2013

A Semantics-Based Approach to Machine Perception

Cory Andrew Henson
Wright State University

A Semantics-based Approach to Machine Perception

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

CORY ANDREW HENSON
B.A., University of Georgia, 2005

2013
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

December 14, 2013

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Cory Andrew Henson ENTITLED A Semantics-based Approach to Machine Perception BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Amit P. Sheth, Ph.D.
Dissertation Director

Arthur A. Goshtasby, Ph.D.
Director, Computer Science Ph.D. Program

R. William Ayres, Ph.D.
Interim Dean, Graduate School

Committee on Final Examination

Amit P. Sheth, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Payam Barnaghi, Ph.D.

Satya S. Sahoo, Ph.D.

John Gallagher, Ph.D.

Abstract

Henson, Cory Andrew. Ph.D., Computer Science and Engineering Ph.D. Program, Wright State University, 2013. *A Semantics-based Approach to Machine Perception*.

Machine perception can be formalized using semantic web technologies in order to derive abstractions from sensor data using background knowledge on the Web, and efficiently executed on resource-constrained devices.

Advances in sensing technology hold the promise to revolutionize our ability to observe and understand the world around us. Yet the gap between observation and understanding is vast. As sensors are becoming more advanced and cost-effective, the result is an avalanche of data of high volume, velocity, and of varied type, leading to the problem of too much data and not enough knowledge (i.e., insights leading to actions). Current estimates predict over 50 billion sensors connected to the Web by 2020.[1] While the challenge of data deluge is formidable, a resolution has profound implications. The ability to translate low-level data into high-level abstractions closer to human understanding and decision-making has the potential to disrupt data-driven interdisciplinary sciences, such as environmental science, healthcare, and bioinformatics, as well as enable other emerging technologies, such as the Internet of Things.

The ability to make sense of sensory input is called *perception*; and while people are able to perceive their environment almost instantaneously, and seemingly without effort, machines

[1] http://share.cisco.com/internet-of-things.html

continue to struggle with the task. Machine perception is a hard problem in computer science, with many fundamental issues that are yet to be adequately addressed, including: (a) annotation of sensor data, (b) interpretation of sensor data, and (c) efficient implementation and execution. This dissertation presents a semantics-based machine perception framework to address these issues.

The tangible primary contributions created to support the thesis of this dissertation include the development of a Semantic Sensor Observation Service (SemSOS) for accessing and querying sensor data on the Web, an ontology of perception (Intellego) that provides a formal semantics of machine perception and reasoning framework for the interpretation of sensor data, and efficient algorithms for the machine perception inference tasks to enable interpretation of sensor data on resource-constrained devices, such as smart phones. Each of these contributions has been prototyped, evaluated, and applied towards solving real-world problems in multiple domains including weather and healthcare.

Table of Contents

List of Figures

List of Tables

Acknowledgement

Of primary importance, I would like to acknowledge the giants. Their shoulders on which we stand have afforded an inspirational view. My journey through graduate school has consisted of scanning a horizon that only a few have the pleasure to enjoy; it has been an extraordinary experience.

During this journey, I have been lucky enough to have two guides that have not only steered my research, but more importantly, taught me to think and to be. First, I would like to thank my advisor, Amit Sheth, for his guidance and vision. Professor Sheth has taught me to look, to see, to envision a future that could be. His ability to not only peer into the future, but to push the present to meet that vision is truly inspirational. I will continue to strive for such clarity and strength. I would also like to thank my (unofficial) advisor, T.K. Prasad, for teaching me that quality research exists at the point where vision and reality intersect, and that true understanding manifests in simple terms. Professor Prasad has the ability to explain even the most complex subjects in the simplest of phrases. It is a rare and powerful gift. I will continue to chip away at the complexity of the world until I too can see its simplicity.

I would also like to express my gratitude to the rest of my dissertation committee: Satya Sahoo, Payam Barnaghi, and John Gallager. I have benefitted greatly from many interactions with them and appreciate their continual support and encouragement. Dr. Satya Sahoo has played many different roles in my life, beginning as my boss, then my colleague, my roommate, and finally as my dissertation committee member. But always he will be my friend. Dr. Payam Barnaghi introduced me to the wonderful world of *Internet of Things*. I enjoy this idea so much that I have decided to base by career around the idea (so far). For this, I will always be grateful.

I would like to acknowledge the role played by the Air Force Research Lab (AFRL) and the Defense Intelligence Agency (DIA). Representing the AFRL, Bill McQuay, Juan Carbonell, and Kenny Littlejohn engaged in a leap of faith by supporting my research from the very beginning. Their support and encouragement gave me the confidence to persist in exploring our research vision. Representing the DIA, Tammy Countryman and Robin de la Vega provided me with an opportunity to see how my research could be effectively used for the betterment of our country. Tammy opened the door for this opportunity through an internship at the National MASINT Office (NMO) and Robin laid out the welcome mat by offering a research staff position at Riverside Research. While I have decided to shift my focus away from government sponsored research, I will always remember the lessons learned.

I owe a great debt to the rotating cast and crew making up the Semantic Sensor Web team. In particular, I would like to thank Pramod Anantharam, Sujan Perera, Harshal Patni, and Josh Pschorr. I have learned many invaluable lessons from each of them through the years, and am continually amazed at their inventiveness, hard work, and dedication. I'm sure they will be supremely successful in their future research and careers. For the remaining members of Kno.e.sis, I would like to thank them for their inspiration and friendship. I could not imagine traveling along the path to graduation with a better group of people.

Last, but certainly not least, I would like to thank my friends and family. My parents, Pat and Carl Henson, are the hardest working and most selfless people I know. They have always believed that I have the potential for success and provided enduring love and encouragement that will never be forgotten. To make them proud is the highest honor of my life.

Dedicated to

My grandmother, Dot

1. Introduction

Machine perception can be formalized using semantic web technologies in order to derive abstractions from sensor data using background knowledge on the Web, and efficiently executed on resource-constrained devices.

Machine perception is the systematic automation of computing machines to sense and interpret the contents of their environment. The automation of this task is a hard problem in computer science, with many fundamental issues that are yet to be adequately addressed, including: (a) annotation of sensor data, (b) interpretation of sensor data, and (c) efficient implementation and execution. The goal of this dissertation is to present a semantics-based machine perception framework, which demonstrates the validity of the thesis stated above. Toward this end, the chapters of this dissertation will discuss the technologies and methodologies of this framework for semantically annotating, querying, and interpreting sensor data. More concretely, the goals are as follows:

1. Develop techniques for semantically annotating sensor descriptions and sensor observation data on the Web to enable advanced integration, query, and inference. This goal is discussed in Chapter 2: Semantic Sensor Web.
2. Develop techniques for interpreting semantically annotated sensor observation data to derive actionable intelligence and situational awareness (i.e., high-level abstractions), using background knowledge on the Web. This goal is discussed in Chapter 3: Semantic Perception.

3. Develop techniques to enable the efficient and scalable interpretation of semantically annotated sensor observation data on resource-constrained devices. This goal is discussed in Chapter 4: Intelligence at the Edge.
4. Develop a prototype application to demonstrate the utility of the semantics-based machine perception framework in a real-world scenario. This goal is discussed in Chapter 5: Knowledge-enabled Healthcare.

1.1. Contributions

The tangible primary contributions created to support the thesis of this dissertation are discussed next.

Semantic Sensor Observation Service (SemSOS) – Sensor Observation Service (SOS) is a Web service specification defined by the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) group in order to standardize the way sensors and sensor data are discovered and accessed on the Web. This standard goes a long way in providing interoperability between repositories of heterogeneous sensor data and applications that use this data. Many of these applications, however, are ill equipped at handling raw sensor data as provided by SOS and require actionable knowledge of the environment in order to be practically useful. There are two approaches to deal with this obstacle, make the applications smarter or make the data smarter. To better enable sharing, integration, and interpretation of the data across different applications, we propose the latter option and accomplish this by leveraging semantic technologies in order to provide and apply more meaningful representation of sensor data. More specifically, we are modeling the domain of sensors and sensor observations in a suite of ontologies, adding semantic annotations to the sensor data (that is offered by the Web Service), using the ontology models to reason over sensor observations, and extending an open source SOS implementation with our semantic knowledge base. This semantically enabled SOS, or SemSOS, provides the ability to query high-level knowledge of the environment as well as low-level raw sensor data.

Ontology of Perception (Intellego) – Today, many sensor networks and their applications employ a brute force approach to collecting and analyzing sensor data. Such an approach often wastes valuable energy and computational resources by unnecessarily tasking sensors and generating observations of minimal use. People, on the other hand, have evolved sophisticated mechanisms to efficiently perceive their environment. One such mechanism includes the use of background knowledge to determine what aspects of the environment to focus our attention. In this dissertation, we develop an ontology of perception, IntellegO, that may be used to more efficiently convert observations into perceptions. IntellegO is derived from cognitive theory and provides a formal semantics of machine perception. We then present an implementation that iteratively and efficiently processes low level, heterogeneous sensor data into higher-level abstractions through use of the perception ontology and domain specific background knowledge. The abstractions thus computed transform raw, low level data into contextually relevant and actionable knowledge. As a demonstration of this capability, we evaluate IntellegO by collecting and analyzing observations of weather conditions on the Web, and show significant resource savings in the generation and storage of perceptual knowledge.

Efficient Algorithms for Perceptual Inference – A primary challenge of machine perception is to define efficient computational methods to derive high-level knowledge from low-level sensor observation data. Emerging solutions are using Semantic Web ontologies for expressive representation of concepts in the domain of sensing and perception, which enable advanced integration and interpretation of heterogeneous sensor data. The computational complexity of the Web Ontology Language (OWL), however, seriously limits its applicability and use within resource-constrained environments, such as mobile devices. To overcome this issue, we employ

OWL to formally define the inference tasks needed for machine perception - explanation and discrimination - and then provide efficient algorithms for these tasks, using bit-vector encodings and operations. The applicability of our approach to machine perception is evaluated on a smart-phone mobile device, demonstrating dramatic improvements in both efficiency and scale.

Knowledge-enabled Healthcare – Knowledge-enabled Healthcare, or kHealth, is a platform that integrates data from passive and active sensing (including both machine and human sensors) with background knowledge from domain ontologies, semantic reasoning, and mobile computing environments to help people make decisions to improve health, fitness, and wellbeing. kHealth is a real-world application that utilizes technology founded in this dissertation – i.e., the semantic annotation of sensor data (Semantic Sensor Web), interpretation of sensor data (Semantic Perception), and efficient algorithms for interpreting sensor data on resource-constrained devices (Intelligence at the Edge) – to enable advanced healthcare applications. Currently, the application of kHealth towards the management of several disorders, including chronic heart disease and asthma, is being investigated in collaboration with clinicians.

1.2. Chapter Overview

The dissertation is organized as follows: Chapter 2, Semantic Sensor Web, discusses the technologies for the representation, annotation, query, and inference of sensor data on the Web. The chapter begins with background on the technologies and standards developed and utilized by the Open Geospatial Consortium's Sensor Web Enablement and the World Wide Web Consortium's Semantic Web. Next, the development of a semantic sensor observation service is discussed, which allows for the annotation and query of sensor descriptions and sensor observation data on the Web. Finally, the Semantic Sensor Network Ontology, developed by the W3C Incubator Group on Semantic Sensor Networks (SSN-XG), is introduced.

Chapter 3, Semantic Perception, discusses a methodology for interpreting sensor data. The chapter begins with a discussion of cognitive models of perception, which form the basis of a model of perception used to interpret the sensor data. Next, an ontology of perception, Intellego, is discussed which provides a formal semantics of machine perception. More specifically, the concepts and inference tasks are formalized in set-theoretic notation. Several of the inference tasks of Intellego are then formalized in the Web Ontology Language, which allows improved integration with background knowledge on the Web.

Chapter 4, Intelligence at the Edge, discusses the development of efficient and scalable algorithms that provide the ability to interpret semantically annotated sensor data on resource-constrained devices. The chapter begins with a description of the method of translating data from a high-level semantic representation to a low-level bit-vector representation, and vice-versa. Next, bit-vector algorithms for the primary perceptual inference tasks – explanation and discrimination – are discussed. These algorithms are evaluated on a mobile device, showing orders-of-magnitude in both efficiency and scalability.

Chapter 5, Knowledge-enabled Healthcare, discusses the application of the technologies and methodologies described the previous chapters for improving health, fitness, and wellbeing. This chapter begins with a motivating scenario in the domain of cardiology. Next, an application is discussed that uses both passive and active sensing, from both machine and human sensors, to monitor a persons physiology in order to determine the persons health condition. This application is developed with the aim of helping to reduce hospital readmissions of patients with Acute Decompensated Heart Failure.

Finally, Chapter 6 concludes with a summary of the research work presented in the dissertation and final remarks.

2. Semantic Sensor Web

In March 2008, heavy rainstorms across the Midwestern region of the US caused many rivers to breach their banks. Residents of Valley Park, a small town along the Meramec River, Missouri, had to decide whether to rely on a newly constructed levee or abandon their homes for higher ground [Salter08]**.** Although the levee held, many chose the latter option and fled their homes; it was a chaotic situation that might have been avoided through access to better situational knowledge regarding the current water pressure and the levee's structural integrity. Pressure sensors embedded in the levee could have provided accurate real-time information that allowed residents to make informed decisions about the safety of the levee, their homes, and themselves. This scenario demonstrates the increasingly critical role of sensors that collect and distribute observations of our world in our everyday lives.

In recent years, sensors have been increasingly adopted by a diverse array of disciplines, such as meteorology for weather forecasting and wildfire detection (www.met.utah.edu/mesowest/), civic planning for traffic management (www.buckeyetraffic.org/), satellite imaging for earth and space observation (http://vast.uah.edu/), medical sciences for patient care using biometric sensors (www.liebertonline.com/doi/abs/10.1089/109350703322682531), and homeland security for radiation and biochemical detection at ports (www.msnbc.msn.com/id/8092280). Sensors are thus distributed across the globe, leading to an avalanche of data about our environment. The rapid development and deployment of sensor technology involves many different types of sensors, both remote and in situ, with diverse capabilities such as range, modality, and maneuverability. Today, it is possible to use sensor networks to detect and identify a multitude of observations, from simple phenomena to complex events and situations. The lack of integration and communication between these networks, however, often isolates important data streams and intensifies the existing problem of too much data and not enough knowledge. With a view to addressing this problem, this chapter discusses a Semantic Sensor Web (SSW) in which sensor data is annotated with semantic metadata to increase interoperability as well as provide contextual information essential for situational knowledge.

2.1. Background

Semantic Sensor Web is reliant on two sets of standardizations, (1) the Sensor Web Enablement languages and service interface specifications defined by the Open Geospatial Consortium (OGC), and (2) the Semantic Web languages defined by the World Wide Web Consortium (W3C).

2.1.1. Sensor Web Enablement

“The Sensor Web is a special type of Web-centric information infrastructure for collecting, modeling, storing, retrieving, sharing, manipulating, analyzing, and visualizing information about sensors and sensor observations of phenomena [Gross99].” The OGC, an international consortium of industry, academic, and government organizations tasked with developing open geospatial standards, describes the sensor Web as sensor networks and sensor data, accessible though the Web, which are discoverable and accessible through standard protocols and application program interfaces [Botts08]. The Sensor Web has vast significance for applications using sensor technologies to attain actionable situation awareness. Lack of standardization, however, is the primary barrier to realizing a progressive sensor Web.

The Open Geospatial Consortium recently established the Sensor Web Enablement as a suite of specifications related to sensors, sensor data models, and sensor Web services that will enable sensors to be accessible and controllable via the Web [Sheth08]. The core suite of language and service interface specifications (depicted in Figure 2.1) includes: Observations and Measurements (O&M), Sensor Model Language (SensorML), Transducer Model Language (TML), Sensor Observation Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS), and Web Notification Service (WNS). For more information about these languages and service interface specifications, see [Botts08].

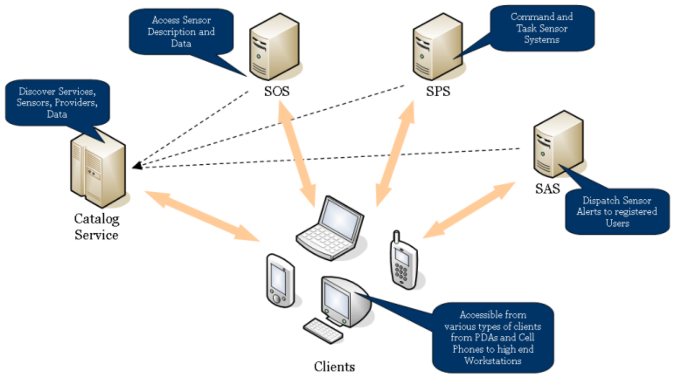

Access Sensor Description and Data
SOS
Command and Task Sensor Systems
SPS
Discover Services, Sensors, Providers, Data
Catalog Service
SAS
Dispatch Sensor Alerts to registered Users
Accessible from various types of clients from PDAs and Cell Phones to high end Workstations
Clients

Figure 2.1. OGC Sensor Web Enablement Services.

2.1.2. SWE Sensor Observation Service

The Sensor Observation Service (SOS) is an OGC-SWE standard that defines a web service interface for providing "access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors [SOS]." SOS groups observations made by related sensor systems into *Observation Offerings*. An *Observation Offering* is a logical collection of sensors and sensor systems that, generally, are located in proximity to one another and sample their environment at shared intervals. *Observation*

Offerings are characterized by the following parameters: sensor, time, sensed phenomena, and location [SOS].

SOS defines four service profiles: core, transactional, enhanced, and entire (which includes all functions from the previous three). For a standards compliant SOS service, only support for the core profile is mandatory, while all other profiles are optional. The core and enhanced profiles provide support for consumers of sensor data. A consumer client of sensor data requires methods for obtaining information about the service itself and requesting observations, sensor descriptions, features, etc. over some spatial and temporal context. This information is useful in applications such as visualization, data fusion, and situation awareness. The transactional profile supports publishers of sensor data. Such publisher clients are responsible for acting as intermediaries between sensor networks generating observations and the SOS service where it inserts sensor descriptions and observations. The core profile includes three operations: *GetCapabilites*, *DescribeSensor*, and *GetObservation*. The *GetCapabilites* function provides a means to request a description of the service. This description includes information such as service identification (service name, keywords, etc.), provider, and most importantly, metadata that allows for the discovery of the capabilities of the service. The capability description includes metadata about all supported functions of the service (including valid values and ranges for query parameters), filtering capabilities (logical operators that may be supplied with query parameters), and a full list of all *Observation Offerings* (including the aforementioned parameters: sensor systems, time, phenomenon, location, etc.) defined within the service. *DescribeSensor* allows the client to request information about a sensor. *DescribeSensor* is parameterized by the ID of the senor and returns a SensorML or TransducerML document describing the sensor and its capabilities. The *GetObservation* function is the heart of the SOS, allowing the client to request observation data generated by a sensor or sensor system contained in a specified *Observation Offering*. *GetObservation* supports a multitude of parameters and filters, which give the client the ability to query over the sensor, time, location, phenomena, features, and measurement values of the observations. The response from *GetObservation* is encoded in O&M. The transactional profile of SOS includes functions that allow a client to insert new sensors and observations, and is composed of two functions: *RegisterSensor* and *InsertObservation*. *RegisterSensor* allows a client to insert a new sensor into an SOS service, including the sensor's capabilities as described in a SensorML or TransducerML document. *InsertObservation* allows a client to insert a new observation into an SOS service. The new observation is provided to the SOS encoded as an O&M document. The enhanced profile provides an assortment of less-frequently needed functions. *GetObservationById* returns an O&M observation based on the ID of the observation. *GetResult* provides a means for a client to obtain sensor data on a frequent basis using less bandwidth, by using a template O&M document from a previous call to *GetObservation*. *GetFeatureOfInterest* returns a description of a feature of interest for which *GetCapabilities* advertised the ID. *GetFeatureOfInterestTime* describes the valid time periods for a feature. *DescribeFeatureType* yields an XML schema for a feature. *DescribeObservationType* returns the XML schema for an observation generated for a type of phenomenon. *DescribeResultModel* yields an XML schema that can further describe the format of results returned by the SOS and referenced in *GetCapabilities*.

2.1.3. Semantic Web

The Semantic Web, as described by the W3C Semantic Web Activity, is an evolving extension of the World Wide Web in which the semantics, or meaning, of information on the Web is formally defined [SWActivity]. Formal definitions are captured in ontologies, making it possible for machines to interpret and relate data content more effectively. The principal technologies of the Semantic Web include the Resource Description Framework (RDF) [RDF] data representation model, and the ontology representation languages RDF Schema (RDF-S) [RDFS] and Web

Ontology Language (OWL) [OWL]. In addition to these representation languages, an RDF query language called SPARQL [SPARQL] is now a W3C recommendation and the common method of querying ontological data. Many rule languages and rule engines are now capable of reasoning with Semantic Web data, including SWRL (Semantic Web Rule Language), RIF (Rule Interchange Format), and the general-purpose rule engine for the Jena Semantic Web Framework [Jena].

An ontology is a formal model that defines concepts and their relations in a standard language, commonly described as a "specification of a conceptualization [Gruber93]." In practice, the Semantic Web defines several ontology languages, RDF, RDF-S, and OWL. The Resource Description Format (RDF) is a graph-based language that allows data within a domain to be linked through named relationships. An RDF graph is encoded as a set of subject-predicate-object triples which resemble the subject, verb, and object of a sentence. The subject and object are nodes in the graph and the predicate is a directional named link between the subject and object.

> "*This simple triple structure turns out to be a natural way to describe a large majority of the data processed by machines. The subjects, verbs and objects are each identified by a Universal Resource Identifier (URI)—an address just like that used for Web pages. Thus, anyone can define a new concept, or a new verb, by defining a URI for it on the Web* [Shadbolt08]."

RDF-S, or RDF Schema, adds the ability to define hierarchies of concepts to RDF. The Web Ontology Language (OWL) is built on top of RDF and adds a logical formalism to the language. OWL is based on a tractable subset of First Order Logic called Description Logic. The logical formalism provided by OWL, in combination with rule engines, is what allows inference over semantically annotated sensor observations.

2.2. Semantic Sensor Observation Service

What are the possible benefits of integrating the Sensor Web with the Semantic Web? Much can be said in answer to this question, including a more expressive graph-based representation that models relationships as first class objects, the use of Uniform Resource Identifier's that allows all concepts to be independently accessible throughout the Web, and a triple-pattern encoding scheme that provides for simplified integration of heterogeneous datasets [Sheth08][Thirunarayan09b]. While these are all important elements of the Semantic Web, in this section we will focus on the need for inference on sensor data enabled by semantic modeling and what advantages this provides to standard SOS.

Reasoning is a useful tool for providing meaning to sensor data and presenting insight into an observed environment. The quantified nature of sensor data, however, is not well suited for logical inference. In order to reason over sensor observations the data must first be annotated with meaningful concepts that can be manipulated with an inference engine. These concepts are defined in an ontology that provides the logical framework for further inference. In the Semantic Web, the Web Ontology Language (OWL) fulfills this role of a meta-language for ontology development.

This collection of annotations and inferences within an ontology make up a knowledge base. The knowledge in this knowledge base can be accessed through a standard SOS request, making the sensor data useful for a wide range of applications that lack the facility to handle raw sensor data but are able to deal with high-level knowledge. On the other hand, supposing an application does have the capability to handle raw sensor data, the lack of a service providing a shared semantics of sensor observations, obligates the client to independently translate the raw sensor data into useful high-level knowledge. This approach may lead to interpretations of data that are exclusive to a single client application and incompatible with applications that may otherwise make use of such knowledge. By committing to the interpretation described within an ontology, applications may benefit from a shared semantics of sensor data, thus leading to improved interoperability. This configuration of a Sensor Observation Service that provides access to ontological knowledge of sensor observations is termed Semantic SOS, or SemSOS. Figure 2.2 shows an implemented architecture of SemSOS.

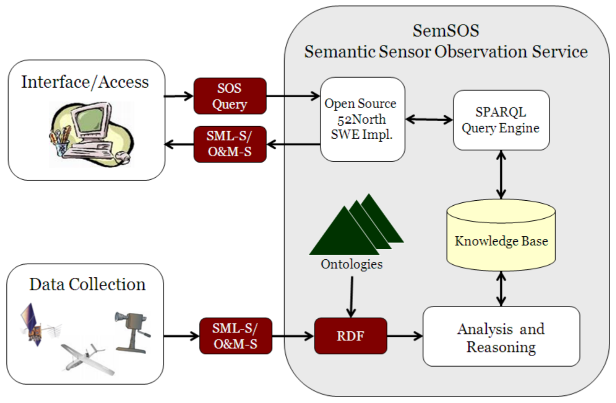


Figure 2.1. High-level View of SemSOS Architecture

2.2.1. Observations and Measurements Ontology

Observations and Measurements (O&M) is an OGC-SWE standard that defines an XML Schema for describing observations and features. Within this standard, an observation (*om:Observation*) is defined as an "act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property," and a feature (*om:Feature*) is defined as an "abstraction of real world phenomenon [O&M]." (Note: *om* is used as a namespace for Observations and Measurements and will be placed, with a colon, before concepts defined in the O&M schema. All defined concepts are italicized). The major properties of an observation include feature of interest (*om:featureOfInterest*), observed property (*om:observedProperty*), sampling time (*om:samplingTime*), result (*om:result*), and procedure (*om:procedure*). Often these properties can be complex entities that may be defined in an external document. For example, *om:FeatureOfInterest* could refer to any real-world entity such as a coverage region, vehicle, or weather-storm, and *om:Procedure* often refers to a sensor or system of sensors defined within a SensorML document. Therefore, these properties are better described as relationships of an observation. In order to encode relationships in XML, the OGC-SWE often make use of XLink, XML Linking Language, a markup language that annotates XML documents by inserting elements to "create and describe links between resources [XLink]."

While XLink allows XML documents to break free of the standard tree-model and define relationships between entities, the triple-pattern approach of RDF provides a far more natural and useful approach to encoding relationships. In RDF and OWL, relationships are considered first-class objects that have many benefits over XLink, such as the ability to assign a URI to a relationship, to classify relationships into hierarchies (RDF-S and OWL), and place constraints on relationships (OWL). For these reasons, we have developed an encoding of the Observations and Measurements language in OWL. In this ontology, we have defined the previous relations, and more, in a form that may be queried and reasoned over effectively in order to derive actionable knowledge of the environment from sensor observations. (Note that the ontology captures a subset of concepts in O&M. A few notable exemptions currently include concepts related to coverage and sampling feature). The translation between O&M in OWL and O&M in XML is straightforward and thus allows SemSOS to remain SOS compliant. (Throughout this chapter, we will refer to O&M in OWL as O&M-OWL and refer to O&M in XML as O&M-XML). Figure 2.3 shows a diagram of the major concepts and relations in O&M-OWL.

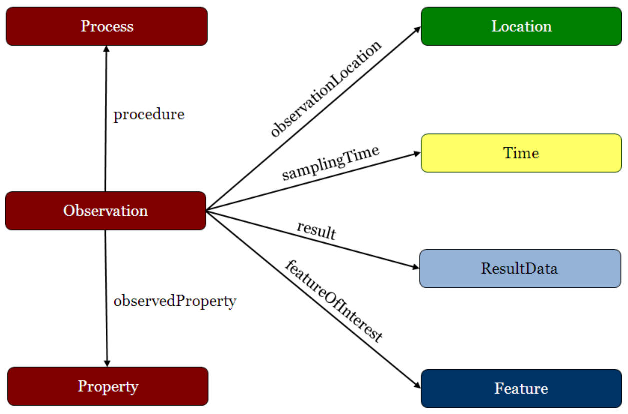


Figure 2.2. Subset of Major Concepts and Relations in O&M-OWL

The following description of relationships in O&M-OWL includes a running example of an observation from the domain of weather (concepts from weather ontology contain namespace "*w*"), encoded as a set of RDF triples. (Each line represents a triple, with the first term representing the subject, the second representing the predicate, the third representing the object, and ending with a period).

```
om:obs_1  rdf:type  om:Observation .
```

om:featureOfInterest is a "representation of the observation target, being the real-world object regarding which the observation is made [O&M]." Example includes a blizzard feature.

```
om:obs_1  om:featureOfInterest  om:blizzard_1 .
om:blizzard_1  rdf:type  w:Blizzard .
w:Blizzard  rdfs:subClassOf  om:Feature .
```

om:observedProperty “identifies or describes the phenomenon for which the observation result provides an estimate of its value. It must be a property associated with the type of the feature of interest [O&M].” Example includes a temperature observed property.

```
om:obs_1  om:observedProperty  w:temperature .
w:temperature  rdf:type  om:Property .
```

om:samplingTime is the “time that the result applies to the feature-of-interest [O&M],” or, in other words, it is the time when the phenomenon was measured in the real-world. Example includes a single instant sampling time at 5:00 am on Jan. 26, 2009.

```
om:obs_1  om:samplingTime  om:time_1 .
om:time_1  rdf:type  owl-time:Instant .
om:time_1  owl-time:date-time  “20090126T05:00:00” .
```

om:observationLocation is the location of an observation event; usually associated with the location of the sensor when an observation occurred (i.e., *om:samplingTime*). Example includes a single point observation location with latitude, longitude, and elevation coordinates.

```
om:obs_1  om:observationLocation  om:location_1 .
om:location_1  rdf:type  gml:Point .
om:location_1  gml:latitude  “41.1915” .
om:location_1  gml:longitude  “-111.8351” .
om:location_1  gml:elevation  “6562.0” .
```

om:result is an "estimate of the value of some property generated by a known procedure [O&M]." Example includes a temperature measurement result of 37 degrees Fahrenheit.

```
om:obs_1  om:result  om:result_1 .
om:result_1  rdf:type  om:ResultData .
om:result_1  om:value  "37" .
om:result_1  om:uom  w:Fahrenheit .
```

om:procedure is a "description of a process used to generate the result. It must be suitable for the observed property [O&M]." Note that in this schema a sensor is defined as a type of process, along with other methods, algorithms, instruments, or systems of these. Example includes a temperature sensor as the procedure.

```
om:obs_1  om:procedure  om:sensor_1 .
om:sensor_1  rdf:type  w:TemperatureSensor .
w:TemperatureSensor  rdfs:subClassOf  om:Sensor .
om:Sensor  rdfs:subClassOf  om:Process .
```

2.2.2. Spatial, Temporal, and Thematic Ontologies

From Figure 2.3, you will notice concepts related to *om:Observation* such as *om:Location*, *om:Time*, and *om:Feature*. While these concepts are defined in O&M-OWL, they are also extended with more expressive descriptions from existing schemas, in the case of *om:Location* and *om:Time*, and from a domain specific ontology, in the case of *om:Feature*. Locations within O&M-OWL are described using concepts from GML, or Geography Markup Language [GML]. In particular, we re-use common concepts such as *gml:Point*, *gml:Polygon*, and *gml:coordinates*.

Time within O&M-OWL is described using concepts from OWL-Time [OWLTime]. OWL-Time, a W3C recommended ontology based on temporal calculus, provides descriptions of temporal concepts such as *owl-time*:*instant* and *owl-time:interval*, which supports defining interval queries such as ‘within’, ‘contains’, and ‘overlaps’. The logical framework provided by OWL-Time for reasoning over time intervals could be very useful when dealing with observations that require complex temporal models. For example, o*m:TimeSeriesObservation* is defined as a *om:CompoundObservation* “whose sampling time is the period encompassing all the member times” such that all “member observations have the same feature of interest, the same observed property, and different sampling times [O&M].” The concept of *om:Feature* within O&M encompasses all real-world entities and thus can be best described through domain-specific thematic ontologies. For example, for use in the domain of weather, *om:Feature* is extended with a weather ontology describing concepts such as *w:SnowStorm*, *w:Blizzard*, and *w:SnowFlurry*.

2.2.3. Semantic Annotation of SWE

While encoding sensor data in OWL is useful for advanced analysis and reasoning, the SOS specification requires observation data to be encoded in XML for several operations. The *InsertObservation* operation takes an O&M-XML document as input and adds the observations to the storage facility. Similarly, the *GetObservation* operation returns an O&M-XML document as response to the query. As previously stated, translating from O&M-XML to O&M-OWL, and vice-versa, is straightforward. However, it is often useful to also embed semantic terminology defined in an ontology model into an XML document. This technique is called semantic annotation and is used for greater semantic interoperability of data encoded in XML, which provides only syntactic interoperability. Ontology terms are embedded in XML documents through model references, or URIs of concepts defined in an ontology. The OGC-SWE standards already provide several mechanisms to reference concepts that are external to the document.

Such concepts are either defined in another XML document and accessed through an XLink element or defined in a registry and accessed through the *swe:definition* attribute. Using either mechanism, we can embed a model reference that will provide more meaningful description and thus enhanced semantic interoperability. Semantically annotated O&M and SML are called O&M-S and SML-S, respectively. This technique is also applied within the *GetCapabilities* operation in order to embed high-level *om:Feature* concepts that may otherwise be unavailable in an SOS *GetCapabilities* response. This is necessary to inform a SemSOS client of the precise description of concepts that may be used to query the knowledgebase.

2.2.4. Rule Based Reasoning

To derive additional knowledge from semantically annotated sensor observations, it may be necessary to define and use rules. To demonstrate rule-based reasoning over sensor observation data, in this section we use the general purpose rule engine from the Jena Semantic Web Framework [Jena]. Such rules deduce new ontological assertions from known instances and class descriptions. This section provides an example of inference through rules in SemSOS. In the weather domain, if a group of sensors provides observations regarding wind speed, visibility, and precipitation, then by using inference rules we can specify existing weather events in the environment, such as a blizzard. The following rule states that if wind speeds are high (*HighWinds*), visibility is low (*LowVisibility*), and it is snowing (*Snowfall*), then there is a blizzard event (*Blizzard*) [NOAA].

```
Blizzard ← HighWinds & LowVisibility & Snowfall
```

Each of these conditions described above is associated with a single time and location, derived from the time and location of the corresponding observations. Subsequently, the *Blizzard* condition is associated with the same time and location as the component weather conditions. The terms *HighWinds* and *LowVisibility* are also derived through rules.

```
HighWinds ← WindSpeed >= 35 MPH
LowVisibility ← Visibility <= ¼ mile
```

Within O&M-OWL, we begin with a quantified observation (*om:Observation*) and data result (*om:ResultData*) and translate this into additional qualified knowledge that can also be used within a reasoning engine. For example, the following set of RDF triples represents data about a wind speed observation.

```
om:windspeed_1  rdf:type  w:WindSpeedObservation .
om:windspeed_1  om:samplingTime  om:time_1 .
om:windspeed_1  om:observationLocation  om:location_1 .
om:windspeed_1  om:result  om:result_1 .
om:result_1  om:value  "37" .
om:result_1  om:uom  w:MPH .
```

From this set of RDF triples we can infer that observation *w:windspeed_1* can also be defined as an instance of class *w:HighWindSpeedObservation*. This new assertion is added to the original set of RDF triples (new triple in bold).

```
om:windspeed_1  rdf:type  w:WindSpeedObservation .
om:windspeed_1  om:samplingTime  om:time_1 .
om:windspeed_1  om:observationLocation  om:location_1 .
om:windspeed_1  om:result  om:result_1 .
```

```
om:result_1  om:value  “37” .
om:result_1  om:uom  w:MPH .
```
`om:windspeed_1  rdf:type  w:HighWindSpeedObservation .`

The rule used to generate this new knowledge, titled *HighWindSpeedObservationRule*, is specified below (in the Jena rule syntax [Jena]).

```
[HighWindSpeedObservationRule:
      (?w_obs  rdf:type  w:WindSpeedObservation)
      (?w_obs  om:samplingTime  ?time)
      (?w_obs  om:observationLocation  ?location)
      (?w_obs  om:result  ?result)
      (?result  om:uom  w:MPH)
      (?result  om:value  ?value)
      greaterThan(?value  35)
→(?w_obs  rdf:type  w:HighWindSpeedObservation)]
```

A low visibility observation (*w:LowVisibilityObservation*) is deduced similarly, and together with a snowfall precipitation observation (*w:SnowfallObservation*) we can infer a blizzard event (*w:Blizzard*) at the same time and location. The rule used to generate this new knowledge is titled *BlizzardObservationRule*.

```
[BlizzardObservationRule:
      (?w_obs  rdf:type  w:HighWindSpeedObservation)
      (?w_obs  om:samplingTime  ?time)
      (?w_obs  om:observationLocation  ?location)
```

```
    (?v_obs  rdf:type  w:LowVisibilityObservation)
    (?v_obs  om:samplingTime  ?time)
    (?v_obs  om:observationLocation  ?location)
    (?p_obs  rdf:type  w:SnowfallObservation)
    (?p_obs  om:samplingTime  ?time)
    (?p_obs  om:observationLocation  ?location)
    makeTemp(?blizzard)
→(?blizzard  rdf:type  w:Blizzard)
    (?blizzard  om:eventTime  ?time)
    (?blizzard  om:eventLocation  ?location)
    (?w_obs  om:featureOfInterest  ?blizzard)
    (?v_obs  om:featureOfInterest  ?blizzard)
    (?p_obs  om:featureOfInterest  ?blizzard)]
```

Note that the *makeTemp(?blizzard)* function in the body of the rule generates a new instance in the knowledge base. Subsequently, we then supply this instance of *om:Blizzard* with relations in the head of the rule. In this example, such relations include *rdf:type*, *om:eventTime*, *om:eventLocation*, and *om:featureOfInterest*. The final set of RDF triples is shown below (ellipses used to truncate set of triples, and new triples in bold).

```
om:windspeed_1  rdf:type  w:WindSpeedObservation .
om:windspeed_1  om:samplingTime  om:time_1 .
om:windspeed_1  om:observationLocation  om:location_1 .
…
om:windspeed_1  rdf:type  w:HighWindSpeedObservation .
om:visibility_1  rdf:type  w:VisibilityObservation .
```

```
…
om:visibility_1  rdf:type  w:LowVisibilityObservation .
om:precipitation_1  rdf:type  w:SnowfallObservation .
…
om:blizzard_1  rdf:type  w:Blizzard .
om:blizzard_1  om:samplingTime  om:time_1 .
om:blizzard_1  om:observationLocation  om:location_1 .
om:windspeed_1  om:featureOfInterest  om:blizzard_1 .
om:visibility_1  om:featureOfInterest  om:blizzard_1 .
om:precipitation_1  om:featureOfInterest  om:blizzard_1.
```

In this manner, we can infer features within the environment, of a particular type, at a specific time and place, and then generate *om:featureOfInterest* relations between the original observations and the new features. These new *om:featureOfInterest* relationships can be used to query for high-level feature concepts in SemSOS.

2.2.5. SemSOS Implementation

In order to validate the framework discussed above, we have constructed a prototype of SemSOS. Our SemSOS extends the open source implementation of SOS from 52North [52North] with an ontological knowledge base in order to provide inference over sensor data and queries of high-level features. For this prototype, the sensor observation data used to populate our ontologies was collected from MesoWest, a repository of weather data at the University of Utah [MesoWest]. MesoWest continually collects data from over 20,000 sensor systems within North America, and stores archives since 2002.

2.3.5.1. 52North SOS

52North's SOS implementation is designed to be highly modular, and adaptable to arbitrary suitable sensor data sources, transport protocols, etc. The larger enclosed box in Figure 2.4 shows the high-level architecture of the 52North SOS.

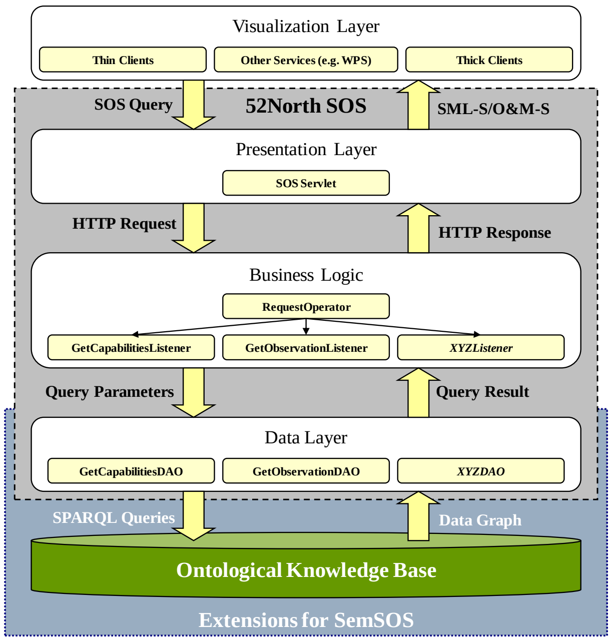


Figure 2.3. 52North SOS Architecture, extended with an ontological knowledge base.

The Visualization Layer shown in Figure 2.4 is not part of the SOS itself, but rather corresponds to external clients that interact with the SOS. These can be either publishers or consumers of sensor data, and may also be other web services. The Presentation Layer of 52North's architecture defines the SOS's interface to the outside world. The default implementation has a Servlet interface that accepts requests and communicates responses via HTTP. If another transport mechanism or protocol is required, this level can be replaced without affecting the other layers of the SOS. The next level is the Business Layer, which receives requests from the Presentation Layer, handles them as appropriate, and returns a response. The Business Layer contains the logic for decoding requests and encoding responses. The main entry-point from the Presentation Layer is the *RequestOperator* object, which validates incoming requests, determines the type of request, and dispatches accordingly. Each operation supported by the SOS (*GetCapabilities*, *GetObservation*, etc.) is embodied by a Listener object which handles the corresponding incoming request (resp. *GetCapabilitiesListener*, *GetObservationListener*, etc.). The Listener objects may be configured externally during deployment of the service. The individual Listeners handle high-level translation of the request into an internal format which is then used to query the respective object in the Data Layer and compose the response. The final layer of the 52North architecture is the Data Layer. The Data Layer is an abstraction of a sensor data source through Data Access Objects (DAO). Each DAO represents a particular interface to the sensor data from the point of view of one of the SOS's operations. For each Listener object in the Business Logic Layer, there is a corresponding DAO object in the Data Layer. The DAO objects are used by their respective Listener objects to obtain the data pertaining to a query. The abstraction provided by the DAOs and the Data Layer is what allows the 52North's SOS implementation to be so easily adapted to new sources of sensor data. For each operation that must be supported, all that is required is a new DAO that works with the data source. The default implementation shipped with 52North uses a PostGIS database with a custom database schema to store observation data, while sensor descriptions are stored on the file system in XML files (using SensorML or TransducerML).

2.2.5.2. SemSOS Extensions to 52North

The box surrounding the bottom third of Figure 2.4 denotes the extensions made to 52North's SOS in order to implement SemSOS. The modular nature of the 52North implementation allowed us to leave the request routing, encoding/decoding, and similar details in place, while replacing the data access implementation with our own. The DAOs for all three operations specified in the

SOS core profile (*GetCapabilities*, *GetObservation*, and *DescribeSensor*) were replaced with implementations that support data access to an O&M-OWL knowledge base. Specifically, SemSOS uses the Jena Semantic Web Framework [Jena] to store and access the O&M-OWL ontology. The stored ontology is then accessed via SPARQL queries that are generated from the incoming SOS query parameters [SPARQL]. In producing the SPARQL queries, the syntactic form of the SOS query parameters (such as date, time, magnitude, etc.) are transformed into appropriate formats for semantic querying over O&M-OWL. Likewise, query filters (such as location, comparison operators, etc.) must be transformed into SPARQL-style filters and relational operations. Evaluating a SPARQL query results in a set of triples representing an RDF graph, with data annotated in O&M-OWL. This graph is then transformed into the internal 52North result structure and returned to the Business Logic Layer. Here, the previous translation to convert SOS queries into SPARQL must be performed in reverse. O&M-OWL concepts instantiated within a set of RDF triples are translated into O&M-XML. The results of SemSOS client queries are thus valid SOS results. SemSOS also provides richer semantic interoperability for clients that are semantically-aware through semantic annotation of the O&M-XML result document. This is achieved by using model references, or URIs of concepts defined in an ontology, as identifiers within O&M-XML.

2.2.5.3. Example SemSOS Query Processing

The first step the SemSOS DAOs must take in serving an SOS request is to translate the incoming SOS query into a SPARQL query which may be run against the knowledge base. Figure 2.5 shows an example SOS query asking for all observations that (1) are generated by procedures (sensors) that are part of offering (sensor constellation) ‘BRAU1’, (2) fall within the time span of 2003-04-03T20:00:00-05 to 2003-04-04T02:00:00-05 (a six-hour interval), and (3) correspond to one of four specific observed properties: air temperature, precipitation, wind speed, wind gust.

```
<GetObservation xmlns=... service="SOS" version="1.0.0"
                          srsName="urn:ogc:def:crs:EPSG:4326">
<offering>BRAU1</offering>
<eventTime>
    <ogc:TM_During>
        <ogc:PropertyName>urn:ogc:data:time:iso8601</ogc:PropertyName>
        <gml:TimePeriod>
            <gml:beginPosition>2003-04-03T20:00:00-05</gml:beginPosition>
            <gml:endPosition>2003-04-04T02:00:00-05</gml:endPosition>
        </gml:TimePeriod>
    </ogc:TM_During>
</eventTime>
<observedProperty>http://.../weather.owl#_AirTemperature</observedProperty>
<observedProperty>http://.../weather.owl#_Precipitation</observedProperty>
<observedProperty>http://.../weather.owl#_WindSpeed</observedProperty>
<observedProperty>http://.../weather.owl#_WindGust</observedProperty>
<responseFormat>text/xml;subtype=&quot;om/1.0.0&quot;</responseFormat>
</GetObservation>
```

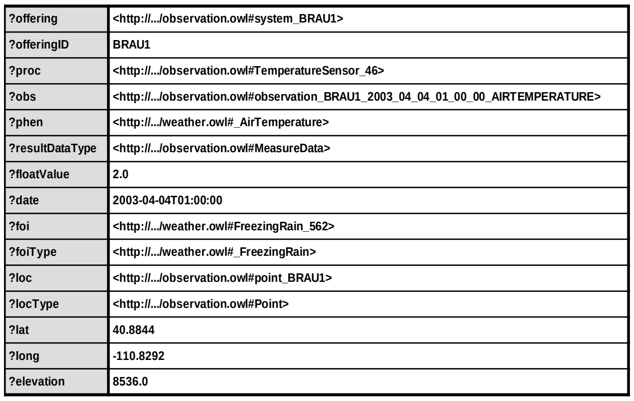
Figure 2.4. Example SOS query.

The SOS query is then transformed into the SPARQL query depicted in Figure 2.6, which expresses the same constraints as the original, but in the language of O&M-OWL. Note that the event time specification in the SOS query becomes a SPARQL filter, as do the observed property specifications. Other SOS query relational operations and filters, such as location or feature of interest, are handled similarly.

```
SELECT DISTINCT  ?offering ?offeringID ?proc ?obs ?phen ?resultDataType ?floatValue
    ?intValue ?booleanValue ?date ?foi ?foiType ?loc ?locType ?lat ?long ?elevation

WHERE {
   ?offering rdf:type observation:System .
   ?offering observation:ID "BRAU1" .
   ?offering observation:ID ?offeringID .
   ?offering observation:systemComponentProcess ?proc .
   ?proc observation:generatedObservation ?obs .
   ?obs observation:instFeatureOfInterest ?foi .
   ?obs observation:featureOfInterest ?foiType .
   ?obs observation:samplingTime ?inst .
   ?inst xsd:datetime ?date .
   ?obs observation:observedProperty ?phen .
   ?obs observation:result ?result .
   ?result rdf:type ?resultDataType .
   {
      {?result observation:floatValue ?floatValue . }
      UNION  {?result observation:intValue ?intValue . }
      UNION  {?result observation:booleanValue ?booleanValue . }
   }
   ?obs observation:observationLocation ?loc .
   ?loc rdf:type ?locType .
   ?loc observation:latitude ?lat .
   ?loc observation:longitude ?long .
   ?loc observation:elevation ?elevation .
   FILTER(?phen=<http://.../weather.owl#_AirTemperature>
      || ?phen=<http://.../weather.owl#_Precipitation>
      ||  ?phen =<http://.../weather.owl#_WindSpeed>
      ||  ?phen=<http://.../weather.owl#_WindGust>  ) .
   FILTER ( ?date > "2003-04-03T20:00:00"^^xsd:dateTime
      && ?date < "2003-04-04T02:00:00"^^xsd:dateTime ) .
}
```

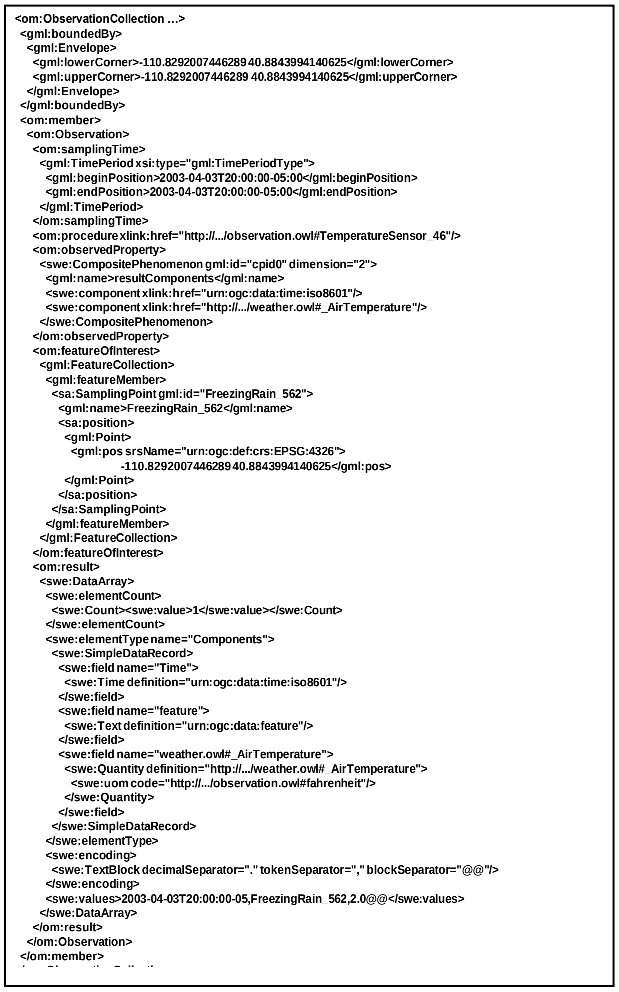
Figure 2.5. Example SPARQL query.



?offering	<http://.../observation.owl#system_BRAU1>
?offeringID	BRAU1
?proc	<http://.../observation.owl#TemperatureSensor_46>
?obs	<http://.../observation.owl#observation_BRAU1_2003_04_04_01_00_00_AIRTEMPERATURE>
?phen	<http://.../weather.owl#_AirTemperature>
?resultDataType	<http://.../observation.owl#MeasureData>
?floatValue	2.0
?date	2003-04-04T01:00:00
?foi	<http://.../weather.owl#FreezingRain_562>
?foiType	<http://.../weather.owl#_FreezingRain>
?loc	<http://.../observation.owl#point_BRAU1>
?locType	<http://.../observation.owl#Point>
?lat	40.8844
?long	-110.8292
?elevation	8536.0

Figure 2.6. Example SPARQL query results.

The table in Figure 2.7 displays one row of the result from the SPARQL query in Figure 2.6. The row contains information pertaining to a single air temperature reading generated by a sensor that is a member of the offering specified in the original SOS query. The result value of the reading is present *(?floatValue*), along with the location (*?loc*, *?locType*, *?lat*, *?long*, *?elevation*) and a related feature of interest (*?foi*, *?foiType*), in this case an instance of freezing rain. The full result of the SPARQL query contains many more rows including observations from the same sensor at different times, and observations from other sensors contained in the same offering, which may have observed different phenomena and relate to different features. The result of the SPARQL query is then used to construct an SOS response document, as shown in Figure 2.8.



```xml
<om:ObservationCollection …>
 <gml:boundedBy>
  <gml:Envelope>
   <gml:lowerCorner>-110.8292007446289 40.8843994140625</gml:lowerCorner>
   <gml:upperCorner>-110.8292007446289 40.8843994140625</gml:upperCorner>
  </gml:Envelope>
 </gml:boundedBy>
 <om:member>
  <om:Observation>
   <om:samplingTime>
    <gml:TimePeriod xsi:type="gml:TimePeriodType">
     <gml:beginPosition>2003-04-03T20:00:00-05:00</gml:beginPosition>
     <gml:endPosition>2003-04-03T20:00:00-05:00</gml:endPosition>
    </gml:TimePeriod>
   </om:samplingTime>
   <om:procedure xlink:href="http://.../observation.owl#TemperatureSensor_46"/>
   <om:observedProperty>
    <swe:CompositePhenomenon gml:id="cpid0" dimension="2">
     <gml:name>resultComponents</gml:name>
     <swe:component xlink:href="urn:ogc:data:time:iso8601"/>
     <swe:component xlink:href="http://.../weather.owl#_AirTemperature"/>
    </swe:CompositePhenomenon>
   </om:observedProperty>
   <om:featureOfInterest>
    <gml:FeatureCollection>
     <gml:featureMember>
      <sa:SamplingPoint gml:id="FreezingRain_562">
       <gml:name>FreezingRain_562</gml:name>
       <sa:position>
        <gml:Point>
         <gml:pos srsName="urn:ogc:def:crs:EPSG:4326">
                 -110.8292007446289 40.8843994140625</gml:pos>
        </gml:Point>
       </sa:position>
      </sa:SamplingPoint>
     </gml:featureMember>
    </gml:FeatureCollection>
   </om:featureOfInterest>
   <om:result>
    <swe:DataArray>
     <swe:elementCount>
      <swe:Count><swe:value>1</swe:value></swe:Count>
     </swe:elementCount>
     <swe:elementType name="Components">
      <swe:SimpleDataRecord>
       <swe:field name="Time">
        <swe:Time definition="urn:ogc:data:time:iso8601"/>
       </swe:field>
       <swe:field name="feature">
        <swe:Text definition="urn:ogc:data:feature"/>
       </swe:field>
       <swe:field name="weather.owl#_AirTemperature">
        <swe:Quantity definition="http://.../weather.owl#_AirTemperature">
         <swe:uom code="http://.../observation.owl#fahrenheit"/>
        </swe:Quantity>
       </swe:field>
      </swe:SimpleDataRecord>
     </swe:elementType>
     <swe:encoding>
      <swe:TextBlock decimalSeparator="." tokenSeparator="," blockSeparator="@@"/>
     </swe:encoding>
     <swe:values>2003-04-03T20:00:00-05,FreezingRain_562,2.0@@</swe:values>
    </swe:DataArray>
   </om:result>
  </om:Observation>
 </om:member>
```

Figure 2.7. Example SOS response.

2.3. Linked Sensor Data

Beyond the Semantic Web languages and technologies discussed above, significant recent progress in the realization of the vision of Semantic Web is the emergence of Linked Data [Bizer09]. Linked Data is a large and growing collection of interlinked public datasets, encoded in RDF, and spanning many diverse domains such as life sciences, nature, science, geography, and entertainment. In the sensors domain, sources of geospatial information such as GeoNames (http://www.geonames.org/) and LinkedGeoData (http://linkedgeodata.org/) are of particular importance. The GeoNames geographical dataset contains over eight million geographical names and consists of 7 million unique features including 2.6 million populated places and 2.8 million alternate names.

Using the sensor model outlined above (Section 2.2.1), we have generated several sensor datasets and made them available as Linked Data [Patni10][Pschorr10]. The datasets contain sensor descriptions and observations collected from weather stations within the United States. These datasets provide links to GeoNames in order to support location-based sensor discovery.

Linked Data as a Sensor Registry – An ideal mechanism for sensor discovery on the Sensor Web should include facilities for expressive query against semantically meaningful user criteria, simple procedures for the inclusion of new sensors and observations, and the ability to extend and build upon existing data. These requirements are all fulfilled by Linked Data, while they highlight weaknesses of traditional service registries. As such, we position Linked Data as an alternative to more conventional registry approaches.

A registry for sensors can expect to have new sensors added occasionally, but must assume additional observation data will be added on a continuous basis. A traditional centralized registry system does not scale to the amount of sensor and observational data that we can expect sensor systems to generate. Linked Data, however, presents a decentralized approach to publishing sensor data by creating relations to existing data and providing dereferenceable URIs.

Extending existing data sets with new relationships is great advantage of using Linked Data as a registry for sensor information. Sensor datasets can make use of temporal, spatial, and thematic concepts published elsewhere in Linked Data. Just as important, however, sensors and observations created by one publisher may be extended by another simply by the generation of new relationships referencing the existing facts. The open and decentralized nature of Linked Data allows rich interaction between sensor and thematic data that is often absent or prohibitively complex given conventional, insular registries.

Sensor Descriptions on Linked Data – Using the model presented in Section 2.2.1, we have generated a dataset of sensor descriptions called *LinkedSensorData*. This dataset is derived from data collected by MesoWest, a project within the Department of Meteorology at the University of Utah [MesoWest]. MesoWest continually collects data from over 20,000 weather stations phenomena within North America. On average, there are about five sensors per weather station measuring phenomena such as temperature, visibility, precipitation, pressure, wind speed, humidity, etc. In addition to location attributes such as latitude, longitude, and elevation, LinkedSensorData also contains links to locations in GeoNames. This dataset is now published as Linked Data.

Sensor Observations on Linked Data – Another dataset, called *LinkedObservationData*, has been generated that contains expressive descriptions of sensor observation data. This dataset is also based on data collected by MesoWest. The observations include measurements of phenomena such as temperature, visibility, precipitation, pressure, wind speed, humidity, etc. The dataset consists of observations made within the United States during the time periods in which several major storms were active (e.g. Hurricane Katrina). These observations were generated by the weather stations described in our sensor descriptions dataset, which they reference. Table 2.1 describes the storms, date ranges, and size of the LinkedObservationData dataset that currently contains over one billion RDF triples and is now published as Linked Data.

Table 2.1. LinkedObservationData statistics

Name	Storm Type	Date	Number of Triples	Number of Observations
Bill	Hurricane	Aug. 17-22, 2009	231,021,108	21,272,790
Gustav	Hurricane	Aug. 25-32, 2008	258,378,511	23,792,818
Bertha	Hurricane	July 6-17, 2008	278,235,734	25,762,568
Wilma	Hurricane	Oct. 17-23, 2005	171,854,686	15,797,852
Katrina	Hurricane	Aug. 23-30, 2005	203,386,049	18,832,041
Charley	Hurricane	Aug. 9-15, 2004	101,956,760	9,333,676
	Blizzard	April 1-6, 2003	111,357,227	10,237,791

Sensor Locations on Linked Data – Once sensor data is encoded in RDF and published as Linked Data, the next step is to leverage the vast spatial information already present on Linked Data. GeoNames provides the type of spatial data necessary not only to relate user-friendly location names to coordinate information, but also to associate contextual information such as region containment and distance from location. Fig. 2.9 shows the overall structure of the datasets and the relationships between them, including links to GeoNames.

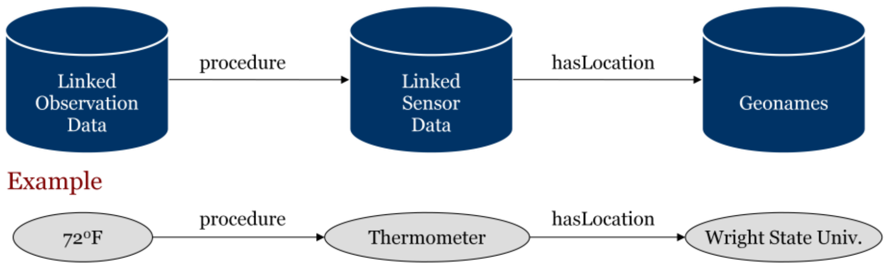


Figure 2.9. Relationships between sensor datasets on Linked Data

For each sensor in our knowledge base, we use the *findNearby*[1] service provided by GeoNames to determine the geographically closest named location, or feature, within the GeoNames dataset. This location is then linked with a sensor through the ‘near’ relationship. This relationship

[1] http://www.geonames.org/export/web-services.html#findNearby

describes not only the location of the sensor, but also contextual information regarding the sensor's distance from the location.

GeoNames classifies locations according to containment (e.g. Wright State University is within the city of Dayton) as well as feature classes and codes (e.g. the feature class of Wright State University is a *spot, building or farm* and its feature code is *school*). This provides an extensive source of semantic spatial information that allows us to construct an intuitive mechanism for finding sensor data by region. In addition to feature hierarchy, each GeoNames location provides a *nearbyFeature* relationship that links to a set of locations that are near the original location. The *nearbyFeature* relationship provides another way to find locations near a sensor.

In order to encode these relations between a sensor and the nearest GeoNames location, sensors are annotated with a link to *LocatedNearRel*. The *LocatedNearRel* concept encodes information about the 'near' relationship that holds between a sensor and a named location. More specifically, it contains the closest GeoNames location and its distance from the sensor. The structure is illustrated in Figure 2.10.

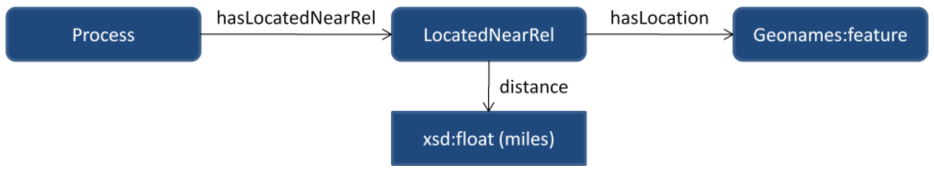


Figure. 2.10. Concepts and relations linking sensors (or processes) described in LinkedSensorData to features described in GeoNames

Sensor Discovery Query over Linked Data – With sensor and observation data published with relationships to spatial datasets on Linked Data, discovery simply becomes a matter of querying RDF data. In our implementation, we perform SPARQL queries over a cached version of the relevant portions of Linked Data, particularly named locations in GeoNames and sensor descriptions in LinkedSensorData described above. Currently, we support discovery of sensors based on GeoNames locations through two basic operations:

- Find the named location closest to a given sensor
- Find all sensors near a given named location

Figure 2.11 shows an example query asking the following question: Find sensors near Wright State University that can tell me about temperature and precipitation. The results from this query will include sensors near the specified location and the associated distance between the sensor and location.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX geonames:<http://www.geonames.org/ontology#>
PREFIX om-owl:<http://knoesis.wright.edu/ssw/sensor-observations.owl#>
PREFIX weather: <http://knoesis.wright.edu/ssw/weather.owl#>

SELECT DISTINCT ?sensor ?dist
WHERE {
        ?sensor rdf:type om-owl:System .
        ?sensor om-owl:hasLocatedNearRel ?near .
        ?sensor om-owl:parameter weather:AirTemperature .
        ?sensor om-owl:parameter weather:Precipitation .
        ?near om-owl:distance ?dist .
        ?near om-owl:hasLocation ?location .
        ?location geonames:name "Wright State University" .
};
```

Figure 2.11. Example discovery query of LinkedSensorData

2.4. Semantic Sensor Network Ontology

The O&M-OWL ontology described above served as inspiration for the Semantic Sensor Network (SSN) ontology [Lefort11][Compton12] developed by the W3C Semantic Sensor Network Incubator Group [SSN-XG]. This group included over 40 researchers from 16 organizations. My role in this group consisted of leading the development of a semantic annotation framework (for annotating SWE documents), contributing to the design and development of the ontology, and acting as editor of the final report. The SSN ontology is currently being used for improved management of sensor data on the Web, involving annotation, integration, publishing, and search [Gray11][Calbimonte11][Pfisterer11]. The ontology defines concepts for representing sensors, sensor observations, and knowledge of the environment. Figure 2.9 provides an overview of the primary classes and properties of the SSN ontology.

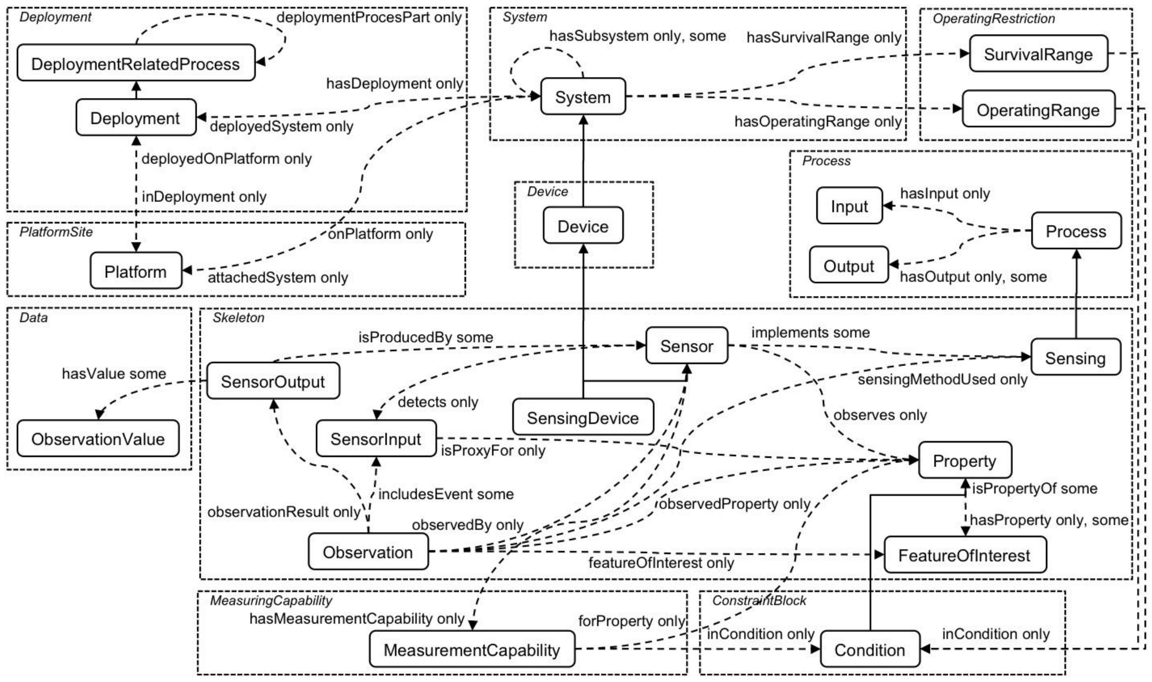


Figure 2.12. Overview of the Semantic Sensor Network ontology classes and properties.

The SSN ontology serves as a foundation to formalize the semantics of perception. In particular, the representation of observations and environmental knowledge are employed. An *observation* (ssn:Observation) is defined as a situation that describes an observed feature, an observed property, the sensor used, and a value resulting from the observation (note: prefix *ssn* is used to denote concepts from the SSN ontology). A *feature* (ssn:FeatureOfInterest; for conciseness, ssn:Feature will be used throughout the paper) is an object or event in an environment, and a *property* (ssn:Property) is an observable attribute of a feature. For example, in cardiology, elevated blood pressure is a property of the feature Hyperthyroidism. To determine that blood pressure is *elevated* requires some pre-processing; however, this is outside the scope of this work. An observation is related to its observed property through the ssn:observedProperty relation.

Knowledge of the environment plays a key role in perception [Neisser76][Gregory97]. Therefore, the ability to leverage shared knowledge is a key enabler of semantics-based machine perception. In SSN, knowledge of the environment is represented as a relation (ssn:isPropertyOf) between a property and a feature. To enable integration with other ontological knowledge on the Web, this environmental knowledge design pattern is aligned with concepts in the DOLCE Ultra Lite ontology[2]. Figure 2.10(a) provides a graphical representation of environmental knowledge in SSN, with mappings to DOLCE. An environmental knowledgebase, storing facts about many features and their observable properties, takes the shape of a bipartite graph. Figure 2.10(b) shows an example knowledge base with concepts from cardiology.

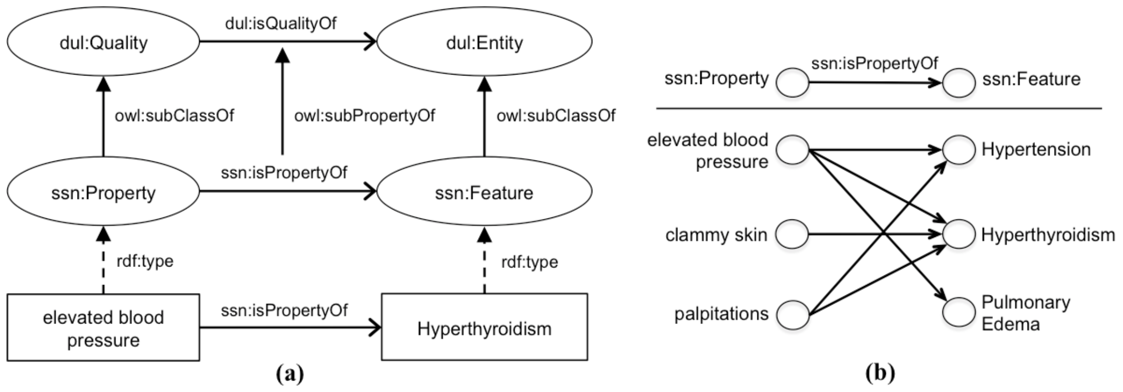


Figure 2.13. (a) Graphical representation of environmental knowledge in the SSN ontology, with mappings to DOLCE Ultra Lite (prefix *dul*). (b) Graphical representation of an example

[2] http://www.loa-cnr.it/ontologies/DUL.owl

environmental knowledgebase in cardiology, taking the shape of a bipartite graph. This knowledgebase is derived from collaboration with cardiologists at ezDI (http://www.ezdi.us/).

2.5. Concluding Remarks

A synthesis of the Sensor Web Enablement standards defined by the OGC and the Semantic Web languages defined by the W3C provides a platform for integration and reasoning over sensor observations in order to attain shared knowledge of an environment. This platform is broadly termed the Semantic Sensor Web [Sheth08], of which SemSOS is a principal component. In the preceding chapter we have described how this is accomplished by modeling the domain of sensors and sensor observations in a suite of ontologies, adding semantic annotations to the sensor data, using the ontology models to reason over sensor observations, and extending an open source SOS implementation with our semantic knowledge base.

In 1999, Neil Gross expressed a vision of the future in which sensors were ubiquitous and engrained in the fabric of our environment:

> "*In the next century, planet earth will don an electronic skin. It will use the Internet as a scaffold to support and transmit its sensations. This skin is already being stitched together. It consists of millions of embedded electronic measuring devices: thermostats, pressure gauges, pollution detectors, cameras, microphones, glucose sensors, EKGs, electroencephalographs. These will probe and monitor cities and endangered species, the atmosphere, our ships, highways and fleets of trucks, our conversations, our bodies--even our dreams* [Gross99]."

I share this vision and wish to provide meaning to this new world, which is the subject of the next chapter.

3. Semantic Perception

Look at the image in Figure 3.1. How quickly were you able to realize the identity of the depicted object? The activity you just engaged in is called perception; and while people are able to perceive their environment almost instantaneously, and seemingly without effort, machines continue to struggle with the task. In the present chapter, we investigate how people are able to perceive the world so effectively and show how an approximation of this process can be formalized to better enable machines to perceive.

Figure 3.1. A red apple.

The study of perception likely began in ancient Greece when Plato first pondered the meaning of shadows dancing on a wall.[1] The Greek word for perception, *intellego*, stems from the Latin *intellegere*: *inter* ("between") and *lego* ("to choose or gather") [Norwich91]. To the ancient Greeks, to perceive was to choose from among alternative explanations which account for our observations. Thus, to perceive an apple is to choose from among a range of possibilities, including an apple, orange, or ball. These acts of observation and perception provide the building

[1] Plato's Cave, http://en.wikipedia.org/wiki/Allegory_of_the_Cave (accessed May, 2011)

blocks for all human knowledge [Locke1960]; they are the processes from which all ideas are born; and the sole bond connecting ourselves to the world around us. Now, with the advent of sensor networks[2] capable of observation, this world may be directly accessible to machines. Missing from this vision, however, is the ability of machines to glean semantics from observation; to apprehend entities from detected qualities; ***to perceive***. The systematic automation of this ability is the focus of machine perception – the ability of computing machines to sense and interpret the contents of their environment [Nevatia82]. Despite early successes within narrow domains (e.g., facial recognition [Zhao03]), however, a general solution remains elusive. *This state of affairs is the result of difficult research challenges, such as the ability to effectively model the process of perception, to provide an appropriate interpretation of observational data with incomplete information, and to efficiently interpret the growing stream of observational data.* The issue of effectively modeling the process of perception is often investigated within specific application areas, such as machine vision [Aloimonos88][Diamant07]. While much progress has been achieved, this approach also results in a fractured assortment of models and algorithms that are effective for narrowly defined problems, such as interpreting sensor data of a single modality. Integrating and interpreting sensor data of multiple modalities for a wide range of applications, however, requires a more encompassing approach. In an attempt to deal with the latter issue, of efficiently interpreting the growing stream of observational data, there is much research within the sensors community to mitigate the effects of observational data overload. Many such efforts concentrate on developing effective schedules and sampling rates for sensor observations [Gürgen06]. Interval-based sampling often generates data (through *brute-force* collection) that is unnecessary for understanding the environment. Another technique for collecting sensor data, called event-based sampling [Pawlowski08][Pawlowski09], generates observation records only after a particular event is detected (e.g., temperature drops below a certain threshold).

[2] A sensor network is a group of specialized measuring devices (i.e., sensors) with a communications infrastructure intended to monitor and record conditions within an environment.

Unfortunately, this provides limited additional benefits towards effective analysis of (often incomplete) data. The dichotomic need for reduced information overload and complete information for effective analysis can be addressed through an understanding, and modeling, of the techniques employed by human perception. *More specifically, the techniques employed by human perception, such as the ability to focus attention and seek out additional information from our environment, holds the key to simultaneously minimizing the amount of information needed for perception and enabling graceful degradation of perception with incomplete information.*

Such challenges are addressed through the development of an ontology of perception, *IntellegO*. This ontology is derived from well-established cognitive theories of perception and establishes a formal semantics for machine perception.

3.1. Cognitive Models of Perception

What are perceptions and how are they formed? What can be perceived? How do perceptions relate to reality? In order to bestow onto a machine the ability to glean semantics from observation, such questions require explicit, implementable, answers. As perceptual beings, people are constantly inundated with sensory input; yet we are able to make sense out of our environment with relative ease. We have a remarkable aptitude for comprehending the world around us; for subconsciously analyzing sensory input and apprehending mental conceptions with efficiency and precision. For centuries, thinkers have endeavored to understand the mechanisms underlying this phenomenon, and through such investigation have advanced complex theories of perception within the fields of philosophy, psychology, physiology, cognitive science, and machine vision. One idea that has emerged from such investigations includes the ability to utilize background knowledge to determine what aspects of the environment to focus our attention

[Bajcsy88][Gregory68][Gregory97][Neisser76], which enables a perceiver to efficiently make sense of the environment.

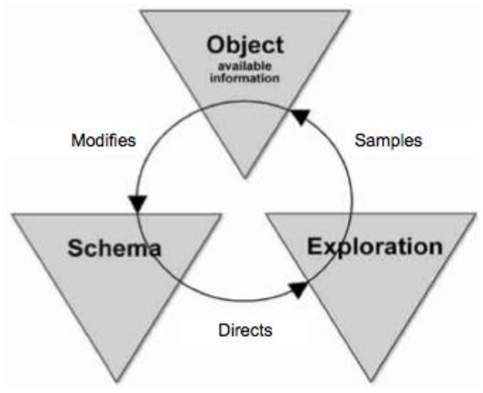


Figure 3.2. Neisser's Perception Cycle

In the late 1970’s, the idea of perception as a cyclical process began to take form. The most famous expression of this idea was proposed by Ulric Neisser in 1976 [Neisser76]. Neisser's Perception Cycle is divided into three stages: (1) sampling (or observing) the environment, (2) modifying our knowledge (schema) of the environment, based on our newly acquired observations, and (3) directing our attention for further exploration. Figure 3.2 shows a graphical representation of Neisser’s Perception Cycle. In contemporary research, this general model is called Active Perception [Bajcsy88]. The idea of perception as an active, cyclical process laid the ground work for our current understanding of human perception. Shortly after, from the early 1980’s and into the late 1990’s, Richard Gregory’s theories of perception began to take shape and were continually refined through investigations into the nature of illusions [Gregory97]. From this work, he showed that the perception cycle iteratively generates and tests hypotheses that explain our observations. As we progress through these tests, by focused attention on our environment, the number of distinct explanatory hypotheses invariably diminishes, leading to more precise, unambiguous hypotheses. In addition, Gregory was able to demonstrate that the hypothesize and test cycle is driven by a-priori background knowledge. In other words, our perception of the world is highly dependent on our knowledge of the world. More recently, the term *top-down processing* has been used to describe the ability to map observations onto background knowledge in order to *fill in the gaps* of this knowledge. This phenomenon has been widely studied within the fields of biological/human vision [Cavanagh99][Gregory97] and machine vision [Aloimonos88][Diamant07]. Meanwhile, in the early 1990's, another unusual model of perception emerged from the field of mathematics. At this time, James Gibson's ideas on perception gained influence; in particular, the idea that physical stimuli carry *information* about the world, and our sensory organs have evolved to intercept, extract, and decode this information [Gibson66]. From this general idea, Peter Norwich was able to extrapolate the conjecture that the stimulus information intercepted by our senses could be measured and understood using Claude Shannon's Information Theory [Shannon48]. This is the basis for the Entropy Theory of Perception which uses mathematical models of entropy and information-gain to determine the informational value of different observations [Norwich91].

Inspired by these theories, this work provides an explicit formalization of perception that may be understood and systematically executed by machines. The formalization is based on three core ideas:

1. Perception is an active, cyclical process of exploration and interpretation (Nessier).
2. The perception cycle is driven by background knowledge in order to generate and test hypotheses (Gregory).

3. In order to effectively test hypotheses, some observations are more informative than others (Norwich).[3]

Below, these ideas are synthesized and perception is viewed as an efficient, cyclical process of actively seeking and detecting those qualities that carry information most useful for testing and evaluating hypothesis.

3.2. Ontology of Perception – Set Theory

Over the years, cognitive theories of perception have been proposed, evaluated, revised, and evolved within an impressive body of research. This research presents a valuable stepping-stone towards the goal of machine perception, to embody this unique human ability within a computational system. In this section, the aim is to explicitly define the information processes involved in perception that will serve as an ontological account of knowledge production. The ontology of perception, or *IntellegO*, attempts to formally model perception in a way that is independent of any particular implementation technology and of suitable generality to encompass both machine perception and human perception.[4] A formal semantics[5] of perception can be defined by providing high-level interpretation of low-level *observational* data, which may be derived through computational means. This ontology is a novel research contribution towards the goal of machine perception.

[3] To our knowledge, Norwich never actually made this connection between his ideas of measuring the informational value of observations and Gregory's ideas of testing hypotheses. This connection stems from our own imagination, and plays a critical role within the ontology of perception.

[4] We are NOT claiming to represent the full spectrum of human perception, but have tried to include ideas from cognitive theory of perception that may be useful for machine perception.

[5] Formal semantics is a rigorous, systematic, and unambiguous description of the meaning of some conceptualization, described in purely symbolic terms; and often embodied within a set of logical axioms and entailments rules.

To communicate the semantics of perception, we must define an appropriate terminology. In the text below, we will describe several concepts, relations, and processes. The concepts and relations include *entity*, *quality*, *quality-type*, *percept*, *observer*, *perceiver*, *focus*, *perceptual-theory*, *inheres-in*, and *has-type*. These concepts and relations are described in Section 3.2.1. The processes include *observation-process*, *perception-process*, and *perception-cycle*. These processes, along with several sub-processes, are formally defined in Section 3.2.2. Table 3.1 provides a quick reference guide.

Table 3.1. Quick reference guide to the terminology of IntellegO.

Term	**Description**	**Example**
entity	An object or event in the world	apple
quality	An inherent property of an entity	red
quality-type	A category (or class) of qualities	color
percept	A quality that has been detected	*red*
observer	An agent that executes the observation-process	sensor
perceiver	An agent that executes the perception-process	computer
focus	A quality-type whose detection may reduce the perceptual-theory	color
perceptual-theory	A set of entities that each explains a set of percepts	{apple, nose}
inheres-in	A relation between a quality and an entity	red inheres-in apple
has-type	A relation between a quality and a quality-type	red has-type color

observation-process	An act of detecting a quality and generating a percept	observation-process(red) → *red*
perception-process	An act of inferring a perceptual-theory from a set of percepts	perception-process(red) →{apple, nose}
perception-cycle	An act of minimizing a perceptual-theory by focusing attention	perception-cycle(…) → {apple}

3.2.1. Semantics of Perception: Concepts and Relations

Imagine again that you are looking at the red apple depicted in Figure 3.1. The apple is an entity and red is a quality. The red quality is an inherent property of the apple entity. An *entity* is an object or event in the world; a *quality* is an inherent property of an entity; and *inheres-in* is a relation between a quality and an entity. Figure 3.3 illustrates an abstract set of inheres-in relations. A *quality-type* is a named category, or class, of qualities; such as color. A quality must have one-and-only-one quality-type; *has-type* is a relation between a quality and a quality-type. Different qualities associated with the same quality-type (through the has-type relation) are mutually exclusive. For example, red and green are mutually exclusive for the quality-type color. The background knowledge needed for perception, termed *perceptual-BK*, is composed of a set of has-type relations between qualities and quality-types, and a set of inheres-in relations between qualities and entities.

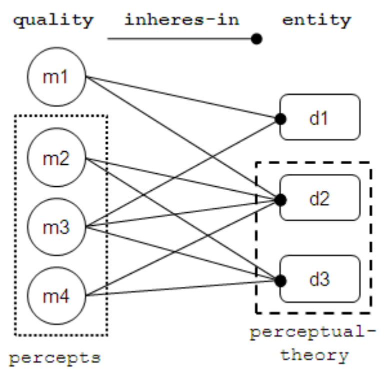


Figure 3.3. Set of inheres-in relations.

An *observer* is an agent that detects a quality. In this case, the observer is a human eye detecting the color red; however, the role of observer can be played by mechanical agents (e.g., sensors), biological agents (e.g., human eyes), or social agents (e.g., micro-blogs [Sheth09]). Upon detecting the color red, the mind brings forth an experience of redness embodied within an experience of an apple (of appleness). These mental experiences are referred to as qualia. While there is a clear distinction between a quality, or entity, and its associated qualia, we will make no such distinction in IntellegO.[6] A *percept* is a quality that has been detected by an observer. A *perceiver* is an agent that generates explanations for a set of percepts; for example, an apple may explain the red percept. In this case the perceiver is a human mind; however, the role of perceiver can be played by mechanical agents or biological agents [Goldstine64]. An explanation of a set of percepts is a set of entities, termed a *perceptual-theory*. Specifically, each entity in the perceptual-theory accounts for (explains) all the percepts. In order to refine or minimize a perceptual-theory, a perceiver may provide instructions to an observer to detect a particular quality-type. The phrase "*detect a particular quality-type*" should be interpreted as the detection of a member quality. When this occurs, the quality-type is termed *focus*. This ability to refine a

[6] While such a distinction may more accurately reflect, or approximate, human perception, in the authors' opinion it would also complicate IntellegO without adding sufficient utility.

perceptual-theory by employing focus is the key to efficient perception and will be further discussed in Section 3.2.2.3. Figure 3.4 provides an example of how qualities, quality-types, entities, percepts, and perceptual-theories are related.

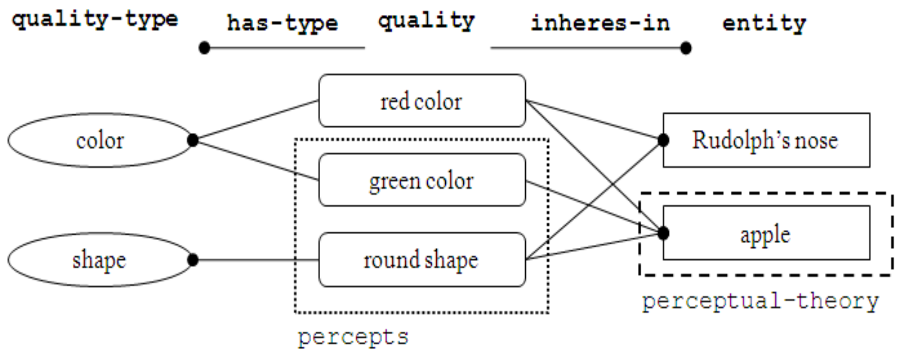


Figure 3.4. Example of how qualities, quality-types, entities, percepts, and perceptual-theories are related. The green color and round shape qualities have been detected and can be explained by the apple entity. Rudolph's ("the red-nosed reindeer") nose is not a member of the perceptual-theory since it cannot explain a set of percepts containing the green color quality.

3.2.2. Semantics of Perception: Processes

The three primary processes of IntellegO are: *observation-process*, *perception-process*, and *perception-cycle*. These processes are formally specified in set-theoretic notation. We have chosen to formalize the semantics of IntellegO in this manner because set-theory provides a notation that is unambiguous, well-established, and suitably expressive. Before these processes are defined, Table 3.2 provides a few required preliminary definitions.

Table 3.2. Required definitions for formalizing processes in IntellegO.

Term	**Description**
perceptual-BK = ⟨Q, E, I, QT, T⟩	Background knowledge about qualities, entities, and their relationships
Q	Set of all qualities
E	Set of all entities
$I \subseteq (Q \times E)$	Set of all inheres-in relations between qualities and entities
QT	Set of all quality-types
$T \subseteq (Q \times QT)$	Set of all has-type relations between qualities and quality-types
$P \subseteq Q$	Set of all percepts

3.2.2.1. Observation Process

While looking at the image in Figure 3.1, your eye detects the color red and generates an abstract representation in your mind.[7] This is an example of an *observation-process*. Given a quality-type as input, observation-process returns a detected quality, termed a percept. The observation-process represents an interface between an agent and the outside world. While we can define the input and output parameters of this process, the method in which a quality is detected is highly application dependent. For example, in many application scenarios the observation-process would activate a transducer which interacts with some physical stimuli, and this interaction is then interpreted as the detection of some quality in the world [Kuhn09]. For this reason, only the input and output parameters of *observation-process* are fully specified.

[7] Of course, this is an over simplification of the human visualization process, but is useful here for illustration purposes.

Definition: **observation-process** ($QT \rightarrow Q$)

```
observation-process(qt) = p, where (p ∈ Q) ∧ (qt ∈ QT) ∧
((p, qt) ∈ T) ∧ “p is detected”
```

A set of qualities is considered *valid* if the set contains at most one quality associated with each quality-type. This validity check reflects real-world constraints on a set of percepts and the nature of the background knowledge.

Definition: **valid** ($Q \rightarrow$ Boolean)

```
valid(ps) ⇒ (ps ⊆ Q) ∧ (∀ p1, p2 ∈ ps : (p1 ≠ p2) ⇒
(∃ qt1, qt2 ∈ QT : ((p1, qt1) ∈ T) ∧ ((p2, qt2) ∈ T) ∧
(qt1 ≠ qt2))
```

3.2.2.2. Perception Process

Again, while looking at the image in Figure 3.1, after detecting the color red, your mind attempts to explain the red color and generates an abstract representation of this explanation in your mind. An entity *explains* a set of qualities if the set of qualities are valid and each quality is an inherent property of the entity.

Definition: **explains** ($E \times$ Powerset(Q) $\rightarrow$ Boolean)

```
explains(e, ps) ⇔ (e ∈ E) ∧ (ps ⊆ Q) ∧ valid(ps) ∧
(∀ p ∈ ps : (p,e) ∈ I)
```

The process of generating explanations for a set of qualities is called the *perception-process*. The perception-process takes a set of qualities as input and yields a set of entities capable of explanation; that is, each entity in the set explains the set of qualities. The set of entities generated by a perception-process is called the perceptual-theory. In practice, the perception-process should attempt to explain only a set of percepts.

Definition: **perception-process** (Powerset(Q) → Powerset(E))

```
perception-process(ps) = { e ∈ E | explains(e, ps) }
```

3.2.2.3. Perception Cycle

The perception-process generates a perceptual-theory containing entities that explain a set of percepts. Notice that the perceptual-theory can contain multiple entities. This is not an ideal situation. When you look at Figure 3.1, you should perceive an apple, not an apple and/or Rudolph's nose. There are examples of human perception, however, where this type of ambiguity prevails. Consider the image in Figure 3.5: does this image depict a cup, or two human faces?

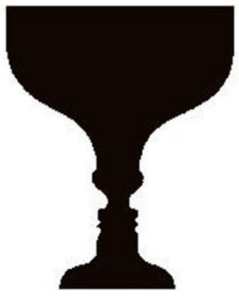


Figure 3.5. Is this a cup, or two human faces?

While such ambiguity may not always be ideal, the ability of the perceptual-theory to contain multiple explanatory entities is also what enables handling of incomplete or missing information (i.e., enables graceful degradation). For example, now (with incomplete information) we say that the image in Figure 3.5 depicts either a cup or two human faces; our perceptual-theory is {cup, two human faces}. However, if we were to subsequently detect eyes and/or ears, then (with additional disambiguating information) our perceptual-theory is refined to {two human faces}. The study of perceptual illusions has provided significant insights into the nature of perception [Gregory68][Gregory97]. Notwithstanding such cases, in general, we would like the resulting perceptual-theory to contain as few entities as possible – ideally, with the goal of reaching exactly one. This is because, in general, specificity improves action-ability. For example, the perceptual-theory resulting from Figure 3.1 is {apple} rather than the ambiguous {apple, Rudolph's nose}. The size of a perceptual-theory can often be reduced (i.e., concepts can be removed/filtered out) through the detection of new qualities, which must subsequently be explained. This cyclical process of observation and perception is called the *perception-cycle*. The perception-cycle is a process that relates the observation-process and perception-process; or, rather, relates observers and perceivers. An observer communicates percepts to a perceiver, representing qualities that have been detected, and the perceiver communicates focus to the observer, representing quality-types that should be detected. Figure 3.6 provides a graphical representation of the perception-cycle.

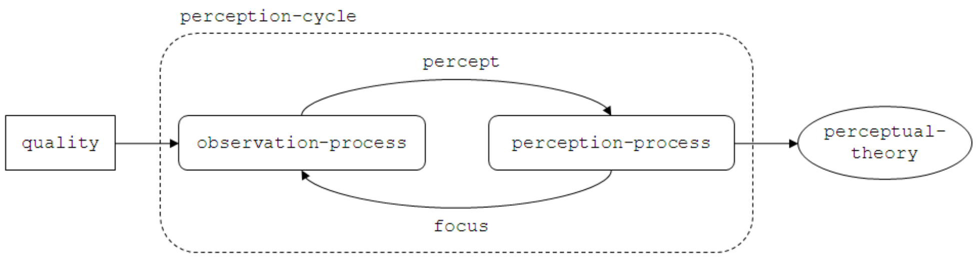


Figure 3.6. Architecture of the perception-cycle.

Within the perception-cycle, as new qualities are detected and the set of percepts grows, the size of the perceptual-theory shrinks. Thus, perception is an anti-monotonic process (that is, if *f* is a function that maps a set of percepts to a set of explanatory entities (i.e., perception-process), and *x*, *y* are sets of percepts, then $((x \subseteq y) \Rightarrow (f(x) \supseteq f(y)))$ **[**Gries99**]**. The anti-monotonic nature of the perception-cycle does not permit a straightforward formalization in first-order logic using standard deductive inference.

In order to optimize the perception-cycle, the observation-process should detect only those qualities that are capable of reducing the perceptual-theory. For example, if a perceptual-theory contains the entities apple and Rudolph's nose, detecting shape is probably of little use – i.e., cannot help discriminate between an apple and Rudolph's nose – since you can probably expect both to be round(ish). In order to clarify which qualities enable the reduction of the perceptual-theory, we will define four types of qualities: *expected*, *unknown*, *extraneous*, and *discriminating*. A quality is *expected* with respect to a set of entities if it is an inherent property of every entity in the set. Thus, if it were detected, it would be explained by every entity in the set. By definition, all percepts are expected. For example, given the perceptual-theory {*apple*, *Rudolph's nose*}, then the quality round-shape would be expected since both an *apple* and *Rudolph's nose* are round.

Definition: **expected** (Q × Powerset(E) → Boolean)

```
expected(q, es) ⇔ (q ∈ Q) ∧ (es ⊆ E) ∧ ¬empty(es) ∧
(∀e ∈ es : (q,e) ∈ I)
```

A quality is *unknown* with respect to a set of entities if it is not an inherent property of any entity in the set. Thus, if it were detected, it would not be explained by any entity in the set. By definition, no percepts are unknown. For example, given the perceptual-theory {*apple*, *Rudolph's* *nose*}, then the quality square-shape would be unknown since neither an *apple* nor *Rudolph's nose* are square.

Definition: **unknown** (Q × Powerset(E) → Boolean)

```
unknown(q, es) ⇔ (q ∈ Q) ∧ (es ⊆ E) ∧ ¬empty(es) ∧
(∀e ∈ es :(q,e) ∉ I)
```

A quality is *extraneous* with respect to a set of entities if it is either an inherent property of every entity in the set (expected), or is not an inherent property of any entity in the set (unknown). Thus, if it were detected, it would either be explained by all entities in the set or explained by no entities in the set. In either case, detection would not help to reduce the perceptual-theory. For example, given the perceptual-theory {*apple*, *Rudolph's nose*}, then the quality round-shape and the quality square-shape would both be extraneous (for different reasons).

Definition: **extraneous** (Q × Powerset(E) → Boolean)

```
extraneous(q, es) ⇔  expected(q, es) ∨ unknown(q, es)
```

A quality is *discriminating* with respect to a set of entities if its detection could potentially be used to reduce the size of the perceptual-theory. The set of discriminating qualities and the set of extraneous qualities are disjoint. Note that the set of discriminating qualities is not required to be a valid set. For example, for the perceptual-theory {*apple*, *Rudolph's nose*}, the quality green-color would be discriminating since an *apple* can be green while *Rudolph's nose* cannot.

Definition: **discriminating** (Q × Powerset(E) → Boolean)

```
discriminating(q, es) ⇔  ¬extraneous(q, es)
```

A perceptual-theory is *minimum* if it cannot be reduced through further observation. In other words, there are no qualities whose detection may discriminate between entities in the perceptual-theory.

Definition: **minimum** (Powerset(E) → Boolean)

```
minimum(es) ⇔ (∀q ∈ Q : extraneous(q, es))
```

With this terminology, we can now define a more specific goal of the perception-cycle: to generate a minimum perceptual-theory for a set of percepts. In order to achieve this goal efficiently, only those qualities capable of discriminating between entities in the perceptual-theory should be detected. To ensure only discriminating qualities are detected, a perceiver may provide instructions to (task) an observer to detect a particular quality-type, termed focus. This ability to refine a perceptual-theory by employing focus is the key to efficient perception. Focus is sent to an observation-process capable of detecting the represented quality-type. Given a set of entities (i.e., perceptual-theory), the *focus-candidates* process returns a set of quality-types which, when detected, can lead to reducing the perceptual-theory.

Definition: **focus-candidates** (Powerset(E) → Powerset(QT))

```
focus-candidates(es) = { qt ∈ QT | (∃q ∈ Q : (q ∉ P) ∧
discriminating(q, es) ∧ ((q, qt) ∈ T)) }
```

If there are several quality-types capable of reducing the perceptual-theory – i.e., several focus candidates – only one (at-a-time) may be designated as focus. The *choose* process takes a set of quality-types as input and returns a quality-type to observe. The method in which a single quality-type may be chosen from a set of quality-types is highly application dependent. For this reason, only the input and output parameters of the choose process are fully specified. In the next section, we evaluate several different implementations of the choose process.

Definition: **choose** (Powerset(QT) → QT)

```
choose(qts) = qt, where (qts ⊆ QT) ∧ (qt ∈ qts) ∧
"one qt is chosen"
```

The *perception-cycle* generates the minimum perceptual-theory. The process begins with the set of all known *entities* and an empty set of *percepts*, and repeatedly seeks suitable observations to better assess the situation. The resultant set of entities is progressively refined, to eventually obtain the minimum perceptual-theory. This perceptual-theory represents the best possible explanation(s) admissible by the given background knowledge (perceptual-BK). The perception-cycle algorithm proceeds as follows: (1) begin with a perceptual-theory and a set of percepts; (2) if the perceptual-theory is minimum then return; otherwise (3) generate and send focus to an observation-process and add the detected quality to the set of percepts; and finally, (4) update the perceptual-theory by removing those entities which cannot explain the updated set of percepts, and recursively continue the perception-cycle.

Definition: **perception-cycle** (Powerset(E) × Powerset(P) → Powerset(E))

Algorithm

```
input: es ⊆ E, ps ⊆ P
if minimum(es) then return es
else let aux = ps ∪ {observation-process(choose(
    focus-candidates(es)))}
    in perception-cycle(perception-process(aux), aux)
```

Function Call

```
initialize: es = E, ps = {}  // {} is the empty set
perception-cycle(es, ps)
```

3.2.3. Evaluation

In the following section, we provide three evaluations of IntellegO. In Section 3.2.3.1, we evaluate the sensing resources required for generating perceptual-theories, and show how focus, determined by the perception-cycle, can lead to improved efficiency. In Section 3.2.3.2, we evaluate the expressivity of IntellegO along two dimensions: (1) the ability to degrade gracefully with incomplete information, and (2) the ability to minimize explanations based on new information. IntellegO's capacity to embody these abilities is compared with current approaches, such as SWRL and first-order logic. In Section 3.2.3.3, we evaluate the resources required for storing sensor observations and perceptual-theories, and show that, for some applications, the generation and storing of perceptual-theories instead of raw observations can lead to significant – an order of magnitude – storage savings.

3.2.3.1. Focus Evaluation

The ability to focus attention enables a perceiver to more efficiently make sense of their environment. To better automate this process, IntellegO formalizes this ability. Specifically, we have implemented a prototype of IntellegO to run three experiments that demonstrate a realization of the perception-cycle and test the influence of focus on the interpretation of sensor data. Our metric of evaluation, used to compare the results of these experiments, includes the number of times the observation-process was executed in order to generate a minimum perceptual-theory. The number of times the observation-process is executed can also be represented by the size of the set of percepts to be explained (since each execution of the observation-process generates one percept). In the first experiment, we disable the focus ability of IntellegO and use background knowledge as discussed in Section 3.2.3.1.1. This is analogous to executing a brute force approach and serves as a base-line for comparison against subsequent experiments. This approach, however, is unfortunately the common modus operandi for collecting and interpreting data from sensor networks. In the second experiment, we enable the ability to focus and use the same background knowledge as the first experiment. In the third experiment, we enable the ability to focus and attempt to select an optimal focus using an enhanced background knowledge represented as a decision tree.

Claim – *The use of focus within the perception-cycle generates a perceptual-theory more efficiently (i.e., generates a smaller set of percepts) than the naïve brute force approach.*

Proof Sketch – Focus-candidates by definition discriminate between entities that can potentially serve as a viable explanation. Thus, observing a quality-type designated as focus is guaranteed to reduce viable explanations. On the other hand, observing quality-types other than the focus-candidates is of no consequence because either those values are already known (expected) or they are not relevant (unknown) given the current set of viable explanations. Thus, observing quality-types that are not focus-candidates requires wasteful computation and delays the determination of the minimum set of explanations.

Table 3.3 provides definitions used for the following experiments. The domain of interest is weather.

Table 3.3. Required definitions for evaluating IntellegO.

Term	**Description**
perceptual-BK = ⟨Q, E, I, QT, T⟩	Background knowledge about qualities, entities, and their relationships (see Figure 3.7)
Q	Set of all qualities = {*freezing-temperature, not-freezing-temperature, snow-precipitation, rain-precipitation, no-precipitation, high-wind-speed, low-wind-speed*}
E	Set of all entities = {*blizzard, flurry, rain-storm, rain-shower, clear*}
$I \subseteq (Q \times E)$	Set of all inheres-in relations (see Figure 9)
QT	Set of all quality-types = {*temperature, precipitation, wind-speed*}
$T \subseteq (Q \times QT)$	Set of all has-type relations (see Figure 9)
$P \subseteq Q$	Set of all percepts (generated during execution of the perception-cycle)

3.2.3.1.1. Background Knowledge for Focus Evaluation

The background knowledge, or perceptual-BK, utilized by these experiments is encoded as a graph representing the relationships between qualities and their types, and between qualities and the entities in which they inhere-in. The domain of interest is weather; therefore, the background knowledge contains weather related entities, such as *blizzard*, *flurry*, *rain-storm*, *rain-shower*, and *clear*. Weather related inherent qualities of these entities include *freezing-temperature*, *not-freezing-temperature*, *snow-precipitation*, *rain-precipitation*, *no-precipitation*, *high-wind-speed*, and *low-wind-speed*. The quality-types include *temperature*, *precipitation*, and *wind-speed*. Figure 3.7 illustrates the background knowledge related to weather. The concepts originated from the National Oceanic and Atmospheric Association (NOAA)[8] and are encoded in an ontology of weather.

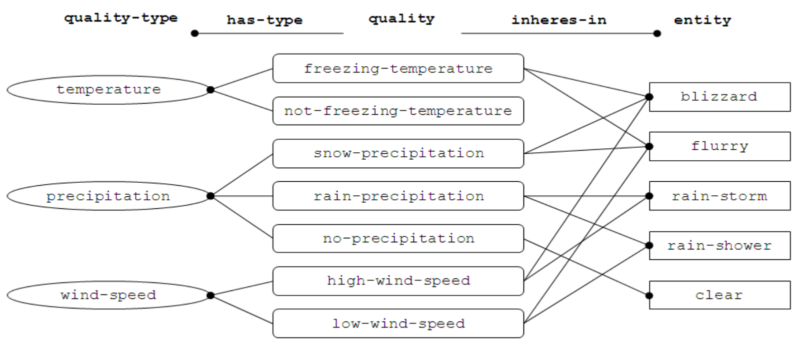


Figure 3.7. Background knowledge in the domain of weather. The graph shows how qualities and quality-types are related through the has-type relationship, and how qualities and entities are related through the inheres-in relationship.

The development of background knowledge is often a difficult challenge. In many domains, the relationships between qualities and entities are unclear, resulting in representations that may be imprecise and/or incomplete. While we acknowledge the difficulty, the development of such domain specific knowledge is out of the scope of this work. For examples of recent work in this area, see [Punuru07][Suchanek09][Thomas08].

3.2.3.1.2. Implementation

Before describing the experiments, we describe our implementation of IntellegO. The implementation is written in Java and conforms to the specification formalized in Section 3.2. Figure 3.8 shows an architecture diagram.

[8] NOAA, http://www.noaa.gov/

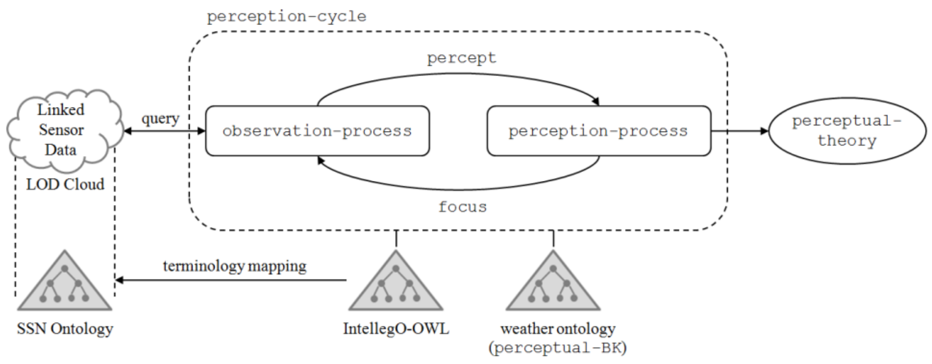


Figure 3.8. Architecture of an implementation of IntellegO.

The implementation of IntellegO respects the specification given in Section 3.2. However, two processes – observation-process and choose – were only partially defined; only the inputs and outputs were defined. The choose process is implemented differently for each experiment below, so the details will be described separately for each experiment. Note that the focus-candidates process returns a set of quality-types as an ordered sequence (the rationale for this decision is discussed in the experiment sections).

The implementation of the observation-process is highly dependent on the way in which sensor data is accessed. For example, the observation-process could be designed to task sensors in an environment and measure qualities in the world. For the evaluation presented in this paper, however, the sensor data has already been collected, encoded in RDF, and made accessible on the Web. Therefore, the observation-process, as implemented here, generates and executes a SPARQL [Prud'hommeaux08] query against the sensor data on LOD. SPARQL (SPARQL Protocol and RDF Query Language) is a W3C recommended language for querying RDF data. A set of Java libraries for managing Semantic Web data, Jena/ARQ [Carroll04], is used to build and execute the query.

The observation-process receives focus as input. In order to utilize this focus to generate an appropriate SPARQL query, in this implementation we also annotate focus with additional metadata, such as a time-interval and an observer (i.e., weather station). For example, given focus of quality-type *temperature*, time-interval *2003-04-01T02:00:00* to *2003-04-01T03:59:59*, and observer *System_SB1*, the observation-process will generate the following SPARQL query to detect the quality *freezing-temperature*.

```
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
prefix lsd: <http://knoesis.wright.edu/ssw#>
prefix weather: <http://knoesis.wright.edu/ssw/ont/weather.owl#>
prefix time: <http://www.w3.org/2006/time#>

ASK {
      ?observation ssn:observedProperty ?qualitytype .
      ?qualitytype rdf:type weather:temperature .
       ?observation ssn:observedBy lsd:System_SB1 .
      ?observation ssn:observationSamplingTime ?time .
      ?time time:inXSDDateTime ?datetime .
      ?observation ssn:observationResult ?result .
      ?result ssn:hasValue ?value .
      ?result weather:uom weather:Fahrenheit .
      FILTER(?value <= 32.0)
      FILTER(?datetime >= "2003-04-01T02:00:00"^^xsd:datetime)
      FILTER(?datetime <= "2003-04-01T03:59:59"^^xsd:datetime)
}
```

If the above ASK query returns *true* then the percept *freezing-temperature* is returned, otherwise the percept *not-freezing-temperature* is returned. The percept returned is an URI for the detected quality. SPARQL queries for the remaining quality-types are generated and executed in a similar manner. The above query is executed against the LinkedSensorData dataset [Patni10].

3.2.3.1.3. Experiment Setup

Between April 1st and April 6th of 2003, a major blizzard hit the state of Nevada. Environmental data within the surrounding area was collected by weather-stations, encoded as RDF, and made accessible on the Web as Linked Data. For every two hour interval and for each observer within a 400 mile radius of the blizzard, we execute the perception-cycle and generate a perceptual-theory. For each execution of the perception-cycle, the observer is a weather-station and the resulting perceptual-theory contains member entities representing the weather event occurring at that time and location (of the weather station). After each execution, the resultant perceptual-theory is checked for correctness and the total number of percepts, in the set of percepts, is counted.

During the execution of the perception-cycle, two variables affect the size of the set of percepts: (1) the order in which quality-types are detected, and (2) the weather conditions surrounding the weather-station at the time of observation. The first issue is addressed by the way a quality-type is chosen to be observed by the *choose* process. In order to address the second issue, we evaluate a variety of weather conditions by executing the perception-cycle with observers at various distances from the blizzard. More specifically, we execute the perception-cycle with observers within a distance of 25 miles (17 observers), 50 miles (70 observers), 100 miles (170 observers), 200 miles (373 observers), and 400 miles (516 observers).

Given a particular time-interval and observer, in addition to the weather background knowledge described in Section 3.2.3.1.1., the minimum perceptual-theory generated by the perception-cycle should always be the same, despite the order in which quality-types are detected. The order only affects how efficiently the perceptual-theory is generated. Thus, precision and recall statistics do not make sense in this evaluation and will not be shown. In these experiments the minimum perceptual-theory always contains either zero or one entity [note that this is a product of how the specific qualities and entities are related within the weather background knowledge and not a general rule]. As such, the resulting perceptual-theory will be labeled with a single term representing the single member entity that explains the set of percepts (e.g., *blizzard*). If the perceptual-theory contains no member entities (representing the empty set) then the perceptual-theory is labeled as *unclassified*.

To allow validation, repeatability, and further experimentation, we have stored the data generated by these experiments as RDF, accessible at: http://wiki.knoesis.org/index.php/Intellego.

3.2.3.1.4. Experiment 1: No Focus (brute force approach)

An experiment to test the naïve brute force approach was conducted by executing the perception-cycle as described in Section 3.2.2.3., with one major retraction: the ability to check for discriminating qualities and generate focus was disabled. The set of quality-types to choose from is given as an ordered sequence: {*temperature*, *precipitation*, *wind-speed*}. The choose process simply returns the first quality-type in this ordered sequence. Since, in this scenario, all quality-types must be detected, the order of detection is irrelevant and the number of percepts generated for each execution of the perception-cycle remains constant. For this experiment, we executed the perception-cycle 37,152 times; once for each combination of time interval (72) and observer

(516). Each execution generated a set of percepts. Table 3.4 shows the total number of percepts generated for each set of observers.

Table 3.4. The total number of percepts generated during all the executions of the perception-cycle during experiment 1 (for different sets of observers).

25 miles (17 observers)	50 miles (70 observers)	100 miles (170 observers)	200 miles (373 observers)	400 miles (516 observers)
3672	15,120	36,720	80,568	111,456

Table 3.5 shows the types of perceptual-theories that resulted from the execution of the perception-cycle for the different sets of observers. From this table, we can see that the *clear* condition is by far the most common while *blizzard* and *unclassified* rarely occur. There is a minor trend towards a decreasing percentage of *blizzard*, *flurry*, and *unclassified* theories, and an increasing percentage of *clear* theories, as the distance from the blizzard increases. This trend seems reasonable, since as you move farther away from the blizzard, the weather is more likely to be clear.

Table 3.5. Shows the percentage of different perceptual-theories generated during the execution of the perception-cycle (for different sets of observers).

distance (# observers)	***blizzard***	***flurry***	***rain- storm***	***rain- shower***	***clear***	***unclassified***
25 miles (17)	1.77%	22.21%	0%	17.65%	58.36%	3.5%
50 miles (70)	0.5%	17.19%	0.44%	14.04%	67.8%	1.7%
100 miles (170)	0.25%	14.08%	0.95%	14.83%	69.86%	1.2%
200 miles (373)	0.11%	9.64%	2.25%	20.5%	67.48%	0.9%

400 miles (516)	0.09%	9.11%	2.41%	21.58%	66.81%	0.85%

3.2.3.1.5. Experiment 2: With Focus

The choose process for this experiment is the same as in the first experiment. This time, however, the ability to generate focus is enabled and thus the focus-candidates process returns an ordered sequence of only those quality-types that may discriminate between entities in the perceptual-theory. Under these conditions, the ordering in the ordered sequence of quality-types affects the number of percepts generated.

Consider the following illustrative examples. Suppose we have a perceptual-theory {*blizzard*, *flurry*, *rain-shower*}. First, consider the ordered sequence of quality-types {*precipitation*, *wind-speed*}. The choose process will pick *precipitation* as focus. Now suppose *snow-precipitation* is detected, resulting in an updated perceptual-theory {*blizzard*, *flurry*}. This perceptual-theory is not minimum, so the perception-cycle will continue. The next ordered sequence of quality-types is now {*wind-speed*}, so the choose process will pick *wind-speed* as focus. If we suppose that *high-wind-speed* is detected, the resulting perceptual-theory is {*blizzard*}. This is a minimum perceptual-theory, so the perception-cycle terminates. In total, two percepts were generated, *snow-precipitation* and *high-wind-speed*. Now consider the case where the ordered sequence of quality-types has been changed to {*wind-speed*, *precipitation*}. The choose process will pick *wind-speed* as focus. Again suppose *high-wind-speed* is detected, resulting in an updated perceptual-theory {*blizzard*}. This perceptual-theory is minimum, so the perception-cycle terminates. In this case, only one percept was generated, *high-wind-speed*.

Since we do not know a-priori which ordered sequence of quality-types will produce the optimal result, we test each possible sequential order. Given the three quality-types in the background knowledge – *temperature* (t), *precipitation* (p), and *wind-speed* (w) – there are six possible orderings. Thus, for this experiment, we executed the perception-cycle 222,912 times; once for each time interval (72), observer (516), and permutation of quality-types (6). As expected, the perceptual-theories generated during the execution of the perception-cycle, as shown in Table 3.5, were found to be identical in this experiment. Figure 3.9 shows the results of executing the perception-cycle for the 516 weather-stations within a radius of 400 miles of the blizzard, for each time interval, and for each ordering of quality-types.

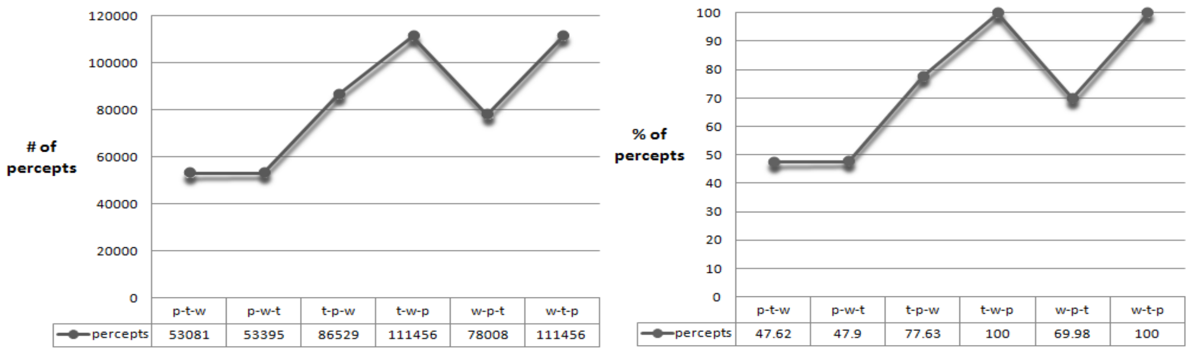


Figure 3.9. Percepts generated by observers within 400 miles of a known blizzard. The horizontal axis represents the different orderings of observable qualities; *p* represents *precipitation*, *w* represents *wind-speed*, and *t* represents *temperature*.

For each ordering of quality-types, 37,152 perceptual-theories were generated from 111,456 potential quality-type detections. However, in many cases the number of percepts is far less than the number of quality-types that could potentially be detected. For example, with two of the orderings (*p*-*t*-*w* and *p*-*w*-*t*) the ratio is less than 48%; this accounts for a significant reduction in the number of percepts needed to generate the minimum perceptual-theory. On the other hand, with two of the orderings (*w*-*t*-*p* and *t*-*w*-*p*) the ratio is 100%; all of the quality-types were detected resulting in the maximum number of percepts. Looking at Figure 3.9, we see that the two orderings with the best results (*p*-*w*-*t* and *p*-*t*-*w*) both begin with *precipitation*; and the two orderings with the worst results (*w-t-p* and *t-w-p*) both place *precipitation* last in the order. This may inform us that a *precipitation* percept is proficient in discriminating between entities in the perceptual-theory. From the weather background knowledge in Figure 3.7 this can be more clearly seen by noticing that the *no-precipitation* quality only inheres-in the entity *clear*. This means that a *no-precipitation* percept is only explained by the entity *clear*; if this quality is detected then the minimum perceptual-theory is found. Therefore, detecting the *precipitation* quality-type early is an efficient approach. This experiment clearly shows that the order in which quality-types are evaluated and detected dramatically affects the efficiency of the perception-cycle.

The statistics of the remaining executions of the perceptual-cycle – for each set of observers within a radius of 25, 50, 100, 200, and 400 miles – are shown in Table 3.6. You may notice that the percentage of percepts remains fairly stable across the different sets of observers. There is a minor trend towards a decreasing percentage of percepts as the distance from the blizzard increases. This trend can be explained by noticing that complex weather conditions (e.g., blizzard) may require more percepts to explain than more simple weather conditions (e.g., clear).

Table 3.6. Shows the percentage of percepts generated during the execution of the perception-cycle (for different sets of observers and orderings of quality-types).

distance (# observers)	**p-t-w**	**p-w-t**	**t-p-w**	**t-w-p**	**w-p-t**	**w-t-p**
25 miles (17)	55.58%	56.67%	80.09%	100%	75.49%	100%
50 miles (70)	50.25%	50.82%	77.21%	100%	73.03%	100%
100 miles (170)	48.38%	48.80%	76.58%	100%	71.80%	100%
200 miles (373)	47.60%	47.92%	77.40%	100%	70.20%	100%
400 miles (516)	47.62%	47.90%	77.63%	100%	69.98%	100%

3.2.3.1.6. Experiment 3: With Optimized Focus

The previous experiment showed that while focus is useful for efficient perception, these gains in efficiency are dependent on knowing the optimal sequential ordering of quality-types to focus attention. Since the number of possible sequential orderings grows exponentially with the total number of quality-types ($O(n!)$, where n = # of quality-types), such an approach to arriving at the optimal order may not be suitable for Web-scale data. One possible solution to the scalability issue would be to learn the optimal ordering of quality-types by analyzing a *representative* training dataset. For this approach, we used the standard decision-tree algorithm (C4.5 [Mitchell97]). Starting with a representative training dataset that has been annotated with correct classifications, a decision-tree representation capable of efficient classification is generated. This algorithm is able to compute the optimal ordering of quality-types in polynomial-time with respect to the total number of quality-types ($O(n^3 * \max(v)^n$, where n = # of quality-types and v = # of discrete qualities for a quality-type). The polynomial-time complexity of this algorithm is a drastic improvement over the exponential-time complexity of the technique employed in the second experiment (Section 3.2.3.1.5.).

In the third experiment, we executed the C4.5 decision-tree algorithm over a training dataset which includes a representative sample of quality detections within 400 miles of the blizzard in Nevada. An excerpt of the training dataset is shown in Table 3.7, and the resulting decision-tree is shown in Figure 3.10. The choose process for this experiment is more complex than in the first or second experiment. Instead of simply returning the first quality-type in the sequence, the choose process returns the quality-type represented by the current node in the decision-tree, which has the highest informational value (i.e., highest information-gain)[9]. The quality-type represented by this node, therefore, is the optimal choice for focusing attention. As noted in Section 3.1, Peter Norwich first used Information Theory to quantify the informational value of observations; he called this the Entropy Theory of Perception [Norwich91]. This experiment goes (slightly) further and orders the observable quality-types based on their informational value and encodes this order in a decision tree data-structure.

Table 3.7. Excerpt from the representative training dataset.

wind speed	temperature	precipitation	classification
high	freezing	snow	blizzard
low	freezing	snow	flurry
low	freezing	snow	flurry
high	not-freezing	rain	rain-storm
low	not-freezing	rain	rain-shower
low	not-freezing	none	clear
low	not-freezing	none	clear
high	not-freezing	none	clear

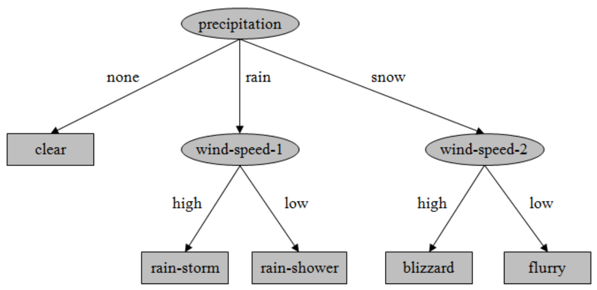


Figure 3.10. Decision tree representing the optimal sequential ordering of quality-types.

The optimal ordering of quality-types, shown in Figure 3.10, begins with *precipitation* – at the root of the tree – followed by *wind-speed*. This is consistent with the previous optimal orderings found in Section 3.2.3.1.5 (i.e., *p*-*w*-*t*). Notice, however, that temperature is not represented

[9] The C4.5 decision-tree algorithm is based on Claude Shannon's Information Theory [Shannon48], and the current node in the tree corresponds to the attribute from the training dataset with the highest *information-gain* [Mitchell97].

within the decision-tree. This omission is the result of a discovery by the decision-tree algorithm that *temperature* is always extraneous.

For this experiment, we executed the perception-cycle 37,152 times; once for each combination of time interval (72) and observer (516). Table 3.8 shows the number and percentage of percepts generated for each set of observers. The results show an average 50% reduction in the number of percepts needed to generate a minimum perceptual-theory; these results mirror those found in Section 3.2.3.1.5 for the sequential ordering of quality-types {*precipitation*, *wind-speed*, *temperature*}.

Table 3.8. Shows the percentage of percepts generated during the execution of the perception cycle (for different sets of observers and orderings of quality types).

distance (# observers)	# of percepts	% of percepts
25 miles (17)	2081	56.67%
50 miles (70)	7684	50.82%
100 miles (170)	17921	48.80%
200 miles (373)	38613	47.92%
400 miles (516)	53395	47.90%

3.2.3.2. Expressivity Evaluation

In our design and representation of perception, we emphasize two important capabilities: (1) the ability to degrade gracefully with incomplete information, and (2) the ability to minimize explanations based on new information. Current solutions to the perception problem often encode the background knowledge within first-order logic (FOL). In particular, Ricquebourg

[Ricquebourg07], Keßler [Keßler09], and Sheth [Sheth08] and have all used the Semantic Web Rule Language (SWRL) [Horrocks04] to encode such background knowledge and infer explanations. SWRL is a restricted fragment of FOL (see [Horrocks04] for additional details). Calder [Calder10] and Henson [Henson09] have also used the Jena Rule Engine for this task. In the following section, we compare IntellegO with SWRL and first-order logic; and summarize our results in Table 3.9. While IntellegO is expressive enough to achieve both capabilities, SWRL achieves neither, and FOL can degrade gracefully with incomplete information, but cannot minimize explanations. The ability to minimize explanations is an anti-monotonic process; and therefore, it is not surprising that the monotonic FOL and the more restrictive SWRL cannot express such a process.

Table 3.9. Qualitative comparison of logic frameworks to express the desired capabilities of perception; including the ability to degrade gracefully with incomplete information and the ability to minimize explanations with additional information.

	Graceful Degradation	Minimize Explanations
IntellegO	Yes	Yes
SWRL	-	-
FOL	Yes	-

3.2.3.2.1. Background Knowledge for Expressivity Evaluation

To compare the expressivity of existing approaches, we will provide an example scenario based on the background knowledge represented in Figure 3.11 and show how each approach behaves.

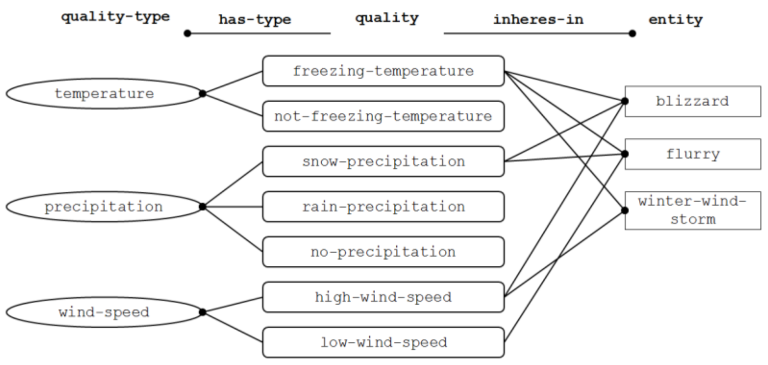


Figure 3.11. Example background knowledge used within the expressivity evaluation of IntellegO (Section 3.2.3.2).

3.2.3.2.1.1. Encoding of Rules in SWRL

A straightforward encoding of Figure 3.11 in SWRL, based on the SSN ontology, is shown below. In this encoding, each entity is defined as a rule, as demonstrated by [Keßler09][Ricquebourg07][Sheth08]. For each rule, we assume that a generic entity individual has been created and added to the knowledge base for each spatial-temporal context. For simplicity, all spatial-temporal and value constraints are removed, since they complicate the rules without differentiating the approach (note that in a real-world application, such constraints must be added).

<u>Blizzard Rule</u>

```
ssn:isQualityOf(high-wind-speed, ?e) Λ
ssn:isQualityOf(freezing-temperature, ?e) Λ
ssn:isQualityOf(snow-precipitation, ?e)
→ blizzard(?e)
```

<u>Flurry Rule</u>

```
ssn:isQualityOf(low-wind-speed, ?e) ∧
ssn:isQualityOf(freezing-temperature, ?e) ∧
ssn:isQualityOf(snow-precipitation, ?e)
→ flurry(?e)
```

<u>Winter Wind Storm Rule</u>

```
ssn:isQualityOf(high-wind-speed, ?e) ∧
ssn:isQualityOf(freezing-temperature, ?e) ∧
→ winter-wind-storm(?e)
```

In order to satisfy the entity rules defined above, each of the *ssn:isQualityOf* predicates in the antecedent must be satisfied. This is achieved if a corresponding observation is found in the knowledge base, as defined in the following rule:

<u>Observation Rule</u>

```
ssn:observedProperty(?o,?q) ∧ ssn:featureOfInterest(?o,?e)
→ ssn:isQualityOf(?q, ?e)
```

3.2.3.2.1.2. Additional Rules in FOL

First-order logic (FOL), in general, is more expressive than SWRL. In particular, FOL provides the disjunction (∨) and negation (¬) operator – which are not permissible in SWRL [Mei04] – that may be utilized to infer more complex explanations. In addition to the SWRL rules defined above, the following FOL rules can be added in order to guarantee that a set of observed qualities (i.e., percepts) are mutually exclusive and collectively exhaustive for each quality-type. These rules also require the introduction of the *has-type* relation from IntellegO, which explicitly relates qualities to quality-types. The *mutually exclusive criterion* says that at most one quality, per quality-type, may be detected for each entity.

Mutually Exclusive Rule

```
ssn:isQualityOf(?q, ?e) ∧ io:has-type(?q, ?t)
→ [∀?q′ ∈ Q : (?q′ ≠ ?q) ∧ io:has-type(?q′, ?t) →
¬ssn:isQualityOf(?q′, ?e)]
```

The *collectively exhaustive criterion* says that at least one quality, per quality-type, must be detected for each entity. This criterion may not be generally true and is not defined in IntellegO; but is necessary to infer suitable explanations with FOL.

Collectively Exhaustive Rule

```
∀?e ∈ E, ∀?t ∈ QT :[∃ ?q ∈ Q : ssn:isQualityOf(?q, ?e) ∧
io:has-type(?q, ?t)]
```

With the addition of these two rules, and given a set of observations, FOL is able to generate an explanation as a disjunction of entities. As seen in the comparisons below, this enables satisfactory results in relation to the ability to degrade gracefully with incomplete information.

3.2.3.2.2. Expressivity Comparison

Given the background knowledge described above, the following scenarios will provide additional facts to the knowledge base. Each will then show the inference results from IntellegO, a SWRL inference engine, and a FOL inference engine.

3.2.3.2.2.1. Comparison 1: The ability to degrade gracefully with incomplete information

Given observations for *freezing-temperature* and *snow-precipitation*, in addition to the background knowledge above, the expected perceptual-theory would be {*blizzard*, *flurry*}. That is, *blizzard* is an explanation, and *flurry* is an explanation. Notice that *freezing-temperature* and *snow-precipitation* do not completely describe either *blizzard* or *flurry*, but they both provide evidence for *blizzard*, and they both provide evidence for *flurry*.

Facts in knowledge base

```
ssn:Entity(e)
ssn:Observation(o1)
ssn:observedProperty(o1, snow-precipitation)
ssn:featureOfInterest(o1, e)
ssn:Observation(o2)
ssn:observedProperty(o2, freezing-temperature)
ssn:featureOfInterest(o2, e)
```

Given the above facts in the knowledge base resulting from observations, in addition to the rules in Section 3.2.3.2.1.1 and Section 3.2.3.2.1.2, the following results:

<u>Resulting explanations (i.e., perceptual-theory)</u>

IntellegO: {blizzard, flurry}

SWRL:

FOL: blizzard(e) ⋁ flurry(e)

From this example, IntellegO provides *blizzard* as an explanation, and *flurry* as an explanation, for the set of observations. SWRL, on the other hand, does not provide any explanation. FOL provides *blizzard* ⋁ *flurry* as the explanation. This result occurs by virtue of the fact that *high-wind-speed* and *low-wind-speed* are the only two qualities of quality-type *wind-speed*. This knowledge thus allows FOL to derive meaningful partial information. We can see that both IntellegO and FOL allow explanations to degrade gracefully with incomplete information, while SWRL does not.

3.2.3.2.2.2. Comparison 2: The ability to minimize explanations based on new information

In the initial situation, we are given observations for *high-wind-speed* and *freezing-temperature*.

<u>Initial situation</u>

Facts in knowledge base

ssn:Entity(e)

ssn:Observation(o1)

ssn:observedProperty(o1, high-wind-speed)

ssn:featureOfInterest(o1, e)

ssn:Observation(o2)

ssn:observedProperty(o2, freezing-temperature)

ssn:featureOfInterest(o2, e)

Given the above facts in the knowledge base resulting from observations, in addition to the rules in Section 3.2.3.2.1.1 and Section 3.2.3.2.1.2, the following results:

<u>Resulting explanations (i.e., perceptual-theory)</u>

IntellegO:	{winter-wind-storm, blizzard}
SWRL:	winter-wind-storm(e)
FOL:	winter-wind-storm(e)

Now suppose that the *precipitation* quality-type is detected and an additional *snow-precipitation* observation is added to the knowledge base. Given observations for *high-wind-speed*, *freezing-temperature*, and *snow-precipitation*, in addition to the background knowledge above, the expected perceptual-theory would be {*blizzard*}.

<u>Updated situation with new observation</u>

Facts in knowledge base

ssn:Entity(e)

ssn:Observation(o1)

ssn:observedProperty(o1, high-wind-speed)

ssn:featureOfInterest(o1, e)

ssn:Observation(o2)

ssn:observedProperty(o2, freezing-temperature)

ssn:featureOfInterest(o2, e)

ssn:Observation(o3)

ssn:observedProperty(o3, snow-precipitation)

ssn:featureOfInterest(o3, e)

Given the above facts in the knowledge base resulting from observations, in addition to the rules in Section 3.2.3.2.1.1 and Section 3.2.3.2.1.2, the following results:

<u>Resulting explanations (i.e., perceptual-theory)</u>

IntellegO:	{blizzard}
SWRL:	winter-wind-storm(e), blizzard(e)
FOL:	winter-wind-storm(e) ∧ blizzard(e)

From this example, IntellegO provides *blizzard* as the only explanation for the set of observations, while SWRL and FOL provide *winter-wind-storm* and *blizzard* as explanations. With IntellegO, as more information (i.e., observations) is provided, the set of explanations is reduced to improve specificity. With SWRL and FOL, on the other hand, as more information is provided, the set of explanations can grow only larger. We can see that IntellegO has the ability to minimize explanations based on new information in an intuitively satisfactory way, while both SWRL and FOL are unable to do so due to their monotonicity.

3.2.3.3. Storage Requirements and Scalability Evaluation

In the digital age, information is being generated at an extraordinary pace. It has even been suggested that *more data has been created in the last three years than in the past 40,000*. Around 2008, this rate of expansion surpassed the generation rate of storage capacity, leading to a future where we can no longer store all the sensor data being generated [Higginbotham10]. With this future ahead, the efficient representation and interpretation of sensor data at scale has become an important area of research.

A single flight from New York to Los Angeles on a twin-engine Boeing 737 generates around 240 terabytes of data; and, on any given day, there are around 28,537 such commercial flights in the United States [Higginbotham10]. But how much of this sensor data is actually useful for decision-making and needs to be stored for later retrieval? The pilot may need to understand the general condition of the plane and the external environment during flight; the ground crew may need to know the speed, location and trajectory of the aircraft; and the mechanic may need to know of any anomalous behavior detected during the flight. Much of this knowledge must be derived from the low-level observational data, but only the high-level inferences may be required for actual decision-making (with possible inspection or analysis of a very small minority of the raw data).

As another example, consider a weather alert service which represents, stores, and alerts people of the type of current weather conditions. In this case, all the low-level observations can be discarded while only the perceptual-theories need to be stored. If the weather alert service only generates alerts for severe weather conditions, then perceptual-theories representing "irrelevant" weather conditions, such as *clear*, can also be discarded; such information is unnecessary for the task of alerting people of severe weather.

We evaluate the resources required for storing sensor observations and perceptual-theories, and show that for some applications the generation of perceptual-theories can lead to significant storage savings. For this evaluation, we will count the number of records that are generated and stored for different dataset storage configurations. A record could represent an observable quality, percept, or perceptual-theory. The statistics used in the following five data storage configurations are summarized in Figure 3.12.

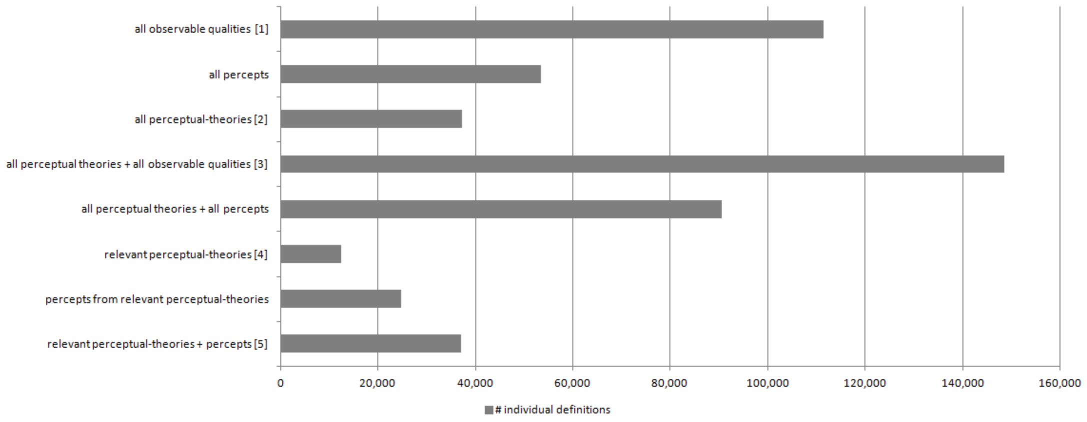


Figure 3.12. Storage requirements for several common dataset configurations; based on statistics gathered from the experiment discussed in Section 3.2.3.1.

- All observable qualities are stored,
- All perceptual-theories are stored,
- All observable qualities and all perceptual-theories are stored,
- Only *relevant* perceptual-theories are stored, and
- Only *relevant* perceptual-theories and *relevant* percepts are stored.

All observable qualities are stored – Within this dataset configuration, percepts for all possible observable qualities are generated and stored. From the experiment in Section 3.2.3.1, this configuration generates 111,456 records.

All perceptual-theories are stored – This dataset configuration generates and stores perceptual-theories for all weather conditions (within some spatial-temporal context). From the experiment in Section 3.2.3.1, this configuration generates 37,152 records. The ratio of records generated for **(1) all observable qualities** versus **(2) all perceptual-theories** is around 3:1. Thus, if only perceptual-theories are required for an application then the storage requirements result in 66% savings.

All observable qualities and all perceptual-theories are stored – This configuration represents the situation from our experiment in Section 3.2.3.1.4 that executed a brute force approach to observe all qualities and generate all percepts and perceptual-theories. During this experiment, 111,456 percept records and 37,152 perceptual-theory records are generated and stored, totaling 148,608 records.

Only *relevant* perceptual-theories are stored – Within this configuration, perceptual-theories for only *relevant* (*severe)* weather conditions (within some spatial-temporal context) are defined and stored. From the experiment in Section 3.2.3.1, we can define a *relevant* perceptual-theory as any perceptual-theory representing a *blizzard*, *flurry*, *rain-storm*, or *rain-shower* condition. 12,331 records are generated with this configuration. The ratio of records generated for **(1) all observable qualities** *versus* **(4) only *relevant* perceptual-theories** is around 9:1. The ratio of records generated for **(3) all observable qualities and all perceptual-theories** *versus* **(4) only *relevant* perceptual-theories** is around 12:1. These represent over an order of magnitude storage savings.

Only *relevant* perceptual-theories and *relevant* percepts are stored – It may also be important to store the percepts associated with the *relevant* perceptual-theories. Such information may be useful for validation, additional analysis, or simply allowing for further investigation. For example, suppose the *severe* weather alert service described above also allowed the user to access the observation records that were used to determine the *severe* weather condition (e.g., to see just how fast the wind is blowing). From the experiment in Section 3.2.3.1, this configuration generates 36,993 records. The ratio of records generated for **(3) all observable qualities and all perceptual-theories** *versus* (**5) only *relevant* perceptual-theories and *relevant* percepts** is around 4:1.

This evaluation illustrates the benefits that come from the ability to abstract away from the details of low-level sensory input and generate high-level explanations. While the application scenarios may vary, and the definition of *relevance* is highly dependent on the domain of interest and application, such examples clearly show the benefits, from generating perceptual-theories with IntellegO.

3.3. Ontology of Perception – OWL

Emerging solutions to the challenge of machine perception are using ontologies to provide expressive representation of concepts in the domain of sensing and perception, which enable advanced integration and interpretation of heterogeneous sensor data on the Web. As discussed in Section 2, the W3C Semantic Sensor Network Incubator Group [SSN-XG] has recently developed the Semantic Sensor Network (SSN) ontology [Lefort11][Compton12] that enables expressive representation of sensors, sensor observations, and knowledge of the environment. The SSN ontology is encoded in the Web Ontology Language (OWL) and has begun to achieve broad adoption within the sensors community [Gray11][Calbimonte11][Pfisterer08]. Such work is leading to a realization of a Semantic Sensor Web.

OWL provides an ideal solution for defining an expressive representation and formal semantics of concepts in a domain. As such, the SSN ontology serves as a foundation for defining the semantics of machine perception in OWL. In this section, we present a formal definition of two primary inference tasks, in OWL, that are generally applicable to machine perception – explanation and discrimination – as an extension of the SSN ontology. A complete representation of IntellegO, including the perception cycle, is not provided since such an encoding is beyond the expressivity of OWL.

3.3.1. Abduction in OWL

Abduction is often described as inference to the best explanation. Given some background knowledge and a set of observations, an abductive reasoner will compute a set of best explanations. In general, abduction is formalized as $\Sigma \wedge \Delta \vDash \Gamma$ where background knowledge Σ and observations Γ are given, and explanations Δ (also called abducibles) are to be computed ($\vDash$ refers to the first-order logic consequence relation). One highly popular abductive reasoning framework is the Parsimonious Covering Theory (PCT) [Reggia87]. PCT has predominantly been used in the domain of medical disease diagnosis. Reasoning in PCT is executed by algorithms that support a hypothesize-and-test inference process, and is driven by background knowledge modeled as a bipartite graph causally linking disorders to manifestations. The basic premise of PCT is that diagnostic reasoning can be divided into two parts: coverage and parsimony. The coverage criterion describes how to generate a set of explanations such that each given observation is caused by a disorder in the explanation (an observation is a manifestation that has been observed). In complicated domains, such as medical disease diagnosis, the number and size of explanations may grow to be large. In order to reduce to a more reasonable size, the parsimony criterion describes how to choose which explanations are best. Many different parsimony criteria have been advanced, including minimum cardinality criterion, subset minimality (irredundancy) criterion, etc. [Reggia87]. The single disorder assumption is a simple, yet effective, parsimony criterion that has also proved popular in the past; it states that explanations may contain only a single disorder.

While OWL [Hitzler09] may not have been designed for abductive reasoning, the integration of OWL and abduction has been explored [Elsenbroich06]; however previous approaches have required modification of OWL syntax and/or an OWL inference engine [Peraldi09]. In this section, we will demonstrate that OWL does provide some of the expressivity needed to approximate diagnostic reasoning – without extension of its syntax or semantics – by outlining a suitable encoding of PCT in OWL-DL. We caution the reader, however, that the OWL representation discussed does not explicitly implement PCT, but only approximates PCT, since OWL inference does not support a hypothesize-and-test inference

3.3.1.1. Parsimonious Covering Theory (PCT)

PCT is an abductive logic framework that provides a formal model of diagnostic reasoning that represents knowledge as a network of binary relations. The goal of PCT is to account for observed symptoms (qualities) with plausible explanatory hypotheses (entities). PCT has predominantly been used in medical disease diagnosis. Reasoning in PCT uses a hypothesize-and-test inference process and is driven by background knowledge modeled as a bipartite graph relating entities to qualities.

PCT divides diagnostic reasoning into two parts: coverage and parsimony. The *coverage criterion* describes how to generate a set of explanations such that each observation is accounted for by an entity in the explanation (where an *observation* is a quality that has been observed). To reduce the set of explanations to a reasonable size, the *parsimony criterion* describes how to select the best explanations. Researchers have advanced many different parsimony criteria: minimum cardinality criterion, subset minimality (irredundancy) criterion, and so on [Reggia87]. The single-entity assumption is a simple yet effective parsimony criterion that has proved popular for medical disease diagnosis. It states that explanations may contain only a single entity.

Consider the process of abduction in which background knowledge $\Sigma = \langle Q, E, C \rangle$, observations Γ are given, and explanations Δ are to be inferred. Specifically, an abduction problem P (in PCT) is a 4-tuple $\langle Q, E, C, \Gamma \rangle$, in which Q is a finite set of qualities, E is a finite set of entities, $C : E \rightarrow$ Powerset(Q) is the causation function that maps an entity to the corresponding set of qualities it causes, and $\Gamma \subseteq Q$ is the set of observations. For any entity $e \in E$ and quality $q \in Q$, effects(q) = C(e) and causes(q) = $\{e \mid q \in C(e)\}$. effects(E) = $\bigcup_{e \in E}$ effects(e). The set $E_I \subseteq E$ is said to be a *cover* of $Q_J \subseteq Q$ if $Q_J \subseteq$ effects(E_I). A set $\Delta \subseteq E$ is an *explanation* of Γ for a problem $\langle E, Q, C, \Gamma \rangle$ if and only if Δ covers Γ and satisfies a given parsimony criterion. A cover E_I of Q_J is said to be minimal if its cardinality is smallest among all covers of Q_J. A cover E_I of Q_J is said to be irredundant if none of its proper subsets is also a cover of M_J [Reggia87].

Thus, an explanation is a *cover* if, for each observation, there is a *causal* relationship within the background knowledge from an entity contained in the explanation to the observed quality. (We are implicitly using the one-to-one correspondence between a function over $E \rightarrow$ Powerset(Q) and its equivalent rendering as a relation over $E \times Q$.) An explanation is *parsimonious* (the best) if it contains only a single entity. Thus, an explanation is a parsimonious cover if it contains only a single entity that explains all observations.

3.3.1.2. Translating PCT into OWL

Using RDF and OWL to represent information on the Web — and employing OWL reasoners to infer new information — is gaining support. For this reason, and given the increasing number of observations on the Web, it makes sense to explore using these languages to model the perception process. However, OWL isn't designed for representing abductive inference. So, existing OWL ontologies have limited ability to formalize perceptions and derive explanations. Nevertheless, OWL does provide some of the expressivity required to derive explanations from observations, and we have developed a suitable encoding of PCT in OWL. Translating PCT into OWL lets us use sensor data in standard Semantic Sensor Web format by adapting OWL reasoning to perform the needed abductive inference.

Researchers have explored integrating OWL with abductive reasoning [Elsenbroich06]. However, this integration would require modifying OWL syntax and/or modifying an OWL inference engine. Here, we demonstrate that OWL provides some of the expressivity needed to derive explanations — without extending its syntax or semantics — by outlining a suitable encoding of PCT in OWL [Henson11a]. Note, however, that the OWL representation discussed only approximates PCT, because OWL inference doesn't support a hypothesize-and-test inference process.

The task of representing PCT in OWL involves encoding the background knowledge Σ and the set of observations Γ in an OWL ontology such that an OWL reasoner can compute explanations Δ that satisfy both the coverage and parsimony criteria. This translation is summarized in Table 3.11.

Table 3.10. Translating PCT to OWL.

	PCT	OWL
1	E	for all $e \in E$ assert *Entity(e)*
2	Q	for all $q \in Q$ assert *Quality(q)*
3	C	for all $q \in C(e)$ assert *causes(e,q)*
4	Γ	$\mathit{Explanation} \equiv \exists \mathit{causes}.\{q_1\} \sqcap \ldots \sqcap \exists \mathit{causes}.\{q_n\}$, where $q_i \in \Gamma$
5	Δ	for each $\Delta = \{e\}$, *Explanation(e)* holds

To translate the set of entities E, we create a class *Entity*, and for all $e \in E$, we create an individual instance of type *Entity* by asserting *Entity*(e). To translate the set of qualities Q, we create a class *Quality*, and for all $q \in Q$, we create an individual instance of type *Quality* by asserting *Quality*(q). Finally, to translate the set of *causes* relation instances C, we create an object property *causes*; and, for all entities in the domain of C and for each $q \in C(e)$, we create a *causes* fact by asserting *causes*(e, q).

To translate the set of observations Γ into OWL, we first select an observation $q_1 \in \Gamma$ and create an existentially quantified property restriction for the *causes* relation, $\exists causes.\{q_1\}$. For each additional observation $q_i \in \Gamma$ ($i = 2, ..., n$), we create an additional existentially quantified property restriction for the *causes* relation and conjoin it to the previous restriction: $\exists causes.\{q_1\} \sqcap \ldots \sqcap \exists causes.\{q_n\}$. Finally, we create a class *Explanation* and define it to be equivalent to the conjunction of restrictions, $Explanation \equiv \exists causes.\{q_1\} \sqcap \ldots \sqcap \exists causes.\{q_n\}$. To generate explanations Δ, we execute a query for all individual instances of type *Explanation* as *Explanation*(*?x*). *Explanation*(e) is a result of this query if and only if $\{e\}$ is a parsimonious cover. The resulting knowledge base lies in the tractable EL profile of OWL 2.

Theorem. Given a PCT problem P = $\langle E, Q, C, \Gamma\rangle$ and its translation $o(P)$ into OWL, $\Delta = \{e\}$ is a PCT explanation if and only if *Explanation*(e) is deduced by an OWL-DL reasoner — that is, if and only if $o(P) \vDash Explanation(e)$.

***Proof.* (⇒)** If $\{e\}$ is a parsimonious cover of $\Gamma = \{q_1, \ldots, q_n\}$, then, by definition, $\Gamma \subseteq C(e)$. By construction of *causes* in $o(P)$, $f : \exists causes.\{q_1\} \sqcap \ldots \sqcap \exists causes.\{q_n\}$. Hence, by definition of *Explanation*, $o(P) \vDash Explanation(e)$ holds.

(⇐) To justify our claim that this OWL representation approximates PCT, we verify that all query results satisfy both the coverage and parsimony criteria. To satisfy the coverage criterion, each binding of *?x* for the query *Explanation*(*?x*) must be an entity that explains all the observations in Γ. This follows from the definition of *Explanation* $\equiv \exists causes.\{q_1\} \sqcap \ldots \sqcap \exists causes.\{q_n\}$. That is, *Explanation*(*e*) implies $causes(e, q_1) \sqcap \ldots \sqcap causes(e, q_n)$. To satisfy the parsimony criterion, each binding of *?x* must be a single entity. This follows since each entity that binds to ?x is a single individual. This completes the proof.

If we want to generalize the definition of *Explanation* to allow for covers with multiple disorders, then the parsimony criterion cannot easily be expressed in OWL, since it would require minimization of the extension of a predicate. Simulation by using multiple queries may be an option, by incrementally generating cover candidates and checking whether each constitutes an explanation. This seems hardly efficient, though, and also unsatisfactory because the parsimony criterion itself is not modeled.[10]

3.3.2. Discussion of Terminology

In order to encode important fragments of the ontology of perception, Intellego, into OWL (Intellego—OWL), as an extension of the SSN ontology, several terminological adjustments will be made. Table 3.10 provides a quick summary of equivalent terms from Intellego—Set Theory, PCT, SSN, and Intellego—OWL.

[10] In a circumscriptive version of OWL [Grimm09][Bonatti09] it could easily be modeled.

Table 3.11. Quick summary of equivalent terms used within the different frameworks/formalizations discussed in this chapter. Ontologies formalized in OWL use namespace prefixes (SSN uses ssn, and the OWL encoding of Intellego uses io).

IntellegO – Set Theory	PCT	SSN	IntellegO – OWL
entity	Entity (E)	ssn:Feature	ssn:Feature
quality	Quality (Q)	ssn:Property	ssn:Property
inheres-in	causes (C) (**inverse of inheres-in*)	ssn:isPropertyOf	ssn:isPropertyOf
percept	Observation (Γ)	ssn:Observation ssn:observedProperty	io:ObservedProperty
perceptual-theory	Explanation (Δ)		io:ExplanatoryFeature
expected			io:ExpectedProperty
unknown			io:NotApplicableProperty
extraneous			
discriminating			io:DiscriminatingProperty
focus-candidate			

Throughout the remainder of Section 3.3, the Intellego—OWL terminology will be used. Also, examples used throughout this section will utilize the knowledge base represented in Figure 3.13.

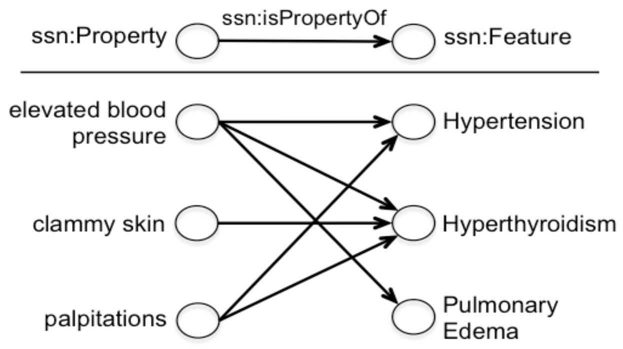


Figure 3.13. Graphical representation of an example knowledgebase in cardiology, taking the shape of a bipartite graph.

3.3.3. Semantics of Explanation in OWL

In this section, explanation is encoded in OWL, as an extension of the SSN ontology. *Explanation* is the act of accounting for sensory observations; often referred to as hypothesis building [Gregory97][Shanahan05]. More specifically, explanation takes a set of observed properties as input and yields the set of features that explain the observed properties. A feature is said to *explain* an observed property if the property is related to the feature through an *ssn:isPropertyOf* relation. A feature is said to explain a set of observed properties if the feature explains each property in the set. *Example*: Given the knowledge base in Figure 3.13, Hyperthyroidism explains the observed properties elevated blood pressure, clammy skin, and palpitations.

Explanation is used to derive knowledge of the features in an environment from observation of their properties. Since several features may be capable of explaining a given set of observed properties, explanation is most accurately defined as an abductive process (i.e., *inference to the best explanation*) [Shanahan05]. *Example*: the observed properties, elevated blood pressure and palpitations, are explained by the features Hypertension and Hyperthyroidism (discussed further below). As discussed above, while OWL has not been specifically designed for abductive inference, it does provide some of the expressivity needed to derive explanations.

The formalization of explanation in OWL consists of two steps: (1) derive the set of observed properties from a set of observations, and (2) utilize the set of observed properties to derive a set of explanatory features.

3.3.3.1. Observed Property

The SSN ontology defines an observation as a situation that describes an observed feature, an observed property, a sensor, the method of sensing used, and a value for the observed property. Currently, however, an SSN observation can't directly yield explanations. To accomplish this, we must translate the SSN observations to OWL as discussed in Section 3.3.1.2.

An *observed property* is a property that has been observed. Note that observations of a property, such as elevated blood pressure, also contain information about the spatiotemporal context, measured value, unit of measure, etc., so the observed properties need to be "extracted" from the observations. To derive the set of observed properties (instances), first create a class ObservedProperty. For each observation o in ssn:Observation create an existentially quantified property restriction for the ssn:observedProperty^{-} relation, and disjoin them as follows (note: x^{-} represents the inverse of relation x):

Definition 1: ObservedProperty

$\texttt{ObservedProperty} \equiv \exists\texttt{ssn:observedProperty}^{-}.\{o_1\} \sqcup \ldots \sqcup \exists\texttt{ssn:observedProperty}^{-}.\{o_n\}$

Example: Assume the properties elevated blood pressure and palpitations have been observed, and encoded in RDF (conformant with SSN):

```
ssn:Observation(o1),
ssn:observedProperty(o1, elevated blood pressure)
ssn:Observation(o2),
ssn:observedProperty(o2, palpitations)
```

Given these observations, the following ObservedProperty class is constructed:

```
ObservedProperty ≡ ∃ssn:observedProperty¯.{elevated blood
pressure} ⊔ ∃ssn:observedProperty¯.{palpitations}
```

Executing the query ObservedProperty(?x) can infer the properties, elevated blood pressure and palpitations, as individuals of type ObservedProperty:

```
ObservedProperty(elevated blood pressure)
ObservedProperty(palpitations)
```

3.3.2.2. Explanatory Feature

An *explanatory feature* is a feature that explains the set of observed properties. To derive the set of explanatory features, create a class ExplantoryFeature, and for each observed property p in ObservedProperty create an existentially quantified property restriction for the ssn:isPropertyOf¯ relation, and conjoin them as follows:

<u>Definition 2: ExplanatoryFeature</u>

```
ExplanatoryFeature ≡ ∃ssn:isPropertyOf¯.{p₁} ⊓ … ⊓
∃ssn:isPropertyOf¯.{pₙ}
```

Recall (from Section 3.3.1) that the set of observations $\Gamma = \{q_1 \ldots q_n\}$, when translated to OWL, are encoded as *Explanation* ≡ ∃*causes*.$\{q_1\}$ ⊓ … ⊓ ∃*causes*.$\{q_n\}$. When extending the SSN ontology with explanation, the ssn:isPropertyOf¯ relation is interpreted to be equivalent to causes in PCT.

To derive the set of all explanatory features, construct the ObservedProperty class and execute the query ObservedProperty(?x) with an OWL reasoner. Then, construct the ExplanatoryFeature class and execute the query ExplanatoryFeature(?y).

Example: As above, assume the properties elevated blood pressure and palpitations have been observed:

```
ObservedProperty(elevated blood pressure)
ObservedProperty(palpitations)
```

Given these observations, the following ExplanatoryFeature class is constructed:

```
ExplanatoryFeature ≡  ∃ssn:isPropertyOf¯.{elevated blood
pressure} ⊓ ∃ssn:isPropertyOf¯.{palpitations}
```

Given the knowledge base in Figure 3.13, executing the query ExplanatoryFeature(?y) can infer the features, Hypertension and Hyperthyroidism, as explanations:

```
ExplanatoryFeature(Hypertension)
ExplanatoryFeature(Hyperthyroidism)
```

This encoding of explanation in OWL (see DEF 2) provides an accurate simulation of abductive reasoning in the Parsimonious Covering Theory [Reggia87], *with the single-entity* (as discussed in Section 3.3.1.1) [Henson11][Henson12]. The Description Logic expressivity of the explanation task is ALCOI[11,12], with ExpTime-complete complexity [Tobies01].

3.3.4. Semantics of Discrimination in OWL

Although the result give above is valid (that is, both Hypertension and Hyperthyroidism are explanations – i.e., parsimonious covers), we might not be satisfied and might want to distinguish, or discriminate, between these two explanations. In this section discrimination (or focus) is encoded in OWL, as an extension of the SSN ontology. As discussed previously, *Discrimination* is the act of deciding how to narrow down the multitude of explanatory features through further observation. The innate human ability to focus attention on aspects of the environment that are essential for effective situation-awareness stems from the act of discrimination [Neisser76][Gregory97][Bajcsy88]. Discrimination takes a set of features as input and yields a set of properties. A property is said to *discriminate* between a set of features if its presence can reduce the set of explanatory features. *Example*: Given the knowledge base in

[11] Using DL constructs: $\sqcap$, $\sqcup$,$\exists$, {a}, R^{-}

[12] http://www.cs.man.ac.uk/~ezolin/dl/

Figure 3.13, the property clammy skin discriminates between the features, Hypertension and Hyperthyroidism (discussed further below).

The ability to identify discriminating properties can significantly improve the efficiency of machine perception [Henson11c]. Such knowledge can then be used to task sensors capable of observing those properties.

To formalize discrimination in OWL, we will define three types of properties: *expected property*, *not-applicable property*, and *discriminating property*.

3.3.4.1. Expected Property

A property is *expected* with respect to (w.r.t.) a set of features if it is a property of every feature in the set. Thus, if it were to be observed, every feature in the set would explain the observed property. *Example*: the property elevated blood pressure is expected w.r.t. the features, Hypertension, Hyperthyroidism, and Pulmonary Edema. To derive the set of expected properties, create a class ExpectedProperty, and for each explanatory feature f in ExplanatoryFeature, create an existentially quantified property restriction for the ssn:isPropertyOf relation, and conjoin them as follows:

Definition 3: ExpectedProperty

$\text{ExpectedProperty} \equiv \exists\text{ssn:isPropertyOf}.\{f_1\} \sqcap \ldots \sqcap$

$\exists\text{ssn:isPropertyOf}.\{f_n\}$

3.3.4.2. Not Applicable Property

A property is *not-applicable* w.r.t. a set of features if it is not a property of any feature in the set. Thus, if it were to be observed, no feature in the set would explain the observed property. *Example*: the property clammy skin is not-applicable w.r.t. the features, Hypertension and Pulmonary Edema. To derive the set of not-applicable properties, create a class NotApplicableProperty, and for each explanatory feature f in ExplanatoryFeature, create a negated existentially quantified property restriction for the ssn:isPropertyOf relation, and conjoin them as follows:

<u>Definition 4: NotApplicableProperty</u>

$$\text{NotApplicableProperty} \equiv \neg\exists \text{ssn:isPropertyOf}.\{f_1\} \sqcap \ldots \sqcap \neg\exists \text{ssn:isPropertyOf}.\{f_n\}$$

3.3.4.3. Discriminating Property

A property is *discriminating* w.r.t. a set of features if it is neither expected nor not-applicable. Observing a discriminating property would help to reduce the number of explanatory features. *Example*: As stated above, the property clammy skin is discriminating w.r.t. the features, Hypertension and Hyperthyroidism, as it would be explained by Hyperthyroidism, but not by Hypertension. To derive the set of discriminating properties, create a class, DiscriminatingProperty, which is equivalent to the conjunction of the negated ExpectedProperty class and the negated NotApplicableProperty class.

<u>Definition 5: DiscriminatingProperty</u>

`DiscriminatingProperty ≡ ¬ExpectedProperty ⊓ ¬NotApplicableProperty`

To derive the set of all discriminating properties, construct the ExpectedProperty and NotApplicableProperty classes, and execute the query DiscriminatingProperty(?x).

Example: Given the explanatory features from the previous example, Hypertension and Hyperthyroidism, the following classes are constructed:

`ExpectedProperty ≡ ∃ssn:isPropertyOf.{Hypertension} ⊓ ∃ssn:isPropertyOf.{Hyperthyroidism}`

`NotApplicableProperty ≡ ¬∃ssn:isPropertyOf.{Hypertension} ⊓ ¬∃ssn:isPropertyOf.{Hyperthyroidism}`

Given the KB in Figure 3-13, executing the query DiscriminatingProperty(?x) can infer the property clammy skin as discriminating:

`DiscriminatingProperty(clammy skin)`

To choose between Hypertension and Hyperthyroidism, task a sensor to measure galvanic skin response (i.e., for clammy skin). The Description Logic expressivity of the discrimination task is ALCO[13], with PSpace-complete complexity [Tobies01].

[13] using DL constructs: ⊓, ∃, {a}, ¬C

3.4. Related Work

In addition to the SSN ontology, there have been several other attempts to develop an ontology for sensors and sensor observations [Compton09]. Kuhn [Kuhn09] and Stasch [Stasch09] have developed an ontology for describing sensors and sensor observations in Haskell. Similar to IntellegO, this ontology attempts to represent the process of observation in a way that is independent of any particular implementation technology (e.g., machine sensor, human eye). The Perception, Cognition and Communication (PCC) Ontology defines a set of classes and relations that provide terminology to describe "what we believe exists, what we experience, what we think and what we communicate [Anandavala10]." PCC shares many concepts with the ontology of perception described in this paper. However, we have been unable to find any use-case, application, or evaluation for this ontology. Even the Gene Ontology (GO) contains representation for sensory perception [SensoryPerception10]. However, perception in GO is defined as a neurophysiological process and is similar to what we describe as the observation-process. Devaraju et al. [Devaraju10] have developed an approach for representing the relationship between observed qualities and the geo-processes that influence those observations. This approach is aligned with the DOLCE foundational ontology [Borgo09] and has been used to represent relations between qualities and entities in the domain of hydrology. This ontology has been used primarily for the integration of sensor data, and there is no attempt to show how to perform inference over observed qualities. Scheider et al. [Scheider10] have developed a general theory for grounding entities and qualities to observation processes, based on ideas from language semantics.

In addition to the development of related ontologies, there have been several efforts to reason over observational data, encoded as RDF, in order to infer entities in the world. In 2008, Sheth

[Sheth08] suggested using the Semantic Web Rule Language (SWRL) to reason over sensor data. Since this time, First-Order Logic (FOL) rules are often employed to derive knowledge of the features in the environment [Keßler09][Scheider10][Devaraju11]. Keßler [Keßler09] has further developed this idea and provides a suitable investigation of the use of SWRL for reasoning over sensor data. Ricquebourg [Ricquebourg07] has utilized SWRL to infer context and determine proper actions (i.e., turn on/off lights) within a smart home environment which contains sensors. Calder [Calder10] has used the Jena Semantic Web Toolkit [Carroll04] to define rules to detect anomalous events (such as *winter weather* and *algal blooms*). Taylor et al. [Taylor11] have used Complex Event Processing to derive knowledge of events from observation data encoded in SSN. These approaches all define first-order logic (deductive) rules to infer entities from qualities. As discussed in this paper, however, such an approach is limited in its ability to represent the anti-monotonic, abductive nature of perception, to handle incomplete information, and to minimize explanations based on new information. In particular, such approaches cannot model perception as a cyclical process that actively seeks out and detects those qualities which carry information most useful for explanation. In addition, as we have shown, several inference tasks useful for machine perception do not require the full expressivity of FOL; they are expressible in OWL, a decidable fragment of FOL.

Reggia and Peng [Reggia86] have discussed techniques for abductive reasoning (called Parsimonious Covering Theory) that is similar to the approach taken in this paper; however, they were mainly interested in inferring medical diseases from symptoms. Perhaps the closest related work is by Shanahan [Shanahan05], who formalized an abductive account of perception using Event Calculus. More specifically, he characterized perception as follows:

> "*Given a stream of low-level sensor data, represented by the conjunction Γ of a set of observation sentences, the task of perception is to find one or more explanations of Γ in*

the form of a consistent logical description Δ of hypothesized objects, such that, $\Sigma \cap \Delta \Rightarrow \Gamma$, where Σ is a background theory describing the environment [Shanahan05]."

Thirunarayan [Thirunarayan09] explored how a logic programming-based abductive reasoning framework can benefit the formalization and interpretation of sensor data to garner situation awareness. Although there are several related efforts, the research discussed in this paper is first of its kind to present and evaluate an approach that is grounded in well-established cognitive theories of perception and also applicable for sensor applications on the Web.

The integration of Web languages, such as OWL, with abductive reasoning has been explored [Elsenbroich06]. However, previous efforts have required modification of OWL syntax and/or inference engine [Peraldi09]. We demonstrated that, under the single-feature assumption, abductive consequences can be computed using standard OWL reasoners.

3.5. Concluding Remarks

Sensors are quickly becoming ubiquitous and are now collecting data about our environment at an extraordinary pace. In this paper, we have demonstrated substantial benefits to be gained in processing sensor data by automating an approximation of how people perceive their environment efficiently. Specifically, this ability is afforded by background knowledge and the cyclical nature of observation and perception processes. While one can take different positions on the philosophical (or technical) foundations of perception, it is clear that a careful ontological specification makes these positions explicit and testable. The ontology described in this chapter establishes a formal semantics for machine perception; and provides a solution to difficult challenges, such as the ability to effectively model the process of perception, to provide an appropriate interpretation of observational data with incomplete information, and to efficiently interpret and store the growing stream of observational data. This approach has been evaluated on a large, open, real-world dataset of weather observations. Three evaluations were provided to demonstrate (1) how focus can lead to improved efficiency in generating and processing sensor observations, (2) how the expressivity of Intellego compares to existing solutions, and (3) how perception can lead to significant storage savings. In the future, such results can be exploited to enable significant savings of energy and computational resources. If current trends in sensor capabilities continue, the predicted data requirements will be in excess of a yottabyte (1024 Bytes) by 2015 [Mitre08]. Besides the obvious issues of data management, knowing how to effectively make sense of sensor data – that will largely be flowing through the Web – represents a significant challenge. This research is an early and modest step in understanding and addressing this challenge.

The next chapter will discuss not only how to make sense of sensor data, but also how to do so efficiently on resource-constrained devices, such as mobile phones.

4. Intelligence at the Edge

In recent years, we have seen dramatic advances and adoption of sensor technologies to monitor all aspects of our environment; and increasingly, these sensors are embedded within mobile devices. There are currently over 4 billion mobile devices in operation around the world; and an estimated 25% (and growing) of those are *smart* devices[1]. Many of these devices are equipped with sensors, such as cameras, GPS, RFID, and accelerometers. Other types of external sensors are also directly accessible to mobile devices through either physical attachments or wireless communication protocols, such as Bluetooth. Mobile applications that may utilize this sensor data for deriving context and/or situation awareness abound. Consider a mobile device that's capable of communicating with on-body sensors measuring body temperature, heart rate, blood pressure, and galvanic-skin response. The data generated by these sensors may be analyzed to determine a person's health condition and recommend subsequent action. The value of such applications such as these is obvious, yet difficult challenges remain.

The act of observation performed by heterogeneous sensors creates an avalanche of data that must be integrated and interpreted in order to provide knowledge of the situation. This process is commonly referred to as perception, and while people have evolved sophisticated mechanisms to efficiently perceive their environment – such as the use of a-priori knowledge of the environment [Neisser76][Gregory97] – machines continue to struggle with the task. *The primary challenge of machine perception is to define efficient computational methods to derive high-level knowledge*

[1] http://www.digitalbuzzblog.com/2011-mobile-statistics-stats-facts-marketing-infographic/

from low-level sensor observation data. From the scenario above, the high-level knowledge of a person's health condition is derived from low-level observation data from on-body sensors.

Given the ubiquity of mobile devices and the proliferation of sensors capable of communicating with them, mobile devices serve as an appropriate platform for executing machine perception. Despite the popularity of cloud-based solutions, many applications may still require local processing, e.g., for privacy concerns, or the need for independence from network connectivity in critical healthcare applications. *The computational complexity of OWL, however, seriously limits its applicability and use within resource-constrained environments, such as mobile devices* [Ali09].

To overcome this issue, we develop encodings and algorithms for the efficient execution of the inference tasks needed for machine perception: explanation and discrimination. *Explanation* is the task of accounting for sensory observations; often referred to as hypothesis building [Gregory97][Shanahan05]. *Discrimination* is the task of deciding how to narrow down the multitude of explanations through further observation [Neisser76][Gregory97]. The efficient algorithms devised for explanation and discrimination use bit vector operations, leveraging environmental knowledge encoded within a two-dimensional bit matrix.

To preserve the ability to share and integrate with knowledge on the Web, lifting and lowering mappings between the semantic representations and the bit vector representations are provided. Using these mappings, knowledge of the environment encoded in RDF (and shared on the Web, i.e., as Linked Data) may be utilized by *lowering* the knowledge to a bit matrix representation. On the other hand, knowledge derived by the bit vector algorithms may be shared on the Web (i.e., as Linked Data), by *lifting* to an RDF representation.

In this chapter, two novel contributions towards efficient machine perception in resource-constrained environments are presented:

1. Lifting and lowering mappings to enable the translation of knowledge between the high-level semantic representations and low-level bit-vector representations, and
2. Efficient algorithms for these inference tasks, using bit vector operations.

The applicability of this approach to machine perception is evaluated on a smart-phone mobile device, demonstrating dramatic improvements in both efficiency and scale. Note that this work does not support reasoning for all of OWL, but supports what is needed for machine perception, which is useful in a variety of applications. Table 4.1 summarizes the data structures used by our algorithms.

Table 4.1. Quick summary of data structures used by the bit vector algorithms (note: |x| represents the number of members of x).

Name	Description	About (type, size)
KB_{BM}	Environmental knowledge	Bit matrix of size |ssn:Property| x |ssn:Feature|
$OBSV_{BV}$	Observed properties	Bit vector of size |ssn:Property|
$EXPL_{BV}$	Explanatory features	Bit vector of size |ssn:Feature|
$DISC_{BV}$	Discriminating properties	Bit vector of size |ssn:Property|

The lifting and lowering mappings are provided in Section 4.1. The efficient bit vector algorithms for explanation and discrimination are discussed in Sections 4.2 and 4.2, respectively. This approach is evaluated in Section 4.3, followed by related work in Section 4.4.

4.1. Lifting and Lowering of Semantic Data

To preserve the ability to share and integrate with knowledge on the Web, lifting and lowering mappings between the semantic representations and bit vector representations are provided. Using these mappings, knowledge of the environment encoded in RDF, as well as observed properties encoded in RDF, may be utilized by *lowering* them to a bit vector representation. Knowledge derived by the bit vector algorithms, including observed properties, explanatory features, and discriminating properties, may be shared on the Web, by *lifting* them to an RDF representation.

Environmental knowledge: An environmental knowledgebase is represented as a bit matrix KB_{BM}, with rows representing properties and columns representing features. $KB_{BM}[i][j]$ is set to 1 (true) iff the property p_i is a property of feature f_j. To *lower* an SSN KB encoded in RDF: for all properties p_i in ssn:Property, create a corresponding row in KB_{BM}, and for all features f_j in ssn:Feature, create a corresponding column. Set $KB_{BM}[i][j]$ to 1 iff there exists a ssn:isPropertyOf(p_i,f_j) relation. Figure 4.1(a) shows an example knowledge base, from Figure 3.13, which has been automatically lowered to a bit matrix representation. Index tables are also created to map between the URI's for concepts in the semantic representation to their corresponding index positions in the bit vector representation. Figures 4.1(b) and 4.1(c) show example index tables for properties and features.

	Hypertension	Hyperthyroidism	Pulmonary Edema
elevated blood pressure	1	1	1
clammy skin	1	0	0
palpitations	1	1	0

(a)

index position	URI
0	http://example.com/cardio.owl#elevated-blood-pressure
1	http://example.com/cardio.owl#clammy-skin
2	http://example.com/cardio.owl#palpitations

(b)

index position	URI
0	http://example.com/cardio.owl#Hypertension
1	http://example.com/cardio.owl#Hyperthyroidism
2	http://example.com/cardio.owl#Pulmonary-Edema

(c)

Figure 4.1. (a) Example environmental knowledgebase in the domain of cardiology, from Figure 3.13, represented as a bit matrix. Index tables are used for lifting and lowering environmental knowledge between a semantic representation and a bit vector representation. (b) Index table for properties. (c) Index table for features.

Observed properties: Observed properties are represented as a bit vector $OBSV_{BV}$, where $OBSV_{BV}[i]$ is set to 1 iff property p_i has been observed. To *lower* observed properties encoded in RDF: for each property p_i in ssn:Property, $OBSV_{BV}[i]$ is set to 1 iff ObservedProperty(p_i). To *lift* observed properties encoded in $OBSV_{BV}$: for each index position i in $OBSV_{BV}$, assert ObservedProperty(p_i) iff $OBSV_{BV}[i]$ is set to 1. To generate a corresponding observation o, create an individual o of type ssn:Observation, ssn:Observation(o), and assert ssn:observedProperty(o,p_i).

Explanatory features: Explanatory features are represented as a bit vector $EXPL_{BV}$. $EXPL_{BV}[j]$ is set to 1 iff the feature f_j explains the set of observed properties represented in $OBSV_{BV}$ (that is, it explains all properties in $OBSV_{BV}$ that are set to 1). To *lift* explanatory features encoded in

EXPL_{BV}: for each index position j in EXPL_{BV}, assert ExplanatoryFeature(f_j) iff EXPL_{BV}[j] is set to 1.

Discriminating properties: Discriminating properties are represented as a bit vector DISC_{BV} where DISC_{BV}[i] is set to 1 iff the property p_i discriminates between the set of explanatory features represented in EXPL_{BV}. To *lift* discriminating properties encoded in DISC_{BV}: for each index position i in DISC_{BV}, assert DiscriminatingProperty(p_i) iff DISC_{BV}[i] is set to 1.

4.2. Efficient Bit Vector Algorithm for Explanation

The strategy employed for efficient implementation of the explanation task relies on the use of the bit vector AND operation to discover and *dismiss* those features that cannot explain the set of observed properties (see Algorithm 1). It begins by considering all the features as potentially explanatory, and iteratively dismisses those features that cannot explain an observed property, eventually converging to the set of all explanatory features that can account for all the observed properties. Note that the input OBSV_{BV} can be set either directly by the system collecting the sensor data or by translating observed properties encoded in RDF.

Algorithm 1: Explanation

```
input $\text{OBSV}_{\text{BV}}$
define BitVector $\text{EXPL}_{\text{BV}}$
for each j := 0 … |ssn:Feature|-1
    $\text{EXPL}_{\text{BV}}$[j] := 1
for each i := 0 … |ssn:Property|-1
    if $\text{OBSV}_{\text{BV}}$[i] = 1 then
        $\text{EXPL}_{\text{BV}}$ := $\text{EXPL}_{\text{BV}}$ AND (row i in $\text{KB}_{\text{BM}}$)
output $\text{EXPL}_{\text{BV}}$
```

We will now sketch the correctness of the explanation algorithm w.r.t. the OWL specification. For each index position in $EXPL_{BV}$ that is set to 1, the corresponding feature explains all the observed properties. (*See note about indices*[2]).

Theorem 1: Given an environmental knowledgebase KB (i.e., KB_{BM}), the following two statements are equivalent:

***S1*:** The set of *m* observed properties $\{p_{k1}, \ldots, p_{km}\}$, i.e., ObservedProperty(p_{k1}) $\sqcap \ldots \sqcap$ ObservedProperty(p_{km}), is explained by the feature f_e, implies ExplanatoryFeature(f_e).

***S2*:** The Hoare triple[3] holds: $\{ \forall i \in \{1, \ldots, m\}: OBSV_{BV}[ki] = 1 \}$

Algorithm 1: Explanation

$\{ EXPL_{BV}[e] = 1 \}$.

Proof (*S1* ⇒ *S2*): The ObservedProperty assertions are captured by the proper initialization of $OBSV_{BV}$, as stated in the precondition. Given (i) *S1*, (ii) the single-feature assumption, (iii) the definition: ExplanatoryFeature $\equiv \exists$ssn:isPropertyOf^{-}.$\{p_{k1}\} \sqcap \ldots \sqcap \exists$ssn:isPropertyOf^{-}.$\{p_{km}\}$, and (iv) the fact that ExplanatoryFeature(f_e) is provable, it follows that $\forall i \in \{1, \ldots, m\}$: ssn:isPropertyOf($p_{ki}$,$f_e$) is in KB. By our encoding, $\forall i \in \{1, \ldots, m\}$: $KB_{BM}[ki][e] = 1$. Using lines 5-7, the fact that $EXPL_{BV}[e]$ is initialized to 1 and is updated only for $i \in \{1, \ldots, m\}$ where $OBSV_{BV}[ki] = 1$, we get the final value of $EXPL_{BV}[e] = KB_{BM}[k1][e]$ AND … AND $KB_{BM}[km][e] = 1$ (true).

[2] Note that property p_{ki} has property index ki and feature f_{ej} has feature index ej. So ki ranges over 0 to |ssn:Property|-1 and e/ej range over 0 to |ssn:Feature|-1. i and j are merely indices into the enumeration of observed properties and their explanatory features, respectively. Thus, i ranges over 1 to |ssn:Property| and j ranges over 1 to |ssn:Feature|. (In practice, initially i is small and j is large, and through each cycle of explanation and discrimination, i increases while j diminishes.)

[3] {P} S {Q} where P is the pre-condition, S is the program, and Q is the post-condition.

(*S2* ⇒ *S1*): Given that $\{\forall i \in \{1, \ldots, m\}: \text{OBSV}_{BV}[ki] = 1\}$ and $\{\text{EXPL}_{BV}[e] = 1\}$ (pre and post conditions), it follows that $\forall i \in \{1, \ldots, m\}: \text{KB}_{BM}[ki][e] = 1$ must hold. According to our encoding, this requires that $\forall i \in \{1, \ldots, m\}$: ssn:isPropertyOf($p_{ki}$,e) holds. Using the definition of ExplanatoryFeature, it follows that ExplanatoryFeature(e) is derivable (that is, f_e explains *all* the observed properties $\{p_{k1}, \ldots, p_{km}\}$).

Theorem 2: The explanation algorithm (Algorithm 1) computes all and only those features that can explain all the observed properties.

Proof: The result follows by applying Theorem 1 to all explanatory features. *Q.E.D.*

4.3. Efficient Bit Vector Algorithm for Discrimination

The strategy employed for efficient implementation of the discrimination task relies on the use of the bit vector AND operation to discover and indirectly *assemble* those properties that discriminate between a set of explanatory features (see Algorithm 2). The discriminating properties are those that are determined to be neither expected nor not-applicable.

Algorithm 2: Discrimination

```
input EXPL$_{BV}$, OBSV$_{BV}$
define BitVector DISC$_{BV}$
for each i := 0 … |ssn:Property|-1
   DISC$_{BV}$[i] := 0
define BitVector ZERO$_{BV}$
for each j := 0 … |ssn:Feature|-1
   ZERO$_{BV}$[j] := 0
for each i := 0 … |OBSV$_{BV}$|-1
   if OBSV$_{BV}$[i] = 0 then
      BitVector PEXPL$_{BV}$ :=
         EXPL$_{BV}$ AND (row i in KB$_{BM}$)
      if PEXPL$_{BV}$ != EXPL$_{BV}$ and
         PEXPL$_{BV}$ != ZERO$_{BV}$ then
         DISC$_{BV}$[i] := 1
output DISC$_{BV}$
```

In the discrimination algorithm, both the discriminating properties bit vector $DISC_{BV}$ and the zero bit vector $ZERO_{BV}$, are initialized to zero. For a not-yet-observed property at index ki, the bit vector $PEXPL_{BV}$ can represent one of three situations: (i) $PEXPL_{BV} = EXPL_{BV}$ holds and the kith property is expected; (ii) $PEXPL_{BV} = ZERO_{BV}$ holds and the kith property is not-applicable; or (iii) the kith property discriminates between the explanatory features (and partitions the set). Eventually, $DISC_{BV}$ represents all those properties that are *each* capable of partitioning the set of explanatory features in $EXPL_{BV}$. Thus, observing any one of these will narrow down the set of explanatory features.

We will now sketch the correctness of the discrimination algorithm w.r.t. the OWL specification. Each explanatory feature explains all the observed properties. Lemma 1 shows that this is equivalent to all the observed properties being expected properties of the explanatory features.

Lemma 1: If *m* observed properties $\{p_{k1}, \ldots, p_{km}\}$, i.e., $\text{ObservedProperty}(p_{k1}) \sqcap \ldots \sqcap \text{ObservedProperty}(p_{km})$, are explained by *n* features $\{f_{e1,} \ldots, f_{en}\}$, i.e., $\text{ExplanatoryFeature}(f_{e1}) \sqcap \ldots \sqcap \text{ExplanatoryFeature}(f_{en})$, then the following holds: $\forall i: 1 \leq i \leq m: \text{ObservedProperty}(p_{ki}) \Rightarrow \text{ExpectedProperty}(p_{ki})$.

Proof Sketch: The result is obvious from the definition: $\text{ExplanatoryFeature} \equiv \exists \text{ssn:isPropertyOf}^{-}.\{p_{k1}\} \sqcap \ldots \sqcap \exists \text{ssn:isPropertyOf}^{-}.\{p_{km}\}$. So, $\forall i, \forall j: 1 \leq i \leq m \wedge 1 \leq j \leq n$: $\text{ssn:isPropertyOf}(p_{ki},f_{ej})$. $\text{ExpectedProperty} \equiv \exists \text{ssn:isPropertyOf}.\{f_{e1}\} \sqcap \ldots \sqcap \exists \text{ssn:isPropertyOf}.\{f_{en}\}$.

Lemma 2 restates the assertion (from Lemma 1) that observed properties are also expected properties of explanatory features, in terms of the bit vector encoding.

Lemma 2: The initial values of $EXPL_{BV}$ and $OBSV_{BV}$ satisfy the assertion: $\forall ki: (OBSV_{BV}[ki] = 1) \Rightarrow [\forall e: (EXPL_{BV}[e] = 1) \Rightarrow (KB_{BM}[ki][e]) = 1)]$. And hence, $\forall i: (OBSV_{BV}[ki] = 1) \Rightarrow [\forall e: (EXPL_{BV}[e] \wedge KB_{BM}[ki][e]) = EXPL_{BV}[e])]$.

Proof Sketch: The claim follows from Lemma 1 and the bit vector encoding.

Lemma 3 generalizes Lemma 2 to elucidate an efficient means to determine when a not-yet-observed property is expected, w.r.t. a set of explanatory features.

Lemma 3: Given property ki (p_{ki}) has not-yet been observed, i.e., $OBSV_{BV}[ki] = 0$, ExpectedProperty(p_{ki}) iff $\forall e: (EXPL_{BV}[e] \wedge KB_{BM}[ki][e]) = EXPL_{BV}[e]$.

Lemma 4 demonstrates an efficient means to determine when a not-yet-observed property is not-applicable, w.r.t. a set of explanatory features.

Lemma 4: For explanatory features $EXPL_{BV}$ $\{f_e \mid EXPL_{BV}[e] = 1\}$**,** NotApplicableProperty(p_{ki}) iff $\forall e: (EXPL_{BV}[e] \wedge KB_{BM}[ki][e]) = ZERO_{BV}[e]$.

Proof Sketch: The result follows from: (i) the definition of NotApplicableProperty w.r.t. the set of explanatory features: NotApplicableProperty(p_{ki}) iff $\forall ki, \forall e$: ExplanatoryFeature(f_e) $\Rightarrow$ $\neg\exists$ssn:isPropertyOf(p_{ki},f_e); (ii) [$\forall e$: ExplanatoryFeature(f_e) iff $EXPL_{BV}[e] = 1$]; and (iii) $\forall ki, \forall e$: [$\neg\exists$ssn:isPropertyOf(p_{ki},f_e) $\Rightarrow KB_{BM}[ki][e] = 0$].

Theorem 3: The discrimination algorithm (Algorithm 2) computes all and only those properties that can discriminate between the explanatory features.

Proof: A not-yet-observed property is *discriminating* if it is neither expected nor not-applicable. The result follows from the definition of discriminating property, Lemma 3, and Lemma 4. *Q.E.D.*

4.4. Evaluation of Bit Vector Algorithms

To evaluate our approach, we compare two implementations of the explanation and discrimination inference tasks. The first utilizes an OWL reasoner, and the second utilizes the bit vector algorithms. Both implementations are coded in Java, compiled to a Dalvik[4] executable, and run on a Dalvik virtual machine within Google's Android[5] operating system for mobile devices. The OWL implementation uses Androjena[6], a port of the Jena Semantic Web Framework for Android OS. The mobile device used during the evaluation is a Samsung Infuse[7], with a 1.2 GHz processor, 16GB storage capacity, 512MB of internal memory, and running version 2.3.6 of the Android OS.

To test the efficiency of the two approaches, we timed and averaged 10 executions of each inference task. To test the scalability, we varied the size of the knowledge base along two dimensions – varying the number of properties and features. In the OWL approach, as the number of observed properties increase, the ExplanatoryFeature class (DEF 2) grows more complex (with

[4] http://code.google.com/p/dalvik/
[5] http://www.android.com/
[6] http://code.google.com/p/androjena/
[7] http://www.samsung.com/us/mobile/cell-phones/SGH-I997ZKAATT

more conjoined clauses in the complex class definition). As the number of features increase, the ExpectedProperty class (DEF 3) and NotApplicableProperty class (DEF 4) grows more complex. In the bit vector approach, as the number of properties increase, the number of rows in KB_{BM} grows. As the number of features increase, the number of columns grows.

To evaluate worst-case complexity, the set of relations between properties and features in the knowledge base form a *complete bi-partite graph*[8]. In addition, for the explanation evaluations, every property is initialized as an observed property; for the discrimination evaluations, every feature is initialized as an explanatory feature. This creates the worst-case scenario in which every feature is capable of explaining every property, every property needs to be explained, and every feature needs to be discriminated between. The results of this evaluation are shown in Figure 4.2.

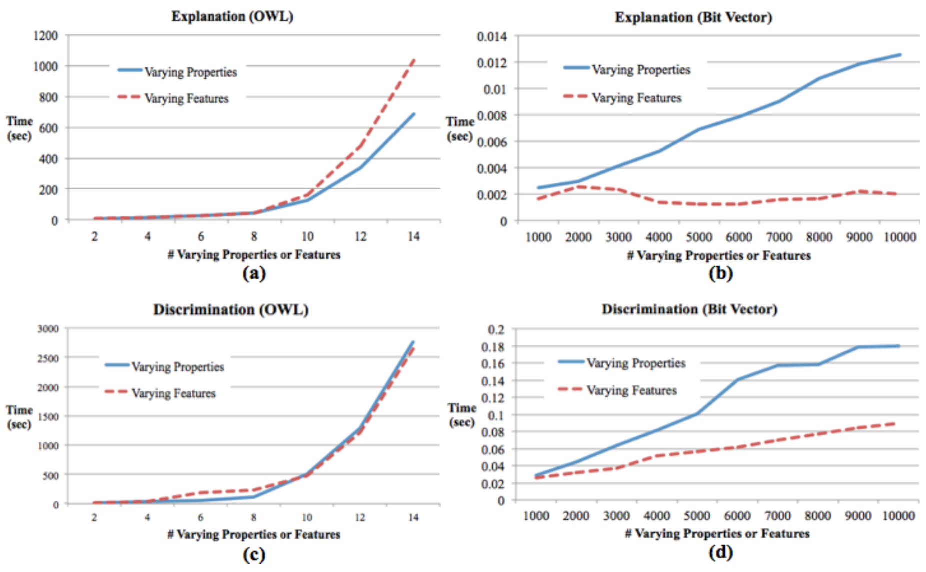


Figure 4.2. Evaluation results: (a) Explanation (OWL) with $O(n^3)$ growth, (b) Explanation (bit vector) with O(n) growth, (c) Discrimination (OWL) with $O(n^3)$ growth, and (d) Discrimination (bit vector) with O(n) growth.

[8] http://en.wikipedia.org/wiki/Complete_bipartite_graph (accessed: June 8, 2012)

Result of OWL evaluations: The results from the OWL implementations of explanation and discrimination are shown in Figures 4.2(a) and 4.2(c), respectively. With a knowledge base of 14 properties and 5 features, and 14 observed properties to be explained, explanation took 688.58 seconds to complete (11.48 min); discrimination took 2758.07 seconds (45.97 min). With 5 properties and 14 features, and 5 observed properties, explanation took 1036.23 seconds to complete (17.27 min); discrimination took 2643.53 seconds (44.06 min). In each of these experiments, the mobile device runs out of memory if the number of properties or features exceeds 14. The results of varying both properties and features show greater than cubic growth-rate ($O(n^3)$ or worse). For explanation, the effect of features dominates; for discrimination, we are unable to discern any significant difference in computation time between an increase in the number of properties vs. features.

Result of bit vector evaluations: The results from the bit vector implementations of explanation and discrimination are shown in Figures 4.2(b) and 4.2(d), respectively. With a knowledge base of 10,000 properties and 1,000 features, and 10,000 observed properties to be explained, explanation took 0.0125 seconds to complete; discrimination took 0.1796 seconds. With 1,000 properties and 10,000 features, and 1,000 observed properties, explanation took 0.002 seconds to complete; discrimination took 0.0898 seconds. The results of varying both properties and features show linear growth-rate ($O(n)$); and the effect of properties dominates.

Discussion of results: The evaluation demonstrates orders of magnitude improvement in both efficiency and scalability. The inference tasks implemented using an OWL reasoner both show greater than cubic growth-rate ($O(n^3)$ or worse), and take many minutes to complete with a small number of observed properties (up to 14) and small knowledge base (up to 19 concepts; #properties + #features). While we acknowledge the possibility that Androjena may have shortcomings (such as an inefficient reasoner and obligation to compute all consequences), our results are in line with Ali et al. [10] that also found OWL inference on resource-constrained devices to be infeasible. On the other hand, the bit vector implementations show linear growth-rate (O(n)), and take milliseconds to complete with a large number of observed properties (up to 10,000) and large knowledge base (up to 11,000 concepts).

Consider the mobile application in which a person's health condition is derived from on-body sensors. A person's condition must be determined quickly, i.e., within seconds (at the maximum), so that decisive steps can be taken when a serious health problem is detected. Also, for the application to detect a wide range of disorders (i.e., features) from a wide range of observed symptoms (i.e., properties) the knowledge base should be of adequate size and scope. In practice, an application may not require a knowledge base of 11,000 concepts; however, many applications would require more than 19 concepts.

The comparison between the two approaches is dramatic, showing asymptotic order of magnitude improvement; with running times reduced from minutes to milliseconds, and problem size increased from 10's to 1000's. For the explanation and discrimination inference tasks executed on a resource-constrained mobile device, the evaluation highlights both the limitations of OWL reasoning and the efficacy of specialized algorithms utilizing bit vector operations.

4.5. Related Work

While OWL is decidable, the computational complexity still limits its practical use within resource-constrained environments. A recent W3C Member Submission [HDT11] proposes a general-purpose RDF binary format for efficient representation, exchange, and query of semantic data; however, OWL inference is not supported. Several approaches to implementing OWL inference on resource-constrained devices include [Ali09][Seitz11][Preuveneers08][Motik12]. Preuveneers et al. [Preuveneers08] have presented a compact ontology encoding scheme using prime numbers that is capable of class-subsumption. Ali et al. [Ali09] have developed Micro-OWL, an inference engine for resource-constrained devices implementing a subset of OWL constructs, but it is not expressive enough for our inference tasks. McGlothlin et al. [McGlothlin10] serialize RDF datasets and materialize data inferred through OWL reasoning using bit vectors. For the inference tasks needed for machine perception, however, it is not scalable. Since we cannot predict which observed properties require explanation, this approach would generate and materialize an ExplanatoryFeature class for all possible (exponentially many) combinations of observable properties. In contrast, we have deployed specially tailored linear algorithms that compute explanation and discrimination efficiently.

4.6. Concluding Remarks

This chapter has demonstrated an approach to machine perception on resource-constrained devices that is simple, effective, and scalable. In particular, two novel contributions were presented: (1) lifting and lowering mappings to enable the translation of knowledge between the high-level semantic representations and low-level bit-vector representations, and (2) efficient algorithms for these inference tasks, using bit vector operations. The bit vector encodings and algorithms yield significant and necessary computational enhancements – including asymptotic order of magnitude improvement, with running times reduced from minutes to milliseconds, and problem size increased from 10’s to 1000’s. The approach is prototyped and evaluated on a mobile device, with promising applications of contemporary relevance (e.g., healthcare/cardiology).

In the future, this approach could be extended to encompass more expressive definitions of explanation (beyond the single-feature assumption), rank explanatory features based on likelihood and/or severity, and rank discriminating properties based on their ability to reduce the number of explanatory features. In addition, the approach could be extended to incorporate stream reasoning through (i) periodic sampling and updating of observations, and (ii) explaining observations within a time window.

As the number and ubiquity of sensors and mobile devices continue to grow, the need for computational methods to analyze the avalanche of heterogeneous sensor data and derive situation awareness will grow. Efficient and scalable approaches to semantics-based machine perception will be indispensable.

The next chapter will discuss the utility of the semantics-based machine perception framework in a real-world scenario. In particular, it will describe the development of a mobile app intended to help reduce hospital readmission rates for patients with congestive heart failure. This is accomplished through the creation of a cardiology knowledgebase and use of the explanation and discrimination inference tasks to recognize a person’s health condition and suggest subsequent actions.

5. Knowledge-enabled Healthcare

Knowledge-enabled Healthcare, or kHealth, is a platform that integrates data from passive and active sensing (including both machine and human sensors) with background knowledge from domain ontologies, semantic reasoning, and mobile computing environments to help people make decisions to improve health, fitness, and wellbeing. kHealth brings together the technologies discussed in previous chapters to enable advanced healthcare applications, including Semantic Sensor Web (Chapter 2), Semantic Perception (Chapter 3), and Intelligence at the Edge (Chapter 4).

Sensors and mobile computing devices are increasingly being used to monitor and manage personal health [Pantelopoulos10][QS12b]. Low-cost (sub-$100), unobtrusive on-body sensors can passively track health-related signals such as heart rate, temperature, galvanic skin response, and activity level. Mobile computing devices and applications can wirelessly collect the sensor data, process the data, and interact with the user. Quantified Self [QS12a], a growing group of individuals using low-cost sensors and mobile apps to track health metrics and share their experiences, exemplifies the trend. This technology has been successful in monitoring and managing simple conditions, i.e., those that can be monitored using a single sensor. The monitoring of complex conditions, such as chronic heart failure, however, involves multiple sensors of different modalities. The data from even a few multimodal sensors can quickly become too complex and confusing for a patient and too time-consuming for a clinician. What is needed is a process to convert the low-level data to high-level knowledge useful for understanding health-related concepts that are relevant to decision-making. To address this challenge, we utilize a semantics-based approach to machine perception to convert data to knowledge through the integration of heterogeneous data and application of perceptual inference. This is now possible through the convergence of multiple technologies, including inexpensive and unobtrusive health sensors, mobile computing platforms, and maturing semantic technologies [Sheth11].

The approach presented above is embodied in an application that integrates data from passive sensors (i.e., on-body sensors), active sensors (i.e., sensors available at home—weight scale, blood pressure monitor, etc.—and personal observation), and health-related background knowledge. This heterogeneous data is integrated and utilized to generate explanations of the low-level physiological data, resulting in high-level knowledge useful for decision-making. The integration of heterogeneous sensor data utilizes Semantic Web technologies, in general, and recent developments in the Semantic Sensor Web (SSW) (Chapter 2), in particular. The application of perceptual inference utilizes recent developments in an ontology of perception (Chapter 3) and efficient algorithms for inference on resource-constrained devices (Chapter 4).

The hypothesis is that semantic integration and perception are effective methods for enabling users and clinicians to find clinically relevant knowledge from multimodal sensing.

5.1. Motivation for Knowledge-enabled Healthcare

Hospital readmission of patients suffering from chronic conditions, such as heart failure, is a growing concern, affecting up to 24.8% of patients [HHS11] and costing $17.4 billion per year [Jencks11]. Heart failure is a chronic disease that affects more than 5 million people in the United States, and more than 550,000 new cases are diagnosed each year [Gheorghiade09]. It accounts for nearly 1.2 million hospitalizations a year as the primary diagnosis [AHA08] and from 2.4 to 3.6 million as a primary or secondary diagnosis [Fang08]. With an aging population, the incidence and prevalence of heart failure is expected to increase. The estimated cost of heart failure in the US for 2008 is $34.8 billion [AHA07]. Approximately 50% of patients are readmitted within 6 months after the index case of heart failure [Butler08] and 70% of readmissions are related to worsening of the previously diagnosed heart failure [Gheorghiade05]. The average rate of readmission within 30 days of discharge for heart failure is 24.8% [HHS11]. Because of the seriousness of this problem, 30-day post-discharge heart failure readmission rates are now being considered as major quality measures for hospitals. In fact, in 2010 President Obama signed into law a new healthcare reform bill that included financial penalties for hospitals with high numbers of preventable readmissions [PPACA10].

A major unresolved challenge is the inability to adequately predict worsening heart failure using either patient self-monitoring or remote telemonitoring of symptoms and daily weight [Abraham11a][Goldberg03][Fonarow04]. Current solutions to this problem employ traditional intervention strategies, such as checkup within 7 days, use of in-body sensors requiring additional surgery at significant expense [Abraham11b], and/or remote-monitoring and telemedicine (sensors/equipment are often very (prohibitively) expensive). The degree of additional commitment (time and money) required from both the patient and the health professionals have impeded adoption of these solutions.

Case Study / Example: In the following scenario, John has been hospitalized over the past five days for Acute Decompensated Heart Failure (ADHF). Unfortunately, patients discharged post-ADHF are frequently readmitted due to poor adherence to both the prescribed medications and low sodium diet. To reduce the risk of readmission over the next 30 days post-discharge, he will be supplied with remote monitoring sensors and a mobile app to help monitor his health. These sensors can measure heart rate, breathing rate, skin temperature, movement, galvanic skin response, electrocardiogram (ECG), weight, blood pressure, and pulse oximetry (SpO2). By monitoring his physiological status through these sensors, the possible deterioration of John's health can be detected, prior to reaching the point of readmission.

After returning home, John ignores dietary restrictions and misses his medications. This results in rapid weight gain due to fluid retention. John begins to have a noticeable increase in respiratory rate and a decrease in oxygen saturation (SpO2). With health-related background knowledge, Intellego can turn these collected data into high-level explanations, and alert both John and a clinician. A clinician reviews John's data from the sensors and proactively contacts him in order to determine that poor adherence is the cause of this deterioration, and advises him accordingly. John increases his dietary and medication adherence, preventing a readmission to the hospital. These interactions are shown in Figure 5.1.

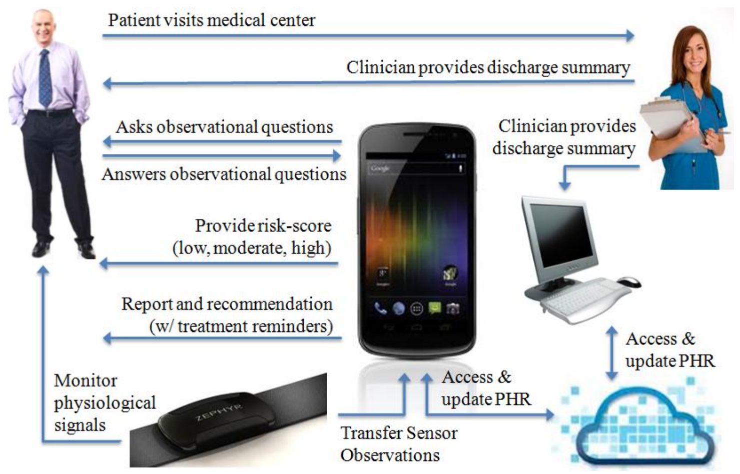

Patient visits medical center
Clinician provides discharge summary
Asks observational questions
Answers observational questions
Clinician provides discharge summary
Provide risk-score (low, moderate, high)
Report and recommendation (w/ treatment reminders)
Access & update PHR
Monitor physiological signals
Access & update PHR
Transfer Sensor Observations

Figure 5.1. Diagram shows the interactions between the patient, clinician, sensors, and mobile device.

5.2. kHealth Application

The activity of observing symptoms and diagnosing a patient's condition is a perceptual act, routinely performed by a clinician. Now, with the advent of sensors, machines also have the ability to measure physiological signals and observe symptoms. Given this ability, many systems simply provide access to raw data, through Internet or mobile access, leading to a deluge of incomprehensible data. What these systems lack, and the clinicians possess, is the ability to effectively glean semantics from observation, to apprehend entities from detected qualities—in short, to perceive.

This section will discuss a practical health application, which uses sensor, mobile, and semantic technologies. In particular, the application leverages *semantic perception* to convert health-related sensor data to knowledge through the integration of heterogeneous data and application of perceptual inference. The integration of heterogeneous sensor data will utilize Semantic Web technologies, in general, and Semantic Sensor Web technologies, in particular. The application of perceptual inference will utilize the ontology of perception, Intellego.

To test the solutions to the challenges of semantic perception in healthcare, a semantics-enhanced sensor and mobile health application is implemented. This application is capable of observing a patient's symptoms, semantically annotating the data, analyzing the data using medical domain knowledge encoded in a clinical cardiology ontology, and providing relevant and useful information to aid the patient and clinician in decision-making. The application is developed for the Android mobile operating system.[1]

5.2.1. Domain Knowledge Base

A cardiology knowledge base was built by extracting knowledge from different sources available on the Web. The knowledge base is expressed using the Resource Description Framework (RDF). The primary source of cardiology knowledge is the Unified Medical Language System (UMLS)

[1] http://www.android.com/

[Bodenreider04]. UMLS is a comprehensive ontology of biomedical concepts designed and maintained by the U.S. National Library of Medicine. All disorders and symptoms in the knowledge base were extracted from UMLS. While UMLS provides the hierarchical relations between terms, it does not provide causal relations between symptoms and disorders. However, using the cardiology-related symptoms and disorders from UMLS, these causal relations were extracted from Healthline.com. Healthline.com (http://www.healthline.com/) is a website that provides access to vast amounts of health-related information. Finally, this knowledge base was vetted by domain experts at ezDI (http://www.ezDI.us) [Perera12]. The resulting cardiology knowledge base used by the kHealth application contains *173 disorders*, *284 symptoms*, and *1944 causal relations* between disorders and symptoms.

Sensors will measure the physiological signals of the user and – using a Bluetooth wireless connection – transmit the measurement data to the application, running on the mobile device. Two types of sensing are utilized, including both passive sensing and active sensing.

5.2.2. Passive Sensing

With passive sensing, low-cost, unobtrusive, on-body sensors continuously monitor the patient. These sensors are aptly referenced as "wear-em-and-forget-em" data tracking devices, and their use requires very little commitment from the user; they must simply wear the sensors, carry the mobile computing device (i.e., smart phone), and in some cases charge them (sensors and mobile devices). Such passive sensors include: heart-rate sensor, accelerometer, temperature sensor, and galvanic skin response sensor. We have been prototyping with an accelerometer (http://fitbit.com) and a heart-rate monitor (http://www.zephyr-technology.com/consumer-hxm). The data from the sensors is automatically transferred to the mobile device through a Bluetooth wireless connection. The heterogeneous data from the different sensors is then semantically annotated with concepts from the SSN ontology and cardiology knowledge base. The low-level semantically annotated observations are then converted to useful and actionable knowledge (i.e., explanations/abstractions).

As an example, suppose that a patient is wearing a heart-rate sensor and a galvanic skin response sensor, resulting in the observations of *tachycardia* and *clammy skin*. Given these observed symptoms and the cardiology background knowledge, the kHealth app may generate a set of explanations including *panic disorder*, *hypoglycemia*, *hyperthyroidism*, *myocardial infarction*, and *septic shock*. The user and/or clinician may then surmise that, given the users recent history of heart disease, this set of explanations is troubling, resulting in follow up treatment.

5.2.3. Active Sensing

Active sensing requires further participation and commitment by the user. The goal is to collect information from additional (active) sensors available to the user (e.g., weight scale) and observations made by the users themselves (e.g., feeling chest pain). This additional information is then used to minimize the set of explanations (generated during the passive sensing phase). The types of sensors used in this phase include: blood pressure monitor and weight scale.

Contemporary services such as WebMD.com and HealthLine.com request that patients enter their symptoms into a Web form so that the system can provide additional information about potential causes. A better approach, as exemplified in this application, is to utilize the derived explanations from the passive sensing phase, together with the background knowledge and the focus functionality of Intellego, to generate and ask relevant and targeted questions about the symptoms of the user. Such questions may require access to sensors available to the user at home (e.g., blood pressure monitor), or the questions may only be answerable by the user themselves (e.g., “Are you experiencing chest pain?”). This question-and-answer interaction between the application and user proceeds in the form of a common chat dialog (Figure 5.2) to efficiently minimize the set of explanations.

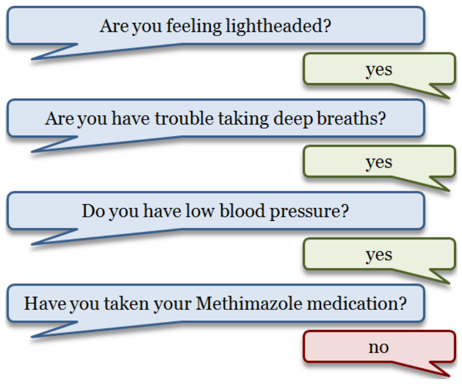

Are you feeling lightheaded?
yes
Are you have trouble taking deep breaths?
yes
Do you have low blood pressure?
yes
Have you taken your Methimazole medication?
no

Figure 5.2. Active sensing through a chat dialog.

Continuing the previous example, recall the *tachycardia* and *clammy skin* observations from the passive sensing phase, resulting in a set of explanations, including *panic disorder*, *hypoglycemia*, *hyperthyroidism*, *myocardial infarction*, and *septic shock*. To minimize this set of explanations, the application seeks informative observations (by asking questions) regarding lightheadedness, trouble breathing, and low blood pressure (since the patient has access to a blood pressure monitor). With these additional observations, the App updates the explanations to include *hypoglycemia* and *hyperthyroidism*.

5.2.3. Application Walkthrough

As discussed above, kHealth is a knowledge-enabled (semantic) application development framework with the ability to create advanced healthcare applications. This section provides a brief walkthrough and screenshots of an implemented kHealth application for Android devices. The main interface screen provides links to the various screens of the application (see Figure 5.3).

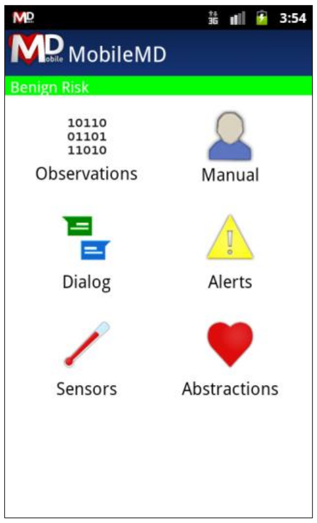


Figure 5.3. Screenshot of the main interface screen of the kHealth application

Below is a quick overview of the functionality:

- *Observations* screen shows current observed symptoms, from both passive and active sensing.
- *Manual* screen allows the user to manually enter their symptoms.
- *Dialog* screen asks the user questions about their symptoms.
- *Alerts* screen provides a log of observed precarious symptoms.
- *Sensors* screen provides access to the various sensors available to the application.
- *Abstractions* screen shows current explanations (i.e., disorders that explain the current set of observed symptoms).

In the *observations* screen, observation measurements and detected symptoms are shown (see Figure 5.4(a)). Machine observation measurements, e.g., heart rate, can be seen in real time. Also, symptoms detected by the user, and entered through either the manual screen or dialog screen, are also provided. The *abstractions* screen shows the disorders that explain (or account for) the observed symptoms (see Figure 5.4(b)).

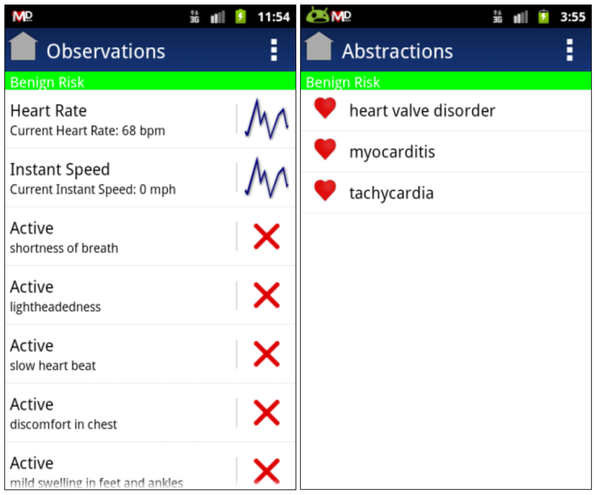
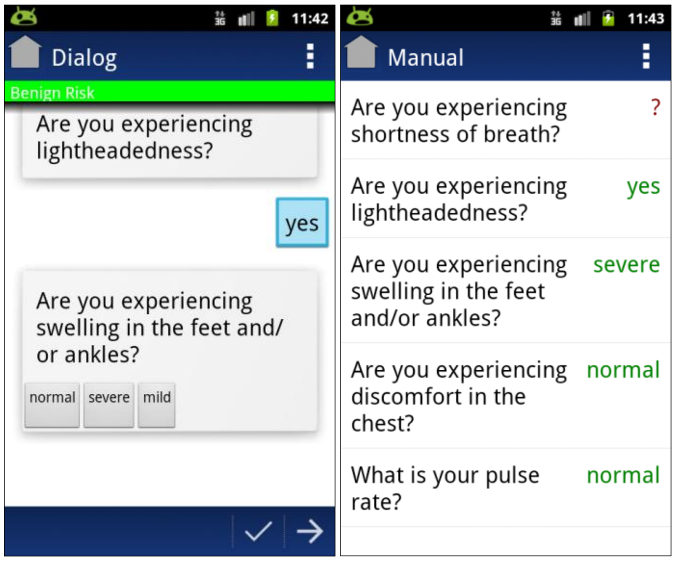


Figure 5.4. Screenshot of the (a) observations screen and (b) abstractions screen of the kHealth application.

In order to narrow down the set of explanatory disorders, the application will ask the user specific questions through a dialog screen (see Figure 5.5(a)). The questions are based on discriminating symptoms (i.e., symptoms that can discriminate between multiple explanatory disorders). The user may also enter their symptoms manual, through the manual screen, if they prefer (see Figure 5.5(b)).


11:42

Dialog

Benign Risk

Are you experiencing lightheadedness?

yes

Are you experiencing swelling in the feet and/ or ankles?

normal severe mild

11:43

Manual

Are you experiencing shortness of breath?	?
Are you experiencing lightheadedness?	yes
Are you experiencing swelling in the feet and/or ankles?	severe
Are you experiencing discomfort in the chest?	normal
What is your pulse rate?	normal

Figure 5.5. Screenshot of the (a) dialog screen and (b) manual screen of the kHealth application.

5.3. Related Work

The most similar related technology to kHealth is a technology called WANDA B – Weight and Activity with Blood Pressure Monitoring System – developed by the University of California - Los Angeles (UCLA) Wireless Health Institute in conjunction with the UCLA Nursing School. WANDA B. is a wireless health technology that uses sensors and wireless communication with a smart phone to remotely monitor patients with cardiovascular disorders [Suh11]. Figure 5.6 shows the architecture of WANDA B.

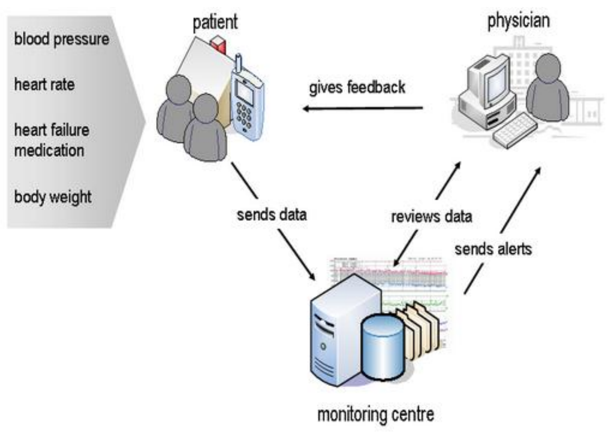


Figure 5.6. WANDA B. Architecture

Similar to the kHealth application discussed in this section, WANDA B. also engages in active sensing by asking questions of the user through SMS (Short Message Service) messaging. Each day the user is using the application, WANDA asks the user 12 questions derived from the Heart Failure Somatic Awareness Scale (HFSAS). HFSAS tracks 12 metrics, or symptoms, that are indicative of Congestive Heart Failure, including shortness of breath, swollen feat, chest pains, and elevated heart rate [Suh11]. Figure 5.7 shows the questionnaire items of the HFSAS.

Questionnaire items
I could feel my heart beat faster
I could not breathe when I laid down
I felt pain in my chest
I had an upset stomach
I had a cough
I was tired
I could not catch my breath
My feet were swollen
I woke up at night because I could not breathe
My shoes were tighter than usual
I gained 3 or more pounds in the past week
I could not do my usual daily activities because I was short of breath

Figure 5.7. Questionnaire items of the Heart Failure Somatic Awareness Scale (HFSAS)

Previous telemedicine approaches, such as WANDA B., have often focused on alerts based on a single parameter threshold. To date, studies utilizing single diagnostic parameters such as intra-thoracic impedance monitoring have been disappointing and, in the only intervention trial yet performed, resulted in an increase in heart failure hospitalizations [Veldhuisen11]. Our hypothesis is that kHealth provides an improvement over current state-of-the-art technologies. A comparison with the closest effort with similar objectives, the WANDA project, is given in Table 5.1.

Table 5.1. Comparison of kHealth with WANDA.

	kHealth	WANDA
Architecture	Uses local computation on mobile devices.	Sends all sensor data to a remote server at a clinic.
Technology: Device	Mobile Device: Android smartphone to collect sensor observations, compute results, and display information. Passive Sensors: Heart Rate Monitor. Active Sensors: Weight Scale, Blood Pressure Monitor, Citizen Sensor (users observing their own symptoms)	Mobile Device: Android smartphone to collect sensor observations and display information. Web Server: sends observations to web server to compute results Active Sensors: Weight Scale, Blood Pressure Monitor, Heart Rate Monitor, Activity Monitor, Citizen Sensor (limited to symptoms within the Heart Failure Somatic

		Awareness Scale (HFSAS)).
Technology: Computation	Abductive inference to *explain* symptoms (utilizing bipartite graph relating symptoms to disorders). Active Perception by asking targeted questions to users.	For each observed symptom, determine if the measured value crosses some threshold (utilizing domain knowledge in the form of thresholds for each symptom value).
Advancing the Four P's of Healthcare: Personalization, Prevention, Participation, and Prediction	Personalization: Limited to sensing parameter thresholds based on population level information. Participation: Provides focused questioning, and proactive engagement. Prediction: Provides an explanation of observed symptoms, which may predict further symptoms. Prevention: Provides explanation of symptoms that can be used to derive actions and treatment.	Personalization: Limited to sensing parameter thresholds based on population level information. Participation: Limited to asking standard questions (e.g., HFSAS). Prediction: Prediction of readmission using machine learning techniques. Prevention: Limited since machine learning techniques are black box (i.e., do not provide reason for prediction).

5.4. Evaluation

In order to demonstrate the value of our approach – a semantics-based approach for machine perception – in the domain of healthcare, an evaluation has been conducted that focuses on the discrimination operation of Intellego. In particular, the evaluation focuses on the ability to discriminate between sets of potential disorders using (1) a restricted set of symptoms (12 used by HFSAS/WANDA), and (2) a more comprehensive set of symptoms (284 used by kHealth).

This evaluation will demonstrate that the knowledge base and methodology used by WANDA is too limited to effectively derive a minimum set of explanations and determine the specific condition of the user. On the other hand, the use of Intellego to determine observable properties (i.e., symptoms) that can discriminate between multiple explanations enables the ability to ask questions of the user from a larger set of possible symptoms. The use of a knowledge base with a larger set of symptoms, and the ability to determine which symptoms are important, improves the ability of the health application to determine the condition of the user.

For this evaluation, 496 electronic medical records were used, provided by ezDI (http://www.ezDI.us). These records provided grounding for the evaluation by specifying the correct diagnosis of a patient. This evaluation is composed of two steps: (1) extracting a diagnosed disorder from an EMR record, and (2) discriminate between the multiple possible disorders and derive the minimum set of disorders, preferably the diagnosed disorder extracted from the EMR (see Algorithm 5.1 below).

Algorithm 5.1: Generate potential disorders – The task of this algorithm is to find the set of disorders that the patient could have had, before the medical professional minimized the set of hypotheses and produced a diagnosis. In other words, determine all the possible disorders that the doctor had to consider and dismiss in order to determine the current disorder and place it within the diagnosis in the EMR. To accomplish this, prior disorders are extracted from the EMR (D_{prior}); there is a section of the EMR dedicated to prior disorders.

[1] Then using knowledge in the cardiology knowledge base, the set of all symptoms that could be caused by one of these prior disorders are found (S_{prior}).

[2] Next, for the diagnosed disorder, the set of symptoms that could be caused by this disorder are found (S_{new}).

[3] The set of symptoms that the new disorder shares with the prior disorders are found (S_{common}). This set of common symptoms represents the observable symptoms that a medical professional would need to consider while discriminating between multiple potential explanations. More specifically, these are the symptoms that are capable of discriminating between disorders the patient has (d_{new} and D_{prior}) and the disorders the patient does not have.

[4] Finally, the set of disorders that explain at least one of the common symptoms are found (D_{poss}). This set of disorders represents the set of disorders that the patient could have had, before being diagnosed by the medical professional. In other words, disorders that the doctor considered and dismissed during the visit.

Input
- D_{prior}: set of previously diagnosed disorders, extracted from EMR (based on prior visits)
- d_{new}: newly diagnosed disorder, extracted from EMR (based on current visit)

Procedure
```
[1] $S_{prior}$ := causes($D_{prior}$)          // union of all symptoms caused by a prior disorder
[2] $S_{new}$ := causes($d_{new}$)              // union of all symptoms of new disorder ($d_{new}$)
[3] $S_{common}$ := intersection($S_{prior}$,$S_{new}$)  // intersection of symptoms that this
                                                // new disorder and prior disorders have in common
[4] $D_{poss}$ := causedBy($S_{common}$)        // disorders that cause a common symptom
```

Algorithm 5.1: Discriminate between potential disorders – The task of this algorithm is to minimize the set of potential disorders and derive the diagnosed disorder extracted from the EMR. This task will use the discrimination operator, given a set of potential disorders to discriminate between, a set of observable symptoms, and a knowledge base encoding the causal relations between symptoms and disorders. More specifically, the algorithm will minimize the set

of potential disorders (all known cardiology-related disorders defined within the cardiology knowledge base) by discriminating them with the diagnosed disorder.

[1] First, a copy of the set of possible disorders to be minimized is created (D_{min}).

[2] Next, the algorithm will iterate through each member of the set of possible disorders (d_i).

[3] For each disorder in the set of potential disorders, the set of observable symptoms capable of discriminating between this potential disorder (d_i) and the actually diagnosed disorder (d_{new}) are found (S_{disc}).

[4] If the set of discriminating symptoms is empty then the minimum set of potential disorders has been found (D_{min} is minimal).

[5] Otherwise, find the set of potential disorders that are causally related to any of the discriminating symptoms (D_{min}).

[6] If the set of filtered potential disorders is of size 1 or less then the minimum set has been found (D_{min} is minimal).

Continue to iterate through this process until the set of potential disorders is minimum; i.e., the set contains only one disorder (the diagnosed disorder) or there are still discriminating symptoms. As discussed in the results section below, the number of times this process executes measures efficiency, and the number of members in the minimum set of potential disorders measures specificity.

Input

- D_{poss}: set of disorders to discriminate between (generated by Algorithm 5.1)
- S_{obs}: set of all observable symptoms, which are accessible by the algorithm
- d_{new}: newly diagnosed disorder, extracted from EMR (based on current visit)

Procedure

```
[1] $D_{min}$ == $D_{poss}$                          // create copy of set of possible disorders
[2] for each $d_i$ in $D_{poss}$
[3]     $S_{disc}$ := discriminator($d_{new}$,$d_i$,$S_{obs}$)  // find discriminators between
```

```
                                          // a disorder the patient could have ($D_i$) and the disorder
                                          // the patient does have ($d_{new}$) from the set of observable
                                          // symptoms ($S_{obs}$)
[4]     if size($S_{disc}$) == 0 then break // set of potential disorders is minimum
[5]     $D_{min}$ := causedBy($S_{disc}$)      // set of disorders that cause a discriminating symptom
[6]     if size($D_{min}$) <= 1 then break // set of potential disorders is minimum
```

Experiment – To compare the ability of the two competing approaches to discriminate between disorders, two experiments were run. The first experiment evaluated the ability of the 12 symptoms from HFSAS to discriminate between potential disorders. The second experiment evaluated the ability of the symptoms defined in the cardology knowledge base (Section 5.2.1) to discriminate between potential disorders. The set of potential disorders (from the cardiology knowledge base) is the same for both experiments. The difference between the two experiments lies in the set of observable symptoms available to help discriminate between the potential disorders (S_{obs} in Algorithm 5.1). Particularly, in the first experiment the set of observable symptoms (S_{obs}) is composed of the 12 symptoms of HFSAS (see Section 5.3). In the second experiment, the set of observable symptoms (S_{obs}) is composed of the 284 symptoms from the cardiology knowledge base (see Section 5.2.1).

Experiment results – For this evaluation, 496 electronic medical records (EMRs) were used, selected at random from 3172 EMR records provided by ezDI. There were an average of 3.2 diagnosed disorders extracted from each EMR. The cardiology knowledge base (from Section 5.2.1) provided 173 potential disorders.

Result from experiment #1: The first experiment tested the ability of the 12 symptoms from the HFSAS to discriminate between potential disorders. This required an average of 7.45 cycles (i.e., executions of the discrimination operator) to find the minimum set of explanations. In this case, "minimum set" means that the set of explanations cannot be further reduced by observing any new symptom. In addition, the minimum set of explanations contained 11.95 disorders, on average. The minimum set of explanations converged to include only the one correct explanation, i.e., the disorder within the EMR diagnosis, 20% of the time.

Result from experiment #2: The second experiment tested the ability of the 284 symptoms defined in the cardiology knowledge base (Section 5.2.1) to discriminate between potential disorders. This required an average of 7.28 cycles (i.e., executions of the discrimination operator) to find the minimum set of explanations. 25% of the chosen discriminators were from the set of 12 HFSAS symptoms, 75% were not included in the set of 12 HFSAS symptoms. The minimum set of explanations contained 1 disorder, on average. In other words, the minimum set of explanations converged to include only the one correct explanation, i.e., the disorder within the EMR diagnosis, each time.

These two experiments clearly demonstrate that the ability to observe and analyze a larger set of symptoms (as used by kHealth) produces more efficient and more precise results as compared to the use of a more restricted set of symptoms (as used by HFSAS/WANDA). More specifically, the minimum set of explanations can be generated quicker (i.e., using less cycles) given the larger set of known symptoms from the cardiology knowledge base (~7.28 cycles) vs. the smaller set of symptoms from HFSAS (~7.45 cycles). In addition, even though the minimum set is generated more efficiently, the resulting set is also more specific using the larger set of known symptoms (1 disorder in minimum set of explanations) vs. the smaller set of symptoms from HFSAS (~11.95 disorders in the minimum set of explanations). *Therefore, the approach utilized by kHealth is shown to be both more efficient and specific than HFSAS/WANDA for discriminating between potential disorders.*

5.5. Concluding Remarks

A semantics-based approach to machine perception is not only novel, but also demonstrably effective and practical. Knowledge-enabled Healthcare (kHealth) is the result of the use of the techniques and methodologies presented in this dissertation in the healthcare domain. In collaboration with cardiology domain scientists at Ohio State University, Wexner Medical Center, a mobile application has been developed to help people improve their cardiovascular fitness. Currently the app is being used by patients at the OSU Wexner Medical Center in a pre-clinical usability trial with the aim of reducing preventable hospital readmissions of patients with Acute Decompensated Heart Failure. Hopefully, in the near future, such research may be further utilized to empower patients with low-cost, easy-to-use technologies to increase their participation in their own healthcare and health decision-making in order to improve their health, fitness, and wellbeing.

6. Conclusion

Machine perception is still a hard problem in computer science with many issues to be addressed. This dissertation addressed the question of whether machine perception could be formalized using semantic web technologies in order to derive abstractions from sensor data using background knowledge on the Web, and efficiently executed on resource-constrained devices. The particular issues addressed include the semantic annotation of sensor data, the interpretation of sensor data, and the efficient and scalable execution on resource-constrained devices. The chapters of this dissertation demonstrated the techniques employed to address these issues, as well as real world use-cases demonstrating the value of the approach towards solving real problems. More concretely, the addressed issues include:

1. Develop techniques for semantically annotating sensor descriptions and sensor observation data on the Web to enable advanced integration, query, and inference (Chapter 2: Semantic Sensor Web).
2. Develop techniques for interpreting semantically annotated sensor observation data to derive actionable intelligence and situational awareness (i.e., high-level abstractions), using background knowledge on the Web (Chapter 3: Semantic Perception).
3. Develop techniques to enable the efficient and scalable interpretation of semantically annotated sensor observation data on resource-constrained devices (Chapter 4: Intelligence at the Edge).
4. Develop a prototype application to demonstrate the utility of the semantics-based machine perception framework in a real-world scenario (Chapter 5: Knowledge-enabled Healthcare).

6.1. Summary

The semantic annotation of sensor data was addressed in Chapter 2: Semantic Sensor Web. This chapter demonstrated techniques for semantically annotating sensor descriptions and sensor observation data on the Web to enable advanced integration, query, and inference. This was accomplished by marrying the Web-based sensor description languages defined by the Open Geospatial Consortium's Sensor Web Enablement (SWE) and the Web-based ontology languages defined by the World Wide Web Consortium. The primary contributions of this chapter included (1) an ontology for representing sensor descriptions and sensor observation data, the Semantic Sensor Network (SSN) ontology (2) a framework for semantically annotating sensor descriptions and sensor observation data encoded SWE's XML based languages and service descriptions, and (3) a semantic sensor observation service (SemSOS) that utilizes the SSN ontology and the semantic annotation framework to enable better integration, query, and inference capabilities over sensor data in comparison with the sensor observation service (SOS) defined by SWE.

The interpretation of sensor data was addressed in Chapter 3: Semantic Perception. This chapter demonstrated techniques for interpreting semantically annotated sensor observation data to derive actionable intelligence and situational awareness (i.e., high-level abstractions), using background knowledge on the Web. The primary contribution of this chapter included an ontology of perception, Intellego, which is derived from cognitive models of perception and provides a formal semantics of machine perception. This ontology is first encoded in set-theoretic notation to provide a formal representation of the concepts and processes involved in perception. Particular aspects of the ontology – i.e., explanation and discrimination – are then encoded in the Web Ontology Language to enable better integration with data and knowledge on the Web.

The ability to interpret sensor data efficiently, and at scale, on resource-constrained devices, such as smart phones, was addressed in Chapter 4: Intelligence at the Edge. In Chapter 3 the explanation and discrimination operators of Intellego were encoded in OWL to enable better integration with data and knowledge on the Web. The computational complexity of the Web Ontology Language (OWL), however, seriously limits its applicability and use within resource-constrained environments, such as mobile devices. To address this issue, this chapter demonstrated techniques to enable the efficient and scalable interpretation of semantically annotated sensor observation data on resource-constrained devices. More specifically, the primary contributions of this chapter included efficient algorithms for the explanation and discrimination operators of Intellego, using bit-vector encodings and operations.

The ability of the approach presented in this dissertation to help address a real-world problem was addressed in Chapter 5: Knowledge-enabled Healthcare. This chapter demonstrated a prototype application to show the utility of the semantics-based machine perception framework to help people make decisions to improve health, fitness, and wellbeing. The primary contribution of this chapter is a semantics-based platform called kHealth, Knowledge-enabled Healthcare, which integrates data from passive and active sensing with background knowledge from domain ontologies, semantic reasoning, and mobile computing environments. kHealth utilizes technology discussed within this dissertation – i.e., the semantic annotation of sensor data (Semantic Sensor Web), interpretation of sensor data (Semantic Perception), and efficient algorithms for interpreting sensor data on resource-constrained devices (Intelligence at the Edge) – to enable advanced healthcare applications. Currently, the application of kHealth towards the management of several disorders, including chronic heart disease and asthma, is being investigated in collaboration with clinicians.

6.2. Final Remarks

Sensors are quickly becoming ubiquitous and are now collecting data about our environment at an extraordinary pace. This dissertation has demonstrated the substantial benefits to be gained in processing sensor data by semantically annotating the data and automating an approximation of how people perceive their environment efficiently. Specifically, this ability is afforded by background knowledge and the cyclical nature of observation and perception processes. While one can take different positions on the philosophical (or technical) foundations of perception, it is clear that a careful ontological specification makes these positions explicit and testable. The approach described in this work establishes a formal semantics for machine perception; and provides a solution to difficult challenges, such as the ability to effectively model the process of perception, to provide an appropriate interpretation of observational data with incomplete information, and to efficiently interpret the growing stream of observational data on resource-constrained devices.

As the number and ubiquity of sensors and mobile devices continue to grow, the need for computational methods to analyze the avalanche of heterogeneous sensor data and derive situation awareness will grow. Efficient and scalable approaches to semantics-based machine perception will be indispensable. This research represents a thoughtful and earnest step towards understanding and addressing this challenge.

Bibliography

[52North] 52North Sensor Web Community, http://52north.org

[Aberer06] Aberer, K., Hauswirth, M., & Salehi, A.: A middleware for fast and flexible sensor network deployment. Proceedings of the 32nd International Conference on Very Large Data Bases, pp.1199–1202 (2006).

[Abraham11a] Abraham, W., Compton, S., Haas, G., Foreman, B., Canby, R., Fishel, R., et al.: Intrathoracic impedance vs daily weight monitoring for predicting worsening heart failure events: results of the Fluid Accumulation Status Trial (FAST). Congest Heart Fail, 17, 51-55 (2011).

[Abraham11b] Abraham, W., Adamson, P., Bourge, R., Aaron, M., Costanzo, M., Stevenson, L., et al.: CHAMPION Trial Study Group. Wireless pulmonary artery haemodynamic monitoring in chronic heart failure: a randomised controlled trial. Lancet, 377 (9766), 658-666 (2011).

[AHA07] American Heart Association: Heart Disease and Stroke Statistics--2008 Update: A Report From the American Heart Association Statistics Committee and Stroke Statistics Subcommittee. Journal of the American Heart Association (2007). http://circ.ahajournals.org/content/117/4/e25.full.pdf

[AHA08] American Heart Association: Heart Disease and Stroke Statistics--2009 Update: A Report From the American Heart Association Statistics Committee and Stroke Statistics Subcommittee. Journal of the American Heart Association (2008). http://circ.ahajournals.org/content/119/3/e21.full.pdf

[Ali09] Ali, S., Kiefer, S.: μOR – A Micro OWL DL Reasoner for Ambient Intelligent Devices. 4th Intl. Conf. on Grid and Pervasive Computing, 5529, pp.305–316, Geneva, Switzerland, May 4-8 (2009).

[Aloimonos88] Aloimonos, J., Weiss, I., Bandyopadhyay, A.: Active Vision. International Journal of Computer Vision, 1(4), 333-356 (1988). MIT Press. doi:10.1007/BF00133571

[Anandavala10] Anandavala: Perception, Cognition, and Communication (PCC) Ontology. Anandavala. Retrieved from http://www.anandavala.info/ontech/PCC/pcc-intro.html

[Baader03] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F.: The Description Logic Handbook: Theory , Implementation , and Applications. (F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, & P. F. Patel-Schneider, Eds.) Description Logic Handbook, pp. 622 (2003). Cambridge University Press. doi:10.2277/0521781760

[Bajcsy88] Bajcsy, R.: Active perception. IEEE, 76(8), pp.996-1005 (1988).

[Bizer09] Bizer, C., Heath, T., & Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems, 5(3), 1-22 (2009). Elsevier. doi:10.4018/jswis.2009081901

[Bodenreider04] Bodenreider, O.: The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. Nucleic acids research, 32 (suppl 1):D267-D270 (2004).

[Bonatti09] Bonatti, P., Lutz, C., Wolter, F.: The Complexity of Circumscription in DLs. J. Artif. Intell. Res. 35: 717-773 (2009).

[Borgo09] Borgo, S., & Masolo, C.: Foundational choices in DOLCE. Applied Ontology, 2, 361-381 (2009). Springer-Verlag.

[Botts08] Botts, M., Percivall, G., Reed, C., & Davidson, J.: OGC (R) Sensor Web Enablement: Overview and High Level Architecture. Lecture Notes in Computer Science, Vol. 4540, pp. 175–190 (2008). Springer. doi:10.1007/978-3-540-79996-2

[Butler08] Butler, J., & Kalogeropoulos, A.: Worsening heart failure hospitalization epidemic we do not know how to prevent and we do not know how to treat! J Am Coll Cardiol, 52, 435-437 (2008).

[Calbimonte11] Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Semantic Sensor Data Search in a Large-Scale Federated Sensor Network. 4th Intl. Workshop on Semantic Sensor Networks, Bonn, Germany, Oct. 23-27 (2011).

[Calder10] Calder, M., Morris, R. A., & Peri, F.: Machine reasoning about anomalous sensor data. Ecological Informatics, 5(1), 9-18 (2010). Elsevier B.V. doi:10.1016/j.ecoinf.2009.08.007

[Carroll04] Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., & Wilkinson, K.: Jena: implementing the semantic web recommendations. Digital Media, (HPL-2003-146), 74-83 (2004). ACM. doi:10.1145/1013367.1013381

[Cavanagh99] Cavanagh, P.: Top-Down Processing in Vision. The MIT Encyclopedia of the Cognitive Sciences, (Costall 1980), 839-840 (1999).

[Compton09] Compton, M., Henson, C., Lefort, L., Neuhaus, H., & Sheth, A.: A Survey of the Semantic Specification of Sensors. In K. Taylor, A. Ayyagari, & D. D. Roure (Eds.), International Workshop on Semantic Sensor Networks, Vol. 522, p.17 (2009). CEUR-WS.org.

[Compton12] Compton, M. et al.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics (2012).

[Corcho10] Corcho, O., & García-Castro, R.: Five challenges for the Semantic Sensor Web. Semantic Web - Interoperability, Usability, Applicability, 1(1), 121-125 (2010). IOS Press. doi:10.3233/SW-2010-0005

[Devaraju10] Devaraju, A., Kuhn, W.: A Process-Centric Ontological Approach for Integrating Geo-Sensor Data. 6th Intl. Conf. on Formal Ontology in Information Systems, pp.199-212, Toronto, Canada, May 11-14 (2010).

[Devaraju11] Devaraju, A., Kauppinen T.: Sensors Tell More than They Sense: Modeling and Reasoning about Sensor Observations for Understanding Weather Events. Special Issue on Semantic Sensor Networks, Intl. Journal of Sensors, Wireless Communications and Control, Bentham Science Publishers (2011).

[Diamant07] Diamant, E.: Modeling human-like intelligent image processing: An information processing perspective and approach. Signal Processing Image Communication, 22(6), 583-590 (2007). doi:10.1016/j.image.2007.05.007

[Elsenbroich06] Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. Workshop on OWL: Experiences and Directions, Athens, GA, USA, Nov. 10-11 (2006).

[Euzenat07] Euzenat, J., & Shvaiko, P.: Ontology Matching. Booksgooglecom, pp. 1 – 333 (2007). Springer-Verlag, Berlin Heidelberg. doi:10.1007/978-3-540-49612-0

[Fang08] Fang, J., Mensah, G., Croft, J., & Keenan, N.: Heart failure related hospitalization in the U.S., 1979 to 2004. J Am Coll Cardiol, 52, 428-434 (2008).

[Fonarow04] Fonarow, G., & Corday, E.: ADHERE Scientific Advisory Committee. Overview of acutely decompensated congestive heart failure (ADHF): a report from the ADHERE registry. Heart Fail Rev, 9, 179-185 (2004).

[Gheorghiade05] Gheorghiade, M., Zannad, F., Sopko, G., Klein, L., Pina, I., & Konstam, M.: Acute heart failure syndromes: current state and framework for future research. Circulation, 112, 3958-3968 (2005).

[Gheorghiade09] Gheorghiade, M., & Pang, P.: Acute heart failure syndromes. J Am Coll Cardiol, 53, 557-573 (2009).

[Gibson66] Gibson, J. J.: The Senses Considered as Perceptual Systems. Leonardo, Vol. 1, p. 335 (1966). Houghton Mifflin. doi:10.2307/1571911

[GML] Geography Markup Language (GML), http://www.opengeospatial.org/standards/gml

[Goldberg03] Goldberg, L., Piette, J., Walsh, M., Frank, T., Jaski, B., Smith, A., et al.: Randomized trial of a daily electronic home monitoring system in patients with advanced heart failure: the Weight Monitoring in Heart Failure (WHARF) trial. Am Heart J., 146 (4), 705-712 (2003).

[Goldstine64] Goldstine, H. H.: Computers and Perception. Proceedings of the American Philosophical Society, 108(4), pp. 282-290 (1964). American Philosophical Society.

[Gray11] Gray, A., et al.: A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data. 9th Extended Semantic Web Conf., Heraklion, Greece, May 29 – June 2 (2011).

[Gregory68] Gregory, R.: (1968. Perceptual illusions and brain models. Proceedings of the Royal Society of London, 171(24), 279-296 (1968).

[Gregory97] Gregory, R.L.: Knowledge in perception and illusion. In: Philosophical Transactions of the Royal Society of London, 352(1358), pp.1121-1127 (1997).

[Gries99] Gries, D.: Monotonicity in Calculational Proofs. Correct System Design, Recent Insight and Advances, pp. 79-85, London, UK: Springer-Verlag (1999).

[Grimm09] Grimm, S., Hitzler, P.: A Preferential Tableaux Calculus for Circumscriptive ALCO. In: Polleres, A., Swift, T. (Eds.), Web Reasoning and Rule Systems, Third International Conference, RR 2009, Chantilly, VA, USA, October 2009, Proceedings. Lecture Notes in Computer Science Vol. 5837, Springer, pp. 40-54 (2009).

[Gross99] Gross, N.: The Earth Will Don an Electronic Skin. BusinessWeek, Aug. (1999). www.businessweek.com/1999/99_35/b3644024.htm

[Gruber93] Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), pp.199-220 (1993).

[Gürgen06] Gürgen, L., Roncancio, C., Labbé, C., & Olive, V.: Transactional issues in sensor data management. Proceedings of the 3rd Workshop on Data Management for Sensor Networks in conjunction with VLDB 2006, 27 (2006). ACM Press. doi:10.1145/1315903.1315910

[HDT11] Binary RDF Representation for Publication and Exchange (HDT). W3C Member Submission (2011). http://www.w3.org/Submission/2011/SUBM-HDT-20110330/

[HHS11] U.S. Department of Health & Human Services: Hospital Compare (2011). http://www.hospitalcompare.hhs.gov (Accessed on February 19, 2012).

[Henson09] Henson, C., Pschorr, J. K., Sheth, A. P., & Thirunarayan, K. (2009). SemSOS: Semantic sensor Observation Service. 2009 International Symposium on Collaborative Technologies and Systems, 0(Cts), 44-53 (2009). Ieee. doi:10.1109/CTS.2009.5067461

[Henson11a] Henson, C., Thirunarayan, K., Sheth, A., Hitzler, P.: Representation of Parsimonious Covering Theory in OWL-DL. 8th Intl. Workshop on OWL: Experiences and Directions, San Francisco, CA, USA, June 5-6 (2011).

[Henson11b] Henson, C., Sheth, A., Thirunarayan, K.: Semantic Perception: Converting Sensory Observations to Abstractions. IEEE Internet Computing, 16(2), pp.26-34, Mar/Apr (2012).

[Henson11c] Henson, C., Thirunarayan, K., Sheth, A.: An Ontological Approach to Focusing Attention and Enhancing Machine Perception on the Web. Applied Ontology, 6(4), pp.345–376 (2011).

[Henson12] Henson, C., Thirunarayan, K., Sheth, A.: An Efficient Bit Vector Approach to Semantics-based Machine Perception in Resource-Constrained Devices. Proceedings of 11th International Semantic Web Conference (ISWC 2012), Boston, Massachusetts, USA, November 11-15 (2012).

[Higginbotham10] Higginbotham, S.: Sensor Networks Top Social Networks for Big Data. Gigaom.com, September (2010). Retrieved from http://gigaom.com/cloud/sensor-networks-top-social-networks-for-big-data-2/

[Hitzler09] Hitzler, P., Parsia, B., Patel-Schneider, P. F., & Rudolph, S.: OWL 2 Web Ontology Language Primer. W3C Recommendation (2009).

[Horrocks04] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Recommendation (2004).

[Janowicz10] Janowicz, K., Schade, S., Bröring, Arne, Keßler, C., Maué, P., & Stasch, C.: Semantic Enablement for Spatial Data Infrastructures. (J. P. Wilson, A. S. Fotheringham, & D. O’Sullivan, Eds.) Transactions in GIS, 14(2), 111-129 (2010). Wiley Online Library, 2010. doi:10.1111/j.1467-9671.2010.01186.x

[Jena] Jena Semantic Web Framework, http://jena.sourceforge.net/

[Jencks11] Jencks, S., Williams, M., & Coleman, E.: Rehospitalizations among patients in the Medicare fee-for-service program. New Englland Journal of Medicine, 360, 1418- 1428 (2011). http://www.nejm.org/doi/full/10.1056/NEJMsa0803563

[Jurgens06] Jurgens, C. Y., Fain, J. A., Riegel, B.: Psychometric testing of the heart failure somatic awareness scale. Journal of cardiovascular nursing, 21(2), 95 (2006).

[Keßler09] Keßler, C., Raubal, M., Wosniok, C.: Semantic rules for context-aware geographical information retrieval. Smart Sensing and Context, 4th European Conf. on Smart Sensing and Context, Guildford, UK, 5741, pp.77-92, Sept. 16-18 (2009).

[Kuhn09] Kuhn, W.: A Functional Ontology of Observation and Measurement. 3rd Intl. Conf. on GeoSpatial Semantics, Mexico City, Mexico, Vol. 3, pp. 26-43, Dec. 3-4 (2009).

[Lefort11] Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Garcia-Castro, R., et al.: Semantic Sensor Network XG Final Report, W3C Incubator Group Report (2011). http://www.w3.org/2005/Incubator/ssn/XGR-ssn/

[Locke1690] Locke, J.: An Essay Concerning Human Understanding. (L. A. Selby-Bigge, Ed.) Philosophy and Phenomenological Research, Vol. 68, p. 496 (1960). Orion Publishing Group, Ltd.

[Manola04] Manola, F., & Miller, E.: RDF Primer. (Frank Manola & Eric Miller, Eds.) W3C Recommendation, Vol. 10, pp. 1-107 92004). W3C.

[McGlothlin10] McGlothlin, J.P., Khan, L.: Materializing and Persisting Inferred and Uncertain Knowledge in RDF Datasets. 24th AAAI Conf. on Artificial Intelligence, Atlanta, GA, USA, July 11-15 (2010).

[MesoWest] MesoWest, http://www.met.utah.edu/mesowest/

[Mitchell97] Mitchell, T. M.: Machine Learning. (J. F. Traub, B. J. Grosz, B. W. Lampson, & N. J. Nilsson, Eds.) Annual Review of Computer Science, Vol. 4, pp. 417-433 (1997). McGraw-Hill. doi:10.1145/242224.242229

[Mitre08] Mitre: Data Analysis Challenges. (2008). Retrieved from http://www.fas.org/irp/agency/dod/jason/data.pdf

[Motik12] Motik, B., Horrocks, I., Kim, S.: Delta-Reasoner: a Semantic Web Reasoner for an Intelligent Mobile Platform. 21st International World Wide Web Conference (WWW2012), Lyon, France, April 16-20 (2012).

[Neisser76] Neisser, U.: Cognition and Reality. Psychology, vol.218, San Francisco: W.H. Freeman and Company (1976).

[Nevatia82] Nevatia, R.: Machine Perception. Perception, 10 (1982). Prentice-Hall.

[NOAA] National Oceanic and Atmospheric Administrationís (NOAA) National Weather Service, Glossary, http://www.nws.noaa.gov/glossary/

[Nokia08] Nokia: Sensing the World with Mobile Devices: The Vision. Nokia Technology Insight Series, Nokia Research Center (NRC), December (2008). Retrieved from http://research.nokia.com/files/insight/NTI_Sensing_-_Dec_2008.pdf

[Norwich91] Norwich, K.: On the fundamental nature of perception. Acta Biotheoretica, 39(1), 81-90 (1991).

[O&M] Observations and Measurements (O&M), http://www.opengeospatial.org/standards/om

[OWL] Web Ontology Language (OWL), http://www.w3.org/TR/owl-ref/

[OWLTime] Time Ontology in OWL (OWL-Time), http://www.w3.org/TR/owl-time/

[Pantelopoulos10] Pantelopoulos, A., & Bourbaki, N. G.: A survey on wearable sensor-based systems for health monitoring and prognosis. Trans. Sys. Man Cyber Part C, 40 (1), 1-12 (2010). http://dx.doi.org/10.1109/TSMCC.2009.2032660

[Patni10] Patni, H., Henson, C., & Sheth, A.: Linked Sensor Data. 2010 International Symposium on Collaborative Technologies and Systems, 362-370 (2010). doi:10.1109/CTS.2010.5478492

[Pawlowski08] Pawlowski, A., Guzman, J. L., Rodriguez, F., Berenguel, M., Sanchez, J., & Dormido, S.: Event-based control and wireless sensor network for greenhouse diurnal temperature control: A simulated case study. 2008 IEEE International Conference on Emerging Technologies and Factory Automation, 500-507 (2008). doi:10.1109/ETFA.2008.4638446

[Pawlowski09] Pawlowski, A., Guzman, J. L., Rodriguez, F., Berenguel, M., Sanchez, J., & Dormido, S. (2009). The influence of event-based sampling techniques on data transmission and control performance. Emerging Technologies Factory Automation. IEEE Conference on ETFA, pp.1-8 (2009). doi:10.1109/ETFA.2009.5347045

[Peraldi09] Peraldi, S.E., Kaya, A., Möller, R.: Formalizing multimedia interpretation based on abduction over description logic aboxes. 22nd Intl. Workshop on Description Logics, Oxford, UK, July 27-30 (2009).

[Perera12] Perera, S., Henson, C., Thirunarayan, K., Sheth, A., & Nair, S.: Data driven knowledge acquisition method for domain knowledge enrichment in healthcare. IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2012), pp.1-8, October (2012).

[Pfisterer11] Pfisterer, D., et al.: SPITFIRE: toward a semantic web of things. IEEE Communications Magazine, 49(11), pp.40-48 (2011).

[PPACA10] The Patient Protection and Affordable Care Act, Pub. L. No. 111-148. 124 Stat. 110 (201). http://www.gpo.gov/fdsys/pkg/PLAW-111publ148/content-detail.html

[Preuveneers08] Preuveneers, D., Berbers, Y.: Encoding Semantic Awareness in Resource-Constrained Devices. IEEE Intelligent Systems, 23(2), pp.26-33, March (2008).

[Prud'hommeaux08] Prud'hommeaux, E., & Seaborne, A.: SPARQL Query Language for RDF. (E. Prud'hommeaux & A. Seaborne, Eds.)W3C Recommendation (2008).

[Pschorr10] Pschorr, J., Henson, C., Patni, H., & Sheth, A.: Sensor Discovery on Linked Data. Kno.e.sis Center Technical Report (2010).

[Punuru07] Punuru, J. R.: Knowledge-based Methods for Automatic Extraction of Domain-specific Ontologies. Louisiana State University (2007).

[QS12a] Quantified Self, http://quantifiedself.com/ (Accessed: May 31, 2013)

[QS12b] Sensors - Quantified Self, http://quantifiedself.com/sensors/ (Accessed: May 31, 2013)

[RDF] Resource Description Framework (RDF), http://www.w3.org/TR/rdfconcepts/

[RDFS] RDF Schema (RDF-S), http://www.w3.org/TR/rdf-schema/

[Reggia87] Reggia, J.A., Peng, Y.: Modeling Diagnostic Reasoning: A Summary of Parsimonious Covering Theory. Computer Methods and Programs Biomedicine, 25, pp.125-34 (1987).

[Ricquebourg07] Ricquebourg, V., Durand, D., Menga, D., Marhic, B., Delahoche, L., Loge, C., & Jolly-Desodt, A. M.: Context Inferring in the Smart Home: An SWRL Approach. International Conference on Advanced Information Networking and Applications Workshops, Vol. 2, pp. 290-295 (2007).

[Salter08] J. Salter: Heavy Rains Close Missouri Roads, Renew Flood Concerns. USA Today, 10 April (2008); www.usatoday.com/weather/storms/2008-04-10-missouri-flooding_N.htm

[Scheider10] Scheider, S., Probst, F., Janowicz, K.: Constructing Bodies and their Qualities from Observations. 6th Intl. Conf. on Formal Ontology in Information Systems, Toronto, Canada, May 11-14 (2010).

[Seitz11] Seitz, C., Schönfelder, R.: Rule-based OWL reasoning for specific embedded devices. 10th Intl. Semantic Web Conf., Bonn, Germany, Oct. 23-27 (2011).

[SensoryPerception10] Sensory Perception. Gene Ontology. (2010). http://www.geneontology.org/GO.doc.sensory-perception.shtml

[Shadbolt08] Shadbolt, N., Berners-Lee, T.: Web Science: Studying the Internet to Protect Our Future. Scientific American, September (2008). http://www.sciam.com/article.cfm?id=web-science

[Shanahan05] Shanahan, M.P.: Perception as Abduction: Turning Sensor Data into Meaningful Representation. Cognitive Science, 29, pp.103-134 (2005).

[Shannon48] Shannon, C.: A Mathematical Theory of Communication. MD computing computers in medical practice, 27(4), 306-17 (1948). ACM. doi:10.1145/584091.584093

[Sheth06] Sheth, A., Agrawal, S., Lathem, J., Oldham, N., Wingate, H., Yadav, P., and Gallagher, K.: Active Semantic Electronic Medical Record. Proceedings of the 5th International Semantic Web Conference, pp. 913-926, Athens, GA, November 6-9 (2006).

[Sheth08] Sheth, A., Henson, C., Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing, 12(4), pp.78-83, July/Aug (2008).

[Sheth09] Sheth, A.: Citizen Sensing, Social Signals, and Enriching Human Experience. IEEE Internet Computing, 13(4), 87-92 (2009). doi:10.1109/MIC.2009.77

[Sheth11] Sheth, A.: Semantics Scales Up: Beyond Search in Web 3.0. IEEE Internet Computing, November/December, 3-6 (2011). http://www.computer.org/csdl/mags/ic/2011/06/mic2011060003-abs.html

[SOS] Sensor Observation Service, http://www.opengeospatial.org/standards/sos

[SPARQL] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparqlquery/

[SSN-XG] W3C Semantic Sensor Network Incubator Group (SSN-XG) Charter. http://www.w3.org/2005/Incubator/ssn/charter.

[Stasch09] Stasch, C., Janowicz, K., Bröring, A, Reis, I., & Kuhn, W.: A Stimulus-Centric Algebraic Approach to Sensors and Observations. (N. Trigoni, A. Markham, & S. Nawaz, Eds.) GeoSensor Networks, 5659, 169-179 (2009). Springer-Verlag. doi:10.1007/978-3-642-02903-5_17

[Suchanek09] Suchanek, F. M.: Automated Construction and Growth of a Large Ontology. Saarland University (2009).

[Suh11] Suh, M., Chen, C., Woodbridge, J., Kai Tu, M., In Kim, J., Nahapetian, A., Evangelista, L., and Sarrafzadeh. M.: A Remote Patient Monitoring System for Congestive Heart Failure. J. Med. Syst. 35, 5, 1165-1179, October (2011),. DOI=10.1007/s10916-011-9733-y http://dx.doi.org/10.1007/s10916-011-9733-y

[SWActivity] W3C Semantic Web Activity, http://www.w3.org/2001/sw/

[Taylor11] Taylor, K., Leidinger, L.: Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks. 8th Extended Semantic Web Conf., Heraklion, Greece, May 29 – June 2 (2011).

[Thirunarayan09a] Thirunarayan, K., Henson, C., Sheth, A.: Situation Awareness via Abductive Reasoning from Semantic Sensor Data: A Preliminary Report. International Symposium on Collaborative Technologies and Systems (CTS2009), Workshop on Collaborative Trusted Sensing, Baltimore, Maryland (2009).

[Thirunarayan09b] Thirunarayan, K., Pschorr, J.: Semantic Information and Sensor Networks. In: Proceedings of the 24th Annual ACM Symposium on Applied Computing (ACM SAC 2009), March (2009).

[Tobies01] Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. Thesis, RWTH Aachen, Germany (2001).

[Thomas08] Thomas, C., Mehra, P., Brooks, R., & Sheth, A.: Growing Fields of Interest. 2008 IEEE International Conference on Web Intelligence and Intelligent Agent Technology, 2(1), 496-502 (2008). doi:10.1109/WIIAT.2008.358

[XLink] XML Linking Language (XLink), http://www.w3.org/TR/xlink/

[Zhao03] Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys, 35(4), 399-458 (2003). ACM. doi:10.1145/954339.95434