Modeling issues and choices in the Data Mining OPtimization Ontology

C. Maria Keet[1], Agnieszka Ławrynowicz[2], Claudia d'Amato[3], and Melanie Hilario[4]

[1] School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, and UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa
keet@ukzn.ac.za
[2] Institute of Computing Science, Poznan University of Technology, Poland
agnieszka.lawrynowicz@cs.put.poznan.pl
[3] Dipartimento di Informatica, Universita degli Studi di Bari, Italy
claudia.damato@uniba.it
[4] Artificial Intelligence Laboratory, University of Geneva, Switzerland
melanie.hilario@unige.ch

Abstract. We describe the Data Mining OPtimization Ontology (DMOP), which was developed to support informed decision-making at various choice points of the knowledge discovery (KD) process. It can be used as a reference by data miners, but its primary purpose is to automate algorithm and model selection through semantic meta-mining, i.e., ontology-based meta-analysis of complete data mining processes in view of extracting patterns associated with mining performance. DMOP contains in-depth descriptions of DM tasks (e.g., learning, feature selection), data, algorithms, hypotheses (mined models or patterns), and workflows. Its development raised a number of non-trivial modeling problems, the solution to which demanded maximal exploitation of OWL 2 representational potential. We discuss a number of modeling issues encountered and the choices made that led to version 5.3 of the DMOP ontology.

1 Introduction

Meta-learning, or learning to learn, is defined in computer science as the application of machine learning techniques to meta-data about past machine learning experiments; the goal is to modify some aspect of the learning process in order to improve the performance of the resulting model. Traditional meta-learning focused on the learning phase of the data mining (or KD) process, and regarded learning algorithms as black boxes, correlating the observed performance of their output (learned model) with characteristics of their input (data). Though learning is the central phase of the KD process, the quality of the mined model depends strongly also on other phases (e.g. data cleaning or feature selection). Knowledge on how the different components of the data mining process interact opens up a way for optimizing KD processes. However, besides the existence of CRISP-DM [1], a high-level standard process model for developing knowledge discovery projects, this is far from being understood.

The primary goal of the Data Mining OPtimization Ontology (DMOP, pronounced dee-mope; `http://www.dmo-foundry.org/`) is to support all decision-making steps that determine the outcome of the data mining process. It focuses specifically on DM tasks that require non-trivial search in the space of alternative methods. While DMOP can be used by data mining practitioners to inform manual algorithm selection and model (or parameter) selection, it has been designed to automate these two operations through *semantic meta-mining* [2]. Semantic meta-mining is distinguished from traditional meta-learning by the following properties: i) it extends the meta-learning approach to the full knowledge discovery process taking into account the interdependencies and interactions between the different process operations; ii) it opens up the black box by explicitly analysing DM algorithms along various dimensions to correlate observed performance of learned hypotheses with both data and algorithm characteristics; iii) it represents expertise on the DM process and its components in the DM ontology and knowledge base.

DMOP was developed within the EU FP7 e-LICO project (`http://www.e-lico.eu`) where it provided DM expertise to the Intelligent Discovery Assistant (IDA), comprised of an AI planner and semantic meta-miner. The planner produces a set of candidate workflows that are correct but not necessarily optimal with respect to a given cost function. To help the planner to select the best workflows from a massive set of workflow candidates, an ontology-based meta-learner mines past data mining experiments in order to learn models for recommending best combinations of DM algorithms to be used in a KD process to achieve the best performance for a given problem, data set and evaluation function.

To support semantic meta-mining, DMOP models a detailed taxonomy of algorithms used in KD processes, each described in terms of its underlying assumptions, the cost functions and optimization strategies it uses, the classes of hypotheses—models or pattern sets—it generates, and other properties. This allows meta-learners using DMOP to generalize over algorithms and their properties, including those algorithms that did not appear in the training set, provided they are annotated in DMOP. DMOP is complemented by the DM knowledge base that uses terms from DMOP to model existing data mining algorithms and their implementations (operators) in popular DM software (such as RapidMiner or Weka). Meta-data recorded during data mining experiments are described using terms from DMOP and its associated KB and stored in data mining experiment repositories, providing training and testing data for the meta-miner.

DMOP provides a unified conceptual framework for analyzing DM tasks, algorithms, models, datasets, workflows and performance metrics, and their relationships, which is described in Section 3. To fulfill requirements of this in-depth analysis, we have encountered a number of non-trivial modeling issues in DMOP development, of which the main ones are discussed in Section 4. DMOP's goals and required coverage resulted in using almost all OWL 2 features.

2 Related work

An overview of early approaches to methodical descriptions of DM processes may be found in [2]. The majority of work concerning formal representation of data mining in ontology languages is aimed at the construction of workflows for knowledge discovery. One line of this research deals with the development of distributed KD applications on the Grid [3, 4]. The pre-OWL DAMON ontology provides a characterization of available data mining software in order to allow the user the semantic search for appropriate

DM resources and tools [3]. The ontology of GridMiner Assistant (GMA) [4] aims to support dynamic, interactive construction of data mining workflows in Grid-enabled data mining systems. Other ontologies developed for DM workflow construction are KDDONTO [5], KD ontology [6] and DMWF [7], all of them using OWL as a major representation language. These ontologies are focused on modeling algorithms' inputs/outputs to enable generation of valid compositions of them. For instance, a Hierarchical Task Network (HTN) based planner eProPlan [7], uses DMWF to plan a set of valid workflows based on operator (algorithm implementation) preconditions and effects modeled in DMWF by means of SWRL rules.

Very few existing DM ontologies go beyond supporting workflow construction. OntoDM [8] aims to provide a unified framework for data mining, contains definitions of the basic data mining concepts, but lacks a particular use case. Exposé [9] aims to provide a formal domain model for a database of data mining experiments. It uses OntoDM together with the data mining algorithms from DMOP, and a description of experiments (algorithm setup, execution, evaluation) to provide the basis of an Experiment Markup Language. The primary use of OntoDM and Exposé may thus be viewed as providing controlled vocabulary for DM investigations.

Concluding related work, none of the related ontologies was developed with as goal the optimization of the performance of KD processes. They do not provide sufficient level of details needed to support semantic meta-mining. In particular, the ontologies focused on workflow construction do not model internal characteristics of algorithms but just their inputs and outputs. Hence they aid in answering the question how to build a valid workflow, but not necessarily how to build an *optimal* workflow.

3 Overview of DMOP

The primary goal of DMOP is to support all decision-making steps that determine the outcome of the data mining process. A set of competency questions were formulated at the start of the project, some of which are: "Given a data mining task/data set, which of the valid or applicable workflows/algorithms will yield optimal results (or at least better results than the others)?", "Given a set of candidate workflows/algorithms for a given task/data set, which workflow/algorithm characteristics should be taken into account in order to select the most appropriate one?", and even more detailed ones, such as "Which learning algorithms perform best on microarray or mass spectrometry data?". Due to space limitations, this overview of DMOP will not discuss the answers to these competency questions explicitly.

The core concepts of DMOP (Fig. 1) are the different ingredients that go into the data mining process (DM-Process). The input is composed of a task specification (DM-Task) and training/test data (DM-Data) provided by the user; its output is a hypothesis (DM-Hypothesis), which can take the form of a global model (DM-Model) or a set of local patterns (DM-PatternSet). Tasks and algorithms as defined in DMOP are not processes that directly manipulate data or models, rather they are specifications of such processes. A DM-Task specifies a DM process (or any part thereof) in terms of the input it requires and the output it is expected to produce. A DM-Algorithm is the specification of a procedure that addresses a given Task, while a DM-Operator is a program that implements a given DM-Algorithm. Instances of DM-Task and DM-Algorithm do no

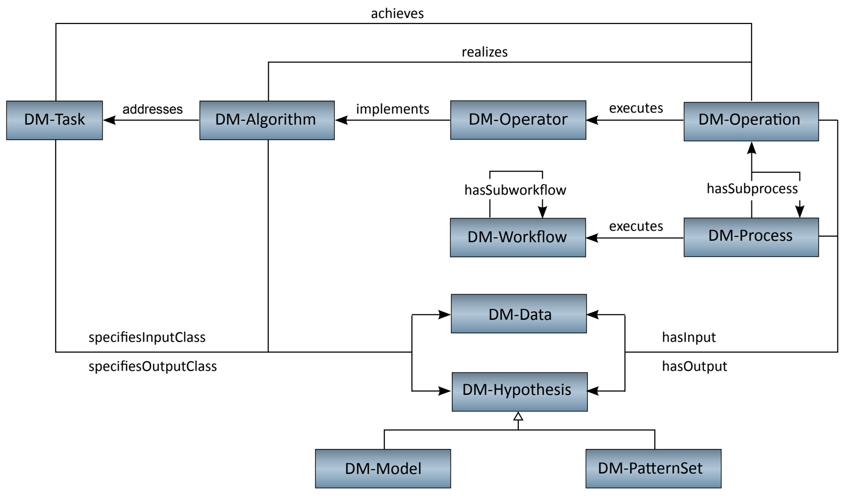

Fig. 1. The core concepts of DMOP.

more than specify their input/output types; only processes called DM-Operations have actual inputs and outputs. A process that executes a DM-Operator also realizes the DM-Algorithm implemented by the operator and achieves the DM-Task addressed by the algorithm. Finally, a DM-Workflow is a complex structure composed of DM operators, a DM-Experiment is a complex process composed of operations (or operator executions). An experiment is described by all the objects that participate in the process: a workflow, data sets used and produced by the different data processing phases, the resulting models, and meta-data quantifying their performance. In the following, the basic elements of DMOP are detailed.

DM Tasks: The top-level DM tasks are defined by their inputs and outputs. A DataProcessingTask receives and outputs data. Its three subclasses produce new data by cleansing (DataCleaningTask), reducing (DataReductionTask), or otherwise transforming the input data (DataTransformationTask). These classes are further articulated in subclasses representing more fine-grained tasks for each category. An InductionTask consumes data and produces hypotheses. It can be either a ModelingTask or a PatternDiscoveryTask, based on whether it generates hypotheses in the form of global models or local pattern sets. Modeling tasks can be predictive (e.g. classification) or descriptive (e.g., clustering), while pattern discovery tasks are further subdivided into classes based on the nature of the extracted patterns: associations, dissociations, deviations, or subgroups. A HypothesisProcessingTask consumes hypotheses and transforms (e.g., rewrites or prunes) them to produce enhanced—less complex or more readable—versions of the input hypotheses.

Data: As the primary resource that feeds the knowledge discovery process, data have been a natural research focus for data miners. Over the past decades meta-learning researchers have actively investigated data characteristics that might explain generalization success or failure. Fig. 2 shows the characteristics associated with the different Data subclasses (shaded boxes). Most of these are statistical measures, such as the number of

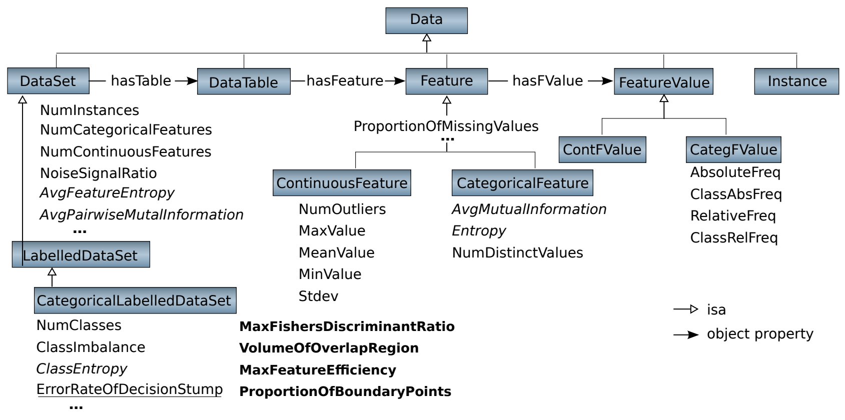

Fig. 2. Data characteristics modeled in DMOP.

instances or the number of features of a data set, or the absolute or relative frequency of a categorical feature value. Others are information-theoretic measures (italicized in the figure). Characteristics in bold font, like the maximum value of Fisher's Discriminant Ratio, which measures the highest discriminatory power of any single feature in the data set, are geometric indicators of data set complexity (see [10] for detailed definitions).

DM Algorithms: The top levels of the Algorithm hierarchy reflect those of the Task hierarchy, since each algorithm class is defined by the task it addresses. However, the Algorithm hierarchy is much deeper than the Task hierarchy: for each leaf class of the task hierarchy, there is an often dense subhierarchy of algorithms that specify diverse ways of addressing each task. For instance, the leaf concept ClassificationModelingTask in the DM-Task hierarchy maps directly onto the ClassificationModelingAlgorithm class, which has three subclasses [11]. *Generative methods* compute the class-conditional densities $p(\mathbf{x}|C_k)$ and the priors $p(C_k)$ for each class C_k. Examples of generative methods are normal (linear or quadratic) discriminant analysis and Naive Bayes. *Discriminative methods* such as logistic regression compute posterior probabilities $p(C_k|\mathbf{x})$ directly to determine class membership. *Discriminant functions* build a direct mapping $f(\mathbf{x})$ from input $\mathbf{x}$ onto a class label; neural networks and support vector classifiers (SVCs) are examples of discriminant function methods. These three Algorithm families spawn multiple levels of descendant classes that are distinguished by the type and structure of the models they generate.

One innovative feature of DMOP is the modeling and exploitation of algorithm properties in meta-mining. All previous research in meta-learning has focused exclusively on data characteristics and treated algorithms as black boxes. DMOP-based meta-mining brings to bear in-depth knowledge of algorithms as expressed in their elaborate network of object properties. One of these is the object property has-quality, which relates a DM-Algorithm to an AlgorithmCharacteristic (Fig. 3). A few characteristics are common to all DM algorithms; examples are characteristics that specify whether an algorithm makes use of a random component, or handles categorical or continuous features. Most other characteristics are subclass-specific. For instance, charac-

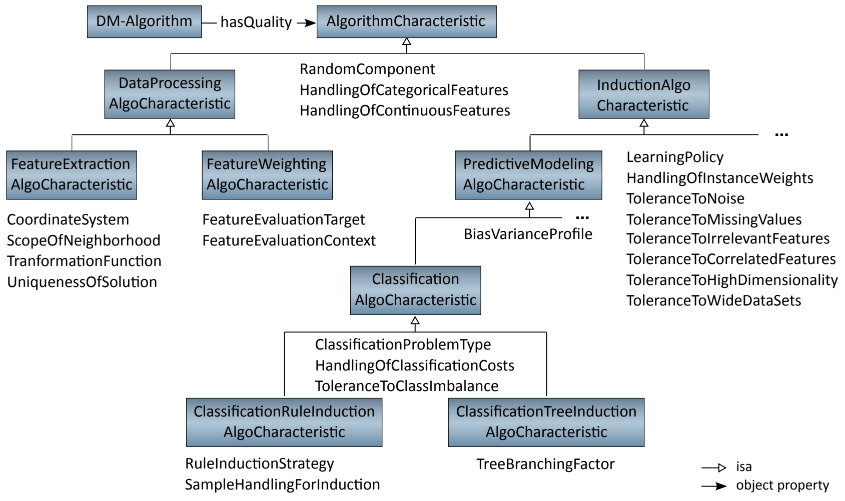
DM-Algorithm
hasQuality
AlgorithmCharacteristic
DataProcessing AlgoCharacteristic
RandomComponent
HandlingOfCategoricalFeatures
HandlingOfContinuousFeatures
InductionAlgo Characteristic
FeatureExtraction AlgoCharacteristic
FeatureWeighting AlgoCharacteristic
PredictiveModeling AlgoCharacteristic
LearningPolicy
HandlingOfInstanceWeights
ToleranceToNoise
ToleranceToMissingValues
ToleranceToIrrelevantFeatures
ToleranceToCorrelatedFeatures
ToleranceToHighDimensionality
ToleranceToWideDataSets
CoordinateSystem
ScopeOfNeighborhood
TranformationFunction
UniquenessOfSolution
FeatureEvaluationTarget
FeatureEvaluationContext
BiasVarianceProfile
Classification AlgoCharacteristic
ClassificationProblemType
HandlingOfClassificationCosts
ToleranceToClassImbalance
ClassificationRuleInduction AlgoCharacteristic
ClassificationTreeInduction AlgoCharacteristic
RuleInductionStrategy
SampleHandlingForInduction
TreeBranchingFactor
isa
object property

Fig. 3. Data mining algorithm characteristics.

teristics such as LearningPolicy (Eager/Lazy) are common to induction algorithms in general, whereas ToleranceToClassImbalance and HandlingOfClassificationCosts make sense only for classification algorithms.

Note that has-quality is only one among the many object properties that are used to model DM algorithms. An induction algorithm, for instance, requires other properties to fully model its inductive bias. For instance, the property assumes expresses its underlying assumptions concerning the training data. SpecifiesOutputType links to the class of models generated by the algorithm, making explicit its hypothesis language or representational bias. Finally, hasOptimizationProblem identifies its optimization problem and the strategies followed to solve it, thus defining its preference or search bias.

4 Modeling choices and their discussion

In this section we present the main modeling choices, issues arisen, and solutions adopted, therewith providing some background as to why certain aspects from the overview in the preceding section are modeled the way they are.

4.1 Meta-modeling in DMOP

Right from the start of DMOP development, one of the most important modeling issues concerning DM algorithms was to decide whether to model them as classes or individuals. Though DM algorithms may have different implementations, the common view is to see particular algorithms as single instances, and not collections of instances. However, the modeling problem arises when we want to express the types of inputs and outputs associated with a particular algorithm. Recall that in fact only processes (executions of workflows) and operations (executions of operators) consume inputs and produce outputs. DM algorithms (as well as operators and workflows) can, in turn, only specify the type of input or output. Inputs and outputs (DM-Dataset and DM-Hypothesis class hierarchy, respectively) are modeled as subclasses of IO-Object class. Then expressing a sentence like "the algorithm C4.5 specifiesInputClass CategoricalLabeledDataSet" became problematic. It would mean that a particular algorithm (C4.5, an instance of DM-Algorithm class) specifies a particular type of input (CategoricalLabeledDataSet, a subclass of DM-Hypothesis class), but classes cannot be assigned as property values to individuals in OWL. To tackle this problem, we initially created one artificial class per each single algorithm with a single instance corresponding to this particular algorithm, as recommended in [12] (e.g. C4.5Algorithm class with single instance C4.5). However, in our case, such modeling led to technical problems. Since each of the four properties—hasInput, hasOutput, specifiesInputClass, specifiesOutputClass—were assigned a common range—IO-Object—it opened a way for making semantically problematic ABox assertions like C4.5 specifiesInputClass Iris, where Iris is a concrete dataset. Clearly, any DM algorithm is not designed to handle only a particular dataset.

We noticed that CategoricalLabeledDataSet could be perceived as an instance of a meta-class—the class of all classes of input and output objects, named IO-Class in DMOP. In this way, the sentence C4.5 specifiesInputClass CategoricalLabeledDataSet delivers the intended semantics. However, we also wanted to express sentences like DM-Process hasInput some CategoricalLabeledDataSet. The use of the same IO object (like CategoricalLabeledDataSet) once as a class (subclass of IO-Object) and at other times as an instance required some form of meta-modeling. In order to implement it, we investigated some available options. This included an approach based on an axiomatisation of class reification proposed in [13], where in a metamodeling-enabled version $\mathcal{O}^{meta}$ of a given ontology $\mathcal{O}$, class-level expressions from $\mathcal{O}$ are transformed into individual asssertions such that each model of $\mathcal{O}^{meta}$ has two kinds of individuals, those representing classes and those representing proper individuals, and meta-level rules are encoded in class level. We resigned from it due to its possible efficiency issues raised in the paper.

To this end, we decided to use the weak form of punning available in OWL 2. Punning is applied only to leaf-level classes of IO-Object; non-leaf classes are not punned but represented by associated meta-classes, e.g., the IO-Object subclass DataSet maps to the IO-Class subclass DataSetClass. Similarly, the instances of DM-Hypothesis class represent individual hypotheses generated by running an algorithm on the particular dataset, while the class DM-HypothesisClass is the meta-class whose instances are the leaf-level descendant classes of DM-Hypothesis. Except for the leaf-level classes, the IO-Class hierarchy structure mimics that of the IO-Object hierarchy.

4.2 Property chains in DMOP

DMOP has 11 property chains, which have been investigated in detail in [14]. The principal issues in declaring safe property chains, i.e., that are guaranteed not to cause unsatisfiable classes or other undesirable deductions, are declaring and choosing properties, and their domain and range axioms. To illustrate one of the issues declaring property chains, we use hasMainTable $\circ$ hasFeature $\sqsubseteq$ hasFeature: chaining requires compatible domains and ranges at the chaining 'points', such as the range of hasMainTable and domain of hasFeature, and with the domain and range of the property on the right-hand side. In this case, hasFeature's domain is DataTable that is a sister-class of hasMainTable's domain DataSet, but the chain forces that each participating entity in hasFeature has to be a subclass of its declared domain class, hence DataSet $\sqsubseteq$ DataTable is derived to keep the ontology consistent. Ontologically, this is clearly wrong, and hasFeature's domain is now set to DataSet or DataTable. Each chain has been analysed in a similar fashion and adjusted where deemed necessary (see [14] for the generic set of tests and how to correct any flaws for any property chain).

DMOP typically contains more elaborate property chains than the aforementioned one. For instance, realizes $\circ$ addresses $\sqsubseteq$ achieves, so that if a DM-Operation realises a DM-Algorithm that addresses a DM-Task, then the DM-Operation achieves that DM-Task, and with the chain implements $\circ$ specifiesInputClass $\sqsubseteq$ specifiesInputClass, we obtain that when a DM-Operator or OperatorParameter implements an AlgorithmParameter or DM-Algorithm and that specifies the input class IO-Class, then the DM-Operator or OperatorParameter specifies the input class IO-Class.

4.3 Alignment of DMOP with the DOLCE foundational ontology

It may seem that the choice whether, and if so how, to map one's domain ontology to a foundational ontology is unrelated to interesting modeling features of the OWL language. This is far from the case, although the modeling arguments (summarised and evaluated in [15]) still do hold. The principal issues from a language viewpoint are: 1) to import or to extend, 2) if import, whether that should be done in whole or just the relevant module extracted from the foundational ontology, 3) how to handle the differences in expressiveness that may exist—and possibly be required—between the foundational ontology and the domain ontology, 4) how to rhyme different modeling 'philosophies' between what comes from Ontology, what is represented in foundational ontologies, and what is permitted in OWL (i.e., features that are objectionable from an ontological viewpoint, such as class-as-instance, nominals, and data properties). Due to space limitations, we only summarise how DMOP was mapped to DOLCE [16].

The two main reasons to align DMOP with a foundational ontology were the considerations about attributes and data properties, where extant non-foundational ontology solutions were partial re-inventions of how they are treated in a foundational ontology (see next section), and reuse of the ontology's object properties. DOLCE, GFO, and YAMATO are available in OWL and have extensive entities on 'attributes' and many reusable object properties. At the time when the need for a mapping arose, YAMATO documentation was limited and GFO less well-known by the authors, which led to the case of mapping DMOP to DOLCE-lite (it now can be swapped for GFO and BFORO anyway, thanks to the foundational ontology library ROMULUS [`http://www.thezfiles.co.za/ROMULUS/`]. The only addition to DOLCE's perdurant branch is that dolce:process has as subclasses DM-Experiment and DM-Operation. Most DM classes, such as algorithm, software, strategy, task, and optimization problem, are subclasses of dolce:non-physical-endurant. Characteristics and parameters of such entities have been made subclasses of dolce:abstract-quality, and for identifying discrete values, classes were added as subclasses of dolce:abstract-region. That is, each of the four DOLCE main branches have been used. Regarding object properties, DMOP reuses mainly DOLCE's parthood, quality, and quale relations. Choosing the suitable DOLCE category for alignment and carrying out the actual mapping has been done manually; some automation to suggest mappings would be a welcome addition. Mapping DMOP into DOLCE had the most effect regarding representing DM characteristics and parameters ('attributes'), which is the topic of the next section.

4.4 Qualities and attributes

A seemingly straightforward but actually rather intricate, and essentially unresolved, issue is how to handle 'attributes' in OWL ontologies, and, in a broader context, measurements. For instance, each FeatureExtractionAlgorithm has as an 'attribute' a transformation function that is either linear or non-linear. One might be tempted to take the easy way out and simply reuse the "UML approach" where an attribute is a binary functional relation between a class and a datatype; e.g., with a simplified non-DMOP intuitive generic example, given a data property hasWeight with as XML data type `integer`, one can declare Elephant $\sqsubseteq$ =1 hasWeight.`integer`. And perhaps a hasWeightPrecise with as data type `real` may be needed elsewhere. And then it appears later on that the former two were assumed to have been measured in kg, but someone else using the ontology wants to have it in lbs, so we would need another hasWeightImperial, and so on. Essentially, with this approach, we end up with exactly the same issues as in database integration, precisely what ontologies were supposed to solve. Instead of building into one's ontology application decisions about how to store the data in the information system (and in which unit it is), one can generalize the (binary) attribute into a class, reuse the very notion of Weight that is the same in all cases, and then have different relations to both value regions and units of measurement. This means to unfold the notion of an object's property, like its weight, from one attribute/OWL data property into at least two properties: one OWL object property from the object to the 'reified attribute'—a so-called "quality property", represented as an OWL class—and then another property to the value(s). The latter, more elaborate, approach is favoured in foundational ontologies, especially in DOLCE, GFO and YAMATO. DOLCE uses the combination Endurant that has a qt relation to Quality (disjoint branches) that, in turn, has a ql relation to a Region (a subclass of the yet again disjoint Abstract branch). While this solves the problem of non-reusability of the 'attribute' and prevents duplication of data properties, neither ontology has any solution to representing the actual values and units of measurements. But they are needed for DMOP too, as well as complex data types, such as an ordered tree and a multivariate series.

We considered in more detail related work on qualities, measurements and similar proposals from foundational ontologies, to general ontologies, to domain ontologies for the experimental sciences [16–20]. This revealed that the measurements for DMOP are not measurements in the sense of recording the actual measurements, their instruments, and systems of units of measurements, but more alike values for parameters, e.g., that the TreeDepth has a certain value and a LearningPolicy is eager or lazy, and that some proposals, such as OBOE [18], are versions of DOLCE's approaches[1]. This being the case, we opted for the somewhat elaborate representation of DOLCE, and added a minor

[1] DOLCE materials differ slightly, with quale as relation in [16] and as unary in [17] and in the `DOLCE-lite.owl`, and Region is a combination of a (data) value + measurement unit

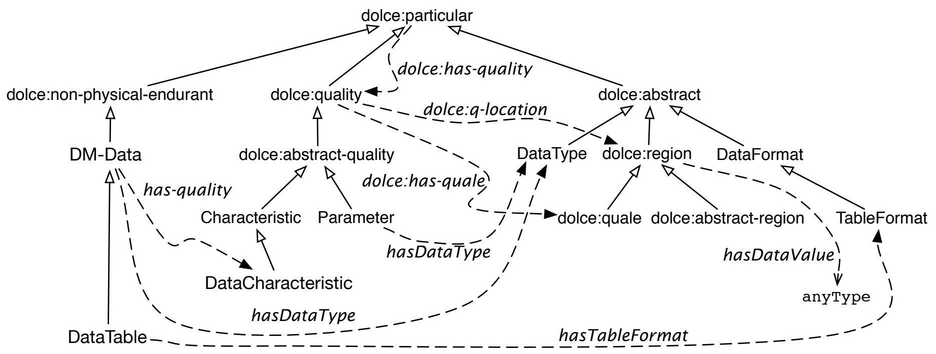

Fig. 4. Condensed section and partial representation of DMOP regarding 'attributes'.

extension to that for our OWL ontology in two ways (see Fig. 4): *i)* DM-Data is associated with a primitive or structured DataType (which is a class in the TBox) through the object property hasDataType, and *ii)*, the data property hasDataValue relates DOLCE's Region with any data type permitted by OWL, i.e., `anyType`. In this way, one obtains a 'chain' from the endurant/perdurant through the dolce:has-quality property to the quality, that goes on through the dolce:q-location/dolce:has-quale property to region and on with the hasDataValue data property to the built-in data type (instead of one single data property between the endurant and the data type). For instance, we have ModelingAlgorithm $\sqsubseteq$ =1 has-quality.LearningPolicy, where LearningPolicy is a dolce:quality, and then LearningPolicy $\sqsubseteq$ =1 has-quale.Eager-Lazy, where Eager-Lazy is a subclass of dolce:abstract-region (that is a subclass of dolce:region), and, finally, Eager-Lazy $\sqsubseteq$ ≤ 1 hasDataValue.`anyType`, so that one can record the value of the learning policy of a modeling algorithm. In this way, the ontology can be linked to many different applications, who even may use different data types, yet still agree on the meaning of the characteristics and parameters ('attributes') of the algorithms, tasks, and other DM endurants.

A substantial amount of classes have been represented in this way: dolce:region's subclass dolce:abstract-region has 43 DMOP subclasses, which represent ways of carving out discrete value regions for the characteristics and parameters of the endurants DM-Data, DM-Algorithm, and DM-Hypothesis. Characteristic and Parameter are direct subclasses of dolce:abstract-quality, which have 94 and 42 subclasses, respectively.

4.5 Other modeling considerations

There are several other OWL 2 features that are or were used in DMOP to model the subject domain. One that was used in some cases was the `ObjectInverseOf`, so that for some object property hasX in an OWL 2 ontology, one does not have to extend the vocabulary with an Xof property and declare it as the inverse of hasX with

condensed into one (e.g. "80 kg") in [16] to deal with attribute values/qualia (there were no examples in [17] and the `DOLCE-lite.owl`)

`InverseObjectProperties`, but one can use the meaning of Xof with the axiom `ObjectInverseOf(hasX)` instead. Both approaches were used initially and eventually inverses were added explicitly where needed, for readability of the class expressions.

The so-called "object property characteristics" have been used sparingly, and only the basic 'functional' characteristic is asserted. Local reflexivity was investigated on a subsumes property for instances in DMOP v5.2, but eventually modeled differently with classes and metamodeling/punning to achieve the desired effect. DOLCE's parthood is transitive, hence it should be transitive in DMOP as well, but it was discovered after the release of v5.3 that the object property copy function in Protégé does not copy any property characteristics. This will be corrected in the next version. In general, though, if one extracts the properties from an ontology, it ought to take with it at least the characteristics of those properties (and, though arguable, also any domain and range axioms declared for the selected properties).

The desire to specify the order of subprocesses remains, and a satisfactory solution has yet to be found. Likewise, there are object properties in the current DMOP version that merit closer investigation whether they are indeed more specific versions of parthood; e.g., a DecisionCriterion is part of DecisionRule, but it merits further investigation whether, e.g., hasDecisionTarget with as domain DecisionStrategy and range DecisionTarget is indeed a subproperty of has-part.

5 Conclusions

In this paper, the DMOP ontology has been presented. It provides a unified conceptual framework for analyzing DM tasks, algorithms, models, datasets, workflows and performance metrics, and their relationships. While modeling data mining knowledge in DMOP, almost all OWL 2 features were used to solve a number of non-trivial modeling issues that had been encountered. These include: i) the hurdle of relating instances to classes and using classes as instances (and vv.), which has been solved by exploiting the weak form of metamodeling with OWL's punning available in OWL 2; ii) finding and resolving in a systematic way the undesirable deductions caused by property chains; iii) representation of 'attributes', where its solution is ontology-driven yet merged with OWL's data property and built-in data types to foster their reuse across applications; iv) linking to a foundational ontology; and v) the considerations on adoption of the OWL 2 `ObjectInverseOf` to avoid extending the vocabulary with new properties to be declared as the inverse of existing properties, but where human readability prevailed.

Current and future work includes investigating aspects such as DMOP object properties, the effects of `ObjectInverseOf` vs. `InverseObjectProperties`, and the deductions with respect to the DM algorithms branch and metamodeling. DMOP (version 5.2) has already been applied successfully to the meta-mining task [2]. Further meta-mining experiments will be performed with the goal of validating the modelling choices introduced in version 5.3.

Acknowledgements. This work was supported by the European Community 7th framework ICT-2007.4.4 (No 231519) "e-LICO: An e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Science". We thank all our partners and colleagues who have contributed to the development of the DMOP ontology: Huyen Do, Simon Fischer, Dragan Gamberger, Lina Al-Jadir, Simon Jupp, Alexandros Kalousis, Petra Kralj Novak, Babak Mougouie, Phong Nguyen, Raul Palma, Robert Stevens, Anze Vavpetic, Jun Wang, Derry Wijaya, Adam Woznica.

References

1. Shearer, C.: The CRISP-DM model: The new blueprint for data mining. Journal of Data Warehousing **5**(4) (2000) 13–22
2. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-based meta-mining of knowledge discovery workflows. In: Meta-Learning in Computational Intelligence. Volume 358 of Studies in Computational Intelligence. Springer (2011) 273–315
3. Cannataro, M., Comito, C.: A data mining ontology for grid programming. In: Proc. of 1st Int. WS. on Semantics in Peer-to-Peer and Grid Computing. (2003) 113–134
4. Brezany, P., Janciak, I., Tjoa, A.M.: Ontology-based construction of grid data mining workflows. In: Data Mining with Ontologies. Hershey (2007)
5. Diamantini, C., Potena, D., Storti, E.: Supporting users in KDD processes design: a semantic similarity matching approach. In: Proc. of the Planning to Learn Works. (2010) 27–34–134
6. Záková, M., Kremen, P., Zelezný, F., Lavrac, N.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE Trans. Automation Science & Engineering **8**(2) (2011) 253–264
7. Kietz, J., Serban, F., Bernstein, A., Fischer, S.: Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In: Proc of the ECML/PKDD'10 Workshop on Third Generation Data Mining (SoKD'10). (2010) 1–12
8. Panov, P., Dzeroski, S., Soldatova, L.N.: OntoDM: An ontology of data mining. In: ICDM Workshops, IEEE Computer Society (2008) 752–760
9. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases - a new way to share, organize and learn from experiments. Machine Learning **87**(2) (2012) 127–158
10. Ho, T.K., Basu, M.: Measures of geometrical complexity in classification problems. In: Data Complexity in Pattern Recognition. Springer (2006) 3–23
11. Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)
12. Noy, N., Uschold, M., Welty, C.: Representing Classes As Property Values on the Semantic Web (2005) W3C Working Group Note, http://www.w3.org/TR/swbp-classes-as-values/.
13. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: Proc. of ISWC'10, Springer-Verlag (2010) 257–272
14. Keet, C.M.: Detecting and revising flaws in OWL object property expressions. In ten Teije, A., et al., eds.: Proc. of EKAW'12. Volume 7603 of LNAI., Springer (2012) 252–266
15. Keet, C.M.: The use of foundational ontologies in ontology development: an empirical assessment. In: Proc. of ESWC'11. Volume 6643 of LNCS., Springer (2011) 321–335
16. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). (2003) http://wonderweb.semanticweb.org.
17. Masolo, C., Borgo, S.: Qualities in formal ontology. In: Proceedings of the Workshop on Foundational Aspects of Ontologies (FOnt 2005). (2005) Koblenz, Germany, Sept. 2005.
18. Saunders, W., Bowers, S., O'Brien, M.: Protégé extensions for scientist-oriented modeling of observation and measurement semantics. In: Proc. of OWLED'11. Volume 796 of CEUR-WS. (2011)
19. Bowers, S., Madin, J.S., Schildhauer, M.P.: A conceptual modeling framework for expressing observational data semantics. In: Proc. of ER'06. Volume 5231 of LNCS., Springer (2008) 41–54
20. Hodgson, R., Keller, P.J.: QUDT - quantities, units, dimensions and data types in OWL and XML. Online (September 2011) http://www.qudt.org/.