Annales Univ. Sci. Budapest., Sect. Comp. 17 (1998) 321-329

THEOREM PROVING METHOD AND THE COMPUTER SCIENCE

K. Pásztor–Varga (Budapest, Hungary)

Abstract. In this paper two decision methods, the resolution principle and the method of tableaux are regarded. We make some comparison between their abilities and properties. Our aim is to give some idea about the reasons why the resolution principle and not the method of tableaux is the basic process in the logic programming.

Some idea is mentioned to regard and to illustrate the work of both methods. Based on that we suggest some modification of method of tableaux allowing its more procedural applications.

1. Introduction

It is an important problem in logic and in computer science to decide whether a formula of predicate logic is satisfiable. With other words researches for algorithms solving the decision problem nowadays are also important. It is known that there exists no general solution for the decision problem, which means that no algorithm can decide the validity (or satisfiability) of arbitrary first order formulas (Church, 1936 [1]). This result initialized the examination of the classes of first order formulas and some decidable classes of formulas were found.

On the basis of the definition of the semantical consequence the equivalent of the fact that a formula B is semantical consequence of a finite set of formulas $F = \{F_1, ..., F_n\}$ is that $F_1 \to F_2 \to ... \to F_n \to B$ is logically true (or $F_1 \wedge ... \wedge F_n \wedge \neg B$ is unsatisfiable). The solution of the decision problem by the resolution principle for Skolem formulas is well known. Using Herbrand's results the first order resolution calculus and different algorithms realizing this calculus were elaborated. Among them the linear input resolution algorithm seemed to be the most suitable for computer implementation. Its realizations represent the base

of the PROLOG type languages ([6, 7]) and the DATALOG type languages [8] being important part of the tools of data base and knowledge base management systems. In this paper the linear resolution will be regarded. One other process to solve the decision problem is the tableau method [2, 5]. Here the deduction rule is a direct application of the semantics of the first order logic and the only restriction is the finiteness of the examined formula. However the deduction rule of the tableau method is more natural as the construction of resolvent, its application in the mechanical theorem proving is not a wide-spread method. On the other side this method is applied to verify the satisfiability of a formula not only in the classical logic but in the temporal logic as well [9].

In this paper both methods will be shortly regarded and we try to find a way of a larger applicability of the method of tableaux.

2. Resolution principle and the method of tableaux

2.1. Basic concepts of the resolution principle

By the resolution method the decision problem of logic is solved through the unsatisfiability of the formula $F_1 \wedge ... \wedge F_n \wedge \neg B$. This formula must be in Skolem form $R = \forall x_1, ..., \forall x_n D$, where D is in CNF ($D = C_1 \wedge ... \wedge C_r$) and the clauses C_j ($1 \leq j \leq r$) have no common variables. Consequently R can be rewritten in form $S = \forall x_1, ..., \forall x_n C_1 \wedge ... \wedge \forall x_1, ..., \forall x_n C_r$. Now we define some main concepts used in the resolution method [3].

Definition 2.1.1. *A first order clause is a formula* $\forall x_1, ..., \forall x_n$ $(L_1 \vee ... \vee L_s)$, *where* L_i $(1 \leq i \leq s)$ *are first order literals.* In the sequel $L_1 \vee ... \vee L_s$ denotes a first order clause.

Definition 2.1.2. *A literal is an atom or the negation of an atom. A positive literal is an atom, a negative literal is the negation of an atom.*

Definition 2.1.3. *A substitution* Θ *is called a unifier for a set* $\{E_1, ..., E_n\}$ *of expressions if and only if* $E_1\Theta = ... = E_n\Theta$. *We tell that the set* $\{E_1, ..., E_n\}$ *is unifiable if there is a unifier for it.*

Definition 2.1.4. *A unifier* σ *for a set* $\{E_1, ..., E_n\}$ *of expressions is a most general unifier (mgu) if and only if for each unifier* Θ *for this set there exists a substitution* λ *such that* $\theta = \sigma \bigotimes \lambda$.

Definition 2.1.5. *If two or more similarly negated literals of a clause* C *have a most general unifier* σ, *then* $C\sigma$ *is called a factor of* C.

Definition 2.1.6. *Let C_1 and C_2 be two clauses (called parent clauses) with no common variables. Let L_1 and L_2 be two literals in C_1 and C_2 respectively. If L_1 and $\neg L_2$ have a most general unifier α, then the clause $(C_1\sigma - L_1\sigma) \vee (C_2\sigma - L_2\sigma)$ is called a binary resolvent of C_1 and C_2. The literals L_1 and L_2 are called the literals resolved upon [3].*

Definition 2.1.7. *A resolvent of (parent) clauses C_1 and C_2 is one of the following binary resolvents:*

1. *a binary resolvent of C_1 and C_2,*
2. *a binary resolvent of C_1 and a factor of C_2,*
3. *a binary resolvent of a factor of C_1 and C_2,*
4. *a binary resolvent of a factor of C_1 and a factor of C_2.*

Definition 2.1.8. *A resolution deduction of a clause C_n from the set of clauses S is a sequence of clauses $(C_1, C_2, ..., C_n)$, where either $C_i \in S$ $(1 \leq i < n)$ or C_i is resolvent of parent clauses C_s and C_t $(s, t < i)$.*

Definition 2.1.9. *A set of clauses S has resolution refutation if the empty clause ([]) has a resolution deduction from S.*

Remark 1. Let us regard the following partitioning $S = \{P \cup G\}$, where P is the set of clauses obtained from $F_1 \wedge ... \wedge F_n$ and G is the set of clauses obtained from $\neg B$. During the construction of the resolution deduction of the empty clause from the set of clauses S (generated from the Skolem formula of $F_1 \wedge ... \wedge F_n \wedge \neg B$) a substitution λ for the variables of the elements of the set of clauses S will be obtained. In principle λ substitutes the universally quantified variables of $F_1 \wedge ... \wedge F_n$ and the existentially quantified variables of B. In reality the restrictions concerning the simultaneous valuations of variables are obtained. Regarding the resolution process this means that the empty clause has a resolution deduction from $\{P \cup G\lambda\}$ or from $S\lambda$ (in the last case without any further substitution).

Remark 2. As we have seen above the clauses of the set of first order clauses S obtained from the formula D have no common variables. This fact allows that the variables appearing in the clauses can get values independently. So the unifying (Def. 2.1.3, 2.1.4) algorithm becomes simpler and as an other consequence any element of the set of clauses S is usable not only one time in the resolution deduction. As we have also seen the composition of the unifier substitutions constructed during the deduction of the empty clause is a correct answer substitution (Def. 2.1.10). In the applications (A.I., theorem proving, PROLOG like languages, knowledge base management ...) the different resolution algorithms start with the clauses of the negated theorem formula. All of the linear resolution deduction from a set of clauses attached to a fixed top clause can be represented by a deduction tree [3, 6]. If the top clause is G then the restrictions coming from the theorem appear already at the start

and then can decrease the used part of the deduction tree [6] until the firstly obtained empty clause. Naturally if we intend to obtain every appearence of the theorem the whole deduction tree is regarded.

Definition 2.1.10. [6] *A substitution λ for $\{P \cup G\}$ is called answer substitution if it substitutes the variables of G only. λ is a correct answer substitution for the set of clauses $S = \{P \cup G\}$ if the empty clause is deducible from $\{P \cup G\lambda\}$.*

The notion of λ (correct) answer substitution is treated in [6, 7] as an important element of logic programming. In PROLOG interpreters generally the linear input resolution is implemented but we regard here the linear resolution.

We can state now several presentation of the decision problem including the case of resolution principle. Let $F = \{F_1, ..., F_n\}$ be a set of formulas and B a formula. Let B be logical consequence of F. Then the following are equivalent. B is the logical consequence of F if and only if

- $\{F \cup \{\neg B\}\}$ is unsatisfiable.
- $F_1 \wedge ... \wedge F_n \wedge \neg B$ is unsatisfiable.
- The set of clauses S obtained from the Skolem formula of $F_1 \wedge ... \wedge F_n \wedge \neg B$ has resolution refutation.
- There is correct answer substitution λ for B.
- By the linear (or any complete) resolution deduction the most general answer substitution is produced for B.

2.2 Basic concepts of the method of tableaux

By the method of tableaux the decision problem of logic is solved through the unsatisfiability of a formula R. To apply the method of tableaux to a formula R the form of the formula R is indifferent. We define here some main concept used in the method of tableaux ([2, 5]).

Definition 2.2.1. *Let C be a formula. A weak subformula of C is a subformula D of C or D in negated form $(\neg D)$.*

If we regard a formula having the form $D = D_1 © D_2$, $D = \neg(D_1 © D_2)$, where ©can be any of the logical connectives $\wedge$, $\vee$, $\rightarrow$ or $D = \forall \underline{x} D_1$, $D = \exists \underline{x} D_1$ the semantics give the conditions for the truth values of the direct subformulas D_1, D_2 resulting the true (or false) truth value of D. For example regarding the following formulas $\forall \underline{x} D_1$, $\exists \underline{x} D_1$, $D_1 \rightarrow D_2$, $D_1 \wedge D_2$ and $\neg(D1 \wedge D_2)$, they are true if and only if when respectively $D_1(x/y)$ is true for any parameter, $D_1(x/a)$ is true for at least one parameter, at least one of the formulas $\neg D_1$, D_2 is true, both of the formulas D_1, D_2 are true, at least one of the formulas $\neg D_1$, $\neg D_2$

is true. We can give these semantical rules in a schematic form (Fig.1). These schemes are the construction rules of a tableau (tableau rules). Regarding the well formed formulas, there are four different type of tableau rules. The type of a formula is determined by its tableau rule (Fig.1). A rule attaches one edge or two edges to a vertex of the tableau with two or one weak direct subformulas of the elaborated formula.

The tableau rules corresponding to the formulas type α, β, γ, δ will be called respectively rule A, B, C, D.

$\frac{D_1 \rightarrow D_2}{\neg D_1 \mid D_2}$	$\frac{D_1 \wedge D_2}{D_1,\ D_2}$	$\frac{\neg(D_1 \wedge D_2)}{\neg D_1 \mid \neg D_2}$	$\frac{\forall \underline{x} D_1}{D_1(x/y)}$	$\frac{\exists \underline{x} D_1}{D_1(x/a)}$
type : β	α	β	γ	δ

Figure 1. Tableau rules and their types

Definition 2.2.2. *A direct tableau of a formula D is obtained by one application of its tableau rule.*

Definition 2.2.3. *The complete tableau of a formula D is obtained applying the tableau rules to D and to all of its descendents until the not elaborated formulas will be literals.*

Definition 2.2.4. *A branch of a tableau is closed if it contains an atomic formula A and its negation $\neg A$. A tableau is closed if it has only closed branches.*

Definition 2.2.5. *A branch of a tableau is called complete if the not elaborated formulas on the branch are literals. A tableau is completed if every branch of it is either closed or complete.*

Remark 3. The structure of a branch of tableau was investigated by K.J.J. Hintikka [4]. If a branch in the tableau of a formula D contains γ-type formulas and it is open, then the branch is infinite. There exist systematic procedures assuring that the formulas of an open branch will be arranged in one sequence. Tableaux constructed with such procedure are called systematic tableaux [5]. In a systematic tableau the open branches are Hintikka sets. The systematic tableaux will be regarded in point 3.

Remark 4. As the subformulas of a first order formula R generally contain common variables this must be taken in consideration during the construction of the tableau of the formula. Here only the tableau rule D can cause some problem. When the rule D is applied, the existentially quantified variable x is substituted by a parameter q with proviso (not knowing but supposing that

x was in the scope of other quantifiers in the original formula). That is using the rule D we can introduce a new parameter q if q has not been previously introduced by rules D or C.

Let a formula R be one of the formulas of a branch in the tableau of a formula D. Since R is a descendent of D then to guarantee that D has the value true R must be true. After Hintikka's results follows that [5]

a) in the tableaux of a logically true formula on every branch the sets of formulas are satisfiable,

b) in the tableaux of a satisfiable formula there is at least one branch where the set of formulas is satisfiable,

c) in the tableaux of an unsatifiable formula on every branch the sets of formulas are unsatisfiable.

Now we give a presentation of the decision problem in case of tableaux. Let B be logical consequence of F. Then the following are equivalent. B is the logical consequence of F if and only if

- $\{F \cup \{\neg B\}\}$ is unsatisfiable.
- $F_1 \wedge ... \wedge F_n \wedge \neg B$ is unsatisfiable.
- Any systematic tableau of the formula $F_1 \wedge ... \wedge F_n \wedge \neg B$ is closed.

3. Systematic tableaux, some idea

The informal definition of systematic tableau construction is the following [5]. First the subformulas of type α, β, δ will be elaborated on the branches of the tableau, then the elaboration of every γ-type formula follows, but together with one elaborated $\gamma(y)$ an occurence of the γ-type formula is repeated. Applying the rule C the parameter y is not restricted.

From point of view of the theorem proving the notion of the answer substitution introduced in the resolution principle was a fruitful idea. In the artificial intelligence this is a tool of obtaining not only the proof of the theorem, but show every (or some) concrete occurence of it. Now we try to find an equivalent to the answer substitution in case of the method of tableaux.

During the tableau construction process a substitution η is developed on every branch for the variables of R originally bounded. Parameters introduced by the rule C can be regarded independent to any other ones on the branch. But it is not true for a parameter q introduced by rule D (Remark 4). If a new parameter y is introduced by the rule C on a branch and at the same time a literal is obtained, then in favour of obtaining complement pairs it is

recommended to take in consideration the earlier obtained atoms on the branch and possibly use a parameter chosen earlier on this branch by any of other rules. That is the whole branch must be in scope to find the conveniable parameter y for the closure of the branch. As the different branches of the tableau are not completely disjoint a controlled tableau construction requires some strategy in choosing parameters.

Definition 3.1. *A branch of a systematic tableau is called quasi complete if the not elaborated formulas on the branch are literals or universal formulas at least once elaborated. A tableau is quasi completed if every branch of it is either closed or quasi complete.*

The tableau of R is closed if and only if R is unsatisfiable ($\neg R$ is logically true). But it is supposed that separately the formulas $\neg B$ and $F_1 \wedge ... \wedge F_n$ are satisfiable.

Let us regard a closed tableau T, where the root formula is $R = F_1 \wedge ... \wedge F_n \wedge \neg B$. In principle a branch in this tableau can be closed a) because of the interpretation represented on this branch does not satisfy F or $\neg B$, b) because of the unsatisfiability of the formula R. But in the tableau the cause of closure is not marked. Now a strategy will be shown by which it is possible to obtain that. The notion of systematic tableau supports the construction of such a strategy.

Algorithm to find the closed branches of a systematic tableau of a satisfiable formula Q.

1. Apply to Q the tableau rules A, B, D until the not elaborated formulas will be atomic or universal formulas. During this process possibly some closed branches appear.
2. Apply once the tableau rule C to every γ-type formula of the open branches.
3. Apply the tableau rules A, B, D to the resulting formulas until the not elaborated formulas will be atomic or universal formulas.
4. Execute steps 2, 3 to the newly obtained γ-type formulas until the not elaborated formulas will be atomic.
5. As the parameters of the tableau rule C are not restricted it is possible to find the potential complement pairs and then at this point every closed branch will be obtained. Mark the obtained closed branches.

Definition 3.2. *Let us construct the tableau of R using the above algorithm and starting with $\neg B$ or with $F_1 \wedge ... \wedge F_n$. The closed branches of the tableau of $\neg B$ or of $F_1 \wedge ... \wedge F_n$ are called firstly closed branches. The other closed branches of the tableau of R are called secondly closed branches.*

Remark 5. On a branch of the tableau of R a substitution η is developed during the application of the tableau rules. In case of an open branch this means that there is such a structure on a fixed universe that for some evaluation of variables (parameters) the formulas on the branch will be true, then so is R. Then an open branch gives some model of R and every model of R is obtained as the union of models given by the open branches. If the branch is closed, then the set of formulas situated on it, including R, is unsatisfiable in the structure given by this branch. If the tableau of R is closed then R is unsatisfiable and so $F \models B$. The secondly closed branches contain information about the theorem B by the substitutions of these branches.

Theorem 3.1. *Let the formula $R = F_1 \wedge ... \wedge F_n \wedge \neg B$ has a closed tableau. The substitution σ obtained on a secondly closed branch gives an answer substitution.*

Proof. Let us use the above algorithm and construct first the quasi complete tableau T_F of the formula $F = F_1 \wedge ... \wedge F_n$. The open branches of T_F contain sets of formulas $[Q_1, Q_2, ..., Q_s]$ defining the structures by the substituted atomic formulas obtained on the i-th branch [5]. Now we continue the developing of the open branches of the tableau by the elaboration of the formula $\neg B$. The tableau of R will be closed. Let σ be a substitution on a secondly closed branch. Generally an open branch of T_F with substitution η_i on it can be followed by several (closed) branches. Let the number of these branches be t_i. We associate the label (i, j), $1 \leq j \leq t_i$, to the secondly closed branches. On a branch (i, j) the substitution $\sigma_{ij} = \eta_i o \rho_{ji}$. That is the formula $\neg B$ is substituted by ρ_{ji}. So $\neg B\rho_{ji}$ is false in the model of $F\sigma_{ij}$. Then $B\sigma_{ij}$ is logical consequence of $F\sigma_{ij}$.

Consequence 3.1. *From the Theorem 3.1 follows that if the tableau has N newly closed branches and σ_j is the substitution on the j-th branch, then $B\sigma_1 \vee ... \vee B\sigma_N$ is logical consequence of F.*

Remark 6. Using the resolution calculus the answer substitutions are generated automatically by the composition of the most general unifiers developped during the resolution deduction. In case of the method of tableaux there is no such direct way. To obtain the answer substitution we introduce a strategy which seems to be good enough, but not optimal.

Strategy for generating the closure of tableau T_R and obtaining the correct answer substitutions of B. We start with the subtableau T_F of T_R.

1. If there are open branches of T_R one of the branches of the tableau T_R is developped. Let us denote every new parameter by different symbol and mark by the type of the used tableau rule (C or D).
2. When a literal L appears on the branch we examine whether $\neg L$ or a literal K (regarding the type of rules) unifiable with $\neg L$ appears on the branch. If

such K exists then we do the unification and we execute the actualization of the substitution on the branch. (This is important because this branch can have common part with other open branches.) Now this branch is closed.

3. If there exists branch of T_R to develop then we continue by the step 1 using the next branch of the tableau. Otherwise the process is finished.

References

[1] **Church A.,** A Note on the Entscheidigungsproblem, *Journal of Symbolic Logic,* **1** (1936), 40-44.

[2] **Bell J.L. and Machover M.,** *A Course in Mathematical Logic,* North Holland, 1977.

[3] **Chang C.L. and Lee R.C.T.,** *Symbolic Logic and Mechanical Theorem Proving,* Academic Press, 1973.

[4] **Hintikka K.J.J.,** Form and content in quantification theory, *Acta Philosophica Fennica,* **8** (1955), 7-55.

[5] **Smullyan R.M.,** *First-Order Logic,* Springer, 1968.

[6] **Lloyd J.W.,** *Foundations of Logic Programming,* Springer, 1984.

[7] **Apt K.R.,** Logic Programming, *Handbook of Theoretical Computer Science,* Elsvier, 1990, 495-574.

[8] **Ullmann J.D.,** *Principles of Database and Knowledge-Base Systems,* Computer Science Press, 1989.

[9] **Rácz É.,** Specifying a transaction manager using temporal logic, *Proc. of the Third Sympsium on Programming Languages and Software Tools,* 1994, 109-119.

K. Pásztor-Varga
Department of General Computer Science
Eötvös Loránd University
VIII. Múzeum krt. 6-8.
H-1088 Budapest, Hungary
pkata@ludens.elte.hu