A note on the complexity of propositional tautologies of individual t-algebras

Zuzana Haniková
Institute of Computer Science
182 07 Prague, Czech Republic
zuzana@cs.cas.cz

September 12, 2002

Abstract

For any t-algebra, the set of its propositional tautologies is coNP complete.

1 Introduction

This paper is a contribution to the system of many-valued logics in the style of [4]. There is a lot of reading material available on this topic, and we refrain from giving a lengthy review of it. The reader is kindly requested to use e. g. [4] for reference.

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 Łukasiewicz, 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 Ł-tautologies comes from [6]).

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

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

This comes from [2] for t-algebras starting with an Ł (proved via reduction of propositional Ł-tautologies, prefixing a negation to each propositional variable) and from [4] for t-algebras not starting with an Ł (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 [5], 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 Łukasiewicz, Gödel, or product t-norm (on $[0,1]$). For this reason each t-algebra is an ordinal sum of copies of Łukasiewicz, Gödel, and product algebras, which we habitually call *segments* and denote with symbols Ł, G, and Π. We stress that each copy of Gödel counts as one segment, thus, e. g., $[0,1]_{\text{Ł}\oplus\text{G}\oplus\Pi}$ is a t-algebra with three segments, namely a sum of a copy of the Łukasiewicz algebra, a copy of the Gödel algebra and a copy of the product algebra; the type of the sum is Ł$\oplus$G$\oplus\Pi$. We disregard the exact positioning of the set I within $[0,1]$.

2 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 Ł-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 $z_0 < \ldots < z_n$ 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 $z_i = y_{i0} < y_{i1} < \ldots < y_{im} = z_{i+1}$ 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 $y_{i0} < \ldots < y_{im}$, 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 Łukasiewicz, 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 Ł-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. $|\varphi|$).*

Proof: Obviously the system is solvable in the standard algebra given by Gödel's t-norm iff it is solvable in an $m+1$-element Gödel chain. It is enough therefore to consider the variables $y_{i0} < \ldots < y_{im}$ as an $m+1$-element Gödel chain and to check whether all the equations hold in it.

The number of equations to be checked is bounded by the number of subformulas of φ, which is at most $m = 2|\varphi|$. QED

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

Lemma 1 *The abovementioned system of equations and sharp inequalities is solvable in* Π *iff it has a solution in an algebra of type* Ł$\oplus$Ł *such that* y_{i0} *is* $0_{Ł\oplus Ł}$ *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 Ł. An $m+1$-tuple $0 = a_0 < \ldots < a_m = 1$ 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 Ł$\oplus$Ł, $0_{Ł\oplus Ł}$ together with $g(a_1), \ldots, g(a_m)$ form a solution in Ł$\oplus$Ł. 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 Ł, introducing a new inequality $0 < y_{i1}$, and check solvability of this system of equations and inequalities using the NP algorithm for solvability in Ł, referred to in [2].

The described check for product is in NP: for solvable systems of equations, a verification is just a solution of the system in $[0, 1]_{Ł\oplus Ł}$, i. e., a sequence of reals with suitably small denominators which evaluate all the variables.

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.

3 Infinite ordinal sums

It is known ([1]) that a t-algebra is BL-generic iff it is an ordinal sum starting with an Ł and with infinitely many copies of Ł. 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 Ł and having infinitely many copies of Ł are SBL-generic. To follow this observation, recall that a counterexample evaluation in Π can be locally embedded into Ł$\oplus$Ł. Now let A be a t-algebra with infinitely many copies of Ł, not starting with an Ł. 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 Ł'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 Ł-segments of A (not necessarily adjacent), each of the following Ł-segments of B to arbitrary

Ł segments of A, and each of the following Π-segments of B to any two Ł-segments of A, all in increasing order w. r. t. the ordering of the intervals in $[0,1]$.

Theorem 3 *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 Ł. 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 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]

Proof: Take a t-algebra with a $e(\varphi) < 1$. It is an ordinal sum with at least one segment. From the sum, retain those segments which evaluate at least one propositional variable of φ, and the first segment. If for some i, $e(p_i)$ is a cutpoint, retain the segment following this cutpoint. This yields an algebra with at most $k+1$ segments for which the formula φ evaluates (under evaluation obtained from e) to less than one. (Deleting the superfluous intervals requires care when there are cutpoints $u < v$ s. t. u evaluates a propositional variable p_1, and the nearest element evaluating a propositional variable p_2 is in a segment $[v, v']$ of type Ł. If we omitted the segment starting with the cutpoint u from the contracted sum, then $e(p_1)$ would become the starting cutpoint of the Ł-interval evaluating p_2, and the evaluation (of implication $p_2 \to p_1$ for example) might behave differently. This is also why we always retain the first segment.) QED

`guessLayout()` Assign to each $i = 1, \ldots, n$ one of the symbols Ł, G, Π, signifying the type of the i-th segment of the sum, in such a way that the first symbol is not an Ł.

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

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

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

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

Fix such an algebra A, denote p the number of its Ł-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 Ł-segments ($S^A[0]$ before the first Ł-segment and $S^A[p]$ after the last Ł-segment in A). $S^A[i], i = 0, \ldots, p$ is one of the following:

– $\emptyset$ if the subsum is void;

– ∞ 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 Ł-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 Ł-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 Ł-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 Ł-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 Ł, 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 Ł of the constructed sum is mapped to an initial Ł in A. It is vital that initial Ł 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 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 Ł-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). We give a precise description of both checks so as to make it clear that they can be performed in polynomial time.

An Ł-segment with an index i exists iff $i \in [1, p]$; a G-segment with an index $\langle i_1, i_2 \rangle$ exists iff $i_1 \in [0, p]$, $S^A[i_1]$ is not ∞ and $i_2 \in [1, p_{i_1}]$; a G-segment with an index $\langle i_1, \mathtt{ANY} \rangle$ exists iff $i_1 \in [0, p]$ and $S^A[i_1]$ is ∞. A Π-segment with an index $\langle i_1, i_2, i_3 \rangle$ exists if $i_1 \in [0, p]$, $S^A[i_1]$ is not ∞, $i_2 \in [0, p_{i_1}]$, $S^A[i_1][i_2]$ is not ∞ and $i_3 \in [1, S^A[i_1][i_2]]$; a Π-segment with an index $\langle i_1, i_2, \mathtt{ANY} \rangle$ exists iff $i_1 \in [0, p]$, $S^A[i_1]$ is not ∞, $i_2 \in [0, p_{i_1}]$ and $S^A[i_1][i_2]$ is ∞; a Π-segment with an index $\langle i_1, \mathtt{ANY} \rangle$ exists iff $i_1 \in [0, p]$ and $S^A[i_1]$ is ∞. In fact, we check whether there is a segment in A which has been previously assigned the index in question (with the reservation that the index guessed must be sufficiently small).

To check that the assignment is 1-1 and increasing, it is enough to consider each pair of adjacent segments in C. For $i = 1, \ldots, n-1$ we check whether the index assigned to the i-th segment of C is lexicographically less than the index assigned to the $i+1$-th segment of C. The subroutine testing the lexicographical order and injective property of the assignment works as usual, progressing from left to right on the two compared strings and comparing numerical elements. We discuss evaluation of strings containing the symbol `ANY`. Note that at this stage we know that segments referred to exist in A, thus the symbol `ANY` occurs either in both indices, in the same position (the second or the third element), or in neither of them. If it occurs in the second position (and the first element is i), it means $S^A[i]$ is ∞, and as any finite sum of G's and Π's can be accommodated in the subsum given by $S^A[i]$, the return value is always 'yes'. If it occurs in the third position (and the first and second element are i_1, i_2, then it means $S^A[i_1][i_2]$ is ∞, thus any finite number of Π's can be accommodated in the subsum given by $S^A[i_1][i_2]$, and the return value is always 'yes'. Note that thanks to the previous existence check, this is an exhaustive list of configurations with the `ANY` symbol. QED

It is perhaps worth pointing out that under this construction, the algorithm "throws away" some of the constructed sums which actually do supply a counterexample evaluation embeddable into A as a sequence of reals, because the sum itself (as segments) is not embeddable into the sum of A. Nevertheless, if there is a counterexample in A at all, then the counterexample evaluation uses only finitely many (at most $k+1$) segments of A, and then this finite subsum appears on some branch of the computation.

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.

4 Acknowledgements

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

The author would like to thank Jirka Hanika and Petr Savický for enlightening her on some complexity issues and providing many useful comments.

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 (2002), 2-11.

[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 (2000), 106-112.

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

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

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