Reasoning about Sensors and Compositions

Michael Compton[1], Holger Neuhaus[2], Kerry Taylor[1], and Khoi-Nguyen Tran

[1] ICT Centre, CSIRO, Canberra
[2] Tasmanian ICT Centre, CSIRO, Hobart
firstname.lastname@csiro.au

Abstract. This paper discusses an OWL ontology for specifying sensors. The ontology is intended as the basis for the semantic representation of sensors and as the formal description for reasoning about sensors and observations. The paper describes the ontology, presents two example sensor descriptions and shows how standard reasoning and querying techniques can be used to perform tasks including classification and composition. In conjunction with the technical material the trade-offs required to express complex material in OWL is also discussed.

1 Introduction

A number of ontologies for specifying sensors have been developed and published. However, all lack the ability to describe sensors as compositions of existing sensors and algorithms. SensorML [3] includes process descriptions, descriptions of how a sensor's parts constitute the whole sensor, as one of its fundamental features. Indeed, if semantic sensor networks are to be used in the complex ways that have been envisaged, then semantic descriptions for sensors must be capable of describing sensors as processing units, and be capable of describing how existing sensors can be composed to form virtual sensors.

This paper presents an ontology that can describe the capabilities and properties of sensors as well as describing sensors as compositions of their components (§3). Two examples of sensors encoded in the ontology are presented (§4). The first (§4.1) shows the encoding of details of an actual sensor into the ontology, and the second (§4.2) discusses composition and virtual instruments. The examples are used to illustrate a discussion on OWL classification (§5) and reasoning other than OWL (§5.1). The main discussion on reasoning (§5.2) shows how the SPARQL plus OWL reasoning in SPARQL-DL [19] can be used to search for and construct virtual instruments.

Sensors observe physical qualities of features: for example, the temperature (quality) of a lake (feature). Here we take sensors to be sources that react to stimulus (physical or digital) and produce values representing a quality, thus including transducers, sensor devices and computations: for example, the specification of wind chill sensor in Section 4.1 could be implemented as an in situ device that measures wind speed and ambient temperature and calculates wind chill or a program that reads wind speed and ambient temperature measurements from external, co-located sensors and calculates wind chill.

While semantics can assist in searching for existing sensors, a more advanced use is to automatically compose a sensor satisfying a query if no such sensor exists. Once composed, such a sensor should be available as a sensor in its own right, and available to be used as a component of new virtual sensors. For example, a query such as "report event E each time condition C is reached in time period P", requires finding or composing a sensor that can detect C, finding or making a process that builds an E from a C and understanding the constraints in availability, power, and eventual degradation of the sensors over P.

Searching for and, if required, automatically or semi-automatically composing web services remains a much researched topic for Web services; see, for example, the September 2008 edition of the Data Engineering Bulletin [20]. While there have been some investigations on sensors and compositions, the work presented here is novel as compositions are built into the ontology and the bulk of the search is done using description logic (DL) reasoning and SPARQL, rather than with bespoke algorithms, and because the compositions use input, output and functional information.

Specifying detailed aspects of sensors and compositions requires a modelling capacity beyond the expressive and reasoning limits of OWL. Numeric information in describing accuracy, for example, cannot be reasoned about in description logic. However, trade-offs between the accuracy of the description and the ability to reason in OWL need to be made even for seemingly simple choices such as describing what a sensor measures or the input and output types of processes. The technical material on sensors and compositions in this paper is used to highlight a number of the choices that must be made to describe sensors in OWL.

In this paper OWL refers to the OWL2 Candidate Recommendation [1]. Concepts are written with an initial capital letter and roles with a lowercase letter. For simplicity we write

$$role \ : \ (ConceptA \times ConceptB)$$

to indicate that *role* is a DL role with domain *ConceptA* and range *ConceptB*. The definition of a role may be superscripted with $\mathcal{F}$ to indicate a functional role, $\mathcal{T}$ for transitive, $\mathcal{S}$ for symetric and $\mathcal{R}$ for reflexive. Composition of roles is written $r_1 \circ r_2$ for roles r_1 and r_2. That individual a is classified into concept C is written $C(a)$. In general, the so-called German DL Syntax is used.

2 Related Work

See the survey paper in the same proceedings [5] for a more complete review of sensor ontologies and technologies surrounding them.

The OntoSensor [18, 17] ontology was intended as a general knowledge-base of sensors for query and inference. Search using a mixture of DL reasoning and Logic Programming (LP) is discussed. Based on SensorML, OntoSensor covers similar concepts to the ontology in this paper. It is more complete in some areas, such as defining a hierarchy of sensor types, but is only able to express part-of

relations and is thus not suitable for describing sensor compositions and virtual instruments as discussed here.

Eid et al. [7, 8] and Kim et al. [12] developed sensor ontologies for enabling semantic Web services. Both use DL reasoning to check the consistency of the ontologies and SPARQL for simple searches. The ontologies are not available.

Calder et al. [4] and Thirunarayan et al. [21] use LP rules to make inferences about data emanating from sensor networks.

Whitehouse, Zhao and Liu [22] use LP to query and compose streams of data from sensors. Liu and Zhao [14] use an ontology of sensing concepts and services to convert declarative queries to service and sensor composition graphs, though note that the compositions are not represented in the ontology.

DL inference and LP are used with the ISTAR sensor ontology to find sensors that have outputs matching a task's requirements; a set covering algorithm is then used to find combinations that can cover all requirements [6, 16, 9]. The method suggested in this paper can find more complex compositions, though the ISTAR tool chain (SAM) is clearly further developed than that reported in this paper.

Horan [11] proposes the OWL-S [15] Web services ontology for a sensor ontology. While no ontology is given, and an OWL-S ontology for sensors would need to include many of the concepts defined in OntoSensor and the ontology discussed here, Horan's approach essentially argues that sensors and Web services are not different enough to require separate ontologies. There is not yet a consensus on the best way to model the semantics of services, and the ontology presented here is focused on sensor concepts and is structurally different from OWL-S, for example in the modelling of composite processes, but it is not yet clear if sensors and Web services, or other workflows and task models, should exist as semantically separate concepts or be unified in a single ontology that allows for different perspectives depending on the main features in any particular modelling task.

Clearly then, research on Web service specification and composition is relevant to the semantic specification of sensors. Mixtures of DL inference, structural similarity and information retrieval techniques, as in Klusch et al. [13], could thus be applied to sensor networks, as could service composition approaches [20]. This paper is the first time SPARQL and DL reasoning have been combined to search for compositions.

3 The Sensor Ontology

Although the ontology[3] is reasonably small with fewer than one hundred definitions of each of concepts and roles, it is still too large to give the complete definition here. Figure 1 pictorially represents the important parts of the ontology for this paper.

[3] A version is available at `http://www.w3.org/2005/Incubator/ssn/wiki/images/4/42/SensorOntology20090320.owl.xml`.

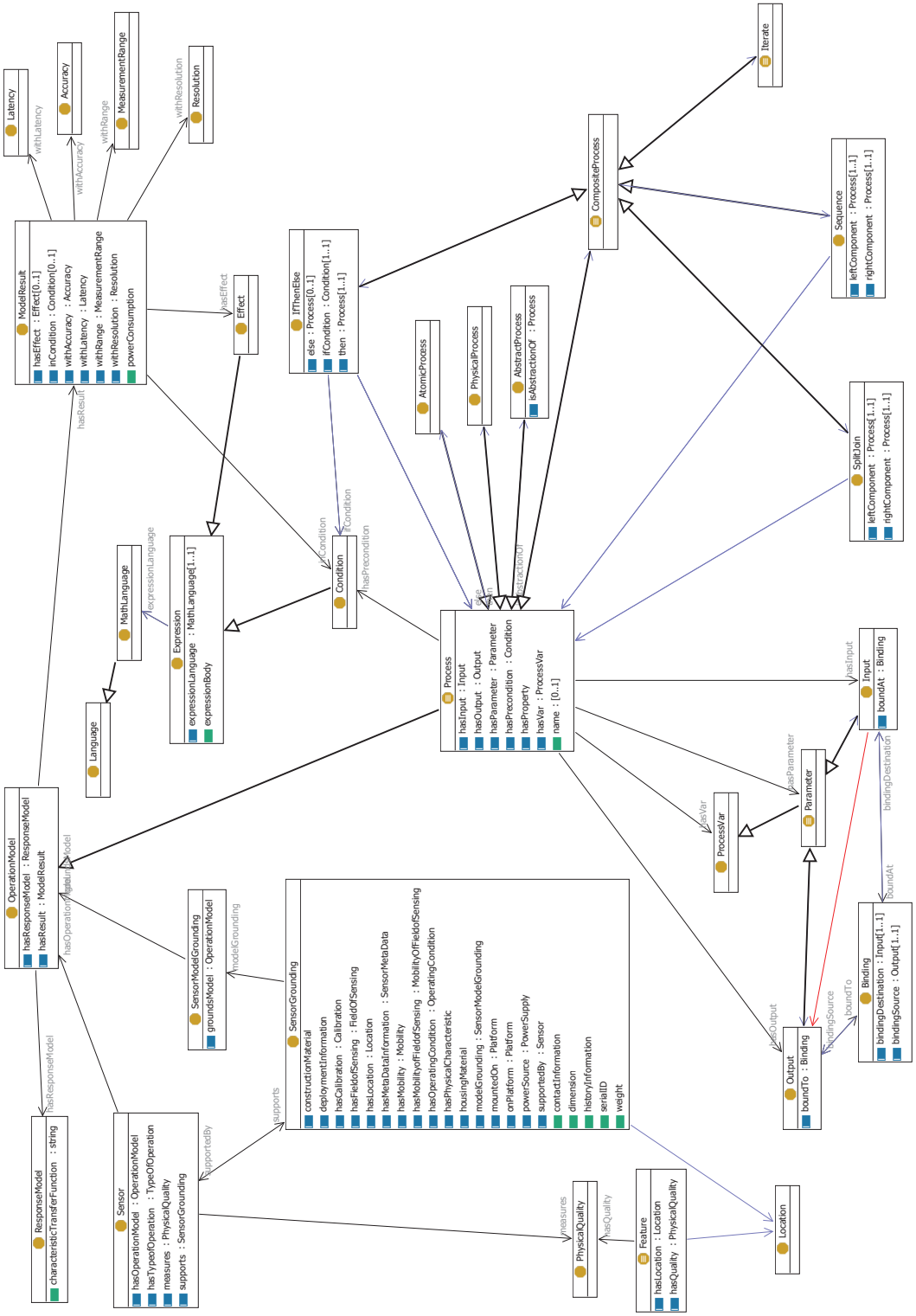

Fig. 1. Sensor Ontology Core Concepts

The ontology is built around the central notion of a sensor, and then three important clusters of concepts referenced by the sensor: domain concepts, abstract sensor properties and concrete properties. The abstract properties specify sensors' functions and capabilities. The concrete properties ground the abstract by providing, for example, the interface details to the functions. It is essentially the difference between specifying the properties of a sorting algorithm and giving a path to a binary. The abstract and concrete sections are independent in that neither requires the other: for example, the wind chill sensor in Section 4.2 specifies only the abstract properties — it is a schema for all wind chill sensors of this type and thus cannot specify the groundings, much as a specification of a sorting algorithm cannot also specify all implementations.

Separating concrete and abstract aspects of sensors means that descriptions of types of sensors, functions and the like can be shared among specifications and also that a single sensor type can have multiple concrete descriptions, promoting reuse and allowing for differences in deployment.

3.1 Domain Features

The domain ontology is left unspecified: a general sensor ontology cannot hope to capture all possible domains. Instead the ontology references abstract representations of real world entities (*Feature*, for example, a lake), which are not observed directly but through their observable qualities (*PhysicalQuality*, for example, temperature or depth). Similarly, units of measurement, locations and time are not described by the ontology, but rather deferred to appropriate authoritative sources. Any external ontologies can be included with OWL imports.

3.2 The concrete sensor

The *SensorGrounding* models the concrete realisation of a sensor. The grounding represents, in the case of an instrument, its physical implementation, including size, shape, materials and location. The grounding also models the concrete aspects of accessing data from the sensor, including the types and expected formats of input when calling functions, the format of output and other details for accessing the sensor (radio, network or physical access, for example).

3.3 The abstract sensor

Each sensor may have any number of *OperationModel*s describing the operations (functions) of the sensor, how the measurements are made and properties of the measurements.

Process is used to model the structure and data flow of an operation. An *AtomicProcess* models a single computation step. A *CompositeProcess* can model many ways of combining processes to form new processes. An *AbstractProcess* is used as a unit of abstraction, allowing a single specification to be described

at multiple levels of abstraction. The *Input* and *Output* of processes is unconstrained, allowing the model to describe physical processes as well as computations.

A *ResponseModel* is use to represent a sensor's responses to stimuli under various conditions. Each *OperationModel* may have a number of *Result*s which specify properties such as the function that the operation computes (its *Effect*), accuracy and latency and in various *Conditions*.

The ontology shows a number of important issues in specifying sensors and processes in OWL. Firstly, it is not possible to actually specify in OWL what a process does: for example, the *WindChillCalculation* process in Section 4.2 cannot express in OWL that it computes the wind chill formula given wind speed and temperature measurements. The processes can show the process structure, data flow, etc., but some details must be handled outside OWL. Here, an *Expression* in some *MathLanguage* is essentially a string in the given language that can be parsed and interpreted outside of OWL — this is the same handling as OWL-S. Also, the types of *Input*s, *Output*s and what a sensor *measures* are not able to be properly expressed in OWL.[4] Following OWL-S, the types of inputs and outputs are represented as the URIs of concepts, while *measures* follows the approach used in previous sensor ontologies.

The following examples illustrate the ontology further and demonstrate how the TBox and ABox are used to encode sensor types and sensor instances.

4 Examples

Sensors are encoded in the ontology using an interplay of TBox concepts, expressing what it means to be a particular type of sensor, and ABox individuals and roles, expressing properties of sensor instances. In the following a number of definitions are omitted but can be reasonably inferred from the context.

4.1 Vaisala WM30 wind sensor

The Vaisala WM30 wind sensor[5] measures wind speed and direction. As an example, it shows two important aspects of the ontology. First, the two different measurements of the WM30 show the capacity of the ontology to describe multiple capabilities of a single sensing device. Second, since the WM30 performs

[4] The definition $\exists measures.Temperature$, intending that the sensor measures temperatures, does not capture the correct meaning because OWL roles link individuals to individuals: there is no way to link to the concept *Temperature* — OWL-FULL in OWL1 can link individuals to concepts with roles, but cannot reason about such roles. Other sensor ontologies seem to take resolve this difficultly by interpreting the intention rather than the definition.

See also the Semantic Web Best Practice Descriptions discussion on this issue `http://www.w3.org/TR/swbp-classes-as-values/`

[5] See `http://www.vaisala.com/files/WM30_Brochure_in_English.pdf` for the the Vaisala WM30 data sheet.

differently under various conditions — it's accuracy is rated at $\pm 0.3m/s$ for wind speeds below $10m/s$, at $\pm 2\%$ for wind speeds up to $60m/s$ and isn't rated for wind speeds over $60m/s$ — it shows how the ontology can encode fine distinctions in operation. In this case showing, not only different accuracies in different conditions, but also accuracy expressed as both relative and absolute error. That the wind and direction measurements have different accuracies, direction being rated at $< \pm 3°$, further shows the detail that can be obtained.

The definitions begin with those for the domain of sensing.

$$\mathit{WindSpeed}, \mathit{WindDirection} \sqsubseteq \mathit{WindQuality} \sqsubseteq \mathit{PhysicalQuality}$$

Next, the main definitions of the WM30 sensor and its restrictions.

$$\begin{aligned}
&\mathit{VaisalaWM30} \sqsubseteq \mathit{Sensor}\\
&\mathit{VaisalaWM30} \sqsubseteq \exists \mathit{hasOperationModel}.\\
&\qquad\qquad \mathit{VaisalaWM30WindDirectionOperationModel}\\
&\mathit{VaisalaWM30} \sqsubseteq \exists \mathit{hasOperationModel}.\\
&\qquad\qquad \mathit{VaisalaWM30WindSpeedOperationModel}\\
&\mathit{VaisalaWM30} \sqsubseteq \exists \mathit{measures}.\mathit{WindDirection}\\
&\mathit{VaisalaWM30} \sqsubseteq \exists \mathit{measures}.\mathit{WindSpeed}\\
&\mathit{VaisalaWM30} \sqsubseteq \exists \mathit{supports}.\mathit{VaisalaWM30Grounding}
\end{aligned}$$

The grounding specifies physical properties and metadata, which includes the URL of Vaisala's data sheet. Some aspects from the data sheet, such as materials, data and electrical connections and power supply are not shown, but are similar to the following definitions. Note that complex values and units of measurement are not yet included in the ontology and are included as text in OWL data type properties, which are written *Concept* "*value*" here to avoid strings of role names; the intent should be clear from the context.

$$\begin{aligned}
&\mathit{VaisalaWM30Grounding} \sqsubseteq \mathit{Grounding}\\
&\mathit{VaisalaWM30Grounding} \sqsubseteq \exists \mathit{hasOperatingCondition}.\\
&\qquad\qquad \mathit{TemperatureRange}\ \text{“}-40 \ldots +55\ \ C\text{”}\\
&\qquad \ldots
\end{aligned}$$

The definition of the WM30's sensing capabilities show how accuracy is encoded (again currently as text while our model of values of units are being

constructed). Only wind speed is shown, direction is similar.

$$
\begin{aligned}
&\mathit{VaisalaWM30WindSpeedOperationModel} \sqsubseteq \\
&\qquad \exists \mathit{hasResult}. \\
&\qquad\quad \exists \mathit{withAccuracy}.\mathit{AbsoluteAccuracy}\ \text{“}\pm 0.3m/s\text{”} \\
&\qquad\quad\ \ \mathit{and} \\
&\qquad\quad \exists \mathit{inCondition}.\mathit{expressionBody}\ \text{“}0.4 \ldots 10m/s\text{”})) \\
&\qquad \mathit{and} \\
&\qquad \exists \mathit{hasResult}. \\
&\qquad\quad \exists \mathit{withAccuracy}.\mathit{RelativeAccuracy}\ \text{“}\pm 2\%\text{”} \\
&\qquad\quad\ \ \mathit{and} \\
&\qquad\quad \exists \mathit{inCondition}.\mathit{expressionBody}\ \text{“}10 \ldots 60m/s\text{”})) \\
&\qquad \mathit{and} \\
&\qquad \exists \mathit{hasResult}. \\
&\qquad\quad \forall \mathit{withRange}.\mathit{MeasurementRange}\ \text{“}0.5 \ldots 60m/s\text{”} \\
&\qquad \mathit{and} \\
&\qquad \exists \mathit{hasOutput}. \\
&\qquad\quad \exists \mathit{parameterType}.\mathit{URI}\ \text{“}\ldots\#\mathit{WindSpeed}\text{”} \\
&\qquad \ldots
\end{aligned}
$$

Other aspects from the data sheet such as the distance constant, starting threshold, transducer output and even the characteristic transfer function ($U = -0.24 + (0.699) \times F$) are encoded as part of the *ResponseModel*.

There are two options for the WM30 wind direction sensor, the WMS301 and the WMS302, with 355° and 360° measurement ranges respectively. The *VaisalaWM30WindDirectionOperationModel* is defined as the disjoint union of these options, with the two sub-concepts inheriting all their properties from *VaisalaWM30WindDirectionOperationModel* except the definition of the measurement range.

These TBox definitions capture what it is to be a Vaisala WM30. The definitions are a schema defining the class of all WM30 sensors in terms of properties that are static for all WM30 instances. WM30 individuals are recorded in the ABox and can be specified with a definition as minimal as asserting an individual to be a WM30.

$$\mathit{VaisalaWM30}(\mathit{wm30}_1)$$

The remainder of the individuals defining $\mathit{wm30}_1$, such as its accuracy and operations, must exist — they are implied by the specification — but the TBox definitions make it unnecessary to include them, unless a particular WM30 instance has properties that require further specification. Properties such as location are unique to each individual and are generally asserted for each sensor instance.

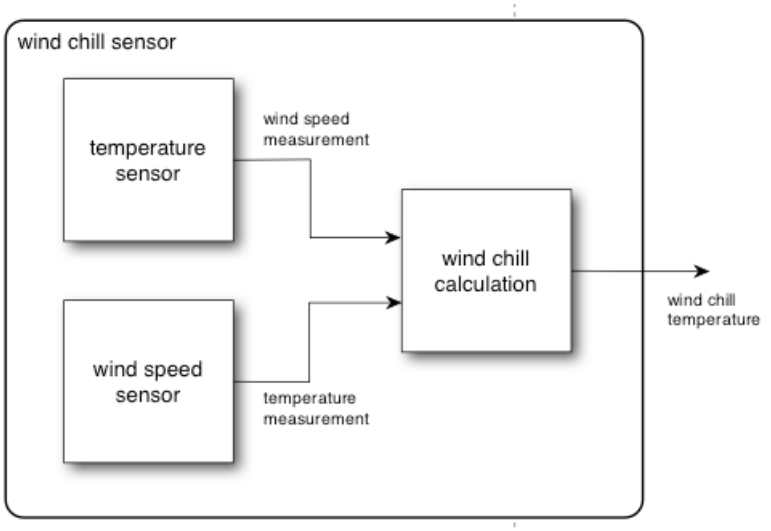

Fig. 2. Wind chill sensor composed from a temperature sensor, a wind speed sensor and a calculation for wind chill.

4.2 SensorML wind chill sensor

The definitions for the wind chill sensor in this section are based on SensorML examples.[6] Ambient temperature, T, and wind speed, V, measurements are required to calculate wind chill using the following formula.[7]

$$35.74 + (0.6215 * T) - 35.75 * (V^{0.16}) + 0.4275 * T * (V^{0.16})$$

Figure 2 shows the structure of the wind chill sensor. There are, of course, any number of modelling options for such a process, those chosen here are meant to demonstrate aspects of modelling and the ontology and aren't prescriptive ways of modelling processes.

First, simple definitions for domain concepts and the basic definition of a wind chill sensor.

$$\begin{aligned} \mathit{WindChill} &\sqsubseteq \mathit{WindQuality} \\ \mathit{WindChillSensor} &\sqsubseteq \mathit{Sensor} \\ \mathit{WindChillSensor} &\sqsubseteq \exists \mathit{measures}.\mathit{WindChill} \\ \mathit{WindChillSensor} &\sqsubseteq \exists \mathit{hasOperationModel}.\mathit{WindChillSensorOutput} \end{aligned}$$

Next, a temperature sensor is defined as a sensor that is also a process to demonstrate that the two concepts can be conflated, giving extra expressive

[6] `http://vast.uah.edu/downloads/documents/Creating_SensorML_Process_Models_preV1.0.pdf`

[7] This is the formula used by the United States National Weather Services, see `http://www.wrh.noaa.gov/slc/projects/wxcalc/windChill.pdf`

power and the option of defining the sensor itself as the process of making an observation.

$$
\begin{aligned}
&\mathit{TemperatureSensor} \sqsubseteq \mathit{Sensor}\\
&\mathit{TemperatureSensor} \sqsubseteq \exists \mathit{measures}.\mathit{Temperature}\\
&\mathit{TemperatureSensor} \sqsubseteq \exists \mathit{hasOperationModel}.\mathit{Self}
\end{aligned}
$$

Then, the calculation of wind chill is defined as a process, showing one option for how the formula itself could be encoded; though, MathML would be a better choice if the aim is automatically composing and orchestrating such a process, rather than human readability. It is also possible to attach a reference to a program that implements the calculation. Wind speed is assumed to come from a process with the only important property that it outputs a wind speed measurement. Types of inputs and outputs are removed throughout this example, see Figure 2 for an indication of what the types are for the various processes and the previous example for how they are defined. The *SplitJoin* process models parallel execution of its left and right components, which in this example could be either speed or temperature — the disjunction is added to indication that the wind speed and temperature process can accept processes with the sensors specified in either of the two possible orderings.

$$
\begin{aligned}
&\mathit{WindChillCalculation} \sqsubseteq \mathit{Process}\\
&\mathit{WindChillCalculation} \sqsubseteq \forall \mathit{hasResult}.\\
&\qquad \exists \mathit{hasEffect}.\exists \mathit{expressionLanguage}.\mathit{mathText}\\
&\qquad\qquad \mathit{and}\\
&\qquad\qquad \mathit{expressionBody}\\
&\quad \text{“}\mathit{35.74} + (\mathit{0.6215} * T) - \mathit{35.75} * (V \hat{} \mathit{0.16}) + \mathit{0.4275} * T * (V \hat{} \mathit{0.16})\text{”}\\
&\mathit{WindSpeedandTemperature} \sqsubseteq \mathit{SplitJoin}\\
&\mathit{WindSpeedandTemperature} \sqsubseteq (\exists \mathit{leftComponent}.\mathit{TemperatureSensorProcess}\\
&\qquad\qquad \mathit{and}\\
&\qquad\qquad \exists \mathit{rightComponent}.\mathit{WindSpeedOutput})\\
&\qquad\qquad \mathit{or}\\
&\qquad\qquad (\exists \mathit{leftComponent}.\mathit{WindSpeedOutput}\\
&\qquad\qquad \mathit{and}\\
&\qquad\qquad \exists \mathit{rightComponent}.\mathit{TemperatureSensorProcess})\\
&\mathit{WindChillSensorOutput} \sqsubseteq \mathit{Sequence}\\
&\mathit{WindChillSensorOutput} \sqsubseteq \exists \mathit{leftComponent}.\mathit{WindSpeedAndTemperature}\\
&\mathit{WindChillSensorOutput} \sqsubseteq \exists \mathit{rightComponent}.\mathit{WindChillCalculation}
\end{aligned}
$$

The definitions to this point specify the components of the wind chill sensor and their properties, but do not yet connect the outputs of the wind speed and temperature sensors to the inputs of the calculation. The idea is simple enough, but cannot be expressed precisely in OWL. In specifying bindings in the TBox, it is possible to state that the wind chill sensor connects inputs and outputs of the correct type to the correct type of processes; however, TBox definitions can't state that the bindings connect the outputs and inputs of the actual left and right components, just any components of the correct type. The same is true of OWL-S and its bindings. The situation in the ABox is somewhat different where the precise specifications of individuals can be made by actually connecting the left and right individual's inputs and outputs. It is another case of needing to interpret the intent of the TBox definitions rather than the DL; however, this has obvious consequences in classification, where the DL reasoner applies the DL semantics, not the intent of the definition.

Similarly to the previous example the TBox definitions here define a schema for wind chill sensors, under which any number of actual, or virtual, sensors can be classified.

5 Reasoning

Reasoning techniques such as DL consistency checking, simple SPARQL queries and LP rules for sensors and data have been published previously and are not discussed further here (see §2). Classification is briefly discussed first, then non-DL reasoning and lastly using DL and SPARQL to automatically compose virtual instruments.

The sensor ontology as presented is an ontology of sensor capabilities. Another view of a sensor ontology is sensors organised into hierarchies of sensing concepts. OWL can classify the sensors into a hierarchy given suitable definitions: for example,

$$WindSpeedSensor \equiv \exists measures.WindSpeed$$

would allow the reasoner to infer $VaisalaWM30 \sqsubseteq WindSpeedSensor$. Similarly, sensors could be classified according to accuracy or other properties. This simple use of classification demonstrates that hierarchies of capabilities and hierarchies of sensor types are compatible ways of viewing the same ontology.

5.1 External reasoners

We used the OWL-API[8] interface to give a Java reasoner access to the ABox. We expressed the Region Connection Calculus (RCC) in OWL. Our OWL embedding of RCC is based on Grütter et al.'s [10], who show that full RCC inference cannot be done in OWL. In our application, files describing sensors are parsed (we intend to allow SensorML and other descriptions) and individuals asserted into the ABox. When required, the Java RCC reasoner harvests quantitative

[8] `http://owlapi.sourceforge.net/`

information (numeric properties) from the ABox (sensor locations, regions of sensing, etc) derives new facts about the regions and asserts qualitative information (RCC relations) back into the ABox. The newly asserted facts are then available to the OWL reasoner. If new sensors are discovered, the procedure can be re-run to generate yet further facts.

This RCC example might be possible with rules; however, as the reasoning becomes more complex or if it involves existing reasoning routines (or computational models), rules become cumbersome. We intend to use this style of interaction to allow the results of a range of reasoning engines and computational models to be made available for DL inference.

The RCC example works because there is no interaction between OWL and RCC — they simply use the same fact base. The differences between closed- and open-world reasoning can be made to fit into a consistent reasoning package with rules (though this does not imply that any given set of rules is correct), and in combining other reasoning with OWL it is important to ensure that the combined reasoning is consistent. If the RCC and OWL assertions were dependent on each other then the situation is much more complex (see Baader et al. [2] for a more detailed discussion on the issues of combining logics).

5.2 Composition

The definitions of the wind chill sensor can be seen as defining a search goal. The search goal is to find existing sensors that can be composed into a wind chill sensor. Given, the TBox definitions of the WM30 and the wind chill sensor, and ABox definitions for

$$\begin{aligned} &\mathit{WindSpeedOutput}(\mathit{ws}_1) \\ &\mathit{TemperatureSensor}(\mathit{ts}_1) \\ &\mathit{WindChillCalculation}(\mathit{calc}_1) \end{aligned}$$

and perhaps many other sensors, but no wind chill sensor individuals, the goal is to construct all the possible wind chill sensors (for the moment constraints like co-location are ignored).

Running a DL reasoner on the definitions shows that *WindChillSensor* is satisfiable — it can have instances. The reasoner can also prove if there are instances composed from existing sensors. If the possible instances for the wind speed, temperature and calculation are enumerated, the reasoner is forced to build a model demonstrating the satisfiability of *WindChillSensor* using only existing sensors. For example, the satisfiability of *WindChillSensor* after asserting

$$\begin{aligned} &\mathit{WindSpeedOutput} \equiv \{\mathit{ws}_1\} \\ &\mathit{TemperatureSensor} \equiv \{\mathit{ts}_1\} \\ &\mathit{WindChillCalculation} \equiv \{\mathit{calc}_1\} \end{aligned}$$

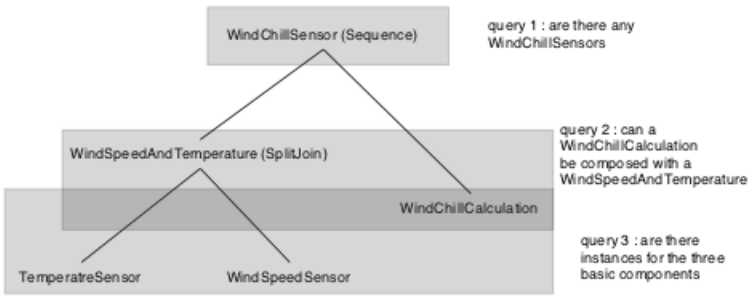

Fig. 3. Parse tree and queries.

shows that there is a composition of these sensors that makes a wind chill sensor. If there were many more sensors, it wouldn't be possible to determine which sensors the reasoner had used in building its model. If the proofs of satisfiability could be extracted from DL reasoners, it could be possible to find compositions by analysing the proofs; however, DL reasoners only build certain kinds of models, so not all compositions could be found this way.

Compositions can be found using SPARQL-DL, a subset of SPARQL with DL inference. SPARQL-DL will only return results for existing instance, not inferred instance, though the inferred instance can be used as bnodes in SPARQL queries (bnodes, or blank nodes are unnamed nodes in RDF graphs; generally, bnodes are explicitly stated as part or the RDF graph, but in SPARQL-DL they needn't be present if they are implied to exist by the ontology). Hence, simply converting the goal (wind chill sensor concept) to a SPARQL-DL query will not find compositions. However, the goal is structured enough that a parse tree can be constructed for the process. Issuing a query for each combination in the parse tree will find all possible compositions (see Figure 3).

For ws_1, ts_1 and $calc_1$, the relevant query is a simple selection on the sensor types (query 3 in Figure 3).

```
SELECT ?ts ?ws ?calc
WHERE {
      ?ts rdf:type sensor:TemperatureSensor .
      ?ws rdf:type sensor:WindSpeedOutput .
      ?calc rdf:type sensor:WindChillCalculation .
}
```

SPARQL-DL returns a result for each binding of variables that satisfies the query. Compositions are then constructed by following the specification to link the individuals. In this case linking the inputs and outputs of ws_1, ts_1 and $calc_1$ and creating new individuals for the bindings, the split join process and the wind chill sensor.

Since this method follows OWL semantics, issues such as searching correctly for sub-concepts are handled by SPARQL-DL. Constraints, such as co-location

or accuracy restrictions on sensors, can be added to the query as SPARQL filters. Types, however are still problematic. In this query there is no requirement to match the input and output types of the processes — the definitions are specific enough to not require it. In general, the query requires constraints ensuring that the input and output types match, which can be handled by simply matching the stated types. However, it is correct to match an output if it is a sub-type of the required input. But this is not easily handled in the query because the definition of a type is a datatype URI, which is not amenable to SPARQL-DL reasoning, so a simple subClassOf constraint can't be added to the query. The required type checking can be handled by interpreting the query results as potential compositions and, for each, checking that the input and output types match correctly.

6 Discussion and Conclusion

This paper has discussed an OWL ontology for describing details of sensors. The ontology is more expressive than previous sensor ontologies as it can express complex compositions and fine details of the function and results of sensors and processes. The ontology can also encode much of the information in SensorML documents. Since OWL is unable to express concepts such as the function computed by a process, the ontology includes the ability to express these aspects in other languages. This capability is based on similar features in OWL-S, as is some of the structure of the ontology; however, for example, the modelling of processes, and in particular composite process, does not follow the OWL-S model.

The ontology is designed such that domain semantics, units of measurement, time and time series, and location and mobility ontologies can be easily included with the usual OWL import mechanism.

The SPARQL-DL method of composition is simple, but powerful enough that it will successfully find compositions even for complex processes. Its biggest drawback is the number of queries that need to be issued as the goal gets more complex; a search strategy is needed if not all compositions are required. We have not yet experimented with its performance.

Sensors cannot be modelled precisely using OWL. Issues involving numbers or describing functions require external mechanisms, but the issues in encoding type information, for example, can have subtle consequences. Here the input and output types of processes were modelled as URIs which captures the correct meaning, but can no longer be reasoned about inside OWL, though they can be processed externally and then used in DL reasoning. The alternate choice of simply using a role, as with *measures* here, warps the logical interpretation somewhat, though retains the possibility of some reasoning in OWL. However, attempting to model inputs and outputs with this method would still require external reasoning because, for example, a query that asks for processes whose inputs and outputs match requires a quantification over the types that can't be expressed in OWL.

In cases such as binding inputs to outputs the TBox isn't able to limit the interpretation to only the intended models. To some extent an external mechanism could be used to interpret such definitions, but it is not clear in general how to reconcile the intention with the definition, or a reasoner's handling of it. In essence, each of these issues implies some trade-off between decidability and expressivity.

In future work, we intend to further investigate how Web service composition can be used to automatically construct virtual sensors. We intend to use our ontology in data integration and scientific workflows. We also plan to extend the ontology to allow descriptions of values and their units as well as observations and results.

Acknowledgements

This research was conducted as part of the CSIRO Water for a Healthy Country National Research Flagship and the Sensor Network Technologies Theme.

The Tasmanian ICT Centre is jointly funded by the Australian Government through the Intelligent Island Program and CSIRO. The Intelligent Island Program is administered by the Tasmanian Department of Economic Development, Tourism and the Arts.

Khoi-Nguyen Tran worked in this research as a CSIRO summer research scholar and as an honours student at the Australian National University.

References

1. OWL 2 Web Ontology Language. W3C Candidate Recommendations and Working Drafts, 2009. Available at `http://www.w3.org/2007/OWL/wiki/OWL_Working_Group`.
2. F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research*, 2002.
3. M. Botts and A. Robin. OpenGIS Sensor Model Language (SensorML) implementation specification. OpenGIS Implementation Specification OGC 07-000, The Open Geospatical Consortium, July 2007.
4. M. Calder, R. Morris, and F. Peri. Machine reasoning about anomalous sensor data. In *International Conference on Eccological Informatics*, 2008.
5. M. Compton, C. Henson, H. Neuhaus, L. Lefort, and A. Sheth. A survey of the semantic specification of sensors. In *2nd International Semantic Sensor Networks Workshop*, 2009.
6. G. de Mel, M. Sensoy, W. Vasconcelos, and A. Preece. Flexible resource assignment in sensor networks: a hybrid reasoning approach. In *1st International Workshop on the Semantic Sensor Web*, 2009.
7. M. Eid, R. Liscano, and A. E. Saddik. A novel ontology for sensor networks data. In *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, 2006.
8. M. Eid, R. Liscano, and A. E. Saddik. A universal ontology for sensor networks data. In *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, 2007.

9. M. Gomez, A. Preece, M. Johnson, G. de Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. Porta, and D. Pizzocaro. An ontology-centric approach to sensor-mission assignment. In *16th International Conference on Knowledge Engineering and Knowledge Management*, 2008.
10. R. Grütter, T. Scharrenbach, and B. Bauer-Messmer. Improving an RCC-derived geospatial approximation by OWL axioms. In *7th International Conference on the Semantic Web*, 2008.
11. B. Horan. The use of capability descriptions in a wireless transducer network. Technical report, Sun Microsystems Laboratories, 2005. SMLI Technical Report TR-2005-131.
12. J. Kim, H. Kwon, D. Kim, H. Kwak, and S. Lee. Building a service-oriented ontology for wireless sensor networks. In *7th IEEE/ACIS International Conference on Computer and Information Science*, 2008.
13. M. Klusch, B. Fries, and K. Sycara. OWLS-MX: a hybrid Semantic Web service matchmaker for OWL-S services. *International Journal of Web Semantics*, 7(2), 2009.
14. J. Liu and F. Zhao. Towards semantic services for sensor-rich information systems. In *IEEE/CreateNet International Workshop on Broadband Advanced Sensor Network*.
15. OWL-S Coalition. OWL-S: Semantic Markup for Web Services. W3C Member Submission, November 2004. Available at `http://www.w3.org/Submission/OWL-S/`.
16. A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, G. Pearson, T. Pham, and T. Porta. Matching sensors to missions using a knowledge-based approach. In *SPIE Defense Transformation and Net-Centric Systems*, 2008.
17. D. Russomanno, C. Kothari, and O. Thomas. Building a sensor ontology: a practical approach leveraging ISO and OGC models. In *2005 International Conference on Artificial Intelligence (vol 2)*, 2005.
18. D. Russomanno, C. Kothari, and O. Thomas. Sensor ontologies: from shallow to deep models. In *37th Southeastern Symposium on System Theory*, 2005.
19. E. Sirin and B. Parsia. SPARQL-DL: SPARQL query for OWL-DL. In *3rd OWL Experiences and Directions Workshop*, 2007.
20. J. Su, editor. *Data Engineering Bulleten, Special Issue on Semantic Web Services: Composition and Analysis*, volume 31. IEEE Computer Society, 2008.
21. K. Thirunarayan, C. Henson, and A. Sheth. Situation awareness via abductive reasoning for semantic sensor data: a preliminary report. In *International Symposium on Collaborative Technologies and Systems*, 2009.
22. K. Whitehouse, F. Zhao, and J. Liu. Semantic streams: a framework for composable inference over sensor data. In *3rd European Workshop on Wireless Sensor Network*, 2006.