

# Rule-based Reasoning in Semantic Text Analysis<sup>1,2</sup>

Ivan Rygaev

Laboratory of Computational Linguistics, A. A. Kharkevich Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, Russia  
irygaev@gmail.com

**Abstract.** In this paper we demonstrate the rule engine of the SemETAP semantic text analyzer being developed in the Laboratory of Computational Linguistics of the Institute for Information Transmission Problems of the Russian Academy of Sciences. The rule engine is based on SPARQL queries and implements the forward chaining algorithm for existential rules. In a case study we show the system's capabilities and limitations, identify topics for further research and provide requirements for a reasoner that would satisfy our needs.

**Keywords:** Semantic text analysis · Semantic concept definitions · Inference · Existential rules

## 1 Introduction

The semantic text analyzer SemETAP, under development in the Laboratory of Computational Linguistics of the Institute for Information Transmission Problems of the Russian Academy of Sciences, is aiming at performing deep semantic analysis of natural language texts. The analyzer includes a powerful linguistic processor and various linguistic and extra-linguistic resources – a combinatorial dictionary, an ontology, a fact base, a set of inference rules and an inference engine [3, 5-8].

One of the applications of SemETAP is a question-answering system able to answer questions for which there is no direct answer in the original text. Consider a simple sentence:

(1) John sold an umbrella to Peter

We expect from the deep understanding system that, given this sentence as an input, it will be able to answer at least the following questions:

- (2)
- a. Who bought the umbrella? (Peter)
  - b. What did John give to Peter? (the umbrella)
  - c. Who owns the umbrella? (Peter)
  - d. What did John get? (money)

These are easy questions for a human although there is no direct answer in the original sentence. We answer them easily because we all know what 'sell', 'buy', 'give', 'own' and 'get' mean. This is part of our common knowledge.

---

<sup>1</sup> This paper presents the results of a joint effort of the team of the Laboratory of Computational Linguistics of the Institute for Information Transmission Problems of the Russian Academy of Sciences. The team includes I. Boguslavsky, L. Iomdin, A. Lazursky, S. Timoshenko, T. Frolova, V. Dikonov, E. Inshakova, V. Sizov and others. I would like to thank my colleagues for their wonderful collaboration.

<sup>2</sup> This work was supported by the RSF grant 16-18-10422, which is gratefully acknowledged.

SemETAP is aiming at formalizing this knowledge using semantic concept definitions. The semantic analyzer transforms the original text into a semantic graph consisting of semantic concepts from the ontology and relations between them. Then semantic concept definition rules are applied to extend the semantic graph with the implicit information. After that the semantic structure is ready to answer deep questions.

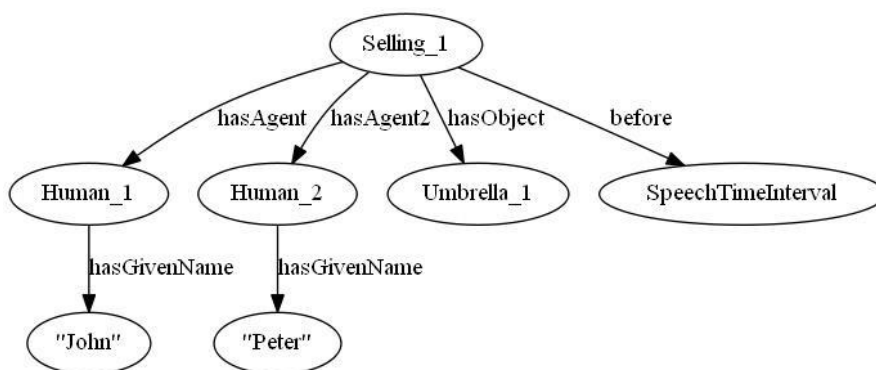
The goal of this paper is threefold: 1) show the current capabilities of the SemETAP inference engine; 2) identify topics for further research and 3) provide requirements for a reasoner that would satisfy our needs. For the reader's convenience we will take the simple example (1) as the input sentence for the case study and will show how the rule application process will lead us to the answers of the questions in (2)<sup>3</sup>.

The paper is structured as follows: Section 2 presents the basic semantic structure which SemETAP generates from the input sentence; Section 3 discusses the syntax of inference rules; Section 4 emphasizes the need for anonymous individuals in the rules; Section 5 provides implementation details of the inference engine; Section 6 discusses the representation of temporal relations, Section 7 shows the successive rule application process, Section 8 discusses issues related to facticity and proposition (un)certainly status, Section 9 identifies the need for disjunction, negation and universal quantification, Section 10 highlights the importance of anonymous individuals in the query results, Section 11 discusses the optimization issues, Section 12 summarizes our requirements for an ideal reasoner, and Section 13 concludes the paper.

## 2 Basic Semantic Structure

The semantic analyzer transforms the original text into a semantic structure consisting of a set of binary predicates (or RDF triples). Such a semantic structure can be represented as a semantic graph. A (simplified) graph and a set of triples corresponding to (1) are shown below<sup>4</sup>:

```
(3)  hasGivenName (Human_1, "John")
      hasAgent (Selling_1, Human_1)
      hasAgent2 (Selling_1, Human_2)
      hasObject (Selling_1, Umbrella_1)
      hasGivenName (Human_2, "Peter")
      before (Selling_1, SpeechTimeInterval)
```



<sup>3</sup> Other examples of concept definitions can be found in [4].

<sup>4</sup> Many names of the semantic concepts used (Human, Selling, Buying etc) are taken from SUMO ontology [18].

There are two types of nodes in the graph – concept nodes and literal nodes. A concept node represents an individual of a certain class (= semantic concept) from the ontology while a literal node contains just a string or numeric value.

A class of an individual is encoded in its name to make the semantic structure more readable. Human\_1 is of class Human by default. The class can also be stated explicitly when necessary as Human(Human\_1) or isA(Human\_1, Human).

In this example we have Selling\_1 object of the class Selling which represents the sale event. Human\_1 of the class Human represents the seller which is indicated with hasAgent relation. The buyer is marked with hasAgent2 relation and the product with hasObject. The last triple indicates that the event happened before the speech time interval i.e. in the past.

### 3 Semantic Concept Definition Rule Syntax

Once the basic semantic structure is built, it is time to apply inference rules to extend it. Most of the inference rules are concept definition rules, which decompose one concept into a connected set of other concepts. In our example we can apply the definition of Selling, which is shown below:

```
(4) Rule Selling:
    Selling(?selling)
Interdefinition:
    hasAgent(?selling, ?seller) & Agent(?seller) &
    hasAgent2(?selling, ?buyer) & Agent(?buyer) &
    hasObject(?selling, ?thing) &
    hasPrice(?selling, ?money) & CurrencyMeasure(?money) &
    sameTime(?buying, ?selling)
Definition:
    Buying(?buying) &
    hasAgent(?buying, ?buyer) &
    hasAgent2(?buying, ?seller) &
    hasObject(?buying, ?thing) &
    hasPrice(?buying, ?money).
```

This rule says that Selling is Buying with swapped arguments.

A rule consists of several sections. This rule has Body (the first section which follows the rule name), Interdefinition and Definition sections. In addition to these, a rule may also contain an Implication section. Each section consists of a conjunction of unary (classes) and binary (relations) predicates. Logical negation and disjunction are temporarily not supported, although we have plans to introduce them into the rule syntax. Variables (unlike constant individuals) are marked with a question mark.

The complicated rule structure follows from the wish to have everything related to a semantic concept in one place (as a ‘dictionary entry’ for that concept). In particular, a concept definition works both as a direct and an inverse implication. The logic of the rule sections application is the following:

```
(5) Body -> Interdefinition & Definition & Implication
    Definition -> Body & Interdefinition
```

Explicit quantifiers are not supported. All variables in the antecedent are treated as implicitly universally quantified. All new variables in the consequent (not present in the antecedent) are assumed to be existentially quantified. However, there is a demand for explicit universal quantifiers in the consequent which we plan to address in the future (see details in Section 9).

Body usually contains the concept being defined, Definition – its necessary and sufficient conditions which can be applied both ways, Implication – its one-way entailments. A special section is Interdefinition which is logically a part of the definition but nevertheless must be in the consequent of the inverse implication. For example, if we combine Interdefinition and Definition into one section in (4), then the inverse implication will work only if there is ?selling variable in the first place which does not make much sense. We want the inverse implication to be able to create the whole Selling object, not just assign Selling class to the object which is already there.

#### 4 Anonymous Individuals

Our rules allow new variables in the consequent which are not present in the antecedent. Since a concept definition is a detailed elaboration of the concept it must almost necessarily contain new variables to represent various parts of the concept definition. Such variables are known as existentials [9] or anonymous individuals.

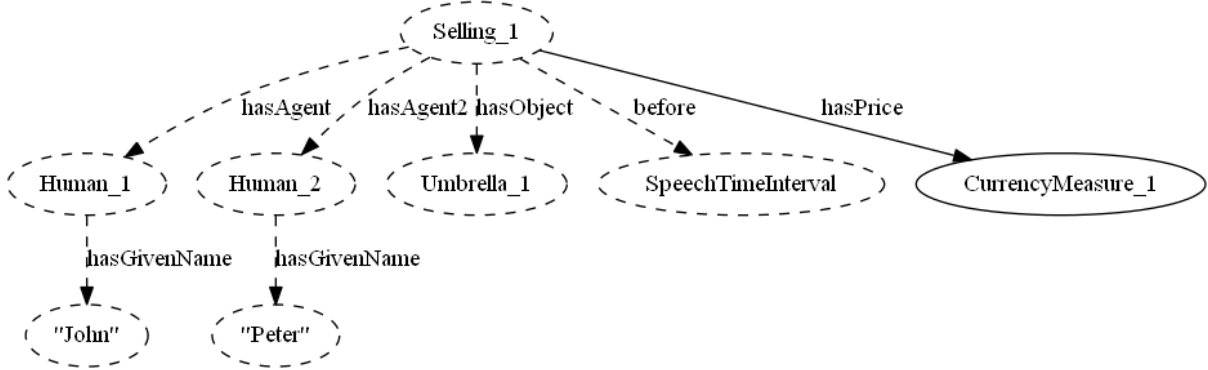
For example, we cannot define a concept without mentioning its arguments. Every sale involves a seller, a buyer, a product, and some amount of money. If any of these is missing then the situation cannot be called ‘selling’. Nevertheless not all of these four arguments are necessarily present in a sentence. In (1) there is no mention of money. In *The car is sold* three of the four arguments are missing. They may be missing in a sentence but they are still there in the situation the sentence describes. So, on the semantic level we have to restore all the missing arguments.

The first four lines of the Interdefinition of Selling in (4) serves for that purpose. It makes sure that all four arguments of sale are there in the semantic structure and assign some classes to them (Agent and CurrencyMeasure) if those classes are missing for some reason.

In the process of rule application we cannot just create new anonymous individuals for all new variables in the consequent. First we need to check if some of them are already there in the original semantic structure. The logic of checking for their existence can be complicated in the general case, but for semantic arguments it is simple. All semantic arguments are marked as a functional property in the ontology. So, if the argument property already has some value in the original structure then it just gets assigned to the existential variable and no new individual is created. In reality one sale event can include, say, multiple products, but this will be modelled using a separate group individual which will have a single hasObject relation to a sale event.

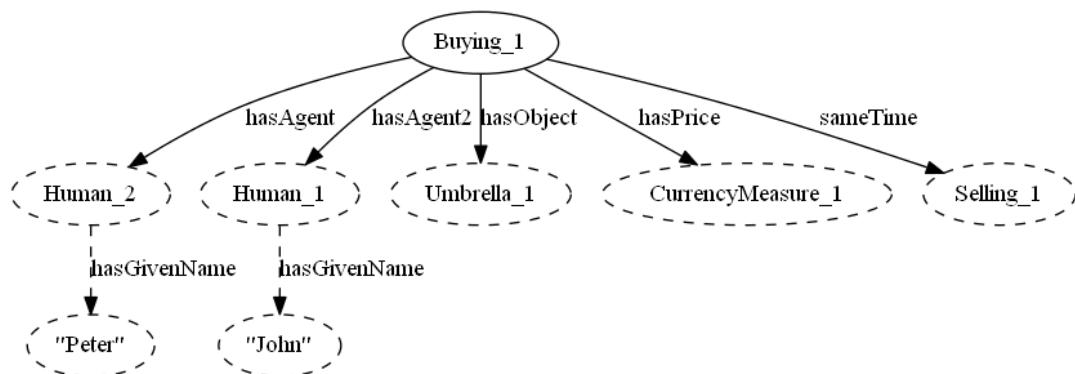
The application of the arguments in the Selling definition will introduce CurrencyMeasure\_1 individual to our semantic structure connected by hasPrice to the original Selling\_1 object (existing objects and relations are marked with dashed lines in the graph):

(6) hasPrice (Selling\_1, CurrencyMeasure\_1)



Now let's apply the rest of the rule (4). This part will generate a `Buying` object which has the same arguments, but `Human_1` and `Human_2` will swap their roles from `Agent` to `Agent2` and vice versa (Selling part of the graph is not shown for simplicity):

```
(7)  hasAgent (Buying_1, Human_2)
      hasAgent2 (Buying_1, Human_1)
      hasObject (Buying_1, Umbrella_1)
      hasPrice (Buying_1, CurrencyMeasure_1)
      sameTime (Buying_1, Selling_1)
```



But what will happen if we try to apply this rule once again? Will it create another `Buying` object? No. But how does it check for existence? After all `Buying_1` is not connected to any of the existing variables by a functional property. In this case we have a simple rule:

```
(8)  Do not add exactly the same subgraph that already exists.
```

By a 'subgraph' here we mean the largest possible subgraph ending on the existing individuals and including all of the anonymous individuals one of its nodes is connected to. This is not a logically perfect rule and not a technically efficient one. In some cases it can create duplicates. For example, if we had another `Buying` object in our original structure with the same arguments but missing `hasPrice`, it would not be matched and a new `Buying` object would be created instead.

So this is an open task to develop a logically consistent and efficient algorithm to match for the existing individuals before adding anonymous ones. One of the ideas to be explored is to look for *noncontradicting* individuals rather than *exactly the same* ones. For example, if we found a `Buying` object but it has a different buyer or seller then it is not a match. And if we found a `Buying` object that has some arguments matching and others missing then it is a potential match. Exact rules of noncontradiction still have to be developed. The task seems to be similar to the anaphora and coreference resolution issue which is also dealt with in our lab.

## 5 Implementation

The inference engine uses RDF storage to store the semantic structure and SPARQL language to apply rules and run queries.

Each rule gets compiled into a number of SPARQL insert statements. Each statement:

1. Searches for the rule antecedent pattern in the semantic structure.
2. Checks if the part of the consequent for which it is responsible is not present yet in the semantic structure.
3. (If 1 and 2 are satisfied) adds the part of the consequent for which it is responsible to the semantic structure, creating new individuals and/or RDF triples.

Thus, the inference engine uses the forward chaining method of reasoning.

The order of rules application is not determined. Rules are applied in several iterations, each of which runs all suitable rules. The application process stops when either the last iteration has not produced any new data or ten iterations have been completed. The latter condition is needed to avoid potential infinite recursion in the rule application process.

The question is processed in the following way:

1. A (basic) semantic structure of the question is built. The semantic structure of the question is similar to the semantic structure of the corresponding affirmative sentence but individuals corresponding to wh-words are marked in a special way.
2. This semantic structure is used as a pattern for a SPARQL select query which is run against the RDF storage returning individuals corresponding to wh-words in the question.
3. The results returned by the query are either displayed to the user in a table format or converted into a natural language sentence depending on the mode. In the latter case the text of the question is used as a pattern to generate the sentence of the answer.

Using RDF storage and SPARQL has its pros and cons. SPARQL is a powerful language which provides built-in capabilities for searching for a subgraph within the semantic structure. But its strength is also its weakness because we have almost no control over the search algorithm and very few possibilities to optimize its performance.

So we are contemplating the possibility of using one of the existing reasoners which would cover our needs, e.g. Graal [2], or, alternatively, rewriting our inference engine with the full control over the algorithms without using SPARQL.

## 6 Temporal Relations

At this point based on (7) we are almost able to answer the question (2a): “Who bought the umbrella?” The only thing we are missing is the indication that `Buying_1` happened in the past.

To represent event timeframes we adopted a slightly modified version of Allen’s temporal interval logic [1]. We use six Allen’s base temporal relations, some additional relations (including `sameTime` for equality) and a number of transitivity rules which form the composition table.

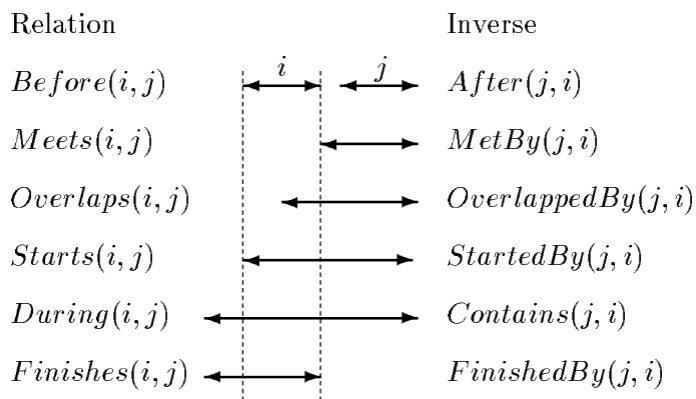


Figure 1. Allen’s base temporal relations.

The transitivity rule which we need in our case is as follows:

- (9) Rule `TimeSameTimeBefore`:
- ```

sameTime(?a, ?b) &
before(?b, ?c)
Implication:
before(?a, ?c).

```

Having `sameTime(Buying_1, Selling_1)` from (7) and `before(Selling_1, SpeechTimeInterval)` from (3) and (6) we can conclude:

(10) `before (Buying_1, SpeechTimeInterval)`

Now structures (7) and (10) provide us with all the information we need to answer the question (2a).

## 7 Going Deeper

Now we can apply the definition of Buying:

(11) Rule Buying:

`Buying(?buying)`

Interdefinition:

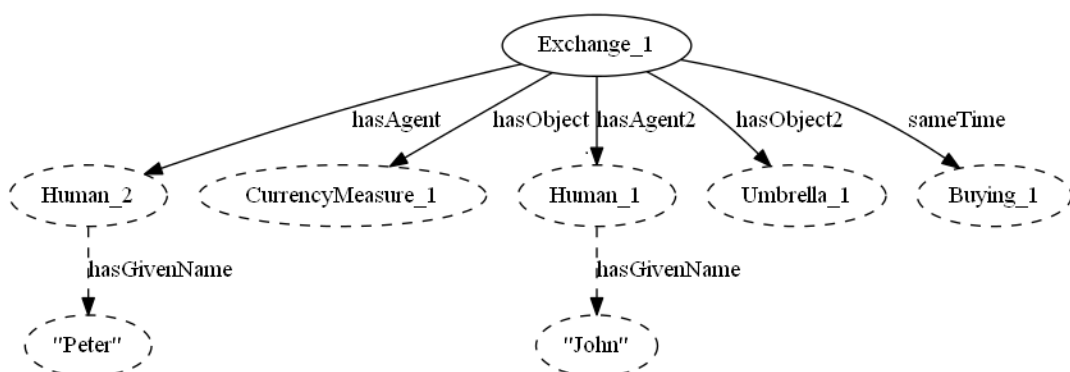
`hasAgent(?buying, ?buyer) & Agent(?buyer) &`  
`hasAgent2(?buying, ?seller) & Agent(?seller) &`  
`hasObject(?buying, ?thing) &`  
`hasPrice(?buying, ?money) &`  
`sameTime(?exchange, ?buying)`

Definition:

`Exchange(?exchange) &`  
`hasAgent(?exchange, ?buyer) &`  
`hasObject(?exchange, ?money) & CurrencyMeasure(?money) &`  
`hasAgent2(?exchange, ?seller) &`  
`hasObject2(?exchange, ?thing).`

This rule says that Buying is an Exchange for money. It is structured in a way similar to the Selling rule, and it will add the following data to our structure:

(12) `hasAgent (Exchange_1, Human_2)`  
`hasObject (Exchange_1, CurrencyMeasure_1)`  
`hasAgent2 (Exchange_1, Human_1)`  
`hasObject2 (Exchange_1, Umbrella_1)`  
`sameTime (Exchange_1, Buying_1)`



Applying the temporal transitivity rule (9) again, we can conclude:

(13) `before (Exchange_1, SpeechTimeInterval)`

This does not provide us with new information needed to answer questions (2b) – (2d), so we proceed to the definition of Exchange:

(14) Rule Exchange:

`Exchange(?exchange1)`

Interdefinition:

```

hasAgent(?exchange1, ?agent) & Agent(?agent) &
hasObject(?exchange1, ?object) &
hasAgent2(?exchange1, ?agent2) & Agent(?agent2) &
hasObject2(?exchange1, ?object2) &
during(?giving1, ?exchange1) &
during(?giving2, ?exchange1)

```

Definition:

```

Giving(?giving1) &
  hasAgent(?giving1, ?agent) &
  hasRecipient(?giving1, ?agent2) &
  hasObject(?giving1, ?object) &
Giving(?giving2) &
  hasAgent(?giving2, ?agent2) &
  hasRecipient(?giving2, ?agent) &
  hasObject(?giving2, ?object2)

```

Implication:

```

Exchange(?exchange2) &
hasAgent(?exchange2, ?agent2) &
hasObject(?exchange2, ?object2) &
hasAgent2(?exchange2, ?agent) &
hasObject2(?exchange2, ?object) &
sameTime(?exchange2, ?exchange1).

```

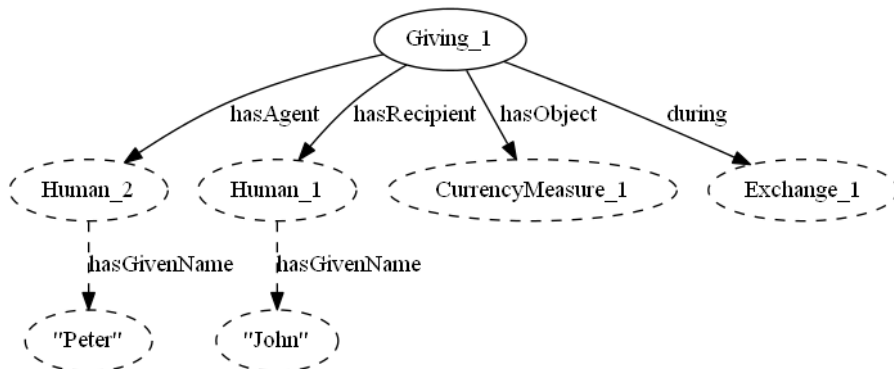
This rule defines Exchange as two mutual Givings. It also contains an Implication section which introduces a symmetric Exchange event proposition. If John exchanged with Peter then it is also true that Peter exchanged with John.

This rule will add two Giving objects and one Exchange object to our semantic structure as shown below:

- ```

(15) hasAgent (Giving_1, Human_2)
      hasObject (Giving_1, CurrencyMeasure_1)
      hasRecipient (Giving_1, Human_1)
      during (Giving_1, Exchange_1)

```

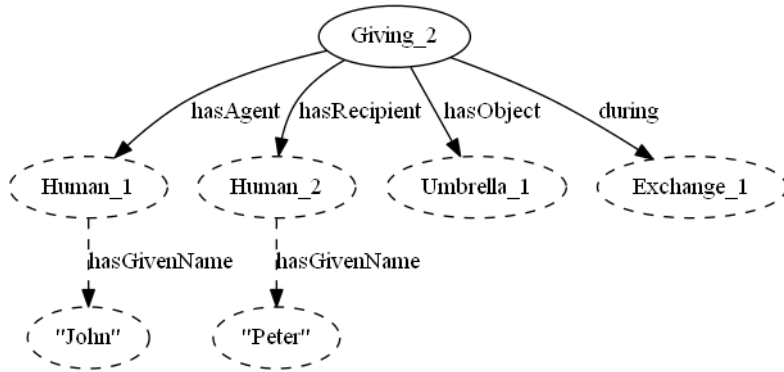


- ```

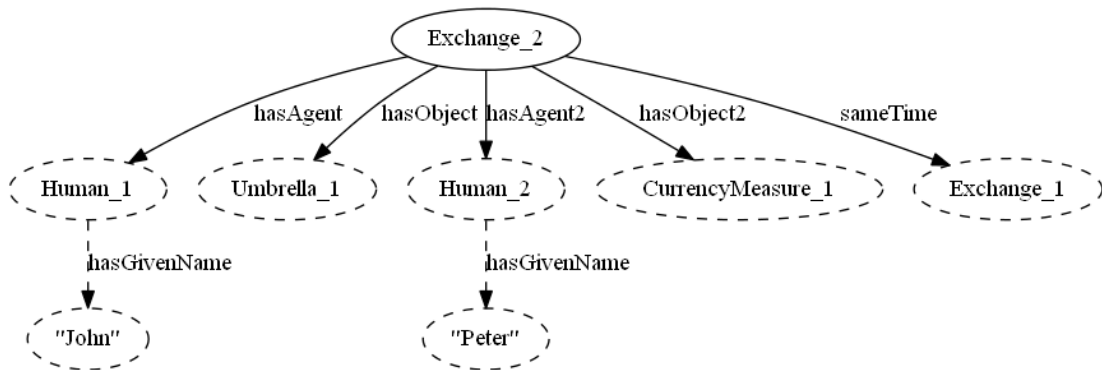
(16) hasAgent (Giving_2, Human_1)
      hasObject (Giving_2, Umbrella_1)
      hasRecipient (Giving_2, Human_2)
      during (Giving_2, Exchange_1)

```





- (17) hasAgent (Exchange\_2, Human\_1)  
 hasObject (Exchange\_2, Umbrella\_1)  
 hasAgent2 (Exchange\_2, Human\_2)  
 hasObject2 (Exchange\_2, CurrencyMeasure\_1)  
 sameTime (Exchange\_2, Exchange\_1)



Both Giving objects are connected to Exchange\_1 by during temporal relation. That means that they occurred within Exchange\_1 timeframe. We can apply another transitivity rule for them:

- (18) Rule TimeDuringBefore:  
 during(?a, ?b) &  
 before(?b, ?c)  
 Implication:  
 before(?a, ?c).

Resulting in:

- (19) before (Giving\_1, SpeechTimeInterval)  
 before (Giving\_2, SpeechTimeInterval)

The subgraph in (16) plus (19) gives us the answer to (2b) “What did John give to Peter?”

## 8 Facticity and Uncertainty

Now we can apply the definition of Giving:

- (20) Rule Giving:  
 Giving(?giving)  
 Interdefinition:  
 hasAgent(?giving, ?agent) & Agent(?agent) &  
 hasRecipient(?giving, ?agent2) & Agent(?agent2) &  
 hasObject(?giving, ?object)

Implication:

```

Own(?own1) &
  hasAgent(?own1, ?agent) &
  hasObject(?own1, ?object) &
  meetsTemporally(own1, ?giving) &
Own(?own2) &
  hasAgent(?own2, ?agent2) &
  hasObject(?own2, ?object) &
  meetsTemporally(?giving, own2).

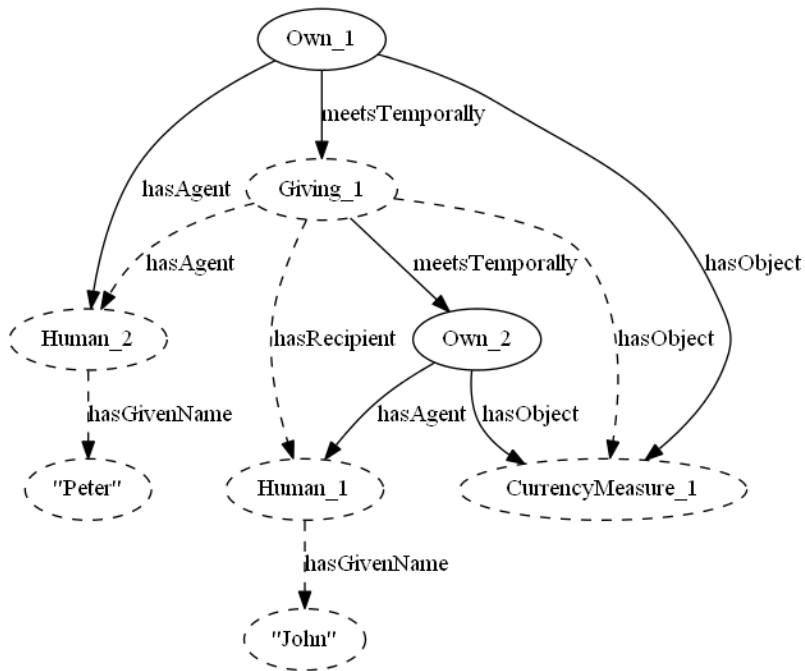
```

This rule says that Giving results in changing the owner. The rule has an Implication section and no Definition because not every change of the owner is Giving. The Interdefinition section without Definition works in the same way as Implication, but it is preserved here for visual split between the arguments and the other type of content. The application of this rule will add four Own events (two for each Giving):

```

(21) meetsTemporally (Own_1, Giving_1)
    hasAgent (Own_1, Human_2)
    hasObject (Own_1, CurrencyMeasure_1)
    meetsTemporally (Giving_1, Own_2)
    hasAgent (Own_2, Human_1)
    hasObject (Own_2, CurrencyMeasure_1)

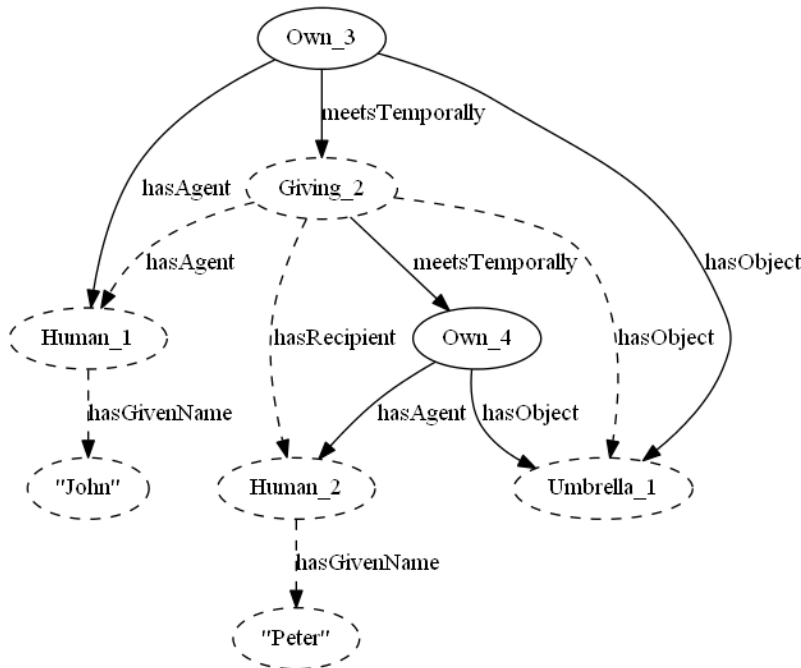
```



```

(22) meetsTemporally (Own_3, Giving_2)
    hasAgent (Own_3, Human_1)
    hasObject (Own_3, Umbrella_1)
    meetsTemporally (Giving_2, Own_4)
    hasAgent (Own_4, Human_2)
    hasObject (Own_4, Umbrella_1)

```



The predicate `meetsTemporally` means that two events follow each other, one event starts right after the other finishes. By applying another transitivity rule:

(23) Rule `TimeBeforeMeets2`:  
`before(?a, ?b) &`  
`meetsTemporally(?a, ?c)`  
 Implication:  
`startsBefore(?c, ?b).`

We can conclude that `Own_2` and `Own_4` events started in the past:

(24) `startsBefore (Own_2, SpeechTimeInterval)`  
`startsBefore (Own_4, SpeechTimeInterval)`

Do we now have enough information to answer the question (2c) “*Who owns the umbrella?*” Yes and no. Strictly speaking, this question does not have a definite answer. If Peter started owning the umbrella sometime in the past he could since then sell it again or lose it somewhere or pass on to somebody else, etc. We cannot be sure that he still owns it *now*. But without any additional information we choose to think that he does. This is called *implicature* [12], or *invited inference* [10]. This pragmatic phenomenon is based on the expectation that if it were not so then the speaker would tell that in an explicit way.

There are also other types of uncertain inferences which we generalize under the term (*plausible*) *expectations*. Expectation is a proposition that is likely to be true based on the information we have so far. It is not strict entailment, so it can be confirmed or disproved by a subsequent discourse.

SemETAP is able to capture plausible expectations at the level of individuals. Event objects can be assigned to special predicates `Fact` or `Implicature` to mark their certainty status. It is also possible that some event objects get assigned neither of the two. Consider the example:

(25) Peter wants to buy an umbrella.

The semantic structure of this sentence will contain predicate `Fact (Want_1)` (by the rule: “Trust the speaker”), but will contain neither `Fact (Buying_1)` nor `Implicature (Buying_1)` since the facticity status of buying is not asserted in the sentence.

Can we use `Implicature` predicate to mark the expectation that Peter still owns the umbrella? Unfortunately, no. We could assign `Implicature` to `Own_4` but it would not be what we need. Moreover, `Own_4` must have already been assigned the `Fact` predicate inherited through the rule application process. It is a fact (if we trust the speaker) that Peter started to own the umbrella sometime in the past. What we need to assign `Implicature` to is not the whole `Own_4` event, but the proposition that this event persists till now. This proposition can be expressed with the following predicate:

```
(26) during (SpeechTimeInterval, Own_4)
```

But our system does not support (yet) having a proposition or a predicate as an argument of another predicate. This is a topic for further development: how to model such a possibility within RDF model and process it consistently.

## 9 Disjunction, Negation and Universal Quantifier

Another type of uncertainty is a situation when the agent does not know a particular thing exactly, but knows the (closed) set of alternatives to which this thing can belong. This knowledge can be represented by a disjunction. An apple can be green, yellow or red. If we encounter an apple in the text we cannot be sure about its color, but at least we can be sure that it is not blue. A soccer pass can be performed with the player's leg, head or chest, but not with the arm etc.

This leads to the demand for the ability to include disjunctive information (possible alternatives) in the semantic structure and to allow disjunction in the consequent of the implication rules. This feature is not yet supported by our system mainly because of the difficulties concerning the introduction of disjunction into the RDF model.

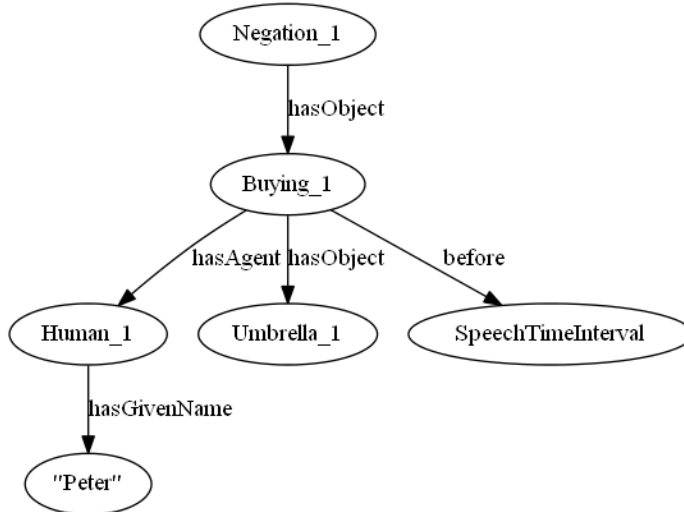
Another issue is the representation of the negative knowledge. We are working in the open world paradigm and cannot rely on the negation as failure. If some proposition is missing in the semantic structure then its status cannot be assumed to be false by default. So, if we need to be able to capture negative sentences such as the following:

```
(27) Peter did not buy an umbrella
```

we need an explicit way to represent negation. This is partially supported in our system through the use of `Negation` predicate. Similar to `Fact` and `Implicature` it can be assigned only to event individuals, not to propositions. It denotes the absence of the whole event. The example (27) will be represented as follows:

```
(28) hasGivenName (Human_1, "Peter")
      hasObject (Negation_1, Buying_1)
      hasAgent (Buying_1, Human_1)
      hasObject (Buying_1, Umbrella_1)
```

before (Buying\_1, SpeechTimeInterval)



In this example the Fact predicate will be assigned to Negation\_1, not to Buying\_1 (the absence of Peter's purchase is the fact). Negation, as any other predicate, can be used in a nonfactive context as well, as in the sentence: *Peter promised not to buy an umbrella.*

Since the scope of the Negation predicate is fixed to the whole event, it cannot cover all the situations when the negation is needed. Consider the following sentence:

(29) It was not Peter who bought the umbrella from John

In this example the scope of negation is only the following predicate:

(30) hasAgent (Buying\_1, Human\_1)

As we mentioned in the previous section we cannot have a predicate or a proposition as an argument of another predicate. So, we cannot represent (29) correctly.

Another problem with Negation is that anonymous individuals within the scope of negation must be universally quantified, but our system support only default existential quantification. The sentence (27) entails that Peter did not buy an umbrella *from anybody*. Existential quantification leads to the wrong understanding that *there is somebody* from whom Peter did not buy an umbrella. It will not produce definite answer to the question "*Did Peter buy an umbrella from John?*" since John and the person from whom Peter did not buy an umbrella do not necessarily coincide.

## 10 Anonymous Individuals in Answers

We are left with the last question (2d) "*What did John get?*". To answer it we need to apply the definition of Getting:

(31) Rule Getting:

Getting(?getting)

Interdefinition:

hasRecipient(?getting, ?agent) & Agent(?agent) &  
 hasSource(?getting, ?agent2) & Agent(?agent2) &  
 hasObject(?getting, ?object) &  
 sameTime(?getting, ?giving)

Definition:

Giving(?giving) &  
 hasAgent(?giving, ?agent2) &

```

hasRecipient(?giving, ?agent) &
hasObject(?giving, ?object).

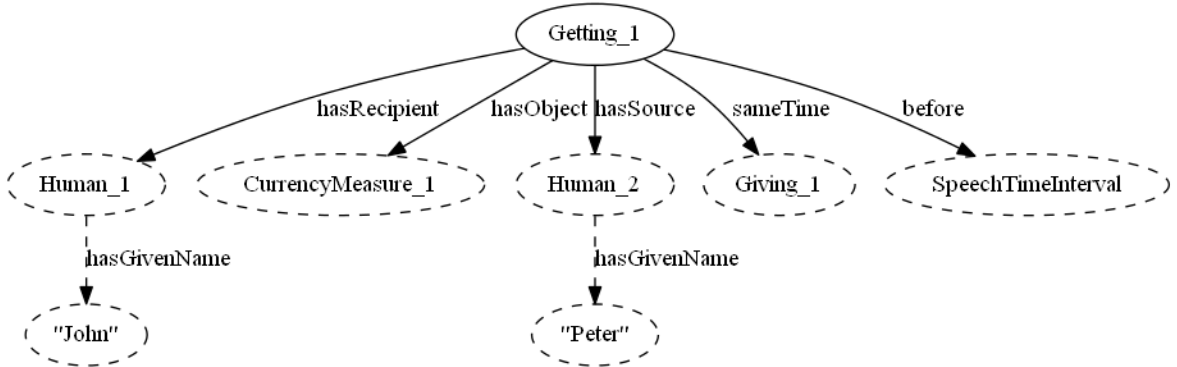
```

This rule says that Getting is a Giving with slightly different roles of the arguments. By applying the rule in the opposite direction followed by the transitivity rule (9) we will get:

```

(32) hasRecipient (Getting_1, Human_1)
      hasObject (Getting_1, CurrencyMeasure_1)
      hasSource (Getting_1, Human_2)
      sameTime (Getting_1, Giving_1)
      before (Getting_1, SpeechTimeInterval)

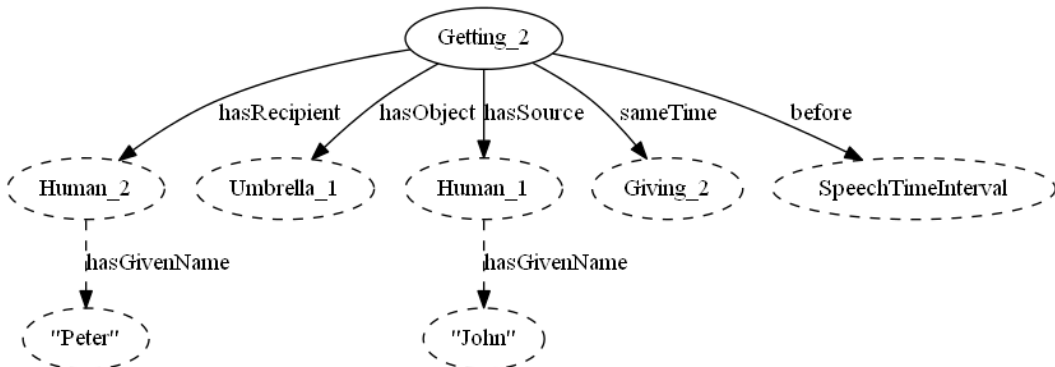
```



```

(33) hasRecipient (Getting_2, Human_2)
      hasObject (Getting_2, Umbrella_1)
      hasSource (Getting_2, Human_1)
      sameTime (Getting_2, Giving_2)
      before (Getting_2, SpeechTimeInterval)

```



The subgraph in (32) is what we need to answer (2d). Note that the answer is CurrencyMeasure\_1 which is anonymous individual created by the inference rules. Nevertheless it is a meaningful answer since we know its class. We do not know which particular amount of money John got but we can be sure that it was money and not something else. This can be useful information for subsequent inferences. This example shows that anonymous individuals are worth being returned by queries but only with (some) predicates involving them (in this example their class).

## 11 Optimization Issues

In this case study we used simplified semantic structure and simplified rules, without modality and other implications that can be made from the concepts we examined. And we even did not touch the definition of other concepts in our semantic structure such as Human, Umbrella and

CurrencyMeasure. Nevertheless we had to go five levels deep through the rule application process to find the answers for questions which a human can answer within a second. And our semantic structure has grown significantly.

This shows that probably forward chaining is not the best algorithm from the performance point of view especially if we aim at processing large texts with many different concepts involved.

The alternative would be to use backward chaining and one of the query rewriting algorithms [11, 13, 15-16]. Instead of modifying the data, such algorithms modify the query extending it with new propositions so the query can be successfully run against the original data.

We will explore this option, but limitations of this method are already visible. For example, the backward chaining would not give us the answer for (2d) “*What did John get?*” because there is no CurrencyMeasure object in the original semantic structure. Also, there could be other tasks involving deep semantic analysis and not related to queries such, as semantics-based translation or making an abstract of a text.

It might be sensible to build a hybrid system using both forward and backward chaining [14, 17, 19]. Yet the main question will be where and how to stop forward chaining in order not to overload the semantic structure with lots of unnecessary data. We do not have a definite answer to this challenge yet. But it needs to be emphasized that we are not trying to build a data-mining system or something similar. We are trying to build a system which would be comparable to humans in the performance of reasoning and not necessarily would outperform them. So the answer may lie in the pragmatically induced constraints (e.g. limiting the depth of the forward chaining) rather than in the theoretical ones (such as limiting the expressive power of the rule language).

## 12 Summary: Requirements for the Reasoner

An ideal reasoner for our needs has to support the following:

1. Anonymous individuals in the consequent of the rule with an efficient “find or create” algorithm for them.
2. Anonymous individuals in the result of a query with all or selected propositions involving those individuals.
3. Representation of uncertainty and the ability to apply it not only to individuals but to a particular predicate or proposition.
4. Disjunction, negation and universal quantification in the consequent of the rules.
5. Selected and directed reasoning depending on the task.

## 13 Conclusion

In a case study we have demonstrated the process of the concept definition rules application in the SemETAP inference engine, its capabilities and limitations, discussed various issues associated with the inference process, and specified the requirements for an ideal reasoner that would suit our needs.

A collection of semantic concept definition rules of a particular knowledge domain can potentially be used to build benchmark tests for the task of conjunctive query answering under existential rules.

## References

1. Allen, J.F. and Ferguson, G. 1994. Actions and Events in Interval Temporal Logic, *J. Logic and Computation* 4, 5, 1994.

2. Baget J.-F. et al. Graal: A Toolkit for Query Answering with Existential Rules / J.-F. Baget, M. Leclère, M.-L. Mugnier, S. Rocher, and C. Sipieter. In: RuleML 2015 Conference, August 2015.
3. 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.
4. Boguslavsky I. 2017. Semantic Descriptions for a Text Understanding System. In: Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference “Dialogue” (2017), p. 14-28.
5. Boguslavsky I. et al. 2010. Interfacing the Lexicon and the Ontology in a Semantic Analyzer / I. Boguslavsky , L. Iomdin, V. Sizov, S. Timoshenko. In: COLING 2010. Proceedings of the 6th Workshop on Ontologies and Lexical Resources (Ontolex 2010), Beijing, August 2010, pages 67-76.
6. Boguslavsky I. et al. 2013. Semantic representation for NL understanding / I. Boguslavsky, V. Dikonov, L. Iomdin, S. Timoshenko. In: Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference “Dialogue” (2013), p.132-144.
7. Boguslavsky I. et al. 2015. Semantic Analysis and Question Answering: a System Under Development / I. Boguslavsky, V. Dikonov, L. Iomdin, A. Lazursky, V. Sizov, S. Timoshenko. In: Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference “Dialogue” (2015), p.62-79].
8. Boguslavsky I. et al. 2016. Plausible Expectations-Based Inference for Semantic Analysis / I. Boguslavsky, V. Dikonov, T. Frolova, L. Iomdin, A. Lazurski, I. Rygaev, S. Timoshenko. In: Proceedings of the 2016 International Conference on Artificial Intelligence (ICAI"2016). USA: CSREA Press, 2016. pp. 477-483. ISBN: 1-60132-438-3.
9. Cali A. et al. 2010. Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications / A. Cali, G. Gottlob, Th. Lukasiewicz, B. Marnette, A. Pieris. Keynote Lecture. 2010 25th Annual IEEE Symposium on Logic in Computer Science.
10. Geis M. L. and Zwicky A. M. 1971. On Invited Inferences. *Linguistics Inquiry*, 2. 1971: 561–566.
11. Gottlob G. et al. 2014. Query Rewriting and Optimization for Ontological Databases / G. Gottlob , G. Orsi, A. Pieris. In: ACM Transactions on Database Systems (TODS), 2014.
12. Grice H. P. 1975. *Logic and Conversation. Syntax and Semantics*, Vol. 3, Speech Acts, ed. by Peter Cole and Jerry L. Morgan. New York: Academic Press 1975: 41–58.
13. Kikot S. et al. 2012. Conjunctive Query Answering with OWL2QL / S. Kikot , R. Kontchakov and M. Zakharyashev. KR-2012.
14. Kontchakov R. et al. 2011. The Combined Approach to Ontology-Based Data Access / R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyashev. In: Proceedings of IJCAI 2011.
15. König M. et al. 2012. A Sound and Complete Backward Chaining Algorithm for Existential Rules / M. König, M. Leclère, M.-L. Mugnier, M. Thomazo. Proc. of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012), September 2012.
16. König M. et al. 2013. On the Exploration of the Query Rewriting Space with Existential Rules / M. König, M. Leclère, M.-L. Mugnier, M. Thomazo. Proc. of the 7th International Conference on Web Reasoning and Rule Systems (RR 2013), July 2013.
17. Lutz C. et al. 2009. Conjunctive query answering in the description logic EL using a relational database system / C. Lutz, Toman, D., Wolter, F. In: Proc. of IJCAI 2009.



18. Niles I. and Pease A. 2001. Towards a Standard Upper Ontology. In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.
19. Zhou Y. et al. 2014. Pay-as-you-go OWL Query Answering Using a Triple Store / Y. Zhou, Y. Nenov, B. C. Grau and I. Horrocks. AAI 2014.