Exploiting Term, Predicate, and Feature Taxonomies in Propositionalization and Propositional Rule Learning

Monika Žáková, Filip Železný

Czech Technical University
Technická 2, 16627 Prague 6, Czech Republic
zakovm1@fel.cvut.cz, zelezny@fel.cvut.cz

Abstract. Knowledge representations using semantic web technologies often provide information which translates to explicit term and predicate taxonomies in relational learning. We show how to speed up the propositionalization by orders of magnitude, by exploiting such taxonomies through a novel refinement operator used in the construction of conjunctive relational features. Moreover, we accelerate the subsequent propositional search using feature generality taxonomy, determined from the initial term and predicate taxonomies and θ-subsumption between features. This enables the propositional rule learner to prevent the exploration of conjunctions containing a feature together with any of its subsumees and to specialize a rule by replacing a feature by its subsumee. We investigate our approach with a deterministic top-down propositional rule learner, and propositional rule learner based on stochastic local search.

1 Introduction

With the development of semantic web technologies and knowledge management using ontologies, increasing amounts of expert knowledge in important knowledge-intensive domains such as bioinformatics is becoming available in the form of ontologies and semantic annotations. However, semantic representation is becoming popular even in industrial use [13] for sharing and efficient searching of information in production enterprizes. Knowledge representation formalisms used to capture ontologies and semantic annotations are based on description logics, which have convenient properties with regard to complexity and decidability of reasoning [1].

Inductive logic programming (ILP) aims at learning a theory in a subset of first-order logic from given examples, taking background knowledge into account. It has been considerably successful in various knowledge discovery problems such as in bioinformatics [4]. Standard ILP techniques cannot efficiently exploit explicit taxonomies on concepts and relations, which are typically available in semantic knowledge representations [13]. While in principle, any taxonomy can be encoded in background knowledge, there is good reason to view ontologies as meta-information, which can be directly exploited to guide the refinement operator used to search through the space of first-order rules.

Recently, effort has been exerted to utilize the information available in ontologies out of the scope of the traditional ILP settings. There are 3 main approaches to this problem: introduce learning mechanisms into description logics [1], [3], hybrid languages integrating Horn logic and description logics [7], [10] and learning in a more expressive formalism [9]. Learning in description logics is useful mainly for the refinement of existing description hierarchies; however, here we are constrained to limitations of description logic and therefore e.g. unable to express formulas with a free variable. Therefore the works investigating learning in description logics are valuable especially in their results on dealing with the open-world assumption in learning and in transformations of description logics into some other subset of the first-order logic. Reasoning and learning in hybrid languages attempts to loosely couple descriptions of concept hierarchies in description logics with rules expressed in function-free Horn logic. Learning in hybrid languages is often split into two phases, each utilizing well-known algorithms particular to each of the respective formalisms. Reasoning in hybrid languages is more complex than reasoning in both its constituent formalisms. E.g. even if reasoning in the chosen subsets of both DL and HL formalisms separately is decidable, reasoning in the corresponding hybrid formalism may be undecidable [6]. A growth in computational complexity is obviously also a deficiency of approaches using representation formalisms with expressivity exceeding that of first-order logic.

This paper concentrates on the problem of exploiting taxonomies in the framework of propositionalization of relational data by constructing relational features and subsequent learning from the propositionalized representation. We exploit the term and predicate taxonomies in the process of relational feature construction, by adopting the formalism of sorted logic for feature representation and by adapting a refinement operator for first-order features to be co-guided by the taxonomies. In contrast to state-of-the-art logic-based propositionalization systems [5], we explicitely store the information that a feature has been obtained by specializing another feature. The propositional algorithm utilizes the resulting feature taxonomy to prevent the exploration of a conjunction containing a feature together with any of its subsumees and to specialize a rule by replacing a feature by its subsumee. Since the feature taxonomy is determined partly by θ-subsumption, it can be exploited whether or not relation and term taxonomies were available.

2 Sorted Logic

Our approach to propositionalization is based on RSD system [14]. In RSD, a predicate declaration assigns a type symbol to each argument, from a finite set of type symbols. The present approach replaces the notion of type with that of *sort* borrowed from the formalism of sorted logic, which is suitable for encoding term taxonomies. We shall introduce sorted logic and its use by an example. The Gene Function Ontology declares a concept `binding` and its subconcept `protein_binding`. Such concepts are reflected by terms in ILP. It is possible to declare in background knowledge e.g.

```
subclass(binding, protein_binding).
```

```
geneFunction(G, F1) :- geneFunction(G, F2), subclassTC(F1, F2).
```

(where `subclassTC/2` is defined as the transitive closure of `subclass/2`). Unfortunately, in such an approach, for the following two exemplary clauses (hypotheses)

$C =$ `activeGene(G):- geneFuction(G, binding).`
$D =$ `activeGene(G):- geneFuction(G, protein_binding).`

it does not hold $C\theta \subseteq D$, so clause D is not obtained by applying a specialization refinement operator onto clause C. Similar reasoning applies to taxonomies on relations (predicates).

Sorted logic contains in addition to predicate and function symbols also a disjoint set of sort symbols. A sort symbol denotes a subset of the domain called a sort [2]. A *sorted variable* is a pair, $x\!:\!\tau$, where x is a variable name and τ is a sort symbol. Semantically, a sorted variable ranges over only the subset of the domain denoted by its sort symbol. The semantics of universally-quantified sorted formulas can be defined in terms of their equivalence to ordinary formulas: $\forall x\!:\!\tau\ \phi$ is logically equivalent to $\forall x : \neg\tau(x) \vee \phi^{'}$ where $\phi^{'}$ is the result of substituting x for all free occurrences of $x\!:\!\tau \in \phi$.

A *sort theory* Σ is a finite set of formulas containing function formulas and subsort formulas. A *function formula* has the form

$$\forall x_1, \ldots, x_n \tau_1(x_1) \wedge \ldots \wedge \tau_n(x_n) \rightarrow \tau(f(x_1, \ldots, x_n)) \qquad (1)$$

where, in this paper, we constrain ourselves to $n = 0$, thus reducing function formulas to the form $\tau(f)$ reflecting that constant f is of sort τ. A *subsort formula* has the form

$$\forall x \tau_1(x) \rightarrow \tau_2(x) \qquad (2)$$

reflecting that τ_1 is a direct subsort of τ_2. It is required that the directed graph corresponding to the subsort theory is acyclic and has a single root denoted *univ*.

For a sort theory Σ, a *Σ-sorted substitution* is a mapping from variables to terms such that for every variable $x\!:\!\tau$, it holds that $\Sigma \models \forall\tau(t)$ where t is $(x\!:\!\tau)\theta$, where θ is the sorted substitution. Informally, this is a substitution that does not violate the sort theory.

In the present propositionalization approach, terms in features are constants or sorted variables. Backround knowledge consists of an ordinary first-order theory and a sort theory Σ. A *declaration* for a predicate of symbol π and arity n has the form

$$\pi(m_1\tau_1, \ldots, m_n\tau_n)$$

where $m_i \in \{+, -\}$ denotes whether i-th argument is an input (+) or an output (-). Besides the constraints imposed on features in RSD, correct features must respect the sort relationships. Formally, a literal *Lit* may appear in a feature only if there is a declaration $\pi(m_1\tau_1, \ldots, m_n\tau_n)$ and a Σ-sorted substitution θ such that $\pi(\tau_1, \ldots, \tau_n)\theta = Lit$. Next we turn attention to the refinement operator through which features are constructed.

3 Feature construction and adaptation of learning

3.1 Refinement

We have adapted the *sorted downward refinement* from [2], which accounts for term taxonomies, to further account for the earlier defined feature constraints and predicate declarations used in propositionalization, and for a further kind of taxonomy – the *predicate taxonomy* – often available in ontology data. This taxonomy is encoded through meta-predicates in the form

`subrelation(`$pred_1/n$`, `$pred_2/n$`).`

providing the explicit meta-information that goal $pred_1(Arg_1, \ldots, Arg_n)$ succeeds whenever goal $pred_2(Arg_1, \ldots, Arg_n)$ succeeds, i.e. $pred_1$ is more general. The directed graph corresponding to the entire set of the `subrelation/2` statements (where direction is such that edges start in the more general goal) is assumed to be a forest. The set of its roots is exactly the set of predicates declared through the predicate declarations defined in the previous section. It is assumed that the non-root predicates inherit the argument declarations from their respective roots.

As feature heads are fixed in our propositionalization framework, we are concerned with refinement of their bodies, i.e. conjunctions. We will use the notion of an *elementary Σ-substitution.* Its general definition can be found in [2], however, adapted to our framework, the definition simplifies.

An elementary Σ-substitution for a sorted conjunction C is $\{x : \tau_1\} \rightarrow \{x : \tau_2\}$ where $\{x : \tau_1\}$ occurs in C and Σ contains the subsort formula $\forall\psi\tau_2(\psi) \rightarrow \tau_1(\psi)$ for some variable ψ. If $\{x : \tau_2\}$ already occurs in C, then x is deterministically renamed[1] to a variable not occurring in C. Unlike in [2], we can disregard the case of substituting a sorted variable by a function (as we work with function-free features) and, similarly to RSD [14], we neither allow to unify two distinct variables (an equality theory can be defined instead in background knowledge).

Let C be a conjunction of non-negated atoms where any term is either a constant or a sorted variable, Σ be a sort theory, and Δ a set of predicate declarations. We define the *downward Δ, Σ-refinement*, written $\rho_{\Delta,\Sigma}(C)$, as the smallest set such that:

1. For each θ that is an elementary Σ-substitution for $C, \rho_{\Delta,\Sigma}(C)$ contains $C\theta$.
2. Let $\pi(m_1\tau_1, \ldots, m_n\tau_n)$ be a declaration in Δ such that for each i for which $m_i = +$, C contains a variable (denote it x_i) of sort τ_i' which equals or is a subsort of τ_i. Let further $\{x_i | m_i = -\}$ be a set of distinct variables not appearing in C. Then $\rho_{\Delta,\Sigma}(C)$ contains $C \wedge \pi(x_1 : \upsilon_1, ..., x_n : \upsilon_n)$, where $\upsilon_i = \tau_i'$ if $m_i = +$ and $\upsilon_i = \tau_i$ otherwise.
3. Let C contain a literal $pred_1(x_1\tau_1, \ldots, x_n\tau_n)$ and let $pred_2$ be a direct subrelation of $pred_1$. Then $\rho_{\Delta,\Sigma}(C)$ contains C', which is acquired by replacing $pred_1(x_1\tau_1, \ldots, x_n\tau_n)$ with $pred_2(x_1\tau_1, \ldots, x_n\tau_n)$ in C.

[1] That is, we do not allow several elementary substitutions differing only in the chosen renaming.

Propositionalize$(\Delta, B, \Sigma, E, l)$: **Given**, a set Δ of predicate declarations, a first-order theory (background knowledge) B, a sort theory Σ, a set of unary ground facts (examples) $E = \{e_1, \ldots, e_m\}$ and a natural number l; **returns** a set $\{f_1, \ldots, f_n\}$ of constructed features, each with at most l atoms in the body, an elementary subsumption matrix $\mathbf{E}$, an exclusion matrix $\mathbf{X}$, and an attribute-value matrix $\mathbf{A}$ where $\mathbf{A}_{i,j} = 1$ whenever f_i is true for e_j and $\mathbf{A}_{i,j} = 0$ otherwise.

```
1.  $n = 0$; $Agenda$ = a single element list $[(C, 0)]$, where $C$ = true;
2.  If $Agenda = []$: go to 10
3.  $(Curr, Parent) := \mathbf{Head}(Agenda)$; $Tail := \mathbf{Tail}(Agenda)$
4.  If $\mathbf{Nonempty}(Curr)$ and $\mathbf{Undecomposable}(Curr)$:
5.      $n := n + 1$; $f_n = \mathbf{AddFeatureHead}(Curr)$;
6.      $\mathbf{E}_{n,Parent} = 1$; $Parent = n$;
7.      $\mathbf{A}_{n,1 \ldots l} = \mathbf{Coverage}(Curr, E, B, \Sigma, \mathbf{A}_{Parent,1 \ldots l})$
8.  $Rfs := \rho_{\Delta,\Sigma}(Curr)$; $Rfs := \{(Cnj, Parent) | Cnj \in Rfs, |Cnj| \leq l\}$
9.  $Agenda := \mathbf{Append}(\text{Rfs, Tail})$; go to 2
10. $\mathbf{X} = \mathbf{Closure}(\mathbf{E})$
11. **Return** $f_1, \ldots, f_n$, **E**, **X**, **A**
```

Fig. 1. A skeleton of the algorithm for propositionalization through relational feature construction using the sorted refinement operator $\rho_{\Delta,\Sigma}$.

Confined to 8 pages, we skip the proof that, under very general assumptions on Δ, the defined refinement operator is (i) finite, (ii) complete, in that all correct features (as defined in [14] and Section 2) up to variable renaming are enumerated by its recursive closure, whenever the initial C in the recursive application of $\rho_{\Delta,\Sigma}(C)$ is true, and also (iii) non-redundant, in that $\rho_{\Delta,\Sigma}(C_1) \cap \rho_{\Delta,\Sigma}(C_2) = \{\}$ if $C_1 \neq C_2$. However, the operator is not neccessarily correct, in that all its products would be correct feature bodies. In particular, it may produce a conjunction violating the undecomposability condition defined in [14].

3.2 Feature Construction Algorithm and Feature Taxonomy

During the recursive application of the refinement operator, a feature generality taxonomy becomes explicit. For purposes of enhancing the performance of the propositional learning algorithm applied subsequently on the propositionalized data, we pass the feature taxonomy information to the learner through two boolean matrices.[2] Assume that features $f_1, \ldots f_n$ have been generated with corresponding conjunctive bodies $b_1, \ldots b_n$. The *elementary subsumption matrix* $\mathbf{E}$ of n rows and n columns is defined such that $\mathbf{E}_{i,j} = 1$ whenever $b_i \in \rho_{\Delta,\Sigma}(b_j)$ and $\mathbf{E}_{i,j} = 0$ otherwise. The *exclusion matrix* $\mathbf{X}$ of n rows and n columns is defined such that $\mathbf{X}_{i,j} = 1$ whenever $i = j$ or $b_i \in \rho_{\Delta,\Sigma}(\rho_{\Delta,\Sigma}(\ldots \rho_{\Delta,\Sigma}(b_j) \ldots))$ and $\mathbf{X}_{i,j} = 0$ otherwise. In this section we shows how the matrices are instantiated during feature construction. Sec. 3.3 shows how they are utilized by the propositional learner.

A skeleton of the propositionalization algorithm is shown in Fig. 1. The algorithm is a depth-first search generally similar to the feature constructor of RSD

[2] While alternative data structures are of course possible for this sake, the elected binary matrix form requires modest space for encoding (our implementation uses one byte for each 8 matrix elements) and also is conveniently proccessed in the propositional algorithm implementation.

[14]. The main difference lies in using the novel sorted refinement operator $\rho_{\Delta,\Sigma}$ and also in creating the matrices $\mathbf{E}$ and $\mathbf{X}$ storing the generality taxonomy of constructed features. The **Undecomposable** procedure checks whether a feature is not a conjuction of already generated features, through a method used in RSD and detailed in [14]. The **AddFeatureHead** forms a feature clause by formally attaching a head to the body, which consists of the constructed conjunction $Curr$. The **Coverage** procedure verifies the truth value of a conjunction for all examples in E returning a vector of Boolean values. The verification is done by a transformation of the sorted conjunction $Curr$ to an ordinary first-order conjunction as explained in Sec. 2 and then using a standard resolution procedure against a Prolog database consisting of B and Σ. For efficiency, the procedure obtains the coverage $\mathbf{A}_{Parent,1...l}$ of the closest ancestor (subsuming) conjunction whose coverage was tested: any example i such that $\mathbf{A}_{Parent,i}$ is false can be left out of testing as it makes the current conjunction neccessarily false as well. The **Closure** procedure computes the transitive closure of the elementary subsumption relation captured in $\mathbf{E}$ in the manner described in Sec. 3.2, and represents the closed relation analogically in matrix $\mathbf{X}$, in which it further sets $\mathbf{X}_{i,i} = 1$ for all $1 \leq i \leq n$.

3.3 Rule Learning

We have adapted two rule learning algorithms to account for the feature taxonomy information provided by the propositionalization algorithm.

The first algorithm stems from the rule inducer of RSD [14]. It is based on a heuristic general-to-specific deterministic beam search for the induction of a single propositional conjunctive rule for a given target class, and a cover-set wrapper for the induction of the entire rule set for the class. Given a set of features $F = \{f_1, \dots f_n\}$, the standard algorithm refines a conjunction C of features into the set $\{C \wedge f_i | f_i \in F, f_i \notin C\}$. In our enhanced version, the algorithm is provided with the elementary subsumption matrix $\mathbf{E}$ and the exclusion matrix $\mathbf{X}$. Using these matrices it can prevent the useless combination of a feature and its subsumee within the conjunction, and specialize a conjunction by replacing a feature with its elementary (direct) subsumee.

Furthermore, we have similarly enhanced the stochastic local DNF search algorithm introduced in [11] and later transferred into the propositionalization framework by [8]. This algorithm conducts search in the space of DNF formulas, i.e. it refines entire propositional rule sets. Refinement is done by local, non-deterministic DNF term changes detailed in [11]. In our version, the $\mathbf{X}$ matrix is used to prevent the combination of a feature and its subsumee within a DNF term.

4 Experimental Results

We designed experiments to assess the runtime impact of (i) the novel taxonomy-aware refinement operator in the propositionalization process, and (ii) the exploitation of the feature-taxonomy in subsequent propositional learning.

We conducted tests in two domains. The first concerns genomics, where we used data and language declarations from [12]. The nature of this learning task

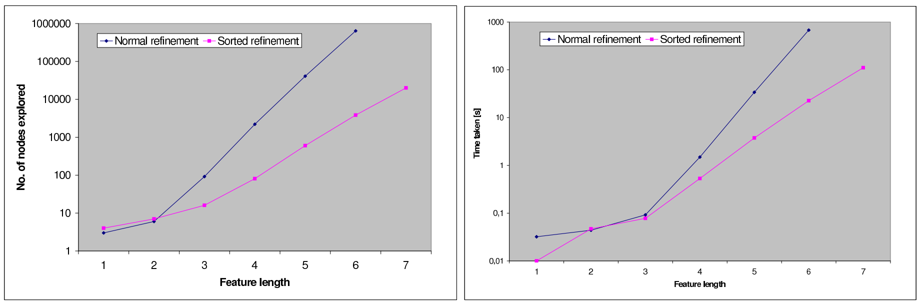

Fig. 2. Sorted refinement vs. standard refinement on CAD and Genomic data. Left: Nodes explored Right: Time taken. (Experiments exceeding 1000s were discarded)

Table 1. Propositional rule learning from CAD and Genomic data

Domain	CAD data		Genomic data	
Algorithm	Time taken	Predict. acc.	Time taken	Predict. acc.
Top-down	0.22 ± 0.08	0.66 ± 0.21	0.99 ± 0.65	0.79 ± 0.13
Top-down, FT	0.06 ± 0.02	0.66 ± 0.22	0.34 ± 0.19	0.76 ± 0.07
SLS	0.63 ± 1.45	0.62 ± 0.18	3.00 ± 2.59	0.79 ± 0.13
SLS, FT	0.28 ± 0.83	0.61 ± 0.19	1.90 ± 1.69	0.76 ± 0.07

has been illustrated in the Introduction and in Sec. 2. The second is concerned with learning from product design data. Here the examples are semantically annotated CAD documents. We used the same learning setting and ontology data as in [13].

Figure 2 illustrates on log scale the number of conjunctions searched (left) and the time spent on search (right) to enumerate all conjunctions true for at least 80% examples, for increasing maximum conjunction size l. Here, we distinguish the sorted refinement operator using a special sort theory Σ encoding the taxonomy information, against the standard refinement operator, which treats the taxonomy information only as part of background knowledge. While in both cases exactly the same set of conjunctions is produced, an order-of-magnitude runtime improvement is observed for the 'taxonomy-aware' operator.

Table 1 shows the runtime spent of inducing a rule set by two algorithms (top-down and stochastic) through 10-fold cross validation in two scenarios: in the first, no feature taxonomy information is used by the algorithms, in the second, feature taxonomy is exploited as described in Sec. 3.3. A significant speedup is observed when feature taxonomy is used without compromising the predictive accuracy.

5 Conclusions

In this work we have proposed principled methods to exploit term, predicate and feature taxonomies to increase the performance of propositionalization and subsequent propositional learning. The significance of our work is supported by three factors: (i) order-of-magnitude runtime improvements with no sacrifice in predictive accuracy, (ii) the practical value and common use [5] of the propositionalization strategy to relational machine learning, which was the target of our methodological enhancements, and (iii) the increasing volumes of semantic knowledge representations providing explicit taxonomies. In future work, we plan to extend the scope of meta-information exploitable by refinement operators beyond taxonomy information. For example, principled methods are needed to deal with meta-knowledge such as "relation R is a function" or "binary relation R is symmetrical," etc.

Acknowledgements. This research was supported by the FP6-027473 Project SEVENPRO. The authors would like to thank Peter Flach and Simon Rawles for valuable advice concerning term taxonomies in ILP.

References

1. Badea, L. and Neinhuys-Cheng, S.-W.: A Refinement Operator for Descriptionn Logics. In Cusens, J. and Frich, A. (eds.): Inductive Logic Programming, LNAI 1866, 40-59, Springer (2000)
2. Frisch, A.: Sorted downward refinement: Building background knowledge into a refinement operator for ILP. In ILP, LNAI 1634:104-115. Springer (1999)
3. Kietz, J.-U.: Learnability of Description Logic Programs,12th International Conference, ILP 2002, Sydney, Australia, LNCS 2583, Springer (2002)
4. King, R. D. et. al.: Functional genomic hypothesis generation and experimentation by a robot scientist.*Nature*, 427:247–252 (2004)
5. Krogel, M.-A., Rawles, S. et al.: Comparative evaluation of approaches to propositionalization. In *Proc. of the 13th ILP*, LNAI 2835:197–214. Springer (2003)
6. Levy, A. Y., Rousset, M.-C.: The Limits on Combining Recursive Horn Rules with Description Logics. AAAI/IAAI 1, 577-584 (1996)
7. Lisi, F. A.: Principles of Inductive Reasoning on the Semantic Web: A Framework for Learning in AL-log. In Principles and Practice of Semantic Web Reasoning, LNCS 3703, 118-132, Springer (2005)
8. Paes, A., Zaverucha, G., Železný, F. et al.: ILP through Propositionalization and k-Term DNF Learning. In *Proc. of the 16th Conference on ILP*, Springer (2007)
9. Popelínský, L.: Inductive inference to support object-oriented analysis and design. Frontiers in Artificial Intelligence and Applications 48, IOS Press (1998)
10. Rouveirol, C., Ventos, V.: Towards learning in CARIN-ALN. ILP, LNAI 1866, 191-208, Springer (2000)
11. Rückert, U., Kramer, S.: Stochastic local search in k-term dnf learning. In *Proc. of the 20th ICML*, 648–655 (2003)
12. Trajkovski I. et al.: Relational Subgroup Discovery for Descriptive Analysis of Microarray Data. In Proc. of CompLife 06, Springer (2006)
13. Žáková, M., Železný, F. et al.: Relational Data Mining Applied to Virtual Engineering of Product Designs. In *Proc. of ILP 06*, Springer (2007)
14. Železný, F. and Lavrač, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62: 33-63 (2006)