A Vision for Diagrammatic Ontology Engineering

Gem Stapleton[1], John Howse[1], Adrienne Bonnington[2], and Jim Burton[1]

[1] University of Brighton, UK,
{g.e.stapleton,john.howse,j.burton}@brighton.ac.uk,
[2] Horizons Regional Council, NZ,
adrienne.bonnington@horizons.govt.nz

Abstract. Ontology engineering is becoming more important, given the rapid rise of data in this information age. To better understand and reason about this data, ontologies provide us with insight into its structure. With this comes the involvement of a wide range of stakeholders, such as information analysts, software engineers, lawyers, and domain experts, alongside specialist ontology engineers. These specialists are likely to be adept at using existing approaches to ontology development, typically description logic or one of its various stylized forms. However, not all stakeholders can readily access such notations, which often have a very mathematical style. Many stakeholders, even including fluent ontology engineers, could benefit from visual approaches to ontology engineering, provided those approaches are accessible. This paper presents ongoing research into a diagrammatic approach to ontology engineering, outlining key research advances that are required.

Keywords: ontology engineering, diagrams, visualization

1 Introduction

Ontologies are used as structural frameworks for organizing information and are a form of knowledge representation. Ontology engineering - the process of producing ontologies - is becoming increasingly important and ubiquitous in society as a way of understanding the vast deluge of data that now exists in this information age. Notable examples include the semantic web, medicine (including SAPPHIRE for public health, SNOMED-CT for healthcare) and bioinformatics (e.g. the gene ontology GenNav). Indeed, there are various large repositories of freely available ontologies such as the Manchester OWL Corpus which contains over 4500 ontologies including, between them, over 3 million axioms [4], the Swoogle repository contains over 10000 ontologies [7] and BioPortal contains nearly 400 ontologies from the biomedical domain [3].

Ontology engineering requires the involvement of domain experts, who need to be able to communicate domain knowledge to ontology engineers. In turn, ontology engineers need to verify ontological choices with the domain experts and amend or refine the ontology based on feedback. Thus, the process of producing ontologies is very much bidirectional and depends on cross-disciplinary

communication. Barriers to communication can lead to errors, compounding an already difficult task.

Description logics and OWL are the commonly used (symbolic) notations for ontology engineering, and the latter is often seen as a stylized form of the former; the W3C's most recent specification of the web ontology language, called OWL 2.0 [5] is a decidable description logic [9]. Description logics promote a hierarchical taxonomy of concepts (called classes in OWL) as a primary modelling feature. Individuals can be members of concepts. Binary relations over concepts, called roles in description logic and properties in OWL, are used to relate individuals and concepts. In particular, individuals, concepts and roles form the vocabulary over which the axioms that form the ontology are defined. The term description logic actually refers to a family of logics, with the simplest such logic of significance being called $\mathcal{ALC}$ and perhaps the most practically used being called $\mathcal{SHOIN}^{(\mathcal{D})}$. The latter description logic is supported in the prominent ontology engineering tool called Protégé [6], which has a strong community of users including academics, governments and corporations [6].

As well as Protégé, other software tools have been implemented to support the creation of symbolically specified ontologies. However, when people meet to develop conceptual structures, including models of knowledge intended to become ontologies, they sometimes quickly move to sketching images to communicate their thoughts; see Howse et al. [16]. The inaccessibility of the DL family of symbolic notations is recognised by Rector et al. [18]: "Understanding the logical meaning of any description logic or similar formalism is difficult for most people, and OWL-DL is no exception."

Warren et al. back this insight up with an empirical study of the most commonly used OWL constructs, including class subsumption, disjointness and equivalence alongside All Values From and other role restrictions [24]. They found that "despite training, users are prone to certain misconceptions" and they observed that "one-third of [participants] commented on the value of drawing a diagram ... In the context of [description logics], diagrams offer a strategy to overcome misconceptions and generally support reasoning." Warren et al. further recognise that existing visualizations are "chiefly aimed at viewing the structure of the overall ontology or parts of the ontology rather than the more cognitively difficult features of Description Logics" and they identify that concept diagrams [16] are the exception. A major aim of our research is to address this identified shortcoming of symbolic notations by providing a diagrammatic alternative based on concept diagrams.

This paper presents an overview of the challenges that must be addressed in order to deliver a practical framework for using concept diagrams for ontology engineering. Key challenges include

1. devising a novel pattern-driven diagrammatic method for ontology engineering,
2. producing a heterogeneous logic that allows diagrams and symbols to be used in combination, and
3. developing techniques for automatically visualizing symbolically ontologies.

Throughout this paper we discuss the advances necessary to address these challenges. To begin, we present a brief overview of the state-of-the-art in ontology visualization research. We then discuss the design of concept diagrams, showing how they reflect known theories of what constitutes a good diagram. This motivates them as a suitable choice of notation for approaching the challenges set out above. We recognise the need for extensive empirical evaluations to ensure that concept diagrams are an accessible alternative to notations such as description logic, but for space reasons we do not discuss this further.

Our ambition is not only to visualize ontologies, but to enable them to be engineered and maintained visually, placing diagrams on an equal footing with traditional approaches. Throughout the paper, we illustrate key points using a real world ontology concerning cemeteries and burials. Ontology engineers interested in using concept diagrams are able to download a free practical user guide from http://www.ontologyengineering.org, under downloads.

2 State-of-the-Art Ontology Visualization Research

Ontology visualization improves access to information by ontology engineers, domain experts and other end-users. Benefits of visualization include revealing information that could be somewhat hidden, or unapparent, when using traditional symbolic notations. Such benefits have long been recognised in the related area of software engineering, as has the need to ensure that software models (akin to ontologies) are accessible to as wide a range of stakeholders as possible [8]. Katifori et al. survey the state-of-the-art in ontology visualization techniques up to 2007 [17], with examples including OWLViz [2], OntoGraf [1], and CMap Tools [14]. Like OWLViz, OntoGraf exploits node-link diagrams (also called graphs) to visualize subsumption relationships between concepts, but does not depict disjointness relationships. There is potential, therefore, for confusion to arise because of the partial information given. CMap Tools also exploit node-link diagrams and, as with OntoGraf, uses directed edges (arrows) to represent both subsumption relationships and role restrictions. Consequently, the saliency of these two different types of information is significantly reduced.

An example produced using CMAP Tools can be seen in figure 1. The fact that each Deceased is also a Person is asserted by the arrow labelled is a between Deceased and Person. The fact that each Deceased has a memorial of some sort is asserted by the use of the same syntactic device: the arrow labelled hasMemorial between Deceased and Memorial. CMaps are also capable of depicting disjointness information via arrows, though none is shown in figure 1. Thus, node-link-based visualizations such as CMaps and OntoGraf use arrows for role restrictions, subsumption and (where these can be depicted) disjointness relationships.

Similarity theory tells us that saliency is an important visual factor and, in particular, that different syntactic devices should represent different types of information [11]. This is because when visually searching for particular types of information, increasing degrees of similarity between the *target* syntax (rep-

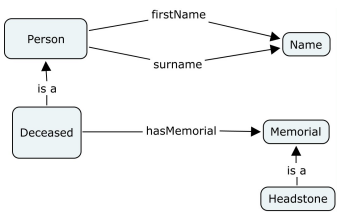

Fig. 1. A CMap visualization.

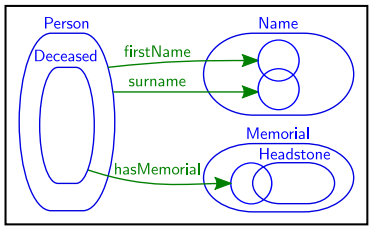

Fig. 2. A concept diagram.

resenting the required information) and *distracter* syntax (representing other information) leads to an increase in the time taken to perform tasks.

Concept diagrams aid information saliency by avoiding the use of identical syntactic types for different informational types: dots (or, in general, node-link diagrams), closed curves, and arrows represent individuals, concepts and roles respectively. An example concept diagram, expressing similar information to the CMap visualization, is in figure 2. The placement of (the curve labelled) Deceased inside Person expresses that Deceased is subsumed by Person. Since Person and Memorial do not overlap, the diagram also expresses that these two concepts are disjoint (have no individuals in common). The arrow labelled firstName, sourced on Person and targeting the unlabelled curve inside Name asserts that people's first names are all names (in DL, this is an All Values From restriction: $\mathsf{Person} \sqsubseteq \forall \mathsf{firstName}.\mathsf{Name}$). Here, the unlabelled curve represents an anonymous concept containing precisely the names that are people's first names. The other two arrows are interpreted in the same way.

One visualization technique which was developed specifically as a diagrammatic logic equivalent to the simple description logic $\mathcal{ALC}$ is a variation on existential graphs by Dau and Eklund [10]. An example of an existential graph is given in figure 3 that expresses the All Values From restriction $\mathsf{Person} \sqsubseteq \forall \mathsf{firstName}.\mathsf{Name}$. The existential graph literally translates to 'it is not the case that there is a person who has a first name that is not a Name'; curves represent negation, lines represent anonymous individuals and the written words correspond to concepts and roles. Existential graphs are not readily accessible since their syntax is somewhat restrictive: they have the flavour of a minimal first-order logic with only existential quantification, negation and conjunction.

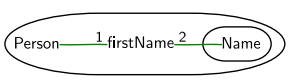

Fig. 3. An existential graph.

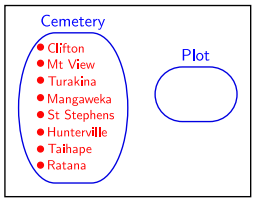

Fig. 4. Representing individuals.

3 The Design of Concept Diagrams

The design of concept diagrams has been informed by theories as to what constitutes an effective visualization. We have already seen that similarity theory leads us to use different syntactic items for different kinds of constructions. This is consistent with the Gestalt principle of *good form*, which tells us that there is a tendency for people to group together graphical objects that share some property, such as colour or shape. In concept diagrams, individuals are visualized using nodes, concepts by closed curves and role restrictions by arrows. We have already seen the use of curves and arrows; figure 4 shows how individuals are represented. This diagram asserts that Clifton is a Cemetery but not a Plot, with the same being true of the other seven individuals. Using different types of graphical objects ensures the same type of syntax is used to represent the same type of semantic property. Further, we use shape to partition the curves into two distinct sets: those which represent named concepts and, respectively, unnamed concepts are represented using rounded rectangles and, respectively, circles.

Similarly, graphical objects that have different properties will typically be considered as being in different groups by people. This suggests that different semantic constructs should be represented by different graphical objects, consistent with similarity theory. Colour could also be used to good effect, if we want to give visual clues about some semantic commonality or distinctness. We have adopted the convention, when colouring concept diagrams, that nodes, curves, and arrows are assigned different colours (in this paper, red, blue and green respectively). Rectangles, which identify diagram boundaries, are all black.

The way in which concept diagrams use this syntax to make statements has been carefully considered. In particular, the spatial relationship between curves reflects the semantic relationships that they convey: concept subsumption, disjointness and intersection are represented by enclosure, disjointness and overlap of curves respectively. Likewise, the location of nodes either inside or outside of curves corresponds to the individuals represented being either members of, or not members of respectively, the corresponding concepts. Properties are directional relationships which are captured by arrows, indicating direction. These types of features, embodied in concept diagrams' design, are known as *well-matched* [13].

An important feature of concept diagrams is that they are fully formalized [22]. As is standard with visual languages [12, 15], concept diagrams are formalized using an abstract syntax (sometimes called type syntax). Their semantics are defined using a standard model-theoretic approach, akin to that used for description logics. Due to space reasons, we refer the reader to [22] for the full details on the formalization of concept diagrams.

4 Patterns

We believe that concept diagrams are able to *readily* express commonly occurring symbolic axioms, for which we have devised a set of simple diagrammatic patterns [23]. Such an approach is thought to aid building ontologies (i.e. defining

the axioms), since the patterns will, essentially, be standard templates for commonly occurring constraints. However, using a separate diagram for each piece of information does not allow the exploitation of many important diagrammatic features. In particular, single diagrams that express rich information can do so in an accessible way and such diagrams can correspond to many symbolic axioms (discussed further in section 5). To this end, we demonstrate how simple diagrammatic patterns can be merged to produce informationally rich diagrams.

4.1 Simple Patterns

A series of simple patterns was presented in [23], expressing concept subsumption and disjointness as well as All Values From and Some Values From role restrictions. Here, we illustrate eight of these simple patterns.

Figures 5 and 6 show instances of patterns for asserting that individuals are and, respectively, are not members of concepts. In particular, figure 5 asserts that M (short for 'male') is a Gender whereas figure 6 asserts that M is not an Occupation. Figures 7 and 8 are instances of the concept subsumption and concept disjointness patterns. The former figure tells us that Deceased is subsumed by Person whereas the latter expresses that Person and Gender are disjoint.

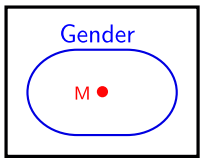

Fig. 5. Inclusion.

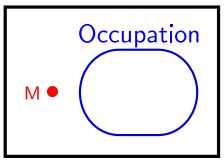

Fig. 6. Exclusion.

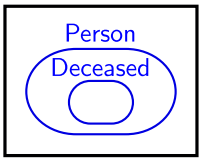

Fig. 7. Subsumption.

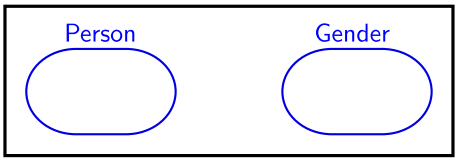

Fig. 8. Disjointness.

Turning our attention to roles, figure 9 diagrammatically expresses an All Values From restriction. In particular, it tells us that under the role gender, Person has all values from the concept Gender. This concept diagram illustrates the use of multiple boundary rectangles. The spatial relationships between syntactic items are only of semantic importance within innermost boundary rectangles. In this case, the two innermost rectangles mean that the diagram does *not* tell us that Person and Gender are disjoint (even though this happens to be true, given the information in figure 8).

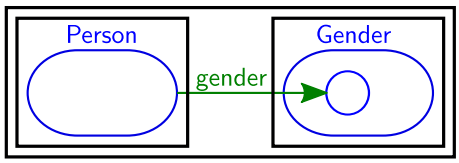

Fig. 9. All Values From.

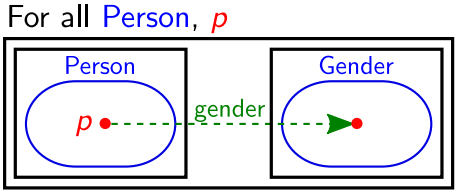

Fig. 10. Some Values From.

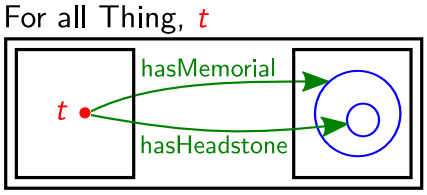

Fig. 11. Role subsumption.

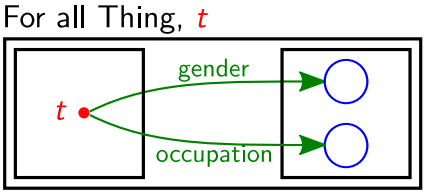

Fig. 12. Role disjointness.

The Some Values From pattern used in figure 10 (which is different from the Some Values From pattern in [23]) illustrates two further features of concept diagrams: dashed arrows and the use of quantifiers to talk about anonymous individuals. This concept diagram makes an assertion about all individuals that are people, using the *quantification expression* For all Person, p. The *dashed* arrow targets an anonymous individual that is a Gender. This dashed arrow tells us that the individual, p, is related to *at least* the individual at the target. That is, the person p has at least one gender.

Patterns for role subsumption and role disjointness are instantiated in figures 11 and 12 respectively. As with concept subsumption and disjointness, they also exploit the topological properties of containment and disjointness to express the required information. For example, figure 11 tells us that, for each thing t, the set of t's headstones in subsumed by the set of t's memorials.

4.2 Merging Patterns

Using one diagram to express a small amount of information may be helpful when developing ontologies, but sometimes having diagrams containing rich information can give a better overview of how concepts and roles interact. We plan to develop methods of merging simple patterns together in order to produce richer diagrams. Such a merging process is illustrated in figures 13 to 15. Firstly, figure 8 tells us that Person and Gender are disjoint, meaning that we can delete the two innermost rectangles from figure 9. The result is in figure 13, which expresses the same information as the two original patterns. The other diagrams in figures 14 and 15 are similarly obtained; here the graph labelled M indicates that M is a Gender where the edge represents disjunction.

Using merging techniques, semantically rich diagrams that display significant proportions of ontologies can be produced. An example is given in figure 16,

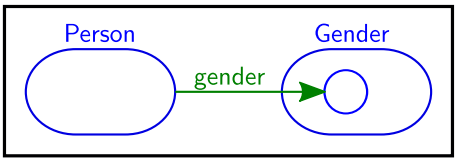

Fig. 13. Merging figures 8 and 9.

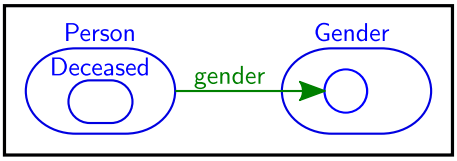

Fig. 14. Merging figures 13 and 7

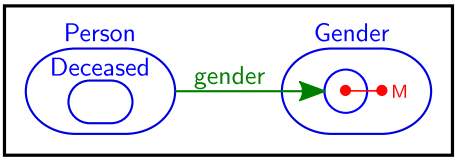

Fig. 15. Merging figures 14 and 5

which depicts a large fragment of a cemetery ontology. This diagram depicts 13 *inclusion* axioms, four *subsumption* axioms, 709 (binary) *disjointness* axioms and 46 *all values from* axioms. It is an interesting avenue of future work to determine which patterns can be merged, and how. For example, in figure 16 only axioms whose patterns do not include explicit quantification have been merged. A key aspect of this work will be to devise heuristics that identify diagrams that can be merged to produce effective visualizations. The fact that figure 16 depicts 772 axioms in a single readable diagram indicates that the notation scales to some reasonable extent; of course, screen real estate presents a scalability problem for both visual and symbolic notations.

5 Heterogeneous Ontology Engineering

Unlike many stakeholders, expert ontology engineers are fluent in the use of symbolic logics. The framework for diagrammatic ontology engineering that we envisage will allow ontologies to be engineered symbolically and diagrammatically and, as a by-product, allow the visualization of already developed symbolic ontologies. A major challenge is to allow diagrams and symbols to be used in tandem by providing a seamless integration of the two paradigms. This requires the two 'views' (diagrammatic and symbolic) of the ontology to be kept in sync, so any update made to the symbolic axioms is reflected diagrammatically and

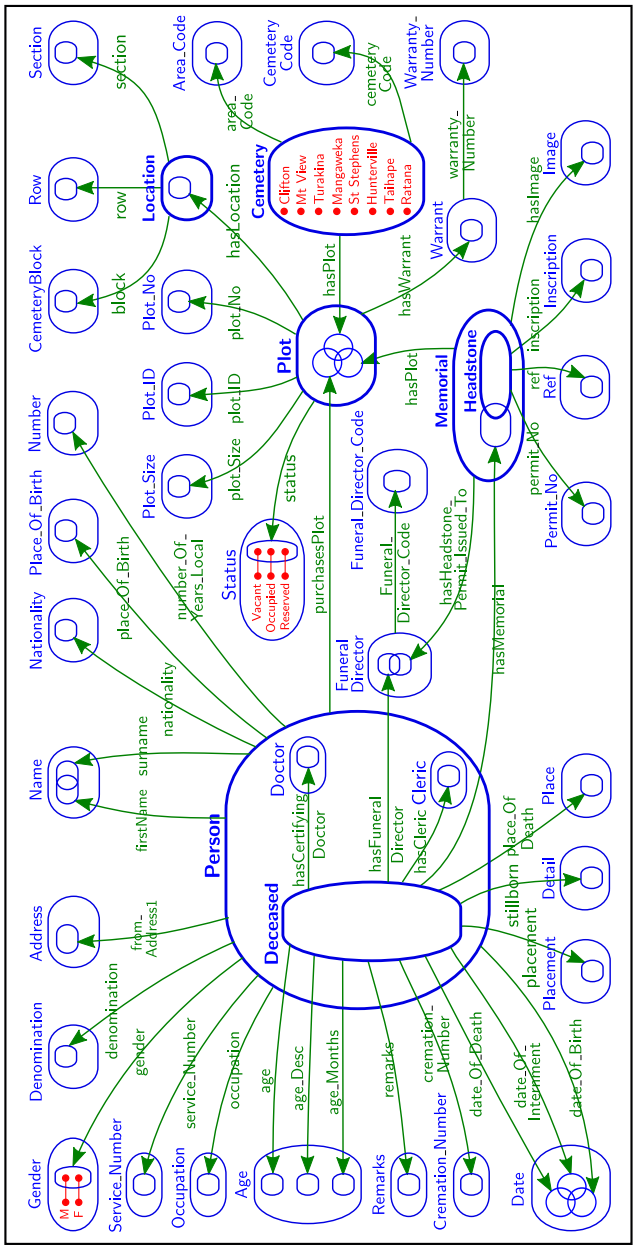

Fig. 16. A large fragment of the cemetery ontology.

vice versa; this idea of two views is illustrated using figure 15 and the DL axioms: $\mathsf{Person} \sqsubseteq \forall \mathsf{gender}.\mathsf{Gender}$, $\mathsf{Person} \sqcap \mathsf{Gender} \sqsubseteq \bot$, $\mathsf{Deceased} \sqsubseteq \mathsf{Person}$, and $\mathsf{Gender}(\mathsf{M})$.

Core to solving this challenge is the derivation of effective translations between diagrammatic axioms and symbolic axioms. It is expected that the patterns described above, with the merging results, will form a basis for translating symbolically-specified ontologies into concept diagrams. However, significant advances are required, using patterns as a starting point. Requirements of the translation methods are likely to include, but not be limited to: visualize the entire ontology; visualize all concepts; visualize all axioms involving a particular concept or set of concepts; and visualize all axioms involving a particular role or set of roles.

Further, we expect to devise translations that permit multiple levels of visualization: top-level overviews, some means of exploring the overview including zooming and filtering, and drilling down to low-level detail as required. Lastly, when symbolic axioms are altered, as will often happen during the refinement and debugging phases of ontology development, the corresponding diagrammatic axioms must be updated. Translations from concept diagrams to description logic will also be devised but, again, finding an optimal symbolic set of axioms will be difficult; as a trivial example, one must choose between the axioms $C_1 \sqsubseteq C_2$ and $C_1 \sqsubseteq C_3$ and the single axiom $C_1 \sqcup C_2 \sqsubseteq C_3$.

6 Automated Layout

When devising heterogeneous approaches to ontology engineering, it is important that one is able to automatically draw – or lay out – concept diagrams. When automatically drawing diagrams, one starts with the abstract syntax of the required diagram or diagrams. The problem is to draw effective diagrams, with the given abstract syntax. There is considerable choice of layout, where figure 17 shows what we consider to be a bad layout of the diagram shown in figure 2.

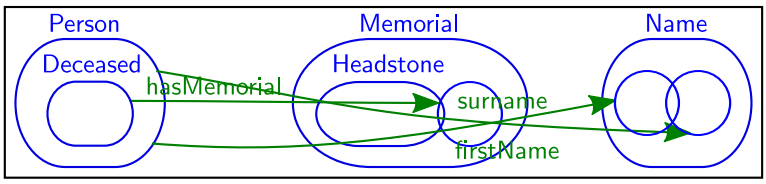

Fig. 17. A bad layout.

Indeed, the problem of drawing multiple diagrams, where common parts of two or more diagrams should – for effective use – look identical substantially increases the difficulty. This is illustrated in figures 18 and 19. Figure 18 presents, in separate diagrams, some relationships involving the classes Memorial and Headstone. The two diagrams have been drawn so that they have a layout that reflects

the structurally similar information represented. However, we expect that the diagrams in figure 19 are a more effective pair of visualizations than those figure 18, since their common parts have the same relative positioning in the plane.

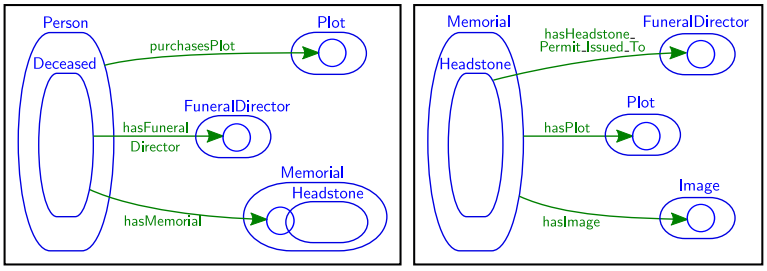

Fig. 18. Some Deceased relationships and some Memorial relationships.

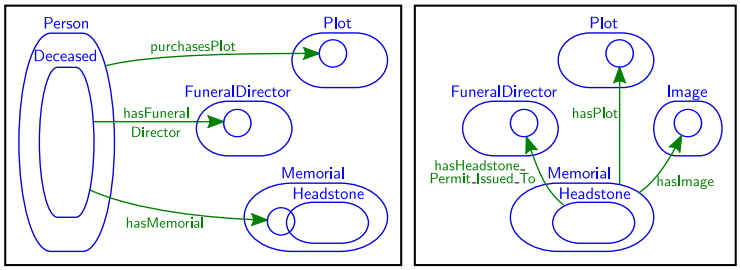

Fig. 19. Memorial relationships redrawn to align with layout of Deceased relationships

The problem of drawing multiple-diagrams when translating between notations, where many diagrams can arise from symbolic axioms. Whilst the drawing problem is inherently computationally complex, it is necessary to seek drawing and layout algorithms that have acceptable run times for most cases that arise in practice and that produce usable results. Empirical evaluations will be necessary to determine what constitutes an effective layout, informing choices about diagram topology and geometry.

As a staring point for a general drawing algorithm for concept diagrams, existing Euler diagram drawing methods such as [19–21] can be extended to simply add the extra syntax required (giving primacy to the Euler diagram). However, this approach will require extensive modifications to existing algorithms, since the initially drawn Euler diagram may compromise the layout of the subsequently drawn syntax. Ideally, the drawing algorithms will consider the entirety of the to-be-drawn diagram when making layout choices, not assigning primacy to any one syntactic part. Moreover, general drawing algorithms are computationally

expensive and may not always produce effective layouts. For this reason, we plan to devise, first, a layout algorithm for classes of concept diagrams that commonly arise in practice.

Lastly, drawing algorithms are required that incrementally update diagrams when edits are made to a symbolically specified ontology, to keep the two views in sync. These edits can include: deleting axioms, adding new axioms or amending existing axioms (which can be considered as a deletion followed by an addition). When an axiom is deleted, this can have profound effects on the layout of diagrams. For instance, deleting a disjointness axiom may require some of the (non-overlapping) curves in a diagram to be re-routed (so they overlap, thus not asserting disjointness). Similarly, adding an axiom can also require significant changes to parts of the diagrams, such as removing an overlap between curves. When diagrams include a lot of syntax, making such changes to curves is not necessarily straightforward.

7 Conclusion

Our vision is to produce a fully supported diagrammatic logic for ontology engineering, which includes implementing software to support its use. If successful, this diagrammatic logic will provide more accessible ways to develop, update and maintain ontologies. Since the diagrammatic framework that we envisage will be fully integrated with existing symbolic approaches, expert ontology engineers will be able to effectively collaborate with stakeholders who prefer a diagrammatic approach. We view this integration as necessary for the take-up of concept diagrams generally.

Acknowledgements We are indebted to Rangitikei District Council for providing us with access to the cemetery data on which the examples in this paper are based.

References

1. OntoGraf. http://protegewiki.stanford.edu/wiki/OntoGraf (2014), accessed Aug 2014
2. OWLViz. http://protegewiki.stanford.edu/wiki/OWLViz (2014), accessed Aug 2014
3. The BioPortal ontology repository. http://bioportal.bioontology.org/ (2014), accessed Aug 2014
4. The Manchester OWL Corpus. http://owl.cs.manchester.ac.uk/publications/ (2014), accessed Aug 2014
5. The OWL 2 Web Ontology Language. http://www.w3.org/TR/owl2-overview/ (2014), accessed Aug 2014
6. The Protégé web site. http://protege.stanford.edu/ (2014), accessed Aug 2014
7. The Swoogle ontology repository. http://swoogle.umbc.edu/ (2014), accessed Aug 2014

8. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Softw. 20, 36–41 (Sep 2003)
9. Baader, F., Calvanese, D., McGuinness, D., Nadi, D., (eds), P.P.S.: The Description Logic Handbook. CUP (2003)
10. Dau, F., Eklund, P.: A diagrammatic reasoning system for the description logic $\mathcal{ACL}$. Journal of Visual Languages and Computing 19(5), 539–573 (2008)
11. Duncan, J., Humphreys, G.W.: Visual search and stimulus similarity. Psychological review 96(3), 433 (1989)
12. Erwig, M.: Abstract syntax and semantics of visual languages. Journal of Visual Languages and Computing 9, 461–483 (1998)
13. Gurr, C.: Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. Journal of Visual Languages and Computing 10(4), 317–342 (1999)
14. Hayes, P., Eskridge, T.C., Mehrotra, M., Bobrovnikoff, D., Reichherzer, T., Saavedra, R.: Coe: Tools for collaborative ontology development and reuse. In: Knowledge Capture Conference (K-CAP). vol. 2005 (2005)
15. Howse, J., Molina, F., Shin, S.J., Taylor, J.: Type-syntax and token-syntax in diagrammatic systems. In: Proceedings FOIS-2001. pp. 174–185. ACM Press (2001)
16. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing ontologies: A case study. In: International Semantic Web Conference. pp. 257–272. Springer (2011)
17. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods a survey. ACM Comput. Surv. 39(4), 10+ (Nov 2007)
18. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL. In: Motta, E., Shadbolt, N., Stutt, A., Gibbins, N. (eds.) Engineering Knowledge in the Age of the Semantic Web, LNCS, vol. 3257, chap. 5, pp. 63–81. Springer (2004)
19. Riche, N., Dwyer, T.: Untangling Euler diagrams. IEEE Transactions on Visualization and Computer Graphics 16(6), 1090–1099 (2010)
20. Simonetto, P.: Visualisation of Overlapping Sets and Clusters with Euler Diagrams. Ph.D. thesis, Université Bordeaux (2012)
21. Stapleton, G., Flower, J., Rodgers, P., Howse, J.: Automatically drawing Euler diagrams with circles. Journal of Visual Languages and Computing 23(3), 163–193 (2012)
22. Stapleton, G., Howse, J., Chapman, P., Delaney, A., Burton, J., Oliver, I.: Formalizing Concept Diagrams. In: Visual Languages and Computing. pp. 182–187. Knowledge Systems Institute (2013)
23. Stapleton, G., Howse, J., Taylor, K., Delaney, A., Burton, J., Chapman, P.: Towards Diagrammatic Ontology Patterns. In: 4th Workshop on Ontology and Semantic Web Patterns. CEUR, Sydney, Australia (Oct 2013)
24. Warren, P., Mulholland, P., Collins, T., Motta, E.: The Usability of Description Logics. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) The Semantic Web: Trends and Challenges, LNCS, vol. 8465, pp. 550–564. Springer (2014)