Propositional Extension Rule with Reduction

Xia Wu†,††, Jigui Sun†,††, Shuai Lv†,†† and Minghao Yin†,††

†College of Computer Science and Technology, Jilin University, 130012 Changchun, China
†† Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education, 130012 Changchun, China

Summary
Method based on extension rule is a new method for theorem proving. It is, in a sense, potentially a complementary method to resolution based method. ER is the basic extension rule algorithm. In order to increase its efficiency, this paper improves it by some reduction rules. And then the soundness and completeness of the improved algorithm is proved. The experiment results show the improved algorithm not only increase the efficiency but also keep the characteristic of the extension rule method, namely it is still potentially complementary methods to resolution based methods. In order to enhance the reasoning speed by making the best of the respective characteristic of extension rule method and resolution method, this paper proposes a combined algorithm with reduction rules, which combines the extension rule and resolution. It is also sound and complete.

Key words:
Extension rule, theorem proving, satisfiability, reduction

1. Introduction

Automated theorem proving (ATP) has matured into one of the most advanced areas of computer science. Fields where ATP has been successfully used include logic, mathematics, computer science, engineering, and social science. There are potentially many more fields where ATP could be used, including biological sciences, medicine, commerce, etc [1]. Many significant problems have been, and continue to be, solved using ATP. The fields where the most notable successes have been achieved are mathematics, and software generation and verification [2], protocol verification, and hardware verification [3].

The usually used deduction methods in ATP include resolution based method, tableau based method, sequent calculus and nature deduction method, etc. The traditional idea used in TP is to try to deduce the empty clause to check the unsatisfiability. Resolution based TP is a paradigm of this idea. But extension rule based TP [4] proceeds inversely to resolution. Namely, extension rule based TP checks the unsatisfiability by deducing the set of clauses consisting of all the maximum terms. Therefore, it is a new theorem proving method.

Extension rule (ER) method can be considered, in a sense, to be a method dual to resolution. Because ER is more efficient when set of clauses includes more complementary literals but low efficient when set of clauses includes less complementary literals. In order to improve the efficient, the method is modified, so that it can be used in ATP better. The experiment results in section 3 show improved methods achieve more efficiency in most cases.

This paper is organized as follows. The ER method is introduced briefly and improved in section 2. Moreover, the soundness and completeness of improved ER is proved. Section 3 compares the relative works by detailed experiment results. In section 4, in order to propose a combined algorithm, a directional resolution with reduction rules is given at first. And then the combined algorithm DR–ER is proposed. Furthermore, the soundness and completeness of them are proved. A conclusion is drawn in the final part.

2. Propositional Extension Rule and Its Improvement

We run back over the central idea of the extension rule based TP at first. The details can be found in [4]. The extension rule method uses the inverse of resolution together with the inclusion-exclusion principle to solve TP problems. The set of all the maximum terms is unsatisfiable. Once a set of clauses deduces it, we can decide the clause set is unsatisfiable, the deduction method used here is called extension rule. While the resolution method goes this way: Since the empty clause is unsatisfiable, once a set of clauses deduces an empty clause, it can decide that the clause set is unsatisfiable, the deduction method used is resolution. The extension rule is defined as follows.

Definition 1. Given a clause C and a set M: $C'=\{C\vee a, C\vee\neg a \mid$ "a" is an atom, $a\in M$, "$\neg$a" and "a" does not appear in C$\}$. The operation proceeding from C to C′ is the extension rule on C. C′ is the result of the extension rule.

So if we want to decide whether a set of clauses is satisfiable, we can proceed by finding an equivalent set of clauses such that all the clauses in it are maximum terms

Manuscript revised August 22, 2005.

by using the extension rule. Evidently, all of the maximum term set consist of n atoms must include 2^n elements. The number of maximum term extended by a clause set can be calculated by using the inclusion-exclusion principle.

Given a set of clauses $\Phi=\{C_1, C_2, \ldots, C_n\}$, let M be the set of atoms which appear in Φ ($|M|=m$). Let P_i be the set of all the maximum terms gotten from C_i by using the extension rule, and let S be the number of distinct maximum terms gotten from Φ. By using the Extension rule, we will have:

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

$$|P_i| = 2^{m-|C_i|} \quad . \quad (2)$$

$$|P_i \cap P_j| = \begin{cases} 0, & \text{There are complementary literals in } C_i \cup C_j; \\ 2^{m-|C_i \cup C_j|}, & \text{Otherwise.} \end{cases} \quad . \quad (3)$$

Example 2. Check the satisfiability of the clause set $\Phi=\{\neg A \vee B \vee \neg C, A \vee C, \neg A\}$ by formula (1).

The maximum term number extended by Φ: $S=2^0+2^1+2^2-0-2^0-0+0=6$

Because $6<2^3$, clause set $\Phi=\{\neg A \vee B \vee \neg C, A \vee C, \neg A\}$ is satisfiable.

The extension rule algorithm in proposition logic is given in [4]. Here we make some modifications to enhance the algorithm's efficiency. There are some clauses in the set of clauses having nothing to do with the satisfiability, such as the clause containing pure literal, the clause including tautology and the clause implied by other clause, etc. Hence, these clauses can be deleted. Several rules in DP algorithm [5] are used to simplify the given clause set, namely, the literals or clauses unrelated to the satisfiability are deleted. Then the satisfiability is checked.

Unit resolution is a very fast but incomplete method to decide the satisfiability, the single literal rule in DP is in fact the unit resolution. During the single literal rule is used to reduce the primitive clause set, it is actually running a more efficient but incomplete method to check the satisfaibility. If the clause set becomes empty then it is satisfiable. Else, if the clause set includes empty clause then it is unsatisfiable. Otherwise, a reduced clause set is given.

When the pure literal rule is used, if the clause set becomes empty then it is satisfiable else a reduced clause set is given. All of the reduced rules are listed below.

Tautology rule: Deleting all the tautologies in the set of clauses Φ. Let the surplus clause set be Φ'. Then Φ is unsatisfiable if and only if Φ' is unsatisfiable.

Definition 3. Say the literal L in the clause set Φ is pure if and only if $\neg L$ is not in Φ.

Pure literal rule: If the literal L in the clause set Φ is pure then delete all the clauses including L. Let the surplus clause set be Φ'. (a) If Φ' is empty then Φ is satisfiable; (b) otherwise Φ' is unsatisfiable.

Definition 4. C_1 and C_2 are any two clauses in set of clauses Φ, say C_1 includes C_2 if every literal in C_1 is also in C_2.

Inclusion rule: Suppose C_1 and C_2 are two clauses in the clause set Φ, where C_1includes C_2. Deleting the clause C_2 from Φ and let the surplus clause set be Φ', then Φ is unsatisfiable if and only if Φ' is unsatisfiable.

Single literal rule: If there is a single literal L in set of clauses Φ, then delete all of the clauses including L. Let the surplus clause set be Φ'. (a) If Φ' is empty then Φ is satisfiable; (b) Otherwise if Φ' is not empty, then delete all of the literals $\neg L$ from the clauses including it in Φ' . Φ' is unsatisfiable if and only if Φ'' is unsatisfiable (suppose there is a unit clause $\neg L$ in Φ', a empty clause □ is achieved by deleting $\neg L$).

Denote tautology rule by RT, pure literal rule by RP, inclusion rule by RI, and single literal rule by RS. Let RL ={RT, RP, RI, RS}, the improved extension rule algorithm in propositional logic is given below.

Algorithm ER1

1. Let $\Phi=\{C_1, C_2, \ldots, C_n\}$.
   While Φ satisfies any rule in RL
   Loop
   Φ_1:= using RL to deal with Φ
   If Φ_1 is empty then return satisfiable
   $\Phi:=\Phi_1$
   Endloop
2. i:=1, sum:=0
3. **while** i<=n
   Loop
   For all sets S that contain i clauses
   Loop
   Union=the union of all the clauses in S
   If there are no complementary literals in Union
   then number:=$2^{m-|Union|}$
   Else Number:=0
   If i mod 2=1 **then** sum:=sum+number
   Else Sum:=sum-number
   Endloop
   If i mod 2=0 and sum=2^m **then** return unsatisfiable
   Else If i mod 2=1 and sum<2^m
   then return satisfiable
   i:=i+1
   Endloop
4. **If** sum=2^m **then** return unsatisfiable
   Else return satisfiable

Theorem 5. Algorithm ER1 is sound and complete for proposition logic theorem proving.

Proof. It has been proved that tautology rule, pure

literal rule, inclusion rule and single literal rule can not change the unsatisfiability of the primary clause set [5]. Thus by the soundness and completeness of algorithm ER [4], the soundness and completeness of algorithm ER1 is straightforward. Q.E.D

The efficiency of algorithm ER relies on the number of nonzero terms counted by formula (1). Clearly, the worst-case time complexity of Algorithm ER is exponential. However, there are cases in which Algorithm ER is tractable. For example, if each pair of clauses in a set contains complementary literal(s), then the complexity of Algorithm ER will be linear in the number of clauses because only the first n terms in Formula (1) are nonzero terms and need to be computed. Intuitively, in this case resolution based methods are likely to be inefficient since there are potentially many resolutions that need to be performed. By contrast, consider the case in which if there are no complementary literals at all. In this case it would have to compute all the 2^m terms in order to decide its satisfiability using the extension rule. But its satisfiability can be decided immediately by using a resolution based method. (Actually no resolution can be or needs to be performed).

Consider the improved algorithm ER1, when there are no complementary literals at all, it can deduce the satisfiablility by using the pure literal rule. Thus it need not invoke the algorithm ER which is inefficient in this case. Moreover, by using the single literal rule to reduce the clause set, it is in fact to use the fast unit resolution to check the satisfiability. So it is naturally the behavior of algorithm ER1 is better than algorithm ER. The experiment results in section 3 show this.

3. Experimental Results

Some elementary experimental results are reported in this section to show how ER1 perform. Since first order theorem proving is reduced to a series of ground-level satisfiability problems, the behavior of ER1 will affect the efficiency of first order ER directly. So the comparison between improved algorithms and primal ones is very important.

In order to explain the relationship between ER and the number of the pairs of clauses in a set contains complementary literal(s) intuitively, [4] gives the definition of complementary factor.

Definition 6. Given a set of clauses $\Phi=\{C_1, C_2, \ldots, C_n\}$, the complementary factor CF of the set is the ratio of the number of pairs that contain complementary literal(s) to the number of all the pairs in the set. That is $S/[n(n-1)/2]$, where S stands for the number of pairs that contain complementary literal(s).

Although it is difficult to calculate the time complexity precisely by using the CF, but the experiment results in [4] show the higher the CF of a problem is, the more efficient algorithm ER can be expected to be. Our experiment results show the efficient of improved algorithms still has such relation to CF, but not so close like algorithm ER.

Three algorithms are compared in this section. They are ER1 proposed in this paper, ER given in [4] as well as DR proposed in [5]. There is an algorithm DR1 in table 1 and table 2. We will discuss it in detail in the next section.

The instances are obtained by a random generator. It takes as an input the number of variable n, the number of clauses m and the most length of each clause k, and obtains each clause randomly by choosing k variables from the set of variables which number less than or equal to n and by determining the polarity of each literal with probability p=0.5.

(10,50,10) denotes a set of clauses, which has 10 variables and 50 clauses and each clause length is not more than 10. There are two decimal fractions below each clause set in table 1, they are the CF of primal set of clauses and reduced set of clauses. There is just one decimal fraction below each clause set in table 2. It is the CF of primal set of clauses. The reason is the satisfiability can be deduced directly during reducing and need not to generate the reduced set. Non-0 terms denotes the nonzero terms generated by ER and ER1 during reasoning. Res-numbers denotes the resolution performed by DR during reasoning. Result denotes the returned result, i.e. the saisfiability or unsatisfiability. Time denotes the total time used by procedure, and the precision is 1 millisecond.

All of the clause sets in table 1 will not be empty or contain empty clause after reduced. When CF of the primal set of clauses is larger, CF of the reduced clause set becomes smaller generally. But in most cases, ER1 still outperform ER. It is because CF becomes smaller, but the variable number and the clause number in the clause set become smaller corresponding. So that the number of the nonzero terms needed to extend is cut down and the speed becomes fast. Seldom, ER1 is inefficient than ER. We think it is because though the variable number and the clause number decrease the complementary literal(s) for any pair is cut down excessively so as to decrease the efficiency. Here, DR is inefficient than other two algorithms evidently.

When CF of the primal clause set is smaller, CF of the reduced clause set becomes larger on the contrary. So the behavior of ER1 is much better than ER. Here, DR is efficient than ER1 and much more efficient than ER. All of above show ER1 enhance the efficiency largely and keep the characteristic of the extension rule method. All of the reduced clause sets in table 2 are empty or contain empty clause. The experimental results show the efficiency of ER1 has nothing with CF in this case. The behavior of ER1 is as good as DR and even better than DR

sometimes. Furthermore, the behavior of it is much better than ER.

Table1: Reduced clause set is neither empty nor contain empty clause

Examples		DR		ER	
		DR	DR1	ER	ER1
(10,50,10) 0.654694 0.536797	Non-0 terms	——	——	1979	477
	Res-numbers	2309	>50000	——	——
	Result	SAT	SAT	SAT	SAT
	Time	1.546	>500.00	0.062	0.016
(10,50,10) 0.641633 0.456522	Non-0 terms	——	——	4990	1568
	Res-numbers	822	52	——	——
	Result	SAT	SAT	SAT	SAT
	Time	0.109	0.000	0.218	0.031
(10,50,10) 0.604082 0.423333	Non-0 terms	——	——	7520	1991
	Res-numbers	3449	156	——	——
	Result	SAT	SAT	SAT	SAT
	Time	2.953	0.015	0.281	0.047
(10,50,10) 0.617959 0.576667	Non-0 terms	——	——	3725	743
	Res-numbers	5557	>50000	——	——
	Result	SAT	SAT	SAT	SAT
	Time	9.390	>500.00	0.141	0.016
(10,50,10) 0.571429 0.400000	Non-0 terms	——	——	7745	1610
	Res-numbers	1159	59	——	——
	Result	SAT	SAT	SAT	SAT
	Time	0.328	0.000	0.219	0.031
(10,50,10) 0.670707 0.670707	Non-0 terms	——	——	144911	144911
	Res-numbers	>50000	>50000	——	——
	Result	UNSAT	UNSAT	UNSAT	UNSAT
	Time	>500.00	>500.00	23.343	24.781
(10,50,10) 0.352653 0.429885	Non-0 terms	——	——	93821	4958
	Res-numbers	66	88	——	——
	Result	UNSAT	UNSAT	UNSAT	UNSAT
	Time	0.000	0.000	6.078	0.094
(10,50,10) 0.351020 0.463054	Non-0 terms	——	——	116159	2047
	Res-numbers	78	61	——	——
	Result	UNSAT	UNSAT	UNSAT	UNSAT
	Time	0.000	0.000	8.594	0.031

Table 2 Reduced clause set is empty or contain empty clause

Examples		DR		ER	
		DR	DR1	ER	ER1
(10,50,10) 0.438367	Non-0 terms	——	——	64348	——
	Res-numbers	181	——	——	——
	Result	UNSAT	UNSAT	UNSAT	UNSAT
	Time	0.016	0.000	3.703	0.000
(10,50,10) 0.444898	Non-0 terms	——	——	67367	——
	Res-numbers	32	——	——	——
	Result	UNSAT	UNSAT	UNSAT	UNSAT
	Time	0.000	0.000	2.531	0.000
(10,50,10) 0.415510	Non-0 terms	——	——	239678	——
	Res-numbers	37	——	——	——
	Result	SAT	SAT	SAT	SAT
	Time	0.000	0.000	12.484	0.000
(10,50,10) 0.460408	Non-0 terms	——	——	38248	——
	Res-numbers	112	——	——	——
	Result	SAT	SAT	SAT	SAT

	Time	0.000	0.000	1.547	0.000

4. Algorithm DR1 and DR–ER

We introduce the central idea of the directional resolution at first. The details can be found in [6]. DR is an ordering-based restricted resolution. Given an arbitrary ordering of the propositional variables, we assign to each clause the index of the highest ordered literal in that clause. Then we resolve only clauses having the same index, and only on their highest literal. The result of this restriction is a systematic elimination of literals from the set of clauses that are candidates for future resolution.

Use the reduced rules above to improve the algorithm DR. Let RL={RT, RP, RI, RS}, the improved algorithm DR1 in propositional logic is given below.

Algorithm DR1

1. Let $\Phi=\{C_1, C_2, \ldots, C_n\}$.
   While Φ satisfies any rule in RL
   Loop
   Φ_1:= using RL to deal with Φ
   If Φ_1 is empty **then** return satisfiable
   $\Phi:=\Phi_1$
   Endloop
2. Let $\Phi=\{C_1, C_2, \ldots, C_p\}(p\leq n)$, an ordering $d=Q_1, \ldots, Q_n$ of its variables.
3. Generate an ordered partition of the clauses, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all the clauses whose highest literal is Q_i.
4. **For** i=n to 1
   (a) Resolve each pair $\{(A\vee Q_i), (A\vee\neg Q_i)\}\subseteq bucket_i$.
   (b) **If** $R=A\vee B$ is empty **then** return satisfiable
   Else determine the index of R and add it to the appropriate buchet.
5. $E_d(\Phi):=\cup_i bucket_i$.
6. **If** $E_d(\Phi)$ is equivalent to Φ **then** return satisfaible
   Else return unsatisfiable.

Theorem 7. Algorithm DR1 is sound and complete for proposition logic theorem proving.

Proof. It has been proved that tautology rule, pure literal rule, inclusion rule and single literal rule can not change the unsatisfiability of the primary clause set [5]. Thus by the soundness and completeness of algorithm DR [6], the soundness and completeness of algorithm DR1 is straightforward. Q.E.D

All of the clause sets in table 1 will not be empty or contain empty clause after reduced. When CF of the primal set of clauses is more than 0.5, but CF of the reduced clause set is less than 0.5, DR is inefficient than all algorithms based on extension rule. Yet DR1 is more efficient than them. When CF of the reduced clause set is more than 0.5, the two algorithms based on extension rule outperform than DR and DR1.

All of the reduced clause sets in table 2 are empty or contain empty clause. Now the behaviors of ER1 and DR1 are the same. The experiment results in table 2 show DR1 is as good as DR and even better than DR sometimes.

Extension rule method can be considered, in a sense, dual to resolution method. In order to make best use of the advantage of the two reasoning method, we believe it will get an efficient deduction method by combining them. The experimental results in section 4 also show it is feasible to combine the two methods and it is helpful to reduce the primary clause set by some reduction rules in advance. So the main idea of the combined algorithm is following. Firstly, the given clause set is reduced by some reduction rules. Then compute CF of the reduced clause set. If the CF is great than or equal to 0.5 then call algorithm ER to check the satisfiability else call algorithm DR to check the satisfaibility. The combined algorithm DR–ER in propositional logic is given below.

Algorithm DR–ER

1. Let $\Phi=\{C_1, C_2, \ldots, C_n\}$.
   While Φ satisfies any rule in RL
   Loop
   Φ_1:= using RL to deal with Φ
   If Φ_1 is empty **then** return satisfiable
   $\Phi:=\Phi_1$
   Endloop
2. Let $\Phi=\{C_1, C_2, \ldots, C_p\}(p\leq n)$, compute its CF.
3. **If** CF >=0.5 **then** invoke Algorithm ER
   Else invoke Algorithm DR.

Theorem 8. Algorithm DR–ER is sound and complete for proposition logic theorem proving.

Proof. It has been proved that tautology rule, pure literal rule, inclusion rule and single literal rule can not change the unsatisfiability of the primary clause set [5]. Thus by the soundness and completeness of algorithm BDR [6] and IER [4], the soundness and completeness of algorithm DR–ER is straightforward. Q.E.D

5. Conclusions

The aim we improve the extension rule method is not only to increase the efficiency in propositional logic but also to accelerate the reasoning speed of first order extension rule. Since first order theorem proving is reduced to a series of ground-level satisfiability problems, the behavior of propositional extension rule will affect the behavior of first order extension rule directly. In a word, our improvement will increase the efficiency of first order extension rule method.

Directional Resolution (DR) is the fastest resolution based theorem proving method in propositional logic. By analyzing the experimental results, we find it will generate a more efficient algorithm by combining DR and ER.

Namely, a given clause set is reduced by some reduction rules at first, and then deduce the saitsfiability by the combined algorithm of DR–ER. The algorithm DR–ER is just a primary idea. The bounded value 0.5 of CF is perhaps not the best one, so we will think over which bounded value is most suited in the future work. [4] has showed it is difficult to illuminate the number of pairs that contain complementary literal(s) precisely by using the complementary factor. So we believe there must be a better criterion to illuminate the number of pairs that contain complementary literal(s). The new criterion should be more accurate than CF to indicate when to invoke ER or DR. It is also one of our future works.

After above future works are finished, we will realize the combined algorithm DR–ER and compare it with relative works.

It is well known many resolution methods are widely used in first logic and many famous first order theorem provers [7][8] are based on them. The algorithm DR–ER proposed in this paper is just a base of the combined algorithm in the first order logic. Thus we will realize the first order extension rule algorithm and give a combined reasoning algorithm in first logic in the future. Because it can make best of the advantage of resolution methods and extension rule methods, the combined algorithm in the first logic deserve us to expect.

Acknowledgments

This paper was supported by National Natural Science Foundation of China (Grant No.60273080, 60473003), the Science and Technology Development Program of Jilin Province of China (Grant No.20040526) and the Outstanding Youth Foundation of Jilin Province of china (Grant No.20030107).

References

[1] J.A. Robinson, et al. eds., Handbook of Automated Reasoning, Elsevier Science Publishers, 2000.
[2] P. Fenkam, M. Jazayeri and G. Reif, “On methodologies for constructing correct event-based applications”, Proc. 3th International Workshop on Distributed Event-Based Systems, Edinburgh, UK, 38-43, 2004.
[3] J. Kubica, E. G. Rieffel, “Collaborating with a genetic programming system to generate modular robotic code”, Proc. Genetic and Evolutionary Computation Conference, New York, USA, 804-811, 2002.
[4] H. Lin, J. G. Sun and Y. M. Zhang, “Theorem proving based on extension rule. Journal of Automated Reasoning.” Vol.31, 11-21, 2003.
[5] X. H. Liu, The theorem proof based on resolution (in Chinese), Science Press, Beijing, 1994.
[6] R. Dechter, I. Rish, “Directional resolution: The Davis-Putnam procedure, revisited”, Proc. 4th International Conference on Principles of KR&R, Bonn, Germany, 134—145, 1994.
[7] C. Dixon, “Using Otter for Temporal Resolution”, Advances in Temporal Logic, vol.16, 149-166, 2000.
[8] Gail W. Pieper ed., Automated reasoning and the discovery of missing and elegant proofs, Rinton Press, 2003.

Xia Wu received the B.E. and M.S degrees in Computer Science from Northeast Normal University of China in 1999 and 2002, respectively. She is a Doctor student of Prof. Sun in software theory from Jilin University of China till now . Her research interest includes automated reasoning in classical logic and non-classical logic.

Jigui Sun received the Dr. degree in Computer Science from Computer Science Department of Jilin University of China in 1993, and was promoted professor in July 199**7.** He is currently head of Ministry of Education Key Laboratory of Symbolic Computation and Knowledge Engineering and associate dean of College of Computer Science and Technology, Jilin University. He is director of China Computer Association and committee member of the sub-division of computer science and technology for guiding higher education of the Ministry of Education. He is the Ministry of Education's Distinguished Talent for the New Century. Sun's research work focuses on intelligent information processing.