*International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*
**Web Site: www.ijettcs.org Email: editor@ijettcs.org**
**Volume 4, Issue 1, January-February 2015** **ISSN 2278-6856**

# Theorem proving by incrementally extending theorem domain

**Ismail A. Ismail[1] and Mohamed G. Awad[2]**

[1]Faculty of information systems and computer science, 6th October University, 6th October city, Egypt.

[2]Faculty of Education, Suez Canal University Al-Arish city, Egypt.

## Abstract

***Automated Theorem provers are used to check if the mathematical theorems are proved true or false. If the theorem is true, the automated theorem prover, usually, prints out the proof. In this paper we present a new approach to deal with the other case, i.e. if the theorem is false. The semantic tableau method is used to extend the theorem domain. The modified automated theorem prover suggests an extension of the domain, if it is available, to present a new provable theorem.***

**Keywords:** Artificial intelligence, Automated theorem provers, Tableaux, PROLOG

## 1. INTRODUCTION

Automated theorem proving (ATP) is the automatic proving of mathematical theorems by computer programs [1]. SPASS, PTTP, ACL2, Vampire, Otter, HOL, and others are examples of already existing ATP programs [2]. These ATPs use different techniques to achieve the target results. Resolution and tableau are the most popular techniques used to implement theorem provers [3]. ATPs that use resolution convert logical programs first to the Conjunctive Normal Form (CNF). The resolution technique depends on choosing two clauses one of them contains a predicate and the other contains the negation of that predicate [3], [4]. The resulting new clause is formed by taking the disjunction of the two clauses after removing this predicate and its negation from the two clauses. The process is repeated until an empty set is formed or no other clauses can be resolved. If an empty set is formed then the program is proved true; otherwise it is false. A mathematical theory is proved by initially negating the goal to be proved and adding it to the premises of the program. If an empty set is driven after applying the resolution one or more times. Then, a contradiction is driven, and so the original goal is proved; otherwise, the goal is not provable using the current premises [5]. On the other hand, the main idea behind the tableau technique is to construct a truth tree by using a logical program in Disjunctive Normal Form (DNF) as the parent node. The tree is expanded by constructing a new branch for each disjunction and a new node in the same branch for each conjunction. A branch is closed if the branch contains a predicate and its negation or if it contains a false predicate [5]-[7]. To prove a theorem, the negation of the goal formula is negated and added to the logical program. If all branches of the tableau are closed, then the original goal is proved true. Otherwise, if at least one branch is not closed, then the original goal cannot be proved true. In this paper we extend the work presented in [8], our goal is to implement an ATP that can either prove a theorem if it is provable or present a modification if applied to this theorem it becomes true. The technique used is to incrementally extending the domain of the theorem by adding the missing predicates in the tableau proof. Of course, the new presented theorem is not the same like the original, but the ATP presents a new theorem with a new domain. Correcting incorrect plans, incorrect mathematical theorems, and inconsistent logic gates is some applications of this concept.

## 2. The tableau proving procedure

In tableau to prove a formula $X$ we begin with $\neg X$ and produce a contradiction [3]-[8]. A tableau proof is a binary tree labeled with formulas, where the tree itself represents the disjunction of its branches, and each branch is the conjunction of formulas appearing on it [9]-[11]. A branch of the tableau tree is closed if a contradiction or false appears on it and the tableau tree is closed if all branches are closed. LeanTap, $T^AP$, Bluegum, Cassandra, and Deep Though are examples of theorem provers that use tableau proving procedure [12]. A PROLOG implementation of a tableau theorem was introduced in [5], a PROLOG implementation of Connection Tableau is presented in [6], and we presented an implementation of a tableau- based theorem prover in [8]. The tableau-based theorem provers check the branches one-by-one, if all branches are closed then the whole tree is closed, otherwise if one of the branches is open then the tableau does not check the rest of branches, and the whole tree is open. A closed tree means that the negated formula in the parent node is a contradiction, and then the original formula is true. On the other side, if the tree is open, then the negated formula is true, and so the original formula is a contradiction. The idea of our work depends on checking the whole tableau tree in both cases, and keeps the open branches, if any. In [8], a tableau-based theorem prover with some improvements is presented. One of these improvements is to split the branch list into two lists, the negative list and the positive list respectively. The branch is closed if a predicate appears in both positive and negative lists. So, if the tableau has an open branch, it can be converted into closed one if a one of the predicates, or all, that appears in the positive list is added to the negative list or one of the predicates, or all, that appears in the

negative list is added to the positive list. We can keep track of all these predicates that appear in the positive/negative list of the open branch but does not appear in the negative/positive list. After checking all open branches of the tableau tree, we will have a list of all predicates that make branches open. In the original Tableau, the formula X is a theorem if the formula ¬X in the parent node of the tableau causes the tableau tree to be closed, i.e., all the branches of the tableau is closed. From the approach presented in [8] point of view, a tableau is closed if the list O is empty, where O is the open branches list. If the list O is not empty, then the original formula X is not a theorem, and the list O will contain the predicates that make the tableau open and more work is needed to make this formula provable, i.e. we need to add disjunction of these predicates to the original formula X to make X provable. In Prolog, the predicates flatten/2 and list_to_set/2 will output a new list RO from the list O without redundancy. Of course, we can simply add the conjunction of all of all predicates in RO to the formula, at the tableau's parent node, ¬X, or the disjunction of the negation of these predicates to the original formula X. this way is both simple and trivial, amore convenient method is needed to add the minimum number of predicates to the original theorem. If the predicates is added one-by-one and in every time the tableau is checked if it is closed or not, does not guarantee that the set of added predicates are the minimum set. A more intelligent and simple way to add the disjunction of the minimum number of predicates to the original theorem is to incrementally extending the domain of the theorem step by step using the power set of the negation of the open list O. If the original formula is X, Y= neg X, O is the open branches of the tableau, and RO is the list of predicates that makes branches open, i.e. the contradiction list. Let NRO is the list that contains the negative of predicates in RO list and let PO is the power set of NRO. Adding any predicates to the closed branches will not change the state of the branch, it is already closed, but adding predicates to the open branch can change the state of the branch from open to close. To reduce the work, if the proposed ATP is attempting to correct a theorem by closing the open branches, it will not add any predicates to the closed branches in the correction stage and it will work only with open ones. The ATP will add the smallest set in PO to all open branches in the tableau, i.e. to all branches in the list O, and it will check if these branches become closed. If the branches in O are closed then the whole tableau tree is closed. If the branches in O are not closed then the ATP will take the next smallest set in PO and repeat the previous work. If PO is empty, then the ATP will not be able to extend the improvable theorem to a provable one. For example, the theorem P and ¬P is improvable and the proposed ATP could not extend it to a provable one, since the. On the other hand, If the branches in O become closed then the open tableau can be extended to a closed one using the current smallest set from list PO.

The following pseudo code shows how the proposed ATP works.

Let X is the original formula
Y= neg X
Construct the tableau tree of Y and save it in Z
If Z is closed then
print(' X is a Theorem')
Else
Store all open branches in O
Store all unrepeated predicates in O in RO
NRO= neg RO
PO is the power set of NRO.
Repeat
H= smallest set in PO
Add H to all branches of O
PO=PO-H
check if O is closed.
Until O is closed or PO is empty or all branches of O are closed
if PO is empty and O is open then
print('can't extend X to a provable theorem')
End
if O is closed then
print('X can be extended to a provable theorem using')
print(H)
End
End..

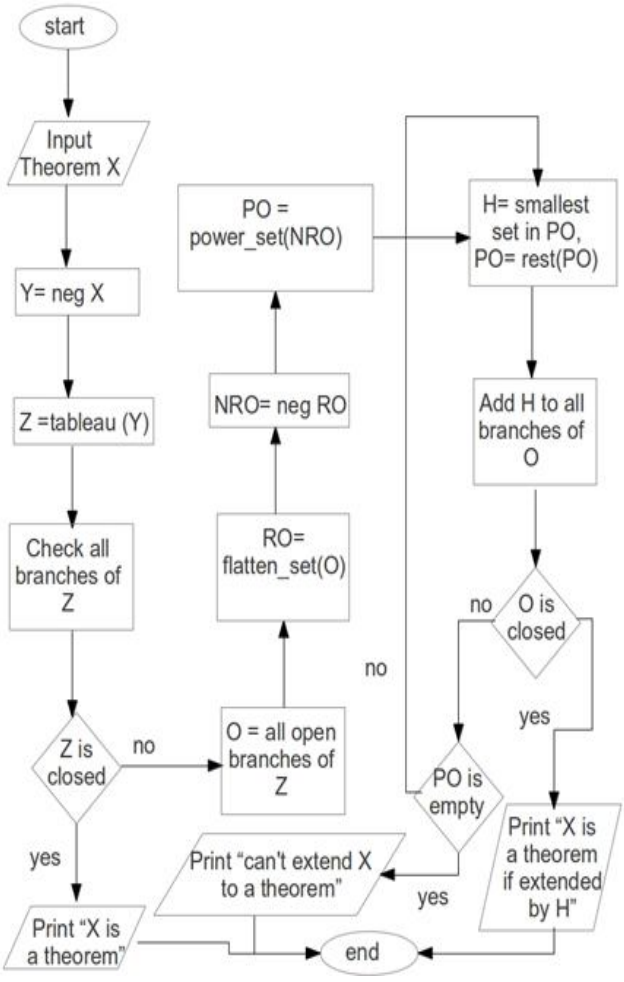



**Figure 1** a flow chart of the proposed ATP

## 3. Conclusion

In this paper we presented a new approach to deduce provable theorems from improvable theorems, by incrementally extending the domain of the theorem and adding the predicates and check if the theorem is provable or not. Some enhancements are added to guarantee that the added conjunction of predicates is minimum. A pseudo code and a flow chart of the proposed approach are presented and discussed.

## References

**[1]** Castelló,R. , Mili, R., (1998). Theorem Provers Survey. The University of Texas at Dallas.(http://www.cs.utexas.edu/~ragerdl/cs378/reading/Castello_-_Theorem_Provers_Survey.pdf)

**[2]** http://www.cs.miami.edu/~tptp/OverviewOfATP.html

**[3]** Hein, J. L. (1996). Theory of Computation: An Introduction. Sudbury, MA: Jones & Bartlett.

**[4]** http://www.cis.cau.edu/prolog/docs/Prolog_Lab_Manual.pdf

**[5]** Fitting, M. (1996). First-Order Logic and Automated Theorem Proving. Graduate Texts in Computer Science. Springer-Verlag, 2nd edition.

**[6]** Otten, J. and Bibel, W. (2003). leanCoP: Lean Connection-Based Theorem Proving. Journal of Symbolic Computation , 36(1-2):139–161.

**[7]** Agostino, M. , Gabbay, D., Haehnle, R., Posegga, J. (Eds) , Handbook of Tableau Methods, Kluwer.

**[8]** Ismail, I., Awad, M. , Rashed E., (2014) An implementation of a tableau-based theorem prover with some improvements. The AIUB Journal of Science and Engineering (AJSE), Vol. 13(1) 93:98.

**[9]** Sutcliffe, G., Suda, M., Teyssandier, A., Dellis, N. and Melo, G.,(2010). Progress Towards Effective Automated Reasoning with World Knowledge. In Hans W. Guesgen and R. Charles Murray, editors, FLAIRS Conference. AAAI Press.

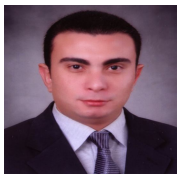
**[10]** Harrison, M. D., Masci, P. and Campos, J. C. Curzon, P. (2013). Automated theorem proving for the systematic analysis of interactive systems. Proceedings of the 5th International Workshop on Formal Methods for Interactive Systems, City University,London, June 24th, 2013.

**[11]** Folkler, A. L. E. (2002).Automated Theorem Proving Resolution vs. Tableau (master's Thesis). Blekinge Institute of Technology.

**[12]** Schumann, J. (1994). Tableaux-based Theorem Provers: Systems and Implementations. Journal of Automated Reasoning, 13(3):409-421.

## AUTHORS

**Prof. Ismail A.** Ismail is the dean of Faculty of Computers and Informatics, 6Th of October University, Egypt. He was born on March 7, 1946. He received the B.S. degree in pure mathematics / physics from Cairo University, Egypt in 1967, the M.Sc. Degree and the Ph.D. degree in Signal Processing from the Cairo University, Egypt in 1971 and 1976, respectively. He has more than 150 published papers and he is the founder of Faculty of Computers and Informatics, Zagazig University in 1997.



**Mohamed G. Awad ,** is with the Department of Mathematics, Faculty of Education, Suez Canal University, Al-Arish, Egypt. He received the B.Sc. Degree in Scientific Computations From Suez Canal University in 1998, the M.Sc. degree in Computer Science from Suez Canal University in 2005.