Complexity of the Propositional Tautology Problem for t-norm Logics

Haniková, Zuzana
2002

Complexity of the propositional tautology problem for t-norm logics

doktorand:
ZUZANA HANIKOVÁ
Ústav informatiky AV ČR, 182 07 Praha 8
zuzana@cs.cas.cz

školitel:
PROF. RNDR. PETR HÁJEK, DRSC.
Ústav informatiky AV ČR, 182 07 Praha 8
hajek@cs.cas.cz

obor studia:
Matematická logika

Abstrakt

The propositional tautology problem for any logic given by a continuous t-norm is coNP complete.

1. Foreword

This paper is a preliminary and incomplete version of [5]; some proofs have been omitted for the sake of brevity, and for the same reason, it is impossible to give a full introduction to the topic. A comprehensive treatment of the approach we follow will be found, e. g., in [4].

2. Introduction

t-algebras (or *standard* algebras) are a frequently used class of algebras of truth values for many-valued logics. Each t-algebra is determined in a unique manner by a continuous t-norm on $[0, 1]$ (hence the term).

It is known that the propositional logic BL, investigated in [4], is complete w. r. t. the tautologies of all t-algebras. This result comes from [3]. It is also known that some t-algebras are BL-generic; [1] gives a characterization of these. Moreover, [2] shows the tautologies of all t-algebras (or equivalently, the propositional BL) to be coNP complete. Thus the complexity of the propositional tautology problem is settled for BL-generic t-algebras.

Three important schematic extensions of BL, namely the logic of Lukasiewicz, of Gödel, and the product logic, have been investigated thoroughly, and their propositional tautologies have also been proved to be coNP complete ([4] gives further references; in particular, the coNP completeness of propositional L-tautologies comes from [7]).

The aim of this paper is to adjust the algorithm presented in [2] and prove the following claim:

Theorem 2.1 *For any t-algebra, the propositional tautology problem is in coNP.*

Once established, this theorem settles the question of complexity of propositional tautologies for an arbitrary t-algebra, in combination with an earlier result:

Theorem 2.2 *For any t-algebra, the propositional tautology problem is coNP hard.*

This comes from [2] for t-algebras starting with an L (proved via reduction of propositional L-tautologies, prefixing a negation to each propositional variable) and from [4] for t-algebras not starting with an L (proved via reduction of propositional Boolean tautologies, prefixing a double negation to each propositional variable).

Throughout the paper we use heavily the Mostert-Shields decomposition theorem for continuous t-norms, originating in [6], and employ some rather informal notation based on it. The statement of the theorem is that the "backbone" of any continuous t-norm is formed by a countable closed subset I of $[0,1]$ (we use the term 'cutpoints' for the elements of I), and on each of the closures of the open intervals which form the complement of I, the t-norm is isomorphic to either Lukasiewicz, Gödel, or product t-norm (on $[0,1]$). For this reason each t-algebra is an ordinal sum of copies of Lukasiewicz, Gödel, and product algebras, which we habitually call *segments* and denote with symbols L, G, and Π. We stress that each copy of Gödel counts as one segment, thus, e. g., $[0,1]_{\mathrm{L}\oplus\mathrm{G}\oplus\Pi}$ is a t-algebra with three segments, namely a sum of a copy of the Lukasiewicz algebra, a copy of the Gödel algebra and a copy of the product algebra; the type of the sum is L⊕G⊕Π. We disregard the exact positioning of the set I within $[0,1]$.

3. Finite ordinal sums

[2] gives an NP algorithm recognizing BL-couterexamples. In fact, it shows more than that: by a trivial modification, for any finite sum of L-segments only, the set of its non-tautologies is in NP.

To prove our claim for finite ordinal sums, we generalize the algorithm of [2] to recognize non-tautologies (i.e., formulas for which there is a counterexample evaluation) in an arbitrary fixed t-algebra which is a finite ordinal sum. Fix A as such a t-algebra, and let n be its cardinality (i. e., the number of segments in the sum). For a propositional formula φ, let $m = 2|\varphi|$, where $|\varphi|$ is the number of occurrences of propositional variables in φ (so m is an upper bound on the number of the subformulas of φ).

What follows is, we claim, an NP algorithm which for an input formula φ decides whether there is an evaluation e in A s. t. $e(\varphi) < 1$. It is a modification of the algorithm of [2]: we drop, for the moment, the step which guesses the cardinality of the sum, since A is fixed. The generalization, which adds a check for G-segments and Π-segments, comes in the `checkInternal()` step, which will be discussed subsequently.

`// algorithm for finite sum A`

{

`cutpointVariables()` Introduce variables $z_0 < \cdots < z_n$ for the cutpoints of A (thus z_0 is intended for 0 and z_n is intended for 1).

`intervalVariables()` For each $i = 0, \ldots, n-1$ introduce variables $z_i = y_{i0} < y_{i1} < \cdots < y_{im} = z_{i+1}$ We call these *the variables belonging to* i. By convention, two variables which are equal are interchangeable in all contexts (thus also z_i, z_{i+1} belong to i, $i = 0, \ldots, n-1$).

Since the values of all subformulas could belong to a single segment and each subformula could evaluate to a different element of the segment, it is vital to have enough variables belonging to each $i = 0, \ldots, n-1$. Note that this is so, since each i contains $m + 1 = 2|\varphi| + 1$ variables, of which two represent the cutpoints, while the total number of subformulas is at most $2|\varphi| - 1$; so the number of variables is sufficient for any type of evaluation.

Set $C = \{z_0, \ldots, z_n\} \cup \{y_{ij} | i = 0, \ldots, n-1,\ j = 0, \ldots, m\}$.

`guessAssignment()` Guess an assignment f of variables in C to subformulas of φ (an "evaluation" of subformulas of φ with variables in C), s. t. $f(\varphi)$ is not z_n.

`checkExternal()` Check external soundness of f: if $u, v \in C$, $f(\varphi_1) = u$, $f(\varphi_2) = v$, then

- if $\varphi_1 \& \varphi_2$ is a subformula of φ and, for some i, $u \leq z_i \leq v$, then $f(\varphi_1 \& \varphi_2) = f(\varphi_1) = u$;
- if $\varphi_1 \to \varphi_2$ is a subformula of φ and $u \leq v$ then $f(\varphi_1 \to \varphi_2) = z_n$;
- if $\varphi_2 \to \varphi_1$ is a subformula of φ and for some i, $u < z_i \leq v$, then $f(\varphi_2 \to \varphi_1) = f(\varphi_1) = u$.

`checkInternal()` Check internal soundness of f for each segment. Consider the i-th segment. For each subformula $\varphi_1 \& \varphi_2$ s. t. $f(\varphi_1) = y_{ij}$ and $f(\varphi_2) = y_{ik}$, if $f(\varphi_1 \& \varphi_2) = y_{il}$ (that is, all three variables involved belong to i), put down an equation $y_{ij} * y_{ik} = y_{il}$, and for each subformula $\varphi_1 \to \varphi_2$ s. t. $f(\varphi_1) = y_{ij}$ and $f(\varphi_2) = y_{ik}$, where $j > k$, if $f(\varphi_1 \to \varphi_2) = y_{il}$, put down an equation $y_{ij} \Rightarrow y_{ik} = y_{il}$. Check whether these equations, together with the sharp inequalities $y_{i0} < \cdots < y_{im}$, have a solution in the i-th segment of the sum, s. t. y_{i0} and y_{im} evaluate to the lower and upper cutpoint of the segment, respectively.

}

The last check in the above algorithm is the same as finding a solution in the Lukasiewicz, Gödel, or product t-algebra (depending on the type of the i-th segment in A), s. t. y_{i0} and y_{im} are evaluated by 0 and 1, respectively. [2] presents an NP algorithm which performs this check for L-segments, so it remains to show how to perform it for G-segments and for Π-segments.

Observation 1 *The solvability of the above system of equations and sharp inequalities in G can be checked in linear time (w. r. t. $|\varphi|$).*

For the product t-algebra we use the following lemma.

Lemma 3.1 *The abovementioned system of equations and sharp inequalities is solvable in* Π *iff it has a solution in an algebra of type* $\mathrm{L} \oplus \mathrm{L}$ *such that* y_{i0} *is* $0_{\mathrm{L} \oplus \mathrm{L}}$ *and* $y_{i1}, \ldots, y_{im}$ *are evaluated in* $(h, 1]$*, where* h *is the non-extremal cutpoint.*

Proof: Follows from the isomorphism of the cut product algebra with L. An $m+1$-tuple $0 = a_0 < \cdots < a_m = 1$ is a solution in Π iff, introducing a cut c so that $a_0 < c < a_1$ and using an isomorphism g to map $a_1, \ldots, a_m$ into $(h, 1]$ in $\mathrm{L} \oplus \mathrm{L}$, $0_{\mathrm{L} \oplus \mathrm{L}}$ together with $g(a_1), \ldots, g(a_m)$ form a solution in $\mathrm{L} \oplus \mathrm{L}$. QED

Thus, to check solvability in the product t-algebra, we first eliminate all equations involving y_{i0}; the soundness of any such equation can be, and indeed has been in part, checked "externally"; for the remaining cases, check, for any u, v belonging to i, that if $u * v = y_{i0}$ then either u or v is y_{i0}, that if $u \Rightarrow v = y_{i0}$ (and $u > v$) then v is y_{i0}, and that $u \Rightarrow y_{i0} = y_{i0}$. Then we consider the remaining equations and sharp inequalities in L, introducing a new inequality $0 < y_{i1}$, and check solvability of this system of equations and inequalities using the NP algorithm for solvability in L, referred to in [2].

Finally, it is obvious from the construction of the algorithm that the output is 'yes' (on at least one branch) iff the formula φ has a counterexample evaluation in A, i. e., is not an A-tautology. Thus the set of A-tautologies is in coNP.

4. Infinite ordinal sums

It is known ([1]) that a t-algebra is BL-generic iff it is an ordinal sum starting with an L and with infinitely many copies of L. Since the tautologies of BL are coNP complete, so are the tautologies of each BL-generic t-algebra.

Also, it is easy to see that t-algebras which are ordinal sums not starting with an L and having infinitely many copies of L are SBL-generic. To follow this observation, recall that a counterexample evaluation in Π can be locally embedded into L⊕L. Now let A be a t-algebra with infinitely many copies of L, not starting with an L. Assume φ is not an SBL-tautology, and let B be an SBL-algebra in which φ does not hold. We may assume that B is a finite sum of L's and Π's only (thus starting with a Π). Then the counterexample evaluation can be locally embedded in A, mapping the initial Π segment of B to any two L-segments of A (not necessarily adjacent), each of the following L-segments of B to arbitrary L segments of A, and each of the following Π-segments of B to any two L-segments of A, all in increasing order w. r. t. the ordering of the intervals in $[0, 1]$.

Theorem 4.1 *The propositional logic SBL is coNP complete.*

Proof: If φ is not an SBL-tautology, then it has a counterexample in a finite ordinal sum whose first element is not an L. Thus we may modify our algorithm by prefixing steps guessing the cardinality of the sum and its type. Let k be the number of propositional variables in φ.

`// algorithm for SBL`

$\{$

`guessCardinality()` Pick at random a natural n, $0 < n \leq k + 1$.

Lemma 4.2 *Let k be the number of propositional variables in a formula φ. If φ has an evaluation $e(\varphi) < 1$ in any t-algebra, then it has an evaluation $e'(\varphi) < 1$ in a t-algebra which is an ordinal sum with cardinality at most $k + 1$.*[1]

`guessLayout()` Assign to each $i = 1, \ldots, n$ one of the symbols L, G, Π, signifying the type of the i-th segment of the sum, in such a way that the first symbol is not an L. We use the term 'constructed sum' and the symbol C to denote this finite sum.

`cutpointVariables()`

`intervalVariables()`

`guessAssignment()`

`checkExternal()`

`checkInternal()`

$\}$

This modification is an NP algorithm recognizing SBL counterexamples, so the propositional tautology problem for the logic SBL is in coNP. QED

It remains to discuss the complexity of tautologies of an arbitrary infinite ordinal sum with only finitely many (possibly no) copies of L.

Fix such an algebra A, denote p the number of its L-segments, and define its representation S^A: a finite sequence of length $p + 1$, each element $S^A[i]$ determining the type of the subsum between two consecutive L-segments ($S^A[0]$ before the first L-segment and $S^A[p]$ after the last L-segment in A). $S^A[i], i = 0, \ldots, p$ is one of the following:

- $\emptyset$ if the subsum is void;

[1] This is just a variant of a similar result in [2].

- ∞ if there are infinitely many G-segments (thus there is an infinite alternating subsum of G's and Π's);

- (for finite number p_i of G-segments) a sequence $S^A[i]$ of length $p_i + 1$, determining the number of Π-segments between each two consecutive G-segments (also before the first and after the last G-segment). The j-th element of the sequence, $j = 0, \ldots, p_i$ is a natural number in the range $[0, \infty]$.

This is a handy finite representation of A. Note that using S^A, we may introduce indices for the segments of A in the following way:

- Any L-segment is uniquely determined by a natural number in the range $[1, p]$.

- A G-segment is either determined by a tuple of natural numbers $\langle i_1, i_2 \rangle, i_1 \in [0, p], i_2 \in [1, p_{i_1}]$, if it is the i_2-th G-segment after the i_1-th L-segment in A, where $S^A[i_1]$ is not ∞; or, if $S^A[i_1], i_1 = 0, \ldots, p$ is ∞, the G-segments in the i_1-th subsum may be for our purposes referred to by a tuple $\langle i_1, \mathtt{ANY} \rangle$.

- A Π-segment is either determined by a triple $\langle i_1, i_2, i_3 \rangle, i_1 \in [0, p], i_2 \in [0, p_{i_1}], i_3 \in N$, if it is the i_3-th Π-segment after the i_2-th G-segment after the i_1-th L-segment in A, where $S^A[i_1]$ is not ∞ and $S^A[i_1][i_2]$ is not ∞; or, if $S^A[i_1]$ is not ∞ but $S^A[i_1][i_2]$ is ∞, $i_1 = 0, \ldots, p, i_2 = 0, \ldots, p_{i_1}$, all the Π-segments in the subsum of Π's after the i_2-th G-segment after the i_1-th L-segment in A may be referred to by a triple $\langle i_1, i_2, \mathtt{ANY} \rangle$; or, if $S^A[i_1]$ is ∞, $i_1 = 0, \ldots, p$, all the Π-segments in the i_1-th subsum may be referred to by a tuple $\langle i_1, \mathtt{ANY} \rangle$.

We shall now present an NP algorithm recognizing counterexamples in A. As before, let the input formula φ be given, k be the number of its variables, and $m = 2|\varphi|$.

`// algorithm for infinite sum A`

{

`guessCardinality()` Pick at random a natural n, $0 < n \leq k + 1$.

`guessLayout()` Assign to each $i = 1, \ldots, n$ one of the symbols L, G, Π, signifying the type of the i-th segment of the sum.

We use the term 'constructed sum' and the symbol C to denote this finite sum.

`// from now on the algorithm works with` C

`checkEmbedding()` Check whether the constructed sum is $1-1$ embeddable into A (as a sequence of symbols into a sequence of symbols), in such a way that a potential initial L of the constructed sum is mapped to an initial L in A. It is vital that initial L remains initial in A, since otherwise a counterexample in the constructed sum need not be a counterexample in A.

`cutpointVariables()`

`intervalVariables()`

`guessAssignment()`

`checkExternal()`

`checkInternal()`

}

We discuss in more detail why the `checkEmbedding()` step does not violate the NP nature of the algorithm.

Lemma 4.3 *The embeddability of the constructed sum C into A can be checked by an NP algorithm (w. r. t. the length n of C).*

Proof: The (nondeterministic) algorithm constructs the embedding a by assigning to each segment of C an index of its image in A, using the abovedescribed indices.

Denote max the maximum of the numbers $p, p_0, \ldots, p_p, n$. This number is the maximum natural number that can occur in any index guessed by the algorithm. Note that this number is independent of the input C. (Although some Π-segments could have indices with arbitrarily high numbers (as the third element), we use n as an upper bound, since C has the cardinality n, thus a suitable embedding can be always found in an initial n-segment fragment of the infinite subsum.)

First the algorithm guesses an index for each segment of C: indices of L-segments are natural numbers; indices of G-segments are tuples, the first element of which is a natural number and the second element is a natural number or the symbol ANY; indices of Π-segments are either tuples, consisting of a natural number and the symbol ANY, or triples, the first and second element of which are natural numbers and the third element is a natural number or the symbol ANY. Any number occurring in any index must be within $[0, max]$.

Subsequently the algorithm performs two checks, to find out whether there are segments in A referred to by the indices (this is checked using S^A) and whether the assignment of indices is 1-1 and increasing (w. r. t. the ordering of segments in C and in A). Both these checks can be performed in polynomial time (the detailed proof is omitted), thus the algorithm is NP. QED

Again, it is clear that the output of the algorithm is 'yes' (on at least one branch) iff the formula φ is not an A-tautology, thus A-tautologies are in coNP.

5. Thanks

Partial support from the grant IAA 103 0004, Grant Agency of the Academy of Sciences, Czech Republic, is acknowledged.

6. PhD thesis

The author is a PhD student of mathematical logic at the Faculty of Mathematics and Physics, Charles University, since October 1999. The topic of her thesis is "Mathematical and metamathematical properties of fuzzy logic".

References

[1] P. Aglianò, F. Montagna, "Varieties of BL-algebras I: general properties", preprint, 2002.

[2] M. Baaz, P. Hájek, F. Montagna, H. Veith, "Complexity of t-tautologies", *Ann. Pure Appl. Logic 113*, No.1-3, pp. 2-11, 2001.

[3] R. Cignoli, F. Esteva, L. Godo, A. Torrens, "Basic Fuzzy Logic is the logic of continuous t-norms and their residua", *Soft Comput. 4*, pp. 106-112, 2000.

[4] P. Hájek, "Metamathematics of Fuzzy Logic", Kluwer Academic Publishers, Dordrecht, 1998.

[5] Z. Haniková, "A note on the complexity of propositional tautologies of individual t-algebras", to appear in NNW.

[6] P. S. Mostert, A. L. Shields, "On the structure of semigroups on a compact manifold with boundary", *Annals of Math. 65*, pp. 117-143, 1957.

[7] D. Mundici, "Satisfiability in many-valued sentential logic is NP-complete", *Theor. Comput. Sci. 52,* pp. 145-153, 1987.