Kalpa Publications in Computing

Volume 1, 2017, Pages 27–42

LPAR-21S: IWIL Workshop
and LPAR Short Presentations

Reasoning with Concept Diagrams about Antipatterns

Zohreh Shams[1], Mateja Jamnik[1], Gem Stapleton[2], and Yuri Sato[2]

[1] Computer Laboratory, University of Cambridge, Cambridge, UK
{zohreh.shams,mateja.jamnik}@cl.cam.ac.uk
[2] Visual Modelling Group, University of Brighton, Brighton, UK
{g.e.stapleton, y.sato}@brighton.ac.uk

Abstract

Ontologies are notoriously hard to define, express and reason about. Many tools have been developed to ease the debugging and the reasoning process with ontologies, however they often lack accessibility and formalisation. A visual representation language, *concept diagrams*, was developed for expressing and reasoning about ontologies in an accessible way. Indeed, empirical studies show that concept diagrams are cognitively more accessible to users in ontology debugging tasks. In this paper we answer the question of " How can concept diagrams be used to reason about inconsistencies and incoherence of ontologies?". We do so by formalising a set of inference rules for concept diagrams that enables stepwise verification of the inconsistency and/or incoherence of a set of ontology axioms. The design of inference rules is driven by empirical evidence that concise (merged) diagrams are easier to comprehend for users than a set of lower level diagrams that offer a one-to-one translation of OWL ontology axioms into concept diagrams. We prove that our inference rules are sound, and exemplify how they can be used to reason about inconsistencies and incoherence. Finally, we indicate how our rules can serve as a foundation for new rules required when representing ontologies in diverse new domains.

1 Introduction

Ontologies are sets of statements that represent properties of individuals, classes and properties, that have mainly been expressed using symbolic notations such as description logics [3] and OWL [1]. Although ontologies are widely used for knowledge representation in domains involving diverse stakeholders, the languages they are expressed in are often inaccessible to those unfamiliar with mathematical notations. To address this shortcoming some ontology editors, such as Protégé [2], provide visualisation facilities. Instead of using diagrams as an auxiliary tool to aid comprehension and accessibility, like in Protégé, some have taken one step further by using a logic that is fundamentally diagrammatic (e.g., [4, 7]). However, these diagrammatic notations are either informal [4] or do not fully exploit the potential of formal diagrammatic notations (e.g., [7]). Concept diagrams [14] are designed for expressing ontologies by taking into account cognitive theories of what makes a diagrammatic notation accessible, in particular to novice users [5]. They are extensions of Euler diagrams and, in addition to closed curves for set representation, they use dots (spiders) and arrows for individuals and properties, respectively.

T.Eiter, D.Sands, G.Sutcliffe and A.Voronkov (eds.), LPAR-21S (Kalpa Publications in Computing, vol. 1), pp. 27–42

Similar to traditional logical systems, concept diagrams are also equipped with inference rules. Thus, they can be used not only for specifying ontologies, but also for reasoning about and evaluating ontologies. Debugging ontologies plays a key role in ontology evaluation [12]. Two main characteristics of an ontology that are often checked during debugging are inconsistency and incoherence. Patterns that give rise to inconsistency and incoherence, i.e., *antipatterns*, capture states of affairs resulting in the creation of unintended model-instances of an ontology [6, 9]. An inconsistent ontology is one that cannot have any model and, so, entails anything [11], whereas an incoherent ontology is one that entails an unsatisfiable (i.e., empty) class or property. In order to derive meaningful conclusions from an ontology, inconsistencies and incoherences must be detected and eliminated, so that the ontology is repaired before publishing.

Concept diagrams have been used for incoherence checking in the past and, in fact, there is empirical evidence [13] that when conducting incoherence checking, novice users perform significantly better using concept diagrams as opposed to using OWL [1] or description logics [3]. The same study also compares user performance in incoherence detection tasks when ontology axioms are translated to concept diagrams one by one, with user performance when the axioms are translated and merged into a single diagram: merging concept diagrams makes it easier for humans to reason about incoherence in ontologies [13]. This result coincides with cognitive evidence [18] that humans often mentally merge the representations corresponding to axioms into one when performing inconsistency and incoherence checking tasks.

In this paper, we address the question of "How can concept diagrams be used to reason about inconsistencies and incoherence of ontologies?". We address this by presenting a set of concept diagrams inference rules that do not lead to one-to-one translations of ontology axioms. Rather, they merge a number of axioms into a concise and cognitively more accessible concept diagram. We base our design of inference rules on empirical evidence that concise (merged) diagrams are easier to comprehend for users than a set of lower level diagrams that express equivalent information [13]. Thus, the central role of our inference rules is to enable stepwise verification of the inconsistency and/or incoherence of a set of ontology axioms. We prove that our concept diagrams inference rules are sound and exemplify how they can be used to spot inconsistency and incoherence. Finally, we indicate how the set of rules that we introduced can serve as a basis for devising new inference rules for representing ontologies in new domains. We argue that concept diagrams are an ontology reasoning tool accessible to diverse non-expert stakeholders.

This paper is organised as follows. In Section 2 we give an overview of the syntax and semantics of concept diagrams, followed by Section 3 that introduces how concept diagrams are reasoned with for inconsistency and incoherence checking of ontologies. In Section 4 we review the related work and finally, we conclude in Section 5.

2 Concept Diagrams

This section presents the syntax and semantics of concept diagrams [19]. We start with an example in Figure 1. This concept diagram has the following syntax and semantic interpretation:

- One dot – called a spider – which represents a named individual, s;
- Two *boundary rectangles* (represented by □) each of which represents the universal set.
- Seven curves, representing seven sets, five of which have labels A to E. The two curves without labels represent *anonymous sets*. The spatial relationships between curves and spiders within a boundary rectangle convey semantics. For examples, the syntax within the LHS rectangle says that the individual s is in the set B; B is a subset of A; the sets A and C are disjoint; and the anonymous set is a subset of C. we discuss arrows).

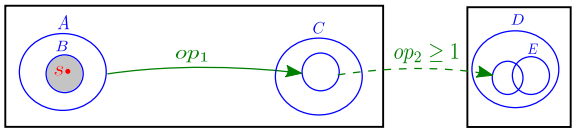


Figure 1: A concept diagram.

- Shading (e.g., inside curve B) which is used to place upper bounds on set cardinality: in a shaded region, all elements are represented by spiders. Thus, the only element in B is s.
- Two arrows, one of which is *solid* and the other one is *dashed* and annotated with ≥ 1. Arrows are used to convey semantics about binary relations, using their sources and targets. The solid arrow asserts that things in A are only related to things in C under op_1. The unlabelled curve, say c_1, targeted by op_1 represents *the* set of things to which elements of A are related under op_1. The dashed arrow is sourced on c_1 and again targets an unlabelled curve. This unlabelled curve, say c_2, represents *a* subset of D to which elements of c_1 are related under op_2. of c_2 inside the curve labelled D expresses that Y is subset of D. The dashed arrow's annotation, ≥ 1, places a constraint on the set c_1: all elements of c_1 must be related to at least one element of c_2 under op_2.

Note also that a solid arrow (e.g., op_1) from a curve to an unnamed subset of another curve represents the well-known ontology axiom of "All Values From". Similarly, a dashed arrow (e.g., op_2) with cardinality ≥ 1 from a curve to an unnamed subset of another curve Next, we give formal accounts of the syntax and semantics of concept diagrams.

2.1 Syntax

When using concept diagrams for ontology representation, ontology classes, individuals in classes, and object properties, are respectively represented by curves, spiders and arrows. These require labels. Therefore, we start by defining three pairwise disjoint sets, $\mathcal{L}_\mathcal{S}$ (for identifying particular individuals), $\mathcal{L}_\mathcal{C}$ (for particular classes), and $\mathcal{L}_\mathcal{A}$ (for object properties), which are, respectively, sets of **names** for spiders, curves and arrows. Informally, in concretely drawn diagrams (as opposed to their sentential abstract representation), spiders and curves are allowed to be unlabelled, as seen in Figure 1. Formally, however, these unlabelled entities act as variables. As such, we define two further pairwise disjoint, countably infinite sets (also disjoint from the former three), $\mathcal{V}_\mathcal{S}$ (for anonymous individuals), $\mathcal{V}_\mathcal{C}$ (for anonymous classes) which are **variables** for spiders and curves respectively. In drawn concept diagram, we typically omit labels for variables to avoid clutter. However, if the same anonymous spider or curve appears more than once, so that the variable label is used on more than one spider or curve, then this label must be drawn. Further, we define $\mathcal{L}_\mathcal{A}{}^- = \{op^- : op \in \mathcal{L}_\mathcal{A}\}$, allowing us to denote inverse properties.

At the abstract level, concept diagrams include a set of spiders that are chosen from a countably infinite set, $\mathcal{S}$. In a drawn diagram, each spider is a tree whose nodes are placed in distinct zones (i.e., dots connected by lines placed in 'minimal' regions in the diagram). Any two spiders may be joined by $=$, to assert that two individuals are the same. For example, if s in Figure 1 was joined to, say s' (i.e., $s_\bullet = {}_\bullet s'$), they would be the same individual. Also, $=$ may be annotated with ? (i.e., $\overset{?}{=}$) to indicate uncertainty about equality: the two spiders may represent either equal or distinct individuals.

Concept diagrams also include closed curves, selected from a countably infinite set, $\mathcal{C}$. The closed curves give rise to zones that are regions inside some or none of the curves. Formally, a zone is a pair of finite, disjoint sets of curves, $(in, C\backslash in)$, where $C \subseteq \mathcal{C}$ is a finite set of curves. Intuitively, $(in, C\backslash in)$ is inside every curve of $in \subseteq C$ and outside every curve of $C\backslash in$. For example, in Figure 1, both LHS and RHS diagram components have five zones.

Arrows are another component of concept diagrams. At the abstract level, arrows are of the form $(s, t, \circ)$ and all of them are labelled. Here, s is the arrows source, t is the target and $\circ$ is either $\rightarrow$ or $\dashrightarrow$. Arrows can be sourced on the boundary rectangle, curves or spiders. Arrow labels can be object properties, or their inverses. Arrows can also be assigned labels that express minimum, maximum and equality cardinality constraints. These labels are written on arrows in diagrams as $\leq n$, $\geq n$ and $= n$; formally they are ordered pairs, such as $(\leq, n)$.

Prior to defining concept diagrams, we define *class and object property diagrams* (Definition 1) that are the main building blocks of concept diagrams, and allow assertions to be made about classes and object properties of an ontology in a universe (i.e., within a boundary rectangle).

Definition 1 (Class and object property diagram)**.** *A* ***class and object property diagram****, $\chi = (S, C, Z, Z^*, \eta, \tau_=, \tau_?, A, \lambda_s, \lambda_c, \lambda_a, \lambda_\#)$ consists of:*

1. *$S \subset \mathcal{S}$ that is a finite set of spiders;*
2. *$C \subset \mathcal{C}$ that is a finite set of curves;*
3. *Z that is a set of zones such that $Z \subseteq \{(in, C\backslash in) : in \subseteq C\}$.*
4. *$Z^* \subseteq Z$ that is a set of shaded zones;*
5. *$\eta : S \rightarrow \mathcal{P}(Z)\backslash\{\varnothing\}$ that is a function that returns the location of each spider. $\mathcal{P}(Z)$ represents the set of subsets (i.e., powerset) of Z, and since the habitat of a spider has to be a non-empty set of zones, we remove $\varnothing$ from $\mathcal{P}(Z)$;*
6. *$\tau_=$ that is a reflexive, symmetric relation on S that identifies whether two spiders are joined by an equals sign; $(s_1, s_2) \in \tau_=$ means that s_1 is joined to s_2 by $=$ (indicating they are the same);*
7. *$\tau_?$ that is a reflexive, symmetric relation on S, disjoint from $\tau_=$. It identifies if two spiders are joined by $\overset{?}{=}$; $(s_1, s_2) \in \tau_?$ means s_1 is joined to s_2 by $\overset{?}{=}$ (indicating that s_1 and s_2 may or may not be equal);*
8. *A that is a finite multiset of arrows such that for all $(s, t, \circ)$ in A, s and t are in $S \cup C \cup \{\square\}$;*
9. *$\lambda_s : S \rightarrow \mathcal{L}_\mathcal{S} \cup \mathcal{V}_\mathcal{S}$ that is a function that maps spiders to spider labels;*
10. *$\lambda_c : C \rightarrow \mathcal{L}_\mathcal{C} \cup \mathcal{V}_\mathcal{C}$ that is a function that maps curves to curve labels;*
11. *$\lambda_a : A \rightarrow \mathcal{L}_\mathcal{A} \cup \mathcal{L}_\mathcal{A}^-$ that is a function that maps arrows to arrow labels or their inverses.*
12. *$\lambda_\# : A \rightarrow (\leq, =, \geq) \times N$ that is a partial function that maps arrows to cardinality constraints.*

We write $S(\chi)$ to denote the set of spiders in χ and so forth for the other sets.

We are now in a position to define concept diagrams. A concept diagram (Definition 2), is a set of class and object property diagrams, possibly with additional arrows that have a source inside one boundary rectangle and a target inside another. Figure 1, op_2 is an arrow that has its source and target in two different rectangles. additional arrows are chosen from $\mathcal{L}_\mathcal{A}$ and they may be annotated with cardinalities. Below is the formal definition of concept diagrams. Item 2 of this definition guarantees that the set of arrows in A_o have their source and targets in two different rectangles (e.g., op_2 in Figure 1). In other words, these arrows are different from the set A in class and object property diagrams, when the source and target of the arrows are within the same rectangle (e.g., op_1 in Figure 1).

Definition 2 (Concept Diagram)**.** *A* ***concept diagram*** *$d = (\mathcal{COP}, A_o, \lambda_o, \lambda_\#)$, has components defined as follows:*

1. $\mathcal{COP}$ *is a finite set of class and object property diagrams such that for any pair of distinct diagrams,* χ_i *and* χ_j*, in* $\mathcal{COP}$*,* $S(\chi_i) \cap S(\chi_j) = \varnothing$ *and* $C(\chi_i) \cap C(\chi_j) = \varnothing$*.*
2. A_o *is a finite multiset of arrows such that for all* $(s, t, \circ)$ *in* A_o*,*
   (a) $s \in \bigcup_{\chi \in \mathcal{COP}} S(\chi) \cup C(\chi) \cup (\{\square\} \times \mathcal{COP})$ *and* $t \in \bigcup_{\chi \in \mathcal{COP}} S(\chi) \cup C(\chi) \cup (\{\square\} \times \mathcal{COP})$*, and*
   (b) *for all diagrams,* χ*, in* $\mathcal{COP}$ *it is not the case that* $s \in S(\chi) \cup C(\chi) \cup \{(\square, \chi)\}$ *and* $t \in S(\chi) \cup C(\chi) \cup \{(\square, \chi)\}$*,*
3. $\lambda_o : A_o \to \mathcal{L}_\mathcal{A} \cup \mathcal{L}_\mathcal{A}^-$ *is a function that maps arrows to object property and their inverses,*
4. $\lambda_\# : A_o \to \{\leq, =, \geq\} \times N$ *is a partial function that maps arrows to cardinality constraints.*

2.2 Semantics

We take a standard approach to defining the semantics of concept diagrams. First, the vocabulary over which the logic is defined is interpreted appropriately (Definition 3), which is the basis for our definition of a model for a concept diagram (Definition 5).

Definition 3 (Interpretation)**.** *An* ***interpretation*** *is a pair,* $I = (U, .^I)$*, where*

- U *is a non-empty set, called the universal set,*
- *for each element* i *in* $\mathcal{L}_\mathcal{S}$*,* i^I *is an element of* U*,*
- *for each element* c *in* $\mathcal{L}_\mathcal{C}$*,* c^I *is a subset of* U*,*
- *for each element* op *in* $\mathcal{L}_\mathcal{A}$*,* op^I *is a binary relation on* U*.*

We also need to interpret the variables and zones in class and object property diagrams. To do so, we first extend interpretations to variables.

Definition 4 (Extended Interpretation)**.** *Given an interpretation,* $I = (U, .^I)$*, an* ***extended interpretation*** *is a pair,* $I' = (U, .^{I'})$*, such that*

- *for each element* x *in* $\mathcal{V}_\mathcal{S}$*,* $x^{I'}$ *is an element of* U*, and*
- *for each element* X *in* $\mathcal{V}_\mathcal{C}$*,* $X^{I'}$ *is a subset of* U*.*

Definition 5 (Model)**.** [1] *Let* $d = (\mathcal{COP}, A_o, \lambda_o, \lambda_\#)$ *be a concept diagram and let* $I = (U, .^I)$ *be an interpretation.* I *is a* ***model*** *for* d *if there exists an extended interpretation,* $I' = (U, .^{I'})$*, such that*

1. *for each class and object property diagram* χ_i*, in* $\mathcal{COP}$
   (a) *the union of the sets represented by the zones in* χ *is* U*, that is* $\bigcup_{z \in Z(\chi)} z^{I'} = U$*, where each zone* $z = (in, C \backslash in)$ *represents the set* $z^{I'} = (\bigcap_{\kappa \in in} \lambda_c(\kappa)^{I'}) \backslash (\bigcup_{\kappa \in C \backslash in} \lambda_c(\kappa)^{I'})$*;*
   (b) *each shaded zone in* χ *represents a set containing only elements mapped to by spiders in* χ*, that is* $z^{I'} \subseteq \bigcup_{\sigma \in S(\chi)} \{\lambda_s(\sigma)^{I'}\}$*;*
   (c) *for each spider,* σ*, in* χ*,* $\lambda_s(\sigma)^{I'}$ *is an element in the set denoted by one of the zones in which* σ *is placed;*
   (d) *any two spiders,* σ_1 *and* σ_2*, in* χ *not joined by* $=$ *map to distinct elements, that is* $\lambda_s(\sigma_1)^{I'} \neq \lambda_s(\sigma_2)^{I'}$*;*
   (e) *any two spiders,* σ_1 *and* σ_2*, in* χ *joined by* $=$ *but not annotated with* ? *map to the same element, that is* $\lambda_s(\sigma_1)^{I'} = \lambda_s(\sigma_2)^{I'}$*;*

[1] For simplicity and succinctness, the definition treats sets rather than a case split for single elements and sets. Thus, we treat single elements as singleton sets. For example, when assigning meaning to arrows in the second part of this definition, a spider represents an element via its label, but we treat this element as a singleton set.

(f) *for each solid arrow, a_j, with source s, target t and label op, the image of op^I when the domain is restricted to the set represented by s, equals to the set represented by t;*

(g) *for each dashed arrow, a_j, with source s, target t and label op, the image of op^I when the domain is restricted to the set represented by s, is a superset of the set represented by t;*

(h) *if an arrow is annotated with cardinality constraint, $(\diamond, n)$, then each element of the source set is related to $\diamond n$ elements in the target set, and*

2. *for each connecting arrow, a_j, in A_o with source in χ_i and target in χ_j,*

(a) *if a_j is a solid arrow with source s, target t and label op, the image of op^I when the domain is restricted to the set represented by s, equals to the set represented by t;*

(b) *if a_j is a dashed arrow with source s, target t and label op, the image of op^I when the domain is restricted to the set represented by s, is a superset of the set represented by t;*

(c) *if a_j is annotated with a cardinality constraint, $(\diamond, n)$, then each element of the source set is related to $\diamond n$ elements in the target set.*

Let $\mathcal{D}$ be a set of concept diagrams. Then I is a model for $\mathcal{D}$ if I is a model for each concept diagram in $\mathcal{D}$.

3 Reasoning with Concept Diagrams

Similar to traditional logical systems, concept diagrams are equipped with inference rules. Reasoning with concept diagrams involves using different kinds of inference rules including first-order logic rules (e.g., substitution [5]), pure diagrammatic rules (e.g., Delete Syntax, see Section 3.1), and rules that combine information from two diagrams. The latter category allows merging concept diagrams. In what follows, we first mention the existing inference rules for concept diagrams. Next, based on existing rules, we introduce inference rules that are tailored for merging concept diagrams and thus support inconsistency/incoherence checking tasks.

3.1 Existing Inference Rules

Here we briefly explain the set of sound inference rules devised before [5]. They are applicable to the fragment of concept diagrams we characterised in Section 2. Figure 2 exemplifies each inference rule. Rules are displayed in two dimensions, the conclusions and premises separated by a line. For instance, $\frac{d_1 \quad d_2}{d_3} R$ shows rule R with premises d_1 and d_2 and conclusion d_3.

- Delete Syntax (*del*): this inference rule removes syntax from a diagram.
- Copy Spider (*c*): this rules copies a spider in a curve from diagram d_1 to d_2, where d_2 contains the same curve with no spider.
- Copy Curve 1 (*cc*1): this inference rule copies curve C from diagram d_2 to d_1, where both diagrams contain curve B, while C is a curve containing B only in d_2.
- Copy Curve 2 (*cc*2): this inference rule copies curve B from diagram d_2 to d_1, where both diagrams contain spider s_2 and s_2 is in curve B in d_2 only.

Apart from *del* that is information weakening, the other three rules are equivalences.

In addition, concept diagrams are a superset of spider diagrams [8], thus we inherit all of the inference rules for spider diagrams (as used in Speedith [20], a diagrammatic reasoner for spider diagrams). Due to space limitation, we refer the readers to [20] for details.

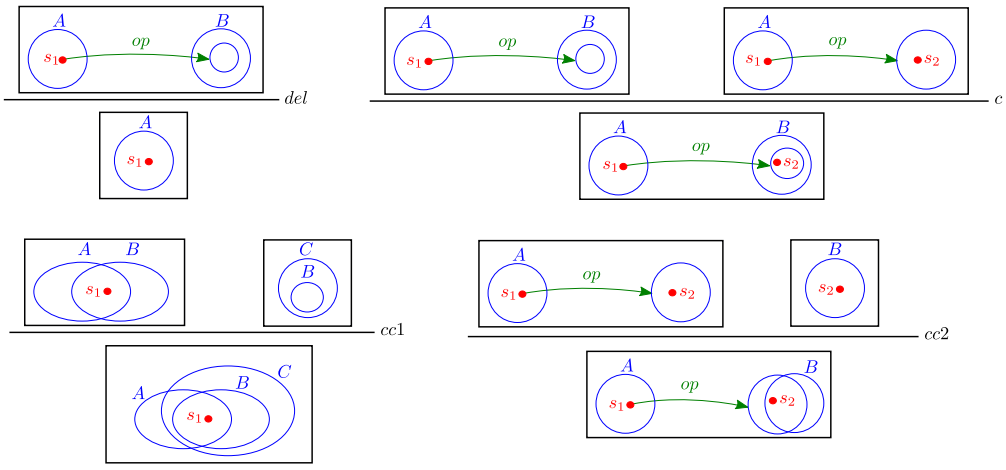


Figure 2: Inference rules from [5].

3.2 Incoherence and Inconsistency

We begin by defining what it means for D, a set of concept diagrams representing axioms in ontology o, to be incoherent and inconsistent [17].

Definition 6 (Incoherence). *A set of concept diagrams, $\mathcal{D}$, is **incoherent**, if one of the following conditions is met:*

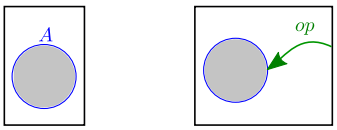
- *there is a label, A, in $\mathcal{L}_\mathcal{C}$ such that for all models, $I = (U, .^I)$, for $\mathcal{D}$ $A^I = \varnothing$,*
- *there is a label, op, in $\mathcal{L}_\mathcal{A}$ such that for all models, $I = (U, .^I)$, for $\mathcal{D}$ $op^I = \varnothing$,*

*Such empty labels are called **unsatisfiable**.*

According to Definition 6, to prove that ontology o is incoherent, we have to show that a class or an object property is unsatisfiable (i.e., empty). When using a set of concept diagrams, $\mathcal{D}$, to define o, the task is thus to prove a lemma of the form:
(i) a curve labelled A necessarily represents an empty class, or
(ii) an arrow labelled op necessarily represents an empty object property.
A lemma of type (i) is proved if, carrying out the proof visually, we derive a diagram in Figure 3a: an entirely shaded region with no spiders represents the empty set in any model. Type (ii) lemmas are proved if the proof derives a diagram in Figure 3b, in which the target of the arrow is entirely shaded with no spiders: this target represents the empty set, implying the image of op is empty, thus op is an empty relation.

For example, both top inference rules in Figure 4 spot an incoherence by showing that A is unsatisfiable. On the left, we have that curve A is disjoint from itself, which is only possible when A is empty. On the right, we have that the the universal image of op is restricted to B, while there is set A such that the partial image of A under op includes C. However, C and B are disjoint. Since the universal image of op is restricted to B, the image of A under p cannot be outside B, which is clearly not the case here. So A is empty. The bottom inference rule shows that object property op is empty, because the first premise displays the image of op as a subset of intersection of B and C, while the second premise defines B and C as disjoint.

We now define what it means for a set of concept diagrams to be inconsistent. Compared to incoherence, inconsistency requires much stronger conditions to be met. As we proceed, we also adopt $\perp$ as a canonical representation of inconsistency.



(a) A is empty. (b) op is empty.

Figure 3: Representation of incoherence in concept diagrams.

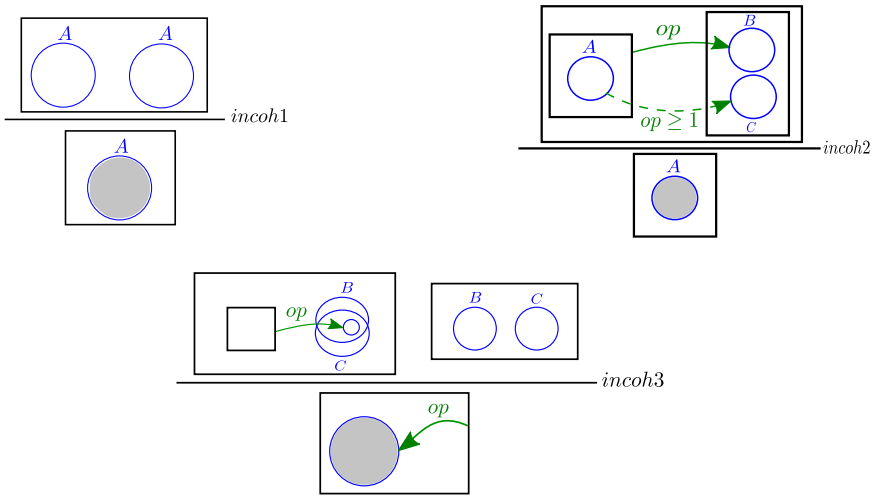


Figure 4: Incoherence examples.

Definition 7 (Inconsistency)**.** *A set of concept diagrams, D, is* ***inconsistent****, if it has no models.*

Diagrams in Figure 5 show antipatterns that lead to inconsistency. In the top left, we state that A and B are disjoint and then we assert that they have at least one element, s_1, in common. Both classes A and B are inconsistent in this case. Inconsistency in the top right figure is caused by class B, since if B is a subset of A it cannot have an element, such as s_1, that is not in A. The bottom rule shows an example of an inconsistent object property, where we first have that the image of A under op is restricted to B, then we have that the image is restricted to C, while B and C are disjoint non-empty sets.

3.3 Using Inference Rules to Detect Incoherence and Inconsistency

Having defined incoherence and inconsistency, we now design inference rules that facilitate their detection. We derive proofs for Lemma 1 and Lemma 2 that show that two sets of axioms are incoherent and inconsistent, respectively. To prove these lemmas we design inference rules that step-by-step take us from the axioms to the goal state in which the lemma is proved. These inference rules are general and can then be used for similar reasoning cases in the same or a different ontology. Our approach to designing inference rules is driven by the requirements of the proof, rather than in isolation from the proof. Therefore, we believe that, in addition to being sound, proof driven inference rules give rise to more natural proofs. In contrast, the established common approach to designing inference rules in logic is primarily driven by the requirements of the theoretical properties (e.g., soundness and completeness) of the rules.

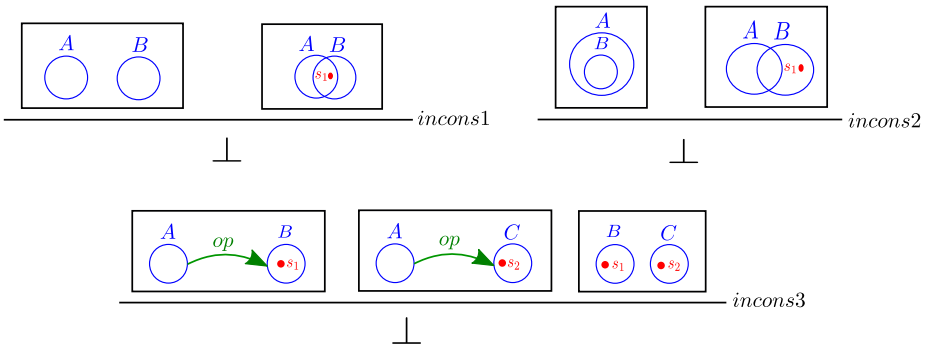


Figure 5: Inconsistency examples.

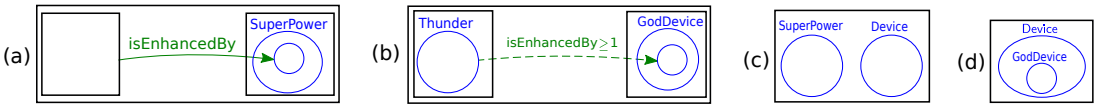


Figure 6: A set of incoherent axioms.

The following lemma shows that the set of axioms in Figure 6 is incoherent.

Lemma 1. *Thunder is empty.*

Proof. Figure 7 shows the proof. □

In this proof (Figure 7), we aim to establish that "Thunder" is empty from axioms in Figure 6. In the first step, Axioms (c) and (d) are merged by using inference rule *cc*. *cc* is one of existing rules for spider diagrams [8] and stands for copy contour (contours in spider diagrams are equivalent to curves in concept diagrams) and is introduced in Speedith [20]. Applying *cc* to diagram d copies the curve labelled "GodDevice" from diagram (d) to diagram (e), giving diagram (e) as conclusion. In the next step (e) and (a) are merged using *cc* twice in a row. "SuperPower" already exists in (a), so the two curves being copied from (e) to (a), are "Device" and "GodDevice". The result of merging (e) and (a) gives (f). Next, (f) and (b) are merged using rule *mrg1*. This rule is formally defined in Definition 8.

Definition 8 (rule *mrg1*)**.** *Let d_1 and d_2 be two concept diagrams, each containing an arrow, a_1 and a_2 respectively, labelled op such that*

1. *a_1 is solid, its source is some boundary rectangle, □, and its target is a curve, c_{1t} in χ_{1t};*
2. *a_2 is dashed or solid and its source and target are curves c_{2s} and c_{2t} in χ_{2s} and χ_{2t}, respectively, while c_{2t} is properly contained in c_3.*

Also let χ_{1t} in d_1 contain curve c_4 disjoint from c_{1t}, such that c_4 has the same label as c_3 in d_2. Now let d_3 be a concept diagram such that d_3 is a copy of d_1 with a new arrow, a_3, and two new curves, c_5, and c_6 as follows:

1. *c_5 is added to the boundary rectangle that is the source of a_1 in d_1,*
2. *c_5 has the same label as c_{2s} in d_2 ,*
3. *c_6 is added inside c_4 from d_1,*
4. *c_6 has the same label as c_{2t} in d_2 ,*
5. *a_3 is sourced on c_5 and targets c_6, and*
6. *a_3 has the same shape (i.e., dashed or solid), label and cardinality as a_2 in d_2.*

Then diagram d_1 can be merged with diagram d_2 to form diagram d_3 using ***rule mrg1****.*

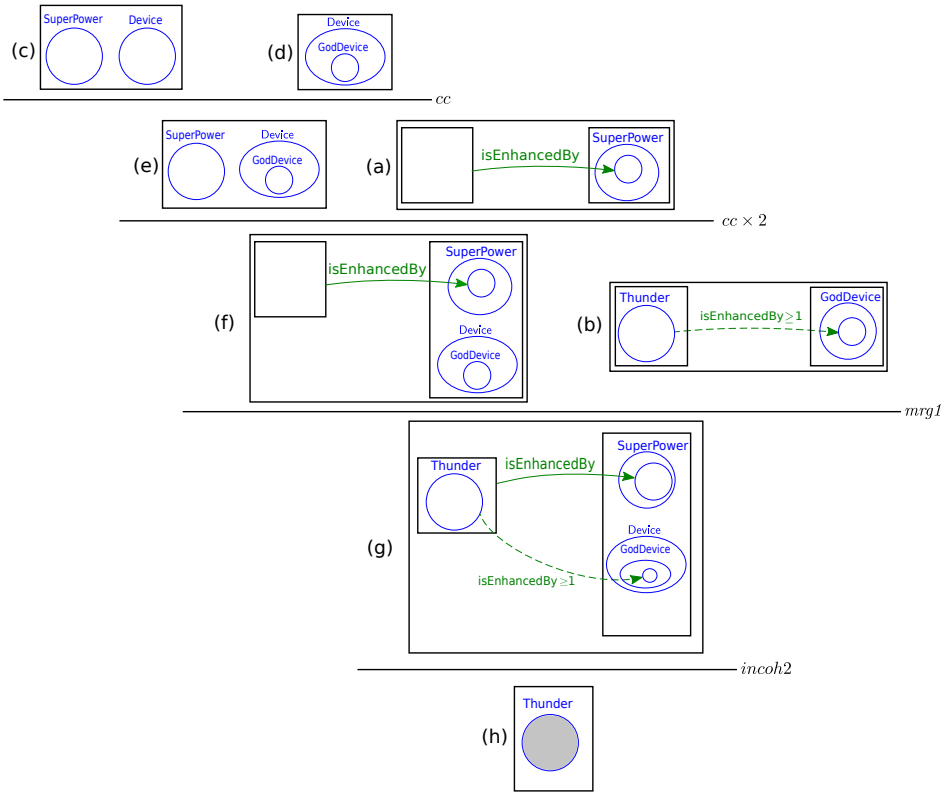


Figure 7: A proof of Lemma 1.

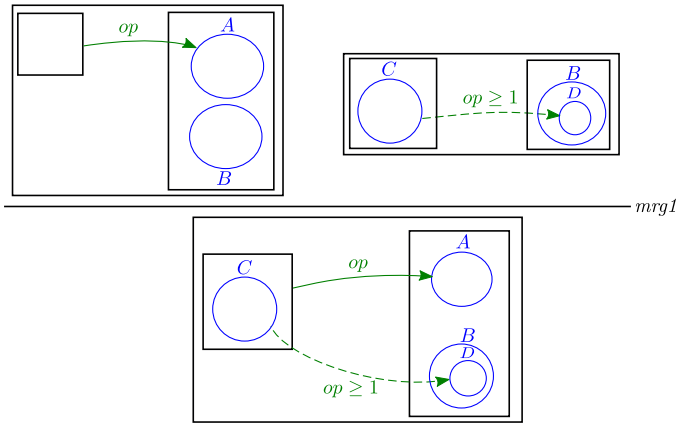


Figure 8: Inference rule $mrg1$.

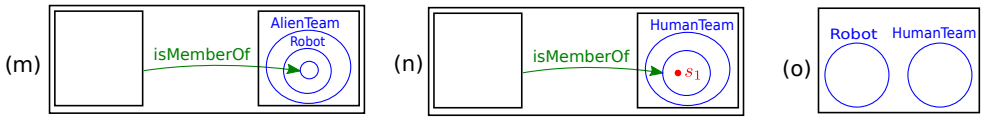


Figure 9: A set of inconsistent axioms.

mrg1 is exemplified in Figure 8, where in the left hand premise c_{1t} and c_4 are represented by curves labelled A and B. On the right, c_{2s}, c_3, and c_{2t} are represented by curves labelled C, B, and D. In the conclusion, c_5,and c_6 are represented by curves C and D.

After merging (f) and (b) and deducing (g), the next inference rule used in Figure 7 is *incoh*2, defined in Definition 9, and exemplified in Figure 4.

Definition 9 (rule *incoh*2)**.** *Let d be a concept diagram with two arrows a_1 and a_2 with the same label op, such that*

1. *a_1 is solid, its source is some boundary rectangle and its target is curve c_{1t} in another boundary rectangle that frames concept and object property diagram χ_t,*
2. *a_2 is dashed or solid, its source is curve c_{2s} whose label is in $\mathcal{L}_\mathcal{C}$ and which is in the boundary rectangle that is the source of a_1, its target is curve c_{2t} in χ_t and it is annotated with $(\diamond, n)$ where $\diamond \in \{\geq, =\}$ and $n \geq 1$,*
3. *c_{2t} and c_{1t} are disjoint.*

Applying ***rule incoh2*** *to d gives d', where d', in its boundary rectangle, contains a single curve c_3 that is all shaded, contains no spiders and has the same label as c_{2s} in d.*

Applying rule *incoh*2 to (g) gives (h), where "Thunder" is fully shaded (i.e., is empty) and therefore the lemma is proved and the set of axioms in Figure 6 is incoherent.

We now prove a lemma that shows that the set of axioms in Figure 9 is inconsistent.

Lemma 2. *Robot is inconsistent.*

Proof. Figure 10 shows the proof. □

In the above proof, we aim to establish that "Robot" is inconsistent, using axioms presented in Figure 9. The first inference rule used in Figure 10 merges Axioms (m) and (n) using *img*5. This inference rule is exemplified in Figure 11 (top left corner). According to this rule if we have two assertions stating the universal image of property *op* as classes c_{1t} and c_{2t}, then they must represent the same set and therefore, all the spiders in one belong to the other one too. The other inference rules in Figure 11, named *img*6, *img*7 and *img*8, are variants of *img*5 with different combination of arrows. We will discuss these variants in Section 3.4. Definition 10 formalises *img*5 and the conditions under which it is applicable.

Definition 10 (rule *img*5)**.** *Let d_1 and d_2 be two concept diagrams, each containing a solid arrow a_1 and a_2, labelled op, such that*

(i) *a_1's source is some boundary rectangle, and its target is curve c_{1t} in χ_{1t}, where c_{1t} is properly contained by some other curve c_3,*

(ii) *a_2's source is some boundary rectangle, and its target is curve c_{2t} in χ_{2t}.*

Let d_3 be a copy of d_2 with an additional curve c_4, such that

(i) *c_4 is added to χ_{2t} and it splits each existing zone into two, one inside and one outside c_4 except the zones inside c_{2t}, which are not split but are all inside c_4, and*

(ii) *the label of c_4 is the same as c_3 in χ_{1t}.*

Diagram d_2 can be merged with curve c_3 from d_1 to form d_3 using ***rule img5****.*

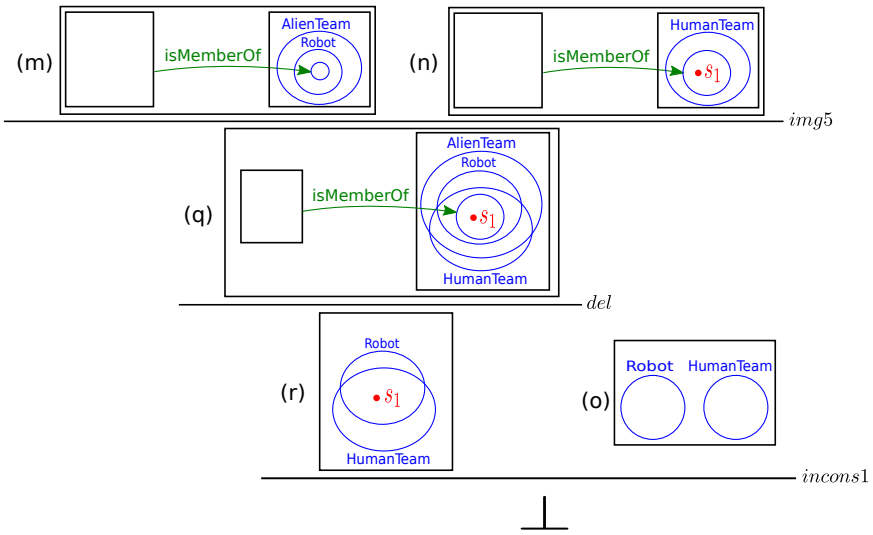


Figure 10: A proof of Lemma 2.

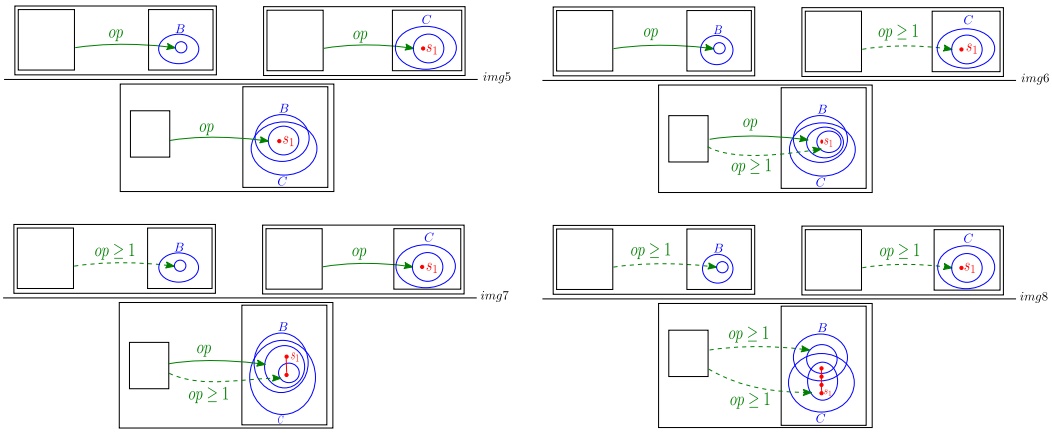


Figure 11: Inference rule *img*5 and its variants.

By applying *img*5 to Axioms (m) and (n), we deduce diagram (q). Next, applying rule *del* to (q) gives (r). *del* is one of the existing rules for concept diagrams that was explained in Section 3.1. Merging (r) with Axiom (o) is done using inference rule *incons*1 (exemplified in Figure 5), that is described in Definition 11.

Definition 11 (rule *incons*1)**.** *Let d_1 and d_2 be two concept diagrams such that there are two curves in d_1, say c_1 and c_2, that are disjoint, while d_2 contains two curves c_3 and c_4 with at least one spider in their intersection. If c_1 and c_2 in d_1 have the same labels as c_3 and c_4 in d_2, respectively,* ***rule incons1*** *spots the inconsistency and concludes false ($\perp$).*

By applying *incons*1 to r and o, we deduce false, and spot the inconsistency of "Robot" and hence the inconsistency of a set of axioms presented in Figure 9.

3.4 Correctness

In order to evaluate the results in this paper, we investigate the soundness and completeness of inference rules introduced. Rules used in proofs of Lemma 1 and Lemma 2 are *cc*, $mrg1$, $incoh2$, $img5$, *del* and $incons1$. *cc* and *del* are proved sound in [5] and [20], respectively. Due to space limitation, below we only prove the soundness of rule $incoh2$. Other rules can be proven sound in a similar manner.

Theorem 3. *Rule* $incoh2$ *(Definition 9) is sound.*

Proof. (Sketch) We need to show that any model for the premise (d), is also a model for the conclusion (d'). Let $I = (U, .^I)$ be an interpretation such that there exists an extension, $I' = (U, .^{I'})$ that shows I is a model for d. Then for the solid arrow a_1, the image of $op^{I'}$ equals $\lambda_c(c_{1t})^{I'}$. Due to the cardinality constraint, $(\diamond, n)$, on a_2, each element of $\lambda_c(c_{2s})^{I'}$ is related to at least one element of $\lambda_c(c_{2t})^{I'}$ under $op^{I'}$. Suppose, for proof by contradiction, that $\lambda_c(c_{2s})^{I'} \neq \varnothing$. Let $e \in \lambda_c(c_{2s})^{I'}$ and let e' be an element in $\lambda_c(c_{2t})^{I'}$ that e is related to under $op^{I'}$. Then e' is an element of $\lambda_c(c_{1t})^{I'}$. So $\lambda_c(c_{1t})^{I'} \cap \lambda_c(c_{2t})^{I'} \neq \varnothing$. On the other hand, since I is a model for d, and c_{1t} and c_{2t} are disjoint in d, we have $\lambda_c(c_{1t})^{I'} \cap \lambda_c(c_{2t})^{I'} = \varnothing$. Therefore the assumption that $\lambda_c(c_{2s})^{I'} \neq \varnothing$ is false. Consider now the only curve, c_3, in d'. Since $\lambda_c(c)$ is a label in $\mathcal{L}_\mathcal{C}$, any extension, I'', of I clearly ensures $\lambda_c(c_3)^{I''} = \varnothing$. Since c_3 is entirely shaded, I satisfies d'. Hence, I is a model for d', as required. □

Ensuring completeness in most logical systems is hard. For a diagrammatic logic for ontologies based on concept diagrams, the difficulty is caused by (i) the existence of several syntactic elements for concept diagrams (e.g., spiders, curves, shading, etc.); and (ii) the constant need to devise new rules to capture the inference required when reasoning about ontologies representing different domains. We conjecture that concept diagrams, as defined here, correspond to a fragment of second-order logic with one and two place predicates. One-place and two-place predicates arise due to the use of labelled curves and arrows respectively. Second-order (existential) quantification occurs through the use of unlabelled curves. Although concept diagrams do not contain quantifiers in their syntax, an equivalent fragment of SOL would need to do so. For instance, two non-overlapping labelled curves, A and B say, give rise to (first-order) universal quantification and express $\forall x\, \neg(A(x) \wedge B(x))$. Due to the restricted way in which second-order quantification arises, finding a complete set of inference rules should be possible. As a step toward completeness, here we explain a systematic approach to extend a set of inference rules for a specific ontology with respect to the syntactic elements of concept diagrams. Exploring the systematic approaches to extend inference rules allows devising a larger and more comprehensive set of rules that are more likely to cover the inference requirements of new ontologies representing other domains.

We explain the extensibility of inference rules by referring to Figures 11 and 12, where we systematically come up with variants for the pattern of rule $img5$ that was directly used in the proof of Lemma 2. Inference rules $img6, img7$ and $img8$ in Figure 11, show the variant of $img5$ for other combination of arrows. We focus on arrows because no previous work has introduced inference rules in presence of both solid and dashed arrows. Since we only have two types of arrows (dashed and solid), the four possible combinations are when (i) they are both solid (as in $img5$), (ii) the first one (i.e. the left hand one) is solid and the second one (i.e. the right hand one) is dashed (as in $img6$); (iii) the first one is dashed and the second one is solid (as in $img7$); and when (iv) they are both dashed (as in $img8$).With respect to arrows, $img5$ can also have variants when both arrows have cardinalities, or none do, or else the first one does, while

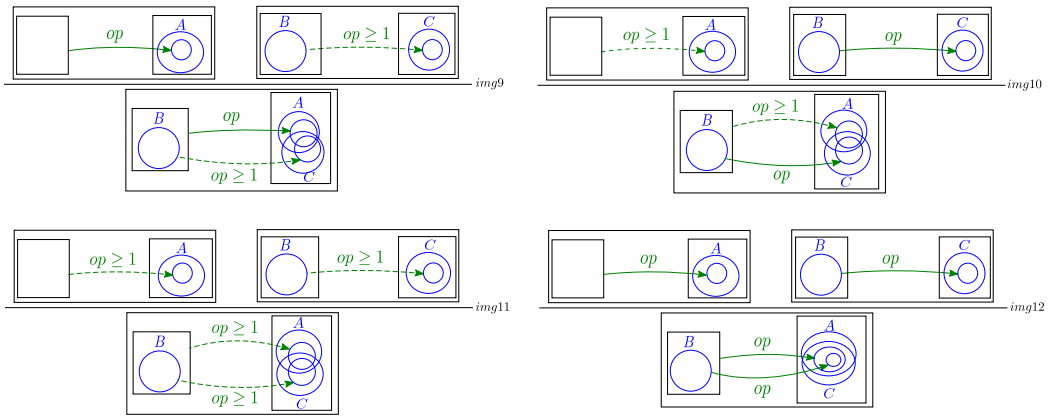


Figure 12: More variants for Inference rule $img5$.

the second one does not. Apart from arrows, $img5$ can have variants with respect to the other syntactic elements such as curves and spiders. For instance, the source of *op* in the premises of each rule can be a specific curve rather than a boundary rectangle. Likewise, the target of the arrows can be a boundary rectangle or a spider instead of a curve. As an example, we provide four more variants for inference rule $img5$ in Figure 12, namely $img9$ to $img12$. These variants assume that the source of *op* in the first premise is a curve rather than a boundary rectangle and also they do not make any assumptions about existence of spiders in curves.

4 Related Work and Discussion

Debugging ontologies is a challenging task. A variety of tools, in particular visualisation tools [15, 16], have been developed to help ontology engineers with the debugging process. Similar to these efforts, concept diagrams attempt to aid debugging ontologies through visualisations. However, we argue that in addition to having cognitive advantages over DL and OWL [13], concept diagrams are cognitively more accessible than ontology visualisation methods based on node-link diagrams (e.g., SOVA [15] and VOWL [16]). This can naturally be explained by referring to better *well-matchedness* [10] of concept diagrams to ontologies. Well-matchedness of a notation is assessed based on how well its syntax and semantics mirror each other. Concept diagrams use syntactic spatial relationships (e.g., curve containment) to reflect the corresponding semantics (e.g., subset relation). In node-link diagrams classes and properties are represented as nodes, while different arrow-like connectors are used to capture the relation between the nodes. In contrast to concept diagrams, node-link diagrams use topological relations to convey semantics (e.g., subset superset relation is expressed using an arrow with a hollow end). Since these diagrams are not well-matched with the semantics, users have to memorise the purpose of multitude of different connectors and arrows. The lack of well-matchedness in tools like [15, 16] suggests that they may not be as cognitively effective as concept diagrams.

In addition to cognitive advantages, concept diagrams are fully formalised, which is not the case for several ontology visualisation tools (e.g., UML diagrams [4]) that are merely for visual modelling. As a formalised logical system, concept diagrams can be used not only as a visualisation tool, but also as a reasoning tool. The reasoning capability of concept diagrams was first highlighted in [5]. The fragment of concept diagrams they use allows quantifiers over elements and subsets of the universe, but unlike the fragment we use, does not include dashed arrows to represent "Some Values From" axioms. In designing our inference rules, we build on

the set of inference rules from [5] that are applicable to the fragment of concept diagrams we use here (i.e., those that are not dealing with quantifiers). Unlike [5], we focus on inference rules for merging and hence for antipattern checking.

Similar to our work, concept diagrams have recently been used for the detection and justification of antipatterns [13]. However, in [13] the focus is on the specification and representation of incoherence using concept diagrams rather than the inference rules and the reasoning mechanism that checks incoherence. In contrast, our goal is to design inference rules for reasoning. Moreover, we use these inference rules for reasoning about both, incoherence and inconsistency.

5 Conclusion and Future Work

In this paper we described how to reason about inconsistency and incoherence in ontologies using concept diagrams. Unlike, many visual tools for ontology engineering [15, 16], concept diagrams are designed to be formal, yet accessible, evidenced by empirical studies [13]. There are two alternatives to use concept diagrams as ontology debugging tool, namely (i) to prove a one to one translation of ontology axioms in concept diagrams; and (ii) to merge ontology axioms in a single concept diagram.We focused on the latter. This choice was informed by existing cognitive empirical evaluations [18, 13] and resulted in proposing a set of inference rules for merging concept diagrams. We showed that the inference rules we introduced are sound, and moreover how they can be used in proofs of inconsistency and incoherence. In addition, we explained how these rules can be extended to cover similar cases.

We conjecture that the fragment of concept diagrams used in this paper (Section 2) is as expressive as second-order logic with one and two place predicates. In the future we will extend the set of inference rules for this fragment with the aim of achieving completeness. Next we will devise inference rules for more expressive versions [5] of concept diagrams that allow variable.

We will use concept diagrams and inference rules presented for them in this paper in building the first mechanised reasoning system for concept diagrams and reasoning about antipatterns in ontology engineering. A very exciting aspect of the future work from this perspective is the empirical studies that we have outlined to inform the intuitiveness of the inference rules we are implementing. We strongly believe that the intuitiveness of the inference rules can significantly contribute to a more accessible reasoner for ontology engineering. In fact, right now we are conducting an empirical study that compares the accessibility and expressiveness of concept diagrams for inconsistency checking in ontologies with OWL [1] and VOWL [16].

Acknowledgments

This research was funded by a Leverhulme Trust Research Project Grant (RPG-2016-082) for the project entitled Accessible Reasoning with Diagrams.

References

[1] The OWL2 web ontology language. https://www.w3.org/TR/owl2-direct-semantics/, Dec. 2016.

[2] Protégé: A free, open-source ontology editor. http://protege.stanford.edu, December 2016.

[3] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In *Handbook on Ontologies*, pages 21–43. Springer, 2009.

[4] Saartje Brockmans, Raphael Volz, Andreas Eberhart, and Peter Löffler. Visual modeling of OWL DL ontologies using UML. In *The Semantic Web 2004: Third International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2004.

[5] Peter Chapman, Gem Stapleton, John Howse, and Ian Oliver. Deriving sound inference rules for concept diagrams. In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2011*, pages 87–94. IEEE, 2011.

[6] Óscar Corcho, Catherine Roussey, Luis Manuel Vilches Blázquez, and Iván Pérez. Pattern-based OWL ontology debugging guidelines. In *Proceedings of the Workshop on Ontology Patterns (WOP 2009)*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[7] Frithjof Dau and Peter W. Eklund. A diagrammatic reasoning system for the description logic *ACL*. *Journal of Visual Languages and Computing*, 19(5):539–573, 2008.

[8] Joseph Gil, John Howse, and Stuart Kent. Formalizing spider diagrams. In *1999 IEEE Symposium on Visual Languages*, pages 130–137. IEEE Computer Society, 1999.

[9] Giancarlo Guizzardi and Tiago Prince Sales. Detection, simulation and elimination of semantic anti-patterns in ontology-driven conceptual models. In *Conceptual Modeling - 33rd International Conference, ER 2014*, pages 363–376. Springer International Publishing, 2014.

[10] Corin A. Gurr. Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *Journal of Visual Languages and Computing*, 10(4):317–342, 1999.

[11] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in OWL ontologies. In *Scalable Uncertainty Management, Third International Conference, SUM 2009*, volume 5785 of *Lecture Notes in Computer Science*, pages 124–137. Springer, 2009.

[12] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[13] Tie Hou, Peter Chapman, and Andrew Blake. Antipattern comprehension: An empirical evaluation. In *Formal Ontology in Information Systems - Proceedings of the 9th International Conference*, volume 283 of *Frontiers in Artificial Intelligence and Applications*, pages 211–224. IOS Press, 2016.

[14] J. Howse, G. Stapleton, K. Taylor, and P. Chapman. Visualizing ontologies: A case study. In *International Semantic Web Conference*, pages 257–272. Springer, 2011.

[15] Nili Itzik and Iris Reinhartz-Berger. SOVA - A tool for semantic and ontological variability analysis. In *Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014 Doctoral Consortium*, volume 1164 of *CEUR Workshop Proceedings*, pages 177–184. CEUR-WS.org, 2014.

[16] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.

[17] Guilin Qi and Anthony Hunter. Measuring incoherence in description logic-based ontologies. In *International Semantic Web Conference*, volume 4825 of *Lecture Notes in Computer Science*, pages 381–394. Springer, 2007.

[18] Marco Ragni, Sangeet Khemlani, and P. N. Johnson-Laird. The evaluation of the consistency of quantified assertions. *Memory & Cognition*, 42(1):53–66, 2014.

[19] Gem Stapleton, John Howse, Peter Chapman, Aidan Delaney, Jim Burton, and Ian Oliver. Formalizing Concept Diagrams. In *Visual Languages and Computing*, pages 182–187. Knowledge Systems Institute, 2013.

[20] Matej Urbas, Mateja Jamnik, and Gem Stapleton. Speedith: A reasoner for spider diagrams. *Journal of Logic, Language and Information*, 24(4):487–540, 2015.