Quantity and Degree Assessment in an RDF-based Semantic Language

Igor Boguslavsky
Universidad Politécnica de Madrid, Spain / Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia

Abstract - *This paper introduces a semantic language developed with the objective to be used in a semantic analyzer based on linguistic and world knowledge. Linguistic knowledge is provided by a Combinatorial Dictionary and several sets of rules. Extra-linguistic information is stored in an Ontology. The meaning of the text is represented by means of a series of RDF-type triples of the form* predicate(subject, object). *Semantic analyzer is one of the options of the multifunctional ETAP-3 linguistic processor. The analyzer can be used for Information Extraction and Question Answering. We describe semantic representation of expressions that provide an assessment of the number of objects involved and/or give a quantitative evaluation of different types of attributes. We focus on the following aspects: 1) parametric and non-parametric attributes; 2) gradable and non-gradable attributes; 3) ontological representation of different classes of attributes; 4) absolute and relative quantitative assessment; 5) punctual and interval quantitative assessment; 6) intervals with precise and fuzzy boundaries.*

Keywords: knowledge representation, semantic analysis, ontology, RDF languages, gradable attributes

1 Semantic Analysis and Semantic Structure

The semantic language described in this paper is used in a semantic analyzer under development in the Institute for Information Transmission Problems of the Russian Academy of Sciences and in the Madrid Polytechnic University. A distinctive feature of this analyzer is that the semantic representation is based on both linguistic and world knowledge. In many aspects our approach is similar to that of Ontological Semantics (Nirenburg, Raskin 2004) but the linguistic framework is substantially different. For more details about the analyzer, the reader is referred to Boguslavsky et al. 2010 and Boguslavsky 2011.

The source of linguistic information is the knowledge base of the ETAP-3 linguistic processor. World knowledge is contained in the Ontology and in the Fact Repository. Semantic representation is constructed in two steps. First, all the sentences of the text sentence are independently of one another transformed into their Backbone Semantic Structures (BSemS), which reflect the meaning directly conveyed in the sentence. Second, BSemSs are enriched by means of the Ontology and the Fact Repository and converted to Enhanced Semantic Structures (EnSemS). As an example, we will show BSemS and EnSemS of a short text (1). The working language of the analyzer is Russian, but for readers' convenience we provide an English translation of the text:

(1) *On October, 8 "Zenit" Football Club had a first friendly match in the tournament, which is taking place in Spain's La Manga. The pupils of Luciano Spaletti lost 0:1 to the Warsaw Club "Polonia".*

Step 1: BSemS construction.

BSemS of the first sentence: *On October, 8 "Zenit" Football Club had a first friendly match in the tournament, which is taking place in Spain's La Manga.*

(2)
```
hasTime(Match-01,DateTimeDescription-01)
hasMonth(DateTimeDescription-01, 2)
hasDay(DateTimeDescription-01, 8)
hasCategory(Match-01, friendly)
isPartOf(Match-01, Tournament-01)
hasParticipant(Match-01, Team-01)
hasName(Team-01, Zenit)
```

```
hasSportDiscipline(Team-01, football)
hasLocation(Tournament-01, La Manga)
hasLocation(La Manga, Spain)
```

As is usual in Artificial Intelligence, instances of classes are supplied with unique numerical identifiers. In order not to distract the reader's attention, from now on, we will omit these identifiers, if only they are not needed to distinguish between different instances.

It should be stressed that many of the classes referred to in the BSemS have many more property slots in the Ontology, but these slots are not represented in this BSemS, since the text does not provide data for filling them.

BSemS of the second sentence: *The pupils of Luciano Spaletti lost 0:1 to the Warsaw Club "Polonia".*

```
(3)  hasWinner(WinEvent, Team-02)
     hasLoser(WinEvent, Team-03)
     inContest(WinEvent, SportEvent)
     inContest(MatchScore, SportEvent)
     hasValue1(MatchScore,1)
     hasValue2(MatchScore,0)
     hasName(Team-02, Polonia)
     represents(Team-02, Warsaw)
     hasCoach(Team-03, Luciano Spalletti)
```

The second sentence was analyzed independently of the first one, and the two teams mentioned (the winner and the loser) received new identifiers – `Team-02` and `Team-03`. These teams are referred to in the text in different ways. One of them is called by its name (`Polonia`), and the only thing that is communicated about the second one is the name of its coach (`Luciano Spaletti`). The result of the match is conveyed by means of two propositions: the first one says that one of the teams defeated the other (by means of the predicate `WinEvent`), and the second one gives the score (the predicate is `MatchScore`). In order to show that both propositions refer to the same match, the `inContest` slot of both predicates is filled by the same match (`SportEvent`).

Step 2: Enhancement of BSems.

The main tasks that are solved at this stage are coreference resolution (i.e. determination of phrases that refer to the same entity) and filling of the slots for which the fillers have not yet been found. The information used to solve these tasks can be taken from the Ontology and the Fact Repository, which contains, among other things, semantic structures of previous sentences of the text processed. In particular, the Ontology should contain all the information available about the individuals. In our example, the Ontology disposes of the following information about the Zenit team:

```
hasName(Team-00, Zenit)
hasSportDiscipline(Team-00, football)
hasCoach(Team-00, Luciano Spalletti)
```

Based on this information, we can identify `Team-01` from the first sentence, whose name is `Zenit`, with `Team-03` from the second one, since its coach is `Luciano Spaletti`.

Semantic links between the sentences in the text are implicit. In EnSemS they should be restored wherever possible. We will show how it can be done in our example. There are several considerations that allow us to infer some implicit information. The first sentence of (1) tells us that a match has taken place. The second sentence gives information about a result of a match. The two sentences belong to the same text and, moreover, one of them directly follows the other one. They share at least one participant (`Zenit`) and there are no textual indications of the topic shift. Therefore, one can conclude that both sentences are describing the same match. That means that in the EnSemS we should identify `Match` from the first sentence with `SportEvent`, to which `WinEvent` and `MatchScore` are linked in the second sentence. As a result, EnSemS of (1) looks like (4) (elements introduced during the SemS extension are printed in bold face and underlined):

```
(4)  hasTime(Match, DateTimeDescription)
```

```
hasMonth(DateTimeDescription, 2)
hasDay(DateTimeDescription, 8)
hasCategory(Match, friendly)
isPartOf(Match, Tournament)
hasParticipant(Match, Team-01)
hasName(Team-01, Zenit)
hasCoach(Team-01, Luciano Spaletti)
hasSportDiscipline(Team-01, football)
hasLocation(Tournament, La Manga)
hasLocation(La Manga, Spain)
hasWinner(WinEvent, Team-02)
hasLoser(WinEvent, Team-01)
inContest(WinEvent, Match-01)
inContest(MatchScore, Match-01)
hasValue1(MatchScore, 1)
hasValue2(MatchScore, 0)
hasName(Team-02, Polonia)
represents(Team-02, Warsaw)
```

Semantic structures (2)-(4) have already given some idea of the semantic language used for representing the meaning of the texts. Below, we will concentrate on one important fragment of this language, namely the one dealing with quantitative assessment of different types of objects.

Quantitative assessment is mostly made by means of g r a d a b l e   a t t r i b u t e s . We will discuss these attributes in section 2, and the values they ascribe to objects - in section 3.

2 Gradable attributes

As mentioned above, prototypically, quantitative assessment of an object is made by means of gradable attributes. Properties that they ascribe to the objects can be compared in terms of “more/less”. For example, `Price` and `Intelligence` are gradable, because one bicycle can be more expensive than another one, and one person can be more intelligent than another. Very often, they characterize an object by assigning it a numerical value (e.g. *a price of 100 Euros*). However, not all the attributes that assign a value to their object are gradable and not all gradable attributes assign a value to their object. Let us explain this.

We will refer to the attributes that assign some definite value to their object as p a r a m e t e r s . The parameters and their values can be quantitative or not. Values of quantitative parameters, such as `Price, Height, Speed, Temperature`, etc. are located on a scale and are prototypically expressed by numerical expressions consisting of a numeral and a measurement unit: *the height of 15 meters, the speed of 180 km/hour, the temperature of 50 degrees Celsius.*

Non-quantitative parameters have values that are not located on a scale and are not expressed by means of a numerical expression. Typical examples of non-quantitative parameters are `Name, Nationality, Address, Color, Form`, etc. For example, the parameter of `Nationality` has values `French, Russian, Malay`, etc.

All the parameters, quantitative or not, have a sufficiently homogenous and compact range of values. The kind of entities that can serve as values of a parameter is always restricted. An important linguistic property of parameters is their behavior in *what*-questions. Any word denoting a parameter can be used in a *what*-question, and this question is always interpreted as a request for the parameter’s value. When somebody asks *What is her name?* it is quite obvious what kind of entity is expected as an answer. Cf. a dialogue in which the answer does not belong to the class of parameter values and therefore does not directly satisfy the information need of the interrogator: *What is his name? - ??An unusual one*. Not all attributes are like this. Such attributes as `Beauty` or `Intelligence` do not presuppose a set of homogeneous values. Therefore, they are not parameters.

On the other hand, `Beauty` and `Intelligence` are gradable attributes, in the sense that different objects may have these properties to different extents – see above.

However, gradability of this property does not only manifest itself in comparative contexts, as is the case of (5):

(5) *Lucy is more intelligent than Mary.*

When we simply say that Lucy is intelligent, we ascribe to Lucy a high degree of intelligence, in a similar way as we ascribe to her a high degree of height, when we say that she is tall.

However, there is also an important difference between such attributes as `Height` on the one hand, and `Intelligence`, on the other. We showed above that `Height` is a parameter, and `Intelligence` is not. `Height` is a two-place predicate, which assigns to its object a value on a well-defined scale, while `Intelligence` is a one-place predicate. It does not assign any definite value to its object, but instead it has degrees.

We would like to capture the similarity between the attributes of the `Height` class and the `Intelligence` class, and on the other hand, take into account the difference between them. We adopted the following solution. Our ontology contains the class of `Parameters` with two subclasses: `GradableParameters` (like `Price`) and `Non-gradableParameters` (like `Name`). All the parameters have a `hasValue` slot. On the other hand, there is a class `GradableAttributes`. One of its subclasses is `GradableParameters` (which at the same time belongs to the `Parameters` class), and another one is `QualitativeAttributes`, which contains attributes of the `Intelligence` type. Gradable parameters have values located on a corresponding scale. Qualitative attributes do not have values but have a `hasDegree` slot instead. Specific behavior of parameters is accounted for by means of the `Parameters` class and its `hasValue` slot. For example, interpretation of the *what*-questions asked of parameters implies filling of the `hasValue` slot. How the quantification of qualitative attributes is assured is seen in SemS (5a), which represents the meaning of (5):

(5a)
```
hasSubject(Intelligence-01, Lucy)
hasDegree(Intelligence-01, var-01)
hasSubject(Intelligence-02, Mary)
hasDegree(Intelligence-02, var-02)
more(var-01, var-02)
```

3 Values of gradable attributes

As stated in the previous section, parameters and qualitative attributes are both gradable but in a slightly different way. Although they have different slots for quantitative elements (`hasValue` and `hasDegree`), for simplicity's sake we will speak about values in both cases.

3.1 Numerical expressions

As for quantitative parameters, their values are compound numerical expressions. They are formed by a numeral and a measurement unit. Such expressions are represented by a series of two-place predicates that express a corresponding measure, such as `LinearMeasure, CurrencyMeasure, DurationMeasure`, etc.

(6) *3 meters*

(6a)
```
hasUnit(LinearMeasure, meter)
hasUnitQuantity(LinearMeasure, 3)
```

(7) *20 euros 45 cents*

(7a)
```
hasUnit(CurrencyMeasure, euro)
hasUnitQuantity(CurrencyMeasure, 20,45)
```

(8) *3 liters (of wine)*

(8a)
```
hasUnit(Amount, liter)
hasUnitQuantity(Amount, 3)
```

(9) *3 glasses (of wine)*

(9a)
```
hasUnit(Amount, glass)
hasUnitQuantity(Amount, 3)
```

A special case is temporal expressions, which are often presented in natural languages by several measurement units of different granularity, as in (10). We chose not to reduce them to a single unit (for example, a second), but to preserve the internal structure of the expression, in order to match the way humans are conceptualizing time:

(10) *3 years 2 months 5 days 10 hours and 25 minutes*

(10a)
```
hasYears(DurationMeasure, 3)
hasMonths(DurationMeasure, 2)
hasDays(DurationMeasure, 5)
hasHours(DurationMeasure, 10)
hasMinutes(DurationMeasure, 25)
```

Another special case is a quantitative parameter which may do without a measurement unit. It is quantity, e.g. *5 books*. Besides the peculiarity of lacking a unit, this parameter has another one. It does not apply to single objects but only to collections of objects. When we say *5 books*, we don't say anything about any single book, but describe quantitatively a whole set of books. Therefore, multiple objects are represented by sets (as in Nirenburg, Raskin 2004), and sets may have a `Cardinality` parameter:

(11) *50 books*

(11a)
```
hasElementType(Set, Book)
hasSubject(Cardinality, Set)
hasValue(Cardinality, 50)
```

There exist parameters whose value is not constituted by a single numerical expression. The state of a sport match at a given moment is characterized by the parameter of `Score`, which is a pair of numbers, e.g. `3:1`. Another example of a complex parameter value is coordinates on the Earth surface, which is a combination of two independent parameters – latitude and longitude, each of which is usually rendered by a triple of numbers – degrees, minutes and seconds.

3.2 Brief typology of quantitative assessment.

We will distinguish between two major types of quantitative assessment – absolute and relative assessment. Relative assessment, in its turn, will be divided into two subclasses: assessment with respect to the whole and with respect to the norm. In each of these three classes, the value can denote either a point on the scale, or an interval. In the latter case, the interval can have either exact or fuzzy boundaries. Below, we will illustrate all these classes.

A. Absolute assessment.

A.1 Punctual absolute assessment.

(12) *The ticket costs 20 euros*

(12a)
```
hasSubject(Cost, Ticket)
hasValue(Cost, CurrencyMeasure)
hasUnit(CurrencyMeasure, euro)
hasUnitQuantity(CurrencyMeasure, 20)
```

(13) *The boy is 4 feet tall*

(13a)
```
hasSubject(Height, Boy)
hasValue(Height, LinearMeasure)
hasUnit(LinearMeasure, foot)
hasUnitQuantity(LinearMeasure, 4)
```

A.2 Interval absolute assessment.

When a quantitative expression is characterized by an interval, its boundaries can be either exact or fuzzy. The difference between exact and fuzzy boundaries is manifested in inferences one can make. If it is claimed that the number of students is in the exact interval [100,150], then this claim is definitely false if in fact there are 98 or 151 students. If the interval is fuzzy, then such an inference cannot be made. The distinction between exact and fuzzy intervals is relevant for representing the meaning of many natural language words and phrases. An interval with exact boundaries is illustrated in (14), and one with fuzzy boundaries – in (15).

(14) *Only children from 5 to 8 years old are admitted to the class.*

(14a)
```
hasObject(Admit, Child)
hasSubject(Age, Child)
hasValue(Age, DurationMeasure)
hasYears(DurationMeasure, Interval)
from(Interval, 5)
to(Interval, 8)
```

(15) *About 10 to 15 students will come to the meeting.*

(15a)
```
hasAgent(Come, Set)
hasElementType(Set, Student)
hasSubject(Cardinality, Set)
hasValue(Cardinality, FuzzyInterval)
from(FuzzyInterval, 10)
to(FuzzyInterval, 15)
```

The word *several* is one of those natural language elements whose meaning cannot be rendered in exact terms. According to Longman English Dictionary *several* means ‘a number of people or things that is more than a few but not a lot’. COBUILD dictionary gives a similar definition: ‘imprecise number of people or things that is not large but is greater than two’. Such definitions, obviously, cannot be used directly. However, this word has a clear enough meaning, which is rather specific. On the one hand, it refers to concrete numbers, as opposed to such words as *many* or *few*. It is impossible to give a number, however approximate it might be, denoting *many* or *few*. It is obviously different in different situations. As for *several*, the situation is different. Most of our informants say that the number referred to by *several* is an integer which is greater than or equal to 3 but less than 9. On the other hand, the upper boundary is not rigid. Some people feel that perhaps 8 is too many to count as several. Other people agree to accept even as many as 10. One could of course ignore this vagueness and describe *several* as any quantity expressed by an integer greater than or equal to 3 but smaller than 9. However, if we wish to approximate meanings expressed by natural language words (and we do), we should find a way to account for such an important property of natural language semantics as uncertainty, which is obviously manifested in the meaning of *several*. Our proposal is to describe *several* as a fuzzy interval between 3 and 8:

(16) *several books*

(16a)
```
hasElementType(Set, Book)
hasSubject(Cardinality, Set)
hasValue(Cardinality, FuzzyInterval)
from(FuzzyInterval, 3)
to(FuzzyInterval, 8)
```

Similarly, one could characterize approximate quantities such as *about 80* as fuzzy intervals with identical lower and upper boundaries:

(17) *about 80 people*

(17a)
```
hasElementType(Set, Person)
```

```
hasSubject(Cardinality, Set)
hasValue(Cardinality, FuzzyInterval)
from(FuzzyInterval, 80)
to(FuzzyInterval, 80)
```

B. Relative Assessment.

There are two varieties of relative assessment: the assessment with respect to the whole and with respect to the norm. In both cases, it can be punctual or interval, as in case of the punctual assessment above, and the interval boundaries can be both exact and fuzzy.

B.1 Relative Assessment with respect to the whole.

To make this kind of assessment, we introduce a predicate `FormsPartOf`, whose first argument (subject) corresponds to the quantity being assessed, the second argument (object) – to the whole with respect to which the assessment is made, and the third argument (value) characterizes the relation between the two.

B.1.1 Punctual assessment

(18) *half of the population*

```
(18a) hasSubject(FormsPartOf, Set)
      hasObject(FormsPartOf, Population)
      hasValue(FormsPartOf, 0.5)
```

B.1.2 Interval Assessment.

Interval with exact boundaries:

(19) *From 13% to 15% of the population*

```
(19a) hasSubject(FormsPartOf, Set)
      hasObject(FormsPartOf, Population)
      hasValue(FormsPartOf, Interval)
      from(Interval, 0.13)
      to(Interval, 0.15)
```

Interval with fuzzy boundaries:

(20) *about 13% - 15% of the population*

```
(20a) hasSubject(FormsPartOf, Set)
      hasObject(FormsPartOf, Population)
      hasValue(FormsPartOf, FuzzyInterval)
      from(FuzzyInterval, 0.13)
      to(FuzzyInterval, 0.15)
```

B.2 Relative assessment with respect to the norm.

In natural languages there are many words that express a value of a gradable attribute, but do so in a rather indefinnite way without giving any concrete value. For example, a stone can be characterized as big. We understand that the adjective *big* places the stone somewhere on the scale of size, but we don't know this position exactly. The only thing that the lexical meaning of *big* tells us is that the size of the stone is above the norm. This idea has long been adopted in linguistics. What exactly the norm is in each particular case is beyond the competence of lexical semantics and is determined by the context and encyclopedic knowledge. What is big for a child may be not so for an adult. A big cat is much smaller than a small horse. However, it is not enough only to state that *big* is above the norm. *Enormous* is also above the norm, but to a different extent. There are series of words and phrases that refer to the same scale but denote different points, or rather, different areas, on that scale. *Never, seldom,* *sometimes, from time to time, often, very often, always* – all these words correspond to different degrees on the frequency scale. A convenient way of representing the meaning of such words has been proposed within the Ontological semantics approach. All the words that express a value of a gradable attribute are assigned a point or a range in the interval between 0 and 1. For example, the meaning of *hot* is represented as the range [0.75–1] on the scale of temperature (Nirenburg, Raskin 2004: 206). We adopt this idea in principle, but realize it in a different way. We prefer to make explicit the idea of relativity of the assessment. We introduce a predicate RelativeToNorm with 2 arguments:

- `normFor`: what kind of norm is used as a reference
- `hasValue`: what value is assigned

When we say that the boy is tall, we normally assess his height with respect to some expectations concerning normal height of boys. In a general case, however, this norm can be more specific

As in previous cases of assessment, the value slot can be filled both by a point and an interval in the range [0,1].

B.2.1 Punctual assessment

(21) *John is short for a basketball player* (John's height is

(21a)
```
hasSubject(Height-01, John)
hasValue(Height-01, RelativeToNorm-01)
normFor(Height-01, Height)
hasSubject(Height, BasketballPlayer)
hasValue(RelativeToNorm-01, 0.2)
```

(21) *false solution* (limit point on the scale of Correctness)

(21a)
```
hasSubject(Correctness, Solution)
hasValue(Correctness, RelativeToNorm)
hasValue(RelativeToNorm, 0)
```

(22) *brilliant answer* (limit point on the scale of Quality)

(22a)
```
hasSubject(RelativeToNorm, Answer)
hasObject(RelativeToNorm, Quality)
hasValue(RelativeToNorm, 1)
```

B.2.2 Interval assessment

Interval assessment with exact boundaries does not seem to be possible. Fuzzy boundaries, on the contrary, are very wide spread.

(23) *a tall boy*

(23a)
```
hasSubject(RelativeToNorm, Boy)
hasObject(RelativeToNorm, Height)
hasValue(RelativeToNorm, FuzzyInterval)
from(FuzzyInterval, 0.6)
to(FuzzyInterval, 0.8)
```

(24) *a very tall boy*

(24a)
```
hasSubject(RelativeToNorm, Boy)
hasObject(RelativeToNorm, Height)
hasValue(RelativeToNorm, FuzzyInterval)
from(FuzzyInterval, 0.8)
```

```
      to(FuzzyInterval, 0.9)
```

As one can easily see from (23) – (24), the difference between a tall boy and a very tall boy is manifested in different intervals on the scale of `Height`: [0.6, 0.8] in the first case and [0.8, 0.9] in the second case.

(26) *story of average length*

```
(26a) hasSubject(RelativeToNorm,Story)
      hasObject(RelativeToNorm,Length)
      hasValue(RelativeToNorm,FuzzyInterval)
      from(FuzzyInterval, 0.4)
      to(FuzzyInterval, 0.6)
```

(27) *negligibly few resources*

```
(27a) hasSubject(RelativeToNorm, Resource)
      hasObject(RelativeToNorm, Amount)
      hasValue(RelativeToNorm, FuzzyInterval)
      from(FuzzyInterval, 0.1)
      to(FuzzyInterval, 0.2)
```

Now, it is not difficult to represent the difference between three related constructions: (28) *John solved* ***many problems,*** (29) *John solved* ***many of the problems***, and (30) *John solved* ***most of the problems***. Sentence (28) simply says that the number of the problems John solved is large. This is done by means of the `RelativeToNorm` predicate. It should be stressed that in this context the set of the problems John solved is interpreted as the set of **all the problems** that John solved. This meaning is rendered by a special attribute – isComplete.

(28) *John solved many problems*

```
(28a) hasAgent(Solve, John)
      hasObject(Solve, Set)
      hasElement(Set, Problem)
      isComplete(Set)
      hasSubject(RelativeToNorm, Set)
      hasObject(RelativeToNorm, Cardinality)
      hasValue(RelativeToNorm, FuzzyInterval)
      from(FuzzyInterval, 0.6)
      to(FuzzyInterval, 0.8)
```

Sentence (29) evaluates the proportion between two sets: some set of problems and its subset that John solved. What *many* claims here is that the second set is a large part of the first one. It is difficult to define precisely what part of the whole qualifies as large. However, what is certain is that it cannot necessarily be more than a half. It is quite possible that many of the problems have been solved and at the same time many of the problems remained unsolved. Consequently, the beginning of the large-part interval should be less than 0.5. On the other hand, a large part of something can easily be more than a half. To express this, one needs to introduce these two sets of problems and evaluate their correlation by means of `FormsPartOf`:

(29) *John solved many of the problems.*

```
(29a) hasElement(Set-01, Problem)
      hasProperSubset(Set-01, Set-02)
      hasAgent(Solve, John)
      hasObject(Solve, Set-02)
      isComplete(Set-02)
```

```
hasSubject(FormsPartOf,Set-02)
hasObject(FormsPartOf, Set-01)
hasValue(FormsPartOf,FuzzyInterval)
from(FuzzyInterval, 0.3)
to(FuzzyInterval, 0.7)
```

Sentence (30) is similar to (29) in that it compares the set of the problems that John solved and the overall set of problems. What it adds to (29) is the idea that the number of solved problems is larger than half of the whole number of problems. One of the ways to convey this meaning is to change the value of the `FormsPartOf` predicate .

(30) *John solved most of the problems.*

(30a)
```
hasElement(Set-01, Problem)
hasProperSubset(Set-01, Set-02)
hasAgent(Solve, John)
hasObject(Solve, Set-02)
isComplete(Set-02)
hasSubject(FormsPartOf, Set-02)
hasObject(FormsPartOf, Set-01)
hasValue(FormsPartOf, FuzzyInterval)
from(FuzzyInterval, 0.55)
to(FuzzyInterval, 0.9)
```

4 Conclusions

We presented a semantic language intended for performing semantic analysis and representing the meaning of NL texts. Semantic analysis is knowledge-intensive and uses two major sources of knowledge: linguistic knowledge incorporated in the combinatorial dictionary and rules of the ETAP-3 Linguistic Processor, and world knowledge stored in the Ontology and Fact Store. At the moment of writing, the Ontology is under construction. In developing it, we take into account various existing ontologies, such as SUMO, Ontological Semantics Ontology, and PROTON, but in many cases take our own solutions. The semantic language in which semantic structures of sentences are formulated consists of RDF-type triples `predicate(subject, object)`. In this paper, we show how this language represents information on quantity and degree assessment.

5 References

[1] Boguslavsky, I., L. Iomdin, V. Sizov, S. Timoshenko. (2010). Interfacing the Lexicon and the Ontology in a Semantic Analyzer. In: COLING 2010. Proceedings of the 6th Workshop on Ontologies and Lexical Resources (Ontolex 2010), Beijing, August 2010, pages 67–76.

[2] Boguslavsky, I. (2011). Semantic Analysis based on linguistic and ontological resources. In: Proceedings of the 5th International Conference on the Meaning - Text Theory. Barcelona, September 8 – 9, 2011. Igor Boguslavsky and Leo Wanner (Eds.), p. 25-36.

[3] Nirenburg, S., Raskin, V. (2004). Ontological Semantics.The MIT Press.Cambridge, Massachusetts. London, England.