(C_2)$ 和 $J(C_3)\cup J(C_4)$，然后对二者进行合并，合并的过程同样可以利用 UKCHER 算法实现.

5　基于超扩展规则的求差知识编译

显然，子句□在变量集 $M(|M|=2^m)$ 上所能扩展出的极大项集为 M 上所有的极大项，令 $J(H)$ 表示子句集 F 所不能扩展出的极大项集，则 $J(H)=J(\square)-J(F)$ 且 $J(F)=J(\square)-J(H)$. 显然，如果利用超扩展规则能够求得 $J(H)$，且 $J(H)$ 保存为一个 EPCCL 理论，那么就能够利用 $J(H)$ 求得 $J(F)$，且所得结果仍然是一个 EPCCL 理论，就能利用这种求差的方式将任意子句集扩展为等价的 EPCCL 理论. 由定理 7 可以得出推论 4.

推论 4.　给定 EPCCL 理论 $E=\{C_1,\cdots,C_e\}$ 和子句 C，则 $J(E)-J(C)$ 可以表示为 $E_1=\bigcup\limits_{1\leq i\leq e}(J(C_i)-J(C))$. E_1 为使用推论 2、定理 5 和超扩展规则计算所得结果，则 E_1 为 EPCCL 理论.

参考定理 7 的证明过程，推论 4 显然成立.

$\{\square\}$ 是一个 EPCCL 理论，则根据推论 2、定理 5 和超扩展规则以及推论 4，能够实现任意子句集所不能扩展出极大项集的求解，并且求解结果以 EPCCL 理论的形式予以保存. 对输入子句集所不能扩展出的极大项集，再次求解其所不能扩展出的极大项集，就能够得到与原公式等价的 EPCCL 理论. 基于这种思想，本文设计了基于超扩展规则的求差知识编译算法 DKCHER，该算法执行过程分为两个阶段：(1) 求解原子句集所不能扩展出的极大项集；(2) 通过原子句集所不能扩展出的极大项集求解与原子句集等价的 EPCCL 理论. 该算法描述如算法 2.

算法 2.　DKCHER.

1. 输入：子句集 $F=\{C_1,\cdots,C_n\}$
2. 初始化：令 $F_1=F_2=\{\square\}$，$i=j=1$
3. WHILE $i\leq|F|$ & $F_1\neq\varnothing$

(a) WHILE $j\leq|F_1|$

i. IF C_i 与 C_j 互补 THEN *skip*

ii. ELSE IF $C_i\models C_j$ THEN $F_1=F_1-\{C_j\}$

iii. ELSE $C_j=\{C_j\vee\neg(C_i-C_j)\}$

iv. j++

(b) $j=1$

(c) i++

4. $i=j=1$
5. WHILE $i\leq|F_1|$

(a) WHILE $j\leq|F_2|$

i. IF C_i 与 C_j 互补 THEN *skip*

ii. ELSE IF $C_i\models C_j$ THEN $F_2=F_2-\{C_j\}$

iii. ELSE $C_j=\{C_j \vee \neg(C_i-C_j)\}$

iv. j++

(b) $j=1$

(c) i++

6. RETURN F_2

定理 8. DKCHER 算法是正确完备的，且输出结果是一个等价的 EPCCL 理论.

证明. 根据推论 4，DKCHER 算法中第 3 行循环中的(a)循环实现了 F_1 与 F 中第 i 个子句 C_i 所能扩展出极大项集的差集，计算结果为一个 EPCCL 理论，由于初始 $F_1=\{\square\}$，因此第 3 行循环之后，F_1 是 F 所不能扩展出的极大项集，且 F_1 是一个 EPCCL 理论. 同理第 5 行循环结束后 F_2 表示了 F_1 所不能扩展出的极大项集，则 F_2 是一个与 F 等价的 EPCCL 理论. 因此 DKCHER 算法是正确完备的. 证毕.

通常 SAT 问题的难解程度跟其解的个数具有直接关系，解的个数较少会使得搜索过程难以收敛，因此越难解的 SAT 问题，相应的解越少. 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，DKCHER 算法第 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.

例 2. 给定一个子句集 $F=\{C_1:\neg x_1 \vee x_2 \vee \neg x_3, C_2:x_2 \vee \neg x_3 \vee x_5, C_3:x_3 \vee \neg x_4, C_4:x_1\}$，图 2 给出了对 F 进行求差知识编译 DKCHER 算法的过

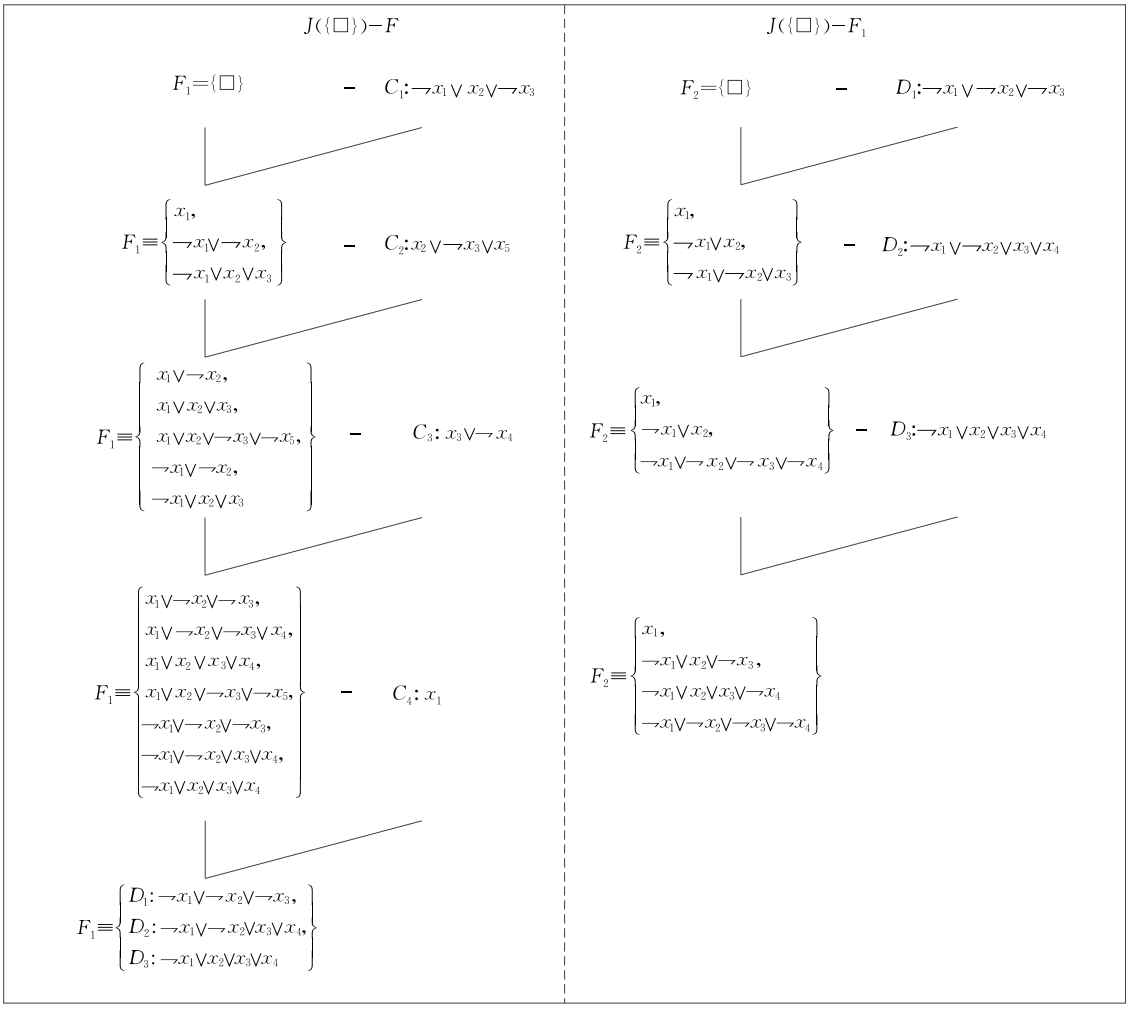

图 2 基于超扩展规则的求差扩展树

程，显然 F_1 和 F_2 自始至终都为 EPCCL 理论.

例 2 中，利用 DKCHER 算法对子句集 F 进行知识编译，首先得到 F 所不能扩展出的极大项集 F_1，再求解 F_1 所不能扩展出的极大项集，保存在 F_2 中. 例 1 和例 2 处理的是相同的实例，在这个实例下 DKCHER 算法和 UKCHER 算法所得规模相同. 然而实际情况下，它们对于相同实例知识编译所得结果的规模并非完全相同，如例 3 所示，编译结果可能不同.

例 3.　给定一个子句集 $F=\{C_1:\neg x_1 \vee x_2 \vee \neg x_3, C_2: x_2 \vee \neg x_3 \vee x_4 \vee x_5, C_3: x_3 \vee \neg x_4, C_4: x_1 \vee x_2\}$，则基于 UKCHER 算法对 F 知识编译所得结果为

$$F_1 \equiv \left\{ \begin{array}{c} \neg x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4, \\ \neg x_1 \vee x_2 \vee \neg x_3 \vee x_4 \vee \neg x_5, \\ x_2 \vee \neg x_3 \vee x_4 \vee x_5 \vee \neg x_1, \\ x_3 \vee \neg x_4 \vee \neg x_1, \\ x_3 \vee \neg x_4 \vee x_1 \vee \neg x_2, \\ x_1 \vee x_2 \end{array} \right\}$$

基于 DKCHER 算法对 F 知识编译所得结果为

$$F_2 \equiv \left\{ \begin{array}{c} \neg x_1 \vee x_2 \vee x_3 \vee \neg x_4, \\ \neg x_1 \vee x_2 \vee \neg x_3, \\ \neg x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4, \\ x_1 \vee x_2, \\ x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \end{array} \right\}$$

显然，$|F_1|>|F_2|$.

6　实验部分

本文提出了知识编译算法：UKCHER 算法和 DKCHER 算法. Lin 等人[30]于 2004 年提出的 KCER 算法和赖永等人[33]于 2013 年提出的 C2E 算法均能够将任意子句集编译为等价的 EPCCL 理论，其中，C2E 算法是目前为止效率最高的 EPCCL 理论编译器. 目前已有的知识编译算法较多，如 c2d、Dsharp[35]、sdd-linux 等，然而这些算法的编译结果有别于 EPCCL 理论，是其他形式的语言，自然在求解特性和查询支持等方面均有所不同.

本文首先在随机问题和国际上通用的标准测试样例上对比测试了 UKCHER、DKCHER、KCER 和 C2E 这 4 种知识编译算法的编译效率和编译质量，并遵照相关论文的比较方法利用编译结果中子句数量衡量编译质量.

本文实验平台如下：CPU 为 Intel Core 2 Duo CPU E8400@3.00 GHz 2.99 GHz；内存为 4 GB；操作系统为 ubuntukylin-14.04.1-desktop-amd64.

6.1　对于随机子句长度的子句集的测试

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）. 下表类似，恕不赘述.

实验结果表明：当变量数固定时，DKCHER 算法随着子句数的增加而减少；当初始子句集规模较小时，UKCHER 算法、C2E 算法和 KCER 算法随着子句数的增加而增加；当子句集规模大于某一临界值时，它们的编译规模会维持不变，甚至减少；由于样例随机生成，因此可能会存在轻微的波动.

从规模上看，表 1 中，当子句数小于 90 时，C2E 算法的编译规模会随着子句数的增加而增加，而当子句数大于等于 90 时，C2E 算法的编译规模会稳定在 2100 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，在子句数小于 50 时，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，当子句数大于等于 50 时，DKCHER 算法的编译规模远远小于 C2E 和 KCER 的编译规模，甚至当子句数为 100 时，C2E 算法编译规模是 DKCHER 算法编译规模的 128 倍，KCER 算法编译规模是 DKCHER 算法编译规模的 254 倍，由此可知求差知识编译算法 DKCHER 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，而与 KCER 算法的编译规模接近，略小于后者. 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，当子句数小于等于 50 时，UKCHER 算法的效率是最高的，KCER 算法次之，C2E 算法居于第 3 位，DKCHER 算法的效率最差，主要原因在于：DKCHER 算法分为两个阶段执行，当子句数较小时，第 1 阶段所得 EPCCL 理论规模过大，导致了第 2 阶段的编译效率较低，影响了整体执行效率. 当子句数大于 50 时，DKCHER 算法

的编译效率最高,同时随着子句数增加 UKCHER 算法和 KCER 算法所需编译时间也随之大幅度增长,二者的编译效率均低于 C2E 算法. 表 2 和表 3 中,由于变量数增加,UKCHER 算法和 KCER 算法的编译效率始终低于 C2E 算法,同时表 3 中,DKCHER 算法的编译效率始终低于 C2E 算法的编译效率.

总体来看,对于随机长度的SAT问题,DKCHER算法适用于变量数较小,子句数较大的子句集;UKCHER 算法和 KCER 算法不适用于变量数较多的子句集,而 C2E 算法在变量数较多的情况下,能够最大地发挥其优势.

表 1 随机实例〈20,*n*,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈20,30,10〉	1790	0.112	1772	0.019	928	0.079	1893	0.029
〈20,40,10〉	1519	0.092	2896	0.035	1221	0.078	2817	0.044
〈20,50,10〉	1111	0.075	3279	0.050	1780	0.084	3432	0.056
〈20,60,10〉	597	0.029	3471	0.066	1768	0.087	4009	0.074
〈20,70,10〉	280	0.020	3619	0.080	2134	0.086	4393	0.098
〈20,80,10〉	116	0.015	2891	0.091	2197	0.086	4312	0.123
〈20,90,10〉	81	0.017	2701	0.101	2019	0.083	4196	0.138
〈20,100,10〉	16	0.014	3010	0.103	2250	0.085	4064	0.164

表 2 随机实例〈25,*n*,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈25,40,10〉	12502	3.475	14887	0.137	6975	0.170	15796	0.256
〈25,50,10〉	7858	1.556	18495	0.300	7306	0.166	18895	0.472
〈25,60,10〉	3686	0.791	27011	0.412	7142	0.151	27227	0.803
〈25,70,10〉	1987	0.264	15268	0.447	9264	0.164	18482	0.668
〈25,80,10〉	1367	0.263	24434	0.601	9360	0.179	26216	1.040
〈25,90,10〉	731	0.221	18227	0.813	12302	0.282	25749	1.197
〈25,100,10〉	329	0.213	18691	0.780	9639	0.181	26868	1.596
〈25,110,10〉	202	0.161	16661	0.818	9193	0.181	24048	1.386

表 3 随机实例〈30,*n*,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈30,50,10〉	33778	21.248	102847	1.376	45363	0.697	80336	4.709
〈30,60,10〉	18752	10.507	142109	2.159	46133	0.676	150186	13.372
〈30,70,10〉	7082	3.228	123242	2.843	41903	0.572	123717	17.254
〈30,80,10〉	4853	2.304	90857	3.238	50169	0.692	136676	19.740
〈30,90,10〉	2231	1.090	91499	3.364	55943	0.694	169235	19.235
〈30,100,10〉	1052	0.614	80883	3.575	52454	0.684	170060	22.181
〈30,110,10〉	1053	0.752	55830	2.988	58068	0.721	101370	9.562
〈30,120,10〉	204	0.853	89962	3.749	48422	0.613	130408	16.250

表 4～表 6 中分别给出了〈*m*,40,10〉、〈*m*,60,10〉和〈*m*,80,10〉这 3 种变量数固定、子句数不同的测试样例的编译结果,实验结果为 50 次实验的平均值.

实验结果表明:当子句数固定,变量数改变时,4 种知识编译算法的编译规模均随着变量数的增加而增加,同时编译过程所需时间也随着变量数的增加而增加;当子句数固定时,变量数较小时,DKCHER 算法的编译效率和编译质量是最高的,进一步验证了表 1～表 3 所得的结论;变量数较大时,C2E 算法的效率能够达到最高,编译质量在小部分情况下能够达到最优(如表 4 中的〈28,40,10〉实例). 因此在对随机子句长度的子句集进行知识编译时,应该根据其子句数与变量数的比值来决定使用哪种知识编译算法.

虽然 UKCHER 算法的编译质量和编译规模往往不占优势,然而它却是 4 种知识编译算法中唯一一个可以并行实现的知识编译算法,若子句集规模过于庞大,需要并行处理,UKCHER 算法将会是首选.

表 4　随机实例〈*m*,40,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈14,40,10〉	135	0.002	367	0.011	172	0.048	393	0.014
〈16,40,10〉	286	0.007	644	0.014	352	0.058	729	0.017
〈18,40,10〉	659	0.024	1598	0.023	676	0.069	1652	0.023
〈20,40,10〉	1845	0.180	2988	0.037	1097	0.080	3293	0.041
〈22,40,10〉	2311	0.274	5346	0.062	2555	0.102	6123	0.080
〈24,40,10〉	3766	0.590	10673	0.118	3886	0.123	14556	0.229
〈26,40,10〉	8330	0.874	16556	0.195	7646	0.167	21998	0.407
〈28,40,10〉	14864	1.595	35145	0.358	11337	0.215	35105	0.915

表 5　随机实例〈*m*,60,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈14,60,10〉	63	0.002	404	0.015	207	0.052	477	0.023
〈16,60,10〉	98	0.005	673	0.017	479	0.061	993	0.027
〈18,60,10〉	247	0.015	1331	0.034	875	0.066	1855	0.037
〈20,60,10〉	576	0.038	3348	0.064	1491	0.082	4067	0.075
〈22,60,10〉	1764	0.268	5601	0.115	3311	0.108	7417	0.149
〈24,60,10〉	2877	0.387	15755	0.262	6827	0.151	17334	0.455
〈26,60,10〉	7473	0.534	27211	0.543	9185	0.185	30911	0.979
〈28,60,10〉	12580	1.261	58240	1.199	24768	0.402	62059	4.010

表 6　随机实例〈*m*,80,10〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈14,80,10〉	9	0.001	288	0.012	187	0.051	471	0.017
〈16,80,10〉	19	0.005	681	0.017	399	0.055	954	0.029
〈18,80,10〉	58	0.012	1369	0.040	893	0.067	1745	0.055
〈20,80,10〉	133	0.025	3226	0.090	1747	0.083	3819	0.120
〈22,80,10〉	745	0.086	7050	0.216	2713	0.103	9618	0.308
〈24,80,10〉	1145	0.217	15868	0.404	5303	0.136	21427	0.721
〈26,80,10〉	3540	0.361	23823	0.719	12854	0.215	41088	2.250
〈28,80,10〉	7912	0.628	72587	1.894	20712	0.374	80878	7.590

6.2　对于标准 3-SAT 子句集的测试

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 7～表 9 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. 实验结果表明：当变量数固定时，随着子句数的增加，DKCHER 算法编译所得规模逐渐减小；对于 3-SAT 问题，DKCHER 算法的编译质量明显高于其他 3 种算法，其编译效率在子句数较小时，低于 C2E 算法，随着子句数的增加，DKCHER 算法的编译效率将优于 C2E 算法的编译效率.

表 7 中，子句集规模较小时，UKCHER 算法和 KCER 算法的编译质量相同，但是前者的编译效率明显高于后者，随着子句数的增加，UKCHER 算法的编译质量渐渐优于 KCER 算法的编译质量. 上述现象产生的原因在于：虽然 UKCHER 算法的执行过程与 KCER 算法不同，然而二者基本原理相同，不同之处在于 UKCHER 算法对于不可满足的子句集的编译过程能够较早判断出不可满足，因此对于任意可满足子句集，二者编译结果相同，随着子句数的增加，会有更多的不可满足性子句集，因此 UKCHER 算法的编译质量整体会渐渐优于 KCER 算法的编译质量. 表 8 进一步验证了表 7 所得到的结论. 表 9 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 9 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.

表 7～表 9 给出了变量数固定，子句集变化的 3-SAT 实例，从中可以部分看出 DKCHER 算法对相变区间内难解问题的知识编译具有较高的质量，为了进一步验证 DKCHER 算法在处理相变区间内难解问题的优秀特性，本文按照子句数/变量数≈4.3 的比例生成了若干实例.

表 7 随机 3-SAT 实例〈20,n〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈20,46〉	1601	0.048	6199	0.105	2972	0.093	6199	0.116
〈20,56〉	857	0.032	6726	0.153	3101	0.091	6726	0.160
〈20,66〉	299	0.029	6945	0.205	3250	0.088	6945	0.203
〈20,76〉	129	0.028	6108	0.251	3241	0.092	6118	0.245
〈20,86〉	63	0.026	6008	0.264	3010	0.101	5859	0.282
〈20,96〉	22	0.026	6556	0.337	3457	0.095	6812	0.351
〈20,106〉	5	0.025	6512	0.337	3303	0.097	6943	0.376
〈20,116〉	3	0.024	6961	0.343	3075	0.091	6709	0.416

表 8 随机 3-SAT 实例〈25,n〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈25,57〉	8031	1.009	41680	0.859	17861	0.230	41680	1.612
〈25,67〉	4237	0.473	43660	1.182	21320	0.268	43660	1.940
〈25,77〉	1849	0.274	42422	1.538	18908	0.244	42422	2.160
〈25,87〉	645	0.224	43344	1.933	17566	0.253	43344	2.687
〈25,97〉	347	0.190	44898	2.394	18826	0.265	45470	3.101
〈25,107〉	145	0.206	43106	2.715	19763	0.254	43360	3.285
〈25,117〉	34	0.216	42009	2.571	21979	0.275	40233	3.745
〈25,127〉	16	0.218	44833	2.804	18243	0.243	43504	3.933

表 9 随机 3-SAT 实例〈30,n〉的实验结果

实例	DKCHER		UKCHER		C2E		KCER	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s	*size*	*time*/s
〈30,69〉	20548	5.859	230468	6.160	112513	1.272	235802	28.009
〈30,79〉	13999	3.401	219648	6.114	107060	1.145	—	—
〈30,89〉	7721	2.074	—	—	103951	1.189	—	—
〈30,99〉	4116	1.947	—	—	112619	1.202	—	—
〈30,109〉	1671	1.629	—	—	109678	1.129	—	—
〈30,119〉	345	1.501	—	—	122524	1.303	—	—
〈30,129〉	107	1.669	—	—	115610	1.247	—	—
〈30,139〉	58	1.567	—	—	106545	1.150	—	—

由于相变现象对于较大规模的子句集比较明显,而 UKCHER 算法和 KCER 算法在处理大规模子句集时容易内存溢出,同时 C2E 是目前为止效率和质量最高的 EPCCL 理论编译器,因此这一部分实验,我们仅比较了 DKCHER 算法和 C2E 算法.为了说明知识编译质量对在线推理时间的影响,本文还比较了基于二者的编译结果实现在线推理(可满足性判定)的效率,实验结果为 50 次实验的平均值,如表 10 所示.

实验结果表明:C2E 算法的编译结果规模和编译所需时间均随着变量数的增加而增加.DKCHER 算法的编译质量明显高于 C2E 算法.例如当变量数达到 35 时,C2E 的编译结果规模是 DKCHER 算法编译结果规模的 3530 倍,直接导致了基于 DKCHER 算法的在线推理时间远远小于基于 C2E 算法的在线推理时间.变量数不大于 32 时,DKCHER 算法编译效率与 C2E 算法的编译效率接近,但是明显次于后者.然而当变量数大于 32 时,DKCHER 算编

表 10 $n/m\approx4.3$ 的随机 3-SAT 实例测试

实例	DKCHER			C2E		
	size	离线编译时间/s	在线推理时间/s	*size*	离线编译时间/s	在线推理时间/s
〈26,111〉	157	0.273	<0.001	28654	0.359	0.002
〈27,116〉	141	0.408	<0.001	43425	0.483	0.004
〈28,120〉	111	0.716	<0.001	54893	0.580	0.005
〈29,124〉	202	0.994	<0.001	89340	0.937	0.007
〈30,128〉	97	1.306	<0.001	119653	1.256	0.010
〈31,133〉	220	2.076	<0.001	147685	1.770	0.015
〈32,137〉	205	2.905	<0.001	221660	2.576	0.023
〈33,141〉	132	3.929	<0.001	342724	5.179	0.033
〈34,146〉	162	5.178	<0.001	447750	8.425	0.043
〈35,150〉	170	7.213	<0.001	600145	22.076	0.067
uf20-01	78	0.022	<0.001	4811	0.085	<0.001
uf20-02	100	0.016	<0.001	2971	0.081	<0.001
uf20-03	20	0.013	<0.001	3480	0.083	<0.001
blockworld-anomaly	48	0.154	<0.001	7804	0.371	0.001
pigeon-hole-6	1	5.530	<0.001	81440	89.038	0.007
par8-1-c	64	0.034	<0.001	64	0.548	<0.001

译所需时间的增长率明显小于 C2E 算法编译所需时间的增长率，因此随着变量数的增加，DKCHER 算法的编译效率远远优于 C2E 算法的编译效率.

就在线推理的效率而言，由于 DKCHER 算法的编译质量非常高，因此基于 DKCHER 算法的在线推理所需时间始终小于 0.001 s，而由于 C2E 算法的编译质量不高，因此基于 C2E 算法的在线推理所需时间取决于 C2E 编译结果的规模，其时间随着变量数的增加而增加. 由此可以合理的推断出，对于相变点附近的难解 SAT 问题，使用 DKCHER 算法进行知识编译，在线推理的效率是最高的. C2E 算法编译所需时间之所以会随着变量数的增加而大幅度增加，原因在于：它自身第 2 阶段的 REDUCE 操作用于归约第 1 阶段所得到的 EPCCL 理论，而 REDUCE 操作的执行效率非常低，直接导致了当编译结果的规模逐渐增加时，C2E 算法整体的编译效率近似指数级下降.

表 10 中，同样对国际上通用 benchmarks 进行了测试，由于这些 benchmarks 大多数在相变点附近，因此可以看到不管是从编译效率还是编译质量，DKCHER 算法均明显优于 C2E 算法，因此对于难解类 SAT 问题的知识编译应该选择 DKCHER 算法.

6.3　与其他目标语言编译器的比较

不同的目标编译语言所能支持的查询种类是不同的. 例如，EPCCL 理论能够支持全部 8 种查询操作[33]，而 d-DNNF 仅能支持 6 种[18]. 所以比较具有不同查询功能的不同类型编译器，编译质量和编译效率仅能够反映目标编译语言的适用范围. c2d（最新版本更新于 2005 年①）是国际上比较通用的 d-DNNF 编译器，文献[36]通过实验验证了 c2d 的编译效率和编译质量均是最优的. Dsharp 是改进的 d-DNNF 编译器，然而，该算法并不能正确的执行[22,37]，因此本文将 c2d 作为对比算法之一. SDD 是一种新的知识编译语言[38]，其对应的编译器 sdd-linux（最新版本更新于 2014 年②）是现今有代表性的高效编译器，因此本文同样 sdd-linux 将作为对比算法之一.

本文随机生成了一些 30 个变量的〈n,k〉实例，n 表示子句数，k 表示每个子句的长度，通过控制 k 来生成不同互补因子率的实例（k 越大，互补因子率越高）. 由于 DKCHER 算法更适用于相变区间点附近的子句集编译，具有较好的效果，因此本小节仅列举了 UKCHER、c2d 和 sdd-linux 算法的比较结果，时间上限设定为 100 s. 实验结果如表 11 所示，表中数据为 50 次实验的平均值.

表 11　不同互补因子率子句集的测试

实例	UKCHER		c2d		sdd-linux	
	size	*time*/s	*size*	*time*/s	*size*	*time*/s
〈50,5〉	782779	5.871	173932	0.391	208271	43.047
〈50,7〉	192638	1.097	72045	0.230	101227	12.268
〈50,10〉	7615	0.038	31939	0.109	21945	1.840
〈70,7〉	641484	4.616	305655	0.442	—	—
〈70,10〉	22631	0.135	91728	0.162	70368	17.813
〈70,13〉	1165	0.011	32343	0.072	16286	1.762

表 11 中，随着互补因子率的提高，3 种算法的编译效率和编译质量均有所提高. 然而，对于子句数 50 的实例，当 k 为 5 或 7 时，c2d 算法的编译质量和编译效率均优于 UKCHER，sdd-linux 的编译质量优于 UKCHER；当 k 为 10 时，UKCHER 算法的编译效率和编译质量均优于 c2d 算法和 sdd-linux 算法. sdd-linux 的编译效率始终是最差的. 对于子句数为 70 的实例，具有同样的现象：当子句长度为 7 时，sdd-linux 超时.

产生这一现象的原因在于：EPCCL 理论建立在互补结构和扩展规则的基础上，因此当输入子句集互补因子率较高时，UKCHER 算法具有较高的编译质量和编译效率，如〈50,10〉和〈70,13〉实例；而 c2d 和 sdd-linux 建立在归结方法的基础上，因此当互补因子率较高时，其编译效率和编译质量会弱于 UKCHER 算法.

由于 UKCHER 算法等同于一种扩展反驳方法，因此本实验更进一步验证了：基于扩展规则的推理方法或知识编译方法对于互补因子率较高的问题具有较好的适用性.

7　结　语

本文提出了扩展反驳，是一种新的定理证明方法，并将其与知识编译建立了联系. 超扩展规则是一种新的推理规则，为了使 EPCCL 理论具有更广的应用范围，本文提出了两种知识编译算法：求并知识编译算法 UKCHER 和求差知识编译算法 DKCHER，是两种新的知识编译算法，通过分析算法 UKCHER 的执行过程，证明了该算法是可以并行化的，是目前为止唯一一个能够并行知识编译的 EPCCL 理论编

① http://reasoning.cs.ucla.edu/c2d/
② http://reasoning.cs.ucla.edu/sdd/

译算法.实验结果表明:算法 DKCHER 对于相变点附近的难解问题具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,而 UKCHER 算法对于互补因子率较高的子句集的编译质量和编译效率是较优的.UKCHER 算法和 DKCHER 算法可用于并行知识编译和难解类 SAT 问题知识编译,为高效推理提供了可能.

未来,我们将进一步研究 SAT 问题的潜在结构,并利用超扩展规则的性质对特殊问题结构的 SAT 问题进行快速知识编译,并通过加入启发式策略进一步提高 UKCHER 算法和 DKCHER 算法的编译效率.

参 考 文 献

[1] Cook S A. The complexity of theorem-proving procedures//Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC 1971). Denver, USA, 1971: 151-158

[2] Mitchell D, Selman B, Levesque H. Hard and easy distributions of SAT problems//Proceedings of the 10th Conference on Artificial Intelligence (AAAI 1992). San Jose, USA, 1992: 459-465

[3] Bai Shuo, Bu Dong-Bo. Analysis for phase transition of the 2-3-SAT problem. Journal of Software, 1998, 9(11): 828-832(in Chinese)
(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)

[4] Cai S W, Su K L. Local search for boolean satisfiability with configuration checking and subscore. Artificial Intelligence, 2013, 204: 75-98

[5] Luo C, Su K L, Cai S W. More efficient two-mode stochastic local search for random 3-satisfiability. Applied Intelligence, 2014, 41(3): 655-680

[6] Luo C, Cai S W, Wu W, Su K L. Double configuration checking in stochastic local search for satisfiability//Proceedings of the 28th National Conference on Artificial Intelligence (AAAI 2014). Québec City, Canada, 2014: 2703-2709

[7] Cai S W, Su K L. Comprehensive score: Towards efficient local search for SAT with long clauses//Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). Beijing, China, 2013

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

[9] Majercik S M, Littman M L. Contingent planning under uncertainty via stochastic satisfiability. Artificial Intelligence, 2003, 147(1-2): 119-162

[10] Palacios H, Geffner H. Compiling uncertainty away in conformant planning problems with bounded width. Journal of Artificial Intelligence Research, 2009, 35: 623-675

[11] Birnbaum E, Lozinskii E L. The good old Davis-Putnam procedure helps counting models. Journal of Artificial Intelligence Research, 1999, 10: 457-477

[12] del Val A. On some tractable classes in deduction and abduction. Artificial Intelligence, 2000, 116(1-2): 297-313

[13] Marquis P. Knowledge compilation using theory prime implicates//Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995). Montréal Québec, Canada, 1995, 1: 837-845

[14] Marquis P. Existential closures for knowledge compilation//Proceedings of the 22nd International Joint Conference on Artificial Intelligence(IJCAI 2011). Barcelona, Spain, 2011: 996-1001

[15] Fargier H, Marquis P. Disjunctive closures for knowledge compilation. Artificial Intelligence, 2014, 216: 129-162

[16] Darwiche A. Compiling knowledge into decomposable negation normal form//Proceedings of the 16th International Joint Conference on Artificial Intelligence(IJCAI 1999). Stockholm, Sweden, 1999: 284-289

[17] Darwiche A. Decomposable negation normal form. Journal of the ACM, 2001, 48(4): 608-647

[18] Darwiche A, Marquis P. A knowledge compilation map. Journal of Artificial Intelligence Research, 2002, 17: 229-264

[19] Fargier H, Marquis P. Extending the knowledge compilation map: Krom, Horn, Affine and beyond//Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI 2008). Chicago, USA, 2008: 442-447

[20] Fargier H, Marquis P, Niveau A. Towards a knowledge compilation map for heterogeneous representation languages //Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). Beijing, China, 2013: 877-883

[21] Fargier H, Marquis P, Niveau A, Schmidt N. A knowledge compilation map for ordered real-valued decision diagrams//Proceedings of the 28th National Conference on Artificial Intelligence (AAAI 2014). Québec City, Canada, 2014: 1049-1055

[22] Koriche F, Lagniez J M, Marquis P, Thomas S. Knowledge compilation for model counting: Affine decision trees//Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013). Beijing, China, 2013: 947-953

[23] Lai Y, Liu D Y, Wang S S. Reduced ordered binary decision diagram with implied literals: A new knowledge compilation approach. Knowledge and Information Systems, 2013, 35(3): 665-712

[24] Lin H, Sun J G, Zhang Y M. Theorem proving based on the extension rule. Journal of Automated Reasoning, 2003, 31(1): 11-21

[25] Yin Ming-Hao, Lin Hai, Sun Ji-Gui. Solving #SAT using extension rules. Journal of Software, 2009, 20(7): 1714-1725(in Chinese)
(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)

[26] Lai Yong, Ouyang Dan-Tong, Cai Dun-Bo, Lü Shuai. Model counting and planning using extension rule. Journal of Computer Research and Development, 2009, 46(3): 459-469 (in Chinese)
(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)

[27] Xu You-Jun, Ouyang Dan-Tong, Ye Yu-Xin, He Jia-Liang. Solving #SAT with extension rule indirectly. Journal of Jilin University (Engineering and Technology Edition), 2011, 41(4): 1047-1053(in Chinese)
(许有军，欧阳丹彤，叶育鑫，何加亮. 间接使用扩展规则求解#SAT问题. 吉林大学学报(工学版)，2011，41(4)：1047-1053)

[28] Li Ying, Sun Ji-Gui, Wu Xia, Zhu Xing-Jun. Extension rule algorithms based on IMOM and IBOHM heuristics strategies. Journal of Software, 2009, 20(6): 1521-1527(in Chinese)
(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)

[29] Sun Ji-Gui, Li Ying, Zhu Xing-Jun, Lü Shuai. A novel theorem proving algorithm based on extension rule. Journal of Computer Research and Development, 2009, 46(1): 9-14 (in Chinese)
(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)

[30] Lin H, Sun J G. Knowledge compilation using the extension rule. Journal of Automated Reasoning, 2004, 32(2): 93-102

[31] Yin M H, Sun J G. Counting models using extension rules// Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007). Vancouver, Canada, 2007: 1916-1917

[32] Gu Wen-Xiang, Wang Jin-Yan, Yin Ming-Hao. 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)
(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] Liu Da-You, Lai Yong, Lin Hai. C2E: An EPCCL compiler with good performance. Chinese Journal of Computer, 2013, 36(6): 1254-1260(in Chinese)
(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)

[34] Liu Lei, Niu Dang-Dang, Li Zhuang, Lü Shuai. Dynamic online reasoning algorithm based on the hyper extension rule. Journal of Harbin Engineering University, 2015, 36(12): 1614-1619(in Chinese)
(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)

[35] Muise C J, McIlraith S A, Beck J C, Hsu E I. Dsharp: Fast *d*-DNNF compilation with sharpSAT//Proceedings of the 25th Canadian Conference on Artificial Intelligence. Toronto, Canada, 2012: 356-361

[36] Huang J B, Darwiche A. The language of search. Journal of Artificial Intelligence Research, 2007, 29: 191-219

[37] Voronov A. On Formal Methods for Large-Scale Product Configuration [Ph. D. dissertation]. Chalmers University, Sweden, 2013

[38] Darwiche A. SDD: A new canonical representation of propositional knowledge bases//Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). Barcelona, Spain, 2011: 819-826

LIU Lei, born in 1960, professor, Ph. D. supervisor. His main research interests include software theory and technology.

NIU Dang-Dang, born in 1990, Ph. D. candidate. His main research interests include intelligent planning and automated reasoning.

LÜ Shuai, born in 1981, Ph. D., associate professor. His main research interests include intelligent planning and automated reasoning.

Background

The research area of this paper is Knowledge compilation. Knowledge compilation can convert a traditional knowledge base to a tractable knowledge base. The EPCCL theory is proposed by Lin and Sun in 2004, which is a proper target language for knowledge compilation. And the authors also proposed KCER to compile any CNF formula to an EPCCL theory. After that, Lai Yong et al. have introduced C2E for EPCCL compilation in 2013 on the basis of #DPLL, the efficiency and quality of C2E is best for EPCCL compilation before. In this paper, we propose two algorithms of EPCCL

compilation, UKCHER and DKCHER. They are all on the basis of hyper extension rule. So far, UKCHER is the only one algorithm of EPCCL compilation which can be parallelized. DKCHER has a best efficiency and quality for the CNF formula with big number of clauses and small number of variables. So DKCHER can be used to compile the hard problems near phase transition point, the efficiency and quality of compiling are all better than C2E.

This work is supported by the National Natural Science Foundation of China under Grant No. 61300049, the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20120061120059 and the Natural Science Research Foundation of Jilin Province of China under Grant Nos. 20130206052GX, 20140520069JH, 20150101054JC, 20150520058JH, 20150520060JH.

The performance of extension rule reasoning is an emergent problem to solve. The method of reasoning based on knowledge compilation is an efficient procedure. EPCCL theory is a proper target language of knowledge compilation. Extension rule has been used to compile any CNF formula to an EPCCL theory, such as KCER and C2E. But the low efficiencies of KCER and C2E for some kinds of CNF formula limit the application of EPCCL theory. Hyper extension rule is an expansion of extension rule, which can be used to detect the undergrounding structure of CNF formula. So we design two algorithms of knowledge compilation based on hyper extension rule, UKCHER and DKCHER. Experimental results show that, UKCHER and DKCHER can efficiently solve the hard problems which cannot be solved efficiently by KCER and C2E.