# Visualizing Ontologies: A Case Study

John Howse[1], Gem Stapleton[1], Kerry Taylor[2], and Peter Chapman[1]

[1] Visual Modelling Group, University of Brighton, UK
{John.Howse,g.e.stapleton,p.b.chapman}@brighton.ac.uk
[2] Australian National University and CSIRO, Australia
Kerry.Taylor@csiro.au

**Abstract.** Concept diagrams were introduced for precisely specifying ontologies in a manner more readily accessible to developers and other stakeholders than symbolic notations. In this paper, we present a case study on the use of concept diagrams in visually specifying the Semantic Sensor Networks (SSN) ontology. The SSN ontology was originally developed by an Incubator Group of the W3C. In the ontology, a sensor is a physical object that implements sensing and an observation is observed by a single sensor. These, and other, roles and concepts are captured visually, but precisely, by concept diagrams. We consider the lessons learnt from developing this visual model and show how to convert description logic axioms into concept diagrams. We also demonstrate how to merge simple concept diagram axioms into more complex axioms, whilst ensuring that diagrams remain relatively uncluttered.

## 1 Introduction

There is significant interest in developing ontologies in a wide range of areas, in part because of the benefits brought about by being able to reason about the ontology. In domains where a precise (formal) specification of an ontology is important, it is paramount that those involved in developing the ontology fully understand the syntax in which the ontology is defined. For instance, one formal notation is description logic [3], for which much is known about the complexity of reasoning over its fragments [4].

Notations such as description logics require some level of mathematical training to be provided for the practioners using them and they are not necessarily readily accessible to all stakeholders. There have been a number of efforts towards providing visualizations of ontologies, that allow their developers and users access to some information about the ontology. For example, in Protégé, the OWLViz plugin [10] shows the concept (or class) hierarchy using a directed graph. Other visualization efforts provide instance level information over populated ontologies [11]. To the best of our knowledge, the only visualization that was developed as a direct graphical representation of description logics is a variation on existential graphs, shown to be equivalent to $\mathcal{ACL}$ by Dau and Eklund [7]. However, existential graphs, in our opinion, are not readily usable since their syntax is somewhat restrictive: they have the flavour of a minimal first-order logic with

only the $\exists$ quantifier, negation and conjunction; the variation developed as an equivalent to $\mathcal{ACL}$ uses ? to act as a free variable.

In previous work, Oliver et al. developed concept diagrams (previously called ontology diagrams) as a formal logic for visualizing ontology specifications [12, 13], further explored in Chapman et al. [5]. Whilst further work is needed to establish fragments for which efficient reasoning procedures can be devised, concept diagrams are capable of modelling relatively complex ontologies since they are a second-order logic.

The contribution of this paper is to demonstrate how concept diagrams can be used to model (part of) the Semantic Sensor Networks (SSN) ontology, in its current version, which was developed over the period February 2009 to September 2010 by an Incubator Group of the W3C, called the SSN-XG [6]. We motivate the need for accessible communication of ontology specifications in section 2, ending with a discussion around why visualization can be an effective approach. Section 3 presents a formalization of some of the SSN ontology's axioms using concept diagrams and using description logic, contrasting the two approaches. Section 4 demonstrates how to translate description logic axioms to concept diagrams and some inference rules. Section 5 concludes the paper.

## 2 Motivation

Complex ontologies are often developed by groups of people working together, consistent with their most important application: to support the sharing of knowledge and data. The most common definition of ontology refers to "an explicit representation of a shared conceptualisation" [9]. A *shared* conceptualisation is usually needed for the purposes for which ontologies are most used: for representation of data to be shared amongst individuals and organisations in a community. The "sharing" is necessary when domain-knowledge capture through an ontology requires modelling of either commonly-held domain knowledge or the common element of domain knowledge across multiple domains. This needs to take account of instances that are asserted to exist in the ontology, and also instances that *might* exist, or come into existence when the ontology is applied to describe some data.

The W3C's Web Ontology Language (OWL 2.0) is a very expressive but decidable description logic: a fragment of first order predicate calculus. A brief and incomplete introduction is given here: the reader is referred to [1] for a complete treatment. In common with all ontology languages, a hierarchical taxonomy of concepts (called *classes* in OWL) is the primary modelling notion. *Individuals* can be members (or *instances*) of concepts and all individuals are members of the predefined concept Thing, no individuals are members of the predefined Nothing. Binary relations over concepts, called *roles*, are used to relate individuals to others, and concept constructors comprising complex logical expressions over concepts, roles and individuals are used to relate all these things together. Most important here are role restrictions: expressions that construct a concept by referring to relations to other concepts. There are also a range of *role charac-*

*teristics* that can constrain the relations wherever they occur: such as domain, range, transitive, subproperty and inverse. Two key features of OWL designed for the semantic web applications is that all entities: classes (concepts), properties (roles) and individuals are identified by URI (a Web identifier that can be a URL), and that it has an RDF/XML serialization (commonly considered unreadable).

In the experience of these authors, when people meet to develop conceptual structures, including models of knowledge intended to become an OWL ontology, they very quickly move to sketching 2D images to communicate their thoughts. At the beginning, these may be simple graph structures of labelled nodes connected by labelled or unlabelled arcs. For example, figure 1 shows the whiteboard used at the first Face-to-face meeting of the W3C Semantic Sensor Networks Incubator Group, in Washington DC, USA, November 2009. Unlike some modelling languages, OWL does not have a heritage in visual representations, and modellers struggle with different interpretations of the visualizations used in the group. For example, in OWL, it is very important to know whether a node represents an individual or a class. In a more advanced example, modellers need to know whether a specified subsumption relationship between concepts is also an equivalence relationship. As we shall see, concept diagrams are capable of visualizing exactly these kinds of definitions.

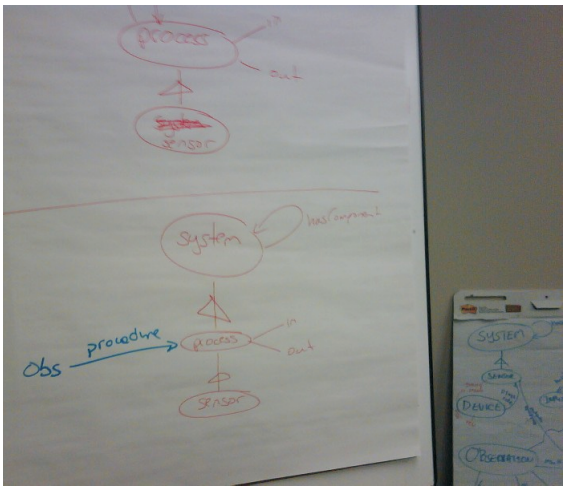



**Fig. 1.** Whiteboard used at face-to-face meeting; Photo: M. Hauswirth.

The major feature of OWL as a modelling language is also its greatest hindrance for shared development: the formal semantics and capability for *reasoning*. The examples we give later, in our case study, demonstrate that information that would sometimes need to be inferred is actually explicitly visible on concept

diagrams (so-called *free-rides* which we explain later). It is commonplace for ontology developers, as domain experts, to be unaware of the formal semantics underlying OWL, and if they are aware it remains very difficult to apply the knowledge of the semantics in practice while developing in a team environment. For example, even the simple difference between universal and existential role restrictions are difficult to represent and to assimilate in diagrammatic form. As another example, the semantic difference between rdfs:domain and rdfs:range constraints on properties, as opposed to local restrictions on those properties in the context of class definitions, is difficult to represent diagrammatically and very hard to take into account when studying spatially-disconnected but semantically-connected parts of an ontology sketch. There is a need for semantically-informed sketching tools that help ontology developers to better understand their modelling in real time.

## 3 Visualizing the SSN Ontology

In this section we walk through parts of the SSN ontology, showing how to express it in our concept diagrams. At the end of section 3.2 we will give a summary of the concept diagram syntax. The SSN ontology is available at purl.oclc.org/NET/ssnx/ssn and extensive documentation and examples of its use are available in the final report of the SSN-XG [2]. An alternative visualization of the SSN ontology was created using CMAP from IHMC (see www.ihmc.us/groups/coe/) and may be compared with the visualisation presented here. The ontology imports, and is aligned with, the Dolce Ultra-Lite upper ontology [14] from which it inherits upper concepts including Event, Object, Abstract, Quality, PhysicalObject, SocialObject, InformationObject, Situation, Description, Method, and Quality.

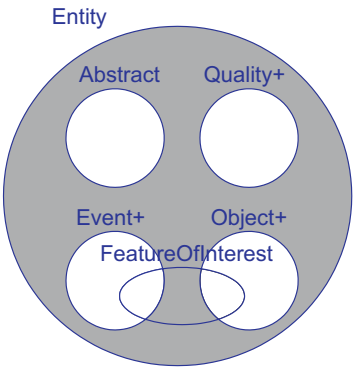
### 3.1 Concept Hierarchy Axioms

To represent the concept hierarchy, concept diagrams use Euler diagrams [8], which effectively convey subsumption and disjointness relationships. In particular, Euler diagrams comprise closed curves (often drawn as circles or ellipses) to represent sets (in our case, concepts). Two curves that have no common points inside them assert that the represented sets are disjoint whereas one curve drawn completely inside another asserts a subsumption relationship. In addition, Euler diagrams use shading to assert emptiness of a set; in general, concept diagrams use shading to place upper bounds on set cardinality as we will see later.

In the SSN ontology, descriptions of the concepts are given as comments in the ssn.owl file [2], which we summarize here. The SSN ontology is defined over a large vocabulary of which we present the subset required for our case study. At the top level of the SSN hierarchy are four concepts, namely Entity, FeatureOfInterest, Input, and Output. The concept Entity is for anything real, possible or imaginary that the modeller wishes to talk about. Entity subsumes five other concepts which, in turn, may subsume further concepts. The five concepts are:

1. Abstract These are entities that cannot be located in space and time, such as mathematical concepts.
2. Event These are physical, social, or mental processes, events, or states. Event is, therefore, disjoint from Abstract.
3. Object These are physical, social or mental objects or substances. Therefore, Object is disjoint from Abstract and Event.
4. FeatureOfInterest A feature of interest is an abstraction of real world phenomena and is subsumed by the union of Event and Object.
5. Quality This is any aspect of an entity that cannot exist without that entity, such as a surface of a solid object. Quality is also disjoint from Abstract, Event, and Object.

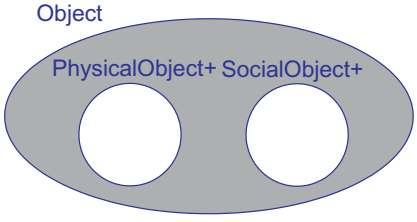
An Euler diagram asserting these subsumption and disjointness properties as a single axiom, alongside the axioms expressed using description logic, can be seen here:




1. Abstract $\sqsubseteq$ Entity
2. Quality $\sqsubseteq$ Entity
3. Event $\sqsubseteq$ Entity
4. Object $\sqsubseteq$ Entity
5. Abstract $\sqcap$ Quality $\sqsubseteq \bot$
6. Abstract $\sqcap$ Event $\sqsubseteq \bot$
7. Abstract $\sqcap$ Object $\sqsubseteq \bot$
8. Quality $\sqcap$ Event $\sqsubseteq \bot$
9. Quality $\sqcap$ Object $\sqsubseteq \bot$
10. Event $\sqcap$ Object $\sqsubseteq \bot$
11. FeatureOfInterest $\sqsubseteq$ Event $\sqcup$Object
12. Entity $\equiv$ Abstract $\sqcup$Object $\sqcup$Event $\sqcup$Quality

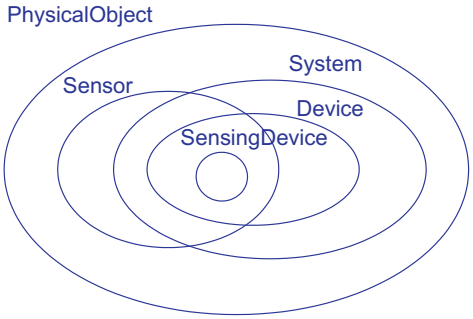
The Euler diagram has a certain succinctness over the description logic in that there are 6 DL axioms asserting disjointness properties, for example.

Notice, in the figure above, the concept Object is annotated with a plus symbol, as are Event and Quality. Whilst not part of the formal syntax, this plus symbol is used to indicate that there are concepts subsumed by each of these concepts that are not displayed in this diagram; with tool support, one could imagine clicking on this plus to 'expand' the diagram, to show the subsumed concepts. In the SSN ontology, Object is the union of two disjoint concepts, PhysicalObject and SocialObject:




1. Object $\equiv$ PhysicalObject $\sqcup$SocialObject
2. PhysicalObject $\sqcap$ SocialObject $\sqsubseteq \bot$

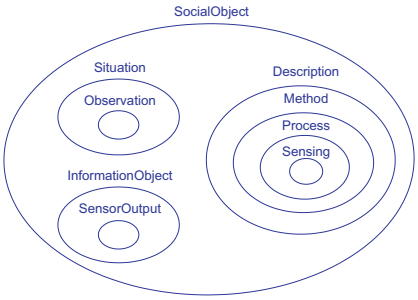
A PhysicalObject is an object that has a proper space region whereas a SocialObject exists only within some communication Event, in which at least one PhysicalObject participates. Again, as indicated by the plus sign, PhysicalObject subsumes various other concepts: Sensor, System, Device, and SensingDevice. A Sensor can do sensing: that is, a Sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. A System is a unit of abstraction for pieces of infrastructure for sensing, namely Device and SensingDevice. A Device is a physical piece of technology, of which SensingDevice is an example. Additionally, SensingDevice is an example of Sensor. This information about the SSN ontology is axiomatized by the single Euler diagram below, and equivalently by the adjacent description logic axioms:




1. Sensor $\sqsubseteq$ PhysicalObject
2. System $\sqsubseteq$ PhysicalObject
3. Device $\sqsubseteq$ System
4. SensingDevice $\sqsubseteq$ Device
5. SensingDevice $\sqsubseteq$ Sensor

It should be clear that the diagram just given makes some informational content explicit, whereas it needs to be derived from the description logic axioms. For instance, one can easily read off, from the diagram, that SensingDevice is subsumed by PhysicalObject, since the closed curve representing the former is contained by the closed curve representing the latter. From the description logic axioms, one must use the transitive property of $\sqsubseteq$ to extract this information: SensingDevice $\sqsubseteq$ Sensor $\sqsubseteq$ PhysicalObject. To make this deduction, one has to identify appropriate description logic axioms from the list given, which requires a little more effort than reading the diagram. This example, illustrating the inferential properties of the diagram, is a typical example of a *free-ride* (sometimes called a *cheap ride*), the theory of which was developed by Shimojima [16], later explored by Shimojima and Katagiri [17]. In general, a free-ride is a piece of information that can be readily 'seen' in a diagram that would typically need to be inferred from a symbolic representation.

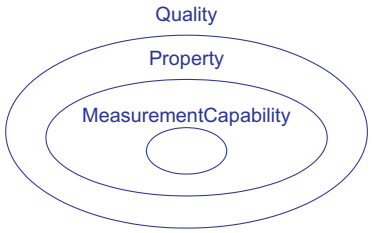
SocialObject also subsumes various other concepts, which we do not describe in full here. Three will be of use to us later: a Situation is a view on a set of entities; an Observation is a Situation in which a SensingMethod has been used to estimate or calculate a value of a Property of a FeatureOfInterest; and Sensing is a process that results in the estimation, or calculation, of the value of a phenomenon. The following Euler diagram defines an axiom from the SSN ontology, and the adjacent description logic statements capture the same information:




1. Situation $\sqsubseteq$ SocialObject
2. Observation $\sqsubseteq$ Situation
3. InformationObject $\sqsubseteq$ SocialObject
4. SensorOutput $\sqsubseteq$ InformationObject
5. Description $\sqsubseteq$ SocialObject
6. Method $\sqsubseteq$ Description
7. Process $\sqsubseteq$ Method
8. Sensing $\sqsubseteq$ Process
9. Situation $\sqcap$ Description $\sqsubseteq \bot$
10. Situation $\sqcap$ SocialObject $\sqsubseteq \bot$
11. InformationObject $\sqcap$ Description $\sqsubseteq \bot$

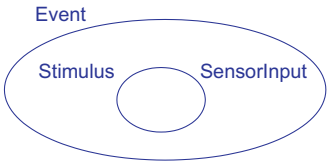
This diagram, presenting information about the concepts subsumed by SocialObject, also has many free-rides, such as Sensing is subsumed by Description, and that Process is disjoint from Observation since the two curves do not overlap. For the latter, to deduce this from the given description logic axioms, one would need to use axiom numbers 2, 6, 7, and 9.

We saw earlier that Quality was subsumed by one of the top-level concepts, Entity. In turn, Quality subsumes Property, which is an observable quality of an event or object. Property subsumes many concepts, but we only make use of one of them later: MeasurementCapability. This concept collects together measurement properties (accuracy, range, precision, etc) as well as the environmental conditions in which those properties hold, representing a specification of a sensor's capability in those conditions:




1. Property $\sqsubseteq$ Quality
2. MeasurementCapability $\sqsubseteq$ Property

The last part of the concept hierarchy that we demonstrate concerns Event, which was subsumed by the top-level concept Entity. Two of the concepts subsumed by Entity are Stimulus and SensorInput. A sensor input is an event that triggers the sensor and the concept SensorInput is equivalent to Stimulus:




1. Stimulus $\sqsubseteq$ Event
2. SensorInput $\sqsubseteq$ Event
3. SensorInput $\equiv$ Stimulus

Notice here that, in the Euler diagram, we have asserted equivalence between concepts by drawing two circles on top of one another.

We have represented 24 of the SSN concepts using Euler diagrams to assert subsumption and disjointness relationships. The discussions around free-rides indicate that Euler diagrams (the basis of concept diagrams) can be an effective method of axiomatizing concept hierarchies. We refer the reader to [2] for further information on the hierarchy.

### 3.2 Role Rescrictions

Moving on to role restrictions, concept diagrams extend Euler diagrams by incorporating more syntax to increase their expressiveness. In particular, arrows are used to represent role restrictions. The source of the arrow is taken to restrict the domain of the role, and the target provides some information about the image of the role under the domain restriction. The nature of the information given is determined by the arrow's type: arrows can be dashed or solid. Given a solid arrow, $a$, sourced on $\mathsf{C}$ and targeting $\mathsf{D}$, representing the role $\mathsf{R}$, $a$ asserts that the image of $\mathsf{R}$ when it's domain is restricted to $\mathsf{C}$ is *equal* to $\mathsf{D}$, that is:

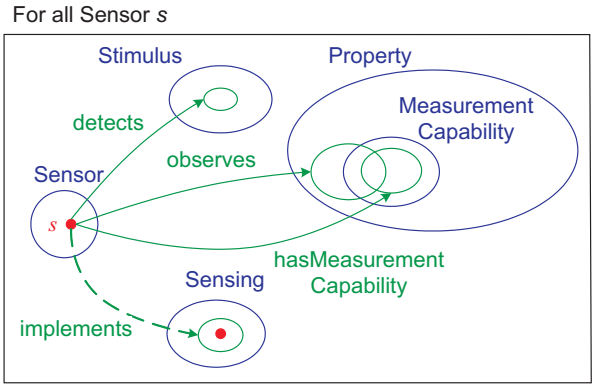
$$image(\mathsf{R}|_{\mathsf{C}}) = \mathsf{D} \qquad \text{where } image(\mathsf{R}|_{\mathsf{C}}) = \{y : \exists x \in \mathsf{C} \ \ (x, y) \in \mathsf{R}\}.$$

By contrast, if $a$ were instead dashed then it would assert that the image of $\mathsf{R}$ when its domain is restricted to $\mathsf{C}$ includes at least the elements in $\mathsf{D}$, that is:

$$image(\mathsf{R}|_{\mathsf{C}}) \supseteq \mathsf{D}.$$

As we shall see in our examples, the syntax that can be used as sources and targets of arrows, including closed curves (both labelled, as in Euler diagrams, or unlabelled), or dots. Unlabelled closed curves represent anonymous concepts and dots represent individuals. As with closed curves, dots can be labelled to represent specific individuals, or unlabelled to represent anonymous individuals. The syntax and semantics will be more fully explained as we work through our examples.

Our first example of some role restrictions concerns the concept Sensor, since this is at the heart of the SSN ontology. There are various restrictions placed on the roles detects, observes, hasMeasurementCapability and implements. The first of these, detects, is between Sensor and Stimulus: sensors detect only stimuli. Next, there is a role observes between Sensor and Property: sensors observe only properties. Thirdly, every sensor hasMeasurementCapability, the set of which is subsumed by MeasurementCapability. Finally, every sensor implements some sensing. That is, sensors have to perform some sensing. The concept diagram below captures these role restrictions:




Here, we are quantifying over the concept Sensor, since we have written 'For all Sensor $s$' above the bounding box of the diagram (in the formal abstract syntax of concept diagrams, this would be represented slightly differently, the details of which are not important here). The dot labelled $s$ in the diagram is then the source of four arrows, relating to the role restrictions just informally described. The solid arrow labelled detects is used to place the following restriction on the detects role:

$$image(\mathsf{detects}|_{\{s\}}) \subseteq \mathsf{Stimulus},$$

treating the individual $s$ as a singleton set. The unlabelled curve is acting as an existentially quantified anonymous set so, strictly, the arrow and the unlabelled curve assert:

$$\exists X\, image(\mathsf{detects}|_{\{s\}}) = X \wedge X \subseteq \mathsf{Stimulus}.$$

Earlier, we defined an axiom that asserted MeasurementCapability is subsumed by Property, along with axioms that give the disjointness information conveyed in the diagram above. We have made use of that information in the diagram above, by drawing the curves with appropriate containment and disjointness properties. A further point of note is that, in this diagram, we have not asserted anything about whether $image(\mathsf{observes}|_{\{s\}})$ and $image(\mathsf{hasMeasurementCapability}|_{\{s\}})$ are disjoint, or whether one subsumes the other. All we know is that the former is subsumed by Property and the latter is subsumed by MeasurementCapability. Finally, the dashed arrow provides partial information:

$$\exists X \exists y\, image(\mathsf{implements}|_{\{s\}}) \supseteq X \wedge X \subseteq \mathsf{Sensing} \wedge y \in X$$

where $X$ arises from the unlabelled curve targeted by the arrow and $y$ arises from the unlabelled dot placed inside this curve; we are using unlabelled dots to assert the existence of individuals.

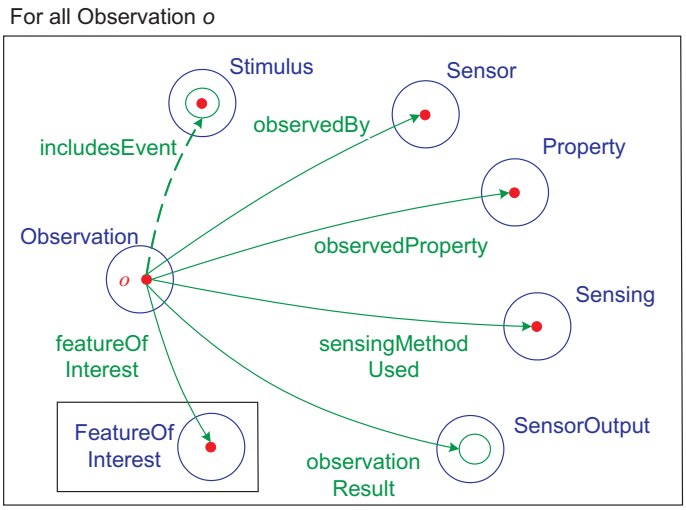
The role restrictions just given, together with the disjointness information, can also be expressed using the following description logic axioms:

1. $\mathsf{Sensor} \sqsubseteq \forall\ \mathsf{detects}.\mathsf{Stimulus}$

2. Sensor $\sqsubseteq \forall$ observes.Property
3. Sensor $\sqsubseteq \exists$ implements.Sensing
4. Sensor $\sqsubseteq \forall$ hasMeasurementCapability.MeasurementCapability
5. Sensor $\sqcap$ Stimulus $\sqsubseteq \bot$
6. Sensor $\sqcap$ Property $\sqsubseteq \bot$
7. Sensor $\sqcap$ Sensing $\sqsubseteq \bot$
8. Stimulus $\sqcap$ Property $\sqsubseteq \bot$
9. Stimulus $\sqcap$ Sensing $\sqsubseteq \bot$
10. Property $\sqcap$ Sensing $\sqsubseteq \bot$

We can see that the concept diagram has free-rides arising from the use of the unlabelled curves. For example, it is easy to see that $image(\mathsf{detects}|_{\{s\}})$ is disjoint from Property, but this information is not immediately obvious from the description logic axioms: one must make this deduction from axioms 1 and 8.

Our second collection of role restrictions concerns the concept Observation. Here, an observation includes an event, captured by the role includesEvent, which is a Stimulus, illustrated by the dashed arrow in the diagram immediately below. In addition, observations are observedBy exactly one (unnamed) individual, which is a Sensor. Similarly, observations have exactly one observedProperty and this is a Property, exactly one sensingMethodUsed and this is a Sensing object, and a set of observationResults all of which are SensorOutputs. Finally, observations have exactly one featureOfInterest (role), which is a FeatureOfInterest (concept). All of these role restrictions are captured in the diagram below, where again we have used previous information about disjointness to present a less cluttered diagram:




Here, of note is the use of a rectangle around the closed curve labelled FeatureOfInterest. The rectangle is used to assert that we are not making any claim about the disjointness of FeatureOfInterest with respect to the other concepts appearing in the diagram.

To allow the reader to draw contrast with description logic, the role restrictions just given are expressed by 21 description logic axioms, comprising 15 disjointness axioms and the following 6 axioms that correspond to the information provided by the six arrows:

1. $\mathsf{Observation} \sqsubseteq \exists\ \mathsf{includesEvent.Stimulus}$
2. $\mathsf{Observation} \sqsubseteq (= 1\ \mathsf{observedBy}) \sqcap (\forall\ \mathsf{observedBy.Sensor})$
3. $\mathsf{Observation} \sqsubseteq (= 1\ \mathsf{observedProperty}) \sqcap (\forall\ \mathsf{observedProperty.Property})$
4. $\mathsf{Observation} \sqsubseteq (= 1\ \mathsf{sensingMethodUsed}) \sqcap (\forall\ \mathsf{sensingMethodUsed.Sensing})$
5. $\mathsf{Observation} \sqsubseteq \forall\ \mathsf{ObservationResult.SensorOutput}$
6. $\mathsf{Observation} \sqsubseteq (= 1\ \mathsf{FeatureOfInterest}) \sqcap (\forall\ \mathsf{FeatureOfInterest.FeatureOfInterest})$

Consider axiom 2, which corresponds to the arrow labelled observedBy. From the description logic axiom, a little reasoning is required to see that every observation is related to exactly one individual, which must be a sensor: one must deduce this from the information that Observation is subsumed by the set of individuals that are related to exactly one thing under observedBy intersected with the set of individuals that are related to only properties under observedBy. In our opinion, the diagram more readily conveys the informational content of the axioms than the description logic syntax and in a more succinct way (although this, of course, could be debated).

To conclude this section, we summarize main syntax of concept diagrams:

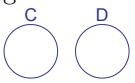
1. **Rectangles** These are used to represent the concept Thing.
2. **Closed Curves** These are used to represent concepts. If the curve does not have a label then the concept is anonymous. The spatial relationships between the curves gives information about subsumption and disjointness relationships.
3. **Dots** These are used to represent individuals. As with closed curves, unlabelled dots represent anonymous individuals. The location of the dot gives information about the type of the individual. Distinct dots represent distinct individuals. When many dots are present in a region, we may use $\leq$, $=$, and $\geq$ as shorthand annotations (this will be demonstrated later).
4. **Shading** Shading in a region asserts that the concept represented by the region contains only individuals represented by dots.
5. **Solid Arrows** These are used to represent role restrictions. In particular, the image of the role whose label appears on the arrow has an image, when the domain is restricted to (the concept or individual represented by) the source, is *equal* to the target.
6. **Dashed Arrows** These are used to represent role restrictions. In particular, the image of the role whose label appears on the arrow has an image, when the domain is restricted to the source, which is a *superset* of the target.

In addition, quantifiers and connectives can be used in the standard way.

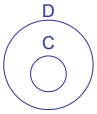
## 4 Discussion

We will now extract, from the case study that we have presented, some general constructions of diagrams, from description logic axioms. Moreover, we will show how to take these simple axioms and merge them into more complex axioms, by providing inference rules. These inference rules are inspired by the manner in which we produced our visualization of the SSN ontology, aiming for diagrams with minimal clutter, without compromising their informational content.

With regard to subclass and disjointness information, where C and D are concepts, we have the following translations:

C is disjoint from D: $C \sqcap D \sqsubseteq \bot \mapsto$ 

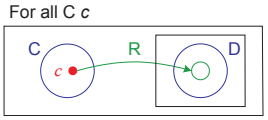

C is equivalent to D: $C \equiv D \mapsto$ 


C is subsumed by D: $C \sqsubseteq D \mapsto$ 

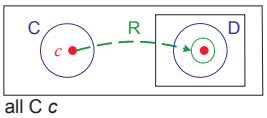

However, using these translations would give one diagram for every disjointness, subsumption and equivalence axiom in the ontology. As we have seen, it is possible to produce readable diagrams that correspond to many axioms of the kind just given (all of our diagrams that conveyed concept hierarchy information expressed more than one description logic axiom). There is clearly a requirement on the ontology deverloper to determine a balance between the number of axioms like these conveyed in a single diagram and the clutter in a diagram. Our diagrams were drawn in a manner that concepts were only in the same diagram if we wanted to assert something about their relationship. Later, we will give some general rules for merging these simple diagrams into larger diagrams.

As we saw earlier, we can readily translate information about 'only' or 'some' role restrictions into diagrammatic form. For example, in the Sensor concept, we have Sensor detects only Stimulus and Sensor implements some Sensing. Abstracting from this, and including more general constraints, we have role restrictions of these forms, where C and D are concepts and R is a role:

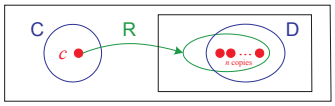
C is subsumed by the anonymous concept containg individuals related, under R, to only individuals in D:

$C \sqsubseteq \forall R.D \mapsto$ 


C is subsumed by the anonymous concept containg individuals related, under R, to an individual in D:

$C \sqsubseteq \exists R.D \mapsto$ 


C is subsumed by the anonymous concept containg individuals related, under R, to at least $n$ individuals in D:

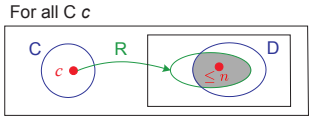
$C \sqsubseteq \geq n\, R.D \mapsto$ 


In the above, instead of drawing $n$ dots, we could use one dot annotated with $\geq n$ as shorthand which is sensible if $n$ gets beyond, say, 4. We can also adopt this shorthand for $\leq n$; we recall that shading is used to place upper bounds on set

cardinality, generalizing the use of shading in Euler diagrams, in a shaded region all elements must be represented by dots. We now give two further translations:

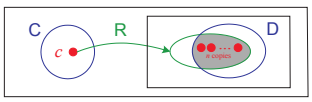
C is subsumed by the anonymous concept containg individuals related, under R, to at most $n$ individuals in D:

$\mathsf{C} \sqsubseteq \leq n\,\mathsf{R}.\mathsf{D} \mapsto$




C is subsumed by the anonymous concept containg individuals related, under R, to exactly $n$ individuals in D:

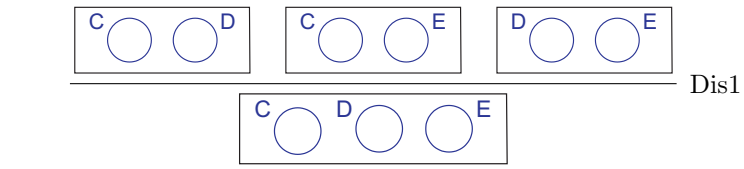
$\mathsf{C} \sqsubseteq = n\,\mathsf{R}.\mathsf{D} \mapsto$

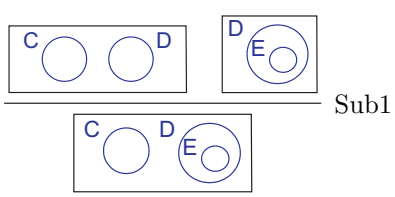


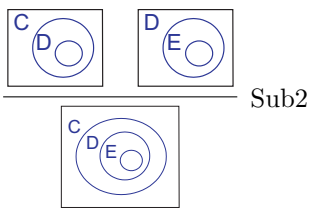

Again, in the diagram just given, we could have used the shorthand $= n$.

The above translations demonstrate how to produce concept diagrams from role restrictions defined using description logic. These translations are sound and, in fact, information preserving. As with the hierarchy information, there are often more succinct, elegant diagrams that can be created by representing many of these axioms in a single diagram. We will call the diagrams obtained by applying the transformations just given *atomic axioms*. We now demonstrate how to produce non-atomic axioms (like the diagrams given in the SSN ontology) from atomic axioms.

We begin by giving some inference rules that allow us to replace some axioms with others; in many cases there are obvious generalizations of the inference rules. We adopt a traditional presentation style, where axioms written above a line can be used to infer those written below the line. Each rule has a name, displayed in shorthand: Dis for Disjunction, Sub for Subsumption, and Mer for Merge. All of these rules are equivalences (no informational content is lost) and can be formalized and proved sound, but here we just present them informally using illustrative diagrams. First we have, concerning hierarchy information:

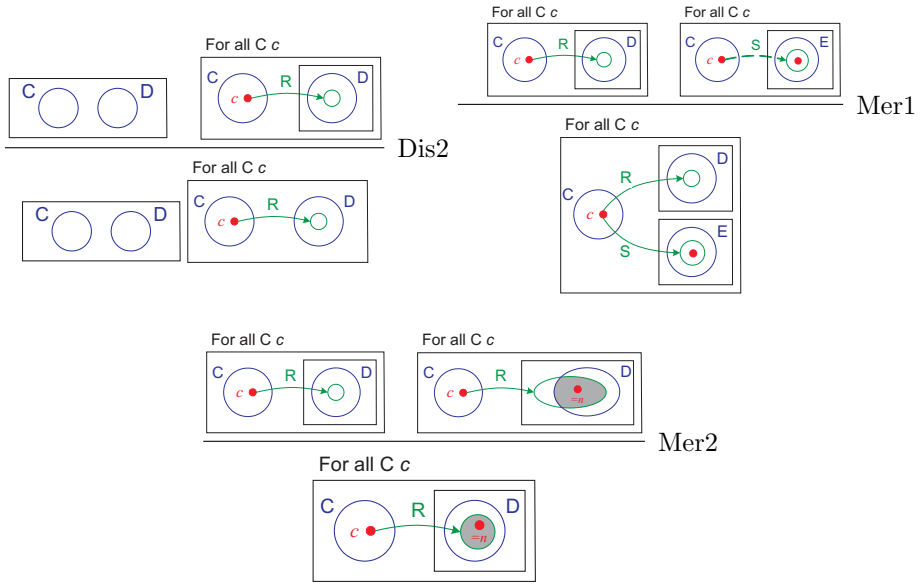


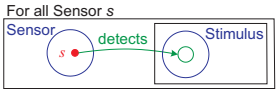



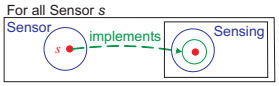

For instance, Dis1 says three axioms that tell us three concepts are pairwise disjoint is equivalent to a single diagram telling us that the concepts are pairwise disjoint. Sub1, tells us, roughly speaking, that if D appears in one axiom, $a$, and we know that E is subsumed by D then we can copy E into $a$, placing it inside D. Sub2 is another instance of this kind of inference.

Regarding role restrictions, we have seen that it is possible to use information about disjointness when creating these kinds of axioms. For instance, if we know that C and D are disjoint then we can simplify an axiom that tells us C is subsumed by $\forall R.D$: we do not have to place D in a separate box. This intuition is captured by our first role restriction rule, Dis2. Our second role restriction rule, Mer1, takes atomic axioms arising from $C \sqsubseteq \forall R.D$ and $C \sqsubseteq \forall S.E$ and merges them into a single diagram. Rule Mer2 is similar.



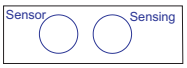

To demonstrate the use of these inference rules, we consider the example from the SSN network concerning Sensor on page 9. Translating the associated description logic axioms numbered 1, 3, 5, 7, and 9, using the techniques of the previous subsection, we get the following five diagrams:

Axiom 1:



Axiom 3:



Axiom 5:

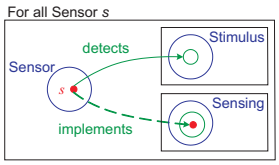


Axiom 7:



Axiom 9:

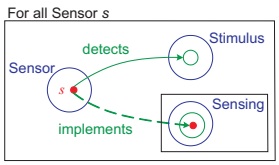


Using the rule Mer2, from axioms 1 and 3 we obtain:

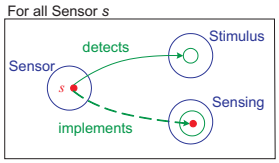



Using axiom 5, and a generalization of Dis2, we can delete the rectangle around

Stimulus (since Stimulus and Sensor are disjoint):




Using axioms 7 and 9, we further deduce:




We leave it to the reader to use these kinds of manipulations to obtain the single diagram given for the role restrictions imposed over the Sensor concept.

## 5 Conclusion

In this paper we have discussed the need for sophisticated ontology visualization techniques that will allow disparate groups of ontology developers and users to communicate effectively. Concept diagrams are a visual notation that were developed with this need in mind. We have used concept diagrams to produce a visualization of (part of) the Semantic Sensor Network ontology, including information about the concept hierarchy and role restrictions. Thus, this paper demonstrates that concept diagrams can be applied to modelling real-world ontologies. Concept diagrams may undergo further refinement as more case studies are developed and as they are applied in other domains.

An important future development is the implementation of tool support. We envisage developing tools which allow the automatic conversion of symbolically specified ontologies to concept diagrams. This will involve solving challenging problems, such as identifying what constitutes an effective diagram (as shown in this paper, there are different diagrams that convey the same information) and how to automatically draw chosen diagrams from abstract descriptions of them. This functionality could build on recent advances in automated Euler diagram drawing [15, 18, 19], although the layout problem for concept diagrams is more challenging. In addition, we want to allow ontology creators to be able to specify the axioms directly with concept diagrams, which may require a sketch recognition engine to be devised; this could also build on recent work that recognizes sketches of Euler diagrams [20]. These automatically drawn sketches can be translated into symbolic form, so that we can make use of sophisticated tool support that already exists for ontology development.

## References


1. W3C OWL Working Group, OWL 2 Web Ontology Language Document Overview, W3C Recommendation 27 October 2009. http://www.w3.org/TR/2009/REC-owl2-overview-20091027/, accessed June 2011.
2. Lefort et al. The W3C Semantic Sensor Network Incubator Group Final Report, www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/, accessed June 2011.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nadi, and P. Patel-Schneider (eds). *The Description Logic Handbook*. CUP, 2003.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nadi, and P. Patel-Schneider (eds). *The Description Logic Handbook*, chapter 3. CUP, 2003.
5. P. Chapman, G. Stapleton, J. Howse, and I. Oliver. Deriving sound inference rules for concept diagrams. In *IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2011.
6. Compton et al. The SSN Ontology of the Semantic Sensor Network Incubator Group, submitted to The Journal of Web Semantics, July 2011.
7. F. Dau and P. Eklund. A diagrammatic reasoning system for the description logic $\mathcal{ACL}$. *Journal of Visual Languages and Computing*, 19(5):539–573, 2008.
8. L. Euler. Lettres a une princesse d'allemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775. Berne, Socit Typographique.
9. T Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.
10. M. Horridge. OWLViz. www.co-ode.org/downloads/owlviz/, accessed June 2009.
11. Jambalaya. http://www.thechiselgroup.org/jambalaya.
12. I. Oliver, J. Howse, G. Stapleton, E. Nuutila, and S. Torma. A proposed diagrammatic logic for ontology specification and visualization. In *International Semantic Web Conference (Posters and Demos)*, 2009.
13. I. Oliver, J. Howse, G. Stapleton, E. Nuutila, and S. Torma. Visualising and specifying ontologies using diagrammatic logics. In *5th Australasian Ontologies Workshop*, 112:87–104. CRPIT, 2009.
14. V. Presutti and A. Gangemi. Content ontology design patterns as practical building blocks for web ontologies. In *Conceptual Modeling - ER 2008*, pages 128–141. Springer, 2008.
15. P. Rodgers, L. Zhang, and A. Fish. General Euler diagram generation. In *Diagrams*, pages 13–27. Springer, 2008.
16. A. Shimojima. Inferential and expressive capacities of graphical representations: Survey and some generalizations. In *Diagrams*, 18–21, Springer, 2004.
17. A. Shimojima and Y. Katagiri. An eye tracking study of spatial constraints in diagrammatic reasoning. In *Diagrams*, pages 64–88. Springer, 2008.
18. P. Simonetto and D. Auber. Visualise undrawable Euler diagrams. In *12th International Conference on Information Visualization*, pages 594–599. IEEE, 2008.
19. G. Stapleton, L. Zhang, J. Howse, and P. Rodgers. Drawing Euler diagrams with circles: The theory of piercings. *IEEE Transactions on Visualisation and Computer Graphics*, 17:1020–1032, 2011.
20. M. Wang, B. Plimmer, P. Schmieder, G. Stapleton, P. Rodgers, and A. Delaney. SketchSet: Creating Euler diagrams using pen or mouse. In *IEEE Symposium on Visual Languages and Human-Centric Computing 2011*. IEEE, 2011.