EPCCL理论的并行知识编译算法

牛当当[1],刘 磊[1],吕 帅[1,2]
(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)

摘 要: 基于超扩展规则,证明了EPCCL(Each Pair Contains Complementary Literal)理论的合并过程是可并行执行的,并设计了针对多个EPCCL理论的并行合并算法PUAE(Parallel computing Union of Any number of EPCCL).通过对EPCCL理论原始子句集的利用,提出了另一种高效的EPCCL理论并行合并算法imp-PUAE(improvement of PUAE).UKCHER(computing Union sets of maximum terms for Knowledge Compilation based on Hyper Extension Rule)是一种可并行的EPCCL理论编译算法,分别利用PUAE和imp-PUAE设计了两个并行知识编译算法P-UKCHER(UKCHER with PUAE)和impP-UKCHER(UKCHER with imp-PUAE).实验结果表明:P-UKCHER算法虽然没有提升UKCHER算法的效率,但能够提升UKCHER算法编译结果的质量,最好情况下可提升4倍;而impP-UKCHER算法能够提高UKCHER算法的效率,同时也能够提升编译结果的质量,同样最好情况下可提升4倍.

关键词: 知识编译;扩展规则;超扩展规则;EPCCL理论;并行编译
中图分类号: TP301, TP181 **文献标识码:** A **文章编号:** 0372-2112 (2018)03-0537-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.03.004

Parallel Knowledge Compilation Algorithms for EPCCL Theory

NIU Dang-dang[1], LIU Lei[1], LÜ Shuai[1,2]
(1. *College of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*;
2. *Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun, Jilin* 130012, *China*)

Abstract: Based on HER (hyper extension rule), we prove that the parallelization of merging multiple EPCCL (each pair contains complementary literal) is feasible, and the corresponding algorithm PUAE (parallel computing union of any number of EPCCL) is proposed. Through using the origin CNF formulae of EPCCL theories, another efficient merging algorithm imp-PUAE (improvement of PUAE) is proposed. UKCHER (computing union sets of maximum terms for knowledge compilation based on hyper extension rule) is a knowledge compilation algorithm for EPCCL, which can be parallelized. Based on the above methods, we proposed two parallel knowledge compilation algorithms i. e., P-UKCHER (UKCHER with PUAE) and impP-UKCHER (UKCHER with imp-PUAE), which use the PUAE algorithm and imp-PUAE algorithm, respectively. Experimentally, although P-UKCHER does not improve the efficiency of UKCHER, the compilation quality is improved. In the best case, the compilation quality can be improved by 4 times with P-UKCHER. The impP-UKCHER can improve the efficiency and compilation quality of UKCHER at the same time, and the compilation quality can also be improved by 4 times in the best case.

Key words: knowledge compilation; extension rule; hyper extension rule; EPCCL theory; parallel compilation

1 引言

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.Darwiche和Marquis介绍了知识编译图谱用于比较不同的目标语言[1].随后,相关学者对知识编译图谱进行了扩充[2,3].Fargier和Marquis针对知识编译研究了几种不同目标语言存在的闭包[4].Kazemi等人针对提升概率推理设计并实现了相关知识编译算法[5].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[6].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

收稿日期:2016-07-15;修回日期:2016-12-15;责任编辑:梅志强
基金项目:国家自然科学基金(No.61300049, No.61502197, No.61503044);教育部高等学校博士学科点专项科研基金(No.20120061120059);吉林省青年科研基金项目(No.20140520069JH, No.20150520058JH);吉林省自然科学基金项目(No.20150101054JC)

较,以及如何提高目标语言的查询能力.算法并行化能够提高其求解效率[7~10],因此如何实现知识编译方法的并行化是一个新的领域.

2003 年,Lin 等人提出了扩展规则推理方法(Extension Rule,ER)[11],被著名人工智能专家 Davis 称为与归结方法互补的方法.基于扩展规则,Lin 等人提出了一种新的命题逻辑理论证明方法 IER[11],还提出了一种的知识编译方法 KCER,可以将子句集编译为 EPCCL 理论[12].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[13].刘大有等人证明了 EPCCL 理论能够满足文献[1]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[14].刘磊等人提出了超扩展规则,并基于超扩展规则提出了三种新的 EPCCL 理论编译算法[15~17].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,因此提高 EPCCL 编译器的编译效率和编译质量具有重要意义.

为了探索 EPCCL 理论并行知识编译的可行性,以及知识编译算法并行后对编译效率和编译质量的影响,本文基于超扩展规则,设计了并行合并多个 EPCCL 理论的 PUAE 算法和 imp-PUAE 算法.UKCHER 算法是一个可并行的 EPCCL 理论编译算法[16],结合 PUAE 算法和 imp-PUAE 算法,本文设计了两种并行 EPCCL 理论编译算法:P-UKCHER(UKCHER with PUAE)和 impP-UKCHER(UKCHER with imp-PUAE).实验结果表明,P-UKCHER 算法虽然没有提升 UKCHER 算法的效率,但是较大地提升了 UKCHER 算法的编译质量;而 impP-UKCHER 算法的执行效率会随着任务划分数的增加而增加,同时也能够提升 UKCHER 算法的编译质量.本文的研究工作能够为其它 EPCCL 理论编译算法的并行化及其它目标语言编译算法的并行化提供借鉴.

2 超扩展规则

为便于表示,用 $V(C)$ 和 $V(G)$ 分别表示子句 C 和子句集 F 中出现的变量集,用 $J_M(C)$ 和 $J_M(F)$ 分别表示子句 C 和子句 F 在变量集 M 上所能扩展出的极大项集.本文所研究的子句集中不含重言式,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 1(扩展规则,Extension rule)[11] 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 $C\equiv D$.

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.

定义 2(超扩展规则,Hyper extension rule)[15] 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.

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),文献[15]中利用互补展开将 $C\vee\neg A$ 转化为两两子句互补的子句集.

定义 3(EPCCL 理论)[12] 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.

3 EPCCL 理论的合并

若存在多个 EPCCL 理论,如何求得它们所能扩展出极大项集的并集对于实现 EPCCL 理论的并行编译是一个关键问题.

引理 1[16] 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 $J_M(C_i)$ 和 $J_M(C_j)$,若 C_i 与 C_j 互补,则 $|J_M(C_i)\cup J_M(C_j)|=|J_M(C_i)|+|J_M(C_j)|$ 且 $J_M(C_i)-J_M(C_j)=J_M(C_i)$.

引理 2[16] 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 $J_M(C_i)$ 和 $J_M(C_j)$,若 $C_i|=C_j$,则 $J_M(C_i)\cup J_M(C_j)=J_M(C_i)$ 且 $J_M(C_j)-J_M(C_i)=\varnothing$.

定理 1 若存在一对互补子句 C_1 和 C_2,和一个 EPCCL 理论 $E=\{E_1,\cdots,E_e\}$,变量集 M 满足 $(V(C_1)\cup V(C_2)\cup V(E))\subseteq M$,子句集 $\Sigma=\{C_1,C_2\}\cup\bigcup_{1\leqslant i\leqslant e}(J_M(E_i)-J_M(C_1)-J_M(C_2))$ 为使用引理 1、引理 2 和超扩展规则的计算结果,则 Σ 是一个 EPCCL 理论,且 $J_M(\Sigma)=J_M(C_1)\cup J_M(C_2)\cup J_M(E)$.

证明 集合 $\Sigma_1=\bigcup_{1\leqslant i\leqslant e}(J_M(E_i)-J_M(C_1)-J_M(C_2))$ 中任意一个子句与 C_1 或 C_2 所能扩展出极大项集的交集均为空,因此 C_i 和 C_j 与 Σ_1 中所有子句互补.根据文献[16]中定理 7 的证明过程,$\bigcup_{1\leqslant i\leqslant e}(J_M(E_i)-J_M(C_1))$ 是一个 EPCCL 理论,因此 $\bigcup_{1\leqslant i\leqslant e}(J_M(E_i)-J_M(C_1)-J_M(C_2))$ 显然是一个 EPCCL 理论.又由于 C_1 和 C_2 互补,因此 Σ 是一个 EPCCL 理论.$\bigcup_{1\leqslant i\leqslant e}(J_M(E_i)-J_M(C_1)-J_M(C_2))=J_M(E)-J_M(C_1)-J_M(C_2)$,因此 $J_M(\Sigma)=J_M(C_1)\cup J_M(C_2)\cup J_M(E)$ 显然成立.综上,定理 1 成立.

证毕.

推论 1　给定 EPCCL 理论 $E_1=\{C_1,\cdots,C_g\}$ 和 $E_2=\{D_1,\cdots,D_h\}$,变量集 M 满足 $(V(E_1)\cup V(E_2))\subseteq M$,则 $J_M(E_1)\cup J_M(E_2)$ 可以表示为:$E_3=E_1\cup\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))$ 或 $E_4=E_2\cup\bigcup_{1\leqslant i\leqslant g}(J_M(C_i)-J_M(D_1)-\cdots-J_M(D_h))$. E_3 和 E_4 为使用引理 1、引理 2 和超扩展规则计算所得结果,则 E_3 和 E_4 均为 EPCCL 理论,且 $J_M(E_3)=J_M(E_4)=J_M(E_1)\cup J_M(E_2)$.

证明　参照定理 1 的证明过程,显然 $\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))$ 和 $\bigcup_{1\leqslant i\leqslant g}(J_M(C_i)-J_M(D_1)-\cdots-J_M(D_g))$ 是 EPCCL 理论,因此 E_3 和 E_4 均为 EPCCL 理论. $\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))=J_M(E_2)-J_M(E_1)$,因此 $E_3=E_1\cup\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))\equiv J_M(E_1)\cup J_M(E_2)$ 成立. 同理,$E_4=E_2\cup\bigcup_{1\leqslant i\leqslant g}(J_M(C_i)-J_M(D_1)-\cdots-J_M(D_g))\equiv J_M(E_1)\cup J_M(E_2)$ 成立,因此 $J_M(E_3)=J_M(E_4)=J_M(E_1)\cup J_M(E_2)$ 成立. 综上,推论 1 成立.

证毕.

推论 1 给出了计算两个 EPCCL 理论所能扩展出极大项集并集的方法,显然该计算过程是可分解的. 给定两个 EPCCL 理论 $E_1=\{C_1,\cdots,C_g\}$ 和 $E_2=\{D_1,\cdots,D_h\}$,假设选择利用 $E_3=E_1\cup\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))$ 计算 $J_M(E_1)\cup J_M(E_2)$,则 $E_3=E_1\cup(J_M(D_1)-J_M(C_1)-\cdots-J_M(C_g))\cup\cdots\cup(J_M(D_h)-J_M(C_1)-\cdots-J_M(C_g))$,根据推论 1,显然任意 $(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))\cap(J_M(D_j)-J_M(C_1)-\cdots-J_M(C_g))=\varnothing(1\leqslant i,j\leqslant h,i\neq j)$. 因此可以并行计算 $(J_M(D_i)-J_M(C_1)-\cdots-J_M(C_g))(1\leqslant i\leqslant h)$,计算结果可以直接输出作为 $J_M(E_1)\cup J_M(E_2)$ 计算结果的一部分. 使用推论 1 合并两个 EPCCL 理论之后,所得结果仍然为 EPCCL 理论,因此,可以采用增量式的思想并行求得任意多个 EPCCL 理论的所能扩展出极大项集的并集.

根据上述过程,本文设计了针对多个 EPCCL 理论的并行求并合并算法 PUAE,算法描述如算法 1.

算法 1 是将 s 个 EPCCL 理论进行合并,并将结果保存为 EPCCL 理论的并行执行过程. 该算法利用 k 表示任务的划分粒度,每次并行会将任务分成 k 份进行相应操作. 算法 1 在第 8 行调用了 CDF 子算法,CDF 的功能是求解一个子句与一个子句集所能扩展出极大项集的差集,并将结果保存为 EPCCL 理论. CDF 子算法是 DKCHER 算法的部分过程,其理论正确性请参考文献[16].

定理 2　**算法 1 是正确且完备的**

证明　由于算法 1 在执行过程中,E 始终是一个 EPCCL 理论,且根据推论 1,算法 1 实现了 $E_1,\cdots,E_s$ 所能扩展出极大项集的并集,因此 PUAE 是正确的. 算法 1 对输入的 EPCCL 理论没有做任何限制,因此它能够处理所有的 EPCCL 理论,进而算法 1 是完备的. 综上,定理 2 成立.

证毕.

通常情况下,算法 1 采用了并行合并的策略,因此能够提高 EPCCL 理论的合并效率. 然而,由于 CDF 的计算过程中,所产生的中间结果会随着迭代次数的增加而产生爆炸式地增长,因此算法 1 的合并过程属于指数级的空间复杂度和时间复杂度,这是知识编译过程的通病,是无法避免的. 为了使得合并过程更加高效地完成,本文从理论层面分析并得出更快的合方式,如推论 2 所示.

算法 1　PUAE

```
Input:s 个 EPCCL 理论{E_1,…,E_s}(s≥1)
Output:E_1∪…∪E_s 的 EPCCL 理论
1   E=E_1,h=2
2   While h≤s
3     Divide E_h to {E_h^1,…,E_h^k}
4     Parallel
5       For every E_h^v(1≤v≤k)
6         d=1
7         While d≤|E_h^v|
8           T_v=T_v∪CDF(E,C_d)(C_d∈E_h^v)
9           d++
10      End Parallel
11      E=E_h∪T_1∪…∪T_k
12      h++
13  Return E
子算法 CDF(CNF F={C_1,…,C_n},子句 C)
1   Initialization:令 F_1={C},i=j=1
2   While i≤|F|&F_1≠∅
3     While j≤|F_1|
4       If C_i 与 C_j 互补 Then skip
5       Else If C_i|=C_j Then F_1=F_1-{C_j}
6       Else C_j={C_j∨¬(C_i-C_j)}
7       j++
8     i++
9     j=1
10  Return F_1
```

推论 2　给定 EPCCL 理论 E_1 和 E_2,变量集 M 满足 $V(E_1)\cup V(E_2)\subseteq M$,且 $F_1=\{X_1,\cdots,X_g\}$ 和 $F_2=\{Y_1,\cdots,Y_h\}$ 是与 E_1 和 E_2 分别等价的子句集,则 $J_M(E_1)\cup J_M(E_2)$ 可以表示为:$E_3=E_1\cup\bigcup_{1\leqslant i\leqslant h}(J_M(D_i)-J_M(X_1)-$

$\cdots - J_M(X_g))$或 $E_4 = E_2 \cup \bigcup_{1 \leqslant i \leqslant g} (J_M(C_i) - J_M(Y_1) - \cdots - J_M(Y_h))$. E_3 和 E_4 为使用引理 1、引理 2 和超扩展规则计算所得结果,则 E_3 和 E_4 均为 EPCCL 理论,且 $J_M(E_3) = J_M(E_4) = J_M(E_1) \cup J_M(E_2)$.

根据推论 1 的证明过程,不难看出推论 2 是正确的,因此本文省略了具体证明过程.

根据推论 2 可知,在进行多个 EPCCL 理论的合并时,不需要直接对 EPCCL 理论操作,可以用它的等价普通子句集来代替. 一般而言,EPCCL 理论的规模要大于与它等价的普通子句集的规模,因此若知道输入 EPCCL 理论的原始子句集,可以利用这些子句集加速 PUAE 的合并过程.

将这种改进的合并算法称为 imp-PUAE,算法描述如算法 2.

定理 3 算法 2 是正确且完备的

证明 由于算法 2 在执行过程中,E 始终是一个 EPCCL 理论,且根据推论 2,算法 2 实现了 $E_1, \cdots, E_s$ 所能扩展出极大项集的并集,因此算法 2 是正确的. 算法 2 对输入的 EPCCL 理论没有做任何限制,算法 2 要求同时输入与所输入 EPCCL 理论等价的普通子句集,若未输入则可直接使用 EPCCL 理论作为与其等价的普通子句集,因此它能够处理所有的 EPCCL 理论,进而算法 2 是完备的. 综上,定理 3 成立.

证毕.

算法 2 imp-PUAE

```
Input: s 个 EPCCL 理论 E_1, ..., E_s (s≥1), F_1, ..., F_s 是分别与 E_1, ..., E_s 等价的普通子句集
Output: E_1 ∪ ... ∪ E_s 的 EPCCL 理论
1  E = E_1, h = 2
2  While h≤s
3     Divide E to {E_h^1, ..., E_h^k}
4     Parallel
5        For every E_h^v (1≤v≤k)
6           d = 1
7           While d≤|E_h^v|
8              T_v = T_v ∪ CDF(F_h, C_d) (C_d ∈ E_h^v)
9              d++
10    End Parallel
11    E = E_h ∪ T_1 ∪ ... ∪ T_k
12    h++
13 Return E
```

在得知输入 EPCCL 理论对应原始子句集的前提下,算法 2 同样能够实现多个 EPCCL 理论的并行合并. 由于 CDF 不要求输入的子句集为 EPCCL 理论,因此可以用与之等价的原始子句集代替,因此算法 2 在第 8 行调用 CDF 直接对与 E_h 等价的普通子句集 F_h 操作. 通常情况下,EPCCL 理论的规模要远远大于与之等价的普通子句集的规模,甚至是普通子句集的指数倍,因此 imp-PUAE 算法中调用 CDF 的过程所需开销应该远远小于 PUAE. EPCCL 理论的编译过程通常以子句集形式的知识库作为输入,因此 imp-PUAE 可以作为合并策略应用到并行 EPCCL 理论编译算法中.

4 EPCCL 理论的并行编译算法

UKCHER 算法是近期提出的可并行执行的 EPCCL 理论编译算法[16],根据推论 1 和推论 2,EPCCL 理论的合并过程是可并行执行的,因此可以设计出基于 UKCHER 算法的并行知识编译过程.

首先,将输入子句集分解为若干子集;然后,利用 UKCHER 算法将所有子集编译为 EPCCL 理论;最后,再利用上节提出的并行过程将这些 EPCCL 理论进行合并,所得到的结果就是与输入子句集等价的 EPCCL 理论.

本文给出了基于 UKCHER 算法的并行 EPCCL 理论编译算法 P-UKCHER,描述如算法 3.

算法 3 P-UKCHER

```
Input: 子句集 F = {C_1, ..., C_n}
Output: F 的 EPCCL 理论
1  Divide F to {F_1, ..., F_k}
2  Parallel
3     For every F_h (1≤h≤k)
4        E_h = UKCHER(F_h)
5  End Parallel
6  Return PUAE(E_1, ..., E_k)
子算法 UKCHER(CNF F = {C_1, ..., C_n})[16]
1  F_1 = ∅, i = j = 1
2  While i≤|F|
3     While j≤|F_1|
5        If C_i 与 C_j 互补 Then skip
6        Else If C_i = C_j Then F_1 = F_1 - {C_j}
7        Else C_j = {C_j ∨ ¬(C_i - C_j)}
8        j++
9        F_1 = {C_i} ∪ F_1
10    i++
11    j = 1
12 Return F_1
```

算法 3 中使用了多次并行处理,最多调用 k 次 UKCHER 算法和 $k \times (k-1)$ 次算法 1 中的 CDF 算法,其中 CDF 算法的作用是计算输入子句 C 和子句集 F 所能扩展出极大项集的差集. 在该算法中主要在两个阶段使用并行策略,第 1 个阶段是对 k 个输入子句集的子集利用 UKCHER 算法进行并行知识编译,对应地将其编译为 k 个 EPCCL 理论,第 2 个阶段是对得到的 k 个 EPCCL 理论进行合并,在合并任意两个 EPCCL 理论的时候使用了并行策略 PUAE.

在基于扩展规则的编译过程中,子句的长度会随着编译的过程不断增加,子句长度增加后,其所能扩展出的极大项数会减少,扩展能力会降低. P-UKCHER 算法在对输入子句集的子集编译后,采用 PUAE 合并策略对子任务所得 EPCCL 理论进行合并. 合并过程中,保留了大量子任务输出 EPCCL 理论中的子句(算法 1 的第 11 行). 因而可预见的是,利用 P-UKCHER 算法并行编译所得结果扩展能力会增强,具体体现在所得结果规模小,平均子句长度短.

P-UKCHER 算法采用了最基本的分治法思想,合并过程利用分治所得结果直接并行合并. 然而这种思想却并不适用于知识编译过程,通过对 PUAE 算法和 imp-PUAE 的分析可知,若合并过程中能够利用"治"过程的输入子句集,则可以使合并过程更加高效. 算法 3 中,"治"阶段所得任意 EPCCL 理论 $E_1,\cdots,E_k$ 对应子句集为 $F_1,\cdots,F_k$,因此可以采用 imp-PUAE 策略合并这些 EPCCL 理论. 将这种新的并行 UKCHER 算法称为 impP-UKCHER,描述如算法 4.

算法 4　impP-UKCHER

Input:子句集 $F=\{C_1,\cdots,C_n\}$
Output:F 的 EPCCL 理论
1　Divide F to $\{F_1,\cdots,F_k\}$
2　Parallel
3　　For every $F_h(1\leqslant h\leqslant k)$
4　　E_h = UKCHER(F_h)
5　End Parallel
6　Return imp-PUAE($E_1,\cdots,E_k,F_1,\cdots,F_k$)

算法 4 与算法 3 的执行过程大致相同,差别之处在于第 6 行,impP-UKCHER 算法采用 imp-PUAE 算法对多个 EPCCL 理论进行合并,而 P-UKCHER 算法采用了 PUAE 算法. impP-UKCHER 算法同 P-UKCHER 算法一样能够减少串行编译所得结果的规模,本文将通过实验部分验证这一结论.

5　实验部分

知识编译算法的编译效率和编译质量是评价编译算法的两个重要参数,其中,编译质量是主要考虑的参数. 由于编译过程所需时间可以通过对编译结果的多次查询得到补偿,所以当两种编译算法的编译质量相差较大且编译效率相差不大时,在衡量该两种算法时通常可以不考虑编译效率的差别. 若两种编译算法的编译质量相差不多,则需考虑编译效率.

本文在随机问题和国际上通用的测试用例上对比测试了 P-UKCHER 算法、impP-UKCHER 算法和 UKCHER 算法[16]的编译效率和编译质量(参照国际惯例,主要使用编译结果中子句数量进行衡量). 并行算法用多核算法设计实现,k 表示任务的划分粒度. 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 7.

5.1　对于随机子句长度的子句集的测试

本文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三个参数 m、n 和 e 的子句集,其中:n 为变量个数,m 为子句个数,e 为每个子句的最大长度. n 个变量对应 $2*n$ 个文字,每个文字出现的概率是相同的,同时控制生成过程,避免生成重言式.

表 1 给出了〈20,m,10〉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);若执行时间超过 1000s,则认定该程序执行失败;表中数据加粗表示当前行最优编译效率,数据加框表示当前行最优编译质量. 下表类似,恕不赘述.

表 1　子句长度随机实例上的实验结果

instances	impP-UKCHER($k=4$)		impP-UKCHER($k=2$)		UKCHER		P-UKCHER($k=4$)		P-UKCHER($k=2$)	
	size	time(s)	size	time(s)	size	time(s)	size	time(s)	size	time(s)
〈20,30,10〉	510	0.006	557	0.008	2165	0.011	517	0.063	475	0.058
〈20,40,10〉	933	0.014	912	0.015	2806	0.019	975	0.219	975	0.179
〈20,50,10〉	1501	0.023	1453	0.025	3613	0.029	1404	0.411	1397	0.240
〈20,60,10〉	2434	0.029	2496	0.032	3284	0.038	2062	0.765	2059	0.338
〈20,70,10〉	2919	0.037	2863	0.041	3671	0.044	2750	1.257	2591	0.502
〈20,80,10〉	3012	0.048	3031	0.053	3415	0.053	3494	1.697	2933	0.518
〈20,90,10〉	3934	0.059	3398	0.057	2701	0.051	5052	2.543	4026	0.715
〈20,100,10〉	4652	0.067	3980	0.069	2788	0.059	3817	2.811	3613	0.868

从编译质量上看:当 $m<90$ 时,impP-UKCHER 和 P-UKCHER 的编译质量相对于 UKCHER 算法均有所提升,最好情况下可提升 4 倍,这一现象验证了上节的结论. P-UKCHER 和 impP-UKCHER 算法是并行编译算法,在合并过程中保留了大量子任务输出的短子句,缩小最终编译结果中的平均子句长度,进而提高编译质量. 然而当 $m\geqslant 90$ 时,impP-UKCHER 和 P-UKCHER 的编译质量相对于 UKCHER 算法均有所削弱,原因在于:

UKCHER 算法采用了动态在线推理的思想,随着子句数的增加,子句集不可满足的概率增大,因此 UKCHER 算法能更早判断出不可满足而提前结束,所得结果的规模相对增加缓慢. 因此从编译质量的角度出发,UKCHER 算法的并行编译算法更适合于子句数较小的实例.

从编译效率上看:表 1 中,P-UKCHER 未起到加速效果,反而极大地降低了编译效率,其原因正如之前分析,它们采用了 PUAE 算法直接合并多个 EPCCL 理论,算法 1 在第 8 行调用 CDF 子函数处理 EPCCL 理论所需时间要远远大于直接处理原始子句集所需的时间,而 UKCHER 算法串行操作中是直接对子句集进行处理的,因此 P-UKCHER 并未起到加速效果;当 $m<90$ 时,impP-UKCHER 起到了一定的加速效果,然而加速效果并不明显,这是由于本文分配任务的策略采用的是按子句数划分,而对于随机子句长度的实例,这种划分策略负载均衡会很差;当 $m\geqslant 90$ 时,impP-UKCHER 未起到加速效果,其原因在于 UKCHER 算法结合了动态在线推理方法,更适合子句数较多的实例. 因此从编译效率的角度出发,UKCHER 算法的并行编译算法同样更适合于子句数较小的实例.

5.2 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并行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 2 给出了〈20,m〉、〈25,m〉和〈30,m〉三种不同 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.

表 2 随机 3-SAT 实例上的实验结果

instances	impP-UKCHER($k=4$)		impP-UKCHER($k=2$)		UKCHER		P-UKCHER($k=4$)		P-UKCHER($k=2$)	
	size	time(s)	size	time(s)	size	time(s)	size	time(s)	size	time(s)
〈20,46〉	1847	0.031	2718	0.047	5767	0.061	2655	2.647	2607	0.675
〈20,56〉	3709	0.061	3773	0.067	6614	0.087	3782	4.523	3874	0.911
〈20,66〉	4685	0.084	4823	0.094	6753	0.121	4829	8.143	4791	1.426
〈25,67〉	20797	0.480	20933	0.564	40925	0.726	24209	629.289	23958	94.145
〈25,77〉	29485	0.636	29602	0.752	43389	1.009	33293	714.305	35548	83.883
〈25,87〉	35704	0.885	37108	1.037	48341	1.322	–	–	37086	119.436
〈30,69〉	101798	2.033	101592	3.159	230468	4.041	–	–	111037	612.358
〈30,79〉	100053	2.100	100005	3.201	219648	4.210	–	–	–	–
〈30,89〉	134387	3.768	134079	5.984	–	–	–	–	–	–

从编译质量上来看:相对于在随机子句长度实例上的并行编译,在 3-SAT 实例上进行并行编译的编译质量提升幅度稍弱,主要原因在于:由于 3-SAT 实例中的子句长度均为 3,而随机子句长度实例中长子句较多,会加快扩展规则编译过程中平均子句长度的增长,因此,串行算法在 3-SAT 实例上有较好的表现,而并行算法的优势相对较弱. 表 2 中,当子句集规模较大时,编译算法还会因为内存溢出而无法处理,因此如何提高其编译质量是一个待解决的开放课题.

从编译效率上来看:表 2 进一步验证了表 1 所得出的结论,同时可以看出并行算法对于 3-SAT 实例编译的加速比更大,原因在于:本文的 3-SAT 实例都是随机生成的,且其中的任务也是均匀划分的,因此各个子任务的完成时间大致相近,负载均衡效果较好,进而得到较好的加速效果.

通过实验部分的测试能够得出,并行知识编译过程中的负载均衡极大地影响着并行算法的加速比,如何对输入子句集进行合理的划分是提升并行编译算法的关键;另外,串行程序的特性需要尽可能保留到并行程序中,这样才能保证并行程序对任意问题都有良好的求解加速比.

6 结论与展望

本文重点研究了如何合并多个 EPCCL 理论,并证明了 EPCCL 理论的合并过程是可并行化的. 设计了两种 EPCCL 理论的并行合并算法:PUAE 和 imp-PUAE,其中 imp-PUAE 算法的执行需要事先知道输入 EPCCL 理论的原始子句集. 基于 PUAE 和 imp-PUAE,算法 P-UKCHER 和 impP-UKCHER 可以对任意子句集并行编译,所得结果为 EPCCL 理论. 实验结果表明,虽然算法 P-UKCHER 和 impP-UKCHER 都是并行执行的,同时二者均可提升 UKCHER 算法的编译质量,然而只有 impP-UKCHER 起到了加速效果. 因此在考虑知识编译算法的并行化时,应该考虑编译后子句集的规模是否远远大于原始子句集. 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它目标语言编译算法的并行化提供借鉴.

未来将研究如何进一步提高 EPCCL 理论合并算法的效率,并使 impP-UKCHER 算法有更广的应用范围. 本文对输入子句集采用了等子句数分割,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.

参考文献

[1] Darwiche A, Marquis P. A knowledge compilation map[J]. Journal of Artificial Intelligence Research, 2002, 17(1): 229 – 264.

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

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

[4] Fargier H, Marquis P. Disjunctive closures for knowledge compilation[J]. Artificial Intelligence, 2014, 216(16): 129 – 162.

[5] Kazemi S M, Poole D. Knowledge compilation for lifted probabilistic inference: Compiling to a low-level language[A]. Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning[C]. Cape Town, South Africa: AAAI Press, 2016. 561 – 564.

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

[7] 申元霞,曾传华,王喜凤,江小燕. 并行协作骨干粒子群优化算法[J]. 电子学报,2016,44(7):1643 – 1648.
SHEN Yuan-Xia, ZENG Chuan-Hua, WANG Xi-Feng, JIANG Xiao-Yan. A parallel-cooperative bare-bone particle swarm optimization algorithm[J]. Acta Electronica Sinica, 2016, 44(7): 1643 – 1648. (in Chinese)

[8] Huang Y F, Xiao J H, Jiang K Q, Chen Z H. Parallel solution for maximum independent set problem by programmable tile assembly[J]. Chinese Journal of Electronics, 2016, 25(2): 203 – 208.

[9] 王伟,余玉揆,郝燕玲. 一种新型混合并行粒子滤波频率估计方法[J]. 电子学报,2016,44(3):740 – 746.
WANG Wei, YU Yu-Kui, HAO Yan-Ling. A novel parallel particle filter for frequency estimation[J]. Acta Electronica Sinica, 2016, 44(3): 740 – 746. (in Chinese)

[10] Katsirelos G, Sabharwal A, Samulowitz H, Simon L. Resolution and parallelizability: Barriers to the efficient parallelization of SAT solvers[A]. Proceedings of the 27th National Conference on Artificial Intelligence[C]. Bellevue, Washington: AAAI Press, 2013. 481 – 488.

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

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

[13] 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[J]. 计算机研究与发展,2011,48(11):2064 – 2073.
GU Wen-Xiang, WANG Jin-Yan, YIN Ming-Hao. Knowledge compilation using extension rule based on MCN and MO heuristic strategies[J]. Journal of Computer Research and Development, 2011, 48(11): 2064 – 2073. (in Chinese)

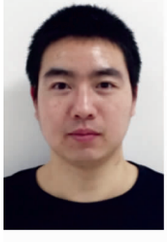
[14] 刘大有,赖永,林海. C2E:一个高性能的 EPCCL 理论编译器[J]. 计算机学报,2013,36(6):1254 – 1260.
LIU Da-You, LAI Yong, LIN Hai. C2E: An EPCCL compiler with good performance[J]. Chinese Journal of Computer, 2013, 36(6): 1254 – 1260. (in Chinese)

[15] 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[J]. 哈尔滨工程大学学报,2015,36(12):1614 – 1619.
LIU Lei, NIU Dang-Dang, LI Zhuang, LÜ Shuai. Dynamic online reasoning algorithm based on the hyper extension rule[J]. Journal of Harbin Engineering University, 2015, 36(12): 1614 – 1619. (in Chinese)

[16] 刘磊,牛当当,吕帅. 基于超扩展规则的知识编译方法[J]. 计算机学报,2016,39(8):1681 – 1696.
LIU Lei, NIU Dang-Dang, LÜ Shuai. Knowledge compilation methods based on the hyper extension rule[J]. Chinese Journal of Computers, 2016, 39(8): 1681 – 1696. (in Chinese)

[17] 牛当当,刘磊,吕帅. EPCCL 理论的求交知识编译算法[J]. 软件学报,2017,28(8):2096 – 2112.
NIU Dang-Dang, LIU Lei, LÜ Shuai. Knowledge compilation algorithm based on computing the intersection for EPCCL theories[J]. Journal of Software, 2017, 28(8): 2096 – 2112. (in Chinese)

作者简介



牛当当　男,1990 年 2 月出生,陕西周至人. 现为吉林大学计算机科学与技术学院博士研究生,主要研究方向为自动推理和抽象论辩. E-mail:ddniu15@mails.jlu.edu.cn

刘　磊　男,1960 年 8 月出生,吉林长春人. 现为吉林大学计算机科学与技术学院教授、博士生导师,主要研究方向为软件理论与技术. E-mail:liulei@jlu.edu.cn

吕　帅(通信作者)　男,1981 年 7 月出生,吉林公主岭人. 2010 年获得吉林大学博士学位,现为吉林大学计算机科学与技术学院副教授,主要研究方向为人工智能、智能规划与自动推理. E-mail:lus@jlu.edu.cn