Teaching Semantic Tableaux Method for Propositional Classical Logic with a CAS

By Gabriel Aguilera-Venegas, José Luis Galán-García, María Ángeles Galán-García, Pedro Rodríguez-Cielos

University of Málaga, Spain
jlgalan@uma.es

Received: 14 September 2014 Revised: 2 January 2015 **DOI: 10.1564/tme_v22.2.07**

Automated theorem proving (ATP) for Propositional Classical Logic is an algorithm to check the validity of a formula. It is a very well-known problem which is decidable but co-NP-complete. There are many algorithms for this problem. In this paper, an educationally oriented implementation of Semantic Tableaux method is described.

The program has been designed to be used as a pedagogical tool to help in the teaching and learning process of the subject Propositional Classical Logic. Therefore, the main goal of the implementation is not to be very efficient but to get a useful tool for education. For this reason, the implementation of the Semantic Tableaux method provides an optional graphical approach which shows, step by step, the trace of the algorithm when applied to a specific problem. The algorithm has been implemented using a CAS (Computer Algebra System), specifically DERIVE, as part of the students training in Mathematics for Engineering. The result is a utility file containing a didactical implementation of the Semantic Tableaux algorithm which can be used not only at lectures but as a self-learning tool by students. With this utility file, students can check the results obtained by hand and, in case they get a wrong result, students can find the step where the mistake occurred. Furthermore, if the student does not know how to solve an exercise, the utility file can show how to overcome this situation step by step.

Since DERIVE does not have the Semantic Tableauxs algorithm *implemented, this utility file increases the capabilities of this CAS. Therefore, DERIVE can act as an automated theorem prover for Propositional Classical Logic from a PeCAS (Pedagogical CAS) point of view. The description of how to use this utility file will be shown throughout different examples related to validity, satisfability and deduction, which are the main kind of problems that an ATP can solve.*

Finally, since the Semantic Tableaux algorithm uses a tree structure, another utility file has been developed to deal with trees. In this paper, this utility file is also described.

1 INTRODUCTION

This work involves three important concepts: Automated Theorem Proving (ATP), programming with Computer Algebra System (CAS) and Mathematics Education in Engineering. It is not very usual to find these three concepts joined in a paper. The main difficulties for this combination are:

- In order to implement an ATP, there exist many general-purpose programming languages which provide easier tools than the ones offered by a CAS. This is one of the reasons why it is difficult to find an ATP developed in a CAS.

- In many occasions, CAS are used in Mathematics Education in Engineering as a power calculator machine which provide a result when required a computation. We think that by combining the power of a CAS with the flexibility of a programming language, better uses of CAS are reached (Galán, 2006). The development of appropriated tasks which combines both features have provided important advances in Mathematics teaching in Engineering (Galán, Galán, Padilla and Rodríguez, 2002).

- When teaching an ATP in engineering degrees, in many cases it is done in an abstract way. This fact could lead to a misunderstanding by students. A graphical tool which shows the execution of the corresponding algorithm improves the teaching and learning process and provides a self-learning procedure.

In this paper the file `Tableaux.mth`, developed in DERIVE, is presented and consists of a didactical implementation of the Semantic Tableaux Method. This utility file is especially useful for teachers and students in subjects on Computational Logic in Computer Science degrees. In particular, the file can be used as a didactical tool for helping in the teaching and learning process of ATP in Propositional Classical Logic.

Semantic Tableaux Method, also known as Analytic Tableaux Method, was invented by Beth (1955) and simplified by Smullyan (1968). The implementation of this method in `Tableaux.mth` considers Smullyan's uniform notation together with Jeffrey's tree structure (Jeffrey, 1981). For this reason, another utility file has been created to deal with tree structure using the programming features in DERIVE. In this paper this utility file (`tree.mth`) will be also described.

The main aim that has guided the development of the packages `tree.mth` and `Tableaux.mth` has been to obtain a practical didactically oriented tool for using automated theorem provers in Propositional Classical Logic. The initial idea was motivated by the necessity of introducing some

computer lectures in the subject of *Computational Logic* using a graphical approach in order to more easily follow the execution of the Semantic Tableaux algorithm. This graphical utility makes it possible that students can check the results obtained by hand and, in case they get a wrong result, students can find the step where the mistake is. Furthermore, if the student does not know how to solve an exercise, the utility file can show graphically how to overcome this situation step by step.

`Tableaux.mth` improves DERIVE facilities in the sense that the CAS can be used as an Automated Theorem Prover (ATP) for Propositional Classical Logic using the *Semantic Tableaux* algorithm. On the other hand, from an educational point of view, the step-by-step graphical approach considered allows to use DERIVE as a PECAS (Pedagogical CAS).

The Semantic Tableaux method can be used not only for pedagogical reasons but for solving "real" reasoning problems. On the other hand, the programs included in `tree.mth` can be used in a very wide range of applications which need the use of tree structures.

Semantic Tableaux method included in the package `Tableaux.mth` is a refutation system, that is, for proving the formula A, the formula A is introduced to the system. If A is a contradiction (unsatisfiable) then A is valid and in other case A is non-valid (and a countermodel is given). Similarly, for proving the reasoning $\{H_1, H_2, \dots, H_n\} = C$ the formula $B = H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge C$ is the input of the method and if the result is that B is satisfiable then the reasoning is non-valid (and a countermodel is given) and in other case the reasoning is valid. Therefore, the method in the package is a method to test the validity or satisfiability of a formula and the validity of a reasoning, providing a model (or countermodel) when needed.

In order to provide the students (and the readers) with a tutorial of how to use both `Tableaux.mth` and `tree.mth` packages, two more files have been developed: `Tableaux.dfw` and `tree.dfw`. These files content the description of both packages together with different examples of execution. These four files can be freely downloaded, used and modified and are allocated at

`http://www.matap.uma.es/jlgalan/Derive/Packages/`

In section 2, the reasons for choosing DERIVE for the development of this work are justified. In section 3, a brief description of *Semantic Tableaux* method is included for helping in the description of the package `Tableaux.mth`. In sections 4 and 5, the utility files `tree.mth` and `Tableaux.mth` are presented respectively, including the syntax of the programs and usage examples. Section 6 is dedicated to showing the graphical approach developed in order to make easier to follow the execution of Semantic Tableaux method. Finally, some conclusions and future work are shown in section 7.

2 WHY DERIVE 6 CAS?

The authors are convinced that the use of CAS in Mathematics teaching for Engineering degrees should be potentiated. One of the reasons is that the use of CAS improves students' fundamental skills which will be needed not only for other subjects but also in their future professional work. On the other hand, the mixture of the power of a CAS and the flexibility of its programming language with appropriated tasks, increases students' abilities and mathematical creativity.

It is well known that facilities offered by some general-purpose programming languages are more comfortable than the ones offered by CAS. However, the above mentioned conviction is so strong that we decided to use a CAS when implementing the didactical Semantic Tableaux algorithm. Furthermore, the addition of utility files (libraries) to a CAS leads to an improvement of their capabilities.

From the different CAS available on the market, we chose DERIVE 6 for several reasons:

1. The wide variety of mathematical resources DERIVE offers, together with the possibility of using a vector with elements of different types, make this software a powerful tool to use in algorithms dealing with tree structures.
2. In order to use an implementation of the Semantic Tableaux method with our Computer Science students, we wanted to use a software they were used to dealing with. In the mathematical engineering subjects, we use DERIVE to develop different programs to solve specific problems. For example, we have developed different utility files using DERIVE to assist in the teaching-learning process, files such as Multiple Integration (Aguilera, Cielos, Galán, Galán, Gálvez, Jiménez, Padilla and Rodríguez 2006), Line Integration (Cielos, Galán, Galán, Gálvez, Jiménez, Padilla and Rodríguez., 2006), Complex Analysis (Galán, Galán, Padilla and Rodríguez, 2004a), Random Variables (Aguilera, Galán, Galán, Padilla and Rodríguez, 2009), Automated Theorem Provers (Aguilera, Galán, Gálvez and Rodríguez, 2004; Aguilera, Galán, Gálvez, Padilla, Rodríguez and Rodríguez, 2012), Graph Theory (Aguilera, Galán and Rodríguez, 2006) or even Music counterpoint generation (Aguilera, Galán, Madrid, Martínez, Padilla and Rodríguez, 2010) and accelerated-time simulations (Aguilera, Galán, García, Mérida and Rodríguez 2014). Furthermore, the authors of DERIVE CAS published two of these utility files in the User Contributed Math Packages section of DERIVE 6 and can be used by any user (Galan, Rodríguez, Galán and Padilla, 2004; Galán, Rodríguez, Galán, Padilla, Sánchez and Sánchez, 2004). Therefore, our students are used to dealing with this software.

On the other hand, the main reasons we took into account when we chose DERIVE to work with our students were:

- Its easiness of use against other "more complex" CAS, mainly due to the available menus and shortcuts.
- Its very flexible syntax.
- Both previous reasons make the student to be able to start solving problems using DERIVE very quickly.
- Its low requirements, with regard either to memory or the physical space necessary for installing it.

3. DERIVE is a popular CAS widely used in the teaching of Mathematics for Engineering and even in High School Education all over the word (Galan, 2006).

With respect to the subject *Computational Logic*, computer lectures were necessary for helping students to work with automated theorem provers but little time was available for these computer lectures. DERIVE is a powerful tool that can be used in this subject and in others of the same degree, especially subjects involving mathematics. The time necessary for explaining basic characteristic of DERIVE is very short so the student can start working in computer lectures almost directly.

3 SEMANTIC TABLEAUX METHOD FOR PROPOSITIONAL CLASSICAL LOGIC

In this section, Semantic Tableaux method, one of the most well-known automated theorem provers for Propositional Classical Logic, will be briefly described. A more detailed description can be found in (Fitting, 1990; Aguilera and de Guzmán,1993 and de la Tour, 1990).

Semantic Tableaux is a refutation method. Initially, a set of propositional formulae $S = \{A_1, A_2, \ldots, A_n\}$ is given. The algorithm has to return `S SATISFIABLE` or `S UNSATISFIABLE`.

The uniform notation due to Smullyam (1968) will be introduced in order to describe the algorithm.

In this notation formulae are classified in α-formulae and β-formulae. Each α-formula is equivalent to the conjunction of two formulae α_1 and α_2 while each β-formula is equivalent to the disjunction of two formulae β_1 and β_2.

The table in Figure 1 shows the different possibilities.

α	α_1	α_2	β	β_1	β_2
$A \wedge B$	A	B	$A \vee B$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$	$\neg(A \wedge B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$	$A \rightarrow B$	$\neg A$	B
$\neg\neg A$	A	A	$A \leftrightarrow B$	$A \wedge B$	$\neg A \wedge \neg B$
			$\neg(A \leftrightarrow B)$	$\neg A \wedge B$	$A \wedge \neg B$

Figure 1 Possibilities for α and β formulae

Another important concept for the description of the algorithm is *tableau* or *Jeffrey's tree* (Jeffrey, 1981). A Jeffrey's tree is a tree which nodes are labelled with propositional formulae. A branch of a Jeffrey's tree is called *closed* if A and $\neg A$ occur in the branch for some formula A.

As an alternative definition, a branch is *closed* if a propositional symbol and its negation occur in the branch. Although the first definition is more general, the computational cost of checking if a formula A and its negation $\neg A$ occurring in a branch for each node of the Jeffrey's tree, is high. Statistically, it is unlikely that this occurs. Therefore, for computational reasons, the implementation developed uses the alternative definition since it will only be necessary to check this property when the content of the node is a literal (a propositional symbol or its negation). Furthermore, the authors' teaching experience reveals that it is easier for students to deal with this second definition.

The initial step in the algorithm is to consider the one branch tree associated with the set $S = \{A_1, A_2, \ldots, A_n\}$, that is:

$$\begin{array}{c} A_1 \\ A_2 \\ \vdots \\ A_n \end{array}$$

Let α be an unmarked α-formula. *To apply an α-rule to α* means to mark the formula and to add the tree T to every leaf in the tree, unclosed and descendant of α; where T is defined by

$$T = \alpha_1 \quad \text{if } \alpha_1 = \alpha_2$$

$$T = \begin{array}{c} \alpha_1 \\ | \\ \alpha_2 \end{array} \quad \text{in other case}$$

Analogously, let β be an unmarked β-formula. *To apply a β-rule to β* means to mark the formula β and to add the tree T to every leaf in the tree, unclosed and descendant of β, where T is defined by

$$T = \beta_1 \quad \text{if } \beta_1 = \beta_2$$

$$T = \begin{array}{c} \wedge \\ \beta_1 \quad \beta_2 \end{array} \quad \text{in other case}$$

A branch of a Jeffrey's tree is called *open* if it is not closed and all the formulae except literals in the branch are marked.

With the above definitions, Semantic Tableaux method can be described as the following algorithm:

1. Consider the one branch Jeffrey's tree corresponding to

   $$S = \{A_1, A_2, \ldots, A_n\}$$

2. If a branch of the Jeffrey's tree is open, then S is satisfiable (literals in the branch provide a model for S) and END.

3. If all the branches of the Jeffrey's tree are closed, then S is unsatisfiable and END.

4. If there exists an unmarked α-formula, then apply an α-rule to the first in deep formula and go to step 6.

5. If there exists an unmarked β-formula, then apply a β-rule to the first in deep formula.

6. Update open and closed branches.

7. Go to step 2.

Note that the algorithm ends since steps 4 and 5 produces smaller sub-formulae which finally lead to literals and steps 2 and 3 directly ends the algorithm. The soundness and completeness proofs can be found in (Aguilera and de Guzmán, 1993).

4 THE UTILITY FILE TREE.MTH

The following functions have been developed in the utility file `tree.mth` to manage binary trees, especially syntactic trees of propositional formulae.

- Basic tree management functions
  - NewBTree() returns the empty (binary) tree [].
    Example:
    # NewBTree()
    # []
  - MakeBtree(ro,le,ri) returns the binary tree which root is ro, his left son is le and his right son is ri.
    Example:
    # MakeBTree("O", 1, -2)
    # [1, O, -2]
  - Root(t) returns the root node of the tree t.
    Example:
    # Root([1, "O", -2])
    # O
  - Left(t) returns the left son of the root node of the tree t.
    Example:
    # Left([1, "O", -2])
    # 1
  - Right(t) returns the right son of the root node of the tree t.
    Example:
    # Right([1, "O", -2])
    # -2

- Strings and Trees
  - StringtoTree(f) makes the syntax analysis of f (f is a propositional formula delimited by quotation marks), returning the corresponding syntactic tree.
    Example:
    # StringtoTree("$(p \to q) \leftrightarrow (q \wedge \neg p)$")
    # [[1, I, 2], F, [2, O, -1]]
  - TreetoString(t) returns the string of the propositional formula which syntactic tree is t.
    Example:
    # TreetoString([[1, "I", 2], F, [2, "O", -1]])
    # $(p \to q) \leftrightarrow (q \wedge \neg p)$

The package `tree.mth` can be downloaded from
`http://www.matap.uma.es/jlgalan/Derive/Packages/`
together with the file `tree.dfw` a tutorial on how to use the package describing the syntax of all the programs with different examples. This tutorial can also be downloaded in `pdf` format.

5 THE PACKAGE TABLEAUX.MTH

The following functions have been developed in the utility file `Tableaux.mth` to use DERIVE as an Automated Theorem Prover for Propositional Classical Logic using *Semantic Tableaux* algorithm.

- SemanticTableaux(f) to check the satisfability of the formula f using Semantic Tableaux method. If A is unsatisfiable then [] is return. If A is satisfiable then a list of literal corresponding to a model of f is returned.
  Examples:
  # SemanticTableaux("$\neg (p \to p)$")
  # []
  # SemanticTableaux("$p \vee q$")
  # [1]

  SemanticTableaux is the main program and it holds Semantic Tableaux algorithm. The following programs use this one translating its result in order to provide the required answer.

- InferenceSemanticTableaux(h,f) to check if the formula f (conclusion) can be deduced from the list of formulae h (hypothesis) using Semantic Tableaux method. In case that the inference were non-valid a counter-model is returned.
  Examples:
  # InferenceSemanticTableaux(["$p \leftrightarrow q$","$p \to r$","$\neg r$"], "$\neg q$")
  # VALID INFERENCE
  # InferenceSemanticTableaux(["$p \leftrightarrow q$","$p \to r$","$\neg r$"], "q")
  # NON VALID INFERENCE. COUNTERMODEL:
  I(p)=0, I(q)=0, I(r)=1

- TAUTSemanticTableaux(f) to check if the formula f is a valid formula using Semantic Tableaux method. In case that the formula were non-valid a countermodel is returned.
  Examples:
  # TAUTSemanticTableaux("(p IMP q) or (q imp p)")
  # VALID FORMULA
  # TAUTSemanticTableaux("(p IMP q) AND (q imp p)")
  # NON VALID FORMULA. COUNTERMODEL:
  I(q)=0, I(p)=1

- SATSemanticTableaux(f) to check if the formula f is a satisfiable formula using Semantic Tableaux method.
  Examples:
  # SATSemanticTableaux("(p imp q) and (q imp p)")
  # SATISFIABLE FORMULA. MODEL:
  I(q)=0, I(p)=0
  # SATSemanticTableaux("(p or q) IFF (not p and NOT q)")
  # UNSATISFIABLE FORMULA

The package `Tableaux.mth` can be downloaded from `http://www.matap.uma.es/jlgalan/Derive/Packages/` together with the file `Tableaux.dfw` a tutorial on how to use the package describing the syntax of all the programs with different examples. This tutorial can also be downloaded in `pdf` format.

6 Graphical Approach

In this section the graphical approach we have developed in the package `Tableaux.mth` is described.

The main aim of this graphical approach is to display the trace of the Jeffrey's tree which holds all the information about an execution of Semantic Tableaux Algorithm.

The notation and symbols used for displaying the trace are:

- For each node in the Jeffrey's tree, the formula which it holds is displayed together with the list of literals that occur previously in the branch. Note that this list of literals will provoke a very fast search of closed branches for both, the computer and the user visualizing the trace of the Jeffrey's tree.

- In order to make a "good" visual display, if a node has only one son, the son is displayed just below it. On the other hand, if a node has two sons, the left and right sons are displayed in the line below, to the left and right hand side of the node, respectively.

- If a node is marked, the symbol √ is displayed next to the formula.

- When the symbol × is displayed, it means that the branch is closed.

- When neither the symbol √ nor × appear in a node, it means that its branch is open.

This graphical approach is optionally displayed, by using and extra parameter set to true, when running the different programs as shown in the following examples:

SemanticTableaux("¬ (p → p)", true)
(¬ (p → p)) √[]
p √[p]
¬p ×[p]
[]

SemanticTableaux("p ∨ q")
(p ∨ q) √[]
p √[p] q √[q]
[1]

InferenceSemanticTableaux(["p↔q","p→r","¬r"], "¬q")
(p ↔ q) √[]
(p → r) √[]
¬r √[¬r]
q √[q, ¬r]
(p → q) √[q, ¬r]
(q → p) √[q, ¬r]
¬p √[¬p,q, ¬r] r ×[q, ¬r]
¬p √[¬p,q, ¬r] q √[¬p,q, ¬r]
¬q ×[¬p,q, ¬r] p ×[¬p,q, ¬r] ¬q ×[¬p,q, ¬r] p ×[¬p,q, ¬r]
VALID INFERENCE

InferenceSemanticTableaux(["p→r","¬p"], "¬r", true)
p→r √[]
¬p √[¬p]
r √[r, ¬p]
¬p √[r,¬p] r √[r,¬p]
NON VALID INFERENCE. COUNTERMODEL:
I(r)=1, I(p)=0

TAUTSemanticTableaux("(p IMP q) or (q imp p)", true)
(¬((p→q) ∨ (q→p))) √[]
(¬(p→q)) √[]
(¬(q→p)) √[]
p √[p]
¬q √[¬q,p] [¬q,p]
q ×[¬q,p]
VALID FORMULA

TAUTSemanticTableaux("(p IMP q) AND (q imp p)", true)
(¬((p→q) ∧ (q→p))) √[]
(¬(p→q)) √[] (¬(q→p)) √[]
p √[p] q √[q]
¬q √[¬q,p] ¬p √[¬p,q]
NON VALID FORMULA. COUNTERMODEL:
I(q) = 0, I(p) = 1

SATSemanticTableaux("(p imp q) and (q imp p)", true)
((p→q) ∧ (q→p)) √[]
(p→q) √[]
(q→p) √[]

```
                    ¬p √[¬p]                                  q √[q]
  ¬q √[¬q,p]        p ×[¬p]          ¬q ×[q]          p √[p,q]
# SATISFIABLE FORMULA. MODEL: I(q) = 0, I(p) = 0
```

The graphical approach described in this section has been used when teaching in the computer lectures and it has been a very important help in the teaching and learning process.

7 CONCLUSIONS AND FUTURE WORK

7.1. Conclusions

In this work, a graphical tool to help in the teaching and learning process of ATP for engineering students, has been introduced. To achieve this goal, the Semantic Tableaux Method has been implemented in the package `Tableaux.mth`, developed in DERIVE 6. This package allows the use of DERIVE as an Automated Theorem Prover for Propositional Classical Logic increasing the facilities of this CAS. The file `Tableaux.dfw` has a brief description of every user function and some examples and it can be used as a tutorial.

On the other hand, the structure of binary tree is widely used in the implementation of the Semantic Tableaux method. Therefore, in order to manage the tree structure, another package `tree.mth` has been added together with the tutorial `tree.dfw`. This utility file also improves DERIVE CAS facilities since it can be used for applications which need to deal with a tree structure.

The execution of the Semantic Tableaux method can be easily followed thanks to the graphical approach developed. It provides the trace of the Jeffrey's tree associated to each execution of the algorithm.

After using the package `Tableaux.mth` with our students, they affirm that it has been quite useful in order to understand the Semantic Tableaux algorithm. Even more, with the help of the graphical approach and the tutorial `Tableaux.dfw`, our students say that they could compare the result of proposed exercises with the solution provided by DERIVE and they could correct the mistakes.

The files `tree.mth`, `tree.dfw`, `Tableaux.mth` and `Tableaux.dfw` can be freely downloaded from:
`http://www.matap.uma.es/jlgalan/Derive/Packages/`

7.2. Future work

We are interested in extending this didactical and graphical approach to implement other ATP. Specifically:

- In the short term, we will implement two other ATP for Propositional Classical Logic: Quine and resolution (linear resolution and ground resolution).

- In the medium term, another ATP called TASD (Aguilera, de Guzmán, Ojeda-Aciego and Valverde, 2001) will be implemented.

- In the long term different ATP for First Order Logic and for some non-standard logics will be considered.

Finally, as part of one of our research interests, we will keep developing other packages for different subjects. We will also translate the packages we have already developed to other CAS, such as TI-NSPIRE CAS or MAXIMA. In fact, the authors have already developed some accelerated-time simulations (Aguilera-Venegas, Galán-García, Mérida-Casermeiro and Rodríguez-Cielos., 2014; Galán-García, Aguilera-Venegas and Rodríguez-Cielos, 2014) where the CAS used is MAXIMA.

REFERENCES

Aguilera, G., Cielos, C., Galán, J. L., Galán, M. Á., Gálvez, A., Jiménez, A. J., Padilla, Y. and Rodríguez, P. (2006) A basic course of multiple integrals with a computer algebra system, *Proceedings of the 3rd International Conference on the Teaching of Mathematics at the Undergraduate Level*. Istambul.

Aguilera, G. and de Guzmán, I. P. (1993) *Lógica para la Computación vol. I*. Agora.

Aguilera, G., de Guzmán, I. P., Ojeda-Aciego, M. and Valverde, A. (2001) Reductions for non-clausal theorem proving, *Theoretical Computer Science*, **266** (1-2), 81-112.

Aguilera, G., Galán, J. L., Galán, M. Á., Padilla, Y. and Rodríguez, P. (2009) Generating random samples from continues and discrete distributions with DERIVE. *DERIVE Newsletter*, **75,** 22-43.

Aguilera, G., Galán, J. L., Gálvez, A., Padilla, Y., Rodríguez, P. and Rodríguez, R. (2012) A stepwise graphical approach for teaching automated theorem proving in engineering. *Proceedings of TIME 2012*, Tartu, Estonia.

Aguilera, G., Galán, J. L., Gálvez, A. and Rodríguez, P. (2004) `Atpcl.mth`: Automated theorem provers for propositional classical logic with DERIVE, *Proceedings of TIME 2004*. Montreal.

Aguilera, G., Galán, J. L., García, J. M., Mérida, E. and Rodríguez, P. (2014) An accelerated-time simulation of car traffic on a motorway using a CAS, *Journal of Mathematics and Computer in Simulation,* **104**, 21-30.

Aguilera, G., Galán, J. L., Madrid, R., Martínez, A. M., Padilla, Y. and Rodríguez, P. (2010) Automated generation of contrapuntal musical compositions using probabilistic logic in DERIVE. *Journal of Mathematics and Computer in Simulation*, **80**(6), 1200-1211.

Aguilera, G., Galán, J. L. and Rodríguez, P. (2006) Graph `algorithms.mth`: Graph algorithms using display step in DERIVE 6, *Proceedings of DES-TIME-2006*, Dresden.

Aguilera-Venegas, G., Galán-García, J. L., Mérida-Casermeiro, E. and Rodríguez-Cielos, P. (2014) An

accelerated-time simulation of baggage traffic in an airport terminal. *Journal of Mathematics and Computer in Simulation*, **104**, 58-66.

Beth, E. W. (1955) Semantic entailment and formal derivability. Mededlingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N.R. Vol 18, no 13, 309-342.

Cielos, C., Galán, J. L., Galán, M. Á., Gálvez, A., Jiménez, A. J., Padilla, Y. and Rodríguez, P. (2006) Line integrals with computer algebra systems, *Proceedings of the 5th International Conference*, *APLIMAT*, Bratislava.

de la Tour, T. B. (1990) Minimizing the number of clauses by renaming, *Proceedings of 10th CADE, Kaiserslautern*, Heidelberg: Springer, 558-572.

Fitting, M., (1990) *First-Order Logic and Automated Theorem Proving*, Springer-Verlag.

Galán, J. L. (2006) The role of computer algebra systems in mathematics teaching for engineering degrees, Plenary Lecture, *Proceedings of the 5th International Conference APLIMAT*, Bratislava.

Galán, J. L., Galán, M. Á., Padilla, Y. and Rodríguez, P. (2002) Are computers under-used in mathematical teaching for engineers? *Educational Technology, Serie Sociedad de la Información*, **9**, 220-225.

Galán, J. L., Galán, M. Á., Padilla, Y. and Rodríguez, P. (2004) `Residues.mth`: Solving problems of integration using the residue theorem, *Proceedings of TIME 2004*, Montreal.

Galán, J. L., Rodríguez., P., Galán, M. Á. and Padilla, Y. (2004) Multiple `integration.dfw`: Line, double, triple and surface integrals, *User Contributed Math Packages of software* DERIVE *6*, Texas Instruments.

Galán, J. L., Rodríguez., P., Galán, M. Á., Padilla, Y., Sánchez, E. and Sánchez, J. (2004) Complex `analysis.dfw`: Complex numbers and functions, *User Contributed Math Packages of software DERIVE 6*, Texas Instruments.

Galán-García, J. L., Aguilera-Venegas, G. and Rodríguez-Cielos, P. (2014) An accelerated-time simulation for traffic flow in a smart city, *Journal of Computational and Applied Mathematics,* **270**, 557-563.

Jeffrey, R. (1981) *Formal Logic: its scope and limits*, Toronto: McGraw-Hill.

Smullyam, R. M. (1968) *First-Order Logic*, Berlin:Springer-Verlag.

BIOGRAPHICAL NOTES

Dr. Gabriel Aguilera-Venegas, Dr. José Luis Galán-García, and Dr. Pedro Rodríguez-Cielos are Associate Professors while Dra. María Ángeles Galán-García is an Assistant Professor of different mathematic subjects in Engineering at the University of Málaga, Spain. One of their research interests is the use of CAS in mathematics lectures in the different Engineering degrees they teach. Dr. José Luis Galán-García and Dr. Pedro Rodríguez-Cielos have Ph.D Degrees in Mathematics Education while Dr. Gabriel Aguilera-Venegas and Dra. María Ángeles Galán-García have Ph.D Degrees in Computer Science. The authors have presented more than 100 talks in different International Conferences on the topic "CAS in Math subjects in Engineering".