A Proposed Diagrammatic Logic for Ontology Specification and Visualization

Ian Oliver
Nokia Research
ian.oliver@nokia.com

John Howse
University of Brighton
john.howse@bton.ac.uk

Gem Stapleton
University of Brighton
g.e.stapleton@bton.ac.uk

Esko Nuutila
Helsinki University of Technology
esko.nuutila@hut.fi

Seppo Törmä
Helsinki University of Technology
seppo.torma@hut.fi

ABSTRACT

We propose a diagrammatic logic that is suitable for specifying ontologies. We provide a specification of a simple ontology and include examples to show how to place constraints on ontology specifications and define queries. The framework also allows the depiction of instances, multiple ontologies to be related, and reasoning about ontologies.

1. INTRODUCTION

The need to specify ontologies frequently arises and specifications can be provided symbolically but many people find such notations inaccessible. Ontology construction and conceptualization can be difficult, hindered by the inaccessibility of the symbolic syntax available to the user; diagrammatic notations are potentially a viable alternative. In The Description Logic Handbook [1], Nardi and Brachman state that a "major alternative for increasing the usability of description logics as a modeling language" is to "implement interfaces where the user can specify the representation structures through graphical operations."

Some diagrammatic notations have been used for representing ontologies, but typically they are not formalized. For instance, Brockmans et.al. [2] investigate using the UML for specifying ontologies but it was not designed for this (some regard the UML as semi-formal and it has not been fully formalized). Diagrammatic notations should be designed for specific tasks: what constitutes an effective diagram is task dependent (for example, work by Grawemeyer [5]). Moreover, it is often emphasized that reasoning is a distinguishing feature of description logics, and current graphical representations for ontology specification do not support this aspect, with a few exceptions including existential graphs [3], which only incorporate diagrammatic versions of the $\exists$ quantifier and the $\wedge$ and $\neg$ operators, making it hard to use.

We propose a diagrammatic notation for specifying and reasoning about ontologies, called *ontology diagrams*. The notation modifies and extends constraint diagrams, making them suitable for ontology specification; constraint diagrams have been fully formalized [4] and fragments of them are sound and complete [8]. The design of ontology diagrams has been derived by considering the semantic concepts that one needs to define syntactically and the tasks one needs to perform using the syntax. This design process allows one to align the syntax with the semantics and the tasks which is likely to achieve a more usable notation than one would obtain by matching the (graphical) syntax with a symbolic notation (e.g. a particular description logic). The alignment of the syntax of constraint diagrams with their semantics is discussed in [7], which extends to ontology diagrams. The design process has been strongly informed by cognitive theories of what constitutes an effective diagram, for example [6].

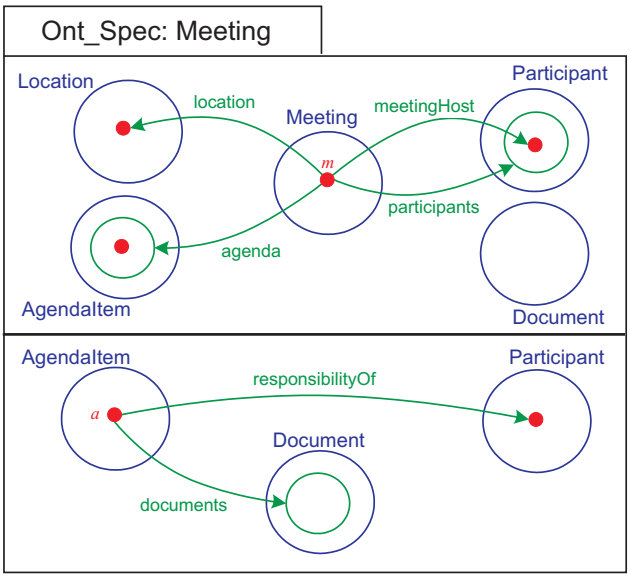

Figure 1: An ontology diagram specifying Meetings.

2. ONTOLOGY DIAGRAMS

To illustrate our proposed ontology diagrams we present a simple case study. This ontology includes various classes such as Location and Participant. The ontology diagram in figure 1 consists of 2 sub-diagrams and captures relationships between the classes, represented by labelled curves. In each sub-diagram, the curves form an Euler diagram and their spatial relationships express semantic relationships between the classes: non-overlapping curves assert that the classes are disjoint; a curve placed inside another asserts a subset relationship. We can immediately see from the top sub-diagram that all the classes represented are pairwise disjoint since all the labelled curves have pairwise disjoint interiors; given that there are 5 classes, this would require ${}^5C_2 =$

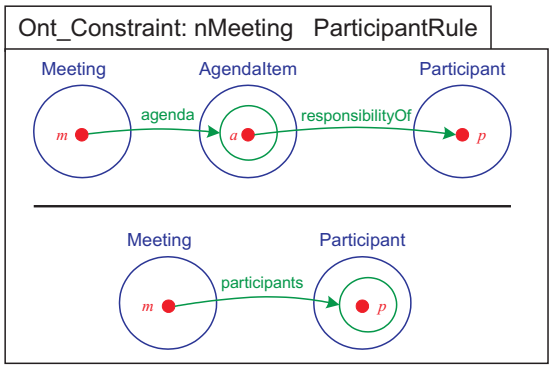

Figure 2: The constraint rule schema.

10 textual assertions (such as Meeting ⊓ Document ⊑ ⊥). The unlabelled dots (called unlabelled spiders) assert the existence of elements in the sets represented by the regions of the diagram in which they are placed. The labelled spiders in this diagram are acting as free variables. So, m is a free variable that is 'talking' about meetings.

Arrows make statements about binary relationships between classes; their labels are analogous to roles in description logics. In the context of an ontology diagram that is specifying an ontology (like that in figure 1), the arrows are interpreted as providing domain and codomain information. In our Meetings example, the agenda arrow expresses that there is a relation (or role) called agenda between Meeting and AgendaItem. Since this arrow's target is an unlabelled curve that contains an unlabelled spider, we have asserted that each meeting (using the free variable m) has a non-empty set of agenda items; the unlabelled curve represents the image of the relation agenda when the domain is restricted to m. This arrow, its source and target (including the spider inside the targeted curve), tells us:

$$\exists\,\mathsf{agenda}.\top \sqsubseteq \mathsf{Meeting}$$
$$\exists\,\mathsf{agenda}.\neg\mathsf{AgendaItem} \sqsubseteq \bot$$
$$\mathsf{Meeting} \equiv \mathsf{Meeting} \sqcap \exists\,\mathsf{agenda}.\mathsf{AgendaItem}$$

The location arrow targets an unlabelled spider and tells us that each Meeting has exactly one Location; thus, location is a function with domain Meeting and codomain Location. We observe that the ontology diagram specifies that meetingHost is a subrole of participants, since the host of the meeting must be one of the meeting participants; equivalently meetingHost ⊑ participants.

We probably want to assert that if a Participant is responsible for an AgendaItem then that Participant is one of the Meeting participants; this is not deducible from figure 1. To allow this type of constraint to be imposed on ontologies, we introduce a constraint rule schema, illustrated in figure. 2; it follows the style of a natural deduction rule. The diagram asserts that if m is a Member, a is an AgendaItem on the agenda of m, and a is the responsibilityOf Participant p (above the line) then the set of m's participants includes p (below the line); note that the Meeting and Participant curves are redundant from the diagram below the line, since the type of m and p is specified above the line.

It is useful to be able to specify queries over ontologies. A query, called getDocuments, is defined in figure 3; the notation getDocuments(m) → D provides the name of the query, the required inputs (in this case, m), and the output (in this case D). Here, m is acting as a constant and is

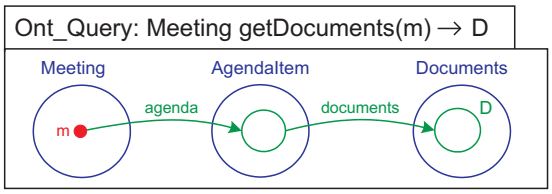

Figure 3: Query.

differentiated from a free variable since it is not italicized. The source of the documents arrow is an unlabelled curve. This curve represents the set of agenda items associated with the meeting m; more formally, it represents the image, I, of the relation agenda when its domain is restricted to m. Continuing in this manner, the set D is the image of the relation documents when the domain is restricted to I; the query returns the set D.

3. CONCLUSIONS

We have proposed ontology diagrams as a means to specify and visualize ontologies. Due to space limitations, not all features of the notation have been demonstrated, including the ability to visualize instances, merge/relate two or more ontologies, and to reason about the ontologies (sound and complete inference rules exist for constraint diagrams). The poster itself will include examples illustrating these features and show how other logical operators, such as disjunction and negation, are represented. There are many directions for future research. First, we need to formalize the syntax and semantics of ontology diagrams, building on the substantial body of work for constraint diagrams, and to define inference rules for them. Once formalized, we need to establish the precise relationship with description logics, (e.g. in terms of expressiveness). Moreover, we will build on recent advances made in diagram layout and reasoning tools to build software to support their use.

Acknowledgements Oliver, Nuutila and Törmä are supported by the TEKES ICT SHOK DIEM project. Howse and Stapleton are supported by the EPSRC Visualization with Euler diagrams project (EP/E011160/1).

4. REFERENCES

[1] F. Baader et al. (eds). *The DL Handbook*. CUP, 2003.
[2] S. Brockmams, R. Volz, A. Eberhart, P. Löffler. Visual modeling of OWL DL ontologies using UML. *Int. Semantic Web Conference*, pp. 198–213. Springer, 2004.
[3] F. Dau, P. Ekland. A diagrammatic reasoning system for the description logic $\mathcal{ALC}$. *Journal of Visual Languages and Computing*, 19(5):539–573, 2008.
[4] A. Fish, J. Flower, J. Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.
[5] B. Grawemeyer. Evaluation of erst – an external representation selection tutor. *Diagrams*, pp. 154–167. Springer, 2006.
[6] A. Shimojima. Inferential and expressive capacities of graphical representations: Survey and some generalizations. In *Diagrams*, pp. 18–21. Springer, 2004.
[7] G. Stapleton, A. Delaney. Evaluating and Generalizing Constraint Diagrams. *Journal of Visual Languages and Computing*, 19(4):499-521, 2008.
[8] G. Stapleton, J. Howse, J. Taylor. A decidable constraint diagram reasoning system. *Journal of Logic and Computation*, 15(6):975–1008, 2005.