Block-Wise Construction of Acyclic Relational Features with Monotone Irreducibility and Relevancy Properties

Ondřej Kuželka KUZELO1@FEL.CVUT.CZ
Filip Železný ZELEZNY@FEL.CVUT.CZ
Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic

Abstract

We describe an algorithm for constructing a set of acyclic conjunctive relational features by combining smaller conjunctive blocks. Unlike traditional level-wise approaches which preserve the monotonicity of frequency, our block-wise approach preserves a form of monotonicity of the irreducibility and relevancy feature properties, which are important in propositionalization employed in the context of classification learning. With pruning based on these properties, our block-wise approach efficiently scales to features including tens of first-order literals, far beyond the reach of state-of-the art propositionalization or inductive logic programming systems.

1. Introduction

Propositionalization aims at converting structured descriptions of examples into attribute-value descriptions which can be processed by most established machine learning algorithms. A major stream of propositionalization approaches (Lavrač & Flach, 2001; Krogel & al, 2003; Železný & Lavrač, 2006) proceeds by constructing a set of features (first-order formulas) which follow some prescribed syntactical constraints and play the role of Boolean attributes. Here we assume that examples are represented as first-order logic interpretations, features are first-order conjunctions and some Horn background theory B (possibly empty) is available. Feature F acquires value *true* for example I (is *covered* by the example) if $m(B) \cup I \models F$ where $m(B)$ is the minimal model of B, otherwise it has the *false* value. With slight abuse of notation, we will write this relation simply as $B \wedge I \models F$. Propositionalization is obviously similar to the framework of *frequent query discovery* (Dehaspe & Toivonen, 1999). The two settings mainly differ in the choice of the feature (query) evaluation criterion; in the latter this is based on the *frequency* on a set of unlabeled examples (i.e. the number of examples covered by the query), whereas in propositionalization it usually combines the values of feature frequencies in respective classes of a labeled example set into a single measure of feature *relevancy*.

Current systems working in both of the mentioned frameworks (e.g. RSD or WARMR) construct conjunctions in a level-wise manner. Each conjunction in level n extends by one literal some conjunction in level $n-1$. *Monotonicity* of frequency (if F is not frequent then $F \wedge l$ is not frequent for any literal l) is greatly exploited for pruning in the frequent-pattern discovery setting. Unfortunately, *relevancy* that is of interest in propositionalization is in general not monotone in this level-wise approach. *Irreducibility*, another desirable property of a feature, is unfortunately not monotone either here. For example $p(X)$ is irreducible, $p(X) \wedge p(Y)$ is reducible, and $p(X) \wedge p(Y) \wedge q(X, Y)$ is again irreducible.

The purpose of this work is to remove these deficiencies by proposing a novel, *block-wise* approach to construct a feature set, by identifying small conjunctions ('building blocks') out of which all features can be composed. The properties of irreducibility and relevancy, and the extensions (the sets of examples covered) of the building blocks are exploited to compute the analogical properties of the resulting composition.

In the next section we formalize our chosen feature language bias requiring a form of acyclicity. Then we provide our main theoretical contributions expressed in two theorems. Theorem 3.3 in Section 3 mainly shows that irreducibility is monotone in the block-wise composition sense and Theorem 4.4 in Section 4 asserts an analogical property for relevancy. In Section 5 we informally demonstrate the completeness of our

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

feature construction algorithm ('RELF'). In Section 6, on three benchmark relational learning problems, we test RELF against the level-wise feature construction algorithm RSD (Železný & Lavrač, 2006) and the inductive logic programming system Progol (Muggleton, 1995). RELF is significantly faster than the competing systems, able to efficiently search among conjunctions of tens of first-order literals, far beyond the reach of RSD or Progol. Of equal importance, classification accuracies achieved with RELF's features are insignificantly different from those achieved with the competing systems, indicating RELF's language bias is not overconstrained. Section 7 concludes the paper.

2. Features

The set of literals in a conjunction C is written as $lits(C)$, $|C| = |lits(C)|$ is size of C. A set of conjuctions is said to be *standardized-apart* if no two conjunctions in the set share a variable. Given a set A of atoms, we denote $Args(A) = \{(a,n)|a \in A, 1 \leq n \leq \mathsf{arity}(a)\}$, i.e. $Args(A)$ is the set of all argument places in A. We will assume that no conjunction contains two equal literals, i.e. $p(X) \wedge p(Y)$ will be allowed, but $p(a) \wedge p(a)$ will not. For an atom a, $arg_i(a)$ is its i-th argument. The admissible features syntax will be constrained by means of *templates*, which are a simple formalization of *mode declarations* used in systems such as Progol (Muggleton, 1995).

Definition 2.1 (Template) *A pre-template is a pair (γ, μ) where γ is a finite set of ground atoms and $\mu \subseteq Args(\gamma)$. Elements of μ ($Args(\gamma)\backslash\mu$) are called* inputs *(*outputs*) in γ. A pre-template (γ, μ) is a* template *if every atom in γ has at most one input argument and there is a partial irreflexive order $\prec$ on constants in γ such that $c \prec c'$ whenever c (c') occurs as an input (output) in γ.*

A template $\tau = (\gamma, \mu)$ is conveniently shown by writing γ with input (output) arguments marked with the sign + (-), such as $\tau \approx \{hasCar(-c), hasLoad(+c,-l), box(+l), large(+l), triangle(+l)\}$.

Definition 2.2 (Feature) *Given a template $\tau = (\gamma, \mu)$, a τ-pre-feature F is a finite conjunction of literals containing no constants or functions, such that $lits(F\theta) \subseteq \gamma$ for some substitution θ. The occurrence of variable in the i-th argument of literal l in F is an* input *(*output*) occurrence in F if the i-th argument of $l\theta$ is (is not) in μ. A variable is* neutral *in F if it has [i] at least one input occurrence in F, and [ii] exactly one output occurrence in F. A variable is* pos *(*neg*) in F if it complies only with [i] ([ii]). A τ-pre-feature is a τ-*feature *F if all its variables are neutral in F; it is a* pos (neg) *τ-*feature *F if it has exactly one pos (neg) variable, denoted $\mathsf{p}(F)$ ($\mathsf{n}(F)$), and the remaining variables are neutral; it is a* pos-neg *τ-*feature *if it has exactly one pos variable and exactly one neg variable and the remaining variables are neutral.*

For example, using the τ defined under Def. 2.1 the following conjunction is a τ-feature

$$hasCar(C) \wedge hasLoad(C, L1) \wedge hasLoad(C, L2) \wedge$$

$$\wedge box(L1) \wedge large(L1) \wedge triangle(L2)$$

$hasCar(C)$ is a neg τ-feature, $hasLoad(C, L1) \wedge box(L1)$ is a pos τ-feature and $hasLoad(C, L1)$ is a pos-neg τ-feature. It is important to note that the only pos (neg) variable $\mathsf{p}(F^+)$ ($\mathsf{n}(F^-)$) in a pos (neg) feature is uniquely given despite the fact that inputs and outputs are not given uniquely in general.

Wherever we shall deal with a single fixed template τ, we will simply speak of (pos, neg, pos-neg) features instead of (pos, neg, pos-neg) τ-features. Given the assumed partial order in templates, a feature F corresponds to a tree graph T_F, here called the F-tree, with vertices v_i corresponding to literals l_i. There is an edge between v_i and v_j if there is a variable which has an output occurrence in l_i and an input occurrence in l_j. We say that a (pos, neg) feature has depth d if the corresponding F-tree has depth d. Analogically, we say that a literal l is in depth d in (pos,neg) feature if it is in depth d in the corresponding F-tree (if $d = 0$, we call l *root* of F).

Definition 2.3 (Graft) *Let F^- be a neg (pos-neg) feature and $\phi^+ = \{F_i^+\}$ be a standardized-apart (possibly with exception $\mathsf{n}(F^-) = \mathsf{p}(F_i^+)$) set of pos features. We define the* graft *$F^- \oplus_{\mathsf{n}(F^-)} \phi^+ = F^- \wedge_i F_i\theta_i$, where each substitution $\theta_i = \{\mathsf{p}(F_i^+)/\mathsf{n}(F^-)\}$. A pos feature F^+ is said to be contained in (pos) feature F if and only if $F^+ \subset F$ and $F\backslash F^+$ is a neg (pos-neg) feature.*

In what follows, the variable in the subscript of the graft operator will be uniquely determined by context. Hence we will mostly drop the subscript for simplicity.

Running Example Let us have a set of two positive examples E^+ and a set of two negative examples E^-

$$E^+ = \{\{hasCar(c1), hasLoad(c1, l1), circ(l1),$$

$$box(l1), hasLoad(c1, l2), tri(l2)\},$$

$$\{hasCar(c2), hasLoad(c2, l3), box(l3), tri(l3)\}\}$$

$$E^+ = \{\{hasCar(c3), hasLoad(c3, l4), box(l4),$$

$$circ(l4)\},$$
$$\{hasCar(c4), hasLoad(c4, l5), tri(l5), circ(l5)\}\}$$

We define a very simple template $\tau \approx \{hasCar(-c), hasLoad(+c, -l), box(+l), tri(+l), circ(+l)\}$ to constrain the features for this data. Although there is an infinite number of features correct w.r.t. τ, there are only finitely many features, which are not *H-reducible*, as we will see in the next section.

3. Irreducibility

To define the reducibility property of conjunctive features, we borrow the notion of θ-subsumption which is usually employed in the context of clauses.

Definition 3.1 (Reducibility) *Let C and D be conjunctions of literals and let there be a substitution θ such that $lits(C\theta) \subseteq lits(D)$. We say that C θ-*subsumes *D (written $C \preceq_\theta D$). If further $D \preceq_\theta C$, we call C and D θ-equivalent (written $C \approx_\theta D$). We say that C is* reducible *if there exists a clause C' such that $C \approx_\theta C'$ and $|C| > |C'|$. A clause C' is said to be a* reduction *of C if $C \approx_\theta C'$ and C' is not reducible.*

An example of a θ-reducible conjunction of literals is $C = hasCar(C) \wedge hasLoad(C, L1) \wedge hasLoad(C, L2) \wedge box(L1)$. Let $D = hasCar(C) \wedge hasLoad(C, L1) \wedge box(L1)$, then $C\theta \subseteq D$, where $\theta = \{L2/L1\}$, and $|D| < |C|$. In this work we cannot rely directly on the established notion of reducibility as defined above. This is because a reduction of a τ-feature may not be a τ-feature itself. For example, for $\tau \approx \{car(-c_1), hasLoad(+c_1, -l), hasLoad(-c_2, +l), hasCar(+c_2)\}$, the conjunction $car(C_1) \wedge hasLoad(C_1, L_1) \wedge hasLoad(C_2, L_1) \wedge car(C_2)$ is a correct τ-feature but its reduction $hasCar(C_1) \wedge hasLoad(C_1, L_1)$ is not.

The fact that reduction does not preserve correctness of feature syntax may represent a problem because, to avoid redundancy, we would like to work only with reduced features. The next definition introduces H-reduction, which has the property that H-reduction of a τ-feature is always a τ-feature. While there is an infinite number of τ-features for a sufficiently rich template τ, there is always only a finite number of non-H-reducible ones.

Definition 3.2 (H-reduction) *We say that (pos) τ-feature f H-subsumes (pos) τ-feature g (written $f \preceq_H g$) if and only if there is a substitution (called H-substitution) θ such that $f\theta \subseteq g$ and for every literal $l \in lits(f)$ it holds $depth_f(l) = depth_g(l\theta)$ for some correct assignment of inputs and outputs of f and g. If further $g \preceq_H f$, we call f and g H-equivalent (written $f \approx_H g$). We say that (pos) τ-feature f is H-reducible if there is a (pos) τ-feature f' such that $f \approx_H f'$ and $|f| > |f'|$. Feature f' is said to be an H-reduction of f if $f \approx_H f'$ and f is not H-reducible.*

In the proof of Theorem 3.3 we will speak interchangeably about substitution θ from variables to terms and about the induced substitution from literals to literals.

Theorem 3.3 *Let F^+ be a pos feature and let F^- be a neg feature. Then following holds: (i) F^+ is H-reducible if and only if F^+ contains pos features F_1^+, F_2^+ such that $F_1^+ \neq F_2^+$, $\mathsf{p}(F_1^+) = \mathsf{p}(F_2^+)$ and $F_1^+ \preceq_H F_2^+$. (ii) If F^+ is H-reducible, then $F^- \oplus F^+$ is also H-reducible. (iii) Whether a (pos) feature F is H-reducible can be computed in polynomial time (in $|F|$).*

Proof In this proof we will use the following observation. Let A, B be pos features and let θ be a H-substitution such that $A\theta \subseteq B$. Observe that if $l \in B \backslash A\theta$, then also $l' \in B \backslash A\theta$ for any literal l' contained in pos feature $F_l^+ \subseteq B$, where F_l^+ has l as its root. (i $\Rightarrow$) Let F_r^+ be H-reduction of F^+ and let θ_1, θ_2 be H-substitutions such that $F_r^+\theta_1 \subseteq F^+$ and $F^+\theta_2 \subseteq F_r^+$. Substitution $\theta_3 = \theta_2\theta_1$ is a mapping $\theta_3 : lits(F^+) \rightarrow lits(F^+)$. Since $F^+\theta_2 \subseteq F_r^+$, $|F^+\theta_2| \leq |F_r^+|$ and consequently $|F^+\theta_3| \leq |F_r| < |F^+|$, because applying a substitution to a feature cannot increase its size. Therefore there is a literal $l \in F^+ \backslash F^+\theta_3$ and, as we have observed, also a whole pos feature $F_1^+ \subseteq F^+ \backslash F^+\theta_3$. Thus, there is a pos feature F_2^+ ($F_2^+ \neq F_1^+$) such that $F_1^+\theta_3 \subseteq F_2^+$. It remains to show that for some such F_1^+, F_2^+, $\mathsf{p}(F_1^+) = \mathsf{p}(F_2^+)$. If $\mathsf{p}(F_1^+) \neq \mathsf{p}(F_2^+)$, then there must be pos features $F_1^{+'}, F_2^{+'}$ such that $F_1^+ \subset F_1^{+'}$, $F_2^+ \subset F_2^{+'}$ and $F_1^{+'}\theta_3 \subseteq F_2^{+'}$. For such $F_1^{+'}, F_2^{+'}$ with maximum size, $\mathsf{p}(F_1^+) = \mathsf{p}(F_2^+)$. (i $\Leftarrow$) Let θ be a H-substitution such that $F^+ \backslash F^+\theta = F_1^+$, then $F^+\theta \approx_H F^+$ and $|F^+\theta| = |F^+| - |F_1^+| < |F^+|$. (ii) This follows from (i). (iii) An approach, which tests whether $F_1^+ \preceq_H F_2^+$ for all pairs of pos features F_1^+, F_2^+ contained in F^+ with equal pos variables, runs in polynomial time in $|F|$ (cf. Algorithm 1).

The second property of H-reduction stated in Theorem 3.3 allows us to filter H-reducible pos features during propositionalization process.

Running Example Let us return to our running example from Section 2. As we have already mentioned, there is an infinite number of features, but only a finite number of non-H-reducible ones. An example of

Algorithm 1 $\mathsf{dom}_{I,B}(S)$

Input: Pos feature F^+, Interpretation I, Background theory B;
$litsDom \leftarrow \{l | pred(l) = pred(root(F^+)) \wedge ((I \wedge B) \models l)\}$
for $\forall$ output variables out_i of F^+'s root **do**
  $argDom_i \leftarrow \cap_{c \in children_{out_i}(l)} \mathsf{dom}_I(c)$
  $litsDom \leftarrow litsDom \cap \{l | arg_i(l) \in argDom_i\}$
end for
return $\{t | t$ is term at input of $l \wedge l \in litsDom\}$

an H-reducible feature for template τ is

$$F_{reducible} = hasCar(C) \wedge hasLoad(C, L) \wedge box(L) \wedge$$
$$\wedge hasLoad(C, L2) \wedge box(L2) \wedge circ(L2) \wedge tri(L2).$$

This feature is indeed H-reducible as may be seen from the following fact

$$hasLoad(C, L) \wedge box(L) \preceq_H$$
$$\preceq_H hasLoad(C, L2) \wedge box(L2) \wedge circ(L2) \wedge tri(L2)$$

When all H-reducible features are removed, there remain only 18 correct τ-features for τ from our running example.

4. Relevancy

Definition 4.1 (Domain) *Let I be an interpretation, B be a background theory, τ be a template and T be a set of terms. Let S be a standardized-apart set of pos τ-features. Then, domain w.r.t. I and B ($\mathsf{dom}_{I,B}$) is a mapping $\mathsf{dom}_{I,B} : S \rightarrow 2^T$ such that $\mathsf{dom}_{I,B}(F^+)$ contains all terms t such that $(I \wedge B) \models F^+\theta$, where $\theta = \{\mathsf{p}(F^+)/t\}$.*

Domain assigns to each pos feature a set of terms $\{t_i\}$, for which there is a grounding of S such that $p(S) = t_i$ and the grounding of S is true in $I \wedge B$. In order to allow efficient computation of domains of pos features, we make the assumption that for every literal l with a subset of output arguments $out_1, \ldots, out_i$ grounded, it is possible to find the set of all possible groundings of this literal efficiently. If this assumption holds, then an algorithm exists, which correctly computes domain and works in time polynomial in the size of F^+ (Algorithm 1). This algorithm is not novel, for $B = \emptyset$ it corresponds to an algorithm known as directed-arc-consistency algorithm in the field of CSP and as conjunctive acyclic query algorithm in database theory (Yannakis, 1981).

Once we have established how domain of a pos feature is computed, it is easy to use this method to decide whether $(I \wedge B) \models F$, where F is a neutral feature. If $l = pred(X_1, \ldots, X_n)$ is root of F, it suffices to replace l by $l' = pred(X_0, X_1, \ldots, X_n)$ in F and extend background theory to $B' = B \cup \{pred(yes, X_1, \ldots, X_n) \leftarrow pred(X_1, \ldots, X_n)\}$. If domain of this newly created pos feature $\mathsf{feat}_+(F)$ is non-empty, then $(I \wedge B) \models F$. This simplifies notation because we do not need to treat neutral features separately.

Example Let us have feature F and interpretation I

$$F = hasCar(C) \wedge hasLoad(C, L) \wedge tri(L) \wedge box(L),$$

$$I = \{hasCar(c), hasLoad(c, l1), hasLoad(c, l2), tri(l1),$$
$$circ(l1), tri(l2), box(l2), hasLoad(c, l3), box(l4)\},$$
$$B = \emptyset.$$

First, we modify F so that we could use Algorithm 1 to decide whether $(I \wedge B) \models F$, i.e. we replace $hasCar(C)$ by $hasCar(X, C)$ and extend the background theory $B = \{hasCar(yes, X) \leftarrow hasCar(X)\}$.[1] Then we may proceed as follows: (i) We compute domains of pos features $box(L)$ and $tri(L)$, i.e. $\mathsf{dom}_I(box(L)) = \{l2, l4\}$, $\mathsf{dom}_I(tri(L)) = \{l1, l2\}$. (ii) Then we compute domain of pos feature with root $hasLoad(C, L)$, i.e. $\mathsf{dom}_I(hasLoad(C, L)) = \{l1, l2, l3\} \cap \{l2\} \cap \{l1, l2\} = \{l2\}$. (iii) Since no domain is empty so far, we proceed further and compute domain of pos feature with root $hasCar(X, C)$, which becomes $\mathsf{dom}_I(hasCar(X, C)) = \{yes\} \cap \{yes\}$. So, we see that $(I \wedge B) \models F$.

The next definition introduces irrelevancy of boolean attribute (Lavrač et al., 1999), which enables one to filter such *irrelevant* attributes from dataset.

Definition 4.2 (Irrelevant Attribute) *Let A be a set of boolean attributes, let $cov(a)$ denote subset of examples covered by $a \in A$ and let $pos(a)$ ($neg(a)$) denote subset of positive (negative) examples covered by $a \in A$. A boolean attribute $a \in A$ is said to be E^+-irrelevant (E^--irrelevant) if and only if there is $a' \in A$, $a \neq a'$ such that $pos(a) \subseteq pos(a')$ and $neg(a') \subseteq neg(a)$ ($neg(a) \subseteq neg(a')$ and $pos(a') \subseteq pos(a)$). If one of the inclusions, for at least one example, is strict, a is said to be strictly E^+-irrelevant (E^--irrelevant). A boolean attribute a is called irrelevant if it is both E^+-irrelevant and E^--irrelevant. It is called strictly irrelevant if it is irrelevant and strictly E^+-irrelevant or strictly E^--irrelevant.*

[1] We assume that there had been no literals $hasCar(X, C)$ before we added them to the original feature and to background theory.

In this section, we develop methods for detection of pos features, which give rise to irrelevant features when grafted with neg features.

Lemma 4.3 *Let I be an interpretation, B background theory and let $S_1 = \{F_1^+, F_2^+, \ldots, F_m^+\}$ and $S_2 = \{G_1^+, G_2^+, \ldots, G_n^+\}$ be standardized-apart sets of pos features such that $\cap_{i=1}^m \mathsf{dom}_{I,B}(F_i^+) \subseteq \cap_{i=1}^n \mathsf{dom}_{I,B}(G_i^+)$, then for any pos-neg feature F^- $\mathsf{dom}_{I,B}(F^- \oplus_V S_1) \subseteq \mathsf{dom}_{I,B}(F^- \oplus_V S_2)$*

Proof Let us first consider the case when $depth_{F^-}(V) = 0$. The only place, where domains of $F_i \in S_1$ ($G_i \in S_2$ respectively) are used, is line 4 in Algorithm 1. Clearly $argDom_{S_1} = \cap_{i=1}^m \mathsf{dom}_{I,B}(F_i^+) \subseteq argDom_{S_2} = \cap_{i=1}^n \mathsf{dom}_{I,B}(G_i^+)$ and consequently $litsDom_{S_1} \subseteq litsDom_{S_2}$ and therefore also $\mathsf{dom}_{I,B}(F \oplus_V S_1) \subseteq \mathsf{dom}_{I,B}(F \oplus_V S_2)$. The general case of lemma may be proved by induction on depth of V. (i) The case for depth 0 has been already proved. (ii) Let us suppose that lemma holds for depth d. Now, we suppose that $depth_{F^-}(V) = d + 1$. We may take the pos-neg feature $T \subset F^-$ which contains V such that $depth_T(V) = 0$, $W = \mathsf{p}(T)$ (respecting inputs/outputs of F^- and $\mathsf{n}(F^-) = V$) and *graft it* with S_1 (S_2, respectively). We have $\mathsf{dom}_{I,B}(T \oplus_V S_1) \subseteq \mathsf{dom}_{I,B}(T \oplus_V S_2)$ and by induction argument finally also $\mathsf{dom}_{I,B}(F^- \oplus_V S_1) = \mathsf{dom}_{I,B}((F^- \backslash T) \oplus_W \{T \oplus_V S_1\}) \subseteq \mathsf{dom}_{I,B}((F^- \backslash T) \oplus_W \{T \oplus_V S_2\}) = \mathsf{dom}_{I,B}(F^- \oplus_V S_2)$, which finishes the proof.

Theorem 4.4 *Let E^+ be a set of positive examples, let E^- be a set of negative examples and let B be background theory. Let $S = \{F_i^+\}_{i=1}^n$ be a set of distinct pos features with equal types of input arguments such that for all $i \neq j$ there is an example $I \in E^+ \cup E^-$ with $\mathsf{dom}_{I,B}(F_i^+) \neq \mathsf{dom}_{I,B}(F_j^+)$. Let $\mathsf{dom}_{I,B}(F_1^+) \subseteq \cap_{i=2}^n \mathsf{dom}_{I,B}(F_i^+)$ be true for all $I \in E^+$ ($I \in E^-$) and let $\cap_{i=2}^n \mathsf{dom}_{I,B}(F_i^+) \subseteq \mathsf{dom}_{I,B}(F_1^+)$ be true for all $I \in E^-$ ($I \in E^+$). (i) For any neg feature F^-, $F^- \oplus \{F_1^+\}$ is E^+-irrelevant (E^--irrelevant). (ii) Let S' be a set of pos features obtained from S by repeatedly removing irrelevant pos features. Set S' is unique.*

Proof We will prove only the case for E^+-irrelevancy because the proof for E^--irrelevancy is analogous.

(i) Let F^- be a neg feature and let $F^\pm = \mathsf{feat}_+(F^-)$ be the corresponding pos-neg feature, which has unique $\mathsf{p}(F^\pm)$. By application of Lemma 4.3, if $\mathsf{dom}_{I,B}(F_1^+) \subseteq \cap_{i=2}^n \mathsf{dom}_{I,B}(F_i^+)$ for all $I \in E^+$, then $\mathsf{dom}_{I,B}(F^\pm \oplus \{F_1^+\}) \subseteq \mathsf{dom}_{I,B}(F^\pm \oplus \{F_2^+, \ldots, F_n^+\})$ for all $I \in E^+$. Similarly, if $\cap_{i=2}^n \mathsf{dom}_{I,B}(F_i^+) \subseteq \mathsf{dom}_{I,B}(F_1^+)$ for all $I \in E^-$, then $\mathsf{dom}_{I,B}(F^\pm \oplus \{F_2^+, \ldots, F_n^+\}) \subseteq \mathsf{dom}_{I,B}(F^\pm \oplus \{F_1^+\})$ for all $I \in E^-$. Therefore if $(I \wedge B) \models F^- \oplus \{F_1^+\}$, then $(I \wedge B) \models F^- \oplus \{F_2^+, \ldots, F_n^+\}$ for all $I \in E^+$ and similarly if $(I \wedge B) \models F^- \oplus \{F_2^+, \ldots, F_n^+\}$, then $(I \wedge B) \models F^- \oplus \{F_1^+\}$ for all $I \in E^-$. This means that F_1^+ is E^+-irrelevant.

(ii) It could happen that by removing some irrelevant pos features from S, F_1^+ could become relevant. We need to show that this is not the case. Let G^+ be a graph with vertices corresponding to S_i. Let there be an edge (v_i, v_j) if and only if $\mathsf{dom}_{I,B}(F_i^+) \subseteq \cap_{k \in A} \mathsf{dom}_{I,B}(F_k^+)$ for all $I \in E^+$ and $\cap_{k \in A} \mathsf{dom}_{I,B}(F_k^+) \subseteq \mathsf{dom}_{I,B}(F_i^+)$ for all $I \in E^-$ and $j \in A$, $i \notin A$. Let v_i be a vertex corresponding to pos feature F_i^+. Any vertex with non-zero in-degree corresponds to an irrelevant pos feature. If we show that G is acyclic, then it is easy to find the unique set of relevant pos features as the set of pos features corresponding to vertices with zero in-degree. To show that G is acyclic, we first notice that if two distinct vertices $v_i, v_j \in G$ are connected by a directed path, then there is also the edge (v_i, v_j). Therefore if there was a directed cycle containing (v_1, v_2), there would have to be also two edges (v_1, v_2) and (v_2, v_1). This would mean that for each positive example it would be true that $\mathsf{dom}_{I,B}(F_2^+) \subseteq \cap_{i=3}^n \mathsf{dom}_{I,B}(F_i^+) \cap \mathsf{dom}_{I,B}(F_1^+)$ and $\mathsf{dom}_{I,B}(F_1^+) \subseteq \cap_{i=3}^n \mathsf{dom}_{I,B}(F_i^+) \cap \mathsf{dom}_{I,B}(F_2^+)$, but then $\mathsf{dom}_{I,B}(F_1^+) = \mathsf{dom}_{I,B}(F_2^+)$ and the same would be true for negative examples implying $\mathsf{dom}_{I,B}(F_1^+) = \mathsf{dom}_{I,B}(F_2^+)$ for all examples, which contradicts assumption that no two distinct pos features have equal domains for all examples. Therefore G must be acyclic and the set of irrelevant pos features (in a given set) must be unique.

5. Algorithm

In this section, we design a propositionalization algorithm RelF (Algorithm 2). RelF merges the two usual phases of propositionalization, i.e. feature construction and extension computation. Specifically, the core algorithm accepts a set of learning examples and a feature template. The features are obtained by combinatorial composition of pos features, which act as the primitive building blocks. The advantage of this assembly approach is that H-redundant or irrelevant pos features may be removed from the set of pos features while guaranteeing that all relevant features will be found. In the filtering phase, the algorithm first filters pos features, which have equal domains for all examples, and only after that it also removes irrelevant pos features (thus satisfying conditions of Theorem 4.4). The rules used for detection of E^+-irrelevant (E^--irrelevant) pos features are based on

Algorithm 2 RELF (Sketch of Algorithm): Given a template and a set of examples, RELF computes the propositionalized table.

1: **Input:** template τ, examples E;
2: $PosFeats \leftarrow \{\}$
3: $OrderedDefs \leftarrow$ topologically ordered predicate definitions computed from τ
4: **for** $\forall pred \in OrderedDefs$ **do**
5: $NewPosFeats \leftarrow \{\}$
6: $NewPosFeats \leftarrow \mathsf{Combine}(pred, PosFeats)$
7: Filter H-reducible pos features
8: Filter pos features with equal domains for all examples
9: Filter irrelevant pos features
10: Add $NewPosFeats$ to $PosFeats$
11: **end for**
12: Save all correct features from $PosFeats$

Algorithm 3 Combine (Procedure used by RELF): Given a predicate symbol and a set of pos features, Combine constructs pos features.

1: **Input:** *predicate*, $PosFeats$;
2: $Constructed \leftarrow \{\}$
3: $ArgCombinations \leftarrow []$
4: **for** $\forall$ output arguments out_i of *predicate* **do**
5: $Smaller \leftarrow$ pos features from $PosFeats$ with *type* of input equal to *type* of out_i
6: $ArgCombinations\,[out_i] \leftarrow$ all combinations without repetition of $F^+ \in Smaller$
7: **end for**
8: $Constructed \leftarrow$ all possible *graftings* of $predicate(X_1, \ldots, X_k)$ with the respective combinations from $ArgCombinations$
9: **return** $Constructed$

Theorem 4.4. That means S_1 is E^+-irrelevant if there is set of pos features $\{S_i\}$ and $Ind \subset N$ such that $\mathsf{dom}_{I,B}(S_1) \subseteq \cap_{i \in Ind} \mathsf{dom}_{I,B}(S_i)$ on positive examples and $\cap_{i \in Ind} \mathsf{dom}_{I,B}(S_i) \subseteq \mathsf{dom}_{I,B}(S_1)$ on negative examples.

The algorithm exploits the partial irreflexive order, which is imposed on types of arguments by Def. 2.2. Due to existence of this order it is possible to sort all declared predicates $l \in \gamma$ topologically with respect to a graph induced by the partial order. When the topological order is found, it is possible to organize generation of features in such a way that pos features are built iteratively by combining smaller pos features into larger ones. With input arguments τ and E, where τ is a template and E is a set of examples, it returns set of neutral features $T_{Alg} \subseteq T_\tau$, where T_τ is set of all correct τ-features. An important property of Algorithm 2 is that for any neutral feature $F \in T_\tau$, which is not strictly irrelevant, there is a neutral feature $F' \in T_{Alg}$ such that F and F' cover the same set of examples. In other words, RELF correctly outputs all relevant boolean attributes, which means that it is complete in a well-specified sense.

Running Example In Section 3, we have made the set of correct τ-features finite. We have shown that, for our particular template τ, there are only 18 features. We will now demonstrate how RELF constructs the E^+-relevant features. We first generate and filter the following three pos features: $box(L)$, $tri(L)$, $circ(L)$. We may check that $circ(L)$ is E^+-irrelevant (due to $box(L)$), so we may safely throw it away. The next pos features created by grafting with $box(L)$ and $tri(L)$ are pos features $F_1^+ = hasLoad(C, L) \wedge box(L)$, $F_2^+ = hasLoad(C, L) \wedge box(L) \wedge tri(L)$ and $F_3^+ = hasLoad(C, L) \wedge tri(L)$. Notice that if we had not removed $circ(L)$, there would have been seven such pos features. We can now filter also F_1^+, F_2^+, F_3^+ in the exactly same manner as we filtered $box(L)$, $tri(L)$, $circ(L)$. In this case, F_2^+ is E^+-irrelevant because

$$\mathsf{dom}_{I_1}(F_2^+) = \emptyset \subseteq \mathsf{dom}_{I_1}(F_1^+) \cap \mathsf{dom}_{I_1}(F_3^+) = \{c1\},$$

$$\mathsf{dom}_{I_2}(F_2^+) = \{c2\} \subseteq \mathsf{dom}_{I_2}(F_1^+) \cap \mathsf{dom}_{I_2}(F_3^+) = \{c2\},$$

$$\mathsf{dom}_{I_3}(F_1^+) \cap \mathsf{dom}_{I_3}(F_3^+) = \emptyset \subseteq \mathsf{dom}_{I_3}(F_2^+) = \emptyset,$$

$$\mathsf{dom}_{I_4}(F_1^+) \cap \mathsf{dom}_{I_4}(F_3^+) = \emptyset \subseteq \mathsf{dom}_{I_4}(F_2^+) = \emptyset.$$

Finally, we may graft these pos features with $car(C')$ to obtain the resulting set of neutral features.

6. Experiments

In this section we evaluate perfomance and accuracy of RELF[2]. We evaluate RELF in two relational learning domains in comparison to RSD (Železný & Lavrač, 2006) and Progol (Muggleton, 1995). Our intention is to demonstrate (i) that RELF can propositionalize relational data orders of magnitude faster than standard algorithms and (ii) that classifiers built using features generated by RELF are not much worse than those built using more general feature seclarations.

In (Krogel & al, 2003) extensive experiments were conducted to compare three state-of-the-art propositionalization systems: RSD, SINUS and RELAGGS. In this study each of the systems obtained best predictive accuracy on exactly two out of six domains. RELAGGS

[2] All program codes can be obtained on request from the first author.

proved itself superior especially in domains where numerical attributes were essential. On the other hand SINUS and RSD performed well in typical ILP tasks such as predicting mutagenicity or learning legal positions of chess-end-games. However, in all experiments RSD was several times faster than SINUS. That is why we chose RSD for comparisons. We also conduct experiments comapring RELF to state-of-the-art ILP system Progol and we also compare our results with those presented in literature. In all experiments described in this section random forest classifiers[3] are used (Breiman, 2001). We follow suggestions given in (Scheffer & Herbrich, 1997) to obtain unbiased estimate of quality of learned classifiers. We perform experiments both with RSD having the same feature declaration bias as RELF and with RSD allowing cyclic features. While for RELF the only necessary restriction on features is given by the templates (which implicitly restrict depth of features), for RSD we also need to bound maximum size of features.

6.1. Mutagenesis

Our first set of experiments was done on the well-known Mutagenesis dataset (Lodhi & Muggleton, 2005), which consists of 188 organic molecules marked according to their mutagenicity. We used atom-bond descriptions and numerical attributes *lumo* and *logP*. The longest features found by RELF had over 20 *bond*-literals and were found in 116 seconds. The longest features found by RSD had 3 *bond*-literals and RSD needed 272 seconds. Bigger features could not be found in 15000s by RSD. RSD without acyclic feature bias was not able to find more features than with the acyclic bias.

Table 1 displays predictive accuracies obtained on the Mutagenesis dataset. The third column refers to Progol with maximum number of searched nodes set to 10000 and maximum clause length set to 4. Theory construction runtime was 398s. All clauses found by Progol were acyclic. The third column displays accuracy obtained by an ensemble method with a set of theories found by Progol reported in (Lodhi & Muggleton, 2005), which is to date the highest predictive accuracy for this dataset. However, in this last experiment more information about moleculs was used (functional groups and indicator variables). Therefore we repeated our experiments, but we added also the indicator variables and functional groups and obtained accuracy 87.4%. Adding functional groups was not very useful in this case, because RELF was already able to capture the functional groups due to its ability to construct long features.

[3]We used the random forest classifier from Weka package (Witten & Frank, 2005).

Table 1. Accuracies on Mutagenesis dataset.

Algorithm	RELF	RSD	Progol	Progol Ens.
Accuracy [%]	89.8	87.8	82.0	95.8

6.2. CAD Documents

The second set of experiments was conducted in a domain describing CAD documents (product structures) (Žáková et al., 2007). The dataset consists of 96 class-labeled examples. This dataset is interesting because long features are needed to obtain reasonable classification accuracy. For RSD we used both the same template (acyclic bias) and a slightly more general template. We needed to significantly constrain size of RSD's features to 12 literals for acyclic case (resulting in runtime 12324s) and 11 literals for cyclic case (resulting in runtime 2198s). This is in contrast with RELF, whose longest features had over 50 literals (with runtime 108s). Importantly, the single feature that separated the dataset best was discovered only by RELF. The accuracy of Progol is low due to the fact that Progol is unable to find clauses with sufficient lengths within 15000s limit, which agrees with findings reported in (Žáková et al., 2007).

Table 2. Accuracies on CAD dataset.

Algorithm	RELF	RSD	RSD (cyclic)	Progol
Accuracy [%]	96.7	96.7	91.2	81.1

We have performed an additional experiment on CAD dataset to make clear to what extent the speedup achieved by RELF compared to RSD is due is due to filtering of irrelevant pos features. With enabled irrelevancy filtering RELF ran 108 seconds, whereas with disabled irrelevancy filtering it crashed after running for several minutes because of lack of free memory. This shows that the key concept, which makes RELF efficient, is irrelevant pos feature filtering.

6.3. Predictive Toxicology Challenge

The last set of experiments was done with data from the Predictive Toxicology Challenge (Helma et al., 2001). The PTC dataset consists of 344 organic molecules marked according to their carcinogenicity on male and female mice and rats. Our experiments were done for male rats. Longest features constructed by RELF had over 20 *bond*-literals and were constructed in 762 seconds, while longest features constructed by RSD had only 4 *bond*-literals in 2971 seconds.

Table 3 refers to predictive accuracies obtained on the PTC dataset. With Progol, we were unable to obtain any theory compression. The third column in Table 3 refers to approach based on *optimal assignment kernel* (Fröhlich et al., 2005). The fourth column also refers to approach based on kernels (Ralaivola et al., 2005). Predictive accuracy reported for this last approach is the highest presented in literature, however, it is a leave-one-out estimate as opposed to 10-fold cross-validation estimates of the other discussed results.

Table 3. Accuracies on PTC dataset for male rats.

Algorithm	RELF	RSD	Kernel1	Kernel2
Accuracy [%]	62.5	64.9	63.0	65.7

7. Conclusions

We have introduced RELF, an algorithm for construction of acyclic relational features. We have shown that blockwise construction of acyclic features enables RELF to remove H-reducible and irrelevant pos features. In experiments, we have shown that RELF is able to construct *relevant* features with sizes far beyond the reach of state-of-the-art propositionalization systems. Of importance, we have also shown that, for three typical ILP datasets, restriction to acyclic features was not very detrimental with respect to predictive accuracy. In fact, the obtained results were very close to the best results reported in literature. Although there are definitely datasets where cyclic features could provide better predictive accuracies, one can always merge results from different propositionalization algorithms and feed the propositional learners with such merged propositionalized tables. An algorithm for construction of a limited class of features such as RELF would be useful also in such cases.

Acknowledgements. We are grateful to ICML 2009 reviewers for their insightful comments. The first author is supported by project 1ET101210513 (Czech Academy of Sciences), the second author is supported by project MSM6840770038 (Czech Ministry of Education). Both authors are further supported by project 201/08/0509 (Czech Science Foundation).

References

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

Dehaspe, L., & Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, *3*, 7–36.

Fröhlich, H., Wegner, J. K., Sieker, F., & Zell, A. (2005). Optimal assignment kernels for attributed molecular graphs. *International Conference on Machine learning (ICML '05)* (pp. 225–232). ACM.

Helma, C., King, R. D., Kramer, S., & Srinivasan, A. (2001). The predictive toxicology challenge 2000-2001. *Bioinformatics*, *17*, 107–108.

Krogel, M.-A., & al (2003). Comparative evaluation of approaches to propositionalization. *International Conference on Inductive Logic Programming (ILP 03')*. Springer.

Lavrač, N., & Flach, P. A. (2001). An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, *2*, 458–494.

Lavrač, N., Gamberger, D., & Jovanoski, V. (1999). A study of relevance for learning in deductive databases. *Journal of Logic Programming*, *40*, 215–249.

Lodhi, H., & Muggleton, S. (2005). Is mutagenesis still challenging. *International Conference on Inductive Logic Programming (ILP '05), Late-Breaking Papers* (pp. 35–40).

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, *13*, 245–286.

Ralaivola, L., Swamidass, S. J., Saigo, H., & Baldi, P. (2005). Graph kernels for chemical informatics. *Neural Networks*, *18*, 1093–1110.

Scheffer, T., & Herbrich, R. (1997). Unbiased assessment of learning algorithms. *15th International Joint Conference on Artificial Intelligence (IJCAI '97)* (pp. 798–803).

Železný, F., & Lavrač, N. (2006). Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, *62(1-2)*, 33–63.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco. 2nd edition.

Yannakis, M. (1981). Algorithms for acyclic database schemes. *International Conference on Very Large Data Bases (VLDB '81)* (pp. 82–94).

Žáková, M., Železný, F., Garcia-Sedano, J., Tissot, C. M., Lavrač, N., Křemen, P., & Molina, J. (2007). Relational data mining applied to virtual engineering of product designs. *International Conference on Inductive Logic Programming (ILP '07)*. Springer.