Relational Data Mining Applied to Virtual Engineering of Product Designs

Monika Žáková[1], Filip Železný[1], Javier A. Garcia-Sedano[2], Cyril Masia Tissot[2], Nada Lavrač[3,4], Petr Křemen[1], and Javier Molina[5]

[1] Czech Technical University, Prague, Czech Republic
[2] Semantic Systems, Av. Del Txoriherri 9, Derio, Spain
[3] Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[4] University of Nova Gorica, Nova Gorica, Slovenia
[5] Fundiciones del Estanda, Beasain (Gipuzkoa), Spain

Abstract. Contemporary product design based on 3D CAD tools aims at improved efficiency using integrated engineering environments with access to databases of existing designs, associated documents and enterprise resource planning. The ultimate goal of this work is to achieve design process improvements by applying state-of-the-art ILP systems for relational data mining of past designs, utilizing commonly agreed design ontologies as background knowledge. This paper demonstrates the utility of relational data mining for virtual engineering of product designs through the detection of frequent design patterns, enabled by the proposed baseline integration of hierarchical background knowledge (a CAD ontology) using sorted refinements.

1 Introduction

Despite considerable successes of ILP in various knowledge discovery problems such as in bioinformatics [7], industrial applications of ILP have been relatively rare. Although the usefulness of ILP has been demonstrated in areas such as finite element mesh design [4], we are not aware of industrial software employing ILP technologies in regular real-life practice. Engineering, as a knowledge-intensive activity, has great potential for ILP. Consider product engineering which involves diverse knowledge types including CAD structures, technical specifications, and standards. In addition, knowledge about the electrical, mechanical, thermodynamic and chemical behavior may be made available, supported by means of empirical data and simulation models. The abundance of data and knowledge motivates the application of ILP to solving numerous relational data mining tasks. For example, discovering design substructures frequently occurring in a corporate CAD repository would allow to establish their easily invocable templates, with a potential of eliminating repetitive designing work.

This paper reports on the approach developed in the SEVENPRO[1] project which aims at developing a semantic virtual engineering environment for product

[1] SEVENPRO, Semantic Virtual Engineering Environment for Product Design, is the project IST-027473 (2006-2008) funded under the 6th Framework Programme of the European Commission. The authors acknowledge the support by this project.

S. Muggleton, R. Otero, and A. Tamaddoni-Nezhad (Eds.): ILP 2006, LNAI 4455, pp. 439–453, 2007.

design, extending traditional CAD tools with semantic web, virtual reality and relational data mining (RDM) technologies. In one of the tasks the aim is to improve the effectiveness of the search for typical patterns stored in design command chains[2]—conducted for a product of a certain class—thus making explicit the tacit knowledge of an experienced engineer. Other objectives like relational classification, clustering or outlier detection are also motivated in this domain, including rather unorthodox tasks such as learning to match between a formalized product requirement set with an appropriate product design, where both the requirements and designs are represented in relational database formalisms.

The information available in CAD files, associated documents, enterprise resource planning (ERP) and other data sources can be formalized and combined by means of a semantically enriched layer of meta-information (i.e., semantic annotation) based on ontologies. Semantic annotations of CAD designs can be generated automatically from commands histories available via an API of a CAD tool, based on a CAD ontology. These annotations, including the ontology of CAD items, typically encoded in the RDF format, can be automatically transformed to Prolog files containing an ontology of CAD items, axioms and data.

There are three main challenges for ILP due to the use of ontologies in the background knowledge, corresponding to hierarchies of concepts, hierarchies of relations and representation conversion (between Prolog and other knowledge representation languages). *SubclassOf* is a core ontological relation corresponding to taxonomy on terms. Therefore an efficient handling of term taxonomies has to be integrated into the employed RDM systems. The RDF formalism also allows to define hierarchies on relations by means of the *subpropertyOf* relation. To exploit the subproperty relation directly, RDM systems would have to deal with taxonomies of predicates.

Motivated by the virtual engineering of product designs application domain, this work focuses both on what ILP can offer to SEVENPRO problem solving, but also on foundational ILP research challenges motivated by SEVENPRO engineering problems. One of these ILP challenges is the effective use of term/predicate taxonomies which have been, to the best of our knowledge, not commonly addressed in ILP. This work therefore focuses on the technique of taxonomy-exploiting search space structuring, which underlies most other specific SEVENPRO RDM tasks such as frequent pattern mining, classification, feature construction, and design clustering.

This paper is organized as follows. Section 2 provides background for our work by introducing the application domain. Section 3 outlines the role of RDM in the SEVENPRO project and then deals more specifically with RDM of CAD data. In section 4 we describe integration of taxonomies into ILP techniques. Section 5 describes experiments that we conducted and their results. The last section contains conclusions and future work.

[2] A design is obtained by successive applications of CAD commands, such as *extrusion*, *rotation*, etc., which are mutually parametrically related. Various command sequences may lead to the same design, while differing greatly in quality respects, such as complexity, reusability, etc.

2 Background

This section first introduces the application domain, followed by the state-of-the-art in ILP.

2.1 Semantic Virtual Engineering for Product Design Environments

Engineering is one of the most knowledge-intensive activities that exist. More specifically, product engineering has been a key to the development of a strong and specialized manufacturing industry in Europe, organized in RTD departments and technical offices. Product engineering deals with very specific knowledge types, like product structures, CAD designs, technical specifications, standards, and homologations. Moreover, specific electrical, mechanical, thermodynamic and chemical knowledge may include empirical data, simulation models and Computer-aided engineering analysis (CAE) tools that serve to optimise relevant features of the design. The result is rich and complex knowledge stored in many heterogeneous formats, of which probably CAD, documentation and ERP/database are the most frequently found, and which constitute the focus of the SEVENPRO project. The project addresses the most important problem encountered by engineering teams: the effective reuse of knowledge and past designs.

Most engineering teams currently have to access heterogeneous information sources from different tools which, in most cases, do not interoperate. The development of a new product, or a product order with high level of customization, requires a new engineering approach. During the development process, engineering staff works out new product item designs by means of CAD tools. CAD designs contain vast amounts of implicit knowledge about the way experienced engineers design very specialized parts. Efficient reuse of knowledge depends on appropriate organization of information and the capability of retrieving it for later use. Engineering teams still have to spend lots of time trying to find existing designs from a vast CAD repository; in many occasions, they design again and again items very similar to others already existing. Moreover, the different types of knowledge described are supported by different systems, which are used separately and have no communication with each other, like CAD tools, ERP systems, documents and spreadsheets, etc. This situation is illustrated in Figure 1(left).

To efficiently retrieve information, it is necessary to be able to carry out complex searches, combining geometrical, technical and managerial aspects. This would allow the engineer, for example, to query about "parts of type clamp [itemFamily], with more than 6 holes [Geometry], set in order later than November/2004 [Management], compliant with ISO-23013 [Documentation]". This is not possible with current information systems, unless an expensive and complex Product Lifecycle Management (PLM) system is set up, whose maintenance in terms of information updates is burdensome for every company and simply unaffordable in terms of cost for SMEs. The only feasible approach to this is by using semantic-knowledge technologies and a well automated semantic-annotation

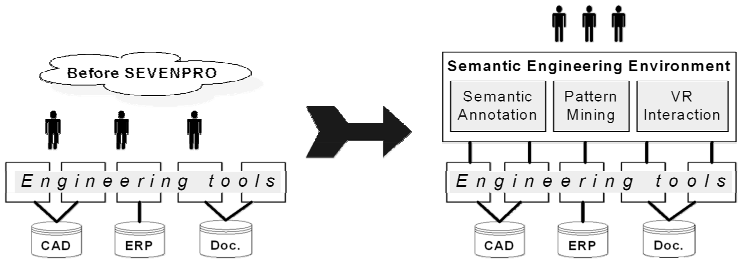

Fig. 1. Engineering heterogeneous information sources before and after SEVENPRO

system from the different information sources, able to extract from them all the knowledge that is useful for the engineering activity. In order to achieve this, an integrated architecture is required, able to extract and maintain a layer of semantic annotations from the different information sources targeted, namely ERP, CAD and Document repositories. As shown in Figure 1(right), a novel semantic virtual engineering product design scenario aims at a better integration and reuse of design knowledge.

2.2 ILP Background: Relational Data Mining by RSD

The RSD relational data mining system [16] enables the discovery of interesting relational subgroups from data (facts) and relational background knowledge. In the engineering design context, an example of a relational subgroup description is e.g.:"a structure containing two co-centric cylinders" (here two substructures of a structure with the mutual relation of co-centricity). A subgroup is then a set of all designs complying with the above description. An interesting subgroup is one that is sufficiently large and in which the distribution of values of a chosen attribute of interest substantially differs from the distribution in the entire data set. For example, the attribute of interest may be the functional category of the design. Similar to Aleph [12], RSD is controlled through command line and accepts data and background knowledge in the Prolog syntax. A distinguishing point of RSD is that it tackles the relational mining task by an (approximate) conversion into a non-relational (propositional) data mining task by constructing a set of truth-valued relational features. The technique (known as propositionalization) implemented in RSD is not limited to the task of subgroup discovery and can be used to transform relational design descriptions into forms that can be used as input to other data mining techniques (e.g. those in WEKA [15]), whose outputs can then be back-converted and interpreted as relational non-recursive patterns/models.

The task addressed in this work is discovery of interesting design patterns describing detected groups of frequent design sequences. Such discovery is made from stored designs and background knowledge in the form of a CAD ontology.

The patterns are also converted to Boolean features using method of propositionalization described in [16] and used for classification by propositional methods.

The usefulness of ILP-generated patterns as attributes for propositional methods has been described e.g. in [14], where ILP-generated attributes were used in addition to expert attributes for regression. The background knowledge used in this work does not contain any hierarchical information. Since in our work we are using only function-free Horn clauses, the generating relational patterns in our work is similar to discovery of frequent patterns in DATALOG described in [3]. This work deals with hierarchical background knowledge by means of `is_a` predicate, providing it in form of facts e.g. `is_a(1001,BSC_disturbance)`, `is_a(BSC_disturbance,BSC_alarm)`. The hierarchy does not distinguish between "subclass" and "member of" relations. The level-wise algorithm for frequent pattern discovery is used. Expressing class hierarchy using `is_a` predicate also appears in [2]. Ceci and Appice compare classification using multi-level association with propositional classification using association rules having only class label in the head converted into Boolean attributes. We are adopting a similar approach to propositional classification, however we are extending the used hierarchical background knowledge to hierarchy on predicates.

3 Relational Data Mining Applied to the Discovery of Product Design Patterns

This section first outlines the role of RDM in the SEVENPRO project. Then a more detailed description of data from CAD designs is provided.

3.1 Overall RDM Role in the SEVENPRO Project

The discovery of patterns from engineering knowledge repositories is expected to be an important facility for reusing engineering knowledge. The amount of data and its availability through the use of several independent tools has always been an obstacle for such reuse. Coupling a unified view of the available knowledge through commonly agreed ontologies with the capabilities of mining the information gathered in various engineering resources will broaden the range of actually reusable engineering knowledge. In particular, it is foreseen that the sequences of CAD design operations (design features) can be exploited to obtain from them abstract design patterns. These patterns (after human revision) can be reused as corporate design standards or recommendations

- to support the work of engineers (reusability),
- to check pattern compliance of new designs (quality checking), and
- to teach novel engineers on how to design specific parts (training).

Engineering designs capturing implicit expert knowledge have a relational nature: they cannot be efficiently described by attribute tuples. Rather, flexible-size structural descriptions are needed, specifying various numbers of primitive objects as well as relations between them. To discover and explicitly define knowledge from such data by means of relational patterns, RDM algorithms are needed.

The status of the current SEVENPRO developments is presented in Sections 4 and 5 of this paper.

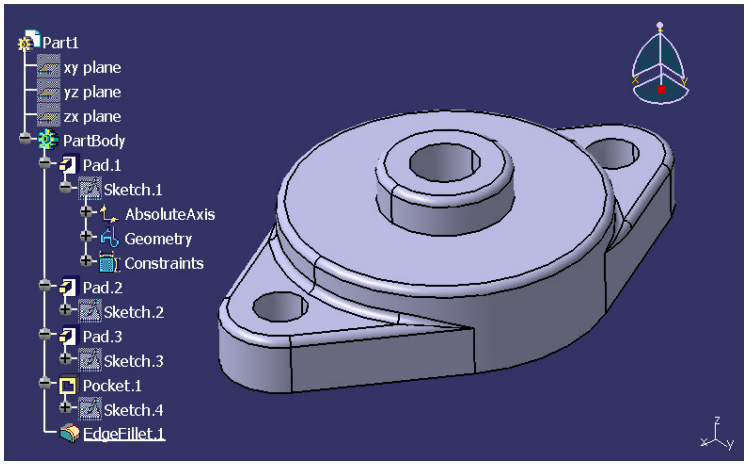

Fig. 2. Example of a CAD design including commands history

3.2 CAD Data Used for Relational Data Mining

The SEVENPRO ontologies and the corresponding annotations cover a large spectrum of engineering concepts (items, orders, norms, problems, versioning, among many others). As mentioned, this allows for complex queries across the available knowledge. An important facet of this knowledge is the CAD design information. Engineering departments generate a large amount of CAD files. Such files can be 3D part-definition files, 3D assembly definition files or 2D drafting files. In addition, relevant information ranges from textual data (like block, operation or part names) and generic document structure (like assembly structure), to detailed design information in the case of 3D parts. In the later case, the shape of a 3D part is the result of sequence of operations specified by the designer. This sequence of design operations (design features) is where most of the designer's knowledge resides, as it is a reflection of the designer's experience.

Figure 2 represents a simple mechanical part, a two bolt flange. Notice the command history (at the left-hand side of the figure) leading to the particular virtually designed object. In command histories the basic operations are "creating" matter (e.g., a pad, a stiffener) and "removing" matter (e.g., a chamfer, an edgeFillet).

This design history conveys the higher level information on how the object was designed as well as high level dimensional information, as the commands have parameters associated to them (like the height of an extrusion or the radius of a fillet). This information would be more difficult to determine using only the final shape of the part. Having it associated to the operation not only makes it easily accessible but also keeps its real meaning. The design history, presented at the left-hand side of Figure 2, is depicted in the annotation layer as a design sequence in terms of ontology classes and instances, as shown in Figure 3.

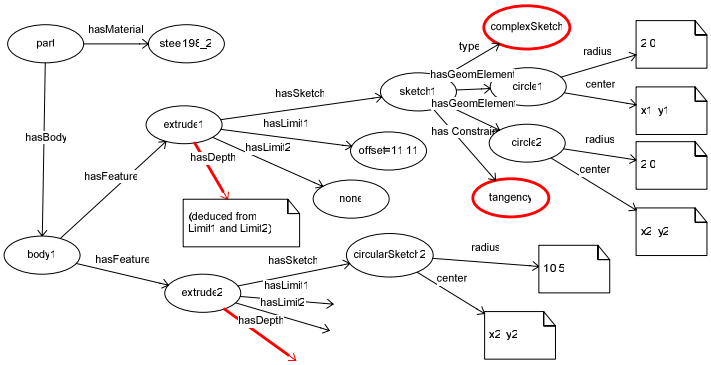

Fig. 3. Part of a semantic annotation of the design shown in Figure 2

This kind of highly relational data exists for all the annotated files, and is the input to a RDM algorithm. The generated instance schema is simplified with respect to the internal CAD representation. For example, if a sketch does not belong to any predefined category, it is identified as a complexSketch and it is not further elaborated. The schema also contains some properties derived from other properties, e.g. property hasDepth of extrude is derived from the two limits. In SEVENPRO, this representation has been converted into Prolog facts, more suitable as input for the RDM algorithms. An example of Prolog facts describing part of CAD design is presented below.

```
hasCADEntity('eItemT_BA1341',part_183260395_10554).
typeOf('eItemT_BA1341', eItemT).
typeOf(part_183260395_10554, cADPart).
hasBody(part_183260395_10554,body_183260395_10555).
typeOf(body_183260395_10555, body).
hasFeature(body_183260395_10555,extrude_183260395_10556).
typeOf(extrude_183260395_10556, extrude).
hasSketch(extrude_183260395_10556,complexSketch_183260395_10557).
typeOf(complexSketch_183260395_10557, complexSketch).
hasGeomElem(complexSketch_183260395_10557,circle_183260395_10558).
typeOf(circle_183260395_10558, circle).
hasDepth(extrude_183260395_10556,0).
```

4 ILP Advances: Integration of Taxonomies

The ontological background knowledge currently available in the described CAD domain is represented in the RDF formalism [11]. The ontology can be represented by an acyclic directed graph (DAG). Concepts are defined only by means of declaring class and its place in the class hierarchy. No set operators or restrictions commonly used in OWL are present in the background knowledge and

dataset. Only the domain and range are defined for the properties and a hierarchy on properties is induced by means of the `subpropertyOf` relation. The definition of `rdfs:subPropertyOf` relation in [11] originally states: If a property P is a subproperty of property P', then all pairs of resources which are related by P are also related by P'. For our purposes the definition of subPropertyOf relation is restricted to cases where domain and range of P and P' are defined by some class or set of classes. Then it must hold that domain of P is equivalent to or subclass of the domain of P' and the same holds for range.

Therefore we have to deal essentially with taxonomies on terms and predicates. Our baseline approach for integration of these taxonomies into RDM is based on the refinement operator proposed in [6].

4.1 Sorted Downward Refinement

The background knowledge built into this refinement is based on sorted logic, which encodes the taxonomies. 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 [6]. 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 sentences can be defined in terms of their equivalence to ordinary sentences: $\forall x : \tau\phi$ is logically equivalent to $\forall x : \neg\tau(x) \vee \phi^{'}$ where $\phi^{'}$ is the result of substituting x for all free occurrences of $x : \tau \in \phi$.

The background knowledge that is to be built into the instantiation, subsumption and refinement of sorted clauses is known as a sort theory. A sort theory is a finite set of sentences that express relationships among the sorts and the sortal behavior of the functions. Sentences of the sort theory are constructed like ordinary sentences of first-order predicate calculus except that they contain no ordinary predicate symbols; in their place are sort symbols acting as monadic predicate symbols. In [6] the form of the sort theory is restricted to two types of sentences: function sentences and subsort sentences.

Function sentence

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

Subsort sentence

$$\forall x \tau_1(x) \rightarrow \tau_2(x).$$

Graph of the sort theory has to be acyclic and singly rooted. In our task we are not dealing with functions, therefore the only type of sort theory is restricted to subsort sentences. As was stated above the background knowledge is acyclic and since no multiple inheritance is used, graph of the background knowledge is a DAG, graphs for the individual sorts are trees. For the sort theory special substitution is defined [6]:

Definition 1. *Sorted substitution θ is a Σ-substitution if for every variable $x : \tau$, it holds that $\Sigma \models \bar{\forall}\tau(t)$ where t is $(x : \tau)\theta$.*

In [6] it was proved there that the sorted downward refinement is finite for finite set of predicate symbols and that it is correct and complete.

4.2 Extending θ-Subsumption with Σ-Substitution

We have extended the traditional θ-subsumption with Σ-substitution obtaining a refinement operator with three substitution rules:

1. specialization through changing the type of a variable to its direct subclass (based on Σ-substitution),
2. specialization through adding a literal (traditional θ-substitution),
3. specialization through replacing predicate P by a predicate P', where it holds subPropertyOf(P',P).

In addition to the two new specialization rules, specialization through adding a literal was extended, so that the types of input variables of a literal to be added can be supertypes of some already used variables. This was done to accommodate for situation similar to the following example: Conjunction created so far is `hasCADEntity(X1:cADFileRevision,X2:cADPart),hasBody(X2:cADPart, X3:body),hasFeature(X3:body,X4:extrude),hasDepth(X4:extrude, X5:length)`, the literal to be added is defined by `hasValue(+literalValue, -float)` (i.e. by predicate `hasValue` with input argument of type `literalValue` and output argument of type `float`). In the background knowledge it is stated that `length` is a subclass of `literalValue`. Therefore the predicate `hasValue` can be added to the conjunction, creating new conjunction: `hasCADEntity(X1:cADFileRevision,X2:cADPart),hasBody(X2:cADPart, X3:body),hasFeature(X3:body,X4:extrude),hasDepth(X4:extrude, X5:length),hasValue(X5:length,X6:float)`. We have not included substitution of variables by constants so far, since in data mining from CAD designs we are currently focusing on the structure of similar designs, not on numeric values of parameters.

4.3 Pattern Discovery

The pattern discovery task using the sorted refinement operator is approached through constructing first-order features. An overview of the system is shown in Figure 4. The ILP system generates features with user-defined maximal length and minimal support. The generated features are connected subgraphs of generalizations of the graphs describing the individual examples. Since the graphs describing the examples are not trees and there are relations connecting variables at the same variable depth, reuse of variables within the features is necessary, i.e. one variable can be used either as input or output variable of several predicates within one feature. An example of relation connecting variables at the same variable depth is `appliesTo(fillet, cADFeature)` in the following example

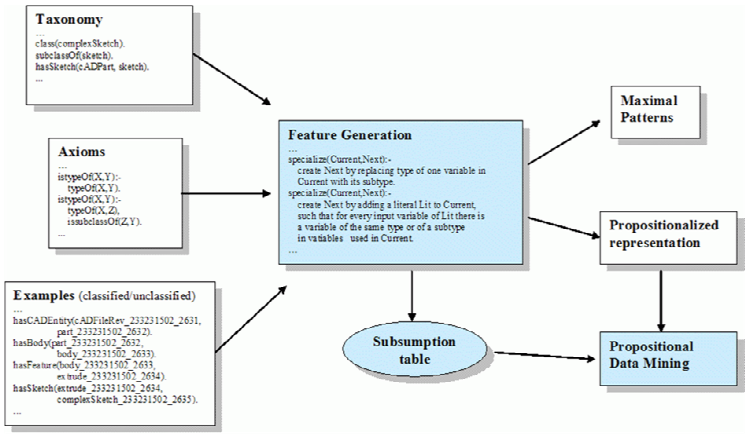

Fig. 4. An overview of the RDM system

`hasCADEntity(X1:cADFileRevision,X2:cADPart),hasBody(X2:cADPart,X3:body), hasFeature(X3:body,X4:cADFeature), next(X3:body,X5:fillet), appliesTo(X5:fillet,X4:cADFeature).`

Depth-first search is used to generate the features. To prevent generation of irrelevant features, the coverage of each feature is computed immediately after the feature is generated. Features with coverage lower than the minimal required support are pruned and not refined further. To prevent generating features that are permutations of features already generated, an explicit order on predicates and concept types is defined and enforced in each feature. Ordering of predicates is checked for the set of literals with the same variable depth of input variables. Moreover, in case of multiple use of the same predicate with same input variables and output variables of the same type, subtree rooted at each occurrence of the predicate has to be smaller than subtree rooted at previous occurrences of this predicate. Total order on the subtrees is induced by order defined on predicates. Therefore the search is complete even with ordering.

An example of a discovered feature, which was the single most important feature for description of the class `itemFamilyLiner`, can be seen below. It contains variables of types at different levels of granularity e.g. `cADFeature` is 2 levels higher in the hierarchy of features than `fillet`.

```
f(X1:eItemT):- hasCADEntity(X1:eItemT,X2:cADPart), hasBody(X2:
cADPart,X3:body), hasFeature(X3:body,X4:pocket), hasSketch(X4:
pocket,X5:complexSketch), hasGeomElement(X5:complexSketch,X6:
circle), next(X4:pocket,X7:fillet), next(X7:fillet,X8:cADFeature)
```

During the feature generation, a table of feature subsumption pointing to all ancestors of the feature is maintained. This is similar to the approach employed in SPADA [1]. This subsumption table is exploited for pruning of features for propositionalization. The subsumption is also exploited in propositional pattern search, which prunes any conjunctions of a subsumer with its subsumee and

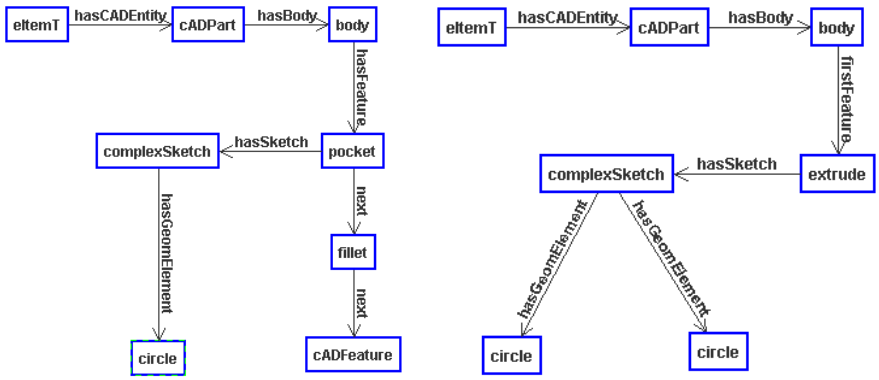

Fig. 5. Examples of discovered features

specializes a conjunction not only by extending it, but also by replacing an included feature with its subsumee.

4.4 Feature Visualization

To improve RDM usability both for users and for developers, graphical visualization of features is useful. A tool based on the JGraph library has been developed. As the input data schema for RDM contains only unary and binary predicates, we can restrict our attention to oriented directed acyclic graphs (DAGs). Nodes in such a DAG represent undistinguished variables labeled by sort atoms, while edges represent binary predicate atoms.

In Figure 5 an example of discovered patterns is shown. The two patterns in the figure simultaneously cover 44 of 62 instances classified as `itemFamilyLiner`, while they also cover 3 other (non-`itemFamilyLiner`) instances. Accordingly, the class `itemFamilyLiner` can be well described as a set of instances having as a starting feature a two circle extrusion, while having also another feature – a circled pocket followed by a fillet and another feature.

5 Experimental Results

Experiments were performed on a dataset containing 160 examples of CAD design drawings provided by a metal casting company that participates in the project: Fundiciones del Estanda. Two main types of experiments were run:

- searching for relational patterns present in all examples of a given class, to compare efficiency of the sorted refinement enriched RDM to a baseline ILP system, and
- classification based on constructing propositional features to evaluate the predictive accuracy of the propositionalisation approach to classification.

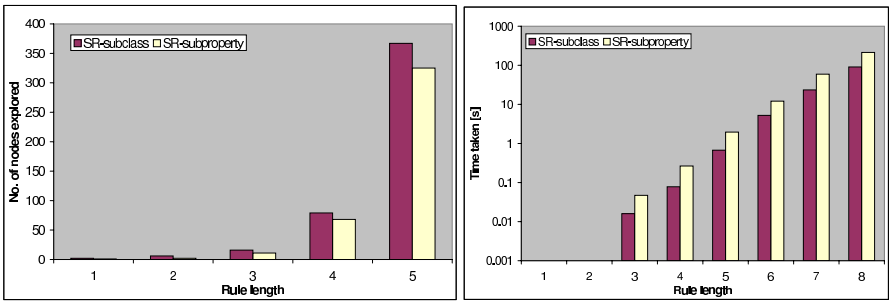

Fig. 6. Comparison of sorted refinement with and without using taxonomy on predicates. Left: Number of nodes explored Right: Time taken.

5.1 Comparison of Sorted Refinement with Aleph

We conducted experiments to compare the efficiency of RDM including sorted refinement (SR) on one hand and a standard ILP system on the other hand. The baseline ILP system chosen for comparison was Aleph. The specific goal of the experiment was to determine the volumes of search space traversed by the respective systems in order to find patterns covering all of the provided positive examples.

The majority class of examples is considered as positive. For the sake of this experiment, no negative examples are needed. There were 57 examples, where each example contained a description of one CAD design drawing. Around 100 predicates were used to describe each example.

The tests were performed for pattern length from 1 to 8. For pattern length greater than 7, pattern generation was no longer tractable for Aleph. In the first set of experiments only term subsumption was used in our system. It can be seen that the number of expanded nodes is decreased very significantly. In the second set of experiments, predicate subsumption was used in our system as well. Results of these experiments can be seen in Figures 6 and 7. Figure 6 shows results of using sorted refinement with and without predicate subsumption. Figure 7 shows results of both our approaches compared to Aleph. The time taken for evaluation roughly doubles w.r.t. experiments using term subsumption only. The number of explored nodes decreases, however the decrease is not very significant. This is due to the fact that the subproperty relation hierarchy that was used has only two levels and includes around 10 predicates. Our system can be used for pattern sizes, which are intractable in Aleph. This is important, because it has been discovered that patterns with length less than 7 do not provide information sufficient for classification.

5.2 Classification Based on Propositional Features

For the data set containing 160 design drawings their classification was provided. Examples were classified into 4 proper classes describing families of designs and

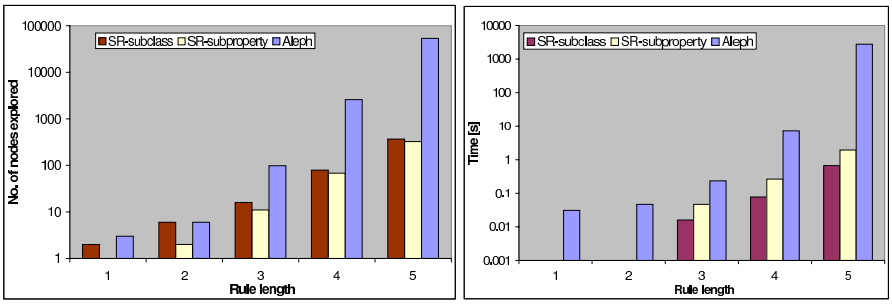

Fig. 7. Comparison of sorted refinement and Aleph. Left: Nodes explored Right: Time taken.

57 examples that did not belong to any of the 4 classes were classified as 'other'. By consultation with the users it was found out that the first feature used is important and also relative order of the features is important. Therefore properties describing the order of CAD features were added to background knowledge and to annotations e.g. `next(+cADFeature,-cADFeature)`, `sequenceStart` and `firstFeature(+body,-cADFeature)`. The following relations were also added to the background knowledge:
`subpropertyOf(firstFeature,hasFeature)`, `subpropertyOf(hasFeature, sequenceStart)`. Special treatment of relations that are subproperties of `next` and `sequenceStart` was implemented. Subproperties of `sequenceStart` can occur only once in a pattern and for subproperties of `next` order on the level of arguments is not checked.

Our system was used to generate a set of features of length 7. The feature set was then pruned by excluding features covering all examples. Also in case a feature covered the same examples as some of its children, the feature was excluded. More general features are pruned rather than more specific ones, since concepts that are leaves of the class hierarchy are mapped to specific design operations available in CAD systems and therefore are more interesting for the user. Propositional algorithm J48 implemented in WEKA [15] was then used for classification using generated features as attributes. For testing 10 fold cross-validation was used. Results of the classification are summarized in Table 1.

Table 1. Results of classification using the J48 algorithm

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
itemFamilyTT	0.826	0.036	0.792	0.826	0.809	0.9
itemFamilyLiner	0.895	0.068	0.879	0.895	0.887	0.927
itemFamilyStdPlate	0.5	0.02	0.571	0.5	0.533	0.834
itemFamilySlottedPlate	0.8	0.02	0.727	0.8	0.762	0.883
other	0.855	0.071	0.883	0.855	0.869	0.897

The prevailing error type is that some items of class `itemFamilyStdPlate` were incorrectly classified as `itemFamilySlottedPlate`. These two classes are both subclasses of class `itemFamilyPlate` and they are more similar to each other than any other pair of classes. More detailed information or longer features would be necessary to distinguish between these two classes more accurately. Other errors were mostly confusions between one of the proper classes and class 'other'.

6 Conclusions and Further Work

In this paper we have described semantic virtual engineering for product design in engineering environments, which integrates information from heterogeneous sources by means of a semantic layer, and identified the role of relational data mining in this application. As a case study, semantic annotation and RDM on CAD designs was chosen, since CAD designs are challenging from the ILP point of view due to the various length and structure of the description of each example combined with taxonomical background knowledge. We have proposed a baseline approach for integrating taxonomical background knowledge into an ILP system by implementing sorted refinement operator and extending it to include taxonomies on predicates.

The efficiency of our approach was demonstrated by comparing it to standard ILP system Aleph without any support for integration of hierarchical background knowledge. The results were strongly convincing in favor of the former. In terms of the volume of search spaced traversed to find a set of frequent patterns, the 'hierarchy-blind' search conducted by Aleph maintains a roughly exponential overhead w.r.t. the ontology-aware refinement, as the maximum pattern size is being increased. This has a strong consequence in this application domain: working in spaces of patterns of length greater than 7 literals becomes intractable for Aleph, while such and longer patterns are important for capturing common design sequences as exemplified earlier in the text. More experiments comparing frequent patterns discovered with different types of hierarchies will be performed, when the CAD design ontology becomes more fine grained. Then also tests of scalability will be conducted.

Features generated by our system were also used for classification of CAD designs. Generally speaking, the accuracies obtained through cross-validation were surprisingly high, which can be ascribed both to the noise-free character of the data and to the sufficient expressivity of the features our system constructed. Analyzing the prevailing classification error type, it was discovered that the order of CAD design features was important for classification, and thus predicates and rules describing the order of predicates were established.

In future work we will consider a more principled approach of integrating more complex ontological background knowledge, including recursive definitions and multiple inheritance, and the order on predicates. The first approach we will consider in future work is using a hybrid language integrating description logics and Horn logic similar to $\mathcal{AL}$-log [9] and CARIN [8]. Another approach is using

a more expressive formalism such as F logic. We will also closely collaborate with the end users to restrict the form of features.

References

1. Appice, A., Ceci, M., Lanca, A., Lisi, F.A., Malerba, D.: Discovery of spatial association rules in geo-referenced census data: A relational mining approach. Intelligent Data Analysis 7, 541–566 (2003)
2. Ceci, M., Appice, A.: Spatial Associative Classification: Propositional vs. Structural approach. In: Proceedings of the ECML/PKDD 04 Workshop on Mining Spatio temporal Data (2004)
3. Dehaspe, L., Toivonen, H.: Discovery of frequent DATALOG patterns. Data Mining and Knowledge Discovery 3(1), 7–36 (1999)
4. Dolšak, B., et al.: Finite element mesh design: An engineering domain for ILP application. In: Proc. of ILP 1994, GMD-Studien, vol. 237, pp. 305–320 (1994)
5. Donini, F.M., Lenzerini, et al.: AL-log: Integrating Datalog and Description Logics. Journal of Intelligent Information Systems 10(3), 227–252 (1998)
6. Frisch, A.: Sorted downward refinement: Building background knowledge into a refinement operator for ILP. In: Džeroski, S., Flach, P.A. (eds.) Inductive Logic Programming. LNCS (LNAI), vol. 1634, Springer, Heidelberg (1999)
7. King, R.D., et al.: Functional genomic hypothesis generation and experimentation by a robot scientist. Nature 427, 247–252 (2004)
8. Levy, A., Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104, 165–209 (1998)
9. Lisi, F.A., Malerba, D.: Ideal Refinement of Descriptions in AL-Log. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 215–232. Springer, Heidelberg (2003)
10. McGuinness, D.L., van Harmelen, F. (eds.): OWL Web Ontology Language Overview. W3C Recommendation (February 10, 2004) Available online at `http://www.w3.org/TR/owl-features/`
11. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation (February 10, 2004) Available at `http://www.w3.org/TR/rdf-schema/`
12. Srinivasan, A.: The Aleph manual version 4 (2003) (June 7, 2006) Available online at `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`
13. Srinivasan, A., Muggleton, S., et al.: Theories for mutagenicity: A study in first-order and feature-based induction. Artificial Intelligence 85(1-2), 277–299 (1996)
14. Srinivasan, A., King, R.: Feature construction with ILP: A study of quantitative predictions of biological activity aided by structural attributes. In: ILP 1996. LNCS, vol. 1314, pp. 352–367. Springer, Heidelberg (1997)
15. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
16. Železný, F., Lavrač, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62, 33–63 (2006)