Expert Systems with Applications 39 (2012) 9494–9503

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An approximation to the computational theory of perceptions using ontologies

Carmen Martínez-Cruz [a,*], Albert van der Heide [b], Daniel Sánchez [b,c], Gracian Trivino [b]

[a] Dept. of Computing, University of Jaén, Campus de Las Lagunillas, SN, 21073 Jaén, Spain
[b] European Centre for Soft Computing, Edificio Científico-Tecnológico. 33600 Mieres, Asturias, Spain
[c] Dept. of Computer Science and A.I., University of Granada, C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Ontology
Computing with Words and Perceptions
Granular Linguistic Model of a Phenomenon

ABSTRACT

New technologies allow users to access huge amounts of data about phenomena in their environment. Nevertheless, linguistic description of these available data requires that human experts interpret them highlighting the relevant details and hiding the irrelevant ones. Experts use their personal experience on the described phenomenon and in using the flexibility of natural language to create their reports. In the research line of Computing with Words and Perceptions, this paper deals with the challenge of using ontologies to create a computational representation of the expert's knowledge including his/her experience on both the context of the analyzed phenomenon and his/her personal use of language in that specific context. The proposed representation takes as basis the Granular Linguistic Model of a Phenomenon previously proposed by two of the authors. Our approach is explained and demonstrated using a series of practical prototypes with increasing degree of complexity.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

Inspired in the way humans use natural language (NL) to describe phenomena in their environment, our aim is to develop computational systems in order to provide users with meaningful linguistic descriptions of data. In these applications, the computer takes the role of an assistant that uses the available experience to interpret the input data and to provide a specific user with relevant information about monitored phenomena in a concrete application context.

One important issue in this area is that of uncertainty in natural language, coming from different sources. In particular, humans use imprecise terms in everyday communication, including data description. Fuzzy Logic has been shown to be a suitable tool for representing imprecision when creating computational models of the meaning of linguistic terms. In recent years, the father of Fuzzy Logic, Lotfi Zadeh, has proposed a new direction in this research line, namely extending Fuzzy Logic towards Computing with Words and Perceptions (CWPs).

The field was introduced in the Zadeh's seminal paper "From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions" (Zadeh, 1999) and further developed in subsequent papers. CWP provides a framework to develop computational systems with the capacity of computing with the meaning of NL expressions, i.e., with the capacity of computing with imprecise descriptions of the world in a similar way that humans do it.

In CWP, a granule is a clump of elements which are drawn together by indistinguishability, similarity, proximity or functionality (Zadeh, 1979). The boundary of a granule is fuzzy. Fuzziness of granules allow us to model the way in which human concepts are formed, organized and manipulated in an environment of imprecision, uncertainty, and partial truth (Zadeh, 1997).

A granule underlies the concept of a linguistic variable (Zadeh, 2008). A linguistic variable is a variable whose values are words or sentences in NL (Zadeh, 1975). The idea of linguistic fuzzy quantifiers was introduced by Zadeh (1983).

In a recent paper, written by Zadeh with others researchers (Mendel, Lawry, & Zadeh, 2010), they distinguish between two levels of development of CWP:

- In Level 1, the carriers of information are numbers and words, e.g., "$X = 5$," "X is small", and "If X is small then Y is large". Objects of computation may be viewed as assignment statements which assign values to variables. Level 1 does not require any knowledge of the semantics of natural languages.
- In Level 2, the carriers of information are numbers, words and propositions or collections of propositions drawn from a natural language like "Most Swedes are tall", and "It is very unlikely that there will be a significant decrease in the price of oil in the near future". Level 2 does require some understanding of how the meaning of a proposition may be composed from the meanings of its constituents.

Therefore, the current challenge is to extend the existing paradigm with new capacities for modeling the meaning of complex linguistic expressions.

* Corresponding author.
E-mail address: cmcruz@ujaen.es (C. Martínez-Cruz).

The concept of fuzzy linguistic summary can be considered a contribution to this objective. The fuzzy linguistic summary has the general form (w, "Q objects in database satisfying D are S") where Q is called the quantifier, D is a fuzzy property characterizing the reference fuzzy set of objects, S is another fuzzy property of the objects called the summarizer, and $w \in [0,1]$ is the degree of validity of the linguistic clause for representing the meaning in the specific context (Kacprzyk, Yager, & Zadrozny, 2000; Yager, 1982), e.g., (0.7, "Usually the temperature during midday is warm").

This basic concept of linguistic summary has been studied and extended in different ways, see (Diaz-Hermida, Ramos, & Bugarin, 2011) for a recent review on the state of the research in this field.

One of the limitations of these approaches is that they represent perceptions directly related with data and they do not allow to construe second order perceptions, i.e., perceptions elaborated on first order perceptions. In order to solve this problem, in previous works, we proposed the idea of Granular Linguistic Model of Phenomena (GLMP). GLMP allows to manage more complex phenomena and therefore more complex linguistic descriptions, including how some linguistic statements explain others. For example, in Trivino et al. (2010), we used GLMP to generate linguistic descriptions of the behavior of traffic in a roundabout, in Mendez-Nunez and Trivino (2010), we generate linguistic reports of financial data and in Alvarez-Alvarez, Sanchez-Valdes, and Trivino (2011), we generated linguistic descriptions of the surface of Mars.

In this paper we considerably extend the possibilities of applying GLMP in the field of linguistic description of data. We implement the general concepts of our approach to CWP using a *high level ontology* and we generate NL sentences after instantiating *application ontologies*. Following Zadeh, our aim consists of using ontologies to provide "some understanding of how the meaning of a proposition may be composed from the meanings of its constituents".

The rest of the paper is organized as follows. Section 2 introduces two basic concepts that we use later, namely, Ontologies and Granular Linguistic Model of Phenomena. Section 3 introduces the contribution, describing the elements in the *high level ontology*, giving our methodology, and explaining its advantages. Sections 4–6 describe three prototypes of *application ontologies* of increasing complexity. Section 7 describes how to generate linguistic summaries using the *application ontologies* and includes an experimental layout, showing some results obtained when applying our third prototype to a real dataset. Finally, Section 8 contains our conclusions and points out some future research work.

2. Preliminaries

2.1. Ontologies for representing the experience of experts

According with our goal, one of the simplest visions of the Ontology concept in Computer Science is proposed by Agarwal (2005) who states that *an ontology is the manifestation of a shared understanding of a domain that is agreed between a number of agents and such agreement facilitates accurate and effective communications of meaning, which in turn leads to other benefits such as inter-operability, reuse, and sharing*. Other definitions are available in (Gómez-Pérez, Férnandez-López, & Corcho-García, 2003; Staab & Studer, 2004).

In the last decade, ontologies have been widely used as one of the preferred knowledge representation models especially because of its usability in the Semantic Web (Berners-Lee, Hendler, & Lassila, 2001). There are several criteria to classify ontologies that can be found in the literature, such as (Gómez-Pérez et al., 2003; Martinez-Cruz, Blanco, & Vila, 2011). One of these criteria addresses the semantics of the information represented and establishes three kinds of ontologies as follows:

- *Meta-ontologies* are those which establish the conceptual vocabulary for representing ontologies. The most common meta-ontology is therefore the one which defines the concept of classes, properties, axioms, etc.
- *High Level Ontologies* describe the more generic concepts or processes. They are not usually oriented to be directly instantiated, due to the fact that they represent relatively general concepts.
- *Representation Ontologies* are the remaining applicable ontologies: Domain ontologies, process ontologies, task ontologies, content ontologies, method ontologies, etc. For our purposes here, *application ontologies* are specially important, this category having several subclassifications in the literature.

In their initial inception ontologies represent the reality and its semantics in a computational way, with no place for uncertainty. Nevertheless, such statement was revised due to the fact that uncertainty is a common requirement in real world applications (Bobillo & Straccia, 2011). Thus, some approximations to represent uncertainty using ontologies have been faced in the literature extending ontology languages or frameworks to define fuzzy concepts and operations. For example, (Bobillo & Straccia, 2011; Bobillo, Delgado, & Gómez-Romero, 2012) extend the OWL2 standard to manage fuzzy data; (Calegari & Ciucci, 2010) presents a framework to represent constraints and perceptions using several ontologies that represent fuzzy concepts; the proposal by (Ghorbel, Bahri, & Bouaziz, 2009) extends the ontology management framework Protégé with capabilities to manage fuzzy data; and (Jiang, Tang, Chen, & Wang, 2011) considers reasoning in fuzzy modular ontologies. Several applications of fuzzy ontologies are described in Lee, Chen, and Jian (2003), Seo, Park, Lee, and Wang (2009), Lee and Wang (2007), where they are used for recognition of fuzzy events in summarization and fuzzy decision support.

2.2. Granular Linguistic Model of a Phenomenon

In this section, we introduce the components of the Granular Linguistic Model of a Phenomenon (GLMP), our approach based on CWP for developing computational systems able to generate linguistic descriptions of data (Mendez-Nunez & Trivino, 2010; Trivino et al., 2010).

2.2.1. Computational perception (CP)

A CP is the computational model of a unit of information acquired by the designer about the phenomenon to be modeled. In general, CPs correspond with specific parts of the phenomenon at certain degrees of granularity. A CP is a couple (A,W) where:

$A = (a_1, a_2, \ldots, a_n)$	is a vector of linguistic expressions (words or sentences in NL) that represents the whole linguistic domain of the CP. Each a_i describes the value of the CP in each situation with specific degree of granularity. These sentences can be either simple, e.g., a_i = *"The temperature is high"* or more complex, e.g., a_i = *"Some times the room could not be comfortable"*.
$W = (w_1, w_2, \ldots, w_n)$	is a vector of validity degrees $w_i \in [0, 1]$ assigned to each a_i in the specific context. The concept of validity depends on the application, e.g., it is a function of the truthfulness and relevance of each sentence in its context of use.

Each pair (a_i, w_i) is called a *computational perception item* (CPI).

2.2.2. Perception mapping (PM)

We use PMs to create and aggregate CPs. There are many types of *PMs* and this paper explores several of them. A PM is a tuple (U, y, g, T) where:

U is a vector of input CPs, $U = (u_1, u_2, \dots, u_n)$, where $u_i = (A_i, W_i)$. In the special case of first order Perception mappings (1PMs), these are the inputs to the GLMP and they are values $z \in R$ being provided either by a sensor or obtained from a database.

y is the output CP, $y = (A_y, W_y)$.

g is an aggregation function employed to calculate the vector of fuzzy degrees of validity assigned to each element in $y, W_y = (w_1, w_2, \dots, w_{n_y})$, as a fuzzy aggregation of input vectors, $W_y = g(W_{u_1}, W_{u_2}, \dots, W_{u_n})$, where W_{u_i} are the degrees of validity of the input perceptions. In Fuzzy Logic many different types of aggregation functions have been developed. For example g could be implemented using a set of fuzzy rules. In the case of 1PMs, g is built using a set of membership functions as follows:

$$W_y = (\mu_{a_1}(z), \mu_{a_2}(z), \dots, \mu_{a_{n_y}}(z)) = (w_1, w_2, \dots, w_{n_y})$$

where W_y is the vector of degrees of validity assigned to each a_y, and z is the input data.

T is a text generation algorithm that allows generating the sentences in A_y. In simple cases, T is a linguistic template, e.g., *"The temperature in the room is* {high |*medium*|*low*}".

2.2.3. Structure of the GLMP

The GLMP consists of a network of PMs. Each PM receives a set of input CPs and transmits upwards a CP. We say that each output CP is *explained* by the PM using a set of input CPs. In the network, each CP covers specific aspects of the phenomenon with certain degree of granularity. Fig. 1 shows an example of GLMP. In this example, the phenomenon can be described at a very basic level in terms of three variables providing values z_1, z_2, and z_3 respectively at a certain moment in time. Each value is in the corresponding domain of each variable. The most basic linguistic descriptions correspond to the computational perceptions CP1, CP2, and CP3, each one consisting of a certain number (not indicated in the figure) of computational perceptions items (CPIs), i.e., pairs (linguistic expression, validity). The correspondence between the z_i and each CPI at a certain moment is defined by the corresponding 1PM.

Using different aggregation functions and different linguistic expressions, the paradigm GLMP allows the designer to model computationally his/her perceptions. In the case of Fig. 1 other two, higher-level descriptions of the phenomenon are provided. These descriptions are given in the form of computational perceptions CP4 and CP5. The 2PMs PM4 and PM5 indicate that CP4 and CP5 can be explained in terms of CP1, CP2, and CP3, i.e., how the validity of each item in CP4 and CP5 is explained by those of CP1-CP3. Finally, the top-order description of the phenomenon is provided, at the highest level of abstraction, by CP6, explained by PM6 in terms of CP4 and CP5. Notice that, by using this structure, one can provide not only a linguistic description of the phenomenon at a certain level, but an explanation in terms of linguistic expressions at a lower level.

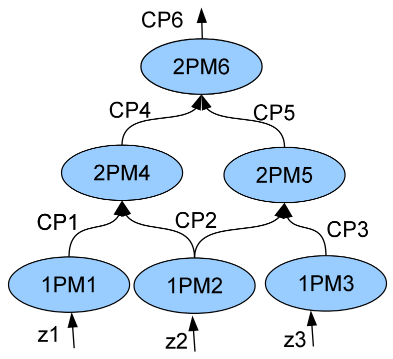

Fig. 1. Example of GLMP.

3. GLMP ontology

The basic idea of our proposal in this paper consists of implementing the concept of GLMP using ontologies. This representation has three levels:

1. A *high level ontology*, comprised of superclasses that cannot be instantiated, that represents the different types of components of the GLMP introduced in the previous section, namely, Inputs, CPs, CPIs, and PMs. This part is common to any GLMP ontology.
2. An *application ontology*, in which classes corresponding to the specific components of a concrete GLMP are defined as subclasses of the corresponding classes in the *high level ontology*. This part is designed when the GLMP is created, though part of it is generated by a population algorithm.
3. A collection of *instances* with which the ontology is populated when fed up with input data about the phenomenon, defining instances of CPIs from which the final linguistic description is obtained. Using instances of input data we obtain instances of 1CPs, and then instances of 2CPs following the links in the network. Each instance of CP will include instances of valid sentences to describe the phenomenon at different degrees of granularity. All of these data and CPs will be stored as part of the ontology. This level of the ontology is hence generated in running time when using the GLMP for obtaining a linguistic description of a specific phenomenon in an certain time period. Finally, a final report consisting of a selection of the CPIs and input data is produced following the specifications given by a query to the ontology.

The components of the *high level ontology* are shown in Fig. 2, and can be described as follows:

- *Input* class: is the superclass of all input classes, each one corresponding to one input to the GLMP. Each input class has two data type functional properties, *val*, that is the value of the variable, and *time*, a day/hour data type. These properties allow us to incorporate in the ontology the time series of values of the different input variables, as a collection of instances in each class, when using the ontology for generating a linguistic report.
- *Computational_Perception_Item* class: is a superclass for all subclasses representing CPIs of the form (a_i, w_i), with a variable *validity* representing the validity of the CPI, that is inherited by the subclasses. The subclass corresponding to the sentence a_i is annotated with this sentence, since this is a common, static property of all the instances of the class.
- *Computational_Perception* class, is the superclass for all computational perceptions classes. Every CP class is related via its corresponding CPI class to a set of CPIs.
- *Perception_Mapping* class: this is the superclass of all PM classes and has two subclasses, 1PM and 2PM, corresponding to first-order and second-order perception superclasses. The output, which is common for both 1PM and 2PM, is defined by the functional object property *produces*, and correspond to a CP. Regarding the input, which is different in both classes, the 1PM class defines the class of its input value, by means of the *get_input* functional property, from the *Input* class, while the 2PM class

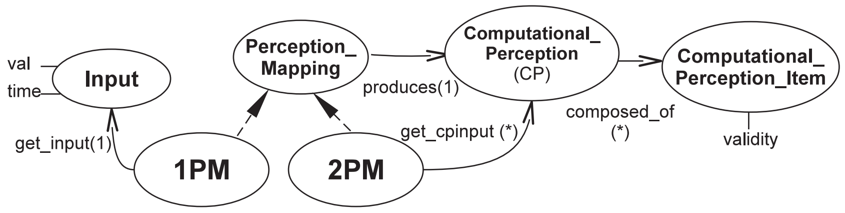

Fig. 2. High level ontology.

has as input a set of CPs from specific CP classes, as indicated by means of the *get_cpinput* property. Finally, PM is annotated with *g* and *T* functions, referring to a predefined collection of functions performing aggregation and text generation.

As a *high-level ontology*, none of these superclasses can be instantiated; instead, subclasses inheriting certain class properties can be added in order to obtain an *application ontology* (description of a specific GLMP). As we indicated before, the *high-level ontology* provides us with a way to give an easy methodology for generating application ontologies. A sketch of this methodology, applied to the example in Fig. 1, is the following:

1. Generate one subclass of the class *Input* for each input variable in the GLMP (three in this case).
2. Generate one subclass of *Computational_Perception_Item* and one subclass of *Computational_Perception* for each CP (six in Fig. 1). Then, for each possible sentence defining a CPI for a given computational perception CP, generate one subclass of the corresponding CPI superclass. For example, if CP_6 in the GLMP in Fig. 1 had associated four possible output sentences (not indicated in the figure, just an hypothesis), then we have to define one subclass CP6 of *Computational_Perception*, another subclass CPI6 of *Computational_Perception_Item*, and four subclasses of CPI6, each annotated with one sentence. The generation of these subclasses may be done either manually or by an initial population process based on the function *T* for text generation.
3. Constraint some object properties to their specific ranges (subclasses) using the Universal restriction (only). These constraints must be declared in the *produces*, *get-input*, *get-cpinput* and *composed_of* properties.
4. Constraint the CP class property *composed_of* to all the subclasses of its corresponding CPI, setting the cardinality constraint to one. This means that every instance of a CP is associated to a single instance of each of the subclasses, that is, to a single pair (a_i, w_i) for each a_i.

In order to better illustrate the definition of GLMPs via ontologies, we show in the following sections examples of this methodology for developing three prototypes of increasing complexity. The corresponding *application ontologies* define three computational systems using a set of classes, properties, constraints and instances. In prototype 1, we introduce a very simple computational application based on a single 1CP. We show how to generate linguistic descriptions of data using this basic schema. Then, in prototype 2, we introduce 2CPs to describe more complex perceptions, i.e., we show how to explain a perception by means of a "explanatory" structure based on more basic perceptions. Finally, in prototype 3, we use the concept of fuzzy summarizer to introduce a second type of 2CP that allows to aggregate the values of a time series of CPs during a period of time. This corresponds specifically to Level 2 of applications in CWP, as explained in the introduction.

Finally, let us remark that the approach outlined in this section offers several advantages, at least:

- All the information, including general concepts about GLMPs, data, and particular elements of the GLMP for the specific application, are represented using a single, well known scheme, an ontology that integrates and relates all the information.
- General concepts related to linguistic description via GLMPs are identified and described via the *high-level ontology*, improving the understandability of the approach. In addition, they provide a very good basis for guiding the process of building a GLMP for a specific application, by means of instantiation/particularization of the general GLMP.
- The language of ontology definition provides us with an implementation language for GLMPs in which there is no necessity for coding a program for each model. Only a general program valid for any model, and the description of the GLMP and its elements, are necessary.
- The generation of a linguistic description for a specific data set can be implemented as a process of population of the ontology using a general procedure, valid for any GLMP.
- The representation using ontologies is a way to integrate other elements related to the application domain, like typical linguistic expressions, and other aspects of the context, with the GLMP. For instance, we can choose the sentences that will be provided depending on the kind of user. For this purpose, we are working on representing the domain of the application and the context using ontologies, that will be easily integrated with the GLMP ontologies.

4. Prototype 1

This prototype is a very simple example of a computational system for generating linguistic descriptions of data. It uses the values provided by a temperature sensor to generate sentences describing this value, i.e., in this basic example the computational system receives as input a real number and it produces the following three sentences: *"The temperature is cold"*, *"The temperature is warm"*,

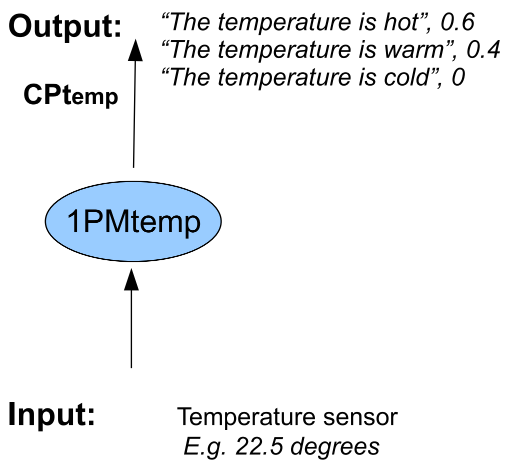
Output:
"The temperature is hot", 0.6
"The temperature is warm", 0.4
"The temperature is cold", 0
CPtemp
1PMtemp
Input:
Temperature sensor
E.g. 22.5 degrees

Fig. 3. GLMP for prototype 1 including several instances of generated sentences.

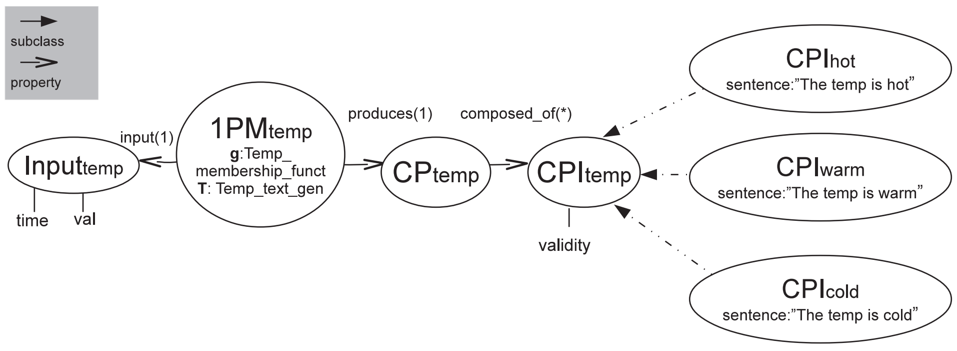

Fig. 4. Scheme of the application ontology for prototype 1.

Table 1
Some instances obtained when populating the *application ontology* for prototype 1 with the input value 22.5.

Instance name	Class	Values	Description
I1	$Input_{Temp}$	22.5	Value got from a temperature sensor
CPI1	CPI_{hot}	0.6	Computational perception for a temperature of 22.5 °C and the *"The temperature is hot"* sentence
CPI2	CPI_{warm}	0.4	Computational perception for a temperature of 22.5 °C and the *"The temperature is warm"* sentence
CPI3	CPI_{cold}	0	Computational perception for a temperature of 22.5 °C and the *"The temperature is cold"* sentence
CP1	$1CP_{temp}$	Composed_of: CPI1, CPI2, CPI3	Computational perception got for a temperature of 22.5 °C
1PM	$1PM_{temp}$	Input:I1 (22.5 °C), Produces: CP1	Perception mapping that relates each input with the output (CP)

and *"The temperature is hot"* with their associated degrees of validity. The structure of a GLMP for this problem is shown in Fig. 3.

This computational system has a unique 1PM that we call $1PM_{temp} = (z, CP_{temp}, g, T)$ where:

z is the input given by the sensor.
CP_{Temp} is the output *CP*.
g is implemented using a strong fuzzy partition of trapezoidal membership functions, i.e., it returns the three membership values corresponding to z.
T provides three possible sentences: *"The temperature is cold"*, *"The temperature is warm"*, *"The temperature is hot"*.

Fig. 4 shows the *Prototype 1 Application Ontology*, with the following elements:

The $Input_{Temp}$ is the class of the measures given by the temperature sensor. This class includes two properties, called *val* and *time*, that represent the temperature value and the daytime respectively.
CPI_{Temp} class (computational perception items for temperature class) is the superclass of all the CPIs for temperature. The subclasses of this class corresponding to each of the three CPI for Temperature are annotated in the *sentence* annotation label in each one of three subclasses:

- CPI_{Hot} class: *"The temperature is hot"*.
- CPI_{Warm} class: *"The temperature is warm"*.
- CPI_{Cold} class: *"The temperature is cold"*.

The CP_{Temp} class corresponds to the CP related with the temperature. The object property *composed_of* is constrained in this class in two senses: by setting the cardinality value to 1 per each subclass of CPI_{Temp} and establishing the only (universal) restriction to the CPI_{Temp} class.[1]

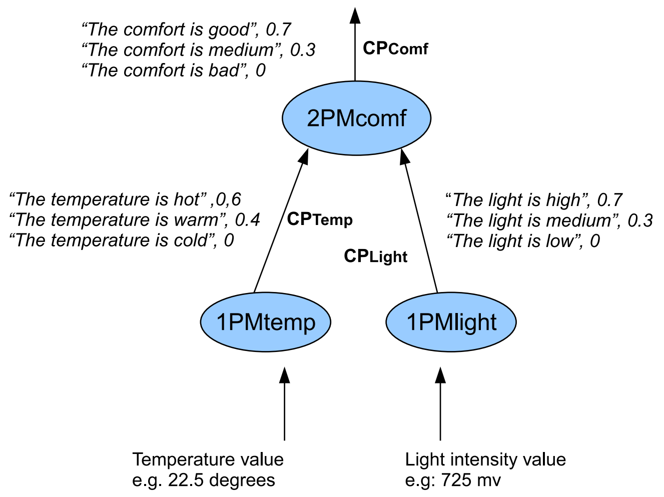

Fig. 5. GLMP for prototype 2 including several instances of generated sentences.

The $1PM_{Temp}$ class represents the 1PM that produces computational perceptions of temperature. $1PM_{temp}$ class includes information about the algorithms associated with it using two annotations labels:

- g is called *Temperature_membership_function*. It generates the validity degree of each linguistic expression.
- T generates the set of sentences included as annotations of CPI_{Temp} subclasses. This algorithm is called *Temperature_text_generator*.

The **population** of this *application ontology* consists of two phases:

- *Initialization*: The definition of the problem involves the definition of a set of possible sentences. This phase is executed once at the beginning of the process and it is made either by hand by the designer or by the annotated algorithm *Temperature_text_generator*. After this algorithm is executed the subclasses of CPI_{Temp} are annotated with the sentence property.

[1] These restrictions are not shown in Fig. 4 for simplicity, but are specified using OWL2 when implementing the ontology. We will use this criterion in the rest of our graphical representations of ontologies along this paper.

Table 2
Set of fuzzy rules employed by $2PM_{comf}$ in prototype 2.

Rule	Description
R1	IF temp IS cold AND light IS low THEN comfort IS bad
R2	IF temp IS warm AND light IS low THEN comfort IS medium
R3	IF temp IS hot AND light IS low THEN comfort is bad
R4	IF temp IS cold AND light IS medium THEN comfort IS medium
R5	IF temp IS warm AND light IS medium THEN comfort IS good
R6	IF temp IS hot AND light IS medium THEN comfort is good
R7	IF temp IS cold AND light IS high THEN comfort IS medium
R8	IF temp IS warm AND light IS high THEN comfort IS good
R9	IF temp IS hot AND light IS high THEN comfort is medium

- *Generation of computational perceptions*: The population of the following classes is performed in this phase: Input, $1PM_{Temp}$, CP_{Temp}, and CPIs. Each time a temperature is given or taken from a data source, a new input instance with time and value is generated, a new PM instance is also generated, and a new CP_{temp} is produced. This information is represented by the object functional property called *produces*. In this phase the algorithm *Temperature_membership_function* is called to get the values of the CPI_{Temp} instances.

As an example, Table 1 shows the instances generated in using this ontology when a value of 22.5 °C is got by the temperature sensor.

5. Prototype 2

In this prototype, we increased the complexity of the problem introducing a 2PM, i.e., the combination of different CPs. Here, the computational system must provide sentences describing the perception of comfort in a room depending on the temperature and light intensity values. Thus, we used a light intensity sensor as an additional input. Fig. 5 shows the GLMP that defines the relationships among the CPs of temperature, light and comfort (CP_{temp}, CP_{light}, and CP_{comf}) respectively.

This computational application increases the Prototype 1 with two new PMs:

$1PM_{light}$ is defined by (z, CP_{light}, g, T) where
z is the input given by the light sensor, provided in mV in the interval [0, 1000].
CP_{light} is the output.
g is implemented using a strong fuzzy partition of trapezoidal membership functions.
T provides three possible sentences: *"The light is low", "The light is medium", "The light it high".*
$2PM_{comf}$ is defined by (U, CP_{comf}, g, T) where
U are the inputs given by CP_{temp} and CP_{light}.
CP_{comf} is the output.
g is a function which is implemented using the set of fuzzy rules shown in Table 2. This function returns the membership degree for three different fuzzy sets, i.e., *good, medium and bad.*
T is implemented using a linguistic template that provides three possible sentences: *"The comfort is bad", "The comfort is medium" and "The comfort is good".*
The new ontology is shown in the Fig. 6 and the new elements (the light and comfort measures) are defined as follows:

$Input_{light}$ class is defined by two functional properties called *val* (light intensity) and *time*.
CP_{light} is a CP related with the light input.
CP_{comf} is a computational perception related with the comfort output. Like in the previous definition, the object property *composed_of* is constrained in two senses: by setting the cardinality value to 1 per each subclass of CPI_{comf} and establishing the only (universal) restriction to the CPI_{comf} class.
CPI_{light} is associated to CP_{light} and has the three following subclasses:

- CPI_{high} class. The sentence annotation value is set to *"The light is high".*

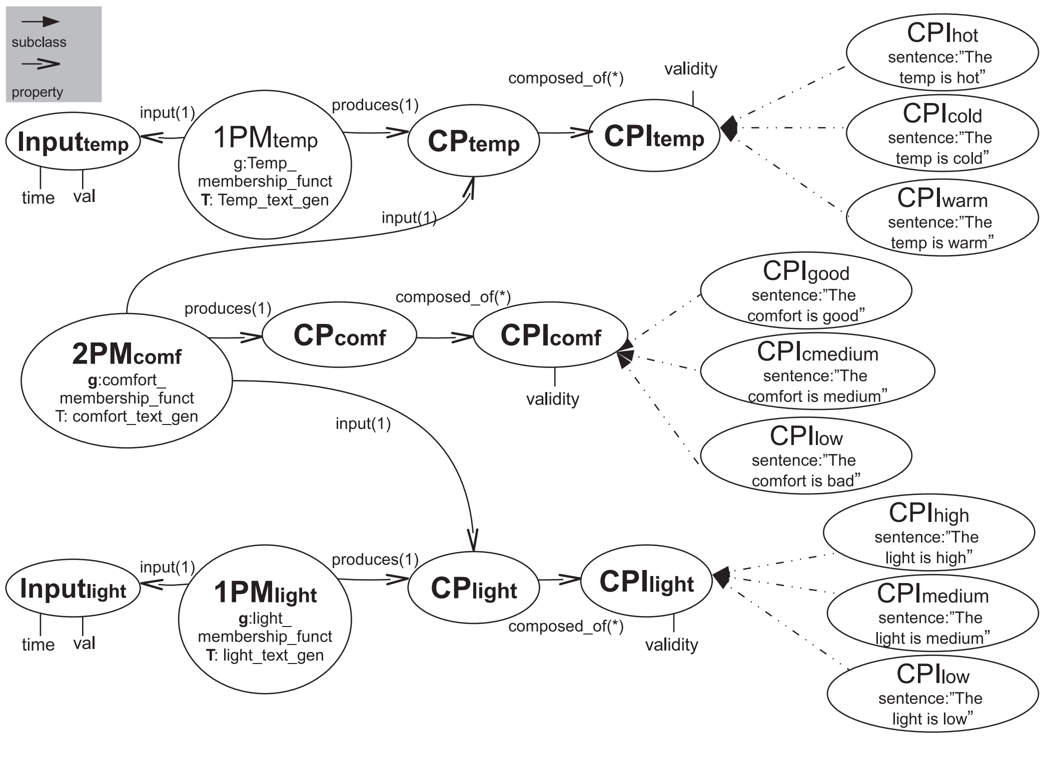

Fig. 6. Scheme of the application ontology for prototype 2.

- CPI$_{medium}$ class. *“The light is medium”.*
- CPI$_{low}$ class. *“The light is low”.*

CPI$_{comf}$ is associated to *CP$_{comf}$* with three subclasses:

- CPI$_{high}$ class. *“The comfort is good”.*
- CPI$_{medium}$ class. *“The comfort is medium”.*
- CPI$_{low}$ class. *“The comfort is bad”.*

1PM$_{light}$ class represents the productions of computational perceptions of light. Each time a level of light is given a new PM instance is generated and a new CP$_{light}$ is produced (*produces* property). It includes two annotations labels:

- *g* is called *Light_membership_function.*
- *T* generates the set of sentences included as annotations of CPI$_{light}$ subclasses. This algorithm is called *Light_text_ generator.*

2PM$_{comf}$ class generates a second order computational perception of comfort (*produces* property). Input property of this class are one CP$_{temp}$ and another CP$_{light}$ instances. Similarly to 1PM classes, 2PM classes include information about the algorithms associated with it:

- *g* calculates the membership degree of each linguistic expression in CPI. This algorithm is called Comfort_ membership_function.

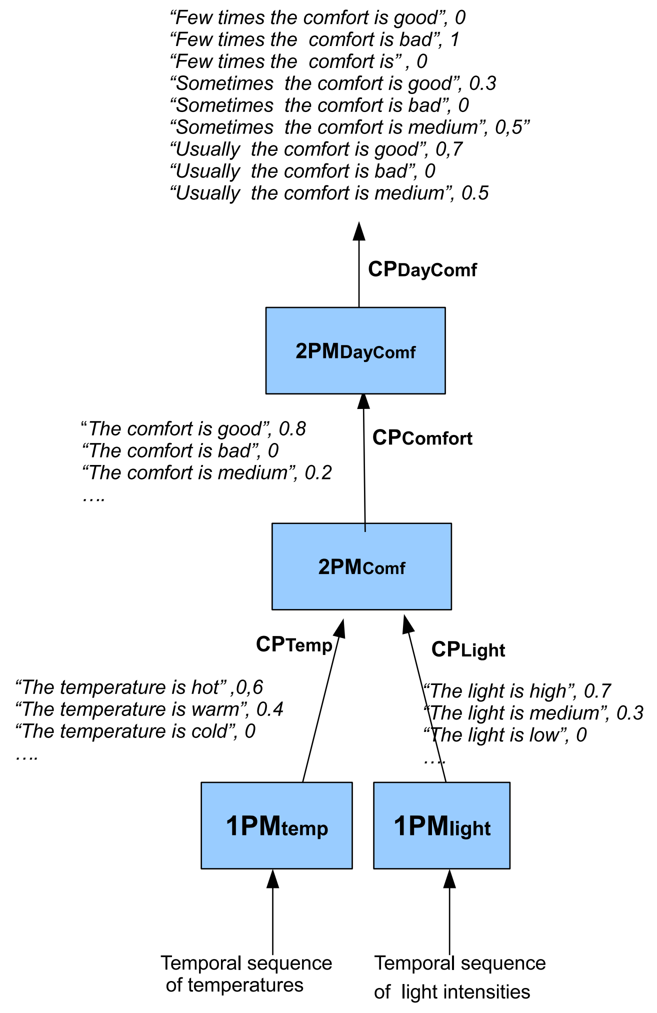
“Few times the comfort is good”, 0
“Few times the comfort is bad”, 1
“Few times the comfort is” , 0
“Sometimes the comfort is good”, 0.3
“Sometimes the comfort is bad”, 0
“Sometimes the comfort is medium”, 0,5”
“Usually the comfort is good”, 0,7
“Usually the comfort is bad”, 0
“Usually the comfort is medium”, 0.5
CPDayComf
2PMDayComf
“The comfort is good”, 0.8
“The comfort is bad”, 0
“The comfort is medium”, 0.2
….
CPComfort
2PMComf
CPTemp
CPLight
“The temperature is hot” ,0,6
“The temperature is warm”, 0.4
“The temperature is cold”, 0
….
“The light is high”, 0.7
“The light is medium”, 0.3
“The light is low”, 0
….
1PMtemp
1PMlight
Temporal sequence of temperatures
Temporal sequence of light intensities

Fig. 7. GLMP for prototype 3 including several instances of generated sentences.

- *T* generates the set of sentences defined as annotations of CPI$_{comf}$ subclasses. This algorithm is called Comfort_text_ generator.

Since we have already illustrated the population of the *application ontology* for the previous prototype, we are not doing it for this second prototype. We will provide a more detailed example for the third prototype.

6. Prototype 3

In this prototype we add a new 2PM to aggregate the CP of comfort during a period of time. The new GLMP is shown in Fig. 7. The new 2PM has been called 2PM$_{daycomf}$ and its output CP is CP$_{daycomf}$. This CP yields sentences that quantify the comfort in a room during a period, e.g., *“Usually the comfort is good”*.

2PM$_{daycomf}$ is formally defined by (*u*, CP$_{daycomf}$, *g*, *T*) where:

u is a time series comprising the validities given by CPIs of CP$_{comf}$ during a period of time.
CP$_{daycomf}$ is the output.
g is a function that summarizes the time series by means of the evaluation of a quantified sentence of the form “Q of the times, the comfort was *S*”, with *Q* one of the quantifiers *Sometimes, Usually* and *Few times*, and *S* one of the possible comfort values *bad, medium,* and *good* given by CP$_{comf}$. For the evaluation of the quantified sentences we used method GD introduced in Delgado, Sánchez, and Vila (2000).
T is implemented using a linguistic template that provides the following sentences: *“Usually the comfort is bad”, “Usually the comfort is medium”, “Usually the comfort is good”, “Sometimes the comfort is bad”, “Sometimes the comfort is medium”, “Sometimes the comfort is good”, “Few times the comfort is bad”, “Few times the comfort is medium”, and “Few times the comfort is good”.*

In this prototype a new set of ontology constituents related with the definition of 2PM$_{daycomf}$ are added to the *application ontology* of prototype 2, and shown in Fig. 8 with gray background color. The classes added to the new ontology are:

The CP$_{daycomf}$ class is a summarized degree of comfort for a period of time. This class has some restrictions related with the object property *composed_of*: a cardinality restriction set to 1 per each subclass of CPI$_{daycomf}$ and an Universal (*only*) restriction set in the CPI$_{daycomf}$ class.
CPI$_{daycomf}$ is the superclass of all classes corresponding to sentences associated to CP$_{daycomf}$.
2PM$_{daycomf}$ generates the perception of degree of comfort for a period. The algorithms associated with this PM are:

- day_aggr_function (*g*).
- day_text_generator algorithm (*T*).

We show in Fig. 9 an integrated view of the *high level* and *application ontologies* for this prototype, where classes with gray background represent the *high level ontology*, and the others are the *application ontology* classes.

In the next section, we use this prototype in order to illustrate how to use GLMP ontologies for generating linguistic reports from data.

7. Generating linguistic descriptions using GLMP ontologies

In previous sections we have shown how to design a GLMP by using an ontology, and we have illustrated our proposals with

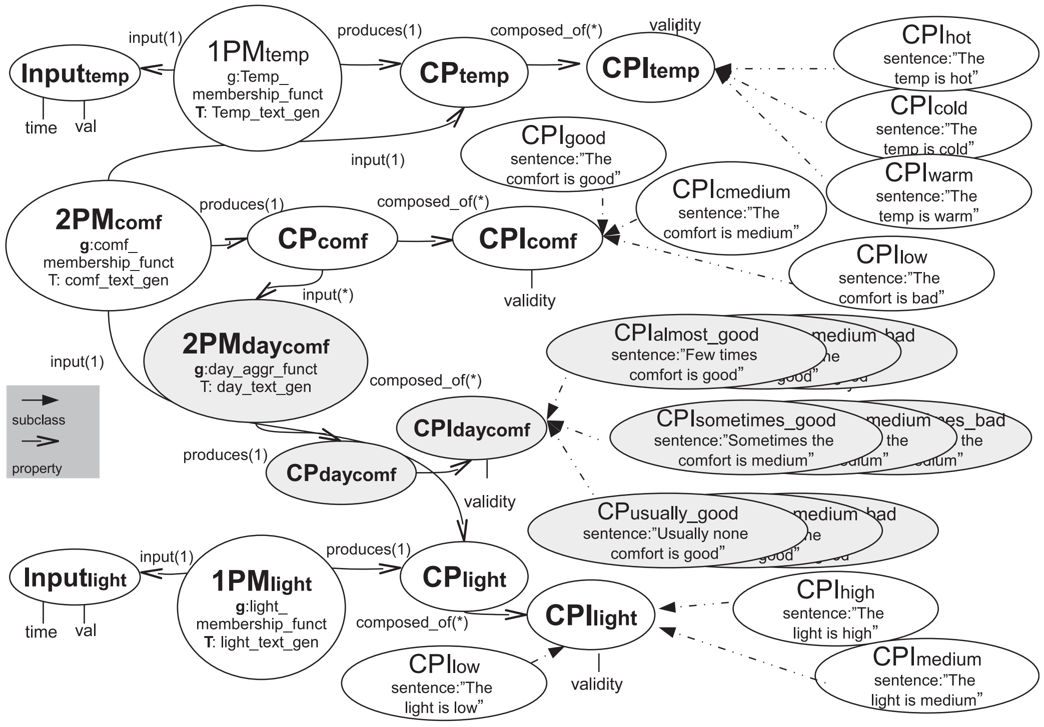
Inputtemp
time
val
input(1)
1PMtemp
g:Temp_ membership_funct
T: Temp_text_gen
produces(1)
CPtemp
composed_of(*)
CPItemp
validity
CPIhot sentence:"The temp is hot"
CPIcold sentence:"The temp is cold"
CPIwarm sentence:"The temp is warm"
CPIlow sentence:"The comfort is bad"
input(1)
CPIgood sentence:"The comfort is good"
CPIcmedium sentence:"The comfort is medium"
2PMcomf
g:comf_ membership_funct
T: comf_text_gen
produces(1)
CPcomf
composed_of(*)
CPIcomf
validity
input(*)
input(1)
2PMdaycomf
g:day_aggr_funct
T: day_text_gen
composed_of(*)
CPIalmost_good sentence:"Few times comfort is good"
CPIsometimes_good sentence:"Sometimes the comfort is medium"
CPusually_good sentence:"Usually none comfort is good"
CPIdaycomf
produces(1)
CPdaycomf
validity
subclass
property
Inputlight
time
val
input(1)
1PMlight
g:light_ membership_funct
T: light_text_gen
produces(1)
CPlight
composed_of(*)
CPIlight
validity
CPIhigh sentence:"The light is high"
CPImedium sentence:"The light is medium"
CPIlow sentence:"The light is low"

Fig. 8. Scheme of the application ontology for prototype 3.

different prototypes. In this section, we explain how GLMP Ontologies may be used to provide final linguistic reports, and we illustrate this with an example using the prototype 3 described in the previous section, and a real dataset.

7.1. Generating the descriptions

In order to generate the linguistic descriptions we need two main components: the designed GLMP Ontology and an input dataset. The creation of the linguistic report has two main parts:

1. Population of the *application ontology* on the basis of the dataset. Data are incorporated to the ontology as instances of the input classes. Instances of 1PMs, their output CPs and corresponding CPI subclasses are also generated for each instance of the input classes. This process is repeated recursively for the 2PMs along the network following the input–output direction. Finally, the ontology is populated not only with data, but also with a collection of CPIs corresponding to specific timestamps or time intervals.
2. Construction of the final report by a suitable selection and combination of the statements in the CPIs obtained. Note that, in general, taking advantage of the ontology representation, we can see the report template as a query to the ontology.

For the implementations we have used the Protégé 4.1 software tool to define the ontology structure, restrictions, and instances. The ontology was codified in OWL2 language. The implementation of the computational processes were developed in Java as a plug-in of the Protégé 4.1 framework. This plug-in is a generic tool that allows the user managing GLMP Ontologies from two different perspectives:

- Supporting the designer to generate the *application ontology*. The membership functions, fuzzy rules, and variable descriptions need to be defined conforming with the JFuzzyLogic API library[2] in a text file. This tool generates classes and constraints automatically. In the current version, some specifications like annotations should be done manually, but we are working on another version in which this is performed automatically. The use of this functionality is optional.
- Generating the results of the GLMP computational application as an ontology. This procedure requires the input data format definition in the source code. After that, the validity degree for each computational perception is calculated automatically. Finally, the report is generated by choosing the CPIs that are relevant to the user, that can be specified using a suitable ontology query language like SPARQL[3]

Let us remark again that one of the main advantages of our approach is that, by implementing generic algorithms for performing the aforementioned tasks, the implementation of a system for the linguistic description of data only requires collecting the data and designing an *application ontology* representing the desired GLMP. Part of our future research effort will be devoted to continue developing this general implementation.

7.2. Practical example

In order to illustrate the approach, we have considered the *Prototype 3 application ontology* and we have collected an input data set. For getting these data, we set sensors of temperature and light intensity on a window in our research center facilities, and we connected them to a data acquisition system plugged to a personal computer. With this simple platform, we obtained temperature and luminosity measures each minute during 10 days, that were stored in a plain file. These data are represented graphically in Fig. 10. Using this data we have populated the *Prototype 3 application ontology*. Obviously, it is not possible to show here the whole populated ontology for all classes related with temperature, light and comfort, during such a period of time. Table 3 shows some

[2] http://jfuzzylogic.sourceforge.net/html/index.html.

[3] http://www.w3.org/TR/rdf-sparql-query/.

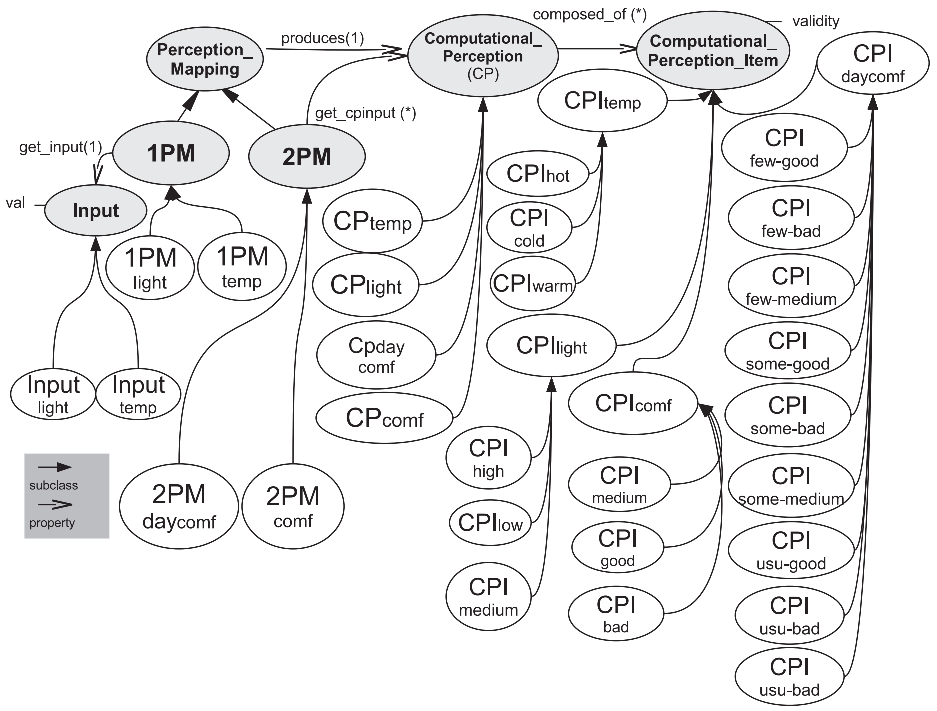

Fig. 9. Integrated *high level* (classes with light gray background) and *application ontology* for prototype 3. Only classes and high level restrictions are shown.

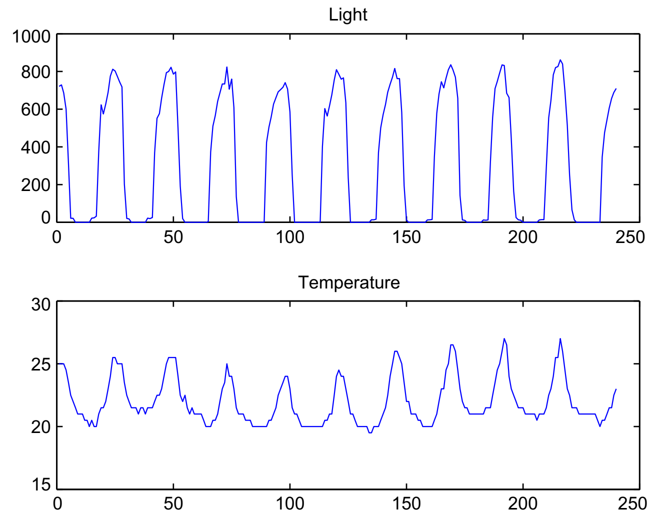

Fig. 10. Temperatures and light intensity got with the sensors, one each hour from September 23rd at 15:57:43 during 10 days.

instances of classes generated when populating the *Prototype 3 application ontology* with data in Fig. 10.

Two basic ideas are learned with these results: (i) There are evaluated sentences that return valid assessments because they have a high validity value, e.g. The *"Usually the comfort is good",0.7*. (ii) There are many sentences that do not give any information. Specially, those with a very low degree of validity, e.g. *"Usually the comfort is bad",0* or *"The temperature is cold", 0*.

Furthermore, we specified the form of the final report by using a SPARQL query in which we selected the sentence with the greatest degree of validity among those expressing what happens *sometimes* or *usually*, adding also as more detailed information the maximum and minimum values of temperatures. This way, the final report obtained for our data was "During this period, usually the comfort was good; the maximum and minimum temperatures were 27 °C and 19 °C, respectively".

Table 3
Some instances generated when populating the *application ontology* for prototype 3 with data shown in Fig. 10.

Instance (name)	Class of	Validity	Sentence
CPI10	CPIalmostgood	0	"Few times comfort is good"
CPI11	CPIalmostbad	1	"Few times comfort is bad"
CPI12	CPIalmostmedium	0	"Few times comfort is medium"
CPI13	CPIsometimesgood	0.3	"Sometimes comfort is good"
CPI14	CPIsometimesbad	0	"Sometimes comfort is bad"
CPI15	CPIsometimesmedium	0.5	"Sometimes comfort is medium"
CPI16	CPIusuallygood	0.7	"Usually comfort is good"
CPI17	CPIusuallybad	0	"Usually comfort is bad"
CPI18	CPIusuallymedium	0.5	"Usually comfort is medium"

8. Conclusions

In this paper, we contribute to the field of CWP by using ontologies as a tool for modeling the meaning of perceptions about complex phenomena. We describe how to use ontologies to implement computational models of phenomena by defining a network of granular perceptions (GLMP). The obtained results open the way for using different types of aggregations indicating the way some statements are consequence of others, hence allowing us to model more complex sentences and linguistic reports. Remarkably, only an *application ontology* consisting of classes and restrictions must be designed. The population of the ontology with data provides automatically the information to populate also with statements and their validity degree, which can be employed finally for building the final report as a query to the ontology. A good part of the power of our approach lies in the actual possibility of developing generic algorithms for performing the ontology population.

A lot of work remains to be done. It is worth remarking that the text generation algorithm (T) is the gate to link this model with the field of Computational Linguistics. An extension of the ontology to manage more complex linguistic expressions including contextual information will increase the usability of these computational applications, making the system closer to human's adaptive management of natural language.

Our experimentation shows that, in the current state of development, this new technology can be applied to solve practical projects.

Acknowledgments

The research reported in this paper was partially supported by the Spanish Government (MICINN) under projects TIN2008-06890-C02-01, TIN2009-08296, and TIN2008-02066, and by the Andalusian Government under projects P07-TIC-03175 and TIC-6109.

References

Agarwal, P. (2005). Ontological considerations in giscience. *International Journal of Geographical Information Science, 19*(5), 501–536. 36.

Alvarez-Alvarez, A., Sanchez-Valdes, D., & Trivino, G. (2011). Automatic linguistic description about relevant features of the mars surface. In: *Proceedings of the 2011 IEEE international conference ISDA*. Cordoba, Spain.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 28–37.

Bobillo, F., Delgado, M., & Gómez-Romero, J. (2012). Delorean: A reasoner for fuzzy owl 2. *Expert Systems with Applications, 39*, 258–272.

Bobillo, F., & Straccia, U. (2011). Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning, 52*(7), 1073–1094.

Calegari, S., & Ciucci, D. (2010). Granular computing applied to ontologies. *Internationa Journal of Approximate Reasoning, 51*, 391–409.

Delgado, M., Sánchez, D., & Vila, M. (2000). Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning, 23*, 23–66.

Diaz-Hermida, F., Ramos, A., & Bugarin, A. (2011). On the role of fuzzy quantified statements in linguistic summarization. In *Proceedings of the 2011 IEEE international conference ISDA*. Cordoba, Spain.

Ghorbel, H., Bahri, A., & Bouaziz, R. (2009). Fuzzy protTgT for fuzzy ontology models. In: *Proceedings of 11th International ProtTgT Conference IPC'2009*. Academic Medical Center, University of Amsterdam, Amsterdam, Netherlands.

Gómez-Pérez, A., Férnandez-López, M., & Corcho-García, O. (2003). *Ontological engineering*. New York: Springer-Verlag, Inc.

Jiang, Y., Tang, Y., Chen, Q., & Wang, J. (2011). Reasoning and change management in modular fuzzy ontologies. *Expert Systems with Applications, 38*(11), 13975–13986.

Kacprzyk, J., Yager, R., & Zadrozny, S. (2000). A fuzzy logic based approach to linguistic summaries of databases. *International Journal of Applied Mathematics and Computer Science, 10*(4), 813–834.

Lee, C.-S., Chen, Y.-J., & Jian, Z.-W. (2003). Ontology-based fuzzy event extraction agent for chinese e-news summarization. *Expert Systems with Applications, 25*(3), 431–447.

Lee, C.-S., & Wang, M.-H. (2007). Ontology-based intelligent healthcare agent and its application to respiratory waveform recognition. *Expert Systems with Applications, 33*(3), 606–619.

Martinez-Cruz, C., Blanco, I., & Vila, M. (2011). Ontologies versus relational databases: are they so different? a comparison. *Artificial Intelligence Review*, 1–20. <http://dx.doi.org/10.1007/s10462-011-9251-9>.

Mendel, J. M., Lawry, J., & Zadeh, L. A. (2010). Foreword to the special section on computing with words. *IEEE Transactions on Fuzzy Systems, 18*(3), 437–440.

Mendez-Nunez, S., & Trivino, G. (2010). Combining semantic web technologies and computational theory of perceptions for text generation in financial analysis. In *Proceedings of the 2010 IEEE international conference on fuzzy systems (FUZZ-IEEE), July 18–23* (pp. 953–960), Barcelona, Spain.

Seo, K.-Y., Park, G.-K., Lee, C.-S., & Wang, M.-H. (2009). Ontology-based fuzzy support agent for ship steering control. *Expert Systems with Applications, 36*(1), 755–765.

Staab, S., & Studer, R. (2004). *Handbook on ontologies*. Springer.

Trivino, G., Sanchez, A., Montemayor, A. S., Pantrigo, J. J., Cabido, R., & Pardo, E. G. (2010). Linguistic description of traffic in a roundabout. In: *Proceedings of the 2010 IEEE international conference on fuzzy systems (FUZZ-IEEE), July 18–23* (pp. 2158–2165). Barcelona, Spain.

Yager, R. R. (1982). A new approach to the summarization of data. *Information Sciences, 28*, 69–86.

Zadeh, L. A. (1975). The concept of linguistic variable and its application to approximate reasoning. *Information sciences, 8*, 199–249.

Zadeh, L. A. (1979). Fuzzy sets and information granularity. *Advances in Fuzzy Set Theory and Applications*, 69–129.

Zadeh, L. A. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computing and Mathematics with Applications, 9*, 149–184.

Zadeh, L. A. (1997). Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems, 90*, 111–127.

Zadeh, L. A. (1999). From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions. *IEEE Transactions on Circuits and Systems, 45*(1).

Zadeh, L. A. (2008). Toward human level machine intelligence – Is it achievable? the need for a paradigm shift. *IEEE Computational Intelligence Magazine*.