FILOMAT (Niš) 9:3 (1995), 591-597
Algebra, Logic & Discrete Mathematics
Niš, April 14–16, 1995.

ORDERED LINEAR RESOLUTION AS THE BASE OF THE SYSTEM FOR AUTOMATIC THEOREM PROVING

Ivana F. Berković

ABSTRACT. U radu se opisuje uređena linearna rezolucija sa markiranim literalima i njene specifičnosti. Da bi se očuvala potpunost metode izvršena je modifikacija algoritma za odredivanje rezolvente.

Na bazi modifikovane uređene linearne rezolucije sa markiranim literalima izgrađen je sistem za automatsko dokazivanje teorema. Sistem je implementiran na PC - računaru i dopušta varijabilne strategije pretraživanja. Pretpostavke i tvrdenja koje treba dokazati, zapisuju se odgovarajućim formulama predikatskog računa prvog reda. U radu se daje opis implementiranog sistema za automatsko dokazivanje teorema, prikazuju se njegove karakteristike i oblasti primene. Posebno se razmatra odnos ovakvog automatskog dokazivača teorema i Prolog-a.

1. Introduction

Automated reasoning is very important area in Artificial Intelligence, but the common sense is difficult to model in a computer. The needed knowledge is not easy to represent. Another problem is how it can deduce something from a set of facts, or how it can prove that a conclusion follows from a given set of premises. Computational logic, based on formulations by some formal-language (propositional logic, predicate logic), provides problem-solving methods.

The developing of theorem-proving can be divided in two directions. The first direction is pure automated theorem proving, which is mostly resolution-based. The other approach is non-resolution-based theorem proving or natural deduction, which includes some heuristics and user-supplied knowledge.

2. The Rule of Ordered Linear (OL) Resolution with Marked Literals

The most popular method for automatic theorem proving is the resolution method, which is discovered by J. A. Robinson in 1965 ([2], [5]). Resolution method is a syntactic method of deduction. This procedure is a general automatic method for determining if a theorem (conclusion) follows from a given set of premises (axioms). Each formula will be transformed to the clauses form. *Reduction ad absurdum* is in the basis of resolution method. Resolution rule will be applied on the set of clauses (axioms) which was expanded by negating the desired conclusion in clause form.

Since 1965., many resolution forms and techniques are developed because the pure resolution rule has been unable to handle complex problems. Also, many resolution theorem provers are created.

Ordered Linear (OL) resolution rule with marked literals ([6]) increases efficiency and doesn't disturb completeness of pure resolution rule.
The generating process of OL-resolvent from central clause (d1) and auxiliary clause (d2):

1. Redesignate variables (without common.variables in the clauses).
2. Determine universal unificator Θ for last literal of d1 and k-literal ($k = 1, 2, ...$) of d2 (if it exists for some k, else it is impossible to generate OL-resolvent for specification clauses).
3. Create resolvent with marked last literal in d1Θ and add the rest of clause d2Θ without k-literal (d1Θ and d2Θ are clauses, which were formed by universal unificator Θ applied on d1 and d2, respectively).
4. Eliminate identical non-marked literals and tautology examination (tautologies are not memorized).
5. The Shortening Operation (delete all ending marked literals).
6. The Compressing Operation (delete the last non-marked literal, which is complemented in relation to negation, with some marked literal for unificator λ).
7. Repeat steps: 5 and 6 until the empty clause is got, or the Compressing Operation is not applied on the last non-marked literal.

The final result of this process is: the forming one OL-resolvent from central clause (d1) and auxiliary clause (d2).

To preserve completeness of the OL-resolution rule with marked literals, some resolvents have to be memorized.

The rule of OL-resolution with marked literals is separated in two parts: in-resolution and pre-resolution. The steps: 1 - 5 are represented in-resolution. The steps: 6 - 7 are represented pre-resolution. Mid-resolvents are the products of in-resolution and without their memorizing, the completeness of

the method can be lost. It can be illustrated by example.

Example 1.
Central clause is: $\neg R(A)$
Auxiliary clauses are:

a1 $\neg S(X,Y) \vee R(X) \vee R(Y)$
a2 $S(A,B)$
a3 $\neg R(B)$

where: X and Y are variables, A and B are constants, R and S are predicates, $\neg$ is negation.

The results without memorizing mid-resolvents are:

Taking into consideration the central clause $\neg R(A)$ and the auxiliary clause a1 by literal $R(X)$ and literal $R(Y)$, one resolvent: $/\neg R(A) \vee \neg S(A,A)$ is generated at the first level. This resolvent has not produced new resolvents and it is not possible to generate empty clause.

Not to lose the completeness of the method some resolvents must be memorized that are got during the resolution procedure.
The results with memorizing mid-resolvents are:

Three resolvents are generated at the first level:

1. $/\neg R(A) \vee \neg S(A,Y) \vee R(Y)$ from $\neg R(A)$ and a1 by literal $R(X)$
2. $/\neg R(A) \vee \neg S(X,A) \vee R(X)$ from $\neg R(A)$ and a1 by literal $R(Y)$
3. $/\neg R(A) \vee \neg S(A,A)$ from 1., or 2. with pre-resolution.

There are two resolvents generated at the second level:

4. $/\neg R(A) \vee \neg S(A,B)$ from 1. and a3 with in-resolution
5. $/\neg R(A) \vee \neg S(B,A)$ from 2. and a3 with in-resolution.

Empty clause is generated at the third level from $/\neg R(A) \vee \neg S(A,B)$ and a2. The set of clauses is contradictory.

From the point of scientific researching this example shows that some resolvents have to be memorized to preserve the completeness of the method. This modification of Ordered Linear resolution rule is served as the base for development of the system for automatic theorem proving ADT.

3. The System for Automatic Theorem Proving ADT

In our country, the first resolution theorem-prover is developed in a scope of GRAPH expert system at the Faculty of Electrical Engineering in Belgrade, ([8]).

The system ADT is based on the resolution rule. The system is developed at Technical Faculty "Mihajlo Pupin" in Zrenjanin. ADT is a system for automatic theorem proving, which is implemented on PC - computer by Pascal (Turbo Pascal ver. 6.0) programming language. The rule of Ordered Linear Resolution with marked literals presents the system base, ([6]).

ADT system differs from the other resolution-based theorem-provers which are characterized by one fixed strategy. The system permits various syntactic search strategies, ([2], [3], [5]).

The system ADT disposes three search strategies: breadth-first, depth-first and their combination. The first and the second strategy are common blind search procedures. The third blind search procedure is constructed as their combination.

In breadth-first search are the nodes starting with the root node of the search tree. They all are generated level by level. In depth-first search, a new node is generated at the next level, from the one current, and the search is continuing deeper and deeper in this way until it is forced to backtracking. In combine-search, the nodes of the search tree are generated and examined in the breadth, until the fulfilling of the level. Then the procedure is backing one level up and continues in depth with backtracking.

The system ADT permits comparisons of strategies. It is also possible to use various strategies to find the proof, especially if it can not be detected by means of other ones.

ADT is projected for scientific - researching, teaching and practical purpose. Some results of the experimental work with ADT system are described in ([3]).

There are many different possibilities for using the system in education. ADT can be used for learning the elements of theorem-proving. It allows the illustration of the Unification Algorithm or the Resolution Rule. It is also possible to use this system for experimental work in: deduction of proofs, comparison of strategies, influence of various factors on efficiency proving.

The methods of automatic theorem proving can be applied in various domains of artificial intelligence. They are applicable in fields as mathematical theorem proving, expert systems, question-answering systems, automatic programming, program verification, situational control and decision, relation data bases, logical programming, etc. It is presented in some concrete examples ([3]).

This system is incorporated in the system for automatic creating of the combinatorial disposition DEDUC ([11]), where it has presented the satisfying practical efficiency. ADT system is the basic generating mechanism in DEDUC system. DEDUC system is aimed to automated creating time-table. It is implemented on PC computer.

4. ADT system and PROLOG

Specific high-level languages have been developed for different application domains. PROLOG and LISP are the most famous programming languages in artificial intelligence.

The logical programming language PROLOG and ADT system are compared.

PROLOG is a logic-oriented language ([4], [10]), which contains a resolution-based theorem-prover. The theorem-prover in PROLOG appears with the depth-first search approach. The first-order predicate logic is the form of representation in PROLOG. Programs in PROLOG consists of axioms (clauses, facts) and a theorem to be proved (goal). The axioms are restricted in "Horn clause" form.

The first-order logic is the form of representation in ADT system, too. But, this system has not restriction in "Horn clause". It appears with clauses. The axioms are presented by auxiliary clause. The central clause is negating the theorem to be proved.

PROLOG has the negation defect. This defect is corrected in ADT system. It can be illustrated by example.

Example 2.

Program in PROLOG:

```
vegetarian(tom).
vegetarian(ivan).
vegetarian(isak).
smoker(tom).
smoker(isak).
ana_likes(X1) :  not (smoker(X1)), vegetarian(X1).
```

PROLOG-system gives unconnected answer on following questions:

```
?- ana_likes(X1).
no
?- ana_likes(ivan).
yes
```

If the last clause is now:

```
ana_likes(X1) :  vegetarian(X1), not (smoker(X1)).
```

PROLOG-system gives wrong answers on following questions:

```
?- ana_likes(X1).
X1=ivan
?- ana_likes(ivan).
yes
```

These answers are incorrect because we have not data about Ivan and smoking. We don't know is Ivan a smoker or not. The correct answer will be: "I don't know".

In both cases ADT system gives the correct answer: "I don't know". In fact, ADT system generates only one resolvent and can not complete the proof with none of the three strategies.

ADT system allows recursion using (example with family relationship, [3]) and works with structures and lists, as well as PROLOG.

5. Conclusion

Completeness and universality of the resolution method, as the base of ADT system, enables it to be applied in various domains of artificial intelligence. In the scientific researching is given an example which shows that some resolvents must be memorized to preserve the completeness of this method. The relationship between ADT system and PROLOG are emphasized. In this sense, the further development and applications of this system is possible. The system is convenient for teaching and has the practical purposes.

References

[1] Albus J.S., *Outline for a Theory of intelligence*, IEEE Transactions on Systems, Man, and Cybernetics, New York **21** (1991), no. 3, 473-509.

[2] Barr A., Cohen P.R., Feigenbaum E.A., *The Handbook of Artificial Intelligence, Vol.I,II,III*, Heuris Tech Press, W. Kaufmann, Inc., California, 1982.

[3] Berković I., *Variable searching strategies in the teaching aimed system for automatic theorem proving*, M.Sci. Thesis, Technical Faculty "M. Pupin", Zrenjanin, 1994. (Serbian)

[4] Bratko I., *Prolog programming for Artificial Intelligence*, Addison–Wesley Publ. Comp., 1986.

[5] Gevarter W.B., *Intelligent Machines*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[6] Hotomski P., Pevac I., *Mathematical and program problems of Artificial Intelligence in the field of automatic theorem proving*, Naučna knjiga, Belgrade, 1988 (1991). (Serbian)

[7] Hotomski P., *Deductive approach to automatic creating of combinatorial disposition*, Proc. of the VI Conf. on Logic and Comp. Sci. LIRA'92, Novi Sad (1992), 35–42. (Russian)

[8] Hotomski P., *Systems of Artificial Intelligence*, Technical Faculty "M. Pupin", Zrenjanin, 1994. (Serbian)

[9] Jones M., *Applications of artificial intelligence within education*, Comp.& Maths. with Appls. **11** (1985), no. 5, 517-526.

[10] Kluźniak F., Szpakowicz S., Bien J., *Prolog for programmers*, Academic Press Inc., Orlando, 1985.

[11] Madžarević T., Cvetković D., *Some supplements to the strategy for a proof directing in the theorem prover of the expert system Graph*, Proc. of SYM-OP-IS '93, Belgrade (1993), 31-34. (Serbian)

[12] Prohaska D., *A deductive method for the combinatorial disposition generating and its program algorithmisation*, M.Sci. Thesis, Technical Faculty "M. Pupin", Zrenjanin, 1993. (Serbian)

[13] Winston P. H., *Artificial Intelligence*, Addison Wesley Publsh. Comp., 1984.

University of Novi Sad, Technical Faculty "Mihajlo Pupin", 23000 Zrenjanin, Yugoslavia