# Two Applications of Concept Refinement

Roberto Confalonieri, Nicolas Troquard, Pietro Galliani,
Oliver Kutz, Rafael Peñaloza, and Daniele Porello

Free University of Bozen-Bolzano, Italy
{Firstname.Lastname}@unibz.it

**Abstract.** We describe two applications of refinement operators that can generalise and specialise concepts expressed in the $\mathcal{ALC}$ description logic language. The first application addresses the problem of analysing the joint coherence of some given concepts w.r.t. a background ontology. To this end, we apply Thagard's computational theory of coherence, in combination with semantic similarity between concepts defined by means of the generalisation operator. The second application focuses on repairing an inconsistent collective ontology that may result from the vote on axioms of multiple experts based on principles from social choice theory and judgment aggregation. We use the refinement operators, together with a reference ontology, to weaken some axioms and to repair the collective ontology.

## 1 Introduction

Consistency checking, concept subsumption, or instance checking, are typical reasoning problems in Description Logic (DL). They allow one to verify critical properties of ontologies before publication. A published ontology will then form one more brick of semantic knowledge on the web. Typical inference problems thus help knowledge experts at developing good ontologies in the same way that model checking is helping hardware engineers at making hardware without design flaws. But reasoning in DL may also have a more active role in the creation of ontologies. Non-standard inference problems [12] can indeed help the knowledge experts in taking sensible decisions during the building and the maintenance of an ontology, or can be used to discover new knowledge altogether. Examples of non-standard problems are least common subsumer and most specific concept [6], matching [1], and axiom pinpointing [18, 15].

Whilst in the DL literature these problems are essentially approached as reasoning tasks [4], concept learning and refinement in Machine Learning is achieved by means of concept refinement operators [13]. Refinement operators, which are used to generalise or specialise concept descriptions, have been transposed to deal with DL concepts [7, 14], with the aim of measuring concept similarity [9], coherence [8], dissimilarity[10], and repairing an inconsistent ontology [17].

In this paper, we extend and amend the definitions of our refinement operator for $\mathcal{ALC}$ concepts as presented in [8, 17], and then describe two application scenarios for concept refinement. The first application addresses the problem of

analysing the joint coherence of some given concepts w.r.t. a background ontology. To this end, we apply Thagard's computational theory of coherence, in combination with semantic similarity between concepts defined by means of the generalisation operator.

The second application is related to the problem of obtaining useful information from possibly inconsistent opinions on the collaborative web. We consider possibly inconsistent collective ontologies that may result when multiple experts are asked to vote on a set of axioms in a domain of interest. We then use the refinement operators, together with a reference ontology, to weaken some axioms and to repair the collective ontology.

## 2 Ontologies and Description logics

We take an ontology to be a set of formulas in an appropriate logical language, describing our domain of interest. Which logic we use precisely is not crucial for illustrating the proposed approach, but as much formal work on ontologies makes use of description logics (DLs), we will use these logics for all of our examples. A significant widely used basic description logic is $\mathcal{ALC}$, which is the logic we shall be working with here.

We present the basics of $\mathcal{ALC}$. For full details we refer to the literature [3]. The language of $\mathcal{ALC}$ is based on an alphabet consisting of *atomic concept names* $N_C$ and *role names* $N_R$. The set of *concept descriptions* (or *concept* for short) is generated by the following grammar (where $A$ represents an atomic concept name and $R$ a role names):

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C \ .$$

We collect all $\mathcal{ALC}$ concepts over $N_C$ and $N_R$ in $\mathcal{L}(\mathcal{ALC}, N_C, N_R)$.

A *TBox* is a finite set of concept inclusions of the form $C \sqsubseteq D$ (where $C$ and $D$ are concept descriptions). It is used to store terminological knowledge regarding the relationships between concepts. An *ABox* is a finite set of formulas of the form $A(a)$ ("object $a$ is an instance of concept $A$") and $R(a, b)$ ("objects $a$ and $b$ stand to each other in the $R$-relation"). It is used to store assertional knowledge regarding specific objects. The semantics of $\mathcal{ALC}$ is defined in terms of *interpretations* $I = (\Delta^I, \cdot^I)$ that map each object name to an element of its domain $\Delta^I$, each atomic concept to a subset of the domain, and each role name to a binary relation on the domain. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies all the axioms in $\mathcal{T}$. Given a TBox $\mathcal{T}$ and two concept descriptions $C$ and $D$, we say that $C$ is subsumed by $D$ w.r.t. $\mathcal{T}$, denoted as $C \sqsubseteq_{\mathcal{T}} D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Given a TBox $\mathcal{T}$ and two concept descriptions $C$ and $D$, we say that $C$ is strictly subsumed by $D$ w.r.t. $\mathcal{T}$, denoted as $C \sqsubset_{\mathcal{T}} D$, iff $C \sqsubseteq_{\mathcal{T}} D$ and it is not the case that $D \sqsubseteq_{\mathcal{T}} C$. Finally, we write $C \equiv_{\mathcal{T}} D$ when $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$.

## 3 Refinement operators of $\mathcal{ALC}$ concepts

Refinement operators are a well-known notion in Inductive Logic Programming where they are used to structure a search process for learning concepts from examples. In this setting, two types of refinement operators exist: specialisation refinement operators and generalisation refinement operators. While the former constructs specialisations of hypotheses, the latter constructs generalisations [13].

Given the quasi-ordered set $\langle \mathcal{L}(\mathcal{ALC}, N_c, N_R), \sqsubseteq \rangle$, a generalisation refinement operator is defined as follows:

$$\gamma_{\mathcal{T}}(C) \subseteq \{C' \in \mathcal{L}(\mathcal{ALC}, N_c, N_R) \mid C \sqsubseteq_{\mathcal{T}} C'\} \ .$$

Whereas a specialisation refinement operator is defined as follows:

$$\rho_{\mathcal{T}}(C) \subseteq \{C' \in \mathcal{L}(\mathcal{ALC}, N_c, N_R) \mid C' \sqsubseteq_{\mathcal{T}} C\} \ .$$

Generally speaking, a generalisation refinement operator takes a concept $C$ as input and returns a set of descriptions that are more general than $C$ by taking a TBox $\mathcal{T}$ into account. A specialisation operator, instead, returns a set of descriptions that are more specific. Whilst specialisation operators for $\mathcal{ALC}$ (and other description logics) have been studied in the literature [14], few proposals have been made for generalisation operators in less expressive logics due to the complexity of dealing with concept generalisations [7, 6, 20].

In order to define the refinement operators for $\mathcal{ALC}$, we need some auxiliary definitions. In the following, we assume that complex concepts $C$ are rewritten into negation normal form, and thus negation only appears in front of atomic concepts. We will use $=$ and $\neq$ between $\mathcal{ALC}$ concepts to denote syntactic identity and difference, respectively.

**Definition 1.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox with concept names from $N_C$. The set of* subconcepts *of $\mathcal{T}$ is given by*

$$\mathsf{sub}(\mathcal{T}) = \{\top, \bot\} \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(C) \cup \mathsf{sub}(D) \ ,$$

*where* sub *is defined over the structure of concept descriptions as follows:*

$$\begin{aligned}
\mathsf{sub}(A) &= \{A\} \\
\mathsf{sub}(\bot) &= \{\bot\} \\
\mathsf{sub}(\top) &= \{\top\} \\
\mathsf{sub}(\neg A) &= \{\neg A, A\} \\
\mathsf{sub}(C \sqcap D) &= \{C \sqcap D\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D) \\
\mathsf{sub}(C \sqcup D) &= \{C \sqcup D\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D) \\
\mathsf{sub}(\forall R.C) &= \{\forall R.C\} \cup \mathsf{sub}(C) \\
\mathsf{sub}(\exists R.C) &= \{\exists R.C\} \cup \mathsf{sub}(C) \ .
\end{aligned}$$

Based on $\mathsf{sub}(\mathcal{T})$, we define the upward and downward cover sets of atomic concepts.[1] Intuitively, the upward set of the concept $C$ collects the most specific subconcepts found in the Tbox $\mathcal{T}$ that are more general (subsume) $C$; conversely, the downward set of $C$ collects the most general subconcepts from $\mathcal{T}$ that are subsumed by $C$. The concepts in $\mathsf{sub}(\mathcal{T})$ are *some* concepts that are relevant in the context of TBox $\mathcal{T}$, and that are used as building blocks for generalisations and specialisations. It is readily seen that given a finite TBox $\mathcal{T}$ the set $\mathsf{sub}(\mathcal{T})$ is also finite. The properties of $\mathsf{sub}(\mathcal{T})$ guarantee that the upward and downward cover sets are finite.

**Definition 2.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox over $N_C$. The* upward cover set *of the concept $C$ with respect to $\mathcal{T}$ is:*

$$\mathsf{UpCov}_{\mathcal{T}}(C) := \{D \in \mathsf{sub}(\mathcal{T}) \mid C \sqsubseteq_{\mathcal{T}} D \text{ and there is no } D' \in \mathsf{sub}(\mathcal{T}) \text{ with } C \sqsubset_{\mathcal{T}} D' \sqsubset_{\mathcal{T}} D\} \ . \tag{1}$$

*The* downward cover set *of the concept $C$ with respect to $\mathcal{T}$ is:*

$$\mathsf{DownCov}_{\mathcal{T}}(C) := \{D \in \mathsf{sub}(\mathcal{T}) \mid D \sqsubseteq_{\mathcal{T}} C \text{ and there is no } D' \in \mathsf{sub}(\mathcal{T}) \text{ with } D \sqsubset_{\mathcal{T}} D' \sqsubset_{\mathcal{T}} C\} \ . \tag{2}$$

In the following, we note $\mathsf{nnf}$ the function that for every concept $C$, returns its negative normal form $\mathsf{nnf}(C)$. We can now define our generalisation refinement operator for $\mathcal{ALC}$ as follows.

**Definition 3.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox. We define $\gamma_{\mathcal{T}}$, the* generalisation refinement operator *w.r.t. $\mathcal{T}$, inductively over the structure of concept descriptions as:*

$$\begin{aligned}
\gamma_{\mathcal{T}}(A) &= \mathsf{UpCov}_{\mathcal{T}}(A) \\
\gamma_{\mathcal{T}}(\neg A) &= \{\mathsf{nnf}(\neg C) \mid C \in \mathsf{DownCov}_{\mathcal{T}}(A)\} \cup \mathsf{UpCov}_{\mathcal{T}}(\neg A) \\
\gamma_{\mathcal{T}}(\top) &= \mathsf{UpCov}_{\mathcal{T}}(\top) \\
\gamma_{\mathcal{T}}(\bot) &= \mathsf{UpCov}_{\mathcal{T}}(\bot) \\
\gamma_{\mathcal{T}}(C \sqcap D) &= \{C' \sqcap D \mid C' \in \gamma_{\mathcal{T}}(C)\} \cup \{C \sqcap D' \mid D' \in \gamma_{\mathcal{T}}(D)\} \cup \mathsf{UpCov}_{\mathcal{T}}(C \sqcap D) \\
\gamma_{\mathcal{T}}(C \sqcup D) &= \{C' \sqcup D \mid C' \in \gamma_{\mathcal{T}}(C)\} \cup \{C \sqcup D' \mid D' \in \gamma_{\mathcal{T}}(D)\} \cup \mathsf{UpCov}_{\mathcal{T}}(C \sqcup D) \\
\gamma_{\mathcal{T}}(\forall R.C) &= \{\forall R.C' \mid C' \in \gamma_{\mathcal{T}}(C)\} \cup \mathsf{UpCov}_{\mathcal{T}}(\forall R.C) \\
\gamma_{\mathcal{T}}(\exists R.C) &= \{\exists R.C' \mid C' \in \gamma_{\mathcal{T}}(C)\} \cup \mathsf{UpCov}_{\mathcal{T}}(\exists R.C)
\end{aligned}$$

When there is no ambiguity, or to refer to an arbitrary TBox, we can omit the subscript $\mathcal{T}$ from the operator $\gamma_{\mathcal{T}}$.

Given a generalisation refinement operator $\gamma$, $\mathcal{ALC}$ concepts are related by refinement paths as described next.

**Definition 4.** *For every concept $C$, we note $\gamma^i(C)$ the $i$-th iteration of its generalisation. It is inductively defined as follows:*

[1] Downward and upward cover sets were similarly defined in [8], but only for atomic concepts. The generalisation operator $\gamma$ is also a variant from the one defined in [8].

– $\gamma^0(C) = \{C\}$;
– $\gamma^{j+1}(C) = \gamma^j(C) \cup \bigcup_{C' \in \gamma^j(C)} \gamma(C')$ , $j \geq 0$.

**Definition 5.** *The minimal number of generalisations to be applied in order to generalise $C$ to $D$ is called the distance between $C$ and $D$, noted $\lambda(C \xrightarrow{\gamma} D)$. Formally, $\lambda(C \xrightarrow{\gamma} D) = \min\{j \mid j \geq 0 \text{ and } D \in \gamma^j(C)\}$.*

$\lambda(\_ \xrightarrow{\gamma} \_)$ is a partial function that is defined only when the concept passed as first argument can eventually be refined into the concept passed as second argument.

**Definition 6.** *The set of all concepts that can be reached from $C$ by means of $\gamma$ in a finite number of steps is*

$$\gamma^*(C) = \bigcup_{i \geq 0} \gamma^i(C) \ .$$

Some basic properties follow.

**Lemma 1.** *For every TBox $\mathcal{T}$ and for every concept $C$:*

1. ***generalisation:*** *if $X \in \gamma_{\mathcal{T}}(C)$ then $C \sqsubseteq_{\mathcal{T}} X$*
2. ***reflexivity:*** *if $C \in \mathsf{sub}(\mathcal{T})$ then $C \in \mathsf{UpCov}_{\mathcal{T}}(C)$*
3. ***relevant completeness:*** $\mathsf{UpCov}_{\mathcal{T}}(C) \subseteq \gamma_{\mathcal{T}}(C)$
4. ***trivial generalisability:*** $\top \in \gamma^*_{\mathcal{T}}(C)$
5. ***generalisation finiteness:*** *$\gamma_{\mathcal{T}}(C)$ is finite*

Although generalisation finiteness holds, it is not the case that $\gamma^*(C)$ is finite for every concept $C$. Indeed, the iterated application of $\gamma$ can produce an infinite chain of generalisations. The following example illustrates that.

*Example 1.* Let $\mathcal{T} := \{A \sqsubseteq \exists r.A\}$. At the first iteration we have $\gamma(A) = \{A, \exists r.A\}$. Then we have $\gamma(\exists r.A) = \{\exists r.A, \exists r.\exists r.A\} \cup \{\top\}$. (Notice that $\top$ is reached: $\top \in \gamma^2(A)$.) Continuing the iteration of the generalisation of the concept description $A$, we have $(\exists r.)^k A \in \gamma^k(A)$ for every $k \geq 0$.

This is not a feature that is caused by the existential quantification alone. Similar examples exist that involve universal quantification, disjunction and conjunction. To address this issue, one can make some assumptions over the structure of the TBox, for instance, by restricting to no-cyclic TBoxes. Alternatively, we can take into account the syntactic depth of concepts, by restricting the number of nested quantifiers, conjunctions, and disjunctions in a concept description to a fixed constant $k$ [7, 2]. To this end, we introduce the definition of syntactic depth of a concept as follows.

**Definition 7.** *The syntactic depth of an $\mathcal{ALC}$ concept description $C$ is defined as:*

$$\begin{aligned}
\mathsf{depth}(\top) &= 0 \\
\mathsf{depth}(\bot) &= 0 \\
\mathsf{depth}(A) &= 0 \\
\mathsf{depth}(\neg A) &= 1 \\
\mathsf{depth}(C \sqcap D) &= \max\{\mathsf{depth}(C), \mathsf{depth}(D)\} + 1 \\
\mathsf{depth}(C \sqcup D) &= \max\{\mathsf{depth}(C), \mathsf{depth}(D)\} + 1 \\
\mathsf{depth}(\exists r.C) &= \mathsf{depth}(\mathsf{C}) + 1 \\
\mathsf{depth}(\forall r.C) &= \mathsf{depth}(\mathsf{C}) + 1 \ .
\end{aligned}$$

Based on the syntactic depth of a concept we modify the definition of the generalisation refinement operator $\gamma$ to take a fixed constant $k \in \mathbb{N}_{>0}$ of nested quantifiers into account. More precisely, let $\gamma_k$ be defined as:

$$\gamma_{\mathcal{T},k}(C) := \begin{cases} \gamma_{\mathcal{T}}(C) & \text{if } \mathsf{depth}(C) \leq k \\ \mathsf{UpCov}_{\mathcal{T}}(\top) & \text{otherwise.} \end{cases}$$

We can define a specialisation operator in an analogous way.

**Definition 8.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox. We define $\rho_{\mathcal{T}}$, the* specialisation refinement operator *w.r.t. $\mathcal{T}$, inductively over the structure of concept descriptions as:*

$$\begin{aligned}
\rho_{\mathcal{T}}(A) &= \mathsf{DownCov}_{\mathcal{T}}(A) \\
\rho_{\mathcal{T}}(\neg A) &= \{\mathsf{nnf}(\neg C) \mid C \in \mathsf{UpCov}_{\mathcal{T}}(A)\} \cup \mathsf{DownCov}_{\mathcal{T}}(\neg A) \\
\rho_{\mathcal{T}}(\top) &= \mathsf{DownCov}_{\mathcal{T}}(\top) \\
\rho_{\mathcal{T}}(\bot) &= \mathsf{DownCov}_{\mathcal{T}}(\bot) \\
\rho_{\mathcal{T}}(C \sqcap D) &= \{C' \sqcap D \mid C' \in \rho_{\mathcal{T}}(C)\} \cup \{C \sqcap D' \mid D' \in \rho_{\mathcal{T}}(D)\} \cup \mathsf{DownCov}_{\mathcal{T}}(C \sqcap D) \\
\rho_{\mathcal{T}}(C \sqcup D) &= \{C' \sqcup D \mid C' \in \rho_{\mathcal{T}}(C)\} \cup \{C \sqcup D' \mid D' \in \rho_{\mathcal{T}}(D)\} \cup \mathsf{DownCov}_{\mathcal{T}}(C \sqcup D) \\
\rho_{\mathcal{T}}(\forall R.C) &= \{\forall R.C' \mid C' \in \rho_{\mathcal{T}}(C)\} \cup \mathsf{DownCov}_{\mathcal{T}}(\forall R.C) \\
\rho_{\mathcal{T}}(\exists R.C) &= \{\exists R.C' \mid C' \in \rho_{\mathcal{T}}(C)\} \cup \mathsf{DownCov}_{\mathcal{T}}(\exists R.C)
\end{aligned}$$

Naturally, properties analogous to the ones of Lemma 1 hold for the specialisation operator as well. Since $\rho$ can yield an infinite specialisation chain, its definition can be extended by taking into account the depth of a concept description as done for the generalisation operator.

$$\rho_{\mathcal{T},k}(C) := \begin{cases} \rho_{\mathcal{T}}(C) & \text{if } \mathsf{depth}(C) \leq k \\ \mathsf{DownCov}_{\mathcal{T}}(\bot) & \text{otherwise.} \end{cases}$$

When there is no ambiguity, or to refer to an arbitrary TBox, we can omit the subscript $\mathcal{T}$ from the operator $\rho_{\mathcal{T}}$. Just as we did for the generalisation operator $\gamma$, we denote $\rho^i$ (resp. $\rho^*$) to be the $i$-th iteration (resp. the unbounded finite iteration) of the specialisation operator $\rho$. The distance between concepts $\lambda(\_ \xrightarrow{\rho} \_)$ w.r.t. the specialisation $\rho$ is also defined as expected.

## 4 Deciding concept coherence

Given a body of knowlegde, Thagard [19] addresses the problem of determining which pieces of information to accept and which to reject based on how they cohere and incohere among them. These pieces of information can be hypotheses, beliefs, propositions or concepts. It is assumed that when two pieces cohere, they tend to be accepted together or rejected together; and when two pieces incohere, one tends to be accepted while the other tends to be rejected.

Thagard provides a clear technical description of the coherence problem as a constraint satisfaction problem, but he does not clarify the nature of the coherence and incoherence relations that arise between pieces of information. In this section, we show how coherence and incoherence relations between concepts can be determined according to a similarity based on the generalisation operator $\gamma$. Secondly, we show how conceptual coherence, one of the types of coherence proposed by Thagard, can be used to decide the coherence between $\mathcal{ALC}$ concepts.

### 4.1 Common generalisations and similarity

We are interested in common generalisations that have minimal distance from the concepts, or in case their distance is equal, the ones that are far from $\top$.

**Definition 9.** *An $\mathcal{ALC}$ concept description $G$ is a common generalisation of $C_1$ and $C_2$ if $G \in \gamma^*(C_1) \cap \gamma^*(C_2)$ and, furthermore, $G$ is such that for any other $G' \in \gamma^*(C_1) \cap \gamma^*(C_2)$ with $(G' \neq G)$ we have:*

- $\lambda(C_1 \xrightarrow{\gamma} G) + \lambda(C_2 \xrightarrow{\gamma} G) < \lambda(C_1 \xrightarrow{\gamma} G') + \lambda(C_2 \xrightarrow{\gamma} G')$, *or*
- $\lambda(C_1 \xrightarrow{\gamma} G) + \lambda(C_2 \xrightarrow{\gamma} G) = \lambda(C_1 \xrightarrow{\gamma} G') + \lambda(C_2 \xrightarrow{\gamma} G')$ *and* $\lambda(G \xrightarrow{\gamma} \top) \geq \lambda(G' \xrightarrow{\gamma} \top)$.

Notice that common generalisations, as per the above definition, are not unique. However, for any two common generalisations $G$ and $G'$ of $C_1$ and $C_2$, $\lambda(C_1 \xrightarrow{\gamma} G) + \lambda(C_2 \xrightarrow{\gamma} G) = \lambda(C_1 \xrightarrow{\gamma} G') + \lambda(C_2 \xrightarrow{\gamma} G')$ and $\lambda(G \xrightarrow{\gamma} \top) = \lambda(G' \xrightarrow{\gamma} \top)$. Thus, any one of them will result in the same value for our generalisation-based similarity measure between concepts, and therefore in the same coherence or incoherence judgements. In the following, we denote $C_1 \blacktriangle C_2$ a common generalisation of $C_1$ and $C_2$; $C_1 \blacktriangle C_2$ is a concept that always exists.

The common generalisation of two concepts $C$ and $D$ can be used to measure the similarity between concepts in a quantitative way. To estimate the quantity of information of any description $C$ we take into account the length of the minimal generalisation path that leads from $C$ to the most general term $\top$.

In order to define a similarity measure, we need to compare what is common to $C$ and $D$ with what is not common. The length $\lambda(C \blacktriangle D \xrightarrow{\gamma} \top)$ estimates the informational content that is common to $C$ and $D$, and the lengths $\lambda(C \xrightarrow{\gamma} C \blacktriangle D)$ and $\lambda(D \xrightarrow{\gamma} C \blacktriangle D)$ measures how much $C$ and $D$ are different. Then, the common generalisation-based similarity measure can be defined as follows [16].

**Definition 10.** *The similarity between two concepts $C$ and $D$, denoted by $S_\lambda(C, D)$, is defined as:*

$$S_\lambda(C,D) = \begin{cases} \dfrac{\lambda(C \blacktriangle D \xrightarrow{\gamma} \top)}{\lambda(C \blacktriangle D \xrightarrow{\gamma} \top) + \lambda(C \xrightarrow{\gamma} C \blacktriangle D) + \lambda(D \xrightarrow{\gamma} C \blacktriangle D)} & \text{if } C \neq \top \text{ or } D \neq \top \\ 1 & \text{otherwise .} \end{cases}$$

The measure $S_\lambda$ estimates the ratio between the amount of information that is shared and the total information content. The range of the similarity function is the interval $[0, 1]$, where 0 represents the minimal similarity between concepts (when their common generalisation is equal to $\top$), and 1 represents maximal similarity (when the concepts are equivalent).

### 4.2 Coherence

We shall assume coherence between two concept descriptions when they are sufficiently similar so that "there are objects to which both apply;" and we shall assume incoherence when they are not sufficiently similar so that "an object falling under one concept tends not to fall under the other concept."

In this formalisation of conceptual coherence, we associate this 'sufficiently similar' condition with a value $\delta$. The intuition behind the following definition is that similar concepts whose common generalisation is far from $\top$ should cohere, and incohere otherwise.

**Definition 11 (Coherence Relations).** *Given a set $\{C_1, \ldots, C_n\}$ of $\mathcal{ALC}$ concepts, concept descriptions $C, D \notin \{\top, \bot\}$, we will say for each pair of concepts $\langle C_i, C_j \rangle$ $(1 \leq i, j \leq n, i \neq j)$:*

- *$C_i$ coheres with $C_j$, if $S_\lambda(C_i, C_j) > 1 - \delta$*
- *$C_i$ incoheres with $C_j$, if $S_\lambda(C_i, C_j) \leq 1 - \delta$*

*where*

$$\delta = \frac{\lambda(C_i \blacktriangle C_j \xrightarrow{\gamma} \top)}{\max\{\lambda(C_i \xrightarrow{\gamma} \top), \lambda(C_j \xrightarrow{\gamma} \top)\}} \ .$$

In this definition, $\lambda(C_i \blacktriangle C_j \xrightarrow{\gamma} \top)$ is normalised to the interval $[0, 1]$ in order to make it comparable with the similarity measure.
Based on the previous definition we can determine the coherence graph as follows.

**Definition 12 (Thagardian Coherence Graph).** *The* coherence graph *for the set of $\mathcal{ALC}$ concepts $\{C_1, \ldots, C_n\}$ is the edge-weighted and undirected graph $G = \langle V, E, w \rangle$ whose vertices are $C_1, \ldots, C_n$, whose edges link concepts that either cohere or incohere according to Definition 11, and whose edge-weight function $w$ is given as follows:*

$$w(\{C, D\}) = \begin{cases} 1 & \text{if } C \text{ and } D \text{ cohere} \\ -1 & \text{if } C \text{ and } D \text{ incohere .} \end{cases}$$

This definition creates a concept graph in the sense of Thagard where only binary values 'coheres' or 'incoheres' are recorded, represented by '+1' and '-1', respectively, but our Def. 11 can give rise also to graded versions of coherence and incoherence relations.

Then, given a coherence graph, one is interested in finding a partition that maximises the number of satisfied constraints, where constraints are coherence and incoherence relations. Roughly speaking, a constraint is satisfied if when two concepts cohere, they fall in the same partition; and when two concepts incohere, each of them falls in a different partition [8].

### 4.3 Analysing the joint coherence of $\mathcal{ALC}$ concepts

Given an $\mathcal{ALC}$ TBox representing a background ontology, and a set of $\mathcal{ALC}$ concepts $\{C_1, \ldots, C_n\}$ as input, we analyse their joint coherence as follows. First, we compute the coherence graph and the maximising partitions for the input concepts, and use them to decide which concepts to keep and which ones to discard. The pairwise comparison and the maximising coherence degree partitions will give us the biggest subsets of coherent input concepts. Then, we compute the nearest common generalisations of the accepted concepts, to convey a justification of why certain concepts were partitioned together.

It is worth noticing that according to our definition of coherence relation, inconsistent concepts can cohere provided that they are sufficiently similar and their common generalisation is far from the $\top$ concept.

## 5 Repairing collective ontologies

For this application, we restrict our attention to TBox axioms. As usual, a TBox $\mathcal{T}$ is *consistent* if it has a model, and *inconsistent* otherwise. The relation $\models_O$ denotes the consequence relation w.r.t. an ontology $O$.

This section follows the presentation of [17].

### 5.1 Aggregating individual ontologies

Consider an arbitrary but fixed finite set $\Phi$ of $\mathcal{ALC}$ TBox statements over this alphabet. We call $\Phi$ the *agenda* and any set $O \subseteq \Phi$ an *ontology*. We denote the set of all those ontologies that are *consistent* by $\mathrm{On}(\Phi)$.

Let $\mathcal{N} = \{1, \ldots, n\}$ be a finite set of *agents* (or *individuals*). Each agent $i \in \mathcal{N}$ provides a consistent ontology $O_i \in \mathrm{On}(\Phi)$. An *ontology profile* is a vector $\boldsymbol{O} = (O_1, \ldots, O_n) \in \mathrm{On}(\Phi)^{\mathcal{N}}$ of consistent ontologies, one for each agent. We write $N_\phi^{\boldsymbol{O}} := \{i \in \mathcal{N} \mid \phi \in O_i\}$ for the set of agents that include $\phi$ in their ontology under profile $\boldsymbol{O}$. Our object of study are *ontology aggregators*.

**Definition 13 (Ontology aggregators).** *An ontology aggregator is a function* $F : \mathrm{On}(\Phi)^{\mathcal{N}} \to 2^{\Phi}$ *mapping any profile of consistent ontologies to an ontology.*

Observe that, according to this definition, the ontology we obtain as the outcome of an aggregation process needs not be consistent. With no surprise, this is the case of the majority rule, which is nonetheless widely applied in any political scenarios. In our setting, the majority rule is defined as follows.

**Definition 14 (Absolute majority rule).** *The absolute majority rule is the ontology aggregator $F_m$ mapping any given profile $\boldsymbol{O} \in \mathrm{On}(\Phi)^{\mathcal{N}}$ to the ontology*

$$F_m(\boldsymbol{O}) := \left\{ \phi \in \Phi \mid \#N_\phi^{\boldsymbol{O}} > \frac{n}{2} \right\} \ .$$

In the remainder of this section we propose to repair inconsistent collective ontologies obtained from the aggregation of experts' individual ontologies.

### 5.2 Axiom Weakening

Roughly speaking, weakening an axiom $C \sqsubseteq D$ amounts to *enlarging* the set of interpretations that satisfy the axiom. This could be done in different ways: Either by substituting $C \sqsubseteq D$ with $C \sqsubseteq D'$, where $D'$ is a more general concept than $D$ (i.e., its interpretation is larger); or, by modifying the axiom $C \sqsubseteq D$ to $C' \sqsubseteq D$, where $C'$ is a more specific concept than $C$; or even by generalising and specialising simultaneously to obtain $C' \sqsubseteq D'$.

Given an ontology $O$, we denote the set of its concept names of $O$ by $N_C^O$. We want to define a procedure to change axioms gradually by replacing them with less restrictive axioms. Recall that $\gamma_O$ denotes the generalisation of a concept and $\rho_O$ denotes its specialisation with respect to a given ontology $O$.

**Definition 15 (Axiom weakening).** *Given an axiom $C \sqsubseteq D$ of $O$, the set of* weakenings *of $C \sqsubseteq D$ in $O$, denoted by $g_O(C \sqsubseteq D)$ is the set of all axioms $C' \sqsubseteq D'$ such that*

$$C' = \rho_O^*(C) \text{ and } D' = \gamma_O^*(D) \ .$$

If the ontology $O$ is consistent, the weakening of an axiom in $O$ is always satisfied by a super set of the interpretations that satisfy the axiom. Let $I = (\Delta^I, \cdot^I)$ be an interpretation. Then by definition the class of all entities that fulfil the axiom $C \sqsubseteq D$ is $(\Delta^I \setminus C^I) \cup D^I$. A weakening of $C \sqsubseteq D$ either specialises $C$, therefore restricting $C^I$, and accordingly extending $\Delta^I \setminus C^I$, or generalises $D$, therefore, extending $D^I$. Hence, the set of entities for which $C \sqsubseteq D$ holds is a subset of the set of entities for which any axiom in $g_O(C \sqsubseteq D)$ holds. The following result holds.

**Lemma 2.** *For every axiom $\phi$, if $\phi \in g_O(\psi)$, then $\psi \models_O \phi$.*

Moreover, note that $\bot \sqsubseteq \top$ always belongs to $g_O(C \sqsubseteq D)$. We want to model how to repair any inconsistent set of axioms $Y$ of $\mathcal{ALC}$, by appealing to a consistent reference ontology R. Notice that, even though it is not desirable, R can be dissociated from the axioms in the collective ontology. If the ontology R does not refer to some of the atomic concepts in $C$ or $D$, then their generalisation is

the most general concept $\top$ and their specialisation is the most specific concept $\bot$.[2]

Any inconsistent set of axioms $Y$ can in principle be repaired by means of a sequence of weakenings of the axioms in $Y$ with respect to R.

**Lemma 3.** *Let* R *be a consistent reference ontology and* $Y$ *a minimally inconsistent set of axioms. There exists a subset* $\{\psi_1, \ldots, \psi_n\} \subseteq Y$ *and weakenings* $\psi'_i \in g_{\mathrm{R}}(\psi_i)$ *for* $i, 1 \leq i \leq n$ *such that* $(Y \setminus \{\psi_1, \ldots, \psi_n\}) \cup \{\psi'_1, \ldots, \psi'_n\} \cup \mathrm{R}$ *is consistent.*

The lemma ensures that a minimally inconsistent set of axioms can be adequately weakened to be integrated consistently into another consistent ontology. In the worst case these axioms are weakened to become a tautology. However, we are interested in weakening axioms as little as possible to remain close to the original axioms. Notice that every axiom in $g_O(C \sqsubseteq D)$ is obtained by applying $\gamma$ and $\rho$ a finite number of times. Hence, we can define $\lambda_O$ to be a refinement distance in an ontology $O$, such that for every $C' \sqsubseteq D' \in g_O(C \sqsubseteq D)$,

$$\lambda_O(C \sqsubseteq D, C' \sqsubseteq D') = \lambda(C \xrightarrow{\rho_O} C') + \lambda(D \xrightarrow{\gamma_O} D') \ .$$

Repair strategies can exploit this distance to guide the weakening of axioms that are the least stringent. Indeed, by minimising $\lambda_O$, we are trying to find the weakening of the axiom that is as close as possible to the original axiom in the context of $O$. Moreover, by trying to minimise the distance, we are trying to prevent non-informative axioms to be selected as weakenings. In other words, axioms like $\bot \sqsubseteq \top$, $\bot \sqsubseteq D$, or $C \sqsubseteq \top$ should only be selected if no other options are available. In principle, we can also provide refined constraints on the generalisation and specialisation paths, e.g. by fixing an ordering of the concepts of the ontology $O$ that determines which concepts are to be generalised or specialised first.

### 5.3 Fixing Collective Ontologies via Axiom Weakenings

When $F(\boldsymbol{O})$ is inconsistent, we can adopt the general strategy described in Algorithm 1 to repair it w.r.t. a given (fixed) reference ontology R. The algorithm finds all the minimally inconsistent subsets $Y_1, \ldots, Y_n$ of $F(\boldsymbol{O})$ (e.g., using the methods from [18, 5]) and repairs each of them by weakening one of its axioms to regain consistency. From all the possible choices made to achieve this goal, the algorithm selects one that minimizes the distance $\lambda_O$ (line 4). This process corrects all original causes for inconsistency, but may still produce an inconsistent ontology due to masking [11]. Hence, the process is repeated until a consistent ontology is found.

[2] Notice that $\gamma_{\mathcal{T}}$ and $\rho_{\mathcal{T}}$ are defined on arbitrary $\mathcal{ALC}$ formulas.

**Algorithm 1** Fixing ontologies through weakening.

**Procedure** FIX-ONTOLOGY($O$,R) ▷ $O$ inconsistent ontology, R reference ontology

**while** $O$ is inconsistent **do**
  $\mathcal{Y} \leftarrow \text{MIS}(O)$ ▷ find all minimally inconsistent subsets of $O$
  **for** $Y \in \mathcal{Y}$ **do**
    **choose** $\psi \in Y, \psi' \in g_{\text{R}}(\psi)$ with $Y \setminus \{\psi\} \cup \{\psi'\}$ consistent, $\lambda_O(\psi, \psi')$ minimal
    $O \leftarrow (O \setminus \{\psi\}) \cup \{\psi'\}$
**return** $O$

## 6 Conclusions and Future Work

We have illustrated the usefulness of our generalisation and specialisation operators in two use cases: the first, involving defining a similarity measure for $\mathcal{ALC}$ concepts and employing this in Thagard's coherence framework, and the second involving inconsistency debugging in socially aggregated ontologies.

Future work includes a more systematic development of the introduced generalisation and specialisation operators employing a variety of semantics-driven definitions. Furthermore, the duality between generalisaton and specialisation needs to be further studied.

On the tool side, we are currently working towards a full implementation of our refinement operators in order to evaluate their usefulness in larger scale real world examples.

## References


1. Baader, F., Küsters, R., Borgida, A., McGuinness, D.: Matching in description logics. Journal of Logic and Computation 9(3), 411–447 (1999)
2. Baader, F.: Computing the Least Common Subsumer in the Description Logic $\mathcal{EL}$ w.r.t. Terminological Cycles with Descriptive Semantics. In: Ganter, B., de Moor, A., Lex, W. (eds.) Conceptual Structures for Knowledge Creation and Communication, Lecture Notes in Computer Science, vol. 2746, pp. 117–130. Springer Berlin Heidelberg (2003)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)
4. Baader, F., Küsters, R.: Non-standard Inferences in Description Logics: The Story So Far. In: Mathematical Problems from Applied Logic I, International Mathematical Series, vol. 4, pp. 1–75. Springer New York (2006)
5. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation 20(1), 5–34 (February 2010), special Issue: Tableaux and Analytic Proof Methods
6. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the least common subsumer w.r.t. a background terminology. Journal of Applied Logic 5(3), 392 – 420 (2007)
7. Confalonieri, R., Eppe, M., Schorlemmer, M., Kutz, O., Peñaloza, R., Plaza, E.: Upward Refinement Operators for Conceptual Blending in $\mathcal{EL}^{++}$. Annals of Mathematics and Artificial Intelligence (2016), doi:10.1007/s10472-016-9524-8

8. Confalonieri, R., Kutz, O., Galliani, P., Porello, D., Peñaloza, R., Schorlemmer, M., Troquard, N.: Coherence, Similarity, and Concept Generalisation. In: Proceedings of the 30th International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017)
9. d'Amato, C., Fanizzi, N., Esposito, F.: A semantic similarity measure for expressive description logics. In: Pettorossi, A. (ed.) Proceedings of Convegno Italiano di Logica Computazionale (CILC-05). Rome, Italy (2005)
10. d'Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for alc concept descriptions. In: Proceedings of the 2006 ACM symposium on Applied computing. pp. 1695–1699. ACM (2006)
11. Horridge, M., Parsia, B., Sattler, U.: Justification masking in ontologies. In: KR 2012. AAAI Press (2012)
12. Küsters, R.: Non-Standard Inferences in Description Logics, Lecture Notes in Artificial Intelligence, vol. 2100. Springer (2001)
13. van der Laag, P.R., Nienhuys-Cheng, S.H.: Completeness and properness of refinement operators in inductive logic programming. The Journal of Logic Programming 34(3), 201 – 225 (1998)
14. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning 78(1-2), 203–250 (2010)
15. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic ALC. In: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1. pp. 269–274. AAAI'06, AAAI Press (2006)
16. Ontañón, S., Plaza, E.: Similarity measures over refinement graphs. Machine Learning 87(1), 57–92 (Apr 2012)
17. Porello, D., Troquard, N., Confalonieri, R., Galliani, P., Kutz, O., Peñaloza, R.: Repairing Socially Aggregated Ontologies Using Axiom Weakening. In: 20th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2017). LNCS, Springer (2017)
18. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. IJCAI-03. pp. 355–362. Morgan Kaufmann (2003)
19. Thagard, P.: Coherence in thought and action. The MIT Press (2000)
20. Zarrieß, B., Turhan, A.Y.: Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes. In: Proceedings of the 23th International Joint Conference on Artificial Intelligence. pp. 1191–1197. IJCAI '13, AAAI Press (2013)