An Efficient Bit Vector Approach to Semantics-based Machine Perception in Resource-Constrained Devices

Cory Henson, Krishnaprasad Thirunarayan, Amit Sheth

Ohio Center of Excellence in Knowledge-enabled Computing (Kno.e.sis)
Wright State University, Dayton, Ohio, USA
{cory, tkprasad, amit}@knoesis.org

Abstract. The 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 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 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.

Keywords: Machine Perception, Semantic Sensor Web, Sensor Data, Mobile Device, Resource-Constrained Environments

1 Introduction

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.

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

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 [1-2] – machines continue to struggle with the task. *The primary challenge of machine perception is to define efficient computational methods to derive high-level knowledge 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.

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. The W3C Semantic Sensor Network Incubator Group [3] has recently developed the Semantic Sensor Network (SSN) ontology [4-5] 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 [6-8]. Such work is leading to a realization of a Semantic Sensor Web [9].

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 our work in defining the semantics of machine perception. And 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* [10].

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 [2,11]. *Discrimination* is the task of deciding how to narrow down the multitude of explanations through further observation [1,2]. 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.

The applicability of our approach to machine perception is evaluated on a smart-phone mobile device, demonstrating dramatic improvements in both efficiency and scale. In this paper, we present three novel contributions towards efficient machine perception in resource-constrained environments:

1. Formal definition of two primary inference tasks, in OWL, that are generally applicable to machine perception – explanation and discrimination.
2. Efficient algorithms for these inference tasks, using bit vector operations.
3. Lifting and lowering mappings to enable the translation of knowledge between the high-level semantic representations and low-level bit-vector representations.

Section 2 discusses the application of the SSN ontology for representing sensor observations and a-priori environmental knowledge. Section 3 specifies explanation and discrimination, as an extension to the SSN ontology. The efficient bit vector algorithms, as well as the lifting and lowering mappings, are provided in Section 4. Our approach is evaluated in Section 5, followed by related work in Section 6, and conclusions in Section 7.

2 Semantic Sensor Network Ontology

The Semantic Sensor Network (SSN) ontology [4-5] was developed by the W3C Semantic Sensor Network Incubator Group [3] to serve the needs of the sensors community. This community is currently using it for improved management of sensor data on the Web, involving annotation, integration, publishing, and search [6-8]. The ontology defines concepts for representing sensors, sensor observations, and knowledge of the environment.

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 [1-2]. 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 1a 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. (Throughout the paper, *KB* will be used to refer to *environmental knowledgebase*). Figure 1b shows an example KB with concepts from cardiology.

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

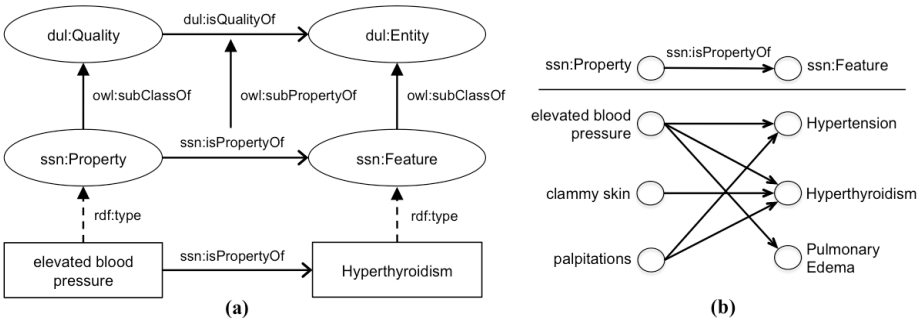

Figure 1. (a) Graphical representation of environmental knowledge in the SSN ontology, with mappings to DOLCE Ultra Lite (prefix *dul*). **(b)** Graphical representation of an example environmental knowledgebase in cardiology, taking the shape of a bipartite graph. This knowledgebase is derived from collaboration with cardiologists at ezDI (http://www.ezdi.us/).

3 Semantics of Machine Perception

Perception is the act of deriving high-level knowledge from low-level sensory observations [11]. The challenge of machine perception is to define computational methods to achieve this task efficiently. Towards the goal of providing a formal semantics of machine perception, we will define the primary components (inference tasks) of perception in OWL, as an extension of the SSN ontology. The two main components of perception are explanation and discrimination.

3.1 Semantics of Explanation

Explanation is the act of accounting for sensory observations; often referred to as hypothesis building [2,11]. 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 KB in Figure 1b, 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*) [11]. *Example*: the observed properties, elevated blood pressure and palpitations, are explained by the features Hypertension and Hyperthyroidism (discussed further below). While OWL has not been specifically designed for abductive inference, we will demonstrate that 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.

ObservedProperty: 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):

DEF 1: $\texttt{ObservedProperty} \equiv \exists \texttt{ssn:observedProperty}^{-}.\{o_1\} \sqcup \ldots \sqcup \exists \texttt{ssn:observedProperty}^{-}.\{o_n\}$

ExplanatoryFeature: 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:

DEF 2: $\texttt{ExplanatoryFeature} \equiv \exists \texttt{ssn:isPropertyOf}^{-}.\{p_1\} \sqcap \ldots \sqcap \exists \texttt{ssn:isPropertyOf}^{-}.\{p_n\}$

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: 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 `ExplanatoryFeature` class is constructed:

$\texttt{ExplanatoryFeature} \equiv \exists \texttt{ssn:isPropertyOf}^{-}.\{\texttt{elevated blood pressure}\} \sqcap \exists \texttt{ssn:isPropertyOf}^{-}.\{\texttt{palpitations}\}$

Given the KB in Figure 1b, 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 [12], *with the single-feature assumption*[3] [13-14]. The Description Logic expressivity of the explanation task is ALCOI[4,5], with ExpTime-complete complexity [15].

[3] Single-feature assumption specifies that an explanatory feature is a single individual.
[4] Using DL constructs: $\sqcap$, $\sqcup$, $\exists$, {a}, R^{-}
[5] http://www.cs.man.ac.uk/~ezolin/dl/

3.2 Semantics of Discrimination

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 [1,2,16]. 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 KB in Figure 1b, 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 [17]. 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*.

ExpectedProperty: 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:

DEF 3: $\texttt{ExpectedProperty} \equiv \exists \texttt{ssn:isPropertyOf}.\{f_1\} \sqcap \ldots \sqcap \exists \texttt{ssn:isPropertyOf}.\{f_n\}$

NotApplicableProperty: 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:

DEF 4: $\texttt{NotApplicableProperty} \equiv \neg\exists \texttt{ssn:isPropertyOf}.\{f_1\} \sqcap \ldots \sqcap \neg\exists \texttt{ssn:isPropertyOf}.\{f_n\}$

DiscriminatingProperty: 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.

DEF 5: 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 (Section 3.1), the following classes are constructed:

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

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

Given the KB in Figure 1b, 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[6], with PSpace-complete complexity [15].

4 Efficient Bit Vector Algorithms for Machine Perception

To enable their use on resource-constrained devices, we now describe algorithms for efficient inference of explanation and discrimination. These algorithms use bit vector encodings and operations, leveraging a-priori knowledge of the environment. 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 1 summarizes the data structures used by our algorithms.

Table 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|

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

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

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.

	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 2. (a) Example environmental knowledgebase in the domain of cardiology, from Figure 1b, represented as a bit matrix. Index tables are used for lifting and lowering environmental knowledge between a semantic representation and bit vector representation. **(b)** Index table for properties. **(c)** Index table for features.

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 2a shows an example KB, from Figure 1b, which has been 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 2b and 2c show example index tables for properties and 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

Algorithm 1: Explanation

```
[1] input OBSV_BV
[2] define BitVector EXPL_BV
[3] for each j := 0 … |ssn:Feature|-1
[4]     EXPL_BV[j] := 1
[5] for each i := 0 … |ssn:Property|-1
[6]     if OBSV_BV[i] = 1 then
[7]         EXPL_BV := EXPL_BV AND (row i in KB_BM)
[8] output EXPL_BV
```

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. 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 (as seen in Section 4.1).

We will now sketch the correctness of the explanation algorithm w.r.t. the OWL specification (Section 3.1). For each index position in $EXPL_{BV}$ that is set to 1, the corresponding feature explains all the observed properties. (*See note about indices*[7]).

Theorem 1: Given an environmental knowledgebase KB, and it's encoding as described in Section 4.1 (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(`$p_{k1}$`)` ⊓ … ⊓ `ObservedProperty(`$p_{km}$`)`, is explained by the feature f_e, implies `ExplanatoryFeature(`f_e`)`.

S2: The Hoare triple[8] holds: { ∀i ∈ {1, …, 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` ≡ ∃`ssn:isPropertyOf`⁻.{p_{k1}} ⊓ … ⊓ ∃`ssn:isPropertyOf`⁻.{p_{km}}, and (iv) the fact that `ExplanatoryFeature(`f_e`)` is provable, it follows that ∀i ∈ {1, …, m}: `ssn:isPropertyOf(`p_{ki}`,`f_e`)` is in KB. By our encoding, ∀i ∈ {1, …, m}: KB_{BM}[ki][e] = 1. Using lines 5-7, the fact that $EXPL_{BV}$[e] is initialized

[7] 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.)

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

to 1 and is updated only for i ∈ {1, …, 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).

(*S2* ⇒ *S1*): Given that {∀i ∈ {1, …, m}: $OBSV_{BV}[ki] = 1$} and {$EXPL_{BV}[e] = 1$} (pre and post conditions), it follows that ∀i ∈ {1, …, m}: $KB_{BM}[ki][e] = 1$ must hold. According to our encoding, this requires that ∀i ∈ {1, …, 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}, …, 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

Algorithm 2: Discrimination

```
[1]  input EXPL_BV, OBSV_BV
[2]  define BitVector DISC_BV
[3]  for each i := 0 … |ssn:Property|-1
[4]     DISC_BV[i] := 0
[5]  define BitVector ZERO_BV
[6]  for each j := 0 … |ssn:Feature|-1
[7]     ZERO_BV[j] := 0
[8]  for each i := 0 … |OBSV_BV|-1
[9]     if OBSV_BV[i] = 0 then
[10]       BitVector PEXPL_BV :=
[11]          EXPL_BV AND (row i in KB_BM)
[12]       if PEXPL_BV != EXPL_BV and
[13]          PEXPL_BV != ZERO_BV then
[14]          DISC_BV[i] := 1
[15] output DISC_BV
```

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. The discriminating properties are those that are determined to be neither expected nor not-applicable.

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 ki^{th} property is expected; (ii) $PEXPL_{BV} = ZERO_{BV}$ holds and the ki^{th} property is not-applicable; or (iii) the ki^{th} 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 (Section 3.2). 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}, …, p_{km}}, i.e., `ObservedProperty(`p_{k1}`)` ⊓ … ⊓ `ObservedProperty(`p_{km}`)`, are explained by *n* features {f_{e1}, …, f_{en}}, i.e., `ExplanatoryFeature(`f_{e1}`)` ⊓ … ⊓ `ExplanatoryFeature(`f_{en}`)`, then the following holds: ∀i: 1 ≤ i ≤ m: `ObservedProperty(`p_{ki}`)` ⇒ `ExpectedProperty(`p_{ki}`)`.

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

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., $OBSV_{BV}[ki] = 0$, $\texttt{ExpectedProperty}(p_{ki})$ 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 $EXPL_{BV}$ $\{f_e \mid EXPL_{BV}[e] = 1\}$, $\texttt{NotApplicableProperty}(p_{ki})$ iff $\forall e: (EXPL_{BV}[e] \wedge KB_{BM}[ki][e]) = ZERO_{BV}[e]$.

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

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.*

5 Evaluation

To evaluate our approach, we compare two implementations of the explanation and discrimination inference tasks. The first utilizes an OWL reasoner as described in Section 3, and the second utilizes the bit vector algorithms described in Section 4. Both implementations are coded in Java, compiled to a Dalvik[9] executable, and run

[9] http://code.google.com/p/dalvik/

on a Dalvik virtual machine within Google's Android[10] operating system for mobile devices. The OWL implementation uses Androjena[11], a port of the Jena Semantic Web Framework for Android OS. The mobile device used during the evaluation is a Samsung Infuse[12], 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 KB 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 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 KB form a *complete bi-partite graph*[13]. 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 3.

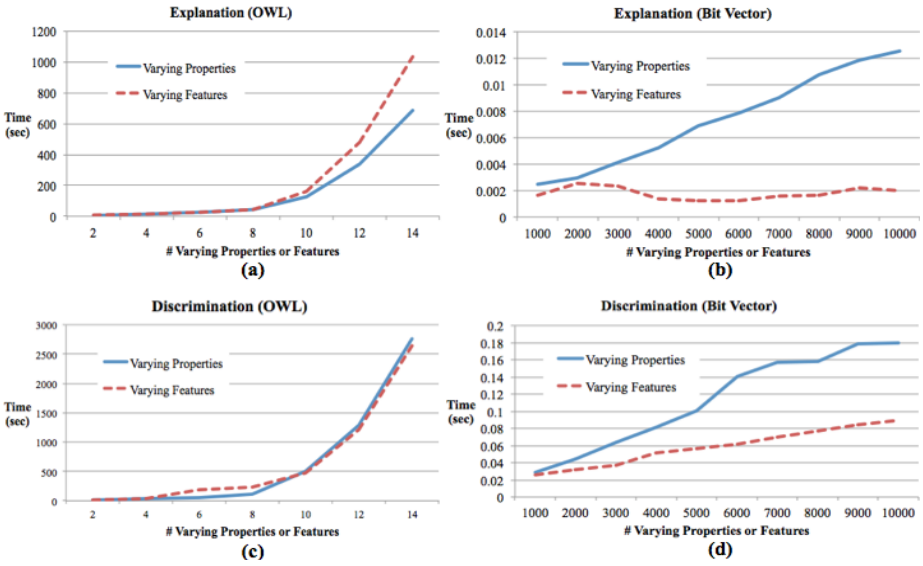

Figure 3. Evaluation results: **(a)** Explanation (OWL) with $O(n^3)$ growth, **(b)** Explanation (bit vector) with O(n) growth, **(c)** Discrimination (OWL) with $O(n^3)$ growth, and **(d)** Discrimination (bit vector) with O(n) growth.

[10] http://www.android.com/
[11] http://code.google.com/p/androjena/
[12] http://www.samsung.com/us/mobile/cell-phones/SGH-I997ZKAATT
[13] 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 3a and 3c, respectively. With a KB 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 3b and 3d, respectively. With a KB 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 KB (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 KB (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 KB should be of adequate size and scope. In practice, an application may not require a KB 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.

6 Related Work

The ability to derive high-level knowledge from low-level observation data is a challenging task. As argued in this paper, a promising approach to machine perception involves the use of Semantic Web technologies. This approach is quickly evolving into an active area of research. Our work differs from related efforts in three ways: (1) the use of OWL for defining the perception inference tasks, (2) the definition of perception as an abductive process, and (3) the efficient execution of the inference tasks using bit vector operations.

Previous works have utilized OWL for representing concepts in the domain of sensing [4,5,18,19]. Subsequently, First-Order Logic (FOL) rules were often employed to derive knowledge of the features in the environment [20-22]. Taylor et al. [23] have used Complex Event Processing to derive knowledge of events from observation data encoded in SSN. However, 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.

Second, as opposed to approaches using deductive (FOL) rules, we believe that perception is an abductive process [11]. The integration of OWL with abductive reasoning has been explored [24]; requiring modification of OWL syntax and/or inference engine [25]. We demonstrated that, under the single-feature assumption, abductive consequences can be computed using standard OWL reasoners.

And third, while OWL is decidable, the computational complexity still limits its practical use within resource-constrained environments. A recent W3C Member Submission [26] 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 [10,27,28,29]. Preuveneers et al. [28] have presented a compact ontology encoding scheme using prime numbers that is capable of class-subsumption. Ali et al. [10] 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. [30] serialize RDF datasets and materialize data inferred through OWL reasoning using bit vectors. For our inference tasks, 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.

7 Conclusions and Future Work

We have demonstrated an approach to machine perception on resource-constrained devices that is simple, effective, and scalable. In particular, we presented three novel contributions: (1) a simple declarative specification (in OWL) of two inference tasks useful for machine perception, explanation and discrimination; (2) efficient algorithms for these inference tasks, using bit vector operations; and (3) lifting and

lowering mappings to enable the translation of knowledge between semantic representations and the bit vector representations.

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). Currently, we are collaborating with cardiologists to develop a mobile app 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.

In the future, we plan to investigate more expressive approaches to 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, we plan to extend our approach to stream reasoning by incorporating (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, such as ours, will be indispensable.

Acknowledgements. This research was supported in part by US NSF award no. 1143717 (III: EAGER — Expressive Scalable Querying over Integrated Linked Open Data). We also thank Michael Cooney for help with implementation and evaluation.

References

1. Neisser, U.: Cognition and Reality. Psychology, 218, San Francisco: W.H. Freeman and Company (1976).
2. Gregory, R.L.: Knowledge in perception and illusion. In: Philosophical Transactions of the Royal Society of London, 352(1358), pp.1121-1127 (1997).
3. W3C Semantic Sensor Network Incubator Group (SSN-XG) Charter. http://www.w3.org/2005/Incubator/ssn/charter.
4. Lefort L., et al.: Semantic Sensor Network XG Final Report. W3C Incubator Group Report, June 28 (2011). www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628.
5. Compton, M. et al.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics (2012) (in press).
6. Gray, A.J.G., 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).
7. 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).
8. Pfisterer, D., et al.: SPITFIRE: toward a semantic web of things. IEEE Communications Magazine, 49(11), pp.40-48 (2011).
9. Sheth, A., Henson, C., Sahoo, S.: Semantic Sensor Web. IEEE Internet Computing, 12(4), pp.78-83, July/Aug (2008).

10. 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).
11. Shanahan, M.P.: Perception as Abduction: Turning Sensor Data into Meaningful Representation. Cognitive Science, 29, pp.103-134 (2005).
12. Reggia, J.A., Peng, Y.: Modeling Diagnostic Reasoning: A Summary of Parsimonious Covering Theory. Computer Methods and Programs Biomedicine, 25, pp.125-34 (1987).
13. 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).
14. Henson, C., Sheth, A., Thirunarayan, K.: Semantic Perception: Converting Sensory Observations to Abstractions. IEEE Internet Computing, 16(2), pp.26-34, Mar/Apr (2012).
15. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. Thesis, RWTH Aachen, Germany (2001).
16. Bajcsy, R.: Active perception. IEEE, 76(8), pp.996-1005 (1988).
17. 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).
18. Kuhn, W.: A Functional Ontology of Observation and Measurement. 3rd Intl. Conf. on GeoSpatial Semantics, Mexico City, Mexico, Dec. 3-4 (2009).
19. Devaraju, A., Kuhn, W.: A Process-Centric Ontological Approach for Integrating Geo-Sensor Data. 6th Intl. Conf. on Formal Ontology in Information Systems, Toronto, Canada, May 11-14 (2010).
20. 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, Sept. 16-18 (2009).
21. 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).
22. 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) (in press).
23. 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).
24. 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).
25. 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).
26. Binary RDF Representation for Publication and Exchange (HDT). W3C Member Submission (2011). http://www.w3.org/Submission/2011/SUBM-HDT-20110330/.
27. Seitz, C., Schönfelder, R.: Rule-based OWL reasoning for specific embedded devices. 10th Intl. Semantic Web Conf., Bonn, Germany, Oct. 23-27 (2011).
28. Preuveneers, D., Berbers, Y.: Encoding Semantic Awareness in Resource-Constrained Devices. IEEE Intelligent Systems, 23(2), pp.26-33, March (2008).
29. 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).
30. 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).