Annales UMCS Informatica AI 3 (2005) 15-25

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Semantic tree method – historical perspective and applications

Izabela Bondecka-Krzykowska*

Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Umultowska 87, 61–614 Poznań, Poland

Abstract

The method of semantic tablaux or semantic tree is a very well known method in logic. Everyone who had ever learned logic knows basic rules and applications of this method. In the paper special attention is drawn to the history of this method as a part of the history of computers (especially of the history of mechanization of reasoning) and its applications. The method itself is presented for the classical propositional calculus and predicate calculus.

1. Short history

The history of computers can be divided into two main parts: the history of automated computing and the history of mechanization of reasonings. Roots of automated computing can be found in ancient abacus. More sophisticated form of aback were bones, developed in 1617 by Neper, which served as a device for multiplication for almost 200 years. The first machines which performed in a mechanical way arithmetical operations were built a few years later. Some of them take advantage of Neper's multiplication bones.

Most known calculating machines were: Pascalina (built by Blaise Pascal) and Leibniz's machine. But already before Leibniz and Pascal an unknown German inventor Wilhelm Schicard built in 1623 a machine which could add, subtract and multiply automatically. The machine was constructed to help the great astronomer Johannes Kepler (who was Schicard's friend) to perform astronomical calculations, but Kepler never used it. Six months after the machine was built, it was burned down in very mysterious circumstances and never reconstructed. For almost 300 years the Schicard's inventions has been forgotten and only after Kepler's death the researchers found letters form Schicard to Kepler in which Schickard informed his friend about the invention and described the construction of his machine and the method of using it.

* *E-mail address*: izab@amu.edu.pl

For many years "Pascalina" was considered the first calculating machine. It was built in 1645 by Blaise Pascal to help his father who was a tax collector. It enabled only adding and subtracting numbers. Automated multiplication of numbers enable the machine built in 1694 on the basis of planes made by G.W.Leibniz 21 year earlier.

G. W. Leibniz was not only an inventor but first of all a philosopher and a logician. He was the author of the idea of a universal language of science and of a universal method of calculation which should enable scientists to provide scientific reasonings in a strictly mechanical and algorithmic way.

The very idea of mechanization of reasonings was developed and used much earlier by Raymond Lullus. In 1275 he created the first machine which helped to perform some kinds of reasonings. The "machine" consisted of three concentric discs, triangles and squares made of paper. On the boundary of discs Lullus placed the names of God and his attributes. User could "calculate" attributes of God and provide reasonings about Goodness by moving discs in the special way. Lullus used this „machine" in his numerous missions in the region of the Mediterranean Sea. Lullu's machine, called *Ars combinatorie* was the first machine which enabled algorithmic and mechanical reasonings.

The Lullus' and Schicard's ideas were combined in works by Charles Stanhope who built "demonstrator" – a machine which helped solving classical syllogisms. In this way the first connection between computing machines and logic was established.

In 19th and at the beginning of 20th century a considerable progress in the counting machines took place. At that time first programmable machines appeared. Charles Babbage designed a difference engine and an analytical machine. It was also the time of the rise and of the considerable development of mathematical logic.

In 1854 George Boole published a book in which he gave a detailed analysis of laws of reasoning. In the forties of the 20th century ideas of Boole (especially his binary system) found applications in conceptions of electromechanical calculators with external programming, and later electronic computers with internal programming. It was a marriage of electronics and human thoughts, which effected in modern computers.

Since Boole's time logic was engaged in the history of calculating machines. Works of great logicians such as Peano, Frege, Russell, Skolem or Hilbert supplied methods of automatization of deductive methods. One of the methods of automatic deduction was sequent calculus created by Gerhard Gentzen. From it originated the method of analytic tableaux and semantic tree.[1]

[1]Many historical and methodological remarks on the development of mechanization and automatization of reasonings can be found in [1].

Semantic tree method originates both from the method of semantic tableaux of Beth and from Hintikka's method of model sets. Evert W. Beth in his work [2] introduced two kinds of tableaux: analytic tableaux and (much easier) deductive tableaux. His method was very complicated, so many attempts of improving it were undertaken. Similar ideas exploited Jaakko Hintika, who presented his method of model sets in 1955. In his system the proof of a formula consists of an attempt to find a counterexample for it. Many systems similar to the system of Beth or Hintika were created, but the beginning of *semantic tree method* are usually related with names of Hintikka and Smullyan and it is not clear who in fact was the father of this method.

Annelis in his article [3] reports that Raymond Smullyan began his work on the method called by him the method of analytic tableaux around 1959. His main work on this subject was a handbook "First order logic (1968) where the method was presented in a very systematic way. Smullyan's tableaux are in fact very similar to model sets by Hintikka, but Smullyan claimed that he did not know Hinttikka's works and even did not hear about such person. Whereas Hintikka claimed that Smullyan invited him to give a talk on model sets at Yeshiva University in New York in 1962 (hence before the publication of "First order logic"), but Smullyan did not remember such visit. Some time later Hintikka was not so sure about the place where he gave his talk and about the person who introduced him. He remembered only that it was in New York and that some mathematician invited him. Because of these misunderstandings it is very difficult to judge who is in fact a pioneer of the method.

It is not commonly known that in the year 1960 in Studia Logica a paper by Zbigniew Lis "*Wynikanie semantyczne a wynikanie formalne*" was published (it was submitted on May 3rd, 1959). In this paper, written in polish, Lis introduced a method of trees, which is very similar to the method of Smullyan. So the name of Zbigniew Lis should be added to the list of "potential fathers" of the method of semantic tree beside Smullyan and Hintikka.

There are many handbooks which present the considered method. The most popular are: the mentioned above book *First order logic* by Smullyan, Jeffrey's *Formal Logic: Its Scope and Limits*, Kleene's *Mathematical Logic*, Bell's and Machover's *A Course in Mathematical Logic* and Hodges' *Logic*. In the Polish literature this method is used in works by Małgorzata Porębska, Wojciech Suchoń and Witold Marciszewski.

Below presenting the method of semantic tree we use an improved version of Smullyan's notion (it is very handy in teaching logic[2]).

[2]Such notation is also used in [4].

2. Semantic tree method

The semantic tree method is an automatic method of semantical analysis, which consists of determinating the logical values of subformulas of the given formula. It reduces to the elimination of logical constants (truth-functional connectives and quantifiers) by placing subformulas of considered formula on branches of a binary tree.

The method has many applications in logic. It is used as a method of verification whether a given formula is a tautology, as a method of proving semantical consistency of a set of formulas and even as an algorithm of verification of the validity of arguments.

The essence of the method consists of finding counterexamples. For example, if we want to show that a given formula A is a tautology, we are looking for such interpretations in which this formula is false. Hence we are building a semantic tree for $\neg A$. If we obtain a contradiction in all possible cases, we can conclude that there is no interpretation in which A is false, hence A is a tautology. The existence of such contradictions is expressed in the language of trees by the notion of closed branches.

A branch of a tree is said to be closed if on this branch there is a formula A and its negation $\neg A$. In the opposite case we call a branch open. A tree, whose all branches are closed, is called a closed tree.

Closed branch will be marked by × with numbers of formulas, which caused the closing of this branch. Open branch are ended by °.

If all branches of a semantic tree of a formula A are closed then it means that there is no such interpretation in which A were true. An open branch of a semantic tree of formula A corresponds to an interpretation in which the formula A is true.

We can build trees not only for single formulas but also for finite sets of formulas. Using semantic trees we can check if a set of formulas $\{A_1, A_2, ..., A_n\}$ is semantically consistent. If there is an interpretation in which all formulas from this set are true, then the set is semantically inconsistent. So, if all branches of a tree for a set of formulas are closed, then this set is semantically consistent.

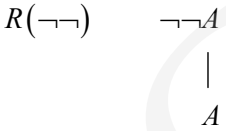
In the case of checking the validity of an argument it suffices to check if a set of formulas $\{A_1, A_2, ..., A_n, \neg B\}$ is consistent, where $A_1, A_2, ..., A_n$ are schemes of premises and B is a scheme of the conclusion. If all branches of the tree for such a set are closed then the set $\{A_1, A_2, ..., A_n\}$ of statement forms entails a statement B.

Consider now rules of building trees for formulas in propositional calculus.

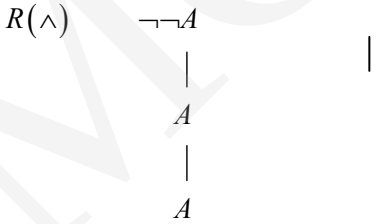
2.1. Semantic tree method for propositional calculus

Rules of building semantic trees are based on some simple properties of truth-functional connectives. Let us consider truth values of formulas depending on truth values of their subformulas.

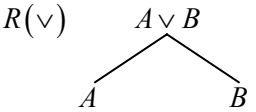
– Truth value of a formula A is the same as truth value of the formula $\neg\neg A$, so we can put formula A on the branch where there is formula $\neg\neg A$; we denote it by:



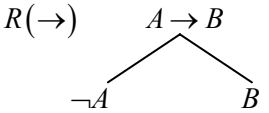
– Conjunction is true if both of its factors are true, so we place both factors an the same branch:



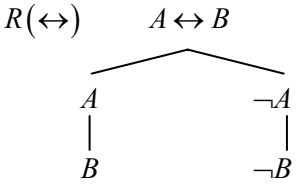
– Alternative is true if at least one of its components is true, so a tree branches off:



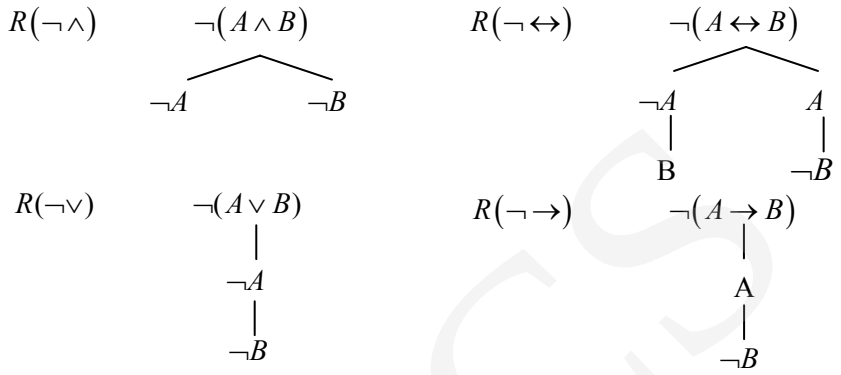
– Implication is true if the predecessor is false or the successor is true:



– Equivalence is true in two cases: if both of its factors are true or both are false:



Considering cases in which conjunction, alternative, implication and equivalence are false we obtain the following rules for negation of formulas:



Example 1. Determine whether the following statement form is valid:

$$\frac{\begin{array}{l}(q\rightarrow p)\rightarrow r\\ \neg r\wedge\neg s\\ (p\rightarrow q)\vee t\\ t\rightarrow(r\vee q)\end{array}}{t}$$

As was mentioned above it suffices to check if a set consisting of the premises and the negation of conclusion is consistent. For this purpose we built a tree beginning with a set of formulas:

$$\{(q\rightarrow p)\rightarrow r,\neg r\wedge\neg s,(p\rightarrow q)\vee t,t\rightarrow(r\vee q),\neg t\}.$$

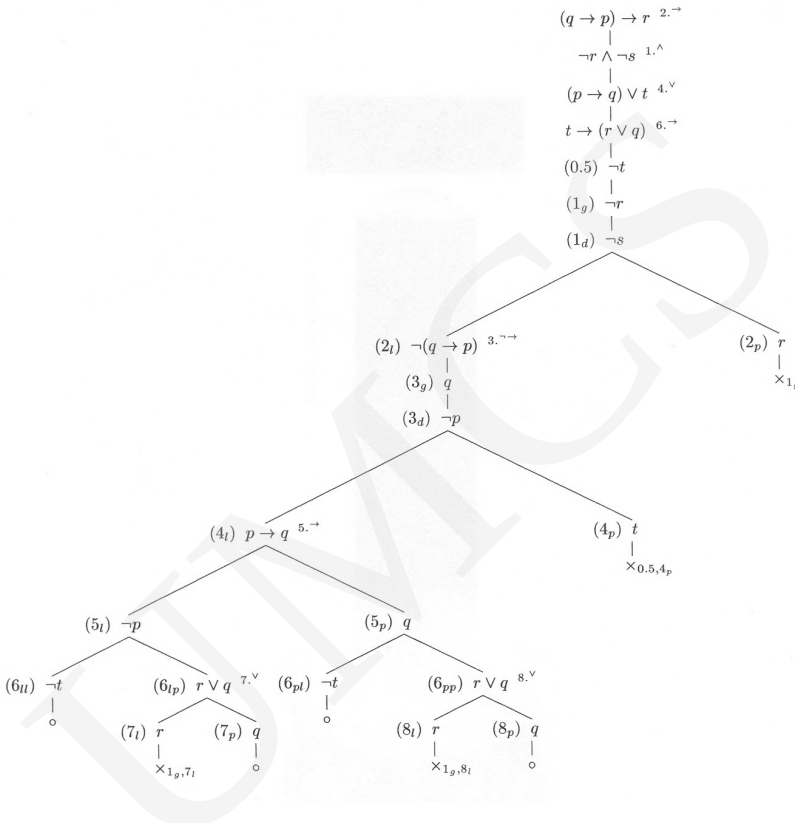
Notation: Numbers on the right side of formulas (with symbols for used rules) indicate an order of the decomposition of formulas. Numbers in brackets (on the left side of formulas) denote in which step this formula was obtained.

We can decompose formulas in any order but it is convenient to decompose first those formulas, which do not cause branching off a tree, because any operation on formula have to be done on all branches beside this formula. Consider a tree.

As first was used the rule for conjunction, because it did not cause a branching off a tree (in the first step formulas (1g) and (1d) were obtained). In the second step first premise was decomposed and on the right branch of a tree a contradiction was obtained, so this branch was closed. In a similar way all formulas on the tree were decomposed. As a result a tree was obtained with four open branches. For all of them one can find a valuation for which all premises are true and a conclusion is false:

p	q	r	s	t
0	1	0	0	0

Consequently the considered argument form is invalid.



2.2. Semantic tree method for predicate calculus with identity

In order to use semantic tree method for predicate calculus a set of rules must be enriched by appropriate rules for eliminating quantifiers:

– If a formula $\exists x A(x)$ is true, then every sentence $A(a)$, where a denotes some object from the domain of interpretation, is also true

$$R(\exists): \frac{\exists x\, A(x)}{A(a/x)}$$

for a new constant a, which does not occur in the branch.

– If a sentence $\forall x A(x)$ is true in a given interpretation, then every sentence $A(a)$ is also true for all constants a occurring on the considered branch:

$$R(\forall): \frac{\forall x\, A(x)}{A(a/x)}$$

for every constant a occurring on the considered branch.

– If a formula $\exists x A(x)$ is false, then every sentence $A(a)$, where a denotes any objects from the domain of interpretation are also false:

$$R(\neg\exists): \frac{\neg\exists x\, A(x)}{\neg A(a/x)}$$

for every constant a occurring on the considered branch.

– If a sentence $\forall x A(x)$ is false in a given interpretation, then there is a constant a for which a sentence $A(a)$ is false:

$$R(\neg\forall): \frac{\neg\forall x\, A(x)}{\neg A(a/x)}$$

for a new constant a which does not occur in the branch.

If we want to consider predicate calculus with identity we must add two new rules of building trees and a new rule of closing branches:

– If on the given branch there are: formula A including some occurrences of a term t_1 and an atomic formula $t_1 = t_2$ (where t_2 is any term), then we can place on such branch a formula $A(t_2 // t_1)$ obtained from A by replacing some occurrence of the term t_1 by the term t_2:

$$R_{12}(=): \frac{\overset{A}{t_1 = t_2}}{A(t_1/t_2)}$$

– If on the given branch there are: formula A including some occurrences of a term t_1 and an atomic formula $t_2 = t_1$ (where t_2 is any term), then we can place on such branch a formula $A(t_2 // t_1)$ obtained from A by replacing some occurrence of the term t_1 by the term t_2:

$$R_{21}(=): \frac{\overset{A}{t_2 = t_1}}{A(t_2/t_1)}$$

– To the rules of closing branches we add a new rule: a branch is closed if there is on it a formula $\neg(t = t)$ for any term t.

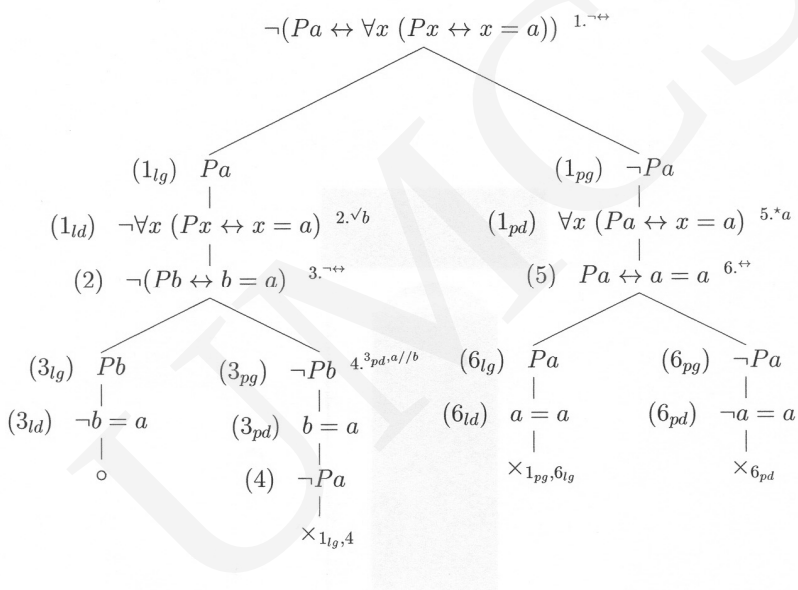
In building semantic trees in the predicate calculus the existential quantifiers are eliminated before general ones. If an existential quantifier on a given branch was eliminated by introducing a new constant, we have to develop all general formulas using this new constant. If there is no existential formula on a branch we can eliminate a general quantifier by introducing a new constant. In examples given below an elimination of an existential quantifier is denoted by $\surd a$ and the developing formula using a constant a is marked as $^{\star}a$.

Notice that semantic trees in prepositional calculus were always finite. In predicate calculus trees can be infinite. If a formula A is a tautology (i.e. there is

no such interpretation in which this formula is false) a semantic tree for A is finite, but if A is not a tautology then its tree can be infinite.

Example 2. Determine whether the statement $Pa \leftrightarrow \forall x\left(Px \leftrightarrow x = a\right)$ is a tautology of the predicate calculus.

We built a tree for the negation of this formula in order to find out if there is an interpretation in which the considered formula is false.

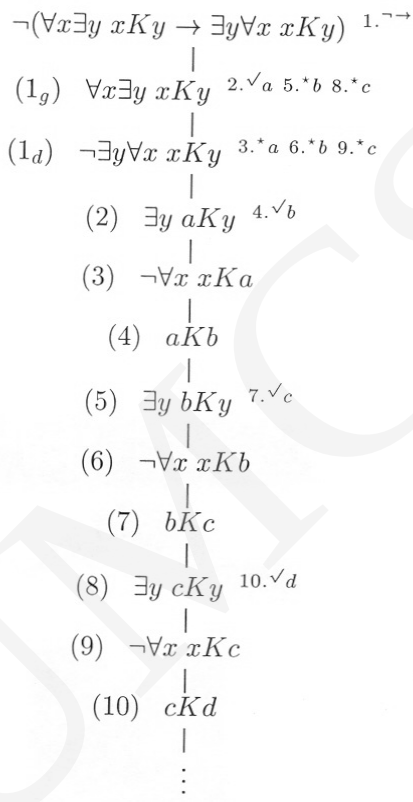


The above tree has one open branch which corresponds to an interpretation in which the formula is false. Hence, it is not a tautology[3].

Example 3. Consider formula: $\forall x\exists y\left(xKy\right) \rightarrow \exists y\forall x\left(xKy\right)$. Determine whether this formula is a tautology.

The tree is infinite; the process of building it cannot be finished, because as a result of the application of the rule $R\left(\neg\rightarrow\right)$ to the considered formula, formula (1g) was obtained. This formula has to be developed (steps 5 and 8) with respect to the constants a introduced earlier (step 4) . In such a way a new existential formula was obtained and elimination of such formula requires introduction a new constant b, etc… This procedure will never stop.

[3]Notice that a branch including formula (6pd) was closed according to a new rule of closing branches. Moreover in the third step the rule for replacing terms in formulas was used.



Therefore considered formula is not a tautology of the predicate calculus. But, observing some regularities in the above tree enables us to find an infinite model for this formula:

1) the domain of interpretation is a set $\{a_1, a_2, a_3, ...\}$,
2) relation *K* is defined on this set as follows:

$$a_i K a_{i+1} \quad \text{for} \quad i > 1.$$

3. Applications

As it has been shown above the semantic tree method has a wide spectrum of applications in logic. Semantic trees for formulas or for sets of formulas can provide additional information concerning considered formulas, for example information on interpretations in which the formula is true or false.

The applications of the semantic tree method (and similar methods)can be divided into two main groups.

The most important application of the semantic tree method is the automated theorem proving. There are many algorithms of automated proving based on

tableaux method. There exists also many handbooks and monographs in which they are presented, e.g. Melvin Fitting's *First-Order Logic and Automated Theorem Proving*. In this book author presents also implementation of the considered method in Prolog.

The second group of applications of the tableaux method is connected with logic of programs. There are many attempts to apply semantic tree method to automatic verification of correctness of protocols and their compatibility with specification.

A lot of information about applications of semantic tree methods can be found in Internet. In fact hundreds of web sites dealing with this subject can be found. Moreover, annually for more that 10 years, an international conference on applications of tableaux methods is being organized. Add that applications of the considered method are connected not only with computer science, but also with biology and chemistry.

4. Conclusions

The semantic tree method is a very interesting and universal method. It forms an important part of the history of mechanization of reasonings. It is also a method, which enables teaching logic in a very nice and easy way. In fact every student, especially a student of the computer science, knows a notion of a tree and its properties, so it is much easier to teach logic using trees than using algebraic notions. Moreover the tableaux methods are very important because of their numerous applications, not only in computer science.

This research was suported by Grant No. 1 H01A 041 27 from State Committee for Scientific Research in Poland.

The author would like to thank Professor Jerzy Pogonowski (Institute of Linguistic, Adam Mickiewicz Uniwersity, Poznań, Poland) for inspiration and helpful discussions.

References

[1] Marciszewski W., Murawski R., *Mechanization of Reasoning in a Historical Perspective*, Rodopi, Amsterdam-Atlanta, (1995).

[2] Beth E.W., *Semantic Entailment and Formal Derivability*. Mededelingen der Koninklijke Nederlandse Akademie van wetenschapen, afd. letterkunde, new series, Amsterdam, 18(13) (1995).

[3] Annelis I.A., *From Semantic Tableaux to Smullyan Trees: A History of the Development of the Falsifiability Tree Method*, Modern Logic, 36.

[4] Pogonowski J., Bondecka-Krzykowska I., to appear, *Metoda drzew semantycznych w klasycznym rachunku logicznym*.