Incremental Knowledge Compilation Using the Extension Rule

Ying Liu 1, 4, Wenxiang Gu* 2, 4, Tingting Zou 3, Qingtao Huang 1, and Qiao Li 1
1. College of Humanities & Science of Northeast Normal University, Changchun, 130117, China
2. College of Computer Science and Technology, Jilin University, Changchun 130012, China
3. College of Basic Subjects Teaching, Changchun Architecture & Civil Engineering College, Changchun, 130607, China
4. School of Computer Science and Information Technology, Northeast Normal University, Changchun, 130117, China
*Corresponding author, Email: gwx@nenu.edu.cn

Abstract—Knowledge compilation by using the extension rule (KCER) has been recognized as a novel compilation approach, although this method can only deal with static knowledge bases. This paper proposes a novel incremental knowledge compilation method by using the extension rule so that it can tackle knowledge compilation problems in the dynamic environment. The method does not recompile the whole knowledge base, while it is to use the information of compiling original knowledge base to speed up the compiling process. The experimental results show that this method can deal with dynamic knowledge bases efficiently.

Index Terms—Dynamic Knowledge Base, Extension Rule, EPCCL, Incremental Knowledge Compilation

I. INTRODUCTION

Automatic reasoning is one of the important research fields of artificial intelligence, and its reasoning complexity is usually NP-complete. Knowledge compilation [1] is effective techniques that have been proposed for addressing such computational difficulties. It is used to network engineering [2], logical reasoning and other fields [3, 4, 5].

The central observation is that in Knowledge representation reasoning problems, the input to a problem is conceptually split in two parts: a knowledge base and a query. Typically, the knowledge base is not modified frequently, and the same knowledge base is applied to answer many queries. The idea of knowledge compilation is to split query answering into two phases: In the first one the knowledge base is preprocessed, thus obtaining an appropriate data structure (such a phase is sometimes called off-line reasoning); In the second phase, the query is actually answered using the output of the first phase (such a phase is sometimes called on-line reasoning). The key is that knowledge compilation needs to be done only once to be accessible for different queries. Hence, the compiling time can be amortized by many queries concerning the compiled knowledge base.

In 1987, Reiter and de Kleer [6] firstly put forward the concept of knowledge compilation, namely, compiling the knowledge base into the equivalent of the new one that can be tractable reasoning, and propose a knowledge compiled algorithm by using the prime implicate for target language. But the knowledge base obtained from this method is not complete for unit resolution.

In 1994 Alvaro del Val [7] improved above method, and put forward a new knowledge compilation approach to ensure the completeness of unit resolution. In 1995, Marquis [8] improved Reiter and de Kleer’s method and proposed the knowledge compilation approach based on a theory prime implicate. All of these methods are based on the prime implicate.

In 1996, Schrag [9] put forward a knowledge compilation approach by using prime implicat for the target language. The above methods can find the knowledge base which is equivalent to the original knowledge base by knowledge compilation, and then reasoning it, but not all of the knowledge bases exist the equivalent ones, so Selman and Kautz [10, 11] proposed an approximate knowledge compilation approach in 1996, which calculated least upper bound (LUB) and greatest lower bound (GLB) of the knowledge base, and found the Horn knowledge base which is approximate to the original knowledge base, then reasoning, but it didn’t answer all the queries of the original knowledge base.

At present, there are many target languages for knowledge compilation, such as the binary decision diagram[12], decomposable negation normal form (DNNF), which supported the target language for polynomial time logical operations, put forward by Darwiche in 1999 [13]. Very recently, Christian Muise1 et al. proposed a new method, DSHARP: Fast d-DNNF Compilation with sharpSAT [14], it is very efficient.

Knowledge compilation has become an efficient method for dealing with intractable reasoning problems for NP-complete or co-NP complete problems. Therefore, many researchers have focused their study on knowledge compilation for description logic. Literature [15] proposed a novel knowledge compilation method for description logic, which aims to further improve the knowledge compilation method based on extension rule, so that it can deal with description logic problems. They defined concept extension rule and EPCCL theory, and then present an algorithm compiling any concept into equivalence EPCCCL theory, and prove that
satisfiability-testing and subsumption-testing can be done in linear time in the size of the compiled concept.

Above knowledge compilation methods can only deal with the reasoning problems in static knowledge base, but knowledge base is usually dynamic. For example, when the knowledge base needs to add new clauses, the new knowledge base cannot ensure to keep the properties of tractable reasoning, so it must be recomposed. For those applications which need to knowledge base updated frequently, the above methods need greater time cost. In order to solve these problems, in 1993, Goran Gotic put forward the method for incremental knowledge compilation [16]. For a knowledge base, it gets the approximate LUB and GLB by using the horn approximation. When knowledge base adds a new clause, using the model updating method, it doesn’t recompile the entire knowledge base, but it needs to update the least horn upper bound (Horn-LUB) and greatest horn lower bound (Horn-GLB) of knowledge base and then gets the updated LUB and GLB.

Linhai [17] proposed a new method of automated theorem proving, namely extension rule, first it calculated all of maximum terms in the set of clauses, then judged the satisfiability by sets of clauses. This effective method used the including-excluding principle to reduce space complexity of judging satisfiability. Later, they expanded the extension rules to compile knowledge [18, 19], and put forward a new target language EPCL and an algorithm which can compile arbitrary knowledge base into EPCL language. Knowledge compilation approach by using the extension rule (KCER) is based on the extension rule whether compilation phase or extension reasoning phase. On the other hand, we applied the extension rule method to the problem of counting model solution, and put forward an algorithm of counting model using extension rule [20, 21].

The purpose of our paper is to promote the knowledge compilation approach based on extension rule, and make it deal with the knowledge compilation problems in dynamic knowledge base. We put forward an algorithm of incremental knowledge compilation by using the extension rule (IKCER), which can update the dynamic knowledge base. It doesn’t attempt to recompile the updated knowledge base, but uses the information which compiled original knowledge base to compile the compiled knowledge base and updated clauses. The experimental results show that the IKCER is able to deal with the update of dynamic knowledge base effectively. Compared with the KCER, our method takes less time, and increases less clauses.

This paper is organized as follows. In section 2, we introduce some related concepts and knowledge compilation based on extension rule. In section 3, we put forward the approach of incremental knowledge compilation by using the extension rule, and its algorithm. Section 4 shows the experimental results data through the concrete experimental results. Finally, section 5 summarizes our work.

II. BASIC KNOWLEDGE

First, we agree with some symbols in this paper. We use \( \sum \) to denote a knowledge base, namely a set of clauses in conjunctive normal form (CNF). C, D... denote the clauses, a, b... denote the atom, a, \( \neg a \)... denote the literal, and M denotes the set of all atoms that appear in knowledge base.

Definition 2.1 [17, 22] Given a non-tautology clause C and a set M, C is the maximum item on a set M, if it contains all atoms in M in either positive form or negative form.

Definition 2.2 [17, 22] Given a clause C and a set M, 
\[
D = \{C \lor a, C \lor \neg a \mid a \in M, \neg a \lor a \not\in C\}
\]
then we call the operation proceeding from C to D as the extension rule on C. We call D is the result of the extension rule.

Theorem 2.1 [17, 22] Given a clause C and its results D, which is applied extension rule, then a clause C is logically equivalent to D.

Theorem 2.2 [17, 22] Given a knowledge base \( \Sigma \), and its set of all atoms M (\( \{M \mid m\} \)). If all the clauses in \( \Sigma \) are all maximum terms on M, then \( \Sigma \) is unsatisfiable, iff it contains \( 2^n \) clauses which are different from each other.

According to the above theorems, a simple way to judge the satisfiability of knowledge base is to count the number of maximum terms, if the number is equal to \( 2^n \), then the knowledge base is unsatisfiable, otherwise is satisfiable. Therefore literature [10] realized this method using including-excluding principle.

Given a knowledge base \( \Sigma = \{C_1, C_2, \ldots, C_n\} \), and its set of all atoms M (\( \{M \mid m\} \)). Let \( P \) be the set of all maximum terms we can get from \( C_i \) by using extension rule, \( i=1, 2, \ldots, n \). Let S be the number of distinct maximum terms we can get from \( \Sigma \) by using the extension rule. Then by using the including-excluding principle, we will have:

\[
S = \sum_{i=1}^{n} |P_i| - \sum_{1 \leq j < \ell \leq n} |P_j \cap P_\ell| + \sum_{1 \leq i < \ell < m} |P_i \cap P_j \cap P_\ell| - \ldots + (-1)^{n-1} |P_1 \cap P_2 \cap \ldots \cap P_n|
\]

where, \( |P_i| = 2^n - 1 \)

For general knowledge base, it needs to calculate \( 2^{n-1} \) items of the formula (1). However, when any two clauses all contain the pairs that contain complementary literal(s) in the knowledge base, it only needs to calculate the former n items of formula (1), and reduces the space complexity. Therefore, the literature [18] puts forward EPCL theory.

Definition 2.3 [17, 22] An EPCL theory is a set of clauses in which each pair (of clauses) contains complementary literal(s).

Literature [18] proved EPCL theory is in the tractable satisfiability class and tractable entailment class. Therefore, EPCL theory can be used as target language of knowledge compilation. The basic process of the
knowledge compilation by using extension rule (KCER) [18] is that: given a proposition knowledge base, we firstly extend a clause by using extension rule, and it and other clauses contain pairs that contain complementary literal(s), and then add the extended clauses into the compiled knowledge base, in this cycle, until all clauses have been processed in original knowledge base. The compiled knowledge base is equivalent to the original knowledge base, and it is an EPCCL theory.

III. INCREMENTAL KNOWLEDGE COMPOSITION USING THE EXTENSION RULE

Given a proposition knowledge base \( \Sigma \), and the compiled knowledge base \( \Sigma_1 \) is equivalent to it. The main idea of incremental knowledge compilation is that, when the original knowledge base \( \Sigma \) add a new clause \( C \), it does not recompile the updated knowledge base \( \Sigma \cup \{C\} \), but use the compiled information in the knowledge base \( \Sigma_1 \) which is equivalent to the original knowledge base, and speed up the compilation of the updated knowledge base \( \Sigma \cup \{C\} \), then get the equivalent knowledge base \( \Sigma' \).

We use the extension rule to the incremental knowledge compilation, and propose a new approach of incremental knowledge compilation by using the extension rule. Given a proposition knowledge base \( \Sigma \), we choose the EPCCL as the target language of knowledge compilation, and compile the knowledge base \( \Sigma_1 \) so we can get a knowledge base \( \Sigma_1 \) which is equivalent to it, where any two clauses contain complementary literals. If any clauses are compiled by using extension rule in the knowledge base \( \Sigma_1 \), the compiled knowledge base is still EPCCL theory. According to the properties, when adding one or more than one clauses to knowledge base, we compile the clauses in knowledge base \( \Sigma_1 \) by using extension rule, make them and clause \( C \) contain complementary literals, so that we can get a new EPCCL knowledge base \( \Sigma_3 \), it is equivalent to the updated knowledge base \( \Sigma \cup \{C\} \).

Now, we give the algorithm for Incremental Knowledge Compilation using Extension Rule (IKCER). First Chosen one clause \( C_i \) in knowledge base \( \Sigma_1 \), and add into the knowledge base \( \Sigma_2 \), then judged the relationship between \( C_i \) and \( C \), deleted a clause or extended \( C_i \) by using extension rule, it is ensure that any clauses in \( \Sigma_2 \) contain the pairs that contain complementary literal(s) with clause \( C \). Repeating the process until it has handled all the clauses in \( \Sigma_1 \). At last, it merges \( C \) into \( \Sigma_2 \). Using solved the information in original knowledge base, and the method accelerates the compilation for the updated knowledge base, and improves the efficiency of the compilation.

Algorithm Ikcer

| Input: Knowledge base \( \Sigma \), compiled knowledge base \( \Sigma_1=\{C_1, C_2, \ldots, C_n\}\), \( \Sigma_2=\emptyset \), updated clause \( C \). |
| Output: \( \Sigma_3 \) is the result of knowledge compilation process |

While \( \Sigma_1 \neq \emptyset \)
Loop
Select a clause \( C_i \) from \( \Sigma_1 \), i=1, 2, ..., n, \( \Sigma_3=\Sigma_1 \cup \{C_i\} \); If \( C_i \) and \( C \) contain pairs that contain complementary literal(s) Then do-nothing;
Else if \( C_i \) entails \( C \) Then \( \Sigma_3=\Sigma_1 \cup \{C_i\} \);
Else
Endloop
Return \( \Sigma_3 \);
include the pairs with the clauses in $\Sigma_1$, so before moving the clauses in $\Sigma_2$ into $\Sigma_3$, any clauses in $\Sigma_2$ contain the pairs with clause C, the clauses in $\Sigma_2$ also contain the pairs with the clauses in $\Sigma_1$. Although the clauses in $\Sigma_2$ may be further extended by executing the algorithm, the clauses in $\Sigma_2$ contain the pairs with all of the extended clauses in $\Sigma_1$, and with clause C. So, any clauses in the result of IKCER $\Sigma_3$ contain the pairs that contain complementary literal(s), $\Sigma_3$ is an EPCCL theory.

**Theorem 3.2** The algorithm IKCER is effective and complete.

**Proof.** According to the theorem 3.1, the algorithm IKCER returns the knowledge base which is an EPCCL theory, so the algorithm is effective. Given a knowledge base $\Sigma$ and a clause C, if there is an EPCCL knowledge base $\Sigma'$, it is equivalent to $\Sigma \cup \{C\}$, the algorithm finally returns $\Sigma'$ as the result. Therefore the algorithm is complete.

Knowledge compilation approach compiles the original knowledge base into the tractable reasoning one, such as Horn knowledge base and EPCCL knowledge base. Then the reasoning problem can be finished in polynomial time, and greatly reduce the time complexity.

IV. EXPERIMENTAL RESULTS

In this section, we compare the algorithm IKCER with the original algorithm KCER [18] on random SAT instances. For the update problems of dynamic knowledge base, we test a great number of examples, the efficiency of IKCER is much better than KCER.

![Figure 1. Random 2-SAT problem, comparing the number of increased clauses of IKCER with KCER on random 2-SAT problem, N=20, M=40.](image1)

![Figure 2. Random 3-SAT problem, comparing the number of increased clauses of IKCER with KCER on random 3-SAT problem, N=20, M=40.](image2)

![Figure 3. Random 4-SAT problem, comparing the number of increased clauses of IKCER with KCER on random 4-SAT problem, N=20, M=40.](image3)

The test cases are obtained by a random generator in the experiment. They are $<20,40,2>$, $<20,40,3>$, $<20,40,4>$. The first parameter denotes the number of variables N in knowledge base, the second denotes the number of clauses M, and the third denotes the maximum length of clauses K. Each clause is obtained at random by choosing K variables from the range of the N variables, and each variable’s probability of negative or positive is equal to 0.5. For each type of difficulty for test cases, it randomly generates 20 instances, and solves them respectively, finally uses the average value as a result.

We compare the number of increasing clauses for IKCER and KCER. From the figure 1 to figure 3, we can see that the number of increase clauses for IKCER is less than KCER’s. Usually, KCER’s number of increase clauses is 10-50 times of IKCER’s. The next, we compare the time cost of them. From figure 4 to figure 6, we can see that the algorithm IKCER use less time to compile the incremental knowledge base, on average, it will save 1-10 times than KCER.

V. CONCLUSIONS

This paper proposes an algorithm for incremental knowledge compilation by using the extension rule. For an updated knowledge base, it does not need to recompile the entire knowledge base, but solves the problem of incremental knowledge compilation by using the information in knowledge base, and improves the efficiency. The experimental results show that, for the problems which have different scales and difficulties, our algorithm IKCER is about 1-10 times of KCER on efficiency, and the number of increase clauses is about 1/10 to 1/50 times of KCER’s. Therefore, the approach of
incremental knowledge compilation by using the extension rule can be more effective to deal with the compilation problem for knowledge base when the knowledge base changed. In future work, we would like to implement our methods and other methods for description logic, and compare these methods for description logic problems.

ACKNOWLEDGMENTS

This research was supported by the National Nature Science Foundation of China under Project Number: 61070084, and the "Twelfth five-year" science and technology research of education department of Jilin under Project Number: 556.

REFERENCES


**Ying Liu** was born in JiLin province, China in 1980. She received her bachelor’s degree in JiLin normal University, Jilin, China in 2003. She received her M. Sc. degree in computer science from Northeast Normal University in 2008. Now, she is a Ph. D. candidate at the School of Computer Science and information technology, Northeast Normal University, Changchun, China. Her current research interests include intelligence plan, plan recognition, intelligence scheduling and network security.

**WenXiang Gu** was born in JinLin province, China in 1947. Now, he is a professor at the School of Computer Science and information technology, Northeast Normal University, Changchun, China. His current research interests include intelligence plan, plan recognition and network security.

He graduated from the east to Beijing normal university department of mathematics and teaches in 1972. From 1972 to 1992 he was in the northeast normal university department of mathematics teaching, was appointed associate professor in 1990. Then, in 1996 he was hired as professor, and hired as a doctoral supervisor in 2006.

**Tingting Zou** was born in JiLin Province, China in 1984. She received her bachelor’s degree in northeast normal University, JiLin, China in 2006. She received her M. Sc. degree in computer science from Northeast Normal University in 2009. Now, she is a Ph. D. candidate at the School of Computer Science and technology, JiLin University, Changchun, China. Her current research interests include intelligence plan, plan recognition, date mining and knowledge complication.

**Qingtao Huang** was born in JiLin province, China in 1979. She received her bachelor’s degree in JiLin normal University, JiLin, China in 2001. She received her M. Sc. degree in computer science from Northeast Normal University in 2009. Her current research interests include intelligence plan, plan recognition, date mining and knowledge complication.

**Qiao Li** was born in JiLin province, China in 1984. She received her bachelor’s degree in northeast normal University, JiLin, China in 2007. She received her M. Sc. degree in computer science from Northeast Normal University in 2012. Her current research interests include intelligence plan, plan recognition, intelligence scheduling and network security.