A Bottom Set Strategy for Tractable Feature Construction

Filip Železný

Czech Institute of Technology (ČVUT) in Prague
zelezny@fel.cvut.cz

Abstract. Recent implementations of the *Extended Transformation Approach* (ETA) to first-order feature construction use powerful pruning mechanisms often enabling a very efficient search for features. This motivates a theoretical challenge to define families of feature languages which provably allow for efficient finding of their elements. The main result of this paper is that if a *bottom set* can be constructed of size polynomial in the preset maximum feature size n, containing literals of all features that comply to typing/moding declarations, then a correct feature can be found (or decided that no such feature exists) in time polynomial in n. This result is a consequence of the 'variable production - consumption' axiom previously established in the ETA on intuitive grounds, and thus provides a theoretical justification thereof. Preliminary results are included on conditions on which a) a polynomial bottom set can/cannot be constructed, b) all features can be efficiently enumerated.

Keywords: First-Order Logic Feature Construction

1 Introduction

This paper is concerned with efficient construction of expressions such as

```
hasCar(C),hasLoad(C,L1),small(L1),hasLoad(C,L2),big(L2)
```

corresponding to conjunctions of constant-free, function-free, non-negated Prolog atoms and subject to some predefined syntactic constraints. The expression is an example of a *feature* related to an *individual* (here a train) meaning that the train has a car carrying a small load and a big load. Note that a notation usual in ETA [3, 4] would rather have `hasCar(T,C)` as the first atom, mentioning the 'key variable' `T` linking to the train. Below I will discuss on the role of the key variable in the notation, but basically, for the feature construction purposes the key variable is a superfluous syntactic sugar. Let me define a 'literal' (here coincides with 'atom') and an expression (features will be searched among expressions) formally. I assume there are some infinite countable sets S ('predicate symbols') and V ('symbols of variables'). N stands for the set of naturals numbers.

Definition 1. *A* literal *is a sequence* $x = s_x, v_{x,1}, v_{x,2}, \ldots, v_{x,a_x}$ *such that* $s_x \in S$, $a_x \in N$ *and* $v_{x,i} \in V$ $(1 \leq i \leq a_x)$. $\mathbf{L}$ *denotes the set of all literals. Any finite* $e \subseteq \mathbf{L}$ *is called an* expression. *Denote* $Arg(e) = \{(x, i) \mid x \in e, 1 \leq i \leq a_x\}$ *and* $Var(e) = \{v_a \mid a \in Arg(e)\}$. $\mathbf{E}$ *denotes the set of all expressions.*

In illustrating examples to follow I will expose literals such as x above in the usual, Datalog-like form $s_x(v_{x,1}, v_{x,2}, \ldots, v_{x,a_x})$. Since I am not concerned with the procedure of *proving* the expressions (finding for which individuals they 'are true'), the literal order is irrelevant. This allows to view expressions simply as *sets* of literals. Note that the indexation in $v_{x,i}$ and a_x should be understood as a functional notation, ie. $v_{l,j}$ (v_{l,a_l}) will always represent the variable at the j-th (last, respectively) argument in literal l (a_l is called the *arity* of l). Also the concept of *substitution* acquires a simple meaning in this constrained framework.

Definition 2. *Let $e \in \mathbf{E}$. A* substitution *is a mapping $\theta : V \to V$. For $x \in e$, $x\theta = s_x, \theta(v_{x,1}), \theta(v_{x,2}), \ldots, \theta(v_{x,a_x})$ and $e\theta = \{x\theta \mid x \in e\}$. Expression e θ-*subsumes *expression f (denoted as $e \preceq_\theta f$) iff there is a substitution θ such that $e\theta \subseteq f$. Finally, e is* equivalent *to f (written $e \approx f$) iff θ is bijective and $e\theta = f$.*

The following definition introduces a *template*, analogous to typing and (input-output) moding declarations known from ILP systems such as Progol and Aleph, but again, abstracting from their irrelevant syntactic sugar. Further it defines when a variable is *proper*, ie. when it has both an input and an output role in an expression, as a means to install a variable 'production-consumption axiom' usually postulated in the ETA [3]. Lastly it defines which expression is a feature.

Definition 3. *A* template *τ is a pair (t_τ, M_τ) where t_τ is an expression and $M_\tau \subseteq Arg(t_\tau)$. $\mathbf{T}$ denotes the set of all templates. Let $e \preceq_\theta t_\tau$. Denote $Arg^+_\tau(e) = \{(x,i) \in Arg(e) \mid (x\theta, i) \in M_\tau\}$ and $Arg^-_\tau(e) = Arg(e) \setminus Arg^+_\tau(e)$. If some $v \in V$ satisfies the equivalence $(v = v_{a^+}, a^+ \in Arg^+_\tau(e)) \Leftrightarrow (v = v_{a^-}, a^- \in Arg^-_\tau(e))$, then v is said to be τ-*proper *in e . A non-empty expression f is a τ-*feature *iff $f \preceq_\theta t_\tau$ and each $v \in Var(f)$ is τ-proper in f.*

An example will clarify the above concepts. Consider a typical Aleph-like declaration {`hasCar(-c), hasLoad(+c,-l), small(+l), big(+l)`}. Lower cases in the brackets define *types* of arguments; a complying expression cannot have one variable at differently-typed arguments. The corresponding template (t_τ, M_τ) is

$$t_\tau = \texttt{hasCar}(\texttt{C}), \texttt{hasLoad}(\texttt{C}, \texttt{L}), \texttt{small}(\texttt{L}), \texttt{big}(\texttt{L})$$
$$M_\tau = \{(\texttt{hasLoad}(\texttt{C}, \texttt{L}), 1), (\texttt{small}(\texttt{L}), 1), (\texttt{big}(\texttt{L}), 1)\}$$

The typing constraint is here defined by the t_τ expression, isolated from the *moding* constraint. This makes it explicit that verifying compliance to a typing constraint corresponds to a subsumption check. Indeed, consider an expression

$$e = \texttt{hasCar}(\texttt{X}), \texttt{hasLoad}(\texttt{X}, \texttt{Y}), \texttt{small}(\texttt{Y}), \texttt{hasLoad}(\texttt{X}, \texttt{Z})$$

e complies to the typing specified by t_τ because $e \preceq_\theta t_\tau$.[1] The moding set M_τ contains the arguments labelled with '+' in the Aleph-like declaration. The 'production-consumption' axiom stipulates that each variable of a feature must

[1] In general, there may be more than one substitution θ such that $e \preceq_\theta t_\tau$. A straightforward way to avoid this ambiguity is to constraint oneself, quite naturally, to templates where t_τ contains each symbol $s \in S$ (such as `hasCar`) at most once.

occur at both an argument contained in M_τ and an argument not in M_τ. Therefore, e above is not a feature, since `Z` is at no argument in M_τ. Note that every template τ has a dual template τ^{-1} with inverse moding $M_\tau^{-1} = Arg(t_\tau) \setminus M_\tau$ and every τ-feature is also a τ^{-1}-feature. Through the inversion, every 'primary structural' (such as `hasCar(C)`) becomes a 'property' and every property (such as `small(L)`) becomes a primary structural. A theoretical consequence is that if a feature class (say features where literals chained by variable sharing form a 'tree') can be efficiently enumerated, then the inverse class (here literals forming a 'root') can also be efficiently enumerated. The symmetric properties commented above are a result of disregarding the key (individual-linking) variable, which would occur only at some 'input' arguments (those in M_τ). In a sense, the sole role of the key type lies in setting the *orientation* of the otherwise symmetric features, whereas the orientation is irrelevant for sakes of feature construction.

Note that also the notation (t_τ, M_τ) above is understood functionally, ie. for any template $\rho \in \mathbf{T}$, t_ρ represents the prescribed typing of ρ and M_ρ its moding. Let me now specify the main problem treated in the remainder of this paper.

Definition 4. *Let $T \subseteq \mathbf{T}$ and $E : T \to 2^{\mathbf{E}}$. The* feature existence problem *for T and E is defined as follows. The problem instance is the tuple $n \in N$, $\tau \in T$. The instance size is n. The instance solution is a τ-feature f such that $|f| \leq n$ and $f \approx f'$ for some $f' \in E(\tau)$ (if such f exists), or "NO" otherwise.*

The function E takes a template $\tau \in T$ and produces a set of expressions in which τ-features are searched (due to the $f \approx f' \in E(\tau)$ requirement, a solution may be not be in $E(\tau)$ but must be equivalent to some expression in $E(\tau)$; this avoids dependency on variable naming). The reason for specifying the problem class this way is that different complexity results can be proved for different functions E's. For example, $E(\tau)$ may consist of *connected* expressions (where all literals are linked via the transitive closure of the variable sharing relation) and then be independent of τ. Less trivially, $E(\tau)$ may be such that all τ-features therein are *loop-free* (edges between literals given by variable sharing, orientation given by moding) and thus obviously depend on the moding of the specific τ.

2 The Bottom Set Theorem

I will first show that the problem of selecting an expression out of a finite set of literals, such that a given variable is proper (has both an input and an output occurrence) in that expression, is equivalent to a problem of satisfying a set of propositional Horn clauses.

Lemma 1. *Let τ be a template, $e = \{l_1, l_2, \ldots, l_p\}$ and $e' \subseteq e$ two expressions, $P = \{P_1, P_2, \ldots P_p\}$ a set of propositional variables and $v \in Var(e)$. Further let*

$$\{in_1, in_2, \ldots, in_r\} = \{1 \leq in \leq p \mid \exists i \ (l_{in}, i) \in Arg_\tau^+(e), v_{l_{in},i} = v\} \quad (1)$$

$$\{out_1, out_2, \ldots, out_s\} = \{1 \leq out \leq p \mid \exists i \ (l_{out}, i) \in Arg_\tau^-(e), v_{l_{out},i} = v\} \quad (2)$$

be two index sets.[2] *Let further* $C_{in}(v)$ *denote the following set of Horn clauses*

$$P_{in_1} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \tag{3}$$
$$P_{in_2} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \tag{4}$$
$$\vdots \tag{5}$$
$$P_{in_r} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \tag{6}$$

Similarly, let $C_{out}(v)$ *be the following Horn clause set*

$$P_{out_1} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \tag{7}$$
$$P_{out_2} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \tag{8}$$
$$\vdots \tag{9}$$
$$P_{out_s} \vee \neg P_{in_1} \vee \neg P_{in_2} \vee \ldots \vee \neg P_{in_r} \tag{10}$$

Let $C(v) = C_{in}(v) \cup C_{out}(v)$ *and* $\xi_{e'} : P \rightarrow \{true, false\}$ *be a truth assignment*

$$\xi_{e'}(P_i) = \begin{cases} false, \text{ if } l_i \in e'; \\ true, \quad \text{if } l_i \notin e'. \end{cases} \tag{11}$$

Then v *is* τ*-proper in* e' *iff* $\xi_{e'}$ *satisfies all clauses in* $C(v)$*.*

Proof. (Sufficiency) Let $\xi_{e'}$ be an arbitrary truth assignment to the variables $P_1, P_2, \ldots, P_p$, such that $\xi_{e'}$ satisfies all clauses in $C(v)$. Let further $u_1, u_2, \ldots u_\mu$ $(1 \leq \mu \leq p)$ be the indexes of those of the variables which are assigned the *false* value by $\xi_{e'}$, ie. $e' = \{l_{u_1}, l_{u_2}, \ldots, l_{u_\mu}\}$. I need to show that $v = v_{a^+}, a^+ \in Arg^+_\tau(e')$ (v appears at an input argument) iff $v = v_{a^-}, a^- \in Arg^-_\tau(e')$ (v appears at an output argument). Let me first show the implication

v appears at an input argument $\Rightarrow$ v appears at an output argument

As the implication assumes, one of the literals $l_{in_1}, l_{in_2}, \ldots l_{in_r}$ containing the input occurrences of v (see Eq. 1) must be present in e', ie. there must be a κ $(1 \leq \kappa \leq r)$ such that $in_\kappa = u_k$. Then P_{in_κ} is assigned the *false* value by $\xi_{e'}$. Because all clauses of $C_{in}(v)$ must be satisfied by $\xi_{e'}$, so must be its κ-th clause

$$P_{in_\kappa} \vee \neg P_{out_1} \vee \neg P_{out_2} \vee \ldots \vee \neg P_{out_s} \tag{12}$$

Since P_{in_κ} is *false*, at least one of $P_{out_1}, P_{out_2}, \ldots P_{out_s}$ must hold the *false* value to keep the clause satisfied; let it be P_{out_λ} $(1 \leq \lambda \leq s)$. If P_{out_λ} is *false* then $l_{out_\lambda} \in e'$. But l_{out_λ} is a literal containing (see Eq. 2) an output occurrence of v. Thus I have proved the above implication. The inverse implication can be shown analogically, using the fact that all clauses in $C_{out}(v)$ must be satisfied.

(Necessity) Let e' $(e' \subseteq e)$ be an arbitrary expression in which v appears as both an input and an output (there are $a^+ \in Arg^+_\tau(e')$ and $a^- \in Arg^-_\tau(e')$

[2] The former index set thus addresses those of literals in e, which contain v at some input argument while the latter set corresponds to literals with v acting as an output.

such that $v = v_{a^+} = v_{a^-}$). Hence at least one of the literals $l_{out_1}, l_{out_1}, \ldots, l_{out_s}$ containing an output occurrence of v (see Eq. 2) must be present in e'; let it be l_{out_κ} $(1 \leq \kappa \leq s)$. Then $\xi_{e'}$ assigns the *false* value to P_{out_κ}. Since all clauses in $C_{in}(v)$ contain $\neg P_{out_\kappa}$, all clauses in $C_{in}(v)$ are satisfied. At the same time, e' must also contain at least one of the literals $l_{in_1}, l_{in_1}, \ldots, l_{in_r}$ where v is at an input argument (see Eq. 1). Let it be l_{in_λ} $(1 \leq \lambda \leq r)$. Then P_{in_λ} is *false* and all clauses in $C_{out}(v)$ are also satisfied as they all contain the propositional literal $\neg P_{in_\lambda}$. Consequently, all clauses in $C(v) = C_{in}(v) \cup C_{out}(v)$ are satisfied by $\xi_{e'}$. □

It is now straightforward to extend the previous lemma to the problem of finding a non-empty expression with all variables proper.

Lemma 2. *Let all assumptions of Lemma 1 hold. Let further* $C = \bigcup_{v \in Var(e)} C(v)$ *and* $C_\emptyset = \{\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_p\}$ *Then the following assertions are equivalent:*

1. *Expression e' is non-empty and each $v \in Var(e)$ is τ-proper in e'.*
2. *Assignment $\xi_{e'}$ satisfies all clauses in the Horn clause set $C \cup C_\emptyset$.*

Proof. $(1 \Rightarrow 2)$ If e' is non-empty then some $l_i \in e'$, which means that P_i is assigned the *false* value by $\xi_{e'}$. Thus $C_\emptyset$ is clearly satisfied. Since each $v \in Var(e)$ is τ-proper in e', every clause in each $C(v)$ $(v \in Var(e))$ is satisfied due to Lemma 1. Therefore all clauses in $C \cup C_\emptyset = \bigcup_{v \in Var(e)} C(v) \cup C_\emptyset$ are satisfied.

$(2 \Rightarrow 1)$ Since $C_\emptyset$ must be satisfied, there is some i $(1 \leq i \leq p)$ such that P_i is false in the assignment $\xi_{e'}$. Then $l_i \in e'$, ie. e' is non-empty. As all clauses in each $C(v)$ $(v \in Var(e))$ are satisfied, all $v \in Var(e)$ are τ-proper in e' due to Lemma 1. □

Lemma 3. *Let all assumptions of Lemma 2 hold. A maximal assignment (ie. one assigning the* false *value to the smallest number of variables) satisfying all clauses in C can be found (or decided that no satisfying assignment exists) in time polynomial in r and s.*

Proof. P-completeness of satisfying a set of propositional Horn clauses (the 'HORNSAT' problem) is shown in [5]. Finding a maximal (or minimal) assignment in polynomial time is shown in [2]. □

I am finally in the position to show the main result of this paper, which is informally as follows. If a polynomial bottom set can be constructed such that all acceptable features (up to variable renaming) are subsets thereof, one can find a feature (if it exists) in polynomial time. This is despite the fact that there is of course an exponential number of subsets of the bottom set.

Theorem 1. *Let $T \subseteq \mathbf{T}$, $E : T \to 2^{\mathbf{E}}$ and let there be $\bot : T \times N \to \mathbf{E}$ such that for all $\tau \in T$, $n \in N$: $\bot(\tau, n)$ is computable in time polynomial in n, $\bot(\tau, n) \preceq_\theta t_\tau$, and for all τ-features f it holds that $f \approx f' \in E(\tau)$ iff $f \subseteq \bot(\tau, |f|)$. Then the feature existence problem for T and E can be solved in polynomial time.*

Proof. Given the bottom set $\bot(\tau, n)$, the feature existence problem is equivalent to deciding if there is a subset f of the bottom set such that $|f| \leq n$ and f is a τ-feature. $\bot(\tau, n) \preceq_\theta t_\tau$ implies[3] $f \preceq_\theta t_\tau$ for arbitrary $f \subseteq \bot(\tau, n)$. Also, as

[3] Clearly, if $e' \subseteq e' \preceq_\theta e''$ then $e' \preceq_\theta e''$ for any expressions e, e', e''.

$\bot(\tau, n)$ is finite, so is every subset thereof. Considering Def. 3, it thus remains to decide whether there exists a non-empty $f \subseteq \bot(\tau, n)$, $|f| \leq n$ where every variable is τ-proper, that is, it appears both at some input argument (ie. $v = v_{a^+}$, $a^+ \in Arg^+_\tau(f)$) and some output argument (ie. $v = v_{a^-}$, $a^- \in Arg^-_\tau(f)$). For brevity denote $e = \bot(\tau, n)$. As Lemma 2 shows, finding a non-empty subset $f \subseteq e$ where each $v \in Var(e)$ (and thus each $v \in Var(f)$) is τ-proper in f, is equivalent to finding a satisfying assignment to a set of propositional Horn clauses $C = \bigcup_{v \in Var(e)} C(v) \cup C_\emptyset$ with propositional variables $P_1, P_2, \ldots, P_{|e|}$. As e is computable in time polynomial in n, $|Arg(e)|$ is at most polynomial in n and so are the numbers r, s in Eq.'s 1 and 2 ($r = Arg^+_\tau(e) \leq |Arg(e)| \geq Arg^-_\tau(e) = s$). Confronting this with the clause set 3-10, each $C(v)$ has a polynomial (in n) number of clauses and literals. As $|Var(e)| \leq |Arg(e)|$ (due to the existence of the function $v : Arg(e) \rightarrow Var(e)$ defined as $v((x, i)) = v_{x,i}$), also C has a polynomial number of clauses and literals. Due to Lemma 3, if there is a satisfying assignment to C, one can find in polynomial time a maximal one, which corresponds to the smallest expression $f_{min} \subseteq e$ where each $v \in Var(f_{min})$ is τ-proper in f_{min}. If f_{min} exists and $|f_{min}| \leq n$, the feature existence problem is answered with f_{min}. Otherwise it is answered with "NO". □

For example, the bottom set for τ specified below Def. 3 and $n = 3$ is $\bot(\tau, n) =$ `hasCar(C),hasLoad(C,L),small(L),big(L)`. Clearly, all τ-features (up to variable renaming) of length up to 3 literals are subsets of this bottom set (incidently equivalent to t_τ). Let the propositional variables assigned to the bottom literals (in the order of their appearance) be P_1, P_2, P_3, P_4. The corresponding HORNSAT instance consists of the Horn clauses $C_{in}(\texttt{C}) = \{P_2 \vee \neg P_1\}$, $C_{out}(\texttt{C}) = \{P_1 \vee \neg P_2\}$, $C_{in}(\texttt{L}) = \{P_3 \vee \neg P_2, P_4 \vee \neg P_2\}$, $C_{out}(\texttt{L}) = \{P_2 \vee \neg P_3 \vee \neg P_4\}$ and $C_\emptyset = \{\neg P_1 \vee \neg P_2 \vee \neg P_2 \vee \neg P_4\}$. One of the maximal solutions assigns the *false* value to P_1, P_2 and P_3 (the reader will check that all mentioned clauses are satisfied), which corresponds to the set `hasCar(C),hasLoad(C,L),small(L)`, which therefore forms a correct feature.

3 Work-in-Progress Outline

Two crucial issues determine the usefulness of the bottom set approach presented above: (I1) for which $T \subseteq \mathbf{T}$ and which $E : T \rightarrow 2^{\mathbf{E}}$ can a polynomial bottom set be constructed? (I2) when does tractability of the feature existence problem entail tractability of the enumeration of *all* τ-features in $E(\tau)$?

The results I have for (I1), which due to the limited space here I can only touch upon, are based on the following terms. Let $T \subseteq \mathbf{T}$, $E : T \rightarrow 2^{\mathbf{E}}$, $\tau \in \mathbf{T}$ and $e \in \mathbf{E}(\tau)$. $x, y \in e$ are said to be *connected* in e iff they share a variable or some $z \in e$ is connected with both x and y. e is is said to be *connected* (or *undecomposable*) iff any literal in e is connected to all other literals in e. There is a *path* in a τ-feature f from $x \in f$ to $y \in f$ of length 1, iff for some $a, b \in Arg(f)$ it holds $v_a = v_b$, $a \notin M_\tau$, $b \in M_\tau$, or a path of length $l + 1$, iff for some $z \in f$ there is a path from x to z of length l and a path from z to y of length 1. The *distance* $\delta_\tau(x, y)$ is the length of the shortest path from x to y, if one exists. The *depth* of f is defined as $\Delta_\tau(f) = \max_{u,v \in f} \delta_\tau(u, v)$. f is said to be *loop-free* if for no x there is a path in f from x to x. A loop-free τ-feature

f is called a *semi-root (semi-tree)* iff no variable has two input (output) occurrences in f w.r.t τ; f is called a *root (tree)* iff it is a semi-root (semi-tree) and no literal in f has two output (input) arguments w.r.t τ. Further, f is called a *semi-chain (chain)* iff it is simultaneously a semi-root (root) and a semi-tree (tree). Finally, τ is said to be hierarchical iff there is a partial irreflexive order $\prec$ on $Var(t_\tau)$ such that $v_{l,i} \prec v_{l,j}$ whenever there are l, i, j such that $(l, i) \in Arg^+_\tau(t_\tau)$ and $(l, j) \in Arg^-_\tau(t_\tau)$.

Theorem 2. *Assumptions of Theorem 1 are satisfied if for each $\tau \in T$: every $e \in E(\tau)$ is connected and any of the following holds: (A) each τ-feature in $E(\tau)$ is either a tree, root or chain, (B) there is a $\Delta_{max} \in N$ such that every τ-feature f in $E(\tau)$ is loop-free and $\Delta_\tau(f) \leq \Delta_{max}$, (C) τ is hierarchical (this implies (B)).*

The proof will appear in an extended version of the paper.

The (I2) issue considers the *feature enumeration problem* which is the same as the problem of feature existence, except that a solution to the former problem is the set of all distinct (ie. mutually non-equivalent) solutions to the latter problem. An auxiliary, yet important lemma for this problem is as follows.

Lemma 4. *Let the feature existence problem for T and E be decidable in polynomial time and let there be a polynomial number of distinct solutions to every instance thereof. Then the feature enumeration problem for T and E can be decided in polynomial time.*

Proof. Direct consequence of the enumeration-tractability result in [1]. □

From preliminary analysis, it seems that satisfying the conditions dictated by Theorem 2 implies a polynomial number of solutions to the existence problem and thus the polynomial decidability of the enumeration problem. Lastly, let me outline two *negative* results I have so far.

Theorem 3. *Neither the feature existence problem for T and E nor the feature enumeration problem for T and E can be decided in polynomial time if $T = \mathbf{T}$ and E is such that any of the following holds for each $\tau \in T$: (A) each $e \in E(\tau)$ is connected, (B) each τ-feature in $E(\tau)$ is a semi-chain.*

Although the full proof does not fit in this paper, let me remark that the former result is based on a polynomial reduction of the (out of NP) graph connectivity problem onto the feature existence problem, while for the latter, a polynomial reduction from the NP-complete integer programming problem is possible (the semi-chain property preserving the correspondence between the variable 'production-consumption' axiom and the arithmetic summation/subtraction operations).

References

1. R. Dechter and A. Itai. Finding All Solutions if You can Find One *AAAI-92 Workshop on Tractable Reasoning*, 1992
2. W. F. Dowling and J. H. Gallier. Linear time algorithm for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267-284, 1984.
3. N. Lavrač and P. A. Flach. An extended transformation approach to inductive logic programming *ACM Transactions on Computational Logic* 2:4, 2001
4. N. Lavrač, F. Železný and P. A. Flach. RSD: Relational Subgroup Discovery through First-Order Feature Construction *12th Int. Conf. on Inductive Logic Programming*, Springer 2002
5. Thomas J. Schaefer. The complexity of satisfiability problems. *Tenth Annual Symposium on Theory of Computing*, 1978.