Relational Pattern and Subgroup Discovery in CAD documents

Monika Žáková[1], Petr Křemen[1], Javier A. Garcia-Sedano[2], Cyril Masia Tissot[2], Nada Lavrač[3,4], Filip Železný[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. In our recent work [1], we have dealt with frequent pattern discovery in CAD designs using a novel approach to integration of hierarchical background knowledge to relational data mining. This paper extends the previous work by applying the methodology of subgroup discovery which first seeks frequent structural patterns in CAD designs and then discovers interesting (large and class-biased) design subgroups defined in terms of the frequent patterns its members contain. Besides the two mentioned kinds of data mining results, further categories of patterns will be required to mine in the present application domain. The large amount of generated results of each kind calls for establishing an efficient pattern management and search functionality. For this purpose we elaborate an ontology of relational data mining results.

1 Introduction

Inductive logic programming (ILP) [15] is concerned with inductive learning of theories from examples and background knowledge, within the first-order predicate logic framework. ILP is currently the most advanced methodology for relational data mining (RDM) [14].

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 [8], we are not aware of industrial software employing ILP technologies in regular real-life practice. Engineering, as one of the most knowledge-intensive activities, has great potential for ILP applications. Consider product engineering which involves diverse knowledge types including CAD structures, technical specifications, and standards. The abundance of data and knowledge motivates the application of ILP to solving numerous RDM tasks. RDM is an approach to machine learning that looks for patterns involving multiple relations. The relations can be defined extensionally, as lists of tuples, or intensionally, as sets of rules. The latter allows relational data mining to take into account generally valid domain knowledge, referred to as background knowledge.

[14] An example of RDM task is discovering design substructures frequently occurring in a corporate CAD repository. It 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 [6] project, which aims at developing a semantic virtual engineering environment for product design, extending traditional CAD tools with semantic web, virtual reality and 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[7]—conducted for a product of a certain class—thus making explicit tacit knowledge of an experienced engineer. A number of other objectives involving 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 [9], can be automatically translated to Prolog, leading to Prolog files containing an ontology of CAD items, axioms and data.

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 efficient use of term/predicate taxonomies which have been, to the best of our knowledge, not commonly addressed in ILP. An attempt to include hierarchical background knowledge has been made by inducing multi-level association rules in [11]. 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, first-order feature construction, and design clustering.

The RSD relational data mining system [5] 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 substruc-

[6] 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.

[7] 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.

tures 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. Similarly to Aleph [13], 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 a (approximate) conversion into a non-relational (propositional) data mining task by constructing a set of truth-valued first-order 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 a wealth of other data mining techniques (e.g. those in WEKA [12]), whose outputs can then be back-converted and interpreted as relational non-recursive patterns/models.

This paper presents an extension of the RDM tool described in [1]. New features include subgroup discovery based on relational features and negation of features and also negations of literals within a feature.

1.1 Relational data mining applied to the discovery of product design patterns

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. In addition, hierarchical background knowledge is available in form of ontologies. To discover and explicitly define knowledge from such data by means of relational patterns and to utilize the background knowledge, RDM algorithms are needed.

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 is where most of the designer's knowledge resides, as it is a reflection of the designer's experience.

Figure 1(left) 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 information on how the object was designed as well as some information about dimensions, 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

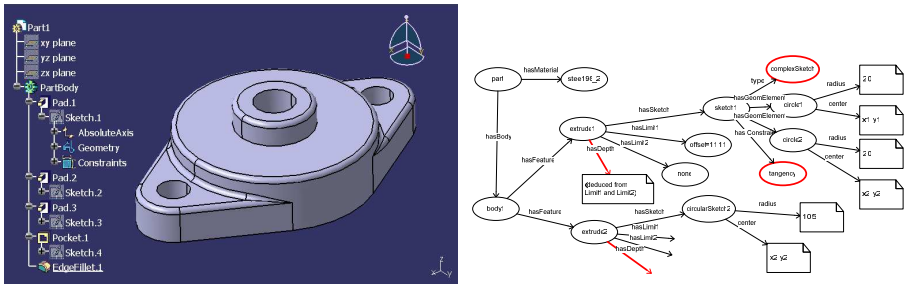

Fig. 1. Left: Example of a CAD design including commands history. Right: Part of a semantic annotation of the design.

of the part, however, 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 1, is depicted in the annotation layer as a design sequence in terms of ontology classes and instances, as shown in Figure 1(right). Only error-prone command series without trial-error-correction sequences are currently used.

This kind of data with rich relational structure 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 in Table 1. Currently the schema focuses on basic operations. Information about restrictions on the design will be dealt with at later stages of the SEVENPRO project.

2 ILP and Integration of taxonomies

The ontological background knowledge currently available in the described CAD domain is represented in RDF formalism. 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 [10] are present in the background knowledge and dataset. Only 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 [9] 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

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).
hasGeomElement(complexSketch_183260395_10557,circle_183260395_10558).
typeOf(circle_183260395_10558, circle).
hasDepth(extrude_183260395_10556,0).
hasFeature(body_183260395_10555,pocket_183260395_10580).
typeOf(pocket_183260395_10580, pocket).
hasSketch(pocket_183260395_10580,complexSketch_183260395_10581).
typeOf(complexSketch_183260395_10581, complexSketch).

Table 1. Prolog facts describing a part of CAD design.

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 current approach for integration of these taxonomies into RDM is based on the refinement operator proposed in [3].

2.1 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. 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 [3] 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 sort theory is restricted to subsort sentences. As was stated above the background knowledge is acyclic and since no multiple inheritance is used, graphs for the individual sorts are also singly rooted. For the sort theory special substitution is defined:

Definition 1. *Let Σ be a sort theory. 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 [3] it was proved there that the sorted downward refinement is finite for finite set of predicate symbols and that it is correct and complete.

2.2 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) with variable binding,
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 there 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.

3 Application results

Experiments were performed on a dataset containing 164 examples of CAD design drawings provided by a metal casting company that participates in the

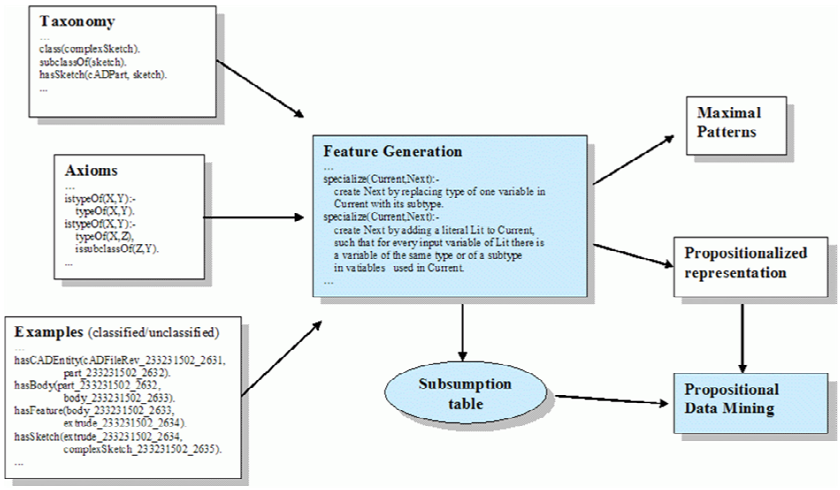

Fig. 2. An overview of the RDM system.

SEVENPRO 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
- subgroup discovery based on constructing relational features.

3.1 Frequent patterns

The pattern discovery task using the sorted refinement operator, which uses Σ-substitution, is approached through constructing first-order features. An overview of the system is shown in Figure 2. The ILP system generates features with user-defined maximal length and minimal support. The generated features are subgraphs of generalizations of the graphs describing the individual examples. Each feature is a connected graph. Since the graphs describing the examples are not trees and there are relations connecting variables at the same variable depth, such as the relation `appliesTo(fillet, cADFeature)`, reuse of variables within the first-order features is necessary, i.e. one variable can be used either as input or output variable of several predicates within one first-order feature.

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.

An example of a discovered feature, which was crucial for description of class *itemFamilyLiner*, can be seen below:

```
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 [11]. This subsumption table is exploited for pruning of features for propositionalization. This subsumption is also exploited in propositional pattern search, which prunes any conjunctions of a subsumer with its subsumee and specializes a conjunction not only by extending it, but also by replacing an included feature with its subsumee.

3.2 Subgroup discovery based on relational features

For the data set containing 164 design drawings their classification was provided. Examples were classified into 4 proper classes describing families of designs and 61 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 CAD feature used is important and also relative order of the CAD features is important. Therefore properties describing the order of CAD features were added to background knowledge and to annotations. These include: `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`. 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 first-order features of length 7. The generated feature set was 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. Then RSD system was used for subgroup discovery using methodology described in [5]. Both the generated features and their negations were used. An unordered set of rules for each target class was induced using weighted covering algorithm with weighted relative accuracy search heuristics. 66% of examples were used for training and 33% for testing set. Cross-validation

Class	No. of examples	ROC Area - Train	ROC Area - Test
itemFamilyTT	23	0.78	0.5
itemFamilyLiner	63	0.83	0.52
itemFamilyStdPlate	8	0.7	0.73
itemFamilySlottedPlate	10	0.1	0.62
other	64	0.7	0.73

Table 2. Results of subgroup dic.

was not used due to small size of the available dataset. An example of the rule generated for class *itemFamilyLiner* can be seen below:

```
   r3: hasCADEntity(X1:eItemT,X2:cADPart),hasBody(X2:cADPart,
X3:body),hasFeature(X3:body,X4:extrude),hasSketch(X4:extrude,
X5:complexSketch),hasGeomElement(X5:complexSketch,X6:circle),
hasGeomElement(X5:complexSketch,X7:geometricElement),
hasFirstLimit(X4:extrude,X8:limitDescriptor)
 and hasCADEntity(X1:eItemT,X12:cADPart),hasBody(X12:cADPart,
X13:body),hasFeature(X13:body,X14:fillet),next(X14:fillet,
X15:pocket),next(X15:..,X16:cADFeature),next(X16:..,X17:cADFeature)
 and hasCADEntity(X1:eItemT,X22:cADPart),hasBody(X22:cADPart,
X23:body),hasFeature(X23:body,X24:extrude),next(X24:extrude,
X25:pocket),hasSketch(X24:extrude,X26:sketch).
```

Results of the experiments are summarized in Table 2. Even though interesting subgroups were discovered, the generalization power of the induced rules is not high enough. This is mainly due to small amount of examples in the dataset.

3.3 Feature visualization

To improve RDM usability both for users and for developers, their graphical visualization 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 DAGs. Nodes in such a DAG represent undistinguished variables labeled by sort atoms, while edges represent binary predicate atoms.

In Fig.3 an example of discovered patterns is visualized. 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.

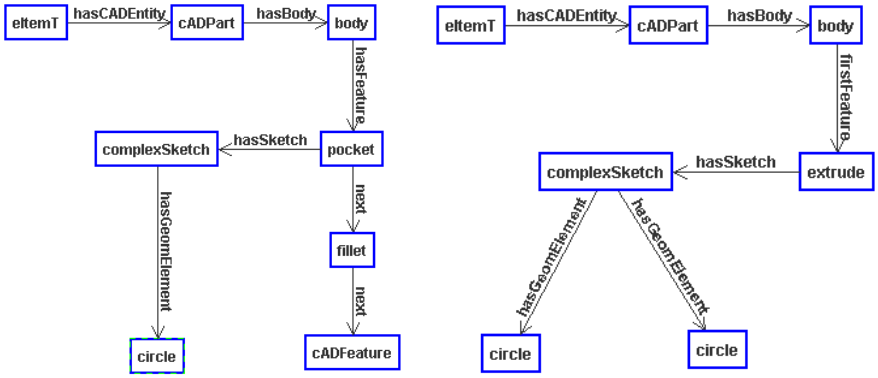

Fig. 3. Examples of discovered features.

4 RDM Results Management

The huge amount and expected diversity of generated RDM results, as discussed in last section, requires an efficient framework for storing and management. For this purpose, an RDM ontology has been developed. This ontology will provide both the logical model of the RDM knowledge base and an interface between the RDM system and the semantic server. The ontology has been designed with respect to two types of queries. Queries of the first type reflect end-user and semantic server requirements like finding all RDM patterns with given support and confidence, finding several classification rules with highest confidence for given example. The second type of queries is intended to support RDM feedback and algorithm tuning, like metric evaluation for clustering algorithms, comparison of two RDM methods w.r.t. their precision, etc.

The ontology core, shown in fig.4, consists of three parts - datasets management, mining results and algorithm configurations. Nodes, representing classes (rectangles) and datatypes (rectangles with cut corner), are connected either by subsumption edges (UML generalization arrows) or by slot links with cardinality restriction (simple arrows with numbers). Datasets (`Dataset`) represent input data for RDM algorithms. To get higher amount of input data, the datasets obtained from the user (`BaseDataset`) can be merged into more complex datasets (`CompositeDataset`). Each dataset, resp. classified dataset (`ClassifiedDataset`), consists of examples (`Example`), resp. classified examples (`ClassifiedExample`). Classified version are derived from these base ones by extension to enable executing unsupervised algorithms on classified data as well. Mining results (`MiningResult`) lie in the core of the ontology. They represent single results of data mining algorithms (clusters, rules, patterns, etc.). Each mining result is related both to the original dataset (relation `hasDataset`) from which it was generated and to the configuration (relation wasProducedBy) of the particular algorithm that was used to generate it. This allows for example tracking of the experiment history. Two special relations, `covers` to describe covering

examples for unsupervised mining results (class UnsupervisedMiningResult) and `classifies` to describe classifying examples for supervised mining results (class SupervisedMiningResult), serve to bind mining results part to the dataset part. The last part consists of algorithm configurations (`AlgorithmConfiguration`) that are expected to be used for RDM feedback queries. To enable automatic RDM tool execution, each algorithm configuration stores (as string) both the exact command to some of the underlying RDM tools and the language/tool in which it shall be executed.

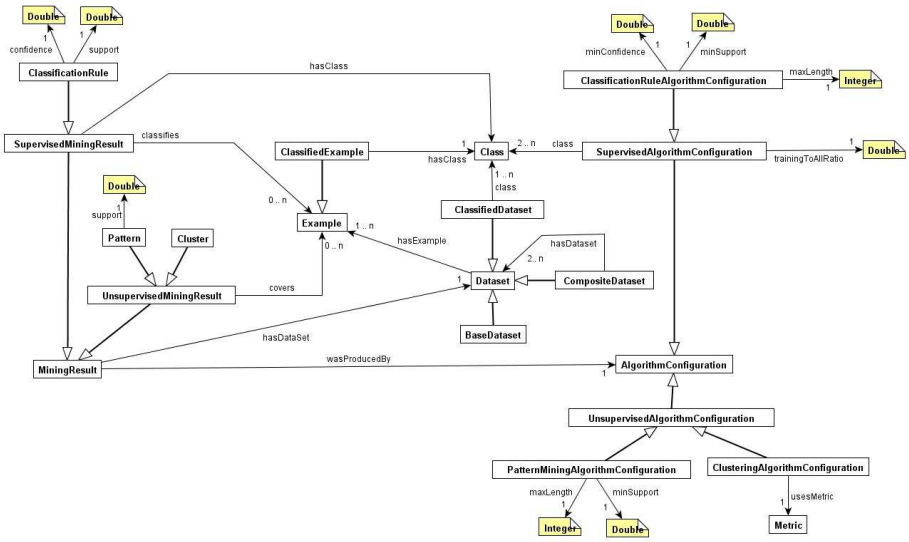

Fig. 4. RDM ontology core

5 Conclusions and further work

In this paper we have tested the feasibility of subgroup discovery using frequent structural patterns as descriptive features. The frequent patterns are generated using the novel seach strategy based on sorted refinement. Applying this strategy onto semantic annotations of CAD designs represents a challenging case-study from the ILP/RDM point of view. This is due to the diverse length and structure of the description of each example combined with taxonomical background knowledge. We have proposed an approach for integrating taxonomical background knowledge into an ILP system by implementing sorted refinement operator and extending it to include taxonomies on predicates.

To deal with the large amount of different RDM results, we have proposed an RDM ontology that will be used in the query interface of the currently developed dedicated data mining tool, as well as the data schema for storing such results.

In future work we will consider a more principled approach of integrating yet more complex ontological background knowledge, including recursive definitions and multiple inheritance, and the order on predicates. The first approach we will consider is using a hybrid language integrating description logics and Horn logic similar to $\mathcal{AL}$-log [4] and CARIN [2]. Other approaches worth exploration are more expressive formalisms such as F-logic. We will also closely collaborate with the end users to sensibly restrict the form of first-order features. Additionally, we are working on the RDM management system based on the presented ontology. For efficient retrieval of stored results a novel query engine will be developed.

References

1. Žáková, M., Železný, F., Garcia-Sedano, J. A.: Relational Data Mining Applied to Virtual Engineering of Product Designs. Accepted for ILP 2006.
2. Levy, A. and Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence, 104:165209, 1998.
3. Frisch, A.: Sorted downward refinement: Building background knowledge into a refinement operator for ILP. In ILP, LNAI 1634:104115. Springer, 1999.
4. Lisi, F.A., Malerba, D.: Ideal Refinement of Descriptions in AL-Log. In proceedings of ILP 2003, Szeged, Hungary. LNCS 2835: 215-232, Springer 2003.
5. Železný, F., Lavrač, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62: 33-63, 2006.
6. A. Srinivasan, S. Muggleton, M. J. E. Sternberg, and R. D. King: Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.
7. R. D. King et. al.: Functional genomic hypothesis generation and experimentation by a robot scientist.*Nature*, 427:247–252, 2004.
8. B. Dolšak et al.: Finite element mesh design: An engineering domain for ILP application. In proc. of ILP 1994, GMD-Studien 237: 305-320, 1994.
9. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Available at http://www.w3.org/TR/rdf-schema/.
10. McGuinness, D. L., van Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004. Available at http://www.w3.org/TR/owl-features/.
11. A. Appice, M. Ceci, A. Lnaza, F. A. Lisi and D. Malerba. Discovery of spatial association rules in geo-referenced census data: A relational mining approach. *Intelligent Data Analysis*, vol. 7, pp. 541-566, 2003.
12. Witten, I. H. and Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
13. A. Srinivasan. The Aleph manual version 4, 2003. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/, June, 7 2006.
14. S. Dzeroski and N. Lavrac (Eds.): Relational Data Mining, Springer, Berlin, 2001.
15. S-H. Nienhuys-Cheng and R. de Wolf: Foundations of Inductive Logic Programming, LNAI 1228, Springer 1997.