An Improved Algorithm for Knowledge Compilation Using the Combination Rule

Hai Lin[1,a]

[1]College of Software, Shenyang Normal University

253 Huanghe North Street, Shenyang, 110034, China

[a]jlu_linhai@163.com

Keywords: knowledge compilation, Combination rule

Abstract. Knowledge compilation is a way of preprocessing a given knowledge base so that queries can be answered more efficiently. In literature, a lot of methods for knowledge compilation have been proposed. In this paper, we propose a new method for knowledge compilation, which is based on the Combination rule. Intuitively, the Combination rule has some nice properties so that it can make the compiled knowledge base more compact and more efficient to query. So our proposed method has some advantages over existing methods.

Introduction

Knowledge compilation is an important problem in logical reasoning in computer science. The idea is to split reasoning into two phases: an offline phase where the knowledge base is preprocessed, and an online phase where the query is answered using the compiled knowledge base from the first phase. The compilation process needs to be done only once so as to answer different online queries. In literature, a lot of methods for knowledge compilation have been proposed [3,4,5,6].

In [1], we proposed a method for knowledge compilation. The idea is to use EPCCL (Each Pair Contains Complementary Literals) theory as the target language. Throughout the compilation process, we use the Extension rule, which was introduced in [2]. We showed that the query can be answered in time linear in the size of the compiled knowledge base. Since the propositional reasoning is intractable in general, the size of the compiled knowledge base is exponential in the size of the original knowledge base in the worst case. So a key question is how to keep the size of the compiled knowledge base as small as possible in practice.

In this paper, we propose a new rule, called the Combination rule. The idea is that we use the Combination rule to combine clauses in the compiled knowledge base so as to keep the size of the compiled knowledge base as small as possible.

This paper is organized as follows. In section 2, we review the Extension rule from [2]. In section 3, we give a brief overview of the knowledge compilation method as proposed in [1]. In the next section, we propose our method based on the combination rule. Finally we give some concluding remarks.

The Extension Rule

In this section, we review the extension rule, which was proposed in [2]. To begin with, we assume that the original knowledge base is given in Conjunctive Normal Form(CNF).

Definition 1 Given a clause C and a set M:

$C'=\{C \vee a, C \vee \neg a \mid$ "a" is an atom, $a \in M$ and "a" does not appear in C$\}$

We call the operation from C to C' the *extension rule* on C. We call C' the result of extension rule.

For example, given a clause $p \vee q$ and a set {p, q, r}, the result of extension rule of on $p \vee q$ is $\{p \vee q \vee r, p \vee q \vee \neg r\}$. Notice that the extension rule is the inverse of resolution. The result of the extension rule is logically equivalent to the original clause.

In [2], we use the extension rule to extend clauses into maximum terms, in which all the atoms appear either positive or negatively. We showed that if a set of clauses only contains maximum terms, it is unsatisfiable iff it contains 2^m clauses. So the extension rule is used as a way of checking unsatisfiability in propositional logic. We use the following trick to count the number of different maximum terms instead of actually generating them.

Given a set of clauses $\Sigma=\{C_1, C_2, \ldots, C_n\}$, let M be the set of atoms which appear in Σ ($|M|=m$),let P_i be the set of all the maximum terms we can get from C_i by using extension rule, let S be the sum of all-different maximum terms we can get from Σ by using Extension rule, we will have:

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

$|P_i|=2^{m-|C_i|}$.

$$|P_i\cap P_j|=\begin{cases}0, \text{ There are complementary literals in } C_i\cup C_j.\\ 2^{m-|C_i\cup C_j|}, \text{ Otherwise.}\end{cases}$$

Knowledge Compilation Using the Extension Rule

In [1], we proposed a way of knowledge compilation. The idea is to using an EPCCL theory as the target language, which is defined below.

Definition 2 An EPCCL theory is a set of clauses in which each pair (of clauses) contains complementary literal(s).

The nice thing about an EPCCL theory is that, for an EPCCL theory, only the first n terms in Formula 1 will be nonzero. So it is enough to just compute the n terms in Formula 1. But the problem is that a lot of theories are not EPCCL theory in practice. So we need a way of converting an arbitrary propositional theory into an EPCCL theory. The following algorithm does this by making use of the Extension rule.

Algorithm KCER

Input: Let $\Sigma_1=\{C_1, C_2, \ldots, C_n\}$ be a set of clauses, $\Sigma_2=\Sigma_3=\varnothing$
While $\Sigma_1\neq\varnothing$
Loop
  Select a clause from Σ_1, say C_1, and add it into Σ_2
  While i<the number of clauses in Σ_1
  Loop
    While j<the number of clauses in Σ_2
    Loop
    If C_i and C_j contain complementary literal(s) Then skip
    Else if C_i subsumes C_j Then eliminate C_j from Σ_2
    Else if C_j subsumes C_i Then eliminate C_i from Σ_1
    Else Extend C_j on a “proper” variable using extension rule
    j:=j+1
    Endloop
    i:=i+1
  Endloop
  $\Sigma_3:=\Sigma_3\cup\Sigma_2, \Sigma_2:=\varnothing$
Endloop
Output: Σ_3 is the result of the compilation process.

We also showed that the output of Algorithm KCER produces an equivalent EPCCL theory.

Our method

Our method is largely based on the following rule.

Definition 3 Given two clauses $C_1 = C \vee a$, $C_2 = C \vee \neg a$, the Combination rule produces C as the result.

Given two clauses that differ only in one complementary literal, the Combination rule removes the complementary literal. Using the Combination rule, we get the following algorithm, which is a modification to Algorithm KCER.

Algorithm KCCR

Input: Let $\Sigma = \{C_1, C_2, \ldots, C_n\}$ be a propositional formula in CNF form.

Use algorithm KCER to produce an equivalent EPCCL theory Σ'.

While there are two clauses in Σ' that differ only in one complementary literal

Replace the two clauses with the result of the Combination rule

Theorem 1 Algorithm KCCR produces an EPCCL theory, which is equivalent to the original formula.

Proof. Since we already showed that algorithm KCER produces an equivalent EPCCL theory, we only need to show that the Combination rule does not destroy the property of EPCCL theory and preserves equivalence. It is obvious that the Combination rule preserves equivalence. We prove that the Combination rule does not destroy the property of EPCCL theory by induction on the number of uses of the Combination rule. Suppose that we use the Combination rule to combine $C_1 = C \vee a$ and $C_2 = C \vee \neg a$, and the result is C. For any other clause C' in Σ', C' has complementary literals with both C_1 and C_2, the only way that this can happen is that C' has complementary literals with C. So by replacing C_1 and C_2 by C, the property of EPCCL theory does not get destroyed.

Summary

In this paper, we proposed a new way for knowledge compilation. It is based on both the Extension rule and the Combination rule. The Extension rule is responsible for converting an arbitrary propositional theory into an EPCCL theory, and the Combination rule is responsible for compressing the result of the compilation. We showed that the Combination rule preserves equivalence and does not destroy the property of EPCCL theory.

Acknowledgement

This work is supported by Liaoning Provincial Natural Science Foundation under grant 201202202, Scientific Research Foundation of Liaoning Provincial Education Department under grant L2012388.

References

[1] Hai Lin and Jigui Sun: *Knowledge Compilation Using the Extension Rule,* Journal of Automated Reasoning, 32:93-102, 2004.

[2] Hai Lin, Jigui Sun and Yimin Zhang: *Theorem Proving Based on the Extension Rule,* Journal of Automated Reasoning 31:11-21, 2003.

[3] Selman, B., & Kautz, H. A (1991). "Knowledge compilation using Horn approximation." In Proceedings of AAAI-1991, 904--909.

[4] A. del Val. "Tractable databases: How to make propositional unit resolution complete through compilation." In J. Doyle, E. Sandewall, and P. Torasso, editors, Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, pages 551--561, Bonn, 1994. Morgan Kaufmann.

[5] R. Schrag. “Compilation for critically constrained knowledge bases.” In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), pages 510--515, 1996.

[6] P. Marquis. “Knowledge compilation using theory prime implicates.” In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), pages 837-- 843, 1995.