An Algorithm for Automatic Demonstration of Logical Theorems

Orlando Zaldivar-Zamorategui – Jorge Carrera-Bolaños

Abstract. The automatic demonstration of theorems (ADT) is and has been an area of intensive development in Artificial Intelligence (AI). There are actually a variety of procedures, some already implemented as software, that offer the possibility to establish the truth value of a given formula (“theorem”) in a given contextual system. In this paper we present also an ADT system for first order logic (propositional calculus). This system is at least as powerful as any other actually available, but simpler and easy to program. It is presented in algorithmic form. The presentation offers also an opportunity to make some considerations about the relation of theoretical logic to its applications and of the meaning of the concepts of “theorem” and “demonstration”. Specially, an effort has been made to give an adequate definition in this context of a “sufficient”, a “necessary” and a “sufficient and necessary condition”.

Introduction.

We present here a powerful algorithm for the demonstration of theorems of propositional calculus of the first order. It is constructed based on some ideas of natural deduction and in sequentional reasoning. It has some new characteristics that make it flexible and efficient.

The theoretical support of the development here presented is given by a known theorem that guarantees the unicity of the principal or canonical disjunctive and conjunctive normal forms of a given well constructed formula (WCF), and the possibility to use this fact to establish if the original formula is a tautology, a contradiction or a contingency. But from the standpoint of Automatic Demonstration of Theorems (ADT) as part of AI, the evaluation of a formula is a necessary, not a sufficient condition for a “demonstration”. We are looking for something similar to a demonstration as it is understood in logic or mathematics, and for an efficient implementable computational algorithm. Thus, we are working in the field of applied logic. Some considerations about
the meaning of a “demonstration of a theorem” are given in Section 3. About the idea of “efficiency” and “computational implementation” an analogy to applied mathematics illustrates this aspect of applied logic.

The theory of partial differential equations (PDE) guarantees, under specific premises, that a given PDE with initial and boundary conditions has a unique solution, but it does not say how to construct it. In Physics it is not only needed to construct the solution, but also to have an efficient algorithm for it. In the here analyzed situation it is not only needed to find the semantics of a given formula, but also to have at least an approximation to a demonstration such as a human agent would consider it, including the question of which axioms are needed and under which hypotheses, and which role they play, how the demonstration is made, and in the negative case, why the demonstration fails. That is, a demonstration should provide an insight in the meaning of a theorem.

The presented algorithm was deduced pragmatically from previous experiences, our own as well as other ones reported, and it offers:

a) Flexibility. The only requirement on the theorem is that it can be expressed as a sequence of WCF.
b) Power. We are convinced that any first order deduction formalizing a real situation can be successfully analyzed. Clearly, not every real situation can be logically formalized.
c) Efficiency. It is at least as efficient as other current systems, but it is shorter.
d) No artificial additional hypotheses are needed.

In the next section we present the logical support. The third section gives our conception of automatic demonstration. Then the algorithm is presented and analyzed. In Section 4 an example-exercise is proposed.
2. Principal conjunctive and disjunctive normal forms.

It is known that to every WCF in propositional logic there are equivalent disjunctive and conjunctive normal forms (DNF, CNF). Unfortunately, they are not unique. But its principal or canonical disjunctive or conjunctive form is unique.

Definition. Be $f$ a WCF of propositional logic. A CNF (DNF) of $f$ is called principal or canonical conjunctive (disjunctive) normal form, PCNF (PDNF) of $f$, if all atomic variables of the original formula appear in every term.

Theorem. The PCNF (PDNF) of a given WCF $f$ is unique [Asser, p. 55].

3. Demonstrations.

A couple of considerations are important in this paper.

3.1. A “theorem” in this kind of systems is not necessarily a tautology. A theorem is valid here if there is no evaluation of the hypotheses that would render them true and the conclusion (the affirmation of the “theorem”) false. Thus, the whole system of premises and conclusion:

$$H_0, H_1, \ldots, H_n \rightarrow C$$

(1)

Is taken in consideration.

Our formal system is indeed a subsystem of an axiomatic-like system where no axioms appear explicitly.

3.2. These considerations show the relativity of the concept of “theorem”. From a strictly formalized point of view, in an axiomatic system any WCF that is not an axiom requires a series of transformations (a “demonstration”) that begins with the formula and ends with one or more axioms (or vice versa) that allow the classification of the formula as a tautology, a
contradiction or a contingency. Thus, any tautology can be called a “theorem”.

A contingency \( f \) can also be called theorems if there is a subsystem in which \( f \) is a tautology.

Our “definition” is even more restricted, because its premises can have a low degree of generality.

In any one of these three cases, any formula \( f \) that complies with the formal requisites can be called a “theorem”. The demonstration is then semantically equivalent to the expressions:

- Under any assignation of truth values to the variables of \( f \), the truth value of \( f \) is T (tautology).
- There is a well defined class of valuations that makes \( f \) take the value T (contingency).
- For any valuation that makes a series of other formulas (“premises”) true, \( f \) takes the value T (Section 3.1).

3.3. However, in mathematics and in theoretical logic not every affirmation is called a “theorem”. It is implicitly assumed that a “theorem” is an affirmation “that makes sense”. This is not a context free characteristic of a formula, and thus not formally definable in the logical system. Our approach to the “meaning” of a formula \( f \) consists in an analysis of the relations between \( f \) and its premises; that is, in a procedure to “gain insight” in the demonstration of \( f \). Our algorithm can be used for it (Section 4.2).

3.4. A conception of “proof”. The mere formal derivation of the validity of a deduction of the form (1) is a necessary but not a sufficient condition to call the procedure a “proof”. One of us (Carrera) has made an analysis of some conceptual requirements for a procedure to be called a “proof” [Carrera]. Even some “emotional” aspects would have to be taken into consideration, specially the “power of conviction”. For example, the recent proof of Fermat’s theorem is surely formally correct, but it lacks some of this power. Clearly, in an automatic demonstration such aspects cannot play a role.
As stated above, in a more radical rational conception a proof should offer an “understanding” of the structural role played by the hypotheses, in order to give the conclusion a “meaning”. Even if an ADT-procedure by its own nature is semantically restricted to considerations of truth values, it should offer at least:

a) the capability to detect superfluous hypotheses;

b) the possibility to establish if a hypothesis is necessary, sufficient or both;

c) if the answer is negative (the deduction is not correct), the possibility to establish the role of a given hypothesis, id est, if the procedure fails because of that hypothesis or not.

Because an ADT system is almost per definition a computational system, it should also be computationally efficient, have a friendly interface, etc. This part is outside the scope of this paper, but we can empirically assure that our algorithm is computationally at least as efficient as any commercial one, if not better.

4. A system for ADT based on natural deduction.

Natural deduction is “closer” to “natural” reasoning, and thus it plays a basic role in ADT and in particular in our algorithm. The operational basis of the (sub) system consists of the De Morgan rules, two rules of inference and a table of tautological implications.

Rule 1. A hypothesis can be introduced anywhere in the derivation (the “deduction”).

Rule 2. A formula $S$ can be introduced in a derivation if it is tautologically implied in some or all of the preceding formulas.

One of the main advantages of the system here presented is its reduced number of tautological implications used, only four of them:

\[
\begin{align*}
I_1 & \quad P \land Q \vdash P \\
I_2 & \quad P \vdash P \lor Q
\end{align*}
\]
\[ I_3 \, P, \, Q \supset P \land Q \]
\[ I_4 \, \neg P, \, P \lor Q \supset Q \]

Given \( n+2 \) WCF \( H_0, H_1, \ldots, H_n \) and \( C \), it is asked if \( C \) can or cannot be deduced from the conjunction of the other formulas. The following system gives an answer, \( T \) or \( F \) or an indication of no solubility, in a finite number of steps and it can be efficiently implemented in a personal computer. It can afterwards be conveniently iterated in order to analyze the role of every single hypothesis (Section 4.2).

4.1. *The system*. It consists of:

1. An alphabet, including symbols for atomic variables and connectives, parentheses, commas.

2. Syntactic rules that determine how to establish if a given sequence of symbols is a WCF and how to construct its PCNF and PDNF.


4. Inference rules:

   De Morgan Rules
   Rule 1
   Rule 2
   Tautological inferences I_1, I_2, I_3, and I_4.

**Algorithm** (in a semiformalized form; only line 28th is signalized for its later use):

1. Initialization.
2. Input: Set of hypotheses \( H_0, H_1, \ldots, H_n \); one conclusion \( C \).
3. If (the set of hypotheses and conclusion has one or less elements) or (at least one \( H_0, H_1, \ldots, H_n, C \) is not a WCF) then
   - Give an output indicating the situation and finish
   - Else
     - Continue.
4. Identify all atomic propositions appearing in all hypotheses and \( C \).
5. For \( I = 0 \) to \( n \)
   - If (\( H_i \) does not contain all atomics) construct an equivalent formula including them; call it again \( H_i \).
   - Else
     - Continue
6. Obtain a formula equivalent to \( C \) that includes all atomics, call it again \( C \).
7. Obtain PDNF of \( H_0 \), call it again \( H_0 \).
8. If \( n \geq 1 \) then
   - For \( I = 1 \) to \( n \)
     - Obtain PCNF of \( H_i \); call it again \( H_i \).
     - Find the common terms of \( H_0 \) and \( H_i \).
     - Rearrange \( H_0 \) as a disjunction of two subformulas, \( H_0 = H_{01} \lor H_{02} \), such that \( H_{01} \) contains only the common terms, \( H_{02} \) the others.
     - From \( H_i \) obtain an expression, using tautological rules, that contains only the common terms, call it again \( H_i \).
     - Obtain the De Morgan dual expression of \( H_i \), call it \( H_i \). It is the negation of \( H_{01} \).
     - Using \( I_4 \) obtain \( H_{02} \) from \( H_i \) and \( H_0 \), taking into account that \( H_i \) is now \( \neg H_{01} \) and \( H_0 = (H_{01} \lor H_{02}) \), thus \( \neg H_{01}, H_{01} \lor H_{02} \Rightarrow H_{02} \).
   - [Line 28\textsuperscript{\textcircled{a}}] If needed, minimize \( H_{02} \).
     - Set \( H_0 := H_{02} \).
   - Else
     - \( H_0 := H_0 \).
9. Find the PDNF of \( C \), call it again \( C \).
10. If (all terms in \( H_0 \) are included in \( C \)) then
Use $I_2$ to introduce in $H_0$ all terms of $C$ that are not included in it.

$H_0$ and $C$ are equivalent.

Else

Give as output $T$.

11. Finish.

Notice that formally the value $F$ is equivalent to the expression “under the given premises the conclusion cannot be given the value $T$”. That can have three meanings:

☞ The conclusion $C$ is a contradiction.

☞ At least one of the premises is not compatible with $C$.

☞ At least one additional hypothesis is needed to prove $C$.

The next section shows how to handle this situation in order to gain insight.

4.2. **Understanding the system and using it to approach understanding.** The main idea of the procedure is to leave in $H_0$ only the specific contribution of every hypothesis, leaving aside all terms that appear in two, four, six or generally in $2k$ hypotheses (including $H_0$). Once it is done, the equivalence of the information condensed in the last version of $H_0$ with $C$ can be determined.

Due to the fact that the premises $H_i$ are processed sequentially, the algorithm can be used to analyze the role played by each $H_i$. First it can be established if a premise is not needed:

A. After line 28:

If needed, minimize $H_{i,2}$.

Introduce the following instructions:

Compare $H_i$ and $H_{i,2}$.

If all terms of $H_{i,2}$ are included in $H_i$ then

Give out the message “Hypothesis $H_i$ superfluous”
Else
    Construct the expression $\text{Info}H_i = H_{02} - H_0$
    Give out the message “The particular
    information
    contained in $H_i$ is “ $\text{Print}(H_{02})$
    Continue
Where the command $\text{Print}(H)$ gives out the expression $H$.
In this way it can be established first if hypothesis $H_i$ is
superfluous or not, and in the last case what information it
provides in relation to $H_0$, $H_1$, ..., $H_{i-1}$. Because the order of
introduction of the hypothesis is not essential, the user of the
program can change it to contrast the information given by one
hypothesis in relation to all others.

B. In the negative case, when the demonstration fails, we propose
the sequentional introduction of the negated hypotheses $\neg H_i$. The
user can then establish if one or more specific hypothesis have to
be changed by its negation to prove the theorem.
The failure may also be due to lack of information. That can be
determined if even changing every hypothesis by its negation the
theorem cannot be proved.

C. The next case is when even if a given hypothesis $H_k$ gives non-
redundant information, $H_0 \cap H_k \neq \emptyset$ (in the k-th iteration), this
information is irrelevant for the considered situation. This is not
detected by the algorithm because the final comparison between
$H_0$ and $C$ is between disjunctive forms. We propose to use the
algorithm eliminating in a sequential way each one of the
hypothesis. This is also a procedure to corroborate the final result.
D. The algorithm can also be used to establish which hypotheses
are necessary and which ones are sufficient for the theorem to be
true. A hypothesis $H$ is necessary if its exclusion from the set of
premises implies the negation of the theorem (as a whole, not only
$C$). A set of necessary premises is only necessary but not sufficient
if their sole presence does not allow the demonstration of the
theorem. A premise $H$ is sufficient if added to a set of necessary
but not sufficient premises allows the demonstration, but there is
at least a different hypothesis $H' \neq H$ that allows the demonstration if $H$ is not present. Notice that there is no formal procedure to find $H'$. A set of necessary hypotheses is also sufficient if it allows the demonstration.
In its present form, the proposed algorithm has to be used as a trial-and-error tool to establish which hypotheses are necessary and/or sufficient.

5. An example-exercise. We encourage the reader to use the algorithm for the following situation (limitations in space does not allow us to show the complete development):
$H_0$: If Kramnik wins the Chess World Championship then Fischer gets second place.
$H_1$: (Fischer does not get second place or Spasski is eliminated) and (Spasski is not eliminated).
$H_2$: It is not the case that (Kramnik does not win the Championship and Topalov is eliminated).
Conclusion $C$: Topalov does not get eliminated.

Formalizing these sentences:
Atomic propositions:
P: Kramnik wins the Championship.
Q: Fischer gets second place.
R: Spasski is eliminated.
S: Topalov is eliminated.
Hypotheses or premises:
$H_0$: $P \rightarrow Q$
$H_1$: $(\neg Q \lor R) \land \neg R$
$H_2$: $\neg(\neg P \land S)$.
Conclusion:
C: $\neg S$

6. Conclusions. The algorithm is based on some basic facts that are used to generate a tool that does not only offers a formal procedure to validate a given sequence of formulas, but can be used to gain theoretical and practical insight on some aspects of the meaning of a “theorem” formalizing a real situation, and also about the meaning of the concept of “theorem”.
In future papers we will further explore the concept of “theorem”.

Orlando Zaldivar-Zamorategui* zazor1@fi-b.unam.mx
Jorge Carrera-Bolaños * jorge.carrera03@hotmail.com

References

Carrera, J., *Notes for a Course in Conceptual Logic*, internal publication of the Postgraduate Division, Faculty of Engineering, UNAM, Mexico, 2003.