# Knowledge Based Situation Discovery for Avionics Maintenance

Luis Palacios Medinacelli, Yue Ma, Chantal Reynaud, Gaëlle Lortal

# Knowledge Based Situation Discovery for Avionics Maintenance

Luis Palacios Medinacelli[1,2], Yue Ma[1],
Chantal Reynaud[1]
[1]LRI-CNRS, Unvi. Paris-Sud, Université Paris-Scalay
91405 Orsay cedex, France

Gaëlle Lortal[2]
[2]Thales TRT
91120 Palaiseau, France

## 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.

**ACM Reference Format:**
Luis Palacios Medinacelli[1,2], Yue Ma[1], Chantal Reynaud[1] and Gaëlle Lortal[2]. 2019. Knowledge Based Situation Discovery for Avionics Maintenance. In *Proceedings of the 10th International Conference on Knowledge Capture (K-CAP '19), November 19–21, 2019, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3360901.3364430

## 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


*K-CAP '19, November 19–21, 2019, Marina Del Rey, CA, USA*

ACM ISBN 978-1-4503-7008-0/19/11…$15.00
https://doi.org/10.1145/3360901.3364430

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 $N_C$ and a set of role names $N_R$ using the constructors *top concept* $\top$, *conjunction* $\sqcap$, and *existential restrictions* $\exists$. The semantics of $\mathcal{ELO}$ is defined using interpretations $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta_{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ mapping role names to binary relations on $\Delta_{\mathcal{I}}$ and concept descriptions to subsets of $\Delta_{\mathcal{I}}$ according to Table 1.

**Table 1: Syntax and Semantics of $\mathcal{ELO}$**

| Name | Syntax | Semantics |
|---|---|---|
| individual name | $a$ | $a^I \in \Delta^I$ |
| concept name | $A$ | $A^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}}$ |
| nominal | $\{o\}$ | $\{o^{\mathcal{I}}\} \subseteq \Delta_{\mathcal{I}}$ |
| role name | $r$ | $r^{\mathcal{I}} \subseteq \Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$ |
| top concept | $\top$ | $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| existential restriction | $\exists r.C$ | $(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y\colon (x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$ |
| full definition | $A \equiv C$ | $A^{\mathcal{I}} = C^{\mathcal{I}}$ |

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 $A \sqsubseteq C$ 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 $\Delta_{\mathcal{I}}$. A TBox $\mathcal{T}$ is a set of definitions of these two types and an ABox $\mathcal{A}$ is a set of individual assertions. We say that the interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ (resp. $\mathcal{A}$) if $A^{\mathcal{I}} = C^{\mathcal{I}}$ (resp. $c^I \in A^{\mathcal{I}}$ or $(a^I, b^I) \in r^I$) holds for every full definition $A \equiv C$ (primitive definition $A \sqsubseteq C$. respectively) from $\mathcal{T}$ (resp. assertions $A(a), r(a,b)$ from $\mathcal{A}$). 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 $O = (\mathcal{T}, \mathcal{A})$ (denoted by $O \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$. 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 $\mathcal{I}$ of $O$. It is well-known that subsumption reasoning in $\mathcal{EL}$is tractable.

In this paper, we assume that ABoxes are acyclic. An ABox $\mathcal{A}$ is called acyclic iff there are no $n \geq 1$ and individuals $a_0, a_1, \cdots, a_n$ and roles $r_1, \cdots, r_n$ such that (1) $a = a_0$, (2) $r_i(a_{i-1}, a_i) \in \mathcal{A}$ 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:*

$$C(x) \text{ holds for all } x \in X, \text{ i.e. } O \models C(x), \text{ and}$$

$$C(y) \text{ does not hold for any } y \in \Delta \setminus X, \text{ i.e., } O \not\models C(y).$$

*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 $O = \langle T, A \rangle$:*

$$\begin{aligned} T &= \{C \equiv \exists r.\top\} \\ A &= \{A(f_1), B(f_2), E(f_3), r(f1, f3), r(f_2, f_3)\} \end{aligned}$$

*To determine if $C$ represents $X$ we check the two following conditions:*

$$\begin{array}{ll} (1) & O \models \{C(f_1), C(f_2)\} \\ (2) & O \not\models \{C(f_3)\} \end{array}$$

*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*

- *$\top$ is a representative concept for $\Delta$.*
- *$\perp$ 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 distinct them from $f_1$. 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.

Proposition 1. *Given an ontology $O$, the set $\Delta$ of individuals in $O$, $X \subseteq \Delta$ and $X' \subseteq \Delta$. If $X$ and $X'$ are representable, then $X \cap X'$ is representable in $\mathcal{ELO}$.*

However, the above conclusion is no longer true for set union or set complement. Consider Example 1, let $X_1 = \{f_2\}, X_2 = \{f_3\}$. We can see that $X_1$ is represented by the concept $B$ w.r.t. $O$ and $\Delta$, and $X_2$ is represented by the concept $E$ w.r.t. $O$ and $\Delta$. However, we have seen that $X_1 \cup X_2$ is not representable.

The following lemma tells 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 represents the set $\{f_3\}$. And the concept $A \sqcap B$ 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:*

*Does $C$ represent $X$ w.r.t. $O$?*

*can be solved in **P-Time**.*

Proposition 3 (Representability$_n$). *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$_n$:*

*Is there a concept $C$ with $|C| < n$ that represents $X$ w.r.t. $O$?*

*can be solved in **ExpTime**.*

By the proof of Proposition 3, if $n$ is bounded by a constant, then the problem Representability$_n$ is solvable in **P-Time**.

Note that a set of instances $X$ can be represented by more than one concept, and that if no restrictions are imposed, their number might even be infinite (Example 4). To avoid dealing with an infinite number of concepts, we use the notion of equivalence classes. The concepts representing $X$ are equivalent in the sense of their instances, and thus they define a class of equivalent concepts. Each one of these classes is called a *situation* in $O$ and it suffices to provide one concept belonging to the class to characterize it. Therefore to define a situation, our problem is reduced to finding a single representative for the situation instead of finding all of the concepts that comprise it.

Definition 2 (Situation in $O$). *Given an ontology $O$, a set $\Delta$ of individuals in $O$, and a set $X \subseteq \Delta$, a* situation *for $X$ in $O$ is the set:*

$$||X||_{\Delta}^{O} = \{C \mid C \text{ represents } X \text{ w.r.t. } O \text{ and } \Delta\}.$$

*Where $\Delta$ is called the domain of the situation.*

When the ontology $O$ and the set of individuals $\Delta$ are clear from the context, $||X||_{\Delta}^{O}$ is shortened as $||X||$. By an abuse of notation, we also refer to an element from $||X||$ as a situation.

Intuitively, a situation in $O$ explicitly characterizes, via concept descriptions, a given set of individuals in the ontology.

Additionally, among the concepts that belong to a situation there might be some of them that are not equivalent in the classical sense (i.e. w.r.t. subsumption), as illustrated by Example 4.

Example 4. *Consider again example 1 and the following T-Box:*

$$\begin{aligned} T_2 = \{ \quad & C_1 \equiv \exists r.\top, \\ & C_2 \equiv \exists r.\{f_3\}, \\ & C_3 \equiv \exists r.\top \sqcap \exists r.\{f_3\}\} \end{aligned}$$

*Then, $C_1, C_2, C_3$ all represent $X = \{f_1, f_2\}$. Therefore $\{C_1, C_2, C_3\} \in ||X||$. Note that a concept of the form $C_4 \equiv \exists r.\top \sqcap \exists r.\top \sqcap \cdots \sqcap \exists r.\top$ would also represent $X$, which shows that the set $||X||$ could be infinite, even with very simple DLs. Furthermore note that $C_1 \not\equiv C_2 \not\equiv C_3$, but $\{C_1, C_2, C_3\} \in ||X||$.*

Indeed, the notion of situation is more general than the notion of standard equivalence class as stated by the following proposition.

Proposition 4. *Given an ontology $O$ and the set $\Delta$ of individuals in $O$, we assume $O \models A \equiv B$. Then we have that $A \in ||X||$ if and only if $B \in ||X||$ for any $X \subseteq \Delta$.*

Note that not every subset of individuals can lead to a (non-empty) situation. For instance, the set $X' = \{f_2, f_3\}$ from Example 1 can not be represented under $\mathcal{ELO}$, therefore the set $||X'||$ is empty.

Assume we are given a set of individuals $X \subseteq \Delta$, and we need to determine whether a situation representing $X$ exists. If the response is negative, to ensure a complete and sound answer, we would require to access all possible situations in $O$, and for each of them, test if it represents $X$. We could also be interested in the possible ways we could use DLs to discriminate between the individuals in $X$, to extract a particular set. It is then natural to ask for the subsets of $X$ for which a DL representation exists, thus discovering the situations in $O$.

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 $X_i \in \Xi_O(X)$ 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 $\Delta$ be a set of individuals in $O$. For any $X \subseteq \Delta$, $\Xi(X) \neq \emptyset$.*

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 $X_1 = \{f_3\}$ 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 $SD_n$ 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 $\mathrm{SD}_n$ 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. *$SD_n$ is in **ExpTime**. Moreover, if $|X|$ and $n$ are bounded by a constant, $SD_n$ is in **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 individuals 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) $||X||_\Delta \subseteq ||X||_{\Delta_1}$
(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 $MSR_X$, is the concept $S_i \in ||X||$ such that:*

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

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

$$\mathcal{A}_3 = \{r(x, y), r(z, z'), A(y), B(y), C(z')\}$$

*and the concepts:*

$$\begin{array}{lll} S_0 \equiv & \exists r.\top & = \{x, z\} \\ S_1 \equiv & \exists r.A & = \{x\} \\ S_2 \equiv & \exists r.\{y\} & = \{x\} \\ S_3 \equiv & \exists r.(A \sqcap B \sqcap \{y\}) & = \{x\} \\ S_4 \equiv & \exists r.(A \sqcap B \sqcap \{y\}) \sqcap \exists r.A & = \{x\} \end{array}$$

*We find that $S_0 \notin ||X||$ since $z \in S_0$. 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:*

$$\begin{array}{ll} S_3 \sqsubseteq & S_1, S_2, S_4 \\ S_4 \sqsubseteq & S_1, S_2, S_3 \\ S_3 \equiv & S_4 \end{array}$$

*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 $\alpha_x(C) = \{C_1, \cdots, C_n\}$ and for each $C_i \in \alpha_x(C)$, $O \models C_i(x)$ and $C_i \sqsubseteq C$. We call a concept refinement operator a* direct *refinement operator if $|Sub(C_i)| - |Sub(C)| = 1$, where $Sub(C)$ 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** SD($X$)

```
1:  input: (C, O, X)
2:  Ξ = {X}
3:  ToRefine = {X}
4:  while ToRefine ≠ ∅ do
5:      for Y ∈ ToRefine do
6:          for y ∈ Y do
7:              for D ∈ α_y(Get-MSR(Y)) do
8:                  Inst_D = {y ∈ Y | O ⊨ D(y)}
9:                  Add Inst_D to ToRefine
10:             end for
11:         end for
12:         remove Y from ToRefine
13:     end for
14:     Add ToRefine to Ξ
15: end while
16: return: Ξ
```

In Algorithm 1 the set of all situations $\Xi$ is initialized with $X$ (Line 2), since $X$ always leads to a situation (Lemma 1). The set

$ToRefine$ 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 [1]. 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 $D \sqsubset MSR$. This concept can be found by applying $\alpha_y(MSR)$ for some $y \in Y$. For every such refinement found, we obtain its instances and record them in $Inst_D$ (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.

[1]This is achieved using a *refinement operator*. The algorithm, proofs and details on the refinement operator can be found in [16].

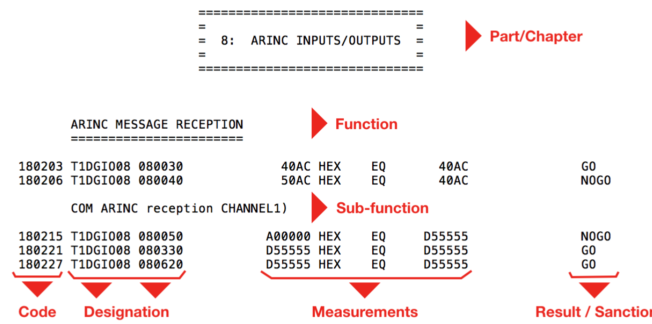


**Figure 1: An extraction of the structure of an .AR file.**

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 $A_{S_{f_x}}$ 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:*

$$\begin{aligned}
S_0 &\equiv \exists hasTestLine.(\exists hasTestResult.\{NOGO\}) \\
S_1 &\equiv \exists hasTestLine.(\exists hasTestCode.\{1234\}) \\
S_2 &\equiv \exists hasTestLine.(\exists hasTestCode.\{1234\} \\
&\quad \sqcap \exists hasTestResult.\{NOGO\}) \\
S_3 &\equiv \exists hasTestLine.(\exists hasTestCode.\{1234\} \\
&\quad \sqcap \exists hasTestResult.\{NOGO\} \sqcap \exists hasTestPart.\{Part1\})
\end{aligned}$$

*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 $S_3$.*

Example 6.

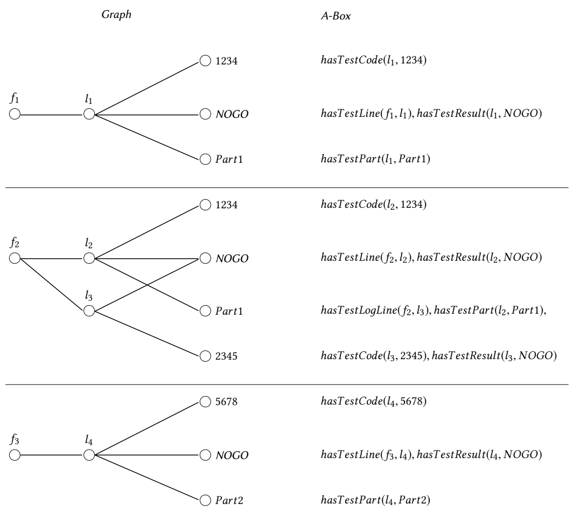


**Figure 2: Graph representation and corresponding A-Box for files $f_1, f_2, f_3$.**

**Step 2: Obtain all the files in KB that belong to the selected situation** Once we have selected the situation for $f_x$ the second step is to obtain all files that belong to the situation. That is the files in $O$ that are its instances.

In our example we have: $S_{f_x} = S_3 = \{f_2, f_3\}$.

**Step 3: Obtain the actions associated to each file in the situation** Once we obtain the .AR files that belong to the most specific situation $S_{f_x} = \{f_2, f_3\}$ for $f_x$, we use a set $FA$ which associates files-actions to obtain its suggestions. Consider the set: $FA = \{(f_1, a_1), (f_2, a_2), (f_3, a_3)\}$. We find that $S_{f_x} = \{f_2, f_3\}$, then the corresponding actions are $A_{S_{f_x}} = \{a_2, a_3\}$.

Finally, all those corrective actions associated to the situation $S_{f_x}$ form the suggestions of file $f_x$: $Suggestions_{f_x} = A_{S_{f_x}}$.

## 5.3 A more Fine-Grained KB through Feedback

After a file is consulted and the corresponding suggestions are proposed to the technician, an investigations & repair phase follows. During this process, the technician resolves the failure detected by the .AR file. Once the true corrective action is known, the feedback can be obtained. This process has two main tasks:

(1) It aims to integrate and validate the feedback from the technician by recording the true corrective action and associate it to the corresponding .AR file. In this way it is made available for future consultations.
(2) The .AR file in the feedback might contain information not seen before in the training stage (additional properties). The second task of the feedback is to analyze this new .AR file, in search for new situation descriptions and, if found, integrate these descriptions in the knowledge base.

As a result of this process, we obtain a more fine grained ontology and a larger set of corrective actions that can be suggested.

# 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 $\mathcal{ELO}$, 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 $\mathcal{ELO}$ 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 $p_1, p_2$ and $p_3$, 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

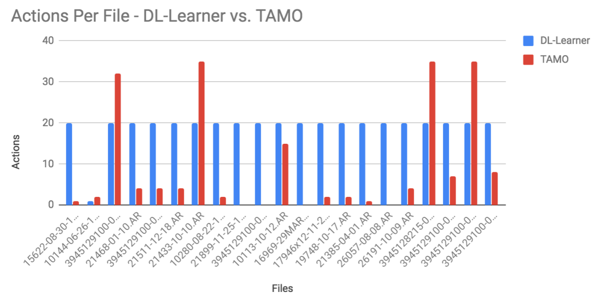


**Figure 3: Number of actions suggested for each file. DL-Learner vs. TAMO.**

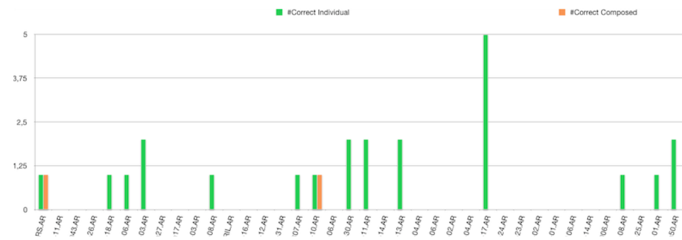


**Figure 4: The suggestions for the 50 files in partition $p3$, when consulting the knowledge base $KB_{100}^{p1+p2}$. In the figure are shown the correct atomic actions (green) and the correct composed actions (orange).**

We show in Figure 4 the number of relevant composed actions and the number of relevant individual actions (Y-axis) for each consulted file (X-axis) in partition $p3$. Each file has been consulted against the knowledge base trained with partitions $p_1 \cup p_2$, consisting of 100 samples ($KB_{100}^{p1+p2}$). A correct composed action, means that one of the suggestions proposed by the tool contains all the individual actions (replacements) that are required to solve the failure detected by the consulted .AR file. From the figure, we can see that this is the case only for 2 out of the 50 files in $p_3$. In the analysis made in the doctoral thesis [16] we showed that only 30% of the files have the possibility to be correctly classified if the set of samples is partitioned into training and validation sets. Therefore a low rate of full correct answers is expected. Nevertheless, in the awareness of these figures, the objective of these experiments is to show if our assumptions really hold, and if the model can provide valuable suggestions even under these circumstances.

To this end, we also consider partially correct suggestions, or correct *individual* answers. These are given by the individual replacements in any of the suggestions, that are inline with the expected results. From figure 4 it can be seen that the number of partial suggestions is much higher than considering fully composed corrective actions only. The model has given a partial answer for 14 out of the 50 files. This shows that if partial answers are considered, more relevant information can be provided. This makes sense in the context of suggested corrective actions, which do not intend to impose a repair, but to give hints on its resolution.

The figure shows that the correct and partial suggestions can be found by the approach, and that some of the suggestions may require further refinement, to make them relevant.

## 6.3 Evolution of the KB

The third experiment, has the objective of evaluating whether there exists an improvement in the quality of the knowledge base as more information is presented to it, and to estimate the evolution of this improvement.

To evaluate the quality of the knowledge base we use two criteria: first, precision and recall based on the ratio of positive and negative samples of the signatures found, and second, the average and the median related to the specificity of the signatures.

Three versions of the knowledge base are constructed: $KB_{25}$, $KB_{50}$ and $KB_{100}$ using training sets of size 25, 50 and 100, respectively, where each knowledge base doubles the size of the previous one. In this paper we compare the results of $KB_{25}$ vs. $KB_{100}$ to highlight their differences.

From all the situations each KB has discovered in the training phase, only some one of them are selected when consulting a file, ie. the most specific. We can see that $KB_{25}$ has selected only 10 situations, compared to 25 situations selected by $KB_{100}$. Evidently, the more files used to train a KB, the more situations it has available. 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 those 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

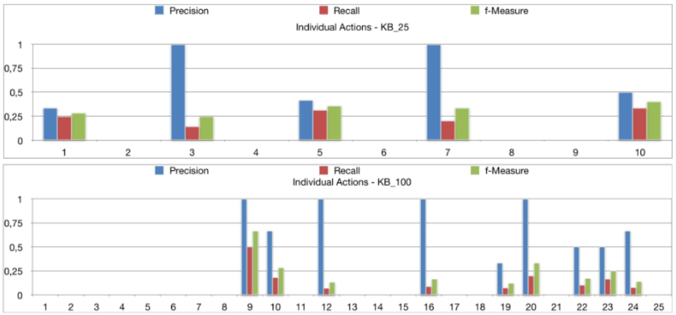


**Figure 5: Experimentation on the evolution of the KB. The figures show the precision, recall and f-measure for the individual actions. On top the results for $KB_{25}$ and on the bottom, the results for $KB_{100}$. The x-axis shows the situations.**

a formal model that considers all the available/necessary heterogeneous sources of information, in a single well-defined representation. We are in line with this direction, and focuses on conceiving methods for capturing novel knowledge from such representations.

*Concept Learning and Distinguishability.* Concept Learning in Description Logics and OWL is a direction of research that aims at learning schema axioms, such as definitions of classes, from existing ontologies and instance data. Most methods in this area are based on Inductive Logic Programming methods [14, 18]. In [23] the related work on methods of concept learning in DLs can be classified into three groups. The first group focuses on learnability in DL and presents some relatively simple algorithms [5]. The second group studies concept learning in DLs using refinement operators as in inductive logic programming [3, 8, 13, 20]. The third group exploits bisimulation for concept learning in DLs [17, 23].

Distinct from ontology based concept learning where a set of instances is given as positive examples of the target concept, the challenge of learning hidden situations consists in discovering significant situations from exponentially many unknown situations. To this end, our approach is based on an unsupervised refinement operator, where our aim is to cluster individuals in a way that we can have a DL concept definition that merely describes them.

## 8 CONCLUSION AND FUTURE WORK

We have presented an approach to discover interesting subsets of individuals in an ontology $O$, called situations. Each situation in $O$ defines a set of individuals that can be described by a DL concept. We have formalized the problem of finding situations, provided some of its main properties and presented an algorithm for situation discovery based on a unsupervised concept refinement operator (detailed in the doctoral thesis [16]). We have also shown how the problem of discovering failure signatures in avionics maintenance can be posed as the problem of finding situations in an ontology, thus enabling the developed algorithms to solve the problem. An additional but also important product of this process is the TAMO ontology, for avionics maintenance. The approach has been implemented using real world data, and the evaluation of the underlying techniques and the relevance of the suggestions obtained by the prototype were provided for the application domain.

As future work, the results and properties of the operator can be specified to support more expressive DLs (e.g. negation $\neg$ and conjunction $\sqcup$), 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


[1] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of IJCAI'05*, 2005.
[2] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
[3] L. Badea and S.-H. Nienhuys-Cheng. A refinement operator for description logics. In *Proceedings of ILP'00*, pages 40–59, 2000.
[4] C. C. Insaurralde. Intelligent autonomy for aerospace engineering systems. In *Proceedings of DASC'18*, pages 1–10, 2018.
[5] W. W. Cohen and H. Hirsh. Learning the classic description logic: Theoretical and experimental results. *KR*, 94:121–133, 1994.
[6] J. De Kleer and J. Kurien. Fundamentals of model-based diagnosis. *IFAC Proceedings Volumes*, 36(5):25–36, 2003.
[7] V. Ebrahimipour and S. Yacout. Ontology-based schema to support maintenance knowledge representation with a case study of a pneumatic valve. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(4):702–712, 2015.
[8] N. Fanizzi, C. d'Amato, and F. Esposito. Dl-foil concept learning in description logics. In *Proceedings of ILP'08*, pages 107–121, 2008.
[9] M. Funk, J. C. Jung, C. Lutz, H. Pulcini, and F. Wolter. Learning description logic concepts: When can positive and negative examples be separated. In *Proceedings of IJCAI'19*, 2019.
[10] JEDEC. *DICTIONARY OF TERMS FOR SOLID-STATE TECHNOLOGY, 7th Edition*. Global Standards for the Microelectronics Industry, 7 edition, 2018.
[11] M. H. Karray, B. Chebel-Morello, and N. Zerhouni. A formal ontology for industrial maintenance. *Applied Ontology*, 7(3):269–310, 2012.
[12] R. M. Keller. Ontologies for aviation data management. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–9. IEEE, 2016.
[13] J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203, 2010.
[14] J. Lehmann and J. Voelker. An introduction to ontology learning. *Perspectives on Ontology Learning. Amsterdam: IOS Press*, 2014.
[15] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, Mar. 2001.
[16] P. Medinacelli. *Knowledge Discovery for Avionics Maintenance, An Unsupervised Concept Learning Approach*. PhD thesis, Université Paris-Saclay / Paris-Sud, 2019.
[17] L. A. Nguyen and A. Szałas. Logic-based roughification. *Rough Sets and Intelligent Systems-Professor Zdzisław Pawlak in Memoriam*, pages 517–543, 2013.
[18] S.-H. Nienhuys-Cheng and R. De Wolf. *Foundations of inductive logic programming*, volume 1228. Springer Science & Business Media, 1997.
[19] L. Palacios Medinacelli, G. Lortal, C. Laudy, C. Sannino, L. Simon, G. Fusco, Y. Ma, and C. Reynaud. Avionics maintenance ontology building for failure diagnosis support. In *Proceedings of IC3K'16*, pages 204–209, 2016.
[20] D. Ratcliffe and K. Taylor. Refinement-based owl class induction with convex measures. In *Proceedings of JISTC'17*, pages 49–65, 2017.
[21] T. Regal and C. Pereira. Building an ontology for intelligent maintenance systems and spare parts supply chain integration. *IFAC Proceedings Volumes*, 19:7843–7848, 01 2014.
[22] M. Richardson and P. Domingos. Building large knowledge bases by mass collaboration. In *Proceedings of K-CAP '03*, 2003.
[23] T.-L. Tran, L. A. Nguyen, et al. Bisimulation-based concept learning for information systems in description logics. *Vietnam Journal of Computer Science*, 2(3):149–167, 2015.
[24] C. Wagner. Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Inf. Resour. Manage. J.*, 19(1):70–83, Jan. 2006.