Scientific Issues
Jan Długosz University
in Częstochowa
Mathematics XXI (2016)
135–143
DOI http://dx.doi.org/10.16926/m.2016.21.12

FROM ARITHMETIC EXPRESSIONS TO PROPOSITIONAL FORMULAE

LIDIA STĘPIEŃ AND MARCIN R. STĘPIEŃ

Abstract

In papers [3], [4], [5] Authors presented a new method of solving some kinds of computational tasks in the area of linear algebra by applying SAT-solver as the highly optimized algorithms for solving the problem of propositional satisfiability. On input SAT-solver (cf. [1], [2]) takes a propositional formula in the clause form. In this paper we show in detail how any arithmetical expression can be translated into propositional formula in the CNF form skipping out its traditional form. For this, we define the notion of consistency of arithmetic and boolean valuations.

1. Background and notations

In [3], [4], [5] it was proposed a new method of declarative programming in the area of linear algebra. To solve a problem, the programmer determines the constraints of an object and leaves searching for this object to a computer. In this new method all algebraic conditions and properties are represented by propositional formulas in such a way that satisfying valuations represent the problem. The task of finding a solution and all calculations are left to a computer equipped with highly optimized algorithms called SAT-testers.

In this work we present a direct translation of an algebraic expression (which describes some algebraic problems) to a propositional formula in the clause form. We consider arithmetic terms $t, u, \ldots$ constructed by means of variables $a, b, \ldots$, operators $+$, $\cdot$ and constants $\mathtt{0}$ and $\mathtt{1}$. We denote the countable set of arithmetic variables by $\mathcal{PW}$, the set of arithmetic expressions by $\mathcal{W}$, the 2-element field with standard operations $+_2$ and $\cdot_2$ by $\mathbb{F}_2 = (\{0,1\}, +_2, \cdot_2)$ and by $w\colon \mathcal{W} \to \{0,1\}$ – an arithmetic valuation such that

- $w(t_1 + t_2) = 1$ iff $w(t_1) +_2 w(t_2) = 1$,
- $w(t_1 \cdot t_2) = 1$ iff $w(t_1) \cdot_2 w(t_2) = 1$.

We consider propositional formulae $\alpha, \beta, \ldots$ constructed by means of propositional variables $p, q, \ldots$, propositional operators $\oplus$, $\wedge$ and propositional constants $\mathbf{F}$, $\mathbf{T}$. We denote the set of propositional variables by $\mathcal{F}$, by $\mathcal{B}$ – a Boolean Algebra and by $v\colon \mathcal{F} \to \{\mathbf{0}, \mathbf{1}\}$ – a boolean valuation.

The paper is organized as follows. In the second section we introduce a new translation of any arithmetic term into a propositional formula and we define consistency of arithmetic and boolean valuations. In the third section we present how any arithmetic term can be translated into a propositional formula in the CNF form over its traditional form. The last section completes the paper with some conclusions.

2. Consistent valuations

Let $f_{\mathcal{F}}\colon \mathcal{W} \to \mathcal{F}$ be a function which translates any arithmetic term into a propositional formula such that:

$$f_{\mathcal{F}}(t) = \begin{cases} p_i, & \text{dla } t = a_i \in \mathcal{PW} \text{ i } i \in \mathbb{N}, \\ f_{\mathcal{F}}(t_1) \oplus f_{\mathcal{F}}(t_2), & \text{dla } t = t_1 + t_2, \\ f_{\mathcal{F}}(t_1) \wedge f_{\mathcal{F}}(t_2), & \text{dla } t = t_1 \cdot t_2. \end{cases} \tag{1}$$

In this natural translation the arithmetic variables are translated into the propositional variables, the constant $\mathbb{1}$ is translated into a propositional variable, the constant $\mathbb{0}$is translated into a negated propositional variable and, finally, the arithmetic operators $+$ and $\cdot$ are translated into the propositional disjunction $\oplus$ and conjunction $\wedge$, respectively. Notice that the function $f_{\mathcal{F}}$ is a well defined bijection due to commutative, associative and distributive properties of the set of arithmetic expressions and the set of propositional formulae, respectively. Moreover, the length of the output propositional formula is equal to the length of an input arithmetic expression t and the number of propositional variables of $f_{\mathcal{F}}(t)$ is equal to the number of arithmetic variables of t.

Example 1. *We transform the arithmetic term $t = a_1 \cdot a_3 + a_2$ to the propositional formula $\alpha = p_1 \wedge p_3 \oplus p_2$ by applying the function $f_{\mathcal{F}}$ and make the truth table.*

a_1	a_2	a_3	$a_1 \cdot a_3$	t	α	$p_1 \wedge p_3$	p_1	p_2	p_3
1	1	1	1	0	**0**	**1**	**1**	**1**	**1**
1	1	0	0	1	**1**	**0**	**1**	**1**	**0**
1	0	1	1	1	**1**	**1**	**1**	**0**	**1**
1	0	0	0	0	**0**	**0**	**1**	**0**	**0**
0	1	1	0	1	**1**	**0**	**0**	**1**	**1**
0	1	0	0	1	**1**	**0**	**0**	**1**	**0**
0	0	1	0	0	**0**	**0**	**0**	**0**	**1**
0	0	0	0	0	**0**	**0**	**0**	**0**	**0**

Definition 1 (Consistency of valuations)**.** We say that an arithmetic valuation w and a boolean valuation v are consistent iff $\forall_{a \in \mathcal{PW}}(w(a) = 1 \Leftrightarrow f_{\mathcal{F}}(a) = \mathbf{1})$.

An arithmetic valuation w and a boolean valuation v are *inconsistent* iff w and v are not consistent.

Lemma 1. *An arithmetic valuation w and a boolean valuation v are consistent iff*

$$\forall_{t \in \mathcal{W}}(w(t) = 1 \Leftrightarrow v(f_{\mathcal{F}}(t)) = \mathbf{1}).$$

Proof. Let $len(t)$ denote the length of an arithmetic term $t \in \mathcal{W}$ (i.e. the number of operators in t). The proof proceeds by the induction over the length of t.

A. For $len(t) = 0$, lemma is valid by definition 1.
B. Induction hypothesis: lemma is valid for $len(t) = n - 1$.
C. We show that for $len(t) = n$ lemma is valid.

   $w(t) = 1 \Leftrightarrow w(t_1 + t_2) = 1 \Leftrightarrow w(t_1) = 1$ or $w(t_2) = 1 \Leftrightarrow$ by induction hypothesis $v(f_{\mathcal{F}}(t_1)) = \mathbf{1}$ or $v(f_{\mathcal{F}}(t_2)) = \mathbf{1} \Leftrightarrow v(f_{\mathcal{F}}(t_1) \oplus f_{\mathcal{F}}(t_2)) = \mathbf{1} \Leftrightarrow v(f_{\mathcal{F}}(t)) = \mathbf{1}$

   $w(t) = 1 \Leftrightarrow w(t_1 \cdot t_2) = 1 \Leftrightarrow w(t_1) = 1$ and $w(t_2) = 1 \Leftrightarrow$ by induction hypothesis $v(f_{\mathcal{F}}(t_1)) = \mathbf{1}$ and $v(f_{\mathcal{F}}(t_2)) = \mathbf{1} \Leftrightarrow v(f_{\mathcal{F}}(t_1) \wedge f_{\mathcal{F}}(t_2)) = \mathbf{1} \Leftrightarrow v(f_{\mathcal{F}}(t)) = \mathbf{1}$

The arithmetic valuation w and boolean valuation v are consistent by definition 1 for $t = a \in \mathcal{PW}$. Hence the valuations v and w are consistent for t which is built with arithmetic variables. Finally, w and v are consistent for all subexpressions of t. □

3. Convertion to CNF formula

In this section we show how any arithmetic expression can be converted to a CNF formula. A standard translation produces the output propositional

formula in the traditional form and then this formula is converted into the CNF form by applying well known algorithm. In our translation we omit the traditional form of a propositional formula by direct encoding arithmetic operations by correspondent logical connectives.

Let $t \in \mathcal{W}$, $f_{\mathcal{F}}: \mathcal{W} \to \mathcal{F}$ (see 1). The function $toCNF()$ produces the CNF formula of a polynomial length relative to the length of t. Let $\mathcal{PV}(f_{\mathcal{F}}(t))$ be the set of propositional variables occuring in formula $f_{\mathcal{F}}(t)$. The function $toCNF()$ expands the set of propositional variables by new, other propositional variables from the set $\mathcal{PV}^C(f_{\mathcal{F}}(t)) \subseteq \mathcal{PV}$ of labels of subformulas $f_{\mathcal{F}}(t)$. It means, that a literal $l_{t'} \in \mathcal{PV}^C(f_{\mathcal{F}}(t))$ represents a subformula $f_{\mathcal{F}}(t')$ of the propositional formula $f_{\mathcal{F}}(t)$, which codes subexpression t' of expression t. In order to standarize notation if $p \in \mathcal{PV}(f_{\mathcal{F}}(t))$ we denote propositional variable p by l_p, but we don't introduce a new literal.

We define the function $f : \mathcal{W} \to 2^C$ which transforms any arithmetic expresion into the set of clauses as follows:

$$(2)\ f(t) = \begin{cases} \mathbf{T}, & \text{for } t = a, \\ (l_{t_1} \vee \neg l_t) \wedge (l_{t_2} \vee \neg l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee l_t), & \text{for } t = t_1 \cdot t_2, \\ (l_{t_1} \vee l_{t_2} \vee \neg l_t) \wedge (\neg l_{t_1} \vee l_{t_2} \vee l_t) \wedge \\ (l_{t_1} \vee \neg l_{t_2} \vee l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t), & \text{for } t = t_1 + t_2. \end{cases}$$

If t is an arithmetic variable, the function f produces the formula $\mathbf{T}$ in the conjunctive normal form. It is a tautology and represents the empty clause set $\emptyset$.

Now, by applying f we define a function $toCNF : \mathcal{W} \to 2^C$ by:

$$(3)\ toCNF(t) = \begin{cases} \mathbf{T}, & \text{for } t \in \mathcal{PW}, \\ toCNF(t_1) \wedge toCNF(t_2) \wedge f(t), & \text{for } t = t_1 \diamondsuit t_2 \text{ i} \\ & \diamondsuit \in \{+, \cdot\}. \end{cases}$$

By length of a propositional formula $toCNF(t)$ in the clause form, which will be denoted by dl, we mean the cardinality of its set of clauses.

Lemma 2. *The output of the transformation $toCNF()$ is of the polynomial length relative to the length of the input arithmetic term t.*

Proof. The proof proceeds by the induction on length of an input arithmetic term t. Below we present the proof for t built with $+$. If t is constructed with $\cdot$, the reasoning is analogous.

1. If $len(t_0) = 0$, then $t_0 = a$ and $dl(toCNF(t_0)) = 0$.
2. If $len(t_1) = 1$, then $t_1 = a_1 + a_2$ and

   $dl(toCNF(t_1)) = dl(toCNF(a_1)) + dl(toCNF(a_2)) + 4 = 0 + 0 + 4 = 4 \cdot 1 = 4 \cdot len(t)$.

3. If $len(t_n) = n$, then $t = t_{n-1} + a_{n+1}$ and

$$dl(toCNF(t)) = dl(toCNF(t_{n-1})) + 0 + 4 =$$
$$= (dl(toCNF(t_{n-2}) + 0 + 4) + 4 = \ldots =$$
$$= \underbrace{(((dl(toCNF(t_0)) + 0 + 4) + 4) + \ldots + 4)}_{n-1} + 4 =$$
$$= \underbrace{(4 + 4 + \ldots + 4)}_{n-1} + 4 = 4 \cdot (n-1) + 4 = 4 \cdot n =$$
$$4 \cdot len(t_n)$$

If t is built with $\cdot$, then by analogous calculation we get

$$dl(toCNF(t)) = 3 \cdot len(t).$$

Generally, for any arithmetic expression t,

$$dl(toCNF(t)) \leqslant 4 \cdot len(t). \qquad \square$$

Example 2.

1. *If* $t = a \in \mathcal{PW}$ *then* $toCNF(t) = \mathbf{T}$
2. *Let* $t = \underbrace{a_1 \cdot a_4}_{t_1} + \underbrace{a_2 \cdot a_3}_{t_2}$. *Then*

$$toCNF(t) = toCNF(t_1) \wedge\ toCNF(t_2)\ \wedge\ f(t) =$$
$$= toCNF(a_1)\ \wedge\ toCNF(a_4)\ \wedge\ f(t_1)\ \wedge$$
$$\wedge\ toCNF(a_2)\ \wedge\ toCNF(a_3)\ \wedge\ f(t_2)\ \wedge\ (l_{t_1} \vee l_{t_2} \vee \neg l_t) \wedge$$
$$\wedge\ (\neg l_{t_1} \vee l_{t_2} \vee l_t) \wedge (l_{t_1} \vee \neg l_{t_2} \vee l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t) =$$
$$=\ \mathbf{T} \wedge \mathbf{T} \wedge (l_{a_1} \vee \neg l_{t_1}) \wedge (l_{a_4} \vee \neg l_{t_1}) \wedge (\neg l_{a_1} \vee \neg l_{a_4} \vee l_{t_1}) \wedge \mathbf{T} \wedge \mathbf{T} \wedge$$
$$\wedge\ (l_{a_2} \vee \neg l_{t_2}) \wedge (l_{a_3} \vee \neg l_{t_2})\ \wedge (\neg l_{a_2} \vee \neg l_{a_3} \vee l_{t_2}) \wedge (l_{t_1} \vee l_{t_2} \vee \neg l_t) \wedge$$
$$\wedge\ (\neg l_{t_1} \vee l_{t_2} \vee l_t) \wedge (l_{t_1} \vee \neg l_{t_2} \vee l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t)$$

Finally:

$$\{(l_{a_1}, \neg l_{t_1}), (l_{a_4}, \neg l_{t_1}), (\neg l_{a_1}, \neg l_{a_4}, l_{t_1}), (l_{a_2}, \neg l_{t_2}), (l_{a_3}, \neg l_{t_2}),$$
$$(\neg l_{a_2}, \neg l_{a_3}, l_{t_2}), (\neg l_{a_2}, \neg l_{a_3}, l_{t_2}), (l_{t_1}, l_{t_2}, \neg l_t), (\neg l_{t_1}, l_{t_2}, l_t),$$
$$(l_{t_1}, \neg l_{t_2}, l_t), (\neg l_{t_1}, \neg l_{t_2}, \neg l_t)\}$$

For any arithmetic term t the transformation $toCNF()$ outputs at most 4-clausal set of at most 3-literals. Hence, the output propositional formula is of polynomial length relative to the length of t. Moreover, all literals corresponding to respective subexpressions have the logical value consistent with arithmetic value of subexpressions.

Definition 1. For any arithmetic term $t \in \mathcal{W}$, we say that boolean valuation v and arithmetic valuation w are *t-consistent* with iff $w(t') = 1 \Leftrightarrow v(l_{t'}) = \mathbf{1}$, for each subexpression t' of the expression t.

Lemma 3. *Let $t \in \mathcal{W}$ and w be any arithmetic valuation. For each boolean valuation v, $v(toCNF(t))=\mathbf{1}$ iff v and w are t-consistent.*

Proof. The proof proceeds by the induction on the stucture of the arithmetic term t.

$\Rightarrow$ Assume that $v(toCNF(t))=\mathbf{1}$. We show that v and w are t-consistent.

A. For $t = a \in \mathcal{PW}$, $toCNF(t) = \mathbf{T}$. Hence, $v(toCNF(t)) = \mathbf{1}$ for any boolean valuation v. Since l_t is the only standarized notation of a propositional variable $p_t = f_{\mathcal{F}}(t)$, v and w are t-consistent by definition 1.

B. For $t=t_1 \cdot t_2$. Assume that $v(toCNF(t_1 \cdot t_2)) = \mathbf{1}$ and we show that v and w are $(t_1 \cdot t_2)$-consistent w.
Since $toCNF(t_1 \cdot t_2) = toCNF(t_1) \wedge toCNF(t_2) \wedge (l_{t_1} \vee \neg l_t) \wedge$
$\wedge (l_{t_2} \vee \neg l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee l_t)$, then $v(toCNF(t_1)) = \mathbf{1}$ (by induction hypothesis v and w are t_1-consistent), $v(toCNF(t_2)) = \mathbf{1}$ (by induction hypothesis v and w are t_2-consistent) and all clauses are true.

1. If $w(t_1) = 0$ and $w(t_2) = 0$, then $w(t) = 0$, $v(l_{t_1}) = \mathbf{0}$ and $v(l_{t_2}) = \mathbf{0}$. Therefore $v(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$. Since $v(l_{t_1} \vee \neg l_t) = \mathbf{1}$ i $v(l_{t_2} \vee \neg l_t) = \mathbf{1}$ then $v(\neg l_t) = \mathbf{1}$. Hence $v(l_t) = \mathbf{0}$.

2. Assume that $w(t_1) = 1$ and $w(t_2) = 0$. Hence $w(t) = 0$, $v(l_{t_1}) = \mathbf{1}$ i $v(l_{t_2}) = \mathbf{0}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ and $v(l_{t_1} \vee \neg l_t) = \mathbf{1}$. Since $v(l_{t_2} \vee \neg l_t) = \mathbf{1}$ then $v(\neg l_t) = \mathbf{1}$. Hence $v(l_t) = \mathbf{0}$.

3. Assume that $w(t_1) = 0$ and $w(t_2) = 1$. Hence $w(t) = 0$, $v(l_{t_1}) = \mathbf{0}$ and $v(l_{t_2}) = \mathbf{1}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ and $v(l_{t_2} \vee \neg l_t) = \mathbf{1}$. Since $v(l_{t_1} \vee \neg l_t) = \mathbf{1}$ then $v(\neg l_t) = \mathbf{1}$. Hence $v(l_t) = \mathbf{0}$.

4. Suppose that $w(t_1) = 1$ and $w(t_2) = 1$. Hence $v(l_{t_1}) = \mathbf{1}$ and $v(l_{t_2}) = \mathbf{1}$ and $v(l_{t_1} \vee \neg l_t) = \mathbf{1}$ and $v(l_{t_2} \vee \neg l_t) = \mathbf{1}$. Since $v(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ then $v(l_t) = \mathbf{1}$.

So we have shown that v and w are $(t_1 \cdot t_2)$-consistent.

C. Let $t=t_1+t_2$. Assume that $v(toCNF(t_1+t_2))=\mathbf{1}$ and we will show that v and w are (t_1+t_2)-consistent.
Since $toCNF(t_1+t_2) = toCNF(t_1) \wedge toCNF(t_2) \wedge (l_{t_1} \vee l_{t_2} \vee \neg l_t) \wedge$
$(\neg l_{t_1} \vee l_{t_2} \vee l_t) \wedge (l_{t_1} \vee \neg l_{t_2} \vee l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t)$ then by induction hypothesis $v(toCNF(t_1)) = \mathbf{1}$ (by induction hypothesis v and w are t_1-consistent), $v(toCNF(t_2)) = \mathbf{1}$ (by induction hypothesis v and w are t_2-consistent) and all clauses are true for v.

1 Assume that $w(t_1)=0$ and $w(t_2)=0$. Hence $w(t)=0$, $v(l_{t_1})=\mathbf{0}$, because v and w are t_1-consistent, and $v(l_{t_2})=\mathbf{0}$, because v and w are t_2-consistent and $v(\neg l_{t_1} \vee l_{t_2} \vee l_t) = \mathbf{1}$, $v(l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t) = \mathbf{1}$. Since $v(l_{t_1} \vee l_{t_2} \vee \neg l_t) = \mathbf{1}$ then $v(\neg l_t) = \mathbf{1}$. Hence $v(l_t) = \mathbf{0}$.

2 Assume that $w(t_1)=1$ and $w(t_2)=0$. Hence $w(t)=1$, $v(l_{t_1})=\mathbf{1}$, because v and w are t_1-consistent, and $v(l_{t_2})=\mathbf{0}$, because v and w are t_2-consistent and $v(l_{t_1} \vee l_{t_2} \vee \neg l_t) = \mathbf{1}$, $v(l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t) = \mathbf{1}$. Since $v(\neg l_{t_1} \vee l_{t_2} \vee l_t) = \mathbf{1}$ then $v(l_t) = \mathbf{1}$.

3 Assume that $w(t_1) = 0$ and $w(t_2) = 1$. Hence $w(t)=1$, $v(l_{t_1}) = \mathbf{0}$, because v w are t_1-consistent, $v(l_{t_2}) = \mathbf{1}$, because v and w are t_2-consistent, and $v(l_{t_1} \vee l_{t_2} \vee \neg l_t) = \mathbf{1}$, $v(\neg l_{t_1} \vee l_{t_2} \vee l_t) = \mathbf{1}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t) = \mathbf{1}$. Since $v(l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ then $v(l_t) = \mathbf{1}$.

4 Assume that $w(t_1) = 1$ and $w(t_2) = 1$. Hence $v(l_{t_1}) = \mathbf{1}$, because v is t_1-consistent and $v(l_{t_2}) = \mathbf{1}$, because v is t_2-consistent and $v(l_{t_1} \vee l_{t_2} \vee \neg l_t) = \mathbf{1}$, $v(\neg l_{t_1} \vee l_{t_2} \vee l_t) = \mathbf{1}$ and $v(l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$ Since $v(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t) = \mathbf{1}$ then $v(\neg l_t) = \mathbf{1}$. Hence $v(l_t) = \mathbf{0}$.

So we have shown that v and w are $(t_1 + t_2)$-consistent.

($\Leftarrow$) Assume that v and w are t-consistent and we are going to show that $v(toCNF(t)) = \mathbf{1}$.

A. If t is propositional variable, then boolean valuation v and the arithmetic valuation w are t-consistent and $toCNF(t) = \mathbf{T}$. Therefore $v(toCNF(t)) = \mathbf{1}$ for any boolean valuation v.

B. Let $t = t_1 \cdot t_2$. Assume that v and w are $(t_1 \cdot t_2)$-consistent and we show that $v(toCNF(t_1 \cdot t_2)) = \mathbf{1}$.

Since $toCNF(t_1 \cdot t_2) = toCNF(t_1) \wedge toCNF(t_2) \wedge (l_{t_1} \vee \neg l_t) \wedge$ $\wedge (l_{t_2} \vee \neg l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee l_t)$ then, by induction hypothesis, v and w are t_1-consistent and t_2-consistents. Hence $v(toCNF(t_1)) = \mathbf{1}$ and $v(toCNF(t_2)) = \mathbf{1}$. Now, we have to show that all clauses are true for the boolean valuation v. By assumption that v and w are $(t_1 \cdot t_2)$-consistent, we have that $w(t)$ is consistent with $v(l_t)$.

1. Let $w(t) = 0$. Hence $v(l_t) = \mathbf{0}$ and clauses $(l_{t_1} \vee \neg l_t)$ and $(l_{t_2} \vee \neg l_t)$ are true for the boolean valuation v. Now, we are going to determine the value of the clause $(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t)$ for v. Since $v(l_t) = \mathbf{0}$, then the value of disjunction depends on its summands and is equivalent to its disjunction. Hence, by $w(t_1) = 0$ or $w(t_2) = 0$ $(w(t_1 \cdot t_2) = 0)$ it follows $v(l_{t_1}) = \mathbf{0}$ or $v(l_{t_2}) = \mathbf{0}$ and $v(\neg l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$. Finally, $v(toCNF(t_1 \cdot t_2)) = \mathbf{1}$.

2. Assume that $w(t) = 1$. Then $v(l_t) = \mathbf{1}$ and the clause $(\neg l_{t_1} \vee \neg t_2 \vee L_t)$ is true for the boolean valuation v. Since $w(t_1) = 1$ and $w(t_2) = 1$ and $v(l_{t_1}) = \mathbf{1}$ and $v(l_{t_2}) = \mathbf{1}$, then clauses $(l_{t_1} \vee \neg l_t)$ and $(l_{t_2} \vee \neg l_t)$ are also true by assumption $w(t_1 \cdot t_2) = 1$. Finally, $v(toCNF(t_1 \cdot t_2)) = \mathbf{1}$.

We have shown that since v and w are $(t_1 \cdot t_2)$-consistent, then $v(toCNF(t_1 \cdot t_2)) = \mathbf{1}$.

C. Let $t = t_1 + t_2$. Assume that v and w are $(t_1 + t_2)$-consistent. We show that $v(toCNF(t_1 + t_2) = \mathbf{1}$.
Since $toCNF(t_1 + t_2) = toCNF(t_1) \wedge toCNF(t_2) \wedge (l_{t_1} \vee l_{t_2} \vee \neg l_t) \wedge (\neg l_{t_1} \vee l_{t_2} \vee l_t) \wedge (l_{t_1} \vee \neg l_{t_2} \vee l_t) \wedge (\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t)$ hence, by induction hypothesis, if v and w are t_1-consistent, then $v(toCNF(t_1)) = \mathbf{1}$ and if v and w are t_2-consistent, then $v(toCNF(t_2)) = \mathbf{1}$. We have to show that all clauses are true for v. By assumption, if v and w are $(t_1 + t_2)$-consistent then $w(t)$ is consistent with $v(l_t)$.

1. Let $w(t_1 + t_2) = 0$. Then $v(l_t) = \mathbf{0}$ and clauses $(l_{t_1} \vee l_{t_2} \vee \neg l_t)$ and $(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t)$ are true for v. Now we determine value of $(\neg l_{t_1} \vee l_{t_2} \vee l_t)$ and $(l_{t_1} \vee \neg l_{t_2} \vee l_t)$ for v. Since $v(l_t) = \mathbf{0}$, value of cluses depends on its summands and is equivalent to its disjunction. Since $w(t_1) = 0$ and $w(t_2) = 0$ or $w(t_1) = 1$ and $w(t_2) = 1$ then $v(l_{t_1}) = \mathbf{0}$ and $v(l_{t_2}) = \mathbf{0}$ or $v(l_{t_1}) = \mathbf{1}$ and $v(l_{t_2}) = \mathbf{1}$. Hence $v(\neg l_{t_1} \vee l_{t_2} \vee l_t) = \mathbf{1}$ and $v(l_{t_1} \vee \neg l_{t_2} \vee l_t) = \mathbf{1}$. Finally $v(toCNF(t_1 + t_2)) = \mathbf{1}$.

2. Assume that $w(t_1 + t_2) = 1$. Then $v(l_t) = \mathbf{1}$ and $(\neg l_{t_1} \vee l_{t_2} \vee l_t)$ and $(l_{t_1} \vee \neg l_{t_2} \vee l_t)$ are true for v. Since $w(t_1 + t_2) = 1$, then $w(t_1) = 1$ and $w(t_2) = 0$ or $w(t_1) = 0$ and $w(t_2) = 1$ and $v(l_{t_1}) = \mathbf{1}$ and $v(l_{t_2}) = \mathbf{0}$ or $v(l_{t_1}) = \mathbf{0}$ and $v(l_{t_2}) = \mathbf{1}$ because v and w are $(t_1 + t_2)$-consistent. Hence $(l_{t_1} \vee l_{t_2} \vee \neg l_t)$ and $(\neg l_{t_1} \vee \neg l_{t_2} \vee \neg l_t)$ are true for v. Finally, $v(toCNF(t_1 + t_2)) = \mathbf{1}$.

By assumption that v and w are $(\alpha_1 \oplus \alpha_2)$-consistent, $v(toCNF(\alpha_1 \oplus \alpha_2)) = \mathbf{1}$.

. □

Theorem 1. *Let $t \in \mathcal{W}$ be any arithmetic expression. For every boolean valuation v and for every arithmetic valuation w if v and w are $t-$consistent, then $w(t) = 1$ iff $v(toCNF(t) \wedge l_t) = \mathbf{1}$.*

Proof. Since for every arithmetic term $t \in \mathcal{W}$ there exists a boolean valuation v which is t-consistent (by definition 1) with w, therefore, by lemma 3, $toCNF(t)$ is satisfiable. If $w(t) = 1$ and v and w are t-consistent, then $v(l_t) = \mathbf{1}$ and $v(toCNF(t) \wedge l_t) = \mathbf{1}$. If $w(t) = 0$, then l_t is false for any boolean valuation t-consistent with w. Otherwise $toCNF(t)$ is false if v and w are not t-consistent, hence $v(toCNF(t) \wedge l_t) = \mathbf{0}$. □

4. Conclusions

The translation of an arithmetic expession to a propositional formula can be executed with skipping out its traditional form. The solution presented above increases memory and time efficiency of the algorithm of automatic translation. It is very important for an automatic search of solutions of some algebraic problems as it was shown in [3], [4], [5] and [6].

References

[1] M. Davis, H. Putnam, *A Computing Procedure for Quantification Theory*, J. of the ACM 7(1) (1960), 201-215.

[2] M. Davis, G. Logemann, D.W. Loveland, *A Machine Program for Theorem Proving*, Comm. of the ACM 5(7) (1962), 394-397.

[3] M. Srebrny, L. Stępień, *A propositional programming environment for linear algebra*, Fundamenta Informaticae, 81 (2007), 325-345.

[4] M. Srebrny, L. Stępień, *SAT as a Programming Environment for Linear Algebra*, Fundamenta Informaticae, 102(1) (2010), 115-127.

[5] L. Stępień, Propositional calculus as a programming environment for linear algebra, PhD thesis (in polish), Inst. of Comp. Science, Polish Academy of Science, Warsaw, Poland, 2009.

[6] L. Stępień, M. R. Stępień, Automatic search of automorphisms of Witt rings, Scientific Issues. Mathematics, XVI (2011), 141-146, Jan Długosz University, Częstochowa.

Received: September 2016

Lidia Stępień
Jan Długosz University in Częstochowa,
Institute of Mathematics and Computer Science,
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
E-mail address: l.stepien@ajd.czest.pl

Marcin R. Stępień
Kielce University of Technology,
Department of Mathematics and Physics,
al. Tysiąclecia Państwa Polskiego 7, 25-314 Kielce, Poland
E-mail address: mstepien@tu.kielce.pl