Data Driven Concept Refinement to Support Avionics Maintenance

Luis Palacios Medinacelli[1,2] Yue Ma[2] Gaëlle Lortal[1] Claire Laudy[1] Chantal Reynaud[2]
[1]LRASC, Thales Research & Technology, Palaiseau, France
[2]LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, France
{gaelle.lortal,claire.laudy}@thalesgroup.com, {palacios,ma,cr}@lri.fr

Abstract

Description Logic Ontologies are one of the most important knowledge representation formalisms nowadays which, broadly speaking, consist of classes of objects and their relations. Given a set of objects as samples and a class expression describing them, we present ongoing work that formalizes which properties of these objects are the most relevant for the given class expression to capture them. Moreover, we provide guidance on how to refine the given expression to better describe the set of objects. The approach is used to characterize test results that lead to a specific maintenance corrective action, and in this paper is illustrated to define sub-classes of aviation reports related to specific aircraft equipment.

1 Introduction

Given a DL-Ontology we might find that some concepts definitions are too generic, in the sense that they are not rich enough to capture only the intended objects, or that the definition does not describe them properly. In order to have more control on what is expressed, having sub-types of such concepts is useful, since we can make distinctions between the objects that could not be made before. Our motivation comes from the avionics maintenance domain [Palacios *et al.*2016] , where we find several levels of maintenance. One of them involves shop repair, where an equipment found faulty on an aircraft is to be repaired or replaced. In this scenario, several tests need to be run to find out the exact part(s) of the equipment which is faulty and causes failures. Once the tests are run, it is up to mechanic experts to determine the possible components of the equipment involved in the failure, and the repairs/replacements to be done. For a maintenance process it is difficult to establish *a priori* what are the actions to be taken to return the equipment to a fully functional state. Establishing the most probable actions is useful to shorten the examination and repair time, thus gaining in efficiency and lowering the costs. In this paper, we aim to model this problem and propose a primitive method. The idea is based on the fact that we know by a large amount of historical shop repair reports the test results and repair decisions made accordingly. We can consider the closed incidents as positive samples of a repair action, the incidents that do not require this repair action as negative ones, and use tests as features of each sample. Then identifying important tests results equals identifying key features that distinguish positives from negatives. Finally we use these key features to provide a description for the sub-set of tests that lead to a specific maintenance action. Our approach thus can be used to obtain formal patterns to capture a set of samples; properties that distinguish positive from negative samples; guide to refine a concept expression and enrich patterns that can be served as features for clustering or classifying samples.

2 Positioning

Mining formal patterns from data has been identified as an important task in many different research communities. Depending on the target language used for representing a pattern, we can divide the existing work into different categories.

Concept Learning Similar to our paper, DL-Learner [Lehmann2009], a state-of-the art tool for concept learning, is based on description logic. It provides a framework for supervised machine learning using sev-

In: Proceedings of IJCAI Workshop on Semantic Machine Learning (SML 2017), Aug 19-25 2017, Melbourne, Australia.

eral algorithms which are highly parameterizable. It is based on refinement operators like CELOE for OWL expressions and ELTL for the *EL* family of DLs. Depending on the desired properties of the operator and the DL-constructors allowed, the operator traverses the space of possible concept expression in a top-down or bottom-up way. Then these concept expressions are evaluated, using heuristics, to find the most suitable ones. Shorter and more simple expressions are preferred by these algorithms.

A useful list of approaches for concept learning and their characteristics can be found in [Tran *et al.*2012] where they position their approach based on bisimulations with respect to other techniques.

In [Divroodi and Nguyen2015], they study how to establish whether the signature of an ontology (the concepts, roles and individuals along with the DL-family chosen) is sufficient to distinguish between two objects. Broadly speaking, if two objects belong to the same equivalence class, they are indistinguishable. On the other hand, given a set of samples, if there exists an expression in the underlying language that can capture this set of samples, then it must be the union of some equivalence classes. They use this notion to provide an algorithm that learns a concept expression, by partitioning the domain with respect to the equivalence classes. One interesting feature of their approach, is that it offers a formal way of defining approximations to a concept expression based on rough sets [Nguyen and Szałas2013] by using the similarity classes and the underlying language.
In the above mentioned approaches, the goal is to find a concept expression that best describes a set of samples, by refining the concept. In a specialization scenario, if a false positive, can be left out of the extension of the concept by adding a restriction to the the objects that belong to it, we know that the selected property separates the false positive from the rest of the samples. Pointing out these properties and objects, is the difference and contribution of this paper with respect to the above mentioned approaches.

Graph mining Graph structures are used within a variety of applications in order to represent structured information. The issue of mining interesting patterns in these graphs structures has thus emerged and a large amount of researches focus on algorithms enabling graph mining. Regarding our application, interesting patterns mean for instance recurring or frequent patterns that can be found either in a big graph or in a set of (smaller) graphs. It may also mean finding significant patterns, either semantically or statistically representative for the instance. From a Machine Learning point of view, interesting patterns are those that well discriminate positive vs. negative samples.

One of the main problems addressed by the different work raised in the subgraph isomorphism issue. Subgraph isomorphism is an NP-complete problem. Thus, mining subgraphs of a graph means studying an exponential number of subgraphs. In [Yan and Han2003], the authors propose an algorithm, CloseGraph to mine closed frequent graphs. The algorithm uses pruning methods in order to reduce the exploration space. A closed graph in a set of graphs is a graph, for which it exists no proper subgraph that has the same support. Therefore, closed graph patterns will be the largest patterns that can be found in a graph database for a given problem.

[Motoda2007], [Inokuchi *et al.*2000] and [Yoshida *et al.*1994] present two approaches for extracting frequent subgraphs: AGM and GBI. AGM relies on the representation of the graphs by adjacency matrices and GBI relies on the chunking of adjacent nodes in order to generate subgraphs and the rewriting of the graphs given the selected subgraphs as new nodes.

Using graph mining, the most relevant structures of a set of graphs can be found. It can be used to find patterns that discriminate positive and negative examples. This is very similar to our problem of learning a concept from examples as patterns describe the common parts of the examples. However, no direct control over the signature is given, neither the possibilities to extend a concept are taken into account. Graph mining techniques can be used as initial step for our approach to find a first substructure that discriminate partially positives and negatives examples. Our feature extraction algorithm may then be applied in order to not remain at a structure level but to consider the semantics, focusing on the relevant properties. Furthermore the associated signature provides theoretical limits to what can be expressed.

The rest of the paper is organized as follows: in section 3 we introduce the approach and the necessary definitions. Then, in section 4 we show a use case based on aviation reports, where we refine a concept based on expert knowledge. Finally, in section 5 the conclusions and further works are presented.

3 Defining the most relevant features

We define ontologies based on [Baader2003], where an ontology is a tuple $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ with $\mathcal{T}$ being the T-Box and $\mathcal{A}$ the A-Box. The T-Box contains the set of concept and role definitions, while the A-Box contains assertions about the definitions in the T-Box. The signature $\Sigma_{\mathcal{O}}$ of an ontology $\mathcal{O}$ is the set of all concept names, role names and individuals in $\mathcal{O}$. For details refer to [Divroodi and Nguyen2015]. To make the approach simpler, in the following we consider $\mathcal{T} = \emptyset$ and the assertions in the A-Box are of the form $A(x)$

or $r(x, y)$, where A and r are atomic.

Given an ontology $\mathcal{O}$, a set of samples $\mathcal{X} = Pos \cup Neg$ (where Pos is a set of positive samples and Neg a set of negative samples) and a concept $\mathcal{C}$ describing $\mathcal{X}$ up to certain degree ψ, we are interested in finding a DL-expression $\mathcal{C}'$ with a better degree ψ' to describe $\mathcal{X}$, if it exists. The value of ψ is to be defined in accordance to the problem by the user (for example the recall, the accuracy, f-measure, *etc.*). Thus, the problem consists in that: a) $\mathcal{C}$ captures some unintended objects (false positives),b) it does not capture some intended objects (false negatives) or c) both.
We take the case of false positives to illustrate the approach, where $\mathcal{C}$ captures some negative samples. In this scenario, we would like to make $\mathcal{C}$ more specific, that is to add restrictions to the objects that belong to this concept, in such a way that a) some (or all) false positives are not considered anymore and b) we preserve all (or most) of the positive samples. Given a concept $\mathcal{C}$ and a positive instance $x \in Pos$, we want to know what are the properties to consider, if we want to specialize $\mathcal{C}$. Intuitively the process is as follows:
Any instance and its relation to other objects can be represented as a directed (acyclic) graph, with the nodes representing the objects and the edges representing the relations between them, as stated by the A-Box. For example consider the A-Box :

$$\mathcal{A} : \{r_1(x, y), r_2(y, w), r_3(y, z)\}$$

we obtain: Since a concept expression $\mathcal{C}$ is given as part

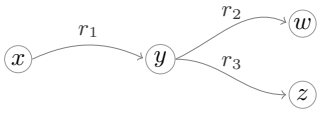

Figure 1: The graph representation of object x

of the input, we can determine which are the properties of object x that are necessary for $\mathcal{C}$ to capture it. Assume $\mathcal{C} \equiv \exists r_1.\exists r_2.\top$, then the assertion $r_3(y, z)$ is not relevant for deciding whether $\mathcal{C}(x) = \top$ holds, and a simpler representation of x can be obtained: This

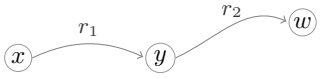

Figure 2: The graph representation of object x, w.r.t. $\mathcal{A} \setminus r_3(y, z)$

representation allows us to establish up to which point the structure of the object x is relevant for $\mathcal{C}$. From the original A-Box, we know that some of the objects to which x is connected, are themselves connected to other objects by relations not considered by $\mathcal{C}$. In our example object y is connected to object z through relation r_3. We are interested in these objects, since they provide all the properties that we can consider to further specialize $\mathcal{C}$ and still capture x, and therefore represent the relevant properties to capture the set of positive instances. In order to provide a formal definition for these objects, let us first introduce some necessary notions.

Definition 1 *Given an ontology $\mathcal{O}$ with signature $\Sigma_{\mathcal{O}}$, two objects x, y and an A-Box $\mathcal{A}$, we say that there exists a* binary relation closure *between x and y, denoted by $r \uparrow(x, y)$, if $x = y$ or if there exists a path of the form:*

$$\{r^1(x, z_1), r^2(z_1, z_2), \ldots, r^m(z_{n-1}, z_n)\} \subseteq \mathcal{A}$$

$$\textit{with } n \geq 1,\ z_n = y, \textit{ and } r^j \in \Sigma_{\mathcal{O}} (1 \leq j \leq m).$$

Next, we want to establish which of the edges of the graph representing the object are necessary for a given concept to capture the object. This is formalized by the following definition:

Definition 2 *Given an object x, a concept $\mathcal{C}$, an A-Box $\mathcal{A}$ and a binary relation $r(y, z) \in \mathcal{A}$, we say that $r(y, z)$ is* necessary *for $\mathcal{C}$ to capture x iff:*

$$\mathcal{C}(x) = \top \textit{ w.r.t } \mathcal{A}, r \uparrow(x, y), r(y, z) \in \mathcal{A} \textit{ but}$$

$$\mathcal{C}(x) = \bot \textit{ w.r.t } \mathcal{A} \setminus r(y, z)$$

Likewise, we say that $r(y, z)$ is unnecessary *if*

$$\mathcal{C}(x) = \top \textit{ w.r.t } \mathcal{A} \setminus r(y, z)$$

still holds. Additionally, we say that an object o is necessary *if $o = x$ or:*

$$\exists z \mid r(z, o) \in \mathcal{A} \textit{ s.t. } r(z, o) \textit{ is necessary.}$$

Note that depending on the concept $\mathcal{C}$ and the content of the A-Box, a unnecessary binary relation might become necessary, therefore several possible answers might exist. For example take the concept $\mathcal{C} \equiv \exists r.\top$ and the A-Box: $\mathcal{A} = \{r(x, y), r(x, z)\}$, then we have:

$$\mathcal{C}(x) = \top \text{ w.r.t } \mathcal{A} \setminus r(x, y)$$

Concluding that $r(x, y)$ is not necessary, but only as long as $r(x, z) \in \mathcal{A}$ (and vice-versa). Given a definition for the properties and objects necessary for an object x to belong to a concept $\mathcal{C}$, we can also obtain those that are not necessary. These (unnecessary) properties are linked to the object x but are not required by $\mathcal{C}$. As such they can be seen as candidate properties for specializing $\mathcal{C}$ and still capture x. These are the properties of special objects hereafter called leafs of x, defined by:

Definition 3 *Given an object x, a concept $\mathcal{C}$ and an A-Box $\mathcal{A}$ the set of* leafs *of x w.r.t $\mathcal{C}$ is given by:*

$$Leafs_{x,\mathcal{C}} = \{y \mid r(y,z) \in \mathcal{A}$$

$$\textit{where } y \textit{ is necessary for } \mathcal{C} \textit{ to capture } x$$

$$\textit{but } r(y,z) \textit{ is not necessary for } \mathcal{C} \textit{ to capture } x\}$$

Intuitively the set $Leafs_{x,\mathcal{C}}$ represents all those objects y in the frontier of x w.r.t. $\mathcal{C}$, in the sense that no further edges of the graph representing x are considered by $\mathcal{C}$ to decide whether the object x belongs to it.

Definition 4 *Given an object x, a concept $\mathcal{C}$ and the set $Leafs_{x,\mathcal{C}}$ w.r.t. an A-Box $\mathcal{A}$, the set of* extensions *$Ext_{x,\mathcal{C}}$ to specialize $\mathcal{C}$ w.r.t. x is defined by:*

$$Ext_{x,\mathcal{C}} = \{r(y,z) \in \mathcal{A} \mid y \in Leafs_{x,\mathcal{C}}$$

$$\textit{and } r(y,z) \textit{ is not necessary for } \mathcal{C} \textit{ to capture } x \ \}$$

Intuitively $Ext_{x,\mathcal{C}}$ provides all those role names through which $\mathcal{C}$ can be specialized and still capture x. The set of leafs and their properties are the answer to our problem (they provide the ways in which we can specialize $\mathcal{C}$, from where we can derive the conflictive properties and the conflicting objects). Before we present an algorithm to obtain the necessary properties, let us show that the definitions are not sufficient alone. Consider now $\mathcal{C} \equiv \exists r.\top$ and $\mathcal{A} = \{r(x,y), r(y,w), r(x,z)\}$, we have $Leafs_{x,\mathcal{C}} = \{x\}$ and $Ext_{x,\mathcal{C}} = \{r(x,y), r(x,z)\}$. We can indeed specialize $\mathcal{C}$ using these properties, but nothing in the above answer gives us the information that either $r(x,y)$ or $r(x,z)$ is required for $\mathcal{C}(x) = \top$ to hold (we just know they are not necessary). To obtain this information, we take the unnecessary relations and remove them one by one from the $A - Box$ until a minimal set of necessary role assertions is reached. We now introduce an algorithm to compute a minimal set of necessary role assertions, from which we can extract $Leafs_{x,\mathcal{C}}$ and $Ext_{x,\mathcal{C}}$:

In Algorithm 1 we first compute R_x (2), which is the subset of the A-Box $\mathcal{A}$ containing all those role assertions in a path from x:

$$r(x,y) \in R_x \text{ since } r \uparrow (x,x) \text{ and } r(x,y) \in \mathcal{A}$$

$$r(y,w) \in R_x \text{ since } r \uparrow (x,y) \text{ and } r(y,w) \in \mathcal{A}$$

$$r(x,z) \in R_x \text{ since } r \uparrow (x,x) \text{ and } r(x,z) \in \mathcal{A}$$

$$\text{We have : } R_x = Copy_R = \{r(x,y), r(y,w), r(x,z)\}$$

(3) makes a copy $Copy_R$ of R_x from which we will sequentially remove the last elements of each path. (4) establishes as the candidates $Cand_R$ to be tested all those role assertions that do not have further outgoing edges (that is, the last elements of a path):

$$Cand_R = \{r(y,w), r(x,z)\}$$

(5) creates the set of all concept assertions about all the objects that take part in a relation in R_x. In our example is empty. Then we start the while loop to test all assertions for necessity until no more candidates are found. (7) takes one candidate at the time, and (8) tests if its necessary. The unnecessary assertions are removed from R_x (9). Any assertion already tested is removed from $Copy_R$ by (11) in order not to test them twice. First we test $r(y,w)$:

$$\mathcal{C}(x) = \top \text{w.r.t.} \{R_x \setminus r(y,w)\} \cup C_x \text{ , remove } r(y,w)$$

$$R_x = \{r(x,y), r(x,z)\} \text{ , } Copy_R = \{r(x,y), r(x,z)\}$$

Then, $r(x,z)$ is tested:

$$\mathcal{C}(x) = \top \text{w.r.t.} \{R_x \setminus r(x,z)\} \cup C_x \text{ , remove } r(x,z)$$

$$R_x = \{r(x,y)\} \text{ , } Copy_R = \{r(x,y)\}$$

Once all identified candidates are tested, the set $Candidates_R$ is re-computed (13) considering only those assertions remaining in $Copy_R$,

$$Cand_R = \{r(x,y)\}$$

A second run of the while loop tests $r(x,y)$ yielding:

$$\mathcal{C}(x) = \bot \text{w.r.t.} \{R_x \setminus r(x,y)\} \cup C_x \text{ , keep } r(x,y)$$

$$R_x = \{r(x,y)\} \text{ , } Copy_R = \{\}$$

Since there are no more candidates to test, the output of the algorithm is the modified set R_x which is a minimal set of necessary role assertions for $\mathcal{C}(x) = \top$ [1]:

$$R_x = \{r(x,y)\}$$

Algorithm 1 Minimal set of necessary role assertions

1: **input:** $(x, \mathcal{C}(x) = \top, \mathcal{A})$
2: $R_x = \{r(y,z) \mid r \uparrow (x,y), r(y,z) \in \mathcal{A}\}$
3: $Copy_R = R_x$
4: $Cand_R = \{r(y,z) \in R_x \mid \nexists w \text{ s.t. } r(z,w) \in R_x\}$
5: $C_x = \{\mathcal{D}(z) \in \mathcal{A} \mid z = x \text{ or } \exists y \text{ s.t. } r(y,z) \in R_x\}$
6: **while** $Cand_R \neq \emptyset$ **do**
7: **for** $r(y,z) \in Cand_R$ **do**
8: **if** $\mathcal{C}(x) = \top$ w.r.t. $\{R_x \setminus r(y,z)\} \cup C_x$ **then**
9: remove $r(y,z)$ from R_x
10: **end if**
11: remove $r(y,z)$ from $Copy_R$
12: **end for**
13: $Cand_R = \{r(y,z) \in Copy_R \mid \nexists w \text{ s.t. } r(z,w) \in Copy_R\}$
14: **end while**
15: **return:** R_x

[1] The proof for this property remains as further work.

From this set we can easily construct $Leafs_{x,\mathcal{C}}$ and $Ext_{x,\mathcal{C}}$, following their definitions:

$$Leafs_{x,\mathcal{C}} = \{x, y\} \ , \ Ext_{x,\mathcal{C}} = \{r(y, w), r(x, z)\}$$

Where the possible extensions to consider to specialize $\mathcal{C}$ are $r(y, w), r(x, z)$, which is the intended answer.

4 Use case

Since the data specific to aviation maintenance is restricted for disclosure, we provide an example based on aviation incidents. Similarly as in aviation maintenance, where we want to find sub-types of failures based on their features, here we want to obtain interesting sub-concepts of aviation issues. We count with an ontology $\mathcal{O}_{ASRS}$ representing the reported aviation incidents from the ASRS [2] database, and a set of equipment used in aviation, from which we selected GPS. In this scenario, we are interested on which properties to take into account to obtain those *"GPS related aviation issues"* and those *"aviation issues where the GPS presented a problem"*. We proceed as follows: in the ASRS website, a classification of some reports made by experts is given, one of such sets is *"GPS related reports"*. This provides us with the set of positive instances Pos [3]. As the set of negative instances Neg, we have selected reports related to other types of incidents disjoint with Pos:

$$Pos = \{1336, 1347, 1359\} \ , \ Neg = \{1361\}$$

The A-Box $\mathcal{A}$ is composed of the following concepts and roles:

$$\begin{aligned}
AviationIssue &= \{1336, 1347, 1359, 1361\} \\
Aircraft &= \{A1, A2, A3, A4\} \\
NavInUse &= \{GPS, FMS\} \\
CompProb &= \{Mf, IO\} \\
involves &= \{(1336, A1), (1347, A2), (1359, A3), \\
&\quad (1361, A4)\} \\
usesNav &= \{(A1, GPS), (A2, GPS), (A3, GPS), \\
&\quad (A3, FMS)\} \\
repProblem &= \{(A1, Mf), (A2, Mf), (A3, IO), \\
&\quad (A4, Mf)\} \\
hasNarrative &= \{(1336, \text{"..GPS.."}), (1347, \text{"..GPS.."}), \\
&\quad (1359, \text{"..GPS.."}), (1361, \text{"..GPS.."})\}
\end{aligned}$$

(where: Mf=Malfunctioning,CompProb=ComponentProblem, IO=Improperly operated,repProblem=reportedProblem, NavInUse=Navigation system in use)

Assume as input we are given a concept expression of the form:

$$\mathcal{C} \equiv \exists involves.\top$$

[2]Aviation Safety Report System (https://asrs.arc.nasa.gov)

[3]The samples have been simplified for this paper.

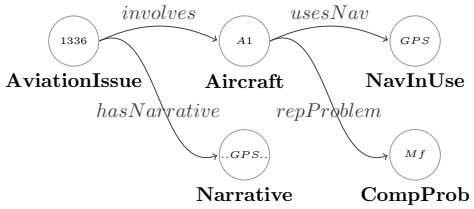

Figure 3: The graph representation of object 1336, instance of AviationIssue concept. Concepts are represented in **bold**, values as nodes, and relations as edges.

It is easy to see that the AviationIssue 1336 is an instance of $\mathcal{C}$, but the problem is that we also find $\mathcal{C}(1361) = \top$, thus $\mathcal{C}$ does not classify the objects in the intended way. To establish how to improve $\mathcal{C}$, we use algorithm 1 to obtain R_x, and from this set of necessary role assertions we obtain its leafs and the possible properties to specialize it:

$$R_x = \{involves(1336, A1)\}$$

$$Leafs_x = \{1336, A1\}$$

$$Ext_{x,\mathcal{C}} = \{hasNarrative(1336, \text{"...GPS..."}),$$
$$usesNav(A1, GPS), repProb(A1, Mf)\}$$

It is out of the scope of this paper how to construct a concept expression using the identified properties, nevertheless we provide some examples to illustrate the approach. If we first consider $hasNarrative$ to specialize $\mathcal{C}$, we can add a restriction in the following way:

$$\mathcal{C}' \equiv \exists involves.\top \wedge \exists hasNarrative.\{\text{"...GPS..."}\}$$

The concept $\mathcal{C}'$ expects that any report mentioning GPS in its narrative is indeed a "GPS related report". But even though report 1361 mentions GPS, it is not classified as "GPS related reports" by the experts (set Pos). Thus, we learn that $hasNarrative$ is not the property that allows us to distinguish them. We proceed now with the property $usesNav$. We can specialize $\mathcal{C}$ by:

$$\mathcal{C}'' \equiv \exists Involves.\exists usesNav.\top$$

Then $\mathcal{C}''$ correctly classifies Pos and Neg (since $\mathcal{C}''(1361) = \bot$). We learn that $usesNav$ is the most relevant property to specialize $\mathcal{C}$ that allows us to make the desired distinction, and that the specialization should be made in the position of the leaf $A1$ in the graph. Finally, assume we want to obtain a more interesting concept expression, representing those "Aviation Issues that involve a **problem** with GPS devices". The sets of positive and negative samples become:

$$Pos = \{1336, 1347\} \ , \ Neg = \{1359, 1361\}$$

Considering again object 1336 and $\mathcal{C}''$ as part of the input, the graph representation of its necessary properties is:

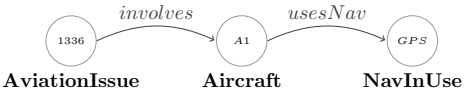

Figure 4: The graph representation of object x, where only the necessary properties for $\mathcal{C}''$ to capture it are shown.

Where:

$$Leafs_{x,\mathcal{C}''} = \{1336, A1\}$$

$$Ext_{x,\mathcal{C}''} = \{hasNarrative, repProb\}$$

If we select $repProb$ we can construct a concept expression of the form:

$$\mathcal{C}''' \equiv \exists Involves.(\exists usesNav.\top \sqcap \exists repProb.\{Mf\})$$

We can see that $\mathcal{C}'''$ properly distinguishes between Pos and Neg, and we learn that the most important property to make this distinction w.r.t. $\mathcal{C}''$ is $repProb$, where the most relevant objects (leafs) and their properties provide the key to construct such expressions.

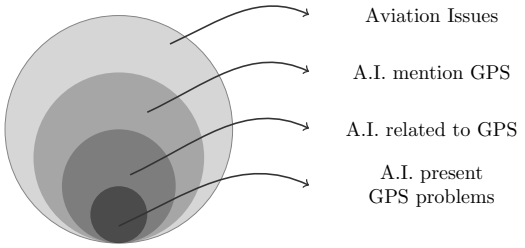

Figure 5: Specialization of concept "Aviation Issue".

5 Conclusions and further works

We consider that the most simple case (without a T-Box) is the most appropriate way to introduce our work and show how it can be used to obtain those properties relevant for a concept to capture an object. This information can be used to guide the refinement process or generalizing a concept, since we know exactly up to which point the properties of the object are taken into account by the concept expression in the input. The approach can also be useful for optimizing concept learning techniques, given that the scenario is restricted to our specifications. A constructive way for obtaining the refined concept can be given, which is dependent of the DL-family chosen for the ontology. Finally, we will study the method for generating rich features for action prediction in the avionics maintenance domain.

References

[Baader2003] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[Divroodi and Nguyen2015] Ali Rezaei Divroodi and Linh Anh Nguyen. On bisimulations for description logics. *Information Sciences*, 295:465–493, 2015.

[Inokuchi *et al.*2000] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '00, pages 13–23, London, UK, UK, 2000. Springer-Verlag.

[Lehmann2009] Jens Lehmann. Dl-learner: learning concepts in description logics. *Journal of Machine Learning Research*, 10(Nov):2639–2642, 2009.

[Motoda2007] Hiroshi Motoda. *Pattern Discovery from Graph-Structured Data - A Data Mining Perspective*, pages 12–22. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[Nguyen and Szałas2013] Linh Anh Nguyen and Andrzej Szałas. Logic-based roughification. *Rough Sets and Intelligent Systems-Professor Zdzisław Pawlak in Memoriam*, pages 517–543, 2013.

[Palacios *et al.*2016] Luis Palacios, Galle Lortal, Claire Laudy, Christian Sannino, Ludovic Simon, Giuseppe Fusco, Yue Ma, and Chantal Reynaud. Avionics maintenance ontology building for failure diagnosis support. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016)*, pages 204–209, 2016.

[Tran *et al.*2012] Thanh-Luong Tran, Quang-Thuy Ha, Linh Anh Nguyen, Hung Son Nguyen, Andrzej Szalas, et al. Concept learning for description logic-based information systems. In *Knowledge and Systems Engineering (KSE), 2012 Fourth International Conference on*, pages 65–73. IEEE, 2012.

[Yan and Han2003] Xifeng Yan and Jiawei Han. Closegraph: Mining closed frequent graph patterns. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 286–295, New York, NY, USA, 2003. ACM.

[Yoshida *et al.*1994] Kenichi Yoshida, Hiroshi Motoda, and Nitin Indurkhya. Graph-based induction as a unified learning framework. *Applied Intelligence*, 4(3):297–316, 1994.