ISSN 2395-8618

An Implementation of Propositional Logic Resolution Applying a Novel Specific Algebra

Eduardo Zurek, Mayra Zurbaran, Margarita Gamarra, and Pedro Wightman

Abstract—**This paper presents a methodology for evaluating propositional logic satisfiability using resolution-refutation. The method applies a strategy based on an algebra developed by the authors that estimates the possible outcomes of the expression and generates a logic value for refuting or accepting the satisfiability of the argument.**

Index Terms—**Algebraic logic, propositional logic, resolution-refutation, tableau.**

I. Introduction

ARTIFICIAL intelligence's main objective has been the construction of systems capable of replicating aspects that can be categorized as intelligent. Formally, "… is the set of techniques, methods, tools and methodologies that help us build systems that behave in a similar way to a human in the concrete problem resolution". To achieve this, it is used logic among other resources, logic can be understood as a combination of language and semantics, to represent the knowledge related with the ability to perform certain reasoning [1].

To give a better understanding to logic, the algebraic logic is introduced, it can be described in general terms as "the discipline that studies logics by associating with them classes of algebras, classes of logical matrices and other algebra related mathematical structures that relates the properties of logics with algebraic properties of the associated algebra" [2].

In this paper will be used the propositional logic or zeroth order logic; it is the simplest logic language, based in a numerable (not necessarily finite) of atomic propositions (AP). A proposition is an expression in natural language that can only be false (F) or true (T). A common form of representing all the evaluations is through the truth table; which is a complete enumeration of the value of the formula for all models. However, for evaluating larger expressions, the truth table becomes infeasible since the complexity is of $O(2^n)$ for a formula containing n propositional variables, therefore, the tableau method was later introduced.

Manuscript received on February 25, 2015, accepted for publication on June 23, 2015, published on October 15, 2015.

E. Zurek is with the Research in Robotics and Intelligent Systems group at the Universidad del Norte, Barranquilla, Colombia (e-mail: ezurek@uninorte. edu.co).

M. Zurbaran is with the GReCIS group at Universidad del Norte, Barranquilla, Colombia (e-mail: mzurbaran@uninorte.edu.co).

M. R. Gamarra Gamarra is with the IET group, Department of Electronic and Telecommunication Engineering, Universidad Autónoma del Caribe, Barranquilla, Colombia (e-mail: margarita.gamarra@edu.co).

P. Wightman is with the GReCIS group at the Universidad del Norte, Barranquilla, Colombia (e-mail: pwightman@uninorte.edu.co).

The Tableau method is a formal proof procedure existing for several logics. It is based in a refutation or indirect proof procedure; showing a formula X is valid by intending to assert it is not with a syntactical expression. The expression is broken down syntactically by splitting it in several cases; this may be referred as the tableau expansion stage, it can be thought of as a generalization of disjunctive normal form expansion [3]. Finally, there are rules for closing cases; this is deriving impossibility conditions or contradictions based on syntax. If each case closes, the tableau can be considered closed. A closed tableau is proof of the validity of the expression X it began with.

With the above in mind, it was developed an algebra of our own by enlarging the domain of the conventional Boolean logic of true, false, and indefinite.

The paper is structured as follows. Section II gives a general background on the classic Tableau methodology and related implementations. Section III explains the methodology introduced by the authors. Section IV explains the algorithm and its implementation and mentions the software used for comparing the results. Section V presents the results and explains how the comparison was evaluated. Finally, Section VI presents the conclusions.

II. Background

The Tableau methodology was introduced in the 1950's by Beth and Hintikka and was later perfected by Smullyan and Fitting. It brings together the proof-theoretical and the semantical approaches to present a logical system, which is also very intuitive and has been broadly used according to [3]; the usages include implementation for circumscriptive reasoning as in [4], for interval temporal logic as seen in [5], for solving satisfiability check problems on Boolean circuits [6].

For the means of this article it is important to define propositional satisfiability and how it differs from propositional validity. Propositional satisfiability determines if there exists an interpretation that satisfies a given Boolean formula. This is, in words of Vardi [7] "The Boolean Satisfiability Problem (SAT, for short) asks whether a given Boolean formula, with Boolean gates such as AND and NOT,

ISSN 2395-8618

has some assignment of 0s and 1s to its input variables such that the formula yields the value 1". This problem is said to be NP-Complete, proved by Steven Cook in 1971. In [8], [9] validity is defined as an argument is logically valid if and only if its conclusion is a logical consequence of its premises. If an argument whose conclusion is β and whose only premise is α is logically valid, then α is said to logically imply β. Other pertinent definitions are tautologies and contradictions; a well-formed formula is a tautology if and only if it is true for all possible truth-value assignments to the statement letters making it up. On the other hand, a well-formed formula is a self-contradiction if and only if it is false for all possible truth-value assignments to the statement letters making it up.

III. Methodology

The objective of this algorithm is to evaluate propositional logic satisfiability using a refutation proof and a novel algebra defined by the authors, to proof the satisfiability of an expression X by deducing that ¬X is not possible. Tableau proofs are originally presented as trees, whose nodes are formulas that are subcases and the tree structure gives the logical dependence between them [10], due to the structure the method uses, it is not trivial to implement in a programming language, requiring data structures advanced knowledge. With this in mind a simpler approach was conceived, which does not require graph structures and little processing. The presented method EZLogicUN is based on a general algebraic logic which considers not only the two Boolean states of *true* and *false*, but includes the following two expressions:

$$q \lor \neg q \quad (1)$$

$$q \land \neg q \quad (2)$$

From this point and by the means of simplifying the codification of the proposed method, the notation used replaces the operators: ∨ for $+$, ∧ for *, → for > and ¬ for $-$. In this way, the mentioned expressions are shown in Table I.

TABLE I.
NOTATION

Expression	Abbreviation
Indefinite	0
q	1
$-q$	–1
$q + -q$	2
$q * -q$	3

The functions used are: not (–), and (*) and or (+). The domain is limited to the expressions above and will be denominated by the numbers 0, 1, –1, 2, 3 accordingly. Table II shows the algebraic operations for an expression X and its negation.

TABLE II.
NOT OPERATOR

X	–X
–1	1
0	0
1	–1
2	3
3	2

The resulting expressions with the NOT operator can be inferred by propositional logic rules of negation. 2 and 3 specifically, are a case of the De Morgan axiom [11]:

$$-(q \pm q) = -q * - - q = -q * q = q * -q \\ \Rightarrow -2 = 3 \quad (3)$$

$$-(q + -q) = -q * - - q = -q * q = q * -q \\ \Rightarrow -2 = 3 \quad (4)$$

Table III presents the results of operating the expression X AND Y. Similarly to the NOT operator table, the results are deduced by applying simple propositional logic rules. The resulting inference that prevails is the one that avoids a contradiction if possible.

TABLE III.
AND OPERATOR

X	Y	X * Y
–1	–1	–1
–1	0	–1
–1	1	3
–1	2	3
–1	3	3
0	0	0
0	1	1
0	2	2
0	3	3
1	1	1
1	2	3
1	3	3
2	2	2
2	3	3
3	3	3

In the same way is calculated the operations for the OR function in Table IV. The OR function gives the possibility to select the case in which no contradictions (resulting in 3) arises if possible. This is particularly useful while analysing all the possibilities for the values of the expression to be a propositional satisfiability.

ISSN 2395-8618

IV. Implementation

For the means of comparing the outcomes of our implementation, EZLogicUN, there will be evaluated the same logic arguments on both; our algorithm which was programmed in a MATLAB [12] script and the LoTREC Tableau Prover [13] which runs under the Java Runtime Environment (JRE) [14]. Even though the process varies in both approaches, the satisfiability of the argument should prove to be the same in both tools.

TABLE IV.
OR Operator

X	Y	X + Y
–1	–1	–1
–1	0	0
–1	1	2
–1	2	–1
–1	3	–1
0	0	0
0	1	1
0	2	0
0	3	0
1	1	1
1	2	1
1	3	1
2	2	2
2	3	2
3	3	3

The proposed algorithm is a variation of the Tableau method, the principle of proving the validity of an argument X by deducing that ¬X is not possible remains. However, instead of looking for contradictions in the deductive process, this method aims to find a valid resolution, discarding the cases that lead to contradictions until there are no alternative cases or until the validity of ¬X has been proved, in which case X is invalid and therefore not propositionally satisfied.

The algorithm is of recursive nature and the use of data structures is limited to arrays instead of Tableau's tree structure. It consists of two phases, in the first phase the goal is to break down the expression and analyse its operators; for this, it is subdivided by identifying where there are parentheses, and then within each group, the operations are analysed, taking into account the precedence of the operators. The implication operator is transformed using the implication law, so that only $\cdot, +$ and – are left in the expression.

After each case has been reduced to a simple operation, these are substituted by the notations of Table I, which covers all the possibilities. Next, the satisfiability of the whole expression is evaluated based on the algebra of Tables II, III, and IV for the corresponding operations.

The final outputs of the algorithm are squared arrays that stand for no contradictions if 3's are not found, or would proof the expression a contradiction if the opposite. The presence of 3 stands for a contradiction, implying that there is no possible outcome in which X is valid.

To summarize, the phases of the algorithm can be presented as follows:

1. Parentheses analysis.
2. Transform implications.
3. Divide the expression in simple operations (subcases).
4. Replace using notation from Table I and present the expression as an array.
5. Compare to check if there are contradictions (3's) with the proposed algebras (Tables II, III, and IV).
6. Output final arrays.

V. Results

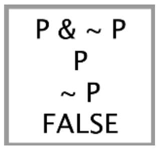
Table V shows the outputs of both programs evaluating the same logical expression for comparison purposes. As shown, the method introduced gives the same evaluation results as the classical tableau implementation of LoTREC.

TABLE V.
Tests

Argument	EZLogicUN	LoTREC
$p \cdot -p$	Contradiction	Contradiction
$q + p > p + q$	Satisfied	Satisfied
$(p > -q) \cdot (p > q)$	Satisfied	Satisfied
$q \cdot p \cdot (p > q)$	Satisfied	Satisfied
$p \cdot q \cdot (r + -q)$	Satisfied	Satisfied
$p + -p$	Satisfied	Satisfied

The following images are introduced to show a step by step of a classic Tableau Tree for evaluating some of the expressions in Table V. The images were generated using LoTREC, firstly it is presented a general structure of the tree and then each leaf node is extended to show if any contradiction arises (the word FALSE will indicate this at the end of the analysis) and to which branch of the tree the node belongs. If there if just one node forming the Tableau Tree as Fig. 1 then the node is displayed directly.

The expression evaluated in Fig. 1, p $\cdot$ −p, did not produce any branches during the tableau tree construction, rather it was concluded that the expression was FALSE in the first and only node of the tree.



```
P & ~ P
P
~ P
FALSE
```

Fig. 1. Tableau tree node output by LoTREC for the expression $p \cdot -p$

ISSN 2395-8618

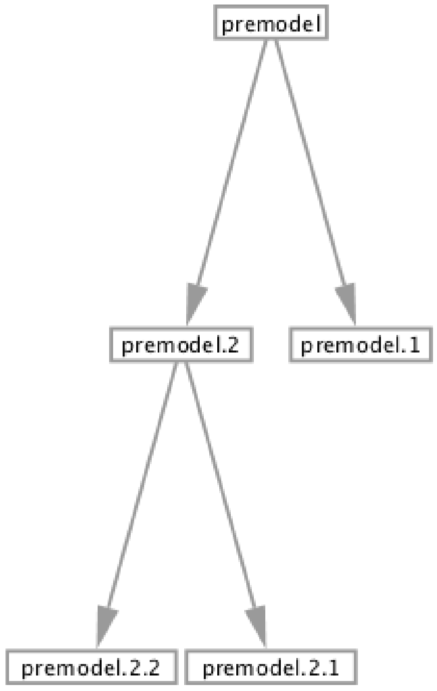
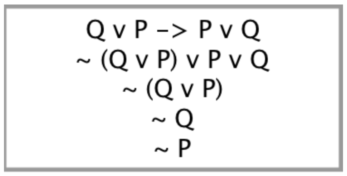


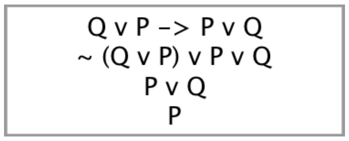
Fig. 2. Tableau tree structure output by LoTREC for the expression $q + p > p + q$

The next images depict the contents of each leaf node or premodel as stated by LoTREC of the tree in Fig. 2. These concluding leaf nodes are: premodel 1, premodel 2.1 and premodel 2.2.



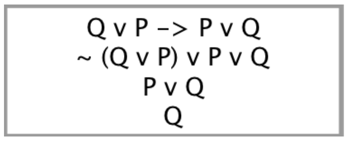
```
Q v P -> P v Q
~ (Q v P) v P v Q
~ (Q v P)
~ Q
~ P
```

Fig. 3. Tableau tree node output by LoTREC of **premodel 1** of the tree structure in Fig. 2.



```
Q v P -> P v Q
~ (Q v P) v P v Q
P v Q
P
```

Fig. 4. Tableau tree node output by LoTREC of premodel 2.1 of the tree structure in Fig. 2



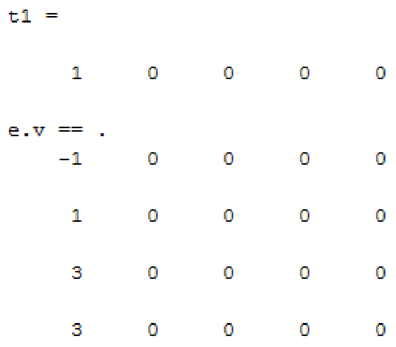
```
Q v P -> P v Q
~ (Q v P) v P v Q
P v Q
Q
```

Fig. 5. Tableau tree node output by LoTREC of **premodel 2.2** of the tree structure in Fig. 2

Since none of the leaf nodes presents (Fig 3, 4, and 5) any contradiction as the premodel in Fig. 1. Then it is concluded that the expression $q + p > p + q$ is TRUE.

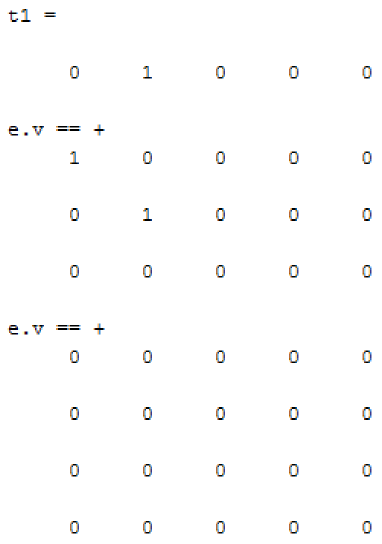
By using the script introduced by the authors for the expressions previously evaluated and explained in LoTREC, the script outputs the following arrays for expression $p.-p$ in Fig. 6 from the MATLAB software [12]. The presence of 3's in the output indicates that the expression is a contradiction and that there is no possible values for the formula that make it propositionally satisfied. Fig. 7 on the contrary does not include any 3's in the output arrays, this indicates that there are values that satisfy the expression $q + p > p + q$, which is in accordance to the output by the LoTREC software in Fig. 2 to 5.

The same analysis is done for expressions: $(\mathrm{p} > -\mathrm{q}) \cdot (\mathrm{p} > \mathrm{q})$ and $\mathrm{p} + -\mathrm{p}$. Analogously, the other expressions in Table V. were compared and analyzed to identify whether they were propositionally satisfied or contradictions.



```
t1 =

     1     0     0     0     0

e.v == .
    -1     0     0     0     0

     1     0     0     0     0

     3     0     0     0     0

     3     0     0     0     0
```

Fig. 6. Output array by EZLogicUN of the expression $p.-p$.



```
t1 =

     0     1     0     0     0

e.v == +
     1     0     0     0     0

     0     1     0     0     0

     0     0     0     0     0

e.v == +
     0     0     0     0     0

     0     0     0     0     0

     0     0     0     0     0

     0     0     0     0     0
```

Fig. 7. Output array by EZLogicUN of the expression $q + p > p + q$.

ISSN 2395-8618

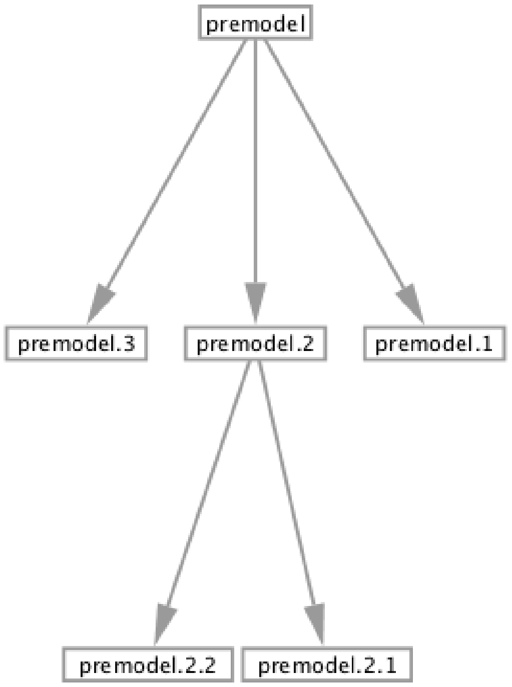
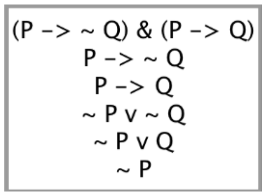
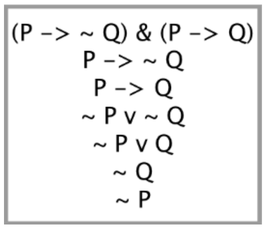


Fig. 8. Output tree by LoTREC of the expression $(p > -q) \cdot (p > q)$.



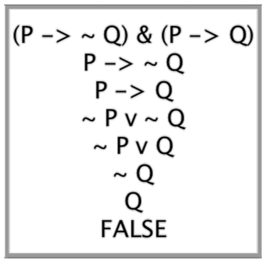
```
(P -> ~ Q) & (P -> Q)
P -> ~ Q
P -> Q
~ P v ~ Q
~ P v Q
~ P
```

Fig. 9. Pre-model 1 of the tree by LoTREC in Fig. 8 of the expression $(p > -q) \cdot (p > q)$.



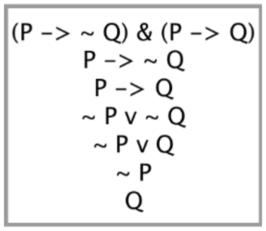
```
(P -> ~ Q) & (P -> Q)
P -> ~ Q
P -> Q
~ P v ~ Q
~ P v Q
~ Q
~ P
```

Fig. 10. Pre-model 2.1 of the tree by LoTREC in Fig. 8 of the expression $(p > -q) \cdot (p > q)$.



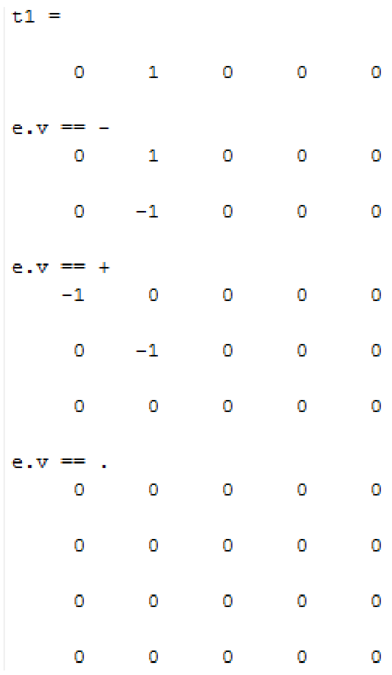
```
(P -> ~ Q) & (P -> Q)
P -> ~ Q
P -> Q
~ P v ~ Q
~ P v Q
~ Q
Q
FALSE
```

Fig. 11. Pre-model 2.2 of the tree by LoTREC in Fig. 8 of the expression $(p > -q) \cdot (p > q)$.



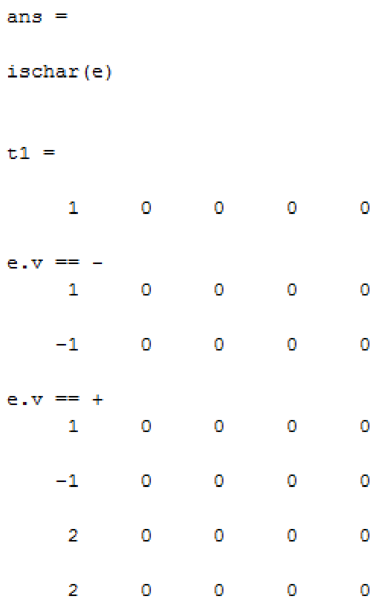
```
(P -> ~ Q) & (P -> Q)
P -> ~ Q
P -> Q
~ P v ~ Q
~ P v Q
~ P
Q
```

Fig. 12. Pre-model 3 of the tree by LoTREC in Fig. 8 of the expression $(p > -q) \cdot (p > q)$.



```
t1 =

     0     1     0     0     0

e.v == -
     0     1     0     0     0

     0    -1     0     0     0

e.v == +
    -1     0     0     0     0

     0    -1     0     0     0

     0     0     0     0     0

e.v == .
     0     0     0     0     0

     0     0     0     0     0

     0     0     0     0     0

     0     0     0     0     0
```

Fig. 13. Output array by EZLoginUN for the expression $(p > -q) \cdot (p > q)$.



```
ans =

ischar(e)


t1 =

     1     0     0     0     0

e.v == -
     1     0     0     0     0

    -1     0     0     0     0

e.v == +
     1     0     0     0     0

    -1     0     0     0     0

     2     0     0     0     0

     2     0     0     0     0
```

Fig. 14. Output array by EZLoginUN for the expression $p + -p$.

ISSN 2395-8618

VI. Conclusions

This paper has presented a novel approach for verifying the veracity of a zeroth order Boolean expression. The results obtained with this approach were compared to LoTREC. The implementation of the proposed approach requires simple data structures and its programming is straightforward due to the replacement policies based on the presented tables. Our method could improve the performance of decision-making systems based on logic sentences.

References

[1] J. T. Palma Méndez and R. L. Marín Morales, *Inteligencia artificial: técnicas, métodos y aplicaciones*. Aravaca: McGraw-Hill Interamericana de España, 2008.

[2] R. Jansana, “Propositional consequence relations and algebraic logic,” in *The Stanford Encyclopedia of Philosophy*, 2011.

[3] M. D’Agostino and D. M. Gabbay, *Handbook of tableau methods*. Springer, 1999.

[4] I. Niemelä, “Implementing circumscription using a tableau method,” in *ECAI*, 1996, pp. 80–84.

[5] P. Wolper, “The tableau method for temporal logic: An overview,” *Logique et Analyse*, vol. 28, no. 110–111, pp. 119–136, 1985.

[6] T. A. Junttila and I. Niemelä, “Towards an efficient tableau method for Boolean circuit satisfiability checking,” in *Computational Logic — CL 2000*, Springer, 2000, pp. 553–567.

[7] M. Y. Vardi, “Boolean satisfiability: Theory and engineering,” *Commun. ACM*, vol. 57, no. 3, pp. 5, 2014.

[8] “Propositional Logic | Internet Encyclopedia of Philosophy”.

[9] P. D. Magnus, “An Introduction to formal logic,” *University at Albany: State University of New York*, 2008.

[10] A. J. A. Robinson and A. Voronkov, *Handbook of Automated Reasoning*. Elsevier, 2001.

[11] H. K. Büning and T. Lettmann, *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999.

[12] “MATLAB—The language of technical computing.” [Online]. Available: http://www.mathworks.com/products/matlab/. [Accessed: 23-Jan-2015].

[13] L. F. del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci, “Lotrec: The generic tableau prover for modal and description logics,” in *Automated Reasoning*, Springer, 2001, pp. 453–458.

[14] “Java Software | Oracle.” [Online]. Available: https://www.oracle.com/java/index.html. [Accessed: 23-Jan-2015].