Automating Knowledge Discovery Workflow Composition Through Ontology-Based Planning

Monika Žáková, Petr Křemen, Filip Železný, and Nada Lavrač

***Abstract*—The problem addressed in this paper is the challenge of automated construction of knowledge discovery workflows, given the types of inputs and the required outputs of the knowledge discovery process. Our methodology consists of two main ingredients. The first one is defining a formal conceptualization of knowledge types and data mining algorithms by means of knowledge discovery ontology. The second one is workflow composition formalized as a planning task using the ontology of domain and task descriptions. Two versions of a forward chaining planning algorithm were developed. The baseline version demonstrates suitability of the knowledge discovery ontology for planning and uses Planning Domain Definition Language (PDDL) descriptions of algorithms; to this end, a procedure for converting data mining algorithm descriptions into PDDL was developed. The second directly queries the ontology using a reasoner. The proposed approach was tested in two use cases, one from scientific discovery in genomics and another from advanced engineering. The results show the feasibility of automated workflow construction achieved by tight integration of planning and ontological reasoning.**

***Note to Practitioners*—The use of advanced knowledge engineering techniques is becoming popular not only in bioinformatics, but also in engineering. One of the main challenges is therefore to efficiently extract relevant information from large amounts of data from different sources. For example, in product engineering, the focus of project SEVENPRO, efficient reuse of knowledge can be significantly enhanced by discovering implicit knowledge in past designs, which are described by product structures, CAD designs and technical specifications. Fusion of relevant data requires the interplay of diverse specialized algorithms. Therefore, traditional data mining techniques are not straightforwardly applicable. Rather, complex knowledge discovery workflows are required. Knowledge about the algorithms and principles of their applicability cannot be expected from the end user, e.g., a product engineer. A formal capture of this knowledge is thus needed, to serve as a basis for intelligent computational support of workflow composition. Therefore we developed a knowledge discovery (KD) ontology describing knowledge types and algorithms required for complex knowledge discovery tasks.**

A planning algorithm was implemented and employed to assemble workflows for the task specified by the user's input-output task requirements. Two versions of the planning algorithm were developed. The first one uses standard PDDL descriptions of algorithms, accessible to third party planning algorithms. A procedure for converting algorithm descriptions into PDDL was developed. The second directly queries the ontology using a reasoner. The proposed approach was tested in two use cases, one from genomics and another from product engineering. The results show the feasibility of automated workflow construction achieved by tight integration of planning and ontological reasoning. The generated workflows can be executed on the SEVENPRO platform; however, since they are annotated using the KD ontology, the planner can be integrated into other workflow execution environments.

***Index Terms*—Data mining, knowledge management.**

Manuscript received March 26, 2009; revised May 11, 2010; accepted June 28, 2010. Date of publication nulldate; date of current version nulldate. This paper was recommended for publication by Associate Editor B. Turchiano and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported by Project No. 201/09/1665 of the Czech Science Foundation and Project MSM6840770038 of the Czech Ministry of Education. The work of N. Lavrač was supported by the Knowledge Technologies Project funded by the Slovenian Research and Technology Agency.

M. Žáková, P. Křemen, and F. Železný are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague 6, Czech Republic (e-mail: {zakovm1@fel.cvut.cz; kremep1@fel.cvut.cz; zelezny@fel.cvut.cz).

N. Lavrač is with the Institute Jožef Stefan, Ljubljana 1000, Slovenia (e-mail: nada.lavrac@ijs.si).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2010.2070838

I. Introduction

INTEGRATION of heterogeneous data sources and inferring new knowledge from such combined information is one of the key challenges in present-day life sciences. Consider, e.g., bioinformatics where for virtually any biological entity (a gene, for example) vast amounts of relevant background information are available from public web resources. This information comes in diverse formats and at diverse levels of abstraction. Continuing the genomic example, the publicly available data sources range from DNA sequence information, homology and interaction relations, Gene Ontology annotations,[1] to information on the involvement in biological pathways, expression profiles in various situations etc. To merge only these exemplary sources of data, one already has to combine specialized algorithms for processing sequences, relational data, ontology information and graph data. It is thus no surprise that a principled fusion of such relevant data requires the interplay of diverse specialized algorithms resulting in highly intricate workflows.

While the mutual relations of such algorithms and principles of their applicability may be mastered by computer scientists, their command cannot be expected from the end user, e.g., a life scientist. A formal capture of this knowledge is thus needed, e.g., in the form of ontologies of relevant services and knowledge/data types, to serve as a basis for intelligent computational support of scientific workflow composition.

The term *knowledge discovery workflow* allows a wide scope of interpretations. For this work, we essentially define it as a *progression of steps (inductive, deductive, format-conversion procedures etc.) involved in generalizing specific data (e.g., measurements) into patterns, which, under appropriate interpretation, may represent novel knowledge about the problem domain under investigation.* Therefore, it can be viewed as a special form of scientific workflows [1], covering the data preparation

[1]http://www.geneontology.org/

and modeling stages of the standard CRISP-DM data mining methodology.[2]

This work was originally motivated by the complex knowledge discovery workflow of interleaving inductive, deductive and format-conversion procedures which had to be manually constructed in our previous study in bioinformatics [2].

The primary objective of this study is to investigate whether such complex workflows can be assembled automatically with the use of a knowledge discovery ontology and a planning algorithm accepting task descriptions automatically formed using the vocabulary of the ontology. To achieve this objective, we have developed and present a knowledge discovery ontology capturing complex background knowledge and relational data mining algorithms. We have developed a planner using standard PDDL descriptions of algorithms generated automatically from the ontology as a base line approach to demonstrate that the algorithm descriptions in the knowledge discovery ontology are suitable for planning. We have also developed an innovative planning algorithm, which obtains possible next steps by directly querying the ontology using a reasoner.

We use the mentioned bioinformatics study as a use case in this paper. To demonstrate the generality of the proposed approach, we also test our methodology in the domain of engineering,[3] where knowledge discovery workflows exhibit features similar to scientific workflows [3], namely, their complexity and their inductive character.

This paper builds upon the state-of-the-art of rather remote fields. First, to conceptualize the knowledge discovery domain, we follow up on the presently emerging research attempting to establish a *unifying theory of data mining*[4], [5]. We built upon the definitions of core knowledge discovery concepts presented in [5] in designing the core parts of the ontology, namely the concepts of knowledge, representation language, pattern, dataset, evaluation, and further more specialized concepts. Using this vocabulary, specific classes of algorithms can be annotated as to their functionality. For example, inductive algorithms (given a particular pattern evaluation function) will produce patterns out of datasets, format conversion algorithms will produce datasets out of datasets, etc. The ontology implicitly delimits the variability of possible workflows for a given task. For example, if the user desires to mine patterns in the language L of propositional formulas, any algorithm may be employed that is annotated as to produce patterns in L or in any language subsumed by L (e.g., propositional conjunctions). Second, in the technical aspects of our methodology, we adhere to proven standards from the fields of the semantic web (namely, the OWL [6] framework for ontology modeling) and planning [the Planning Domain Definition Language (PDDL) [7] standard for planning problem description].

Note that currently there is a significant gap between the two foundations of our work mentioned above. Unified data mining conceptualizations including learning from structured data with background knowledge, such as those presented in [4] and [5] do not possess an actionable technical grounding. This is despite certain promising proposals, e.g., in the frame of inductive databases [8], which have yet to find their way to implementation and practice. Inversely, most of the previously proposed data mining platforms such as [9] deal only with "propositional" data mining requiring all data in the flat representation of attribute-value tuples. Propositional (or "attribute-value") data mining is a traditional framework which generally does not match the demands of mining tasks in domains exhibiting rich knowledge representations such as description or relational logic [10]. Here lies the secondary contribution of our paper. Our methodology bridges the gap by providing a working prototype of an actionable data mining conceptualization including learning from structured and relational data, enabling automated assembly of knowledge discovery workflows.

This paper is structured as follows. Section II provides an extensive overview of related work. In Section III, a formal conceptualization of the knowledge discovery domain is proposed for a segment of data types, data processing types, and data mining algorithms used in this study. Section IV proposes an approach to automated knowledge discovery workflow construction through ontology-based planning, evaluated on two case studies in Section V.

II. Related Work

Intelligent management of data analysis workflows has attracted a lot of development in recent years. Such development builds upon technologies provided by several information science fields, the two most notable of them being the *semantic web* and *grid computing*. The former provides the building blocks through which workflows can be annotated, facilitates automatic service discovery, efficient management of workflows or even their automated composition. The latter technology allows to execute workflows in a distributed computing environment while optimizing factors such as total runtime, security, etc. Both technologies actively overlap, such that, e.g., annotations extend also to physical constituents of the computing environment enabling an intelligent translation of an *abstract* (resource independent) workflow to a *concrete* one, where tasks are mapped onto particular computing resources.

Our work is mainly concerned with automatic *composition* of data mining and knowledge discovery workflows by planning. We currently focus on generating abstract workflows rather than providing a workflow editing environment focused on the integration of computational resources and middleware and efficient execution, such as Triana [11], the system for scientific workflows developed in Kepler[4], WCT developed within the K-WF grid[5] and the tools developed within the DiscoveryNet project[12] and project ADMIRE [13].

Similarly to the FAEHIM [14] project, we concentrate on the subdomain of scientific knowledge discovery connected to data mining. In contrast to our approach, the toolkit developed within FAEHIM allows only for manual composition of workflows and does not use any formally defined conceptualization of the domain. The Taverna [15] environment for workflow development

[2]http://www.crisp-dm.org

[3]Specifically within the project SEVENPRO, Semantic Virtual Engineering Environment for Product Design, IST-027473 (2006–2008), Sixth Framework Program of the European Commission.

[4]http://kepler-project.org

[5]http://www.kwfgrid.eu/

and execution was developed within the myGrid[6] project. It uses an ontology focused on operations specific to bioinformatics tasks. The workflows are not represented in an ontology language and the workflow design is user-driven.

To the best of our knowledge, there is so far no previous work providing an actionable ontology for data mining including data mining from structured data with complex background knowledge. There have been efforts to provide a systematic description of data and processes for the classical data mining tasks, e.g., in projects MiningMart [16], DataMiningGrid [9] and systems CAMLET [17], CITRUS [18], and NExT [19]. There have been some other efforts to formalize concepts for knowledge discovery on the Grid [20], [21]. However, these ontologies also cover only propositional data mining.

DataMiningGrid focuses on producing a set of generic tools and services for deploying data mining applications on standards compliant grid service infrastructures. MiningMart focuses on guiding the user to choose the appropriate preprocessing steps in propositional data mining. Both systems contain a metamodel for representing and structuring information about data and algorithms, however, none of the metamodels is expressed in an ontology language. Also, the systems do not provide means for automatic workflow creation. The systems CITRUS and CAMLET make a limited use of planning for process decomposition starting from a manually defined structure. CITRUS uses an object oriented schema to model relationships between the algorithms, while CAMLET uses an ontology of algorithms and data structures.

The most systematic effort to construct a general knowledge discovery ontology is described in [19]. The ontology used by the NExT system is built on OWL-S [22] and provides a relatively detailed structure of the propositional data mining algorithms. It focuses on classical data mining processes, consisting of three subsequent steps: preprocessing, model induction and postprocessing. In contrast to NExT, we address relational data mining workflows with possibly multiple interleaved occurrences of steps pertaining to the three categories. Furthermore, the workflows generated by the NExT system are linear, whereas our workflows are directed acyclic graphs.

The development of a unified theory (conceptualization) of data mining was recently identified as the first of ten most challenging problems for data mining research [4]. While we do not claim completeness or universal applicability of the ontology developed in this work, in its design we did try to follow the state-of-the-art works attempting to establish such a unified theory including [5] and [23]. In parallel to our work, the OntoDM [24] ontology is being developed on the basis of [5]. A principled top-down approach was adopted to the development of OntoDM aiming at its maximum generality and describing even inner working of the algorithms. Given the complexity of the domain subject to modeling, the ontology is currently not sufficiently specific for purposes of workflow construction [25]. Also, unlike our ontology, OntoDM is not compatible with OWL-S.

Previous work exists on the conceptualization of planning [26], [27]. The sources, however, do not provide details on workflow description. Therefore, a workflows subontology is developed within our work.

Several previous works have explored planning in the context of workflows. Notably, in the Wings component of the Pegasus project [28] a planner employing semantic reasoning is used to construct a concrete workflow from a given abstract workflow based on concrete input data [29]. In our research we tackle a related yet different goal; given an ontology and a task description, we use a planner to construct an abstract workflow. Also, in Pegasus, an algorithm integrating planning with reasoning is used to validate abstract workflows and to suggest next steps to the user, while we are proposing whole abstract workflows, which do not require the user to be familiar with each part of the knowledge discovery process. A similar aim was followed by [30], however, this work is focused only on automatic formation of linear sequences of tasks.

Also relevant is tackling the problem of *web service composition* in the framework of planning. [31] uses BPEL4WS[7] for task formulation and workflow representation. Since the adaptation of BPEL4WS to scientific workflows is still not standardized [32], we have decided not to use BPEL4WS in our work.

The relevant work of [33] relies on computing a *causal link matrix* for all available services. Informally, this matrix captures semantic input/output compatibility among pairs of services. Services can be then viewed as nodes in a graph with the link matrix defining the edge labels. Finding a suitable sequence of services can then be elegantly reduced to finding a path in this graph. In our framework, however, we work with a more general, nonlinear notion of a plan, where the inputs of an algorithm (action) combine the outputs of multiple other algorithms. Thus, *pairwise* semantic compatibility does not carry sufficient information to construct a plan in our framework and we have to rely on general planning strategies.

Similarly to our approach, [34]–[36] translate an OWL description to a planning formalism based on PDDL. While work presented in [35] and [36] use classical STRIPS [37] planning, in [34], Hierarchical Task Network (HTN) planning [38] is employed, which relies on an explicitly defined task decomposition hierarchy. HTN is not applicable in our framework not constrained to tree-based task decomposition.

The approach presented in [36] and [35] uses a reasoner in the pre-processing phase; we make a step beyond by investigating the possibility of integrating a reasoning engine directly with the planner. As another difference, our procedure for converting the task descriptions to PDDL does not rely on OWL-S, therefore, we do not require the involved algorithms to be implemented as web services.

Planning directly in description logics is addressed in [39]. Currently, the algorithm can only deal with DL-Lite descriptions with reasonable efficiency.

III. Knowledge Discovery Ontology

Central to our approach is a formal conceptualization of the knowledge discovery domain provided by the *Knowledge Discovery Ontology* (KD ontology, for short). The ontology defines relationships among the ingredients of knowledge discovery

[6]http://www.mygrid.org.uk/

[7]http://www.ibm.com/developerworks/library/specification/ws-bpel/

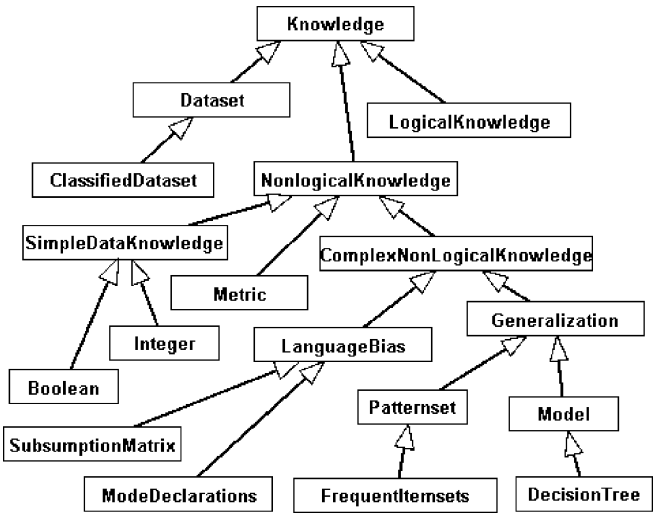

Fig. 1. Part of the top level structure of the knowledge type part of the ontology with subclass relations shown through arrows.

scenarios, both declarative (various knowledge representations) and algorithmic. The primary purpose of the ontology is to enable the workflow planner to reason about which algorithms can be used to produce intermediary or final results required by a specified data mining task.

A framework for data mining proposed in [5] identifies three basic concepts of data mining: "data," "patterns and models," and "data mining task." Following this view, our three core concepts are: *knowledge*, capturing the declarative elements in knowledge discovery, *algorithms*, which serve to transform knowledge into (another form of) knowledge, and *knowledge discovery task*, which we have extended to involve workflows.

The ontology is implemented in the description logic variant of the semantic web language OWL-DL [6]. Our primary reasons for this choice were OWL's sufficient expressiveness, modularity, availability of ontology authoring tools and optimized reasoners. It currently contains around 150 concepts and is available online.[8]

A. Knowledge

Any declarative ingredient of the knowledge discovery process such as datasets, constraints, background knowledge, rules, etc., are instances of the `Knowledge` class. Fig. 1 shows an illustrative part of the class hierarchy of knowledge types.

In data mining, many knowledge types can be regarded as sets of more elementary pieces of knowledge [5]. For example, first-order logic theories consist of formulas. Similarly, the common notion of a *dataset* corresponds either to a set of attribute-value tuples or to a set of relational structures, each of which describes an individual object. This structure is accounted for through the predicate `contains`, so, e.g., a first-order theory `contains` a set of first-order formulas.

Moreover, some knowledge types may be categorized according to the expressivity of the language in which they are encoded. For this purpose, we have designed a hierarchy of language expressivity, of which Fig. 2 shows a fraction. The

[8]http://krizik.felk.cvut.cz/ontologies/2008/kd.owl

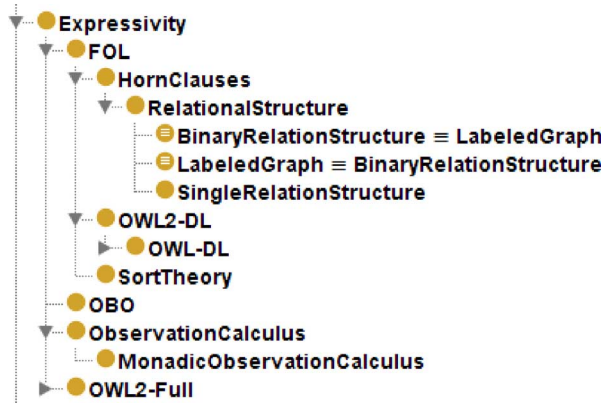

Fig. 2. A part of the expressivity hierarchy in the Protege ontology editor. Expressivity is defined as an essential part of `LogicalKnowledge` class.

hierarchy is an acyclic directed graph, however, for better readability only tree structure is shown in Fig. 2.

We further distinguish certain knowledge types which play special roles in knowledge discovery. A basic concept is that of a dataset. The Dataset class is defined as `Knowledge`, which contains `Examples`. The `hasExpressivity` property can be also applied to datasets to distinguish between propositional datasets and relational datasets.

All the other types of knowledge such as pattern sets, models and constraints are clustered in the class `Non − LogicalKnowledge`. It contains the essential concept of a generalization, which is a knowledge class with the special property that it *defines a mapping* from one or more knowledge classes to another knowledge class. Intuitively, this class serves to hold the results of inductive mining algorithms; such results generally can be viewed in a unified fashion as mappings [5]. Of course, the generalization's mapping, i.e., its semantics, is ultimately assigned to it by an *algorithm* used to interpret it. The `Generalization` class contains two subclasses, which can be distinguished by the property of decomposability and by the type of algorithms used to produce it. `Model` is a result of a predictive algorithm and it cannot be decomposed into independent parts. `Patternset`, on the other hand, can be decomposed into independent parts and is usually produced by a descriptive algorithm, such as an association rules learner.

Algorithms

The notion of an algorithm involves all executable routines that can be used in a knowledge discovery process, like inductive algorithms and knowledge format transformations. Any algorithm turns a knowledge instance into another knowledge instance. For example, inductive algorithms will typically produce a `Patternset` or `Model` instance out of a Dataset instance. Of importance are also auxiliary representation changers, transforming datasets to other datasets. These may be simple format converters (e.g., only changing the separator character in a textual data file), or more complex transformations characterized by information loss. This may be incurred either due to a conversion into a language class with lower expressiveness (e.g., for "propositionalization" [40] algorithms) or even without expressiveness change (e.g., for principal component representation of real vectors).

The Algorithm class is a base class for all algorithms, like JRip (an algorithm for decision rules induction implemented in Weka [41]), in the example below. The hierarchy contains also fully defined classes, like FrequentPatternsAlgorithm or PredictiveRuleAlgorithm for fine-grained categorization of data mining algorithms according to their functionality.

Each algorithm configuration is defined by its input and output knowledge specifications and by its parameters. In order to maintain the compatibility with OWL-S, the Algorithm class is defined as a specialization of the OWL-S class Process and an algorithm configuration is an instance of its subclass NamedAlgorithm. Both the input knowledge and the parameters are instances of AlgorithmParameter and defined using the input property. The output knowledge specifications are instances of AlgorithmParameter and defined using the output property. The parameter instances are then mapped to the appropriate Knowledge subclasses using the isRangeOf property.

Furthermore, each named algorithm is linked to its implementation using the stringRepresentation property, which is aimed at automatically running the generated workflows within a knowledge-discovery engine in a uniform way. To run the algorithm, an instance of AlgorithmExecution is created containing actual values of the algorithm parameters, which are passed to the algorithm implementation.

As an example we present the definition of the JRip algorithm in the description logic notation using the extended ABox syntax [42]

$$
\begin{aligned}
\{\texttt{JRip}\} \sqsubseteq\ & \texttt{NamedAlgorithm} \\
& \sqcap \exists\, \texttt{output} \cdot \{\texttt{JRip-O-PredictiveRules}\} \\
& \sqcap \exists\, \texttt{input} \cdot \{\texttt{JRip-I-Dataset}\} \\
& \sqcap \exists\, \texttt{input} \cdot \{\texttt{JRip-I-Pruning}\} \\
& \sqcap \exists\, \texttt{input} \cdot \{\texttt{JRip-I-MinNo}\} \\
& \texttt{JRip-I-Dataset-Range} \\
\equiv\ & \exists\, \texttt{isRangeOf} \cdot \{\texttt{JRip-I-Dataset}\} \\
\equiv\ & \texttt{ClassifiedDataset} \\
& \sqcap \forall\, \texttt{hasExpressivity} \cdot \\
& \texttt{SingleRelationStructure} \\
& \sqcap \forall\, \texttt{hasFormat} \cdot \{\texttt{ARFF}, \texttt{CSV}\}.
\end{aligned}
$$

The JRip algorithm is defined as an algorithm that has two parameters: one stipulating whether to use pruning and one determining the minimum number of examples covered by each single rule. It can be applied to a single relation classified dataset in the CSV or ARFF format and produces a result in the form of predictive rules (i.e., patterns defining a mapping to a distinguished set of classes).

B. Workflows Subontology

In order to formalize the problem description and for storing the created workflows in a knowledge-based representation, we have created a small ontology for workflows, which extends the KD ontology. The workflows subontology has two central notions: KnowledgeDiscoveryTask and Workflow.

Each KnowledgeDiscoveryTask is defined by its init and goal specifications. As an example we present the definition of the problem of generating predictive rules from relational data (RRules) in the description logic notation

$$
\begin{aligned}
&\texttt{RRulesTask} \\
&\quad \sqsubseteq \texttt{KnowledgeDiscoveryTask} \\
&\quad \sqcap \exists\, \texttt{goal} \cdot \texttt{PredictiveRules} \\
&\quad \sqcap \exists\, \texttt{init} \cdot (\texttt{ClassifiedDataset} \sqcap \\
&\qquad \forall\, \texttt{hasExpressivity} \cdot \texttt{RelationalStructure} \sqcap \\
&\qquad \forall\, \texttt{hasFormat} \cdot \{\texttt{RDFXML}\}) \\
&\quad \sqcap \exists\, \texttt{init} \cdot (\texttt{LogicalKnowledge} \sqcap \\
&\qquad \forall\, \texttt{hasExpressivity} \cdot \texttt{OWL-DL} \sqcap \\
&\qquad \exists\, \texttt{hasFormat} \cdot \{\texttt{RDFXML}\}).
\end{aligned}
$$

RRules problem is defined as problem of generating relational predictive rules from a relational classified dataset and an ontology in OWL-DL as background knowledge, both expressed in the RDFXML format. Currently the ontology describes a few problem types, which were developed for our use cases and which should serve as a template for the user to specify problem types relevant for his/her KD tasks.

An abstract workflow is represented by the Workflow class, which is a subclass of the Algorithm class in the KD ontology. This allows encapsulating the often repeated workflows and construct hierarchical workflows.

The abstract workflow is represented as a set of Actions specified using hasAction property. An action is defined by the hasAlgorithm, specifying the algorithm configuration used by this action, and by startTime specifying the step within the plan in which the action should be carried out. The dependencies between actions are represented using the predecessor property. The property can express both control and data flow dependency. The formal representation of abstract workflows is used for workflow instantiation and execution within the knowledge discovery engine and also for storing and reuse of the generated workflows.

IV. Automatic Workflows Construction

In this paper we focus on automatic construction of abstract workflows. Each generated abstract workflow is stored as an instance of the Workflow class and can be instantiated with a specific algorithm configuration either manually or using a predefined default configuration. We treat the automatic workflow construction as a classical planning task, in which algorithms represent operators and their required input and output knowledge types represent preconditions and effects.

Both the information about the available algorithms and knowledge types as well as the specification of the knowledge discovery task is encoded through an ontology. At the same time, we want to be compatible with established planning standards. For these reasons, we have decided to explore two approaches to solving the planning task. The first, baseline approach consists of generating a description of the domain and the problem description in the PDDL language [7] using elements of the KD ontology and implementing a planning

```
uknow = []; used subclasses of Knowledge
actions = []; created PDDL actions
types = []; hierarchy PDDL types
onto2pddl():
        classify KD ontology;
        algs := {instances of NamedAlgorithm};
        for ( al : algs)
            act := new Action(al.name);
            iospecs := {input and output specifications};
            for (ios : iospecs)
                eqios = transformIO(ios);
                if (eqios == null) continue;
                act.addIO(convertIO2pddl(equios, uknow,
                    act.varnames));
            actions.add(act);
        add uknow classes to KD ontology and classify;
        types := classes2types(uknow);
        return createDomainPDDL(actions,types);

preds = []; a list of predicates describing the io specification
params = []; a list of variables used in the io specification
convertIO2pddl(ios,uknow,varnames):
        if (ios.isNamedClass())
            return {{available(v_i)}, {v_i - ios.name}};
        else if (ios.isIntersectionClass())
            rest4cls = {operands of ios represented by
                named classes or restrictions
                on hasExpressivity and contains};
            rest4pred = {operands of ios represented by
                restrictions on other properties};
            comp = createCompositeClass(rest4cls);
            uknow.add(comp);
            params.add(v_j - comp.name);
            preds.add(available(v_j));
            rest2preds(rest4pred,v_j,preds,params);
        return preds, params;
```

Fig. 3. A skeleton of the procedure for converting descriptions of algorithms from the KD ontology into PDDL.

algorithm, which uses PDDL descriptions. The second, less orthodox approach, implements a planning algorithm capable of directly querying the KD ontology using a reasoner. The Pellet [43] reasoner is used.

A. Generating Domain and Problem Descriptions in PDDL

We use PDDL 2.0 with type hierarchy and domain axioms. Planning algorithms require two main inputs. The first one is the description of the domain specifying the available types of objects and actions. The second one is the problem description specifying the initial state, goal state and the available objects. We have implemented a procedure for generating the domain description from the KD ontology.

The domain description is generated by converting Named `algorithms` into PDDL actions, with `inputs` specifying the preconditions and `outputs` specifying the effects. Both inputs and outputs are restricted to conjunctions of OWL classes. We consider only those inputs that are specified by instances of classes disjoint with `SimpleDataKnowledge`, which is used to represent algorithm configuration parameters, e.g., minimum support of a rule. Since PDDL can handle only named types and their hierarchies, it is necessary to preprocess classes defined using owl:Restriction.

A skeleton of the conversion procedure is in Fig. 3. Both the list of instances of `NamedAlgorithm` and the list of input and output specifications are obtained by a SPARQL-DL query. Procedure **transformIO** converts an i/o specification defined by an instance of `AlgorithmParameter`into a class equivalent to its range, which consists of an intersection of the `Knowledge` subclasses and restrictions defined in the KD ontology. The equivalent class is obtained by a SPARQL-DL query. For `SimpleDataKnowledge` subclasses representing algorithm parameters the procedure returns null.

The procedure **convertIO2pddl** converts an i/o specification defined by a named class or an owl:intersectionOf class into PDDL predicates. The operands of the owl:intersectionOf class specified by named classes and universal restrictions on properties `contains` and `hasExpressivity` are converted into named classes. The named classes are added to the *uknow* list and their hierarchy is later inferred by an OWL reasoner. The named class is converted to a unary predicate `available` and also added to action parameters. Operands specified by restrictions on other properties are converted using the procedure **rest2preds** to binary predicates with the first argument being the previously defined named class and the second argument is given by the restriction value. All the generated predicates and parameters are then added to action preconditions or effects using **addIO**.

As an example the definition of the action representing the JRip algorithm described in Section III is presented in PDDL below

```
(:action JRip
 :parameters(
  ?v0 - Dataset_hasExpressivity
   _SingleRelationKnowledge
  ?v1 - CSV
  ?v2 - PredictiveRules)
 :precondition (and (available ?v0)
      (format ?v0 ?v1))
 :effect (and (available ?v2))).
```

The information about the output of the JRip algorithm is expressed using the named class `PredictiveRules`. Therefore the effects of the action using the JRip algorithm are represented using the unary predicate `available` applied on the named class `PredictiveRules`.

Finally, procedure **createDomainPDDL** takes the list of actions and hierarchy of PDDL types and fits them into a domain file template in PDDL.

Problem description in PDDL is generated in a very similar way, except we are dealing with objects instead of variables. The objects appearing in `init` and `goal` conditions are generated from an individual of type `KnowledgeDiscoveryTask` in the KD ontology, which represents a particular problem, e.g., producing a set of predictive rules from a dataset stored in a relational database.

B. Planning Algorithm

We have implemented a planning algorithm based on the Fast-Forward (FF) planning system [44] to generate abstract

workflows automatically. The FF planning system uses a modified version of a hill climbing algorithm called *enforced hill climbing* to perform forward state space search. The heuristics used by the enforced hill-climbing algorithm is defined as the number of operators in the plan constructed using relaxed GRAPHPLAN [45].

If the enforced hill-climbing algorithm fails, the problem is solved using a complete search algorithm. In [44] the search space is pruned using two heuristics: a helpful actions heuristic, which considers only actions that add at least one goal at the first time step, and added goal deletion heuristics, which exploits goal ordering.

We have implemented the basic architecture of the FF planning system consisting of the enforced hill climbing algorithm and the relaxed GRAPHPLAN. Since the planning problem in workflow construction contains no goal ordering, no mechanisms for exploiting goal ordering are implemented. Our implementation is also capable of handling PDDL only with STRIPS [37] expressivity and using types.

In order to produce several relevant workflows in a reasonable time frame, the algorithm is run repeatedly, randomly permuting the order in which immediate neighbors of one state are added to the open-set during the breadth-first search.

Following the reasoning from the beginning of Section IV, we have implemented the above principles into two versions of the planning algorithm. The first, PDDLPlanner, assumes that all available actions are described through a standard PDDL file created in a preprocessing stage using the KD ontology. The KD ontology is not used during the actual planning.

The second version, PelletPlanner, obtains neighboring states during enforced hill-climbing by matching preconditions of available algorithms with currently satisfied conditions. Here, each such matching is conducted in the planning time via posing an appropriate SPARQL-DL [46] query towards the KD ontology. The query templates have been created manually to look for most specific matches. The reasoner is used mainly to infer implicit hierarchies of knowledge types.

C. Workflow Storage and Execution

The workflow management and execution functionality is encapsulated in the RDM Manager (shown in Fig. 4), which forms a part of the SEVENPRO software infrastructure [47]. The central component of the RDM Manager is the RDM Engine responsible for designing, storing, retrieving and executing workflows. For this sake, the RDM Engine has access to the KD ontology, and can launch the planner, the ontology based constructor of PDDL files (Onto2PDDL box in the figure) as well as all the various algorithms appearing in workflows. The RDM Engine is equipped with a web service interface allowing a standardized access. A graphical user interface (RDM GUI) has been developed enabling specification of the knowledge discovery task and passing on the specification to the RDM Engine web services. The Semantic Repository box also shown in Fig. 4 is a central storage point of the SEVENPRO software platform. The RDM Manager Tools stores all results of knowledge discovery processes, including the constructed workflows into the Semantic Repository for later retrieval by itself or by other software components of the SEVENPRO platform. Conversely, the

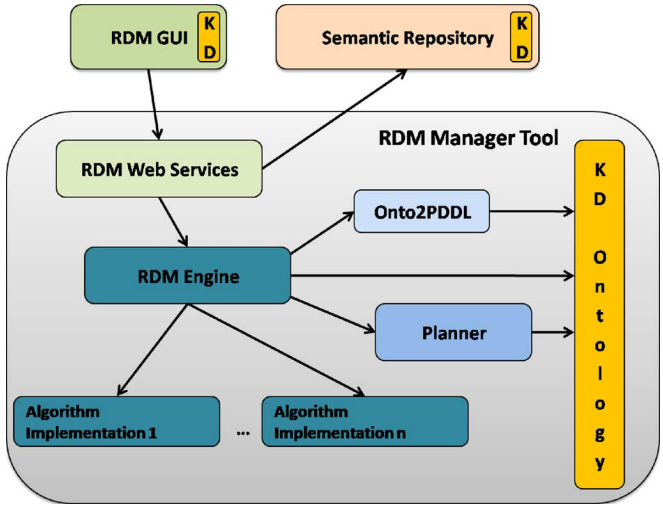

Fig. 4. An overview of the RDM Manager architecture.

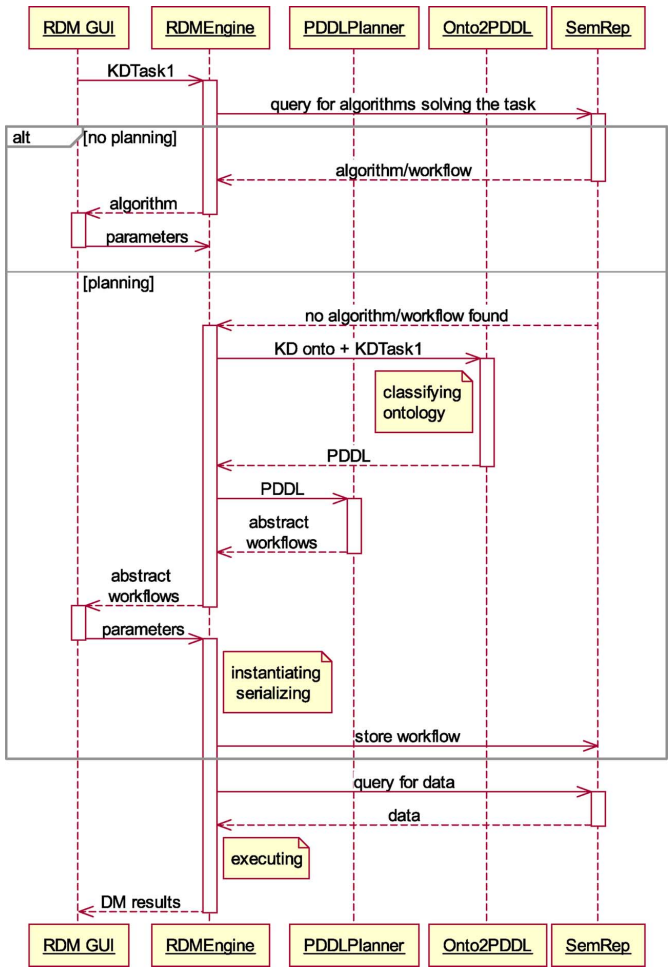

Fig. 5. Sequence diagram showing a typical scenario for PDDLPlanner.

Semantic Repository also holds all data onto which knowledge discovery workflows are applied.

The general workflow maintenance scenario is shown in Fig. 5. The user formulates a knowledge discovery task using the RDM GUI, which formalizes the task specification into a SPARQL query, passed to the RDM Engine. The RDM Engine

queries the Semantic Repository for an existing workflow (possibly a single algorithm) solving the task. If such a workflow is found, it is presented to the user who can set or update its parameters. Otherwise, the RDM Engine calls the Planner. If the PDDLPlanner is used, the Onto2PDDL component is called first to produce the suitable PDDL file.

A plan generated by the Planner is a directed acyclic graph with nodes representing `Algorithm` instances, which do not contain any values of algorithm parameters specified by simple datatypes (e.g., `MinNo` – the minimal number examples covered by one rule). Therefore, in the next stage, it is necessary to convert the plan actions into a sequence of instances of `AlgorithmExecution`.

A SPARQL-DL query is used to search for the instances of `AlgorithmExecution` used by the actions in the plan. In the current version of our system, the user has three options: to use default configurations for all the algorithms, to choose from among previously used configurations or to set all parameters manually.

When all the actions of the plan have been instantiated, they are combined into an abstract workflow represented by an instance of the `Workflow` class, which is stored in the Semantic Repository. Since the current version of the RDM Engine does not provide any means for parallelization, the actions of the workflow are converted to a sequence. The RDM Engine then generates a query for execution of each algorithm configuration in the sequence.

The data are then retrieved from the Semantic Repository using a SPARQL query. Then, for each algorithm in the sequence, the RDM Engine extracts the algorithm's class from its `AlgorithmExecution`. Knowing the class, it then launches the algorithm's wrapper passing the retrieved the data and parameters to it. When the algorithm terminates, the RDM Engine passes its results to the wrapper of the next algorithm in the sequence.

The algorithms currently available in the RDM Engine include specialized algorithms for relational learning through propositionalization [48] and subsequent propositional search described in [49], a collection of algorithms from Weka data mining platform [41] including the JRip rule learner, the J48 decision tree induction algorithm and Apriori algorithm. In addition, algorithms for data preprocessing and format conversions are also available within the RDM Engine. New algorithms can be easily added to the RDM Engine by developing a wrapper for the particular algorithm and possibly also an API for accessing the results in case the particular result type is not yet included in the RDM Engine.

V. Evaluation

As explained in Section II, our system solves a task not tackled by existing algorithms. Empirical tests should thus primarily serve as a proof of concept, showing that the approach scales, with acceptable computational demands, to reasonably large real-life problem instances. We have conducted workflow construction experiments in two domains: genomics and product engineering. The workflows pertaining to both of the use cases are required to merge data with nontrivial relational structure, including ontology background knowledge.[9] Again, this setting precludes the application of previous workflow construction systems, limiting the scope for comparative evaluation. However, we do run comparative experiments to evaluate the effects of employing either of the two earlier described planning strategies.

Also, to trace the dependence of runtime on the size of the KD ontology and the number of available algorithms annotated using the ontology, we perform experiments with two versions of the KD ontology and with growing set of algorithms for the first version. The second version of the KD ontology is a strict extension of our original KD ontology with added classes required for annotating algorithms from the Orange [50] system.

A. Use Cases

1) Genomics: In analyzing gene expression data, we are dealing with the following sources of information: gene expression microarray data sets, Gene Ontology (GO) [51] and gene annotations. Annotations of genes using GO terms can be extracted from a public database.

Task: The task was to apply relational machine learning algorithms to produce a set of descriptive rules for groups of genes differentially expressed in specific conditions, more specifically for the acute lymphoblastic leukemia and acute myeloid leukemia. The data sources available were a gene expression microarray data set, GO and gene annotations from the Entrez database[10]. The operators are algorithms for preparing inputs for the relational data mining (RDM) described in [2] and components of the framework for RDM with taxonomic background knowledge described in [49].

2) Engineering: Product engineering deals with very specific knowledge types such as CAD, documentation, ERP/database, etc. The SEVENPRO project addressed the problem of the effective reuse of heterogeneous knowledge and past designs by providing a unified view of the available knowledge through commonly agreed ontologies. Engineering designs capturing implicit expert knowledge have relational nature, specifying various numbers of primitive objects and relations between them. In the SEVENPRO environment data are encoded in a subset of the RDFS formalism.

Task: One of the tasks solved within the SEVENPRO project was to generate descriptive and predictive rules from annotations of CAD drawings of different products. We were particularly interested in descriptive rules characterizing a particular class. The classification task was carried out as well in order to verify that we can distinguish between the product classes based on the provided information. The input data consisted of a list of identifiers of CAD drawings, CAD ontology and the annotations of individual CAD drawings.

B. Results

Experiments were carried out on a 1.8 GHz Intel Centrino PC with 1 GB memory. We used each planner for the two tasks described above and we used two versions of the KD ontology.

[9]Ontologies acting as knowledge entering the workflows should not be confused with the KD ontology guiding the construction of the workflows.

[10]Maintained by US National Center for Biotechnology Information, ftp://ftp.ncbi.nlm.nih.gov/gene/.

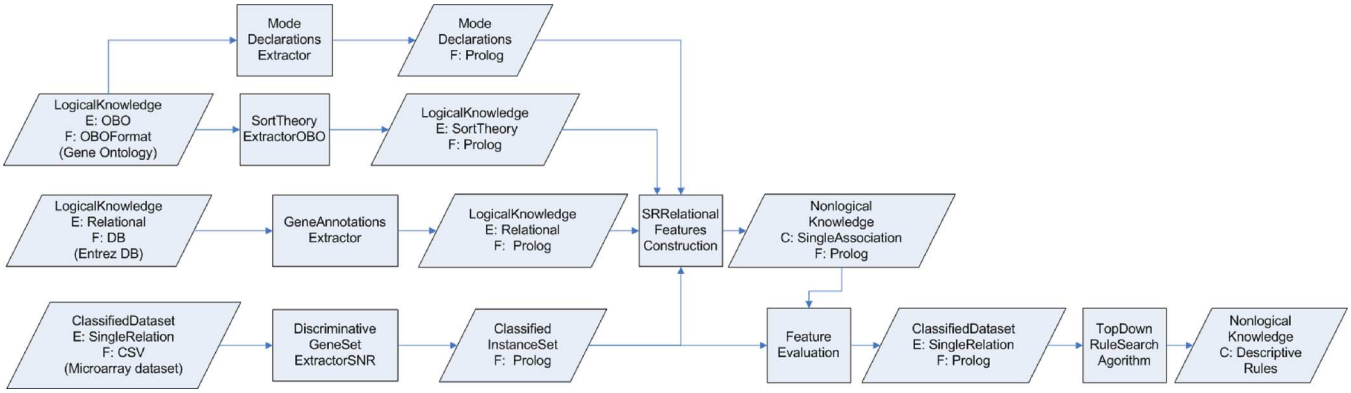

Fig. 6. Abstract workflow generated for obtaining descriptive rules for groups of differentially expressed genes for AML versus ALL. Rectangles represent algorithms and parallelograms represent data passed between them. Properties are abbreviated as follows: E: hasExpressivity, C: contains, and F: hasFormat.

TABLE I
PLANNER PERFORMANCE RESULTS, WITH RESPECT TO THE DOMAIN AND ONTOLOGY USED AND THE NUMBER OF ALGORITHMS AVAILABLE. THE TIME FOR PREPROCESSING (PREP.) AND PLANNING (PLAN) IS SHOWN IN SECONDS

Task	Ontology	No. of algorithms	PDDLPlanner		PelletPlanner	
			Prep.	Plan	Prep.	Plan
GEN	KD-RDM	18	27	0.641	13	0.766
GEN	KD-RDM	43	75	0.828	38	2.703
GEN	KD-RDM	60	480	0.906	133	3.891
GEN	KD-Orange	43	402	0.984	205	3.688
ENG	KD-RDM	18	28	0.407	15	0.782
ENG	KD-RDM	43	68	0.906	33	2.438
ENG	KD-RDM	60	387	0.922	134	3.250
ENG	KD-Orange	43	349	0.625	218	3.062

The first version contains classes necessary for annotation of algorithms available in the RDM Manager tool (KD-RDM), whereas the second version (KD-Orange) contains also classes necessary for annotations of algorithms available in the Orange data mining platform. KD-RDM contains 187 classes, 38 object properties, and 114 individuals. KD-Orange contains 284 classes, 51 object properties, and 191 individuals. The ontology KD-RDM was used to annotate 18 algorithms, which are part of the RDM Engine. The ontology KD-Orange was used to annotate also algorithms available in Orange [50], in total 43 algorithms.

Other algorithm annotations for KD-RDM were created artificially. For PDDLPlanner, the preprocessing stage includes conversion into PDDL. The results are summarized in Table I. None of the Orange algorithms were employed in the produced workflows, they only served to make the search task harder.

The results primarily show that successful workflows (exemplified below) can be automatically achieved in small absolute run times. Further, we observe rather small sensitivity of the run times to the size of the KD ontology (more specifically, the number of algorithms it contains).

Interestingly, the results also show the superiority of the innovative PelletPlanner strategy of "online querying for actions" over the baseline PDDLPlanner strategy in case of formulating new and diverse tasks, which is a typical scenario in both investigated domains. The single factor contributing to this superiority is the preprocessing time, smaller for PelletPlanner. This is mainly because ontology classification, the most time consuming operation within preprocessing, has to be performed twice by the reasoner when converting to PDDL. On the other hand, in preprocessing for PelletPlanner, this operation is performed only once. The described headstart of PelletPlanner is then reduced in the actual planning phase but still remains significant due to the relatively small proportion of planning time within the combined run time. In case of a set of planning tasks using the same domain description, the PDDLPlanner is however a better choice, since in this case the preprocessing phase can be run only once for the whole set of tasks.

An example of an abstract workflow generated for the genomics task described in Section V-A1 is shown in Fig. 6. The generated workflow utilizes algorithms developed by several different researchers and some of the tasks (e.g., discriminative gene set extraction) are independent of the rest. Using an automatically generated and semantically described workflow makes it far easier to conduct a series of experiments focusing influence of variations in one particular step of the process on the result of the whole data mining process without having to understand some other steps.

An example of an abstract workflow generated for the engineering task described in Section V-A2 is shown in Fig. 7. The same workflow had been produced manually within the SEVENPRO project and it was successfully rediscovered by the planner and executed using the RDM Manager tool developed within the SEVENPRO project.

VI. CONCLUSION AND FUTURE WORK

The primary objective of this study was to investigate whether complex scientific and engineering knowledge discovery workflows, such as those we had to develop manually in previous studies [2], [3], can be proposed automatically. We have developed a methodology for automatic composition of abstract workflows, which are proposed to the user and can be instantiated interactively. Our methodology focuses on workflows for complex knowledge discovery tasks dealing with structured data and background knowledge, while the previous studies deal only with classical propositional data mining tasks or are specialized for one domain only.

Our methodology consists of two main ingredients. The first on is a formal conceptualization of knowledge types and algorithms implemented in the KD ontology following up on state-of-the-art developments of a unified data mining theory,

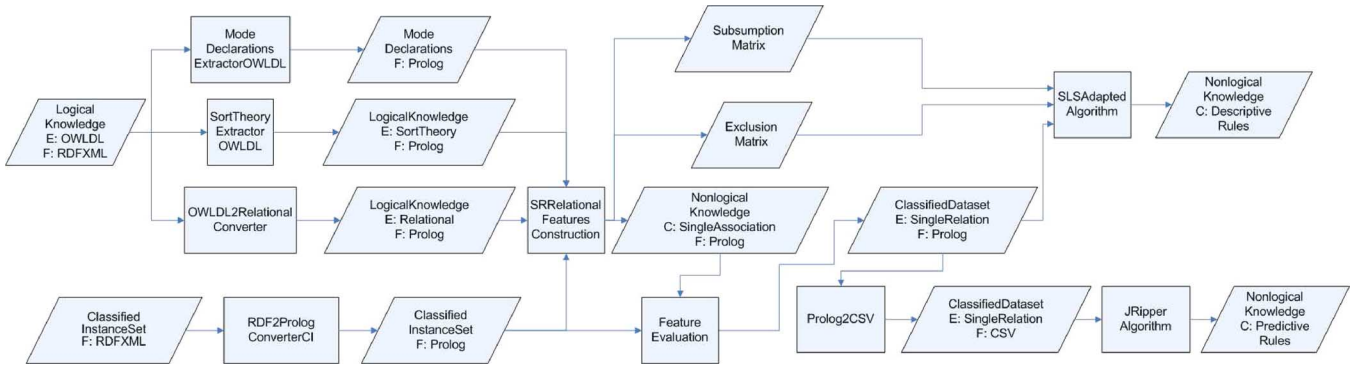

Fig. 7. Abstract workflow generated for obtaining predictive and descriptive rules from annotations of CAD design drawings.

which can describe complex background knowledge and relational data mining algorithms. The ontology is expressed in a standard semantic language OWL, therefore, it can be queried using reasoners such as Pellet. Moreover it is compatible with the OWL-S standard. Only one of the other currently available data mining ontologies deals with relational algorithms and complex knowledge types [24] and it is a heavy-weight ontology aimed at capturing even inner working of the algorithm and therefore currently not suitable for planning. Using the ontology for planning and reuse of workflows grounds the efforts in constructing a unified data mining conceptualization and provides competency questions for its further development.

The developed KD ontology was used to annotate algorithms for relational data mining available within the RDM Manager tool and algorithms available in the Orange data mining platform. We have proposed a subontology for representing data mining workflows in such a way that they can be considered as algorithms and thus allow encapsulating often repeated workflows and constructing hierarchical workflows.

Second, a planning algorithm was implemented and employed to assemble workflows for the task specified by the user's input-output task requirements. We have developed two versions of the algorithm. PDDLPlanner is the baseline solution, which demonstrates the suitability of KD ontology for planning. It uses planning task descriptions in PDDL extracted from the KD ontology and the given user's input-output task requirements. An innovative PelletPlanner is based on directly querying the ontology. We are not aware of any work comparing these two approaches experimentally.

The most time consuming part of the process is ontology classification by the reasoner, however this needs to be performed only once for each session for PelletPlanner and even less often for PDDLPlanner. With increasing complexity of the ontology and the number of annotated algorithms, the PDDLPlanner scales better, however, the PelletPlanner also performs well enough for this application.

We have successfully applied the methodology for constructing workflows in two domains (science and engineering). The workflows generated by our algorithm were complex, but reasonable in that there was no apparent way of simplifying them while maintaining the desired functionality.

A formal capture of the knowledge discovery task by means of a KD ontology can be used to improve repeatability of experiments and to enable reasoning on the results to facilitate reuse of workflows and results. Manual annotation of algorithms, which require extending the core ontology with new knowledge classes can be time consuming and requires expertise in semantic modeling. However, more often algorithms working with already defined knowledge classes are added.

With weak constraints using of rich ontological representation could easily lead to a combinatorial explosion during planning. On the other hand, the ontological representation and using the reasoner during automatic workflow construction enables us to work on different levels of abstraction. Even a hierarchy on algorithms can be exploited in planning. Also, user constraints can be expressed at different abstraction levels. Moreover, automatic workflow construction is expected to facilitate use of complex data mining algorithms by domain experts, e.g., from life sciences and engineering, and also reuse of complex third party algorithms.

In the ongoing work, we are developing and implementing an approach for integrating the PelletPlanner into the Orange data mining toolkit to increase reusability of the methodology. We are also developing an adjusted version of the PelletPlanner exploiting also hierarchy on algorithms and are planning to experiment with planning and goals at different abstraction levels.

In future work, we plan to extend the ontology by descriptions of available computational resources (such as in a GRID environment). This will enable us to produce workflows optimized for execution in a given computing environment as a step towards future automated generation of workflows of data mining services available on the web. We also want to extend modeling of constraints on the algorithms and workflows and to align the ontology to a top-level ontology. Furthermore, we want to introduce more complex heuristics for evaluating the workflows and metrics for workflow similarity and focus on planners more tightly integrating the planner with a reasoner.

REFERENCES

[1] *Workflows for e-Science, Scientific Workflows for Grids*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.. New York: Springer, 2007.

[2] I. Trajkovski, F. Železný, N. Lavrač, and J. Tolar, "Learning relational descriptions of differentially expressed gene groups," *IEEE Trans. Syst. Man, Cybern. C*, vol. 38, no. 1, pp. 16–25, Jan. 2008.

[3] M. Žáková, F. Železný, J. A. Garcia-Sedano, C. Massia-Tissot, N. Lavrač, P. Křemen, and J. Molina, "Relational data mining applied to virtual engineering of product designs," in *Proc. 16th Int. Conf. Inductive Logic Programming*, 2006, pp. 439–453.

[4] Q. Yang and X. Wu, "10 challenging problems in data mining research," *Intl. J. Inf. Tech. Decision Making*, vol. 5, no. 4, pp. 597–604, 2006.

[5] S. Džeroski, "Towards a general framework for data mining," in *Proc. 5th Int. Workshop, Knowledge Discovery in Inductive Databases, KDID'06*, 2007, vol. 4747, LNCS, pp. 259–300.

[6] P. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL web ontology language semantics and abstract syntax," W3C recommendation, 2004. [Online]. Available: http://www.w3.org/TR/owl-semantics/

[7] D. Smith and D. Weld, "Temporal planning with mutual exclusion reasoning," in *Proc. 1999 Int. Joint Conf. Artif. Intell. (IJCAI-1999)*, 1999, pp. 326–333.

[8] L. DeRaedt, "A perspective on inductive databases," *SIKDD Explorations*, vol. 4, no. 2, pp. 69–77, 2002.

[9] V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, and W. Dubitzky, "Grid-enabling data mining applications with DataMiningGrid: An architectural perspective," *Future Generation Comput. Syst.*, vol. 24, no. 4, pp. 259–279, 2008.

[10] *Relational Data Mining*, S. Džeroski and N. Lavrač, Eds. New York: Springer, 2001.

[11] I. Taylor, M. Shields, I. Wang, and A. Harrison, "The Triana workflow environment: Architecture and applications," in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. New York: Springer, 2007, pp. 320–339.

[12] A. Rowe, D. Kalaitzopoulos, M. Osmond, M. Ghanem, and Y. Guo, "The Discovery Net system for high throughput bioinformatics," *Bioinformatics*, vol. 19, pp. 225–231, 2003.

[13] N. L. Khac, M. T. Kechadi, and J. Carthy, "Admire framework: Distributed data mining on data grid platforms," in *Procs. 1st Int. Conf. Softw. Data Technol.*, 2006, vol. 2, pp. 67–72.

[14] A. Ali, O. Rana, and I. Taylor, "Web services composition for distributed data mining," in *Proc. 2005 IEEE Int. Conf. Parallel Processing Workshops, ICPPW'05*, 2005, pp. 11–18.

[15] D. DeRoure, C. Goble, and R. Stevens, "The design and realisation of the myExperiment virtual research environment for social sharing of workflows," *Future Gen. Comput. Syst.*, vol. 25, pp. 561–567, 2008.

[16] K. Morik and M. Scholz, "The MiningMart approach to knowledge discovery in databases," in *Proc. Int. Conf. Machine Learning*, 2004, pp. 47–65.

[17] A. Suyama, N. Negishi, and T. Yamagchi, "Composing inductive applications using ontologies for machine learning," in *Proc. 1st Int. Conf. Discovery Sci.*, 1998, pp. 429–431.

[18] R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schloesser, C. Breitner, R. Engels, and G. Lindner, "Towards process-oriented tool support for knowledge discovery in databases," in *Proc. 1st Eur. Symp. Principles of Data Mining and Knowledge Discovery*, 1997, vol. 1263, pp. 243–253.

[19] A. Bernstein and M. Deanzer, "The NExT system: Towards true dynamic adaptions of semantic web service compositions (system description)," in *Proc. 4th Eur. Semantic Web Conf. (ESWC'07))*, 2007, vol. 4519, LNCS, pp. 739–748.

[20] A. Congiusta, D. Talia, and P. Trunfio, "Distributed data mining services leveraging WSRF," *Future Gen. Comput. Syst.*, vol. 23, no. 1, pp. 34–41, 2007.

[21] Y. Li and Z. Lu, "Ontology-based universal knowledge grid: Enabling knowledge discovery and integration on the grid," in *Proc. 2004 IEEE Int. Conf. Services Comput. (SCC'04)*, 2004, pp. 557–560.

[22] *"OWL-S: Semantic Markup for Web Services", W3C Member Submission*, , 2004. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[23] H. Mannila, "Aspects of data mining," in *Proc. MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995, pp. 1–6.

[24] P. Panov, S. Džeroski, and L. N. Soldatova, "OntoDM: An ontology of data mining," in *Proc. IEEE ICDM Workshops*, 2008, pp. 752–760.

[25] P. Panov and S. Džeroski, Personal Communication, 2009.

[26] D. Rajpathak and E. Motta, "An ontological formalization of the planning task," in *Proc. Int. Conf. Formal Ontologies in Inform. Syst. (FOIS'04)*, 2004, pp. 305–316.

[27] P. Mika, D. Oberle, A. Gangemi, and M. Sabou, "Foundations for service ontologies: Aligning OWL-S to DOLCE," in *Proc. World Wide Web Conference (WWW2004), Semantic Web Track*, 2004, pp. 563–572.

[28] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, S. Koranda, A. Lazzarini, G. Mehta, M. A. Papa, and K. Vahi, "Pegasus and the pulsar search: From metadata to execution on the grid," in *Parallel Processing and Applied Mathematics*, 2004, pp. 821–830.

[29] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim, "Wings for Pegasus: Creating large-scale scientific applications using semantic representations of computational workflows," in *Proc. 19th Annu. Conf. Innovative Appl. Artif. Intell.*, 2007, pp. 1767–1774.

[30] A. Min Tjoa, P. Brezany, and I. Janciak, "Ontology-based construction of grid data mining workflows," in *Data Mining with Ontologies: Implementations, Findings and Frameworks*. Hershey: IGI Global, 2007.

[31] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso, "Planning and monitoring web service composition," in *Proc. AIMSA 2004*, 2004, pp. 106–115.

[32] A. Slominski, "Adapting BPEL to scientific workflows," in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. New York: Springer, 2007, pp. 208–226.

[33] F. Lécué, A. Delteil, and A. Léger, "Applying abduction in semantic web service composition," in *Proc. IEEE Int. Conf. Web Services (ICWS 2007)*, 2007, pp. 94–101.

[34] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN planning for web service composition using SHOP2," *J. Web Semantics*, vol. 1, no. 4, pp. 377–396, 2004.

[35] M. Klusch, A. Gerber, and M. Schmidt, "Semantic web service composition planning with OWLS-XPLAN," in *Proc. 1st Intl. AAAI Fall Symp. Agents and the Semantic Web*, 2005, pp. 55–62.

[36] Z. Liu, A. Ranganathan, and A. Riabov, "A planning approach for message-oriented semantic web service composition," in *Proc. Nat. Conf. AI*, 2007, vol. 5, no. 2, pp. 1389–1394.

[37] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, pp. 189–208, 1971.

[38] E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artif. Intell.*, vol. 5, no. 2, pp. 115–135, 1974.

[39] J. Hoffmann, "Towards efficient belief update for planning-based web service composition," in *Proc. ECAI 2008*, 2008, pp. 558–562.

[40] Železný and N. Lavrač, "Propositionalization-based relational subgroup discovery with RSD," *Machine Learning*, vol. 62, no. 1-2, pp. 33–63, 2006.

[41] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA: Morgan Kaufmann, 2005.

[42] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. E. Patel-Schneider, *The Description Logic Handbook, Theory, Implementation and Applications*. Cambridge, MA: Cambridge Univ. Press, 2003.

[43] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *J. Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007.

[44] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *J. Artif. Intell. Res.*, vol. 14, pp. 253–302, 2001.

[45] A. Blum and M. Furst, "Fast planning through planning graph analysis," *Artif. Intell.*, vol. 90, pp. 281–300, 1997.

[46] E. Sirin and B. Parsia, "SPARQL-DL: SPARQL query for OWL-DL," in *Proc. OWLED 2007 Workshop on OWL: Experiences and Directions*, 2007. [Online]. Available: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-258/paper14.pdf

[47] M. Strauchmann, T. Haase, E. Jamin, H. Cherfi, M. Renteria, and C. Masia-Tissot, "Coaction of semantic technology and virtual reality in an integrated engineering environment," in *KCAP Workshop on Knowledge Management and Sem. Web for Engineering Design*, 2007, pp. 39–47.

[48] M.-A. Krogel, S. Rawles, P. A. Flach, N. Lavrač, and S. Wrobel, "Comparative evaluation of approaches to propositionalization," in *Proc. 13th Int. Conf. Inductive Logic Programming, ILP*, 2003, vol. 2835, LNAI, pp. 197–214.

[49] M. Žáková and F. Železný, "Exploiting term, predicate, and feature taxonomies in propositionalization and propositional rule learning," in *Proc. 18th Eur. Conf. Machine Learning, ECML*, 2007, pp. 798–805.

[50] J. Demsar, B. Zupan, and G. Leban, "Orange: From experimental machine learning to interactive data mining," White Paper, 2004. [Online]. Available: www.ailab.si/orange

[51] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: Tool for the unification of biology," *Nature Genetics*, vol. 25, pp. 25–29, 2000.

Monika Žáková is currently working towards the Ph.D. degree at the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic.

She is a member of the Intelligent Data Analysis Research Group at the Gerstner Laboratory, Czech Technical University in Prague. Her main research interest is relational machine learning, in particular learning with complex background knowledge, and semiautomatic creation of semantic annotations and knowledge discovery workflows.

Petr Křemen is currently working towards the Ph.D. degree at the Department of Cybernetics, Faculty of Electrical Engineering of the Czech Technical University in Prague, Czech Republic.

He is a member of the Knowledge-Based Systems Group. His research interests include semantic web technologies, in particular error explanation in ontologies, ontology development, query answering in semantic web and OWL language.

Filip Železný received the Ph.D. degree in artificial intelligence and biocybernetics from the Czech Technical University in Prague, Czech Republic, and carried out postdoctoral training at the University of Wisconsin, Madison.

He is Head of the Intelligent Data Analysis Research Group at the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague. He was a Visiting Professor at the State University of New York, Binghamton. His main research interest is relational machine learning and its applications in bioinformatics.

Nada Lavrač is Head of the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia. She was the scientific coordinator of the European Scientific Network in Inductive Logic Programming (ILPNET, 1993–1996) and co-coordinator of the 5FP EU project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (SolEuNet, 2000–2003). She is author and editor of several books and conference proceedings, including *Inductive Logic Programming: Techniques and Applications* (Kluwer, 1997) and *Relational Data Mining* (Springer, 2002). Her main research interests are in machine learning, relational data mining, knowledge management, and applications in medicine and bioinformatics.