Knowledge Based Situation Discovery for Avionics Maintenance
Luis Palacios Medinacelli, Yue Ma, Chantal Reynaud, Gaëlle Lortal

To cite this version:


HAL Id: hal-02930192
https://hal.archives-ouvertes.fr/hal-02930192
Submitted on 4 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT

For knowledge intensive domains, such as Avionics Maintenance, applying automated analysis comes with a major challenge: formalizing complex domain knowledge and conceiving suitable automated algorithms for real world requirements. In this paper, we propose a study on knowledge discovery to assist avionics maintenance via identifying meaningful Description Logic based complex concepts, called situation discovery, that corresponds to crucial scenarios during device repair.

We propose an approach to automatic learning of relevant situations hidden in an ontology, in an unsupervised way. Distinct from ontology based concept learning, where a set of instances is given as positive examples of a target concept, the challenge of learning hidden situations consists in discovering significant situations from exponentially many unknown situations. In this paper we formalize the problem and study some related complexity results as well as the algorithms to solve the problem, together with its application to Avionics Maintenance. The approach has been integrated into an enterprise system and achieves the state-of-the-art result in this application.

1 INTRODUCTION

Complex real world processes generate large amounts of heterogeneous data, from heterogeneous sources and multiple actors and locations. To exploit, verify, manage, access and share the information, AI and machine learning techniques have got an increased attention in the last decade. Part of the ramifications of this interest, is the necessity to leverage from existing data and information that the organizations already posses, and add value to it by making analysis and structuring the available data, obtaining information, and generating knowledge out of it.

Consider the avionics maintenance domain, where historical repair data can be a valuable information source to assist technicians when repairing a new failure. In avionics, a failure denotes the loss of the ability of a device to meet the performance specifications that it was intended to meet. A failure scenario in this context, represents all sufficient and necessary information that is strongly related to the failure mechanism [10]. Knowing the explicit description of the scenarios allows us to better understand the failure, to predict the behavior of the equipment, and to repair it. Formalizing such a knowledge intensive application requires expertise in the domain as well as in knowledge engineering. Moreover, even having both skills, the knowledge acquisition bottleneck [15, 22, 24] poses a major challenge: the knowledge might not be specific enough, that is, information could be hidden in the ontology that is not directly accessible via deduction.

As possible solutions to solve this task, the field of ontology learning studies techniques that aim to the automatic or semi-automatic construction of ontologies. Some of these approaches [3, 5, 8, 13, 14, 17, 18, 20, 23] apply machine learning notions to symbolic settings in a hybrid fashion, taking advantage of the two fields to offer a system with both characteristics: clear and well defined knowledge/semantics combined with automatic learning.

In this paper, we propose a novel way to discover (explainable) knowledge that can be used to enrich an existing knowledge base, and the newly acquired knowledge allows us to access to interesting subsets of elements of a domain that share certain common properties. Note that different from the previous work, these subsets of elements are unknown in advance. Once these sets of individuals are given a name (concept), they are made available for further references and the concept that describes them serves not only as an explanation on why the instances can be put together, but on why they can be separated from the rest.

As the explanation language, we adopt Description Logic (DL) [2], which complies with the standards of OWL2, has a rich expressivity, and enables the machines not only to understand its contents, but to automatically draw inferences over the represented knowledge. The main contributions in this paper are the following:

1. In this paper, we are interested in the ability to determine when a set of individuals can be perfectly distinguished from the rest. Each such set, for which we can find a proper definition in DL that covers exactly the set elements and no others, is called a situation in our work. A concept description gives a detailed characterization of the set of its instances, thus serving as an explanation for the set. We formally define this problem in DL and study the complexities of the related problems.

2. We propose a first algorithm to discover situations that can be then used for grouping instances and generating their concept descriptions. The model itself (ontology) is domain specific, and the learning algorithm is of general use.

3. We discuss the application of the proposed situation discovery technique to solve a repair suggestion scenario in Avionics Maintenance, highlight the main features of the implemented prototype.
and show some analytic results as well as some experiments on real world data.

The paper is structured as follows: in Section 2, we briefly overview the necessary notions. In Section 3, we formally define our problem and study its properties. In Section 4, we present an algorithm for the situation discovery problem. We discuss its application for Avionics maintenance in Section 5 and evaluate the approach in Section 6. Related work is discussed in Section 7 before the conclusion given in Section 8.

2 PRELIMINARIES

We consider the lightweight Description Logic \( \mathcal{ELO} \) [1, 2], whose concept descriptions are built from a set of concept names \( \mathcal{C} \) and a set of role names \( \mathcal{R} \) using the constructors \( \top, \bot, \cap, \cup, \exists \). The semantics of \( \mathcal{ELO} \) is defined using interpretations \( I = (\mathcal{A}_I, \mathcal{I}) \) consisting of a non-empty domain \( \mathcal{A}_I \) and an interpretation function \( \cdot^I \) mapping role names to binary relations on \( \mathcal{A}_I \) and concept descriptions to subsets of \( \mathcal{A}_I \) according to Table 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>individual name</td>
<td>a</td>
<td>( a^I \in \mathcal{A}_I )</td>
</tr>
<tr>
<td>concept name</td>
<td>A</td>
<td>( A^I \subseteq \mathcal{A}_I )</td>
</tr>
<tr>
<td>nominal</td>
<td>(o)</td>
<td>( {o^I} \subseteq \mathcal{A}_I )</td>
</tr>
<tr>
<td>role name</td>
<td>r</td>
<td>( r^I \subseteq \mathcal{A}_I \times \mathcal{A}_I )</td>
</tr>
<tr>
<td>top concept</td>
<td>T</td>
<td>( T^I = \mathcal{A}_I )</td>
</tr>
<tr>
<td>conjunction</td>
<td>C \cap D</td>
<td>( (C \cap D)^I = C^I \cap D^I )</td>
</tr>
<tr>
<td>existential restriction</td>
<td>( \exists r.C )</td>
<td>( (\exists r.C)^I = {x \mid \exists y: (x, y) \in r^I } )</td>
</tr>
<tr>
<td>full definition</td>
<td>A \equiv C</td>
<td>( A^I = C^I )</td>
</tr>
</tbody>
</table>

As axioms we allow full definitions and individual assertions. Full definitions are statements of the form \( A \equiv C \). Primitive definitions are statements of the form \( A \subseteq C \) where \( A \) is a concept name and \( C \) is a concept description. Individual assertions are statements of the form \( A(a) \) or \( r(a, b) \) where \( A \) is a concept name and \( a, b \) are individuals in \( \mathcal{A}_I \). A TBox \( T \) is a set of definitions of these two types and an ABox \( A \) is a set of individual assertions. We say that the interpretation \( I \) is a model of \( T \) (resp. \( A \)) if \( A^I = C^I \) (resp. \( c^I \in A^I \) or \( (a^I, b^I) \in r^I \)) holds for every full definition \( A \equiv C \) (primitive definition \( A \subseteq C \), respectively) from \( T \) (resp. assertions \( A(a), r(a, b) \) from \( A \)). An ontology is composed of a TBox and an ABox. A concept description \( C \) is said to be subsumed by the concept \( D \) with respect to an ontology \( O = (T, A) \) (denoted by \( O \models C \subseteq D \)) if \( C^I \subseteq D^I \) holds for all models \( I \) of \( T \). An individual assertion \( A(a) \) (resp. \( r(a, b) \)) is implied by an ontology, written \( O \models A(a) \) (resp. \( O \models r(a, b) \)) if \( a^I \in A^I \) (resp. \( (a^I, b^I) \in r^I \)) for all models \( I \) of \( O \). It is well-known that subsumption reasoning in \( \mathcal{ELO} \) is tractable.

In this paper, we assume that ABoxes are acyclic. An ABox \( A \) is called acyclic iff there are no \( n \geq 1 \) and individuals \( a_0, a_1, \ldots, a_n \) and roles \( r_1, \ldots, r_n \) such that (1) \( a = a_0 \), (2) \( r_i(a_{i-1}, a_i) \in A \) for \( 1 \leq i \leq n \), (3) there is \( j, 0 \leq j < n \) such that \( a_j = a_n \).

3 SITUATION DISCOVERY: DEFINITIONS AND PROPERTIES

Given an ontology \( O \) we aim to find interesting subsets of its individuals, and for each one of these sets provide a description in Description Logic terms. Each such set is represented by a class of DL concepts. We call each one of these DL concepts a situation in \( O \) (defined next).

This section formally introduces the problem of finding situations in an ontology in the form of complex DL concept definitions. We start by introducing some terminology.

**Definition 1 (A representative concept).** Let \( \Delta \) be a set of all individuals in an ontology \( O \), and let \( X \subseteq \Delta \). For a concept \( C \), we say that \( X \) is represented by \( C \) (or \( C \) represents \( X \)) w.r.t. \( O \) and \( \Delta \), if:

- \( C(x) \) holds for all \( x \in X \), i.e. \( O \models C(x) \), and
- \( C(y) \) does not hold for any \( y \in \Delta \setminus X \), i.e. \( O \not\models C(y) \).

If there exists a concept \( C \) that represents the set \( X \), we say that \( X \) is representable.

**Example 1 (Representative Concept).** Consider the sets of individuals \( \Delta = \{f_1, f_2, f_3\}, X = \{f_1, f_2\} \), and the following ontology \( O = \{T, A\} \):

\[
T = \{C \equiv \exists r.T\} \\
A = \{A(f_1), B(f_2), E(f_3), r(f_1, f_3), r(f_2, f_3)\}
\]

To determine if \( C \) represents \( X \) we check the two following conditions:

1. \( O \models \{C(f_1), C(f_2)\} \)
2. \( O \not\models \{C(f_3)\} \)

Since (1) and (2) hold, \( X \) is represented by \( C \).

However, it is not true that every set of instances can always be represented, as illustrated in the following example.

**Example 2 (Example 1 contd.).** Consider the set \( X' = \{f_2, f_3\} \), there is no \( \mathcal{ELO} \) concept that can represent \( X' \).

Note that there are two sets of individuals which can always be represented, as shown by the following lemma.

**Lemma 1.** Given an ontology \( O \) and a set \( \Delta \) of individuals, we have

- \( T \) is a representative concept for \( \Delta \).
- \( \bot \) is a representative concept for \( \emptyset \).

A set of individuals that can be represented need to share some common properties merely among them, which are made explicit by the representative concept. By reading the concept definition, we get an explicit explanation of their common properties. In example 1, the individuals \( f_1 \) and \( f_2 \) share the property that they are connected to some individual via the role \( r \), whilst \( f_2 \) and \( f_3 \) do not have any property in common that can distinguish them from \( f_2 \). In fact the problem of when a set of individuals can be distinguished, is central to the approach, and is not trivial. The problem of separability has been formally studied, and it remains undecidable even for very simple DLs as \( \mathcal{ELO} \) [9].

The following proposition states that the intersection of two sets which can be represented, is representable as well.
We can see that \( X \) is one concept belonging to the class to characterize it. Therefore \( \mathcal{ELO} \) can be solved in \( \text{P-Time} \). Where \( \Delta \) can be represented by the concept \( B \) w.r.t \( O \) and \( \Delta \). And the concept \( A \cap B \) represents \( \emptyset \).

The following lemma tells us that any concept naturally represents a special set of individuals.

**Lemma 2.** Given a concept \( C \), an ontology \( O \) and the set \( \Delta \) of individuals in \( O \), \( C \) represents the set of individuals \( S = \{ x \in \Delta \mid O \models C(x) \} \).

**Example 3 (Example 1 contd.).** Concept \( A \) represents the set \( \{ f_1 \} \), \( B \) represents the set \( \{ f_2 \} \), and \( E \) represents the set \( \{ f_3 \} \). And the concept \( A \cap B \) represents \( \emptyset \).

Note that when a concept represents an empty set of individuals of an ontology, it means that this concept is irrelevant to characterize the properties of individuals from this ontology. Hence, from now on, we are only interested in the concepts that represent a non-empty set.

**Proposition 2 (Representability).** Given an \( \mathcal{ELO} \) ontology \( O \), a set \( \Delta \) of individuals, a concept \( C \) and a set \( X \subseteq \Delta \), the decision problem Representability:

\[
\text{Does } C \text{ represent } X \text{ w.r.t. } O?\]

**Proposition 3 (Representability).** Let \( O \) be an \( \mathcal{ELO} \) ontology and \( \Delta \) the set of individuals in \( O \). For a given set of individuals \( X \subseteq \Delta \), and an integer \( n > 0 \), the decision problem Representability:

\[
\text{Is there a concept } C \text{ with } |C| < n \text{ that represents } X \text{ w.r.t. } O?\]

**Proposition 4 (Representability).** Let \( O \) be an \( \mathcal{ELO} \) ontology and \( \Delta \) a set of individuals in \( O \). For \( X \subseteq \Delta \), the decision problem Representability:

\[
\text{Is there a concept } C \text{ with } |C| < n \text{ that represents } X \text{ w.r.t. } O?\]

**Proposition 4 (Representability).** Let \( O \) be an \( \mathcal{ELO} \) ontology and \( \Delta \) a set of individuals in \( O \). For \( X \subseteq \Delta \), the decision problem Representability:

\[
\text{Is there a concept } C \text{ with } |C| < n \text{ that represents } X \text{ w.r.t. } O?\]

**Definition 3 (Situation discovery problem).** Let \( O \) be an ontology and \( \Delta \) a set of individuals in \( O \). For \( X \subseteq \Delta \), the situation discovery problem is to compute the following set:

\[
\Xi_O(X) = \{ X_1, \ldots, X_n \mid X_i \subseteq X, \|X_i\|_O \neq \emptyset \}
\]

That is, to find all the subsets of \( X \) that are representable w.r.t. \( O \).

We also shorten \( \Xi_O(X) \) as \( \Xi(X) \) when the ontology \( O \) is clear from the context. Since each \( X_j \in \Xi_O(X) \) leads to a situation \( ||X_i||_O \), we will also call such \( X_i \) a situation by an abuse of terminology.

By Lemma 1, it is easy to see that \( \emptyset \) is representable by \( \bot \), i.e. \( \emptyset \in \Xi(X) \), leading to the following conclusion.

**Lemma 3.** Let \( O \) be an ontology and \( \Delta \) a set of individuals in \( O \). For \( X \subseteq \Delta \), \( \Xi(X) \neq \emptyset \).

Lemma 3 shows that we can obtain at least one situation with no computational cost. Henceforth, we omit this trivial situation in the rest of this paper.
Moreover, Example 2 shows that it happens that X is not representable, but $\Xi(X)$ contains subsets of individuals that are nevertheless representable, such as $X_1 = \{ f \}$ having a representative concept $E$.

**Definition 4.** Let $O$ be an $\mathcal{ELO}$ ontology and $\Delta$ a set of individuals in $O$. For a given set of individuals $X \subseteq \Delta$ and an integer $n > 0$, the decision problem $SD_n^\Delta$ is defined as follows:

*Does there exist a situation $C$ for some $X' \subseteq X$ in $O$ with $|C| \leq n$?*

If the answer to $SD_n^\Delta$ is positive, it means that there exists a nonempty subset $X' \subseteq X$ such that $C$ is a representative concept for $X'$ w.r.t. $O$ and $|C| \leq n$.

**Proposition 5.** $SD_n^\Delta$ is in $\text{ExpTime}$. Moreover, if $|X|$ and $n$ are bounded by a constant, $SD_n^\Delta$ is in $\text{P-Time}$.

The following conclusion shows that for a set of individuals $X$, the $\Xi(\cdot)$ operator satisfies monotonicity in the sense that (1) the set of concepts representing $X$ might decrease when the situation domain increases; (2) a concept $C$ that characterizes $X$ still characterizes some set of individuals when the situation domain increases. Nevertheless, the set of concepts that concept $C$ characterizes might no longer be $X$. In fact, it could be the case that $X$ is no longer representable.

**Proposition 6.** Let $O$ be an ontology and $\Delta$ be a set of individuals in $O$. Consider $\Delta_1 \subseteq \Delta$. Suppose that $X \in \Xi(\Delta_1)$ is represented by a concept $C$ w.r.t. $O$ and $\Delta_1$. Then the following conclusions hold:

1. $|X|_{\Delta_1} \leq |X|_{\Delta}$
2. $X$ is not necessarily representable w.r.t. $\Delta$.
3. The concept $C$ still represents some set of individuals $X'$, that is, $X' \in \Xi(\Delta)$.

## 4 COMPUTING SITUATIONS

In this section, we introduce an algorithm to compute situations for a set of instances $X$. The intuition is: we first define a refinement operator that can find the most specific concept from a general one (e.g. $\top$), called MSR, that represents the given instances. Once we can obtain the MSR for a set of individuals $X$, we know that any refinement of such MSR obtained by the operator will define a strict subset $X' \subset X$. This subset $X'$ is characterized by a concept refined from the MSR for $X$ (via a refinement operator described later in this section). Since all these individuals in $X'$ are instances of the obtained refinement, the set $X'$ defines a situation. By iterating this process over each subset found, and for each corresponding MSR, we obtain situations in $X$.

Let us start with the definition of the most specific representative.

**Definition 5 (Most Specific Representative MSR).** Given a set of individuals $X = \{ x_1, \ldots, x_n \}$ and the set of its representative concepts $||X|| = \{ S \mid S \text{ represents } X \}$, the Most Specific Representative of the set $X$, written $\text{MSR}_X$, is the concept $S_i \in ||X||$ such that:

$$\forall S_j \in ||X||, \text{ we find } S_i \subseteq S_j.$$

**Example 5 (Most Specific Representative).** As an example consider $\Delta = \{ x, y, z, z' \}$, the set $X = \{ x \}$, the A-Box:

$A_3 = \{ r(x, y), r(z, z'), A(y), B(y), C(z') \}$

and the concepts:

| $S_0$ | $\exists.\top$ | $\{ x, z \}$
| $S_1$ | $\exists.A$ | $\{ x \}$
| $S_2$ | $\exists.y$ | $\{ x \}$
| $S_3$ | $\exists.(A \land B \land \{ y \})$ | $\{ x \}$
| $S_4$ | $\exists.(A \land B \land \{ y \}) \land \exists.A$ | $\{ x \}$

We find that $S_0 \not\subseteq ||X||$ since $z \notin S_0$. In contrast, all other concepts do represent $X = \{ x \}$, thus we have $S_1, S_2, S_3, S_4 \in ||X||$ (note that the set $||X||$ can be infinite). The subsumption relation between these concepts is given by:

| $S_3 \sqsubseteq S_1, S_2, S_4$
| $S_4 \sqsubseteq S_1, S_2, S_3$
| $S_3 \equiv S_4$

Two of these concepts are equivalent, and more specific than the rest: $S_3$ and $S_4$. To select among equivalent concepts, we prefer shorter concepts. Since $|S_3| < |S_4|$, the most specific representative MSR$_X$ is $S_3$.

Next we define the notion of concept refinement operator.

**Definition 6.** Given an ontology $O$, a concept $C$, and an instance $x$, an operator $\alpha_x(\cdot)$ is called a concept refinement operator if $\alpha_x(C) = \{ C_1, \ldots, C_n \}$ and for each $C_j \in \alpha_x(C), O \models C_j(x)$ and $C_j \subseteq C$. We call $\alpha_x(C)$ a direct refinement operator if $|\text{Sub}(C_j)| = |\text{Sub}(C)| = 1$, where $\text{Sub}(C)$ is the set of subconcepts of a concept $C$.

Using a refinement operator we can traverse the space of concept expressions. Out of the concepts obtained through the operator, we can obtain the most specific one. We now assume that Get-MSR($X$) is the procedure to compute the MSR of a set of instances $X$, and $\alpha$ is an operator that can refine a concept to a direct refinement, Algorithm 1 uses these elements to specify the process to extract situations in a set $X$. That is the subsets of individuals in $X$ that can be represented by an $\mathcal{ELO}$ expression.

**Algorithm 1** $\text{SD}(X)$

1: **input:** ($C, O, X$)
2: $\Xi = \{ X \}$
3: $\text{ToRefine} = \{ X \}$
4: **while** $\text{ToRefine} \neq \emptyset$ **do**
5:   **for** $Y \in \text{ToRefine}$ **do**
6:     **for** $y \in Y$ **do**
7:       **for** $D \in \alpha_y(\text{Get-MSR($Y$)})$ **do**
8:         $\text{Inst}_D = \{ y \in Y \mid O \models D(y) \}$
9:       **end for**
10:     **end for**
11:   **end for**
12: **end while**
13: **return:** $\Xi$
ToRefine contains all those sets of individuals that need to be analyzed to search for situations (Line 3). For each such set \( Y \) (Line 5) we obtain its MSR computed by algorithm Get-MSR. Then, for each individual in every set \( Y \) (Line 6), the MSR(\( Y \)) is refined. In this fashion, we obtain the representable subsets of \( Y \). Intuitively, if there exists a sub-set of \( Y \) that can be represented, there exists a concept \( D \sqsubseteq MSR \). This concept can be found by applying \( \delta_2(MSR) \) for some \( y \in Y \). For every such refinement found, we obtain its instances and record them in \( InstD \) (Line 8). Because there exists a concept \( D \) for each one of these sets, they define a situation as well. And thus all the different subsets of \( Y \) are added to ToRefine (Line 9), to explore if further sub-situations can be found in the next iteration of the for loop (line 5). Since all subsets found this way are representable, we add all of them to \( \Xi \). This process is repeated for the MSR of every subset found this way, and refined with every instance. Finally, when no more subsets can be found. The refinements of every MSR will be empty (there no longer exists concepts that are more specific than those already found) and thus the while loop (Line 5) will stop. The output of Algorithm 1 are the situations in \( X \) as illustrated in the following proposition.

**Proposition 7.** Given an ontology \( O \), a concept \( C \) and the set of its instances \( X \). All elements in the output \( \Xi \) of Algorithm 1 are situations in \( X \).

## 5 APPLICATION DOMAIN AND PROTOTYPE

In this section we briefly describe the maintenance and diagnosis process for the Elevator and Aileron Computer (ELAC) equipment, for which we want to provide support through suggested repair actions made to the technician. We start by presenting the data sources and the ontology that captures the identified knowledge and the functions the prototype should provide.

The ontology has been designed considering the two main data sources: the .AR files and the corrective actions, presented next.

### 5.1 The Data Sources and the Ontology TAMO

The information about the diagnosis process has two main sources: the .AR files containing the results of the tests made to each equipment, and the corrective actions associated to each equipment. This information has been modeled in the Thales Avionics Maintenance Ontology (TAMO) [19], aided by the Thales Avionics experts.

In the diagnosis process the technician tests the ELAC in a special unit called a Test-Bench. This unit checks exhaustively all the ELAC functions, and the output of this process is an .AR file (All Results).

The .AR files are the main source of information for the ontology. They are presented in plain text format and contain up to thousands of lines. Each line of an .AR file represents an individual test on a specific function of the ELAC, with the sanction GO or NOGO which indicates if the test was passed. Thus, an .AR file is a set of individual test results (thus the name All Results file). Figure 1 shows an extract of an .AR file, and the main sections it contains. Note that the structure of each .AR file can be different according to different tests performed.

On the other hand we have the corrective actions. In avionics maintenance there are multiple types of maintenance actions (repair, cleaning, preventive maintenance tasks, upgrades, etc). From these types of maintenance actions, we have selected the replacements of the components in the ELAC as the actions to be modeled, since these components have a direct influence in the results of the .AR files. Each ELAC is a computer composed of several boards (six plus two interface boards), and each board has hundreds of components of different types that can be replaced.

A support tool for avionics maintenance should provide two main functions: consult the ontology and integrate the users feedback. Note that we call TAMO plus all the situations discovered as discussed above a knowledge base (KB).

### 5.2 Consult the KB

When the KB is used to obtain suggestions for a new .AR file \( f_x \), we first determine the most specific situation in \( O \) for \( f_x \), then the files already in the KB that belong to this situation are retrieved, and finally the actions associated to each such files are extracted. These actions represent the suggestions to solve the failure detected by \( f_x \). These three steps are detailed in the following.

Let \( \Delta = \{ f_1, \ldots, f_n \} \) be the set of all .AR files in \( O \), and assume \( O \) has been enriched (trained) using the refinement process in Section 3, where all situations in \( O \) have been discovered. Given a new file \( f_x \notin \Delta \), our task is to find the set \( AS_{f_x} \) of actions that can be associated to \( f_x \).

**Step 1: Obtain the most specific situation for an .AR file**

The file \( f_x \) might belong to more than one situation in \( O \), thus we select the most specific one, since it provides the most detailed description for the failure.

Consider the following example:

Given the set of individuals \( \Delta = \{ f_1, f_2, f_3 \} \) the following DL concept definitions are examples of concept refinements:

- \( S_0 \) is defined as \( \exists \text{hasTestLine}.(\exists \text{hasTestResult}.\{\text{NOGO}\}) \)
- \( S_1 \) is defined as \( \exists \text{hasTestLine}.(\exists \text{hasTestCode}.\{1234\}) \)
- \( S_2 \) is defined as \( \exists \text{hasTestLine}.(\exists \text{hasTestCode}.\{1234\} \land \exists \text{hasTestResult}.\{\text{NOGO}\}) \)
- \( S_3 \) is defined as \( \exists \text{hasTestLine}.(\exists \text{hasTestCode}.\{1234\} \land \exists \text{hasTestResult}.\{\text{NOGO}\} \land \exists \text{hasTestPart}.\{\text{Part1}\}) \)

Let \( f_x \equiv f_3 \) (meaning that both files have exactly the same test results). Then the situations for \( f_x \) are \( S_0 \) and \( S_3 \) (since \( f_x \in S_0, S_3 \)).

Whilst, the most specific situation of \( f_x \) in symbols \( S_{f_x} \) is \( S_3 \).
6 EVALUATION

In this section we present the evaluation of the approach through the implementation of the prototype. Section 6.1 provides a comparison with DL-Learner, showing that the proposed approach achieves a state-of-the-art result on the ELAC maintenance task. Then Section 6.2 further evaluates the relevance and the number of suggestions proposed by the approach. Finally, Section 6.3 evaluates the evolution of the knowledge base. Our hypothesis here is that the more fine-grained the knowledge base, the more specific situations we can find and therefore we can minimize the number of suggested corrective actions. Full details on the experiments and results can be found in [16].

6.1 Results TAMO vs DL-Learner

For the sake of clarity, in this section we refer to our approach as TAMO, to differentiate our results from those obtained with DL-Learner\(^2\). For this experiment we have selected a random subset of 25 files out of the total files (150) available from the ELAC repair workshop. For each file, we have obtained corrective actions assigned, both with DL-learner and TAMO trained on other files.

Since our goal is to minimize the number of suggested actions, we want to evaluate how many actions are proposed by each tool to each file. In Figure 3 we show the number of actions returned using the concepts learned by DL-Learner and the concepts learned by TAMO, for each of the 25 selected files. Each file may belong to one or more DL-Learner concepts, and therefore it will be associated to all the actions those concepts represent. The concepts that are too general, capture most/all individuals. From the figure we can see that most of the concepts from DL-Learner will associate around 20 actions to each file, whereas in our case, most of the files are associated to 3 or less actions. There are also a few cases where we associate more than 30 actions to a file, this is mostly because those files were not related to the set of files we used, to create our classes (learning phase).

The low precision of the DL-Learner concepts, can be explained by the fact that the tests that are solved by the same action might be not only very different from each other, but they might not even share anything in common among them. Given that the underlying language is ELO, no disjunction is allowed (which would help to capture files that are different by a single concept), there is no single representation for all those tests in ELO and DL-learner returned short but very imprecise concepts.

6.2 Relevance and Specificity of the Suggestions

In this second experiment, we evaluate the relevance of the returned suggestions and the specificity of the discovered signatures. For both evaluations we use k-fold cross validation with a size of \(k = 3\), which represents a third of the samples. This means that the full set of 150 samples (.AR files) is divided into three partitions \(p_1, p_2, p_3\), each one containing 50 .AR files. Each partition is used once as the validation set, while the other two are used for training the knowledge base.

\(^2\)https://dl-learner.org
The third experiment has the objective of evaluating whether there exist valuable suggestions even under these circumstances. KBs of different sizes are constructed: $KB_{25}$, $KB_{50}$, and $KB_{100}$, and the improvement is measured as a function of the knowledge base size. From these, we would expect that those found by smaller training sets ($KB_{25}$) are more general than those found by larger training sets ($KB_{50}, KB_{100}$).

If we focus on the individual actions in Figure 5, we can see that $KB_{25}$ has been able to provide relevant suggestions for the 50 files using 5 situations (the other 5 situations shown were used, but the answers were irrelevant), whereas $KB_{100}$ has used 9 relevant situations to classify the same 50 files. Even though the results from $KB_{25}$ are very good for its size, it has to be noticed that from the 10 files that are given a partial relevant suggestion, for half of them (5 files) the situations are too general and provide all possible suggestions available in $KB_{25}$. This is why “so many” files are provided with a “correct” partial suggestion. This can be partially seen by the precision of the situations, where a low precision means more undesired files belong to the situations, increasing the number of false positives. In the case of $KB_{100}$ we can see that the quality of the situations is increased, since more situations have higher precision, and more situations are used to classify the 50 files. Showing that a richer way to distinguish among files is obtained as more information is presented to the KB.

7 RELATED WORK

Model Based Diagnosis. In equipment diagnosis, the manifestation of a failure is put down to the bad interaction between some of its components. Identifying the components involved, provides the signature of the failure. In model-based diagnosis this is known as a diagnosis [6] and the model aims to predict the intended behaviour of the modeled system. In our case we do not count with such a model since it is sometimes unavailable. Instead, we are given tests that report the status of the functions in the equipment, and corrective actions made by the maintenance technicians.

Ontologies in Approaches for Maintenance. There exist several works on the applicability, advantages and considerations of using ontologies to model maintenance, and support the overall process. The main objective in these works [4, 7, 11, 12, 19, 21] is to provide...
We have formalized the problem of finding situations, provided we have presented an approach to discover interesting subsets of individual actions. On top the results for $KB_{25}$ and on the bottom, the results for $KB_{100}$. The x-axis shows the situations.

As future work, the results and properties of the operator can be specified to support more expressive DLs (e.g. negation $\neg$ and conjunction $\land$), since they can greatly increase the applicability of the approach. Regarding the specific application domain, we have considered only a specific equipment (ELAC) and a type of maintenance action (replacement). The ontology and the KB can be extended so that additional equipment and additional maintenance tasks can be considered. In this sense, in general terms, the approach can be applied to any maintenance process with similar characteristics. Finally, some bottlenecks and limitations have been evidenced thanks to the implementation and evaluation of the prototype. Improvements in parallel processing and partitions of the ontology can greatly benefit industrial implementations.

**Acknowledgement.** This work is partially supported by the ANR project GOASQ under the number ANR-15-CE23-0022.

**REFERENCES**


**Figure 5:** Experimentation on the evolution of the KB. The figures show the precision, recall and f-measure for the individual actions. On top the results for $KB_{25}$ and on the bottom, the results for $KB_{100}$. The x-axis shows the situations.