Building the Supremum of Semantic Schemas Endowed with Speech Synthesis Output

Claudia Alice STATE, Ionel IORGA

Faculty of Mathematics and Computer Science,
Department of Computer Science,
University of Craiova, Romania
cldstate@yahoo.com, ioneliorga@yahoo.com

Abstract. We made this study in order to evaluate a implementation which has, as output space for a semantic schema [1], speech synthesis. Speech synthesis as output of a inheritance-based knowledge system was treated in [11], for example. In literature, for semantic schemas, other output spaces where employed, for example, finite sets of strings corresponding to certain grammars rules of natural languages ([3],[6]), the 2D geometric space([6],[4]). In particular, in our approach, the output is generated in english language through the use of FreeTTS [7]. The implementation was made using an objectual language, a logic programming language and a connection between them. In order to make the elements of the interface the java language [9] was used. The computability regarding checking for consistency of a semantic schema, building the intermediary and final conclusions was made in swi-prolog logic environment [8]. The jpl connection [10] was used in order to made the java-prolog connection. The supremum of semantic schemas ([3],[4]) is built together with his associated interpretation, if the semantic schemas have interpretations. This mean that we provide an automatic method for building the supremum for any semantic schemas which are properly encoded.
Keywords: semantic schema, speech synthesis, text-to-speech, objectual programming, java prolog connection, java speech api, logic programming
Math. Subject Classification 2000:68T30, 68T35, 68Q55, 68N17

1 Introduction

The concept of **semantic schema** was introduced in Ţăndăreanu [1].
As in Ţăndăreanu-Ghindeanu [2] we understand by a *semantic $\theta-$schema* a system $\mathcal{S} = (X, A_0, A, R)$ which elements are denoted in the following manner:

- X a finite, non-empty set of symbols called object symbols
- A_0 a finite, non-empty set of elements called label symbols
- $A_0 \subseteq A \subseteq \overline{A_0}$ where $\overline{A_0}$ is the *Peano θ-algebra* [5] generated by A_0:
  $\hookrightarrow$ $\overline{A_0} = \bigcup_{n \geq 0} A_n$

$\hookrightarrow$ $A_{n+1} = \{\theta(u,v) \mid u,v \in A_n\}, n \geq 0$
$\hookrightarrow$ A_0 is a finite and a non-empty set
$\hookrightarrow$ θ a binary algebraic operation

- $R \subseteq X \times A \times X$ is a non-empty set which fulfills the following conditions:

$$(x,\theta(u,v),y) \in R \Longrightarrow \exists z \in X : (x,u,z) \in R, (z,v,y) \in R \tag{1}$$

$$\theta(u,v) \in A, (x,u,z) \in R, (z,v,y) \in R \Longrightarrow (x,\theta(u,v),y) \in R \tag{2}$$

$$pr_2 R = A \tag{3}$$

- where $pr_i R$ is defined in the following manner

$$pr_i R = \{x \in R_i | \exists (x_1,...,x_{i-1},x,x_{i+1},...x_n) \in R\} \tag{4}$$

if $R \subseteq R_1 \times ... \times R_n$ and $i \in \{1,...,n\}$

2 Interpretation of a semantic schema

Definition 1. (Ţăndăreanu [6] p. 33):
Consider an interpretation $\mathcal{I} = (Ob, ob, \{Alg_u\}_{u \in A})$ for $\mathcal{S}$. The output space Y of $\mathcal{S}$ is defined as follows:

$$Y = \bigcup_{u \in A} Y_u \tag{5}$$

where

$$Y_a = \{Alg_a(ob(x), ob(y)) \mid (x,a,y) \in R_0\}$$

if $a \in A_0$ and

$$Y_{\theta(u,v)} = \{Alg_{\theta(u,v)}(o_1,o_2) \mid o_1 \in Y_u, o_2 \in Y_v\}$$

where Alg is a set of algorithms with two input parameters and one output parameter and $R_0 = R \times (X \times A_0 \times X)$.

Remark.
If $\mathcal{I}$ is an interpretation of the semantic schema $\mathcal{S}$ then the output mapping (Ţăndăreanu [6] p. 34) is of the following form:

$$Out_{\mathcal{I}} = X \times X \to 2^Y$$

3 Building the supremum of schemas

In order to build the supremum of schemas, we use the following [4]:
Definition 3.
Consider the schemas $\mathcal{S} = (X, A_0, A, R)$ and $\mathcal{P} = (Y, B_0, B, Q)$. We define the relation $\mathcal{S} \sqsubseteq \mathcal{P}$ if $X \subseteq Y$ and $R \subseteq Q$.
Proposition 1.
Consider the schemas $\mathcal{S}_i = (X^i, A_0^i, A^i, R^i)$ such that A^i is finite, $i \in \{1, .., n\}$. We denote $X = \bigcup_{i=1}^{n} X^i$, $A_0 = \bigcup_{i=1}^{n} A_0^i$, $A = \bigcup_{i=1}^{n} A^i$ and
$R_0^i = R^i \cap ((X^i \times A_0 \times X^i)$ for $i \in \{1, .., n\}$.
The following sets are defined recursively:

$$\begin{cases} Z_0 = R_0^1 \cup ..... \cup R_0^n \\ Z_{j+1} = Z_j \cup \{(x, \theta(u, v), y) \in X \times A \times X | \exists z : (x, u, z) \in Z_j, \\ (z, v, y) \in Z_j\}, j \geq 0 \end{cases} \tag{6}$$

If the semantic schemas have interpretations the interpretation of their supremum is built upon their interpretation.
To be more exactly, the functions *ob* are compared and a new family of algorithms $\{Alg_u\}_{u \in A}$ is built. The resulted function *ob* should be also a bijective one.
The supremum of semantic schemas has as components the set $\{X, A_0, A, R\}$ and as an interpretation $\mathcal{I} = (Ob, ob, \{Alg_u\}_{u \in A})$, where $R = \bigcup_{n \geq 0} Z_n$.
Remark.
It seems that from the definition of R, for the supremum of semantic schema, determining it may result in an infinite process of computation. Practically this is not true because we use, in our implementation, a fix point theorem like method to ensure computation in finite steps.
Moreover from theoretical point of view, in [3] p.2 it is proved that $\exists n_0 \in \mathcal{N}$ such that

$$Z_0 \subset Z_1 \subset ... \subset Z_{n_0} = Z_{n_0+1} = ... \tag{7}$$

4 Implementation

The implementation of the deduction engine was made in a logic programming language of type prolog and the interface was realised in a cross-platform object oriented language.
In particular the **swi-prolog language** [8] was employed for the logic derivations and **java language** [9] was used for managing the user interactions with the interface.
A **java-prolog connection** was used in order to take the inference results of the logic program implemented in swi-prolog. In order to implement the java-prolog connection the **JPL** [10] package was used. The JPL package comes with the standard distribution of swi-prolog.
The swi-prolog language is a free, open source and cross-platform, so the resulted java-prolog connection will also be.

The java - swi prolog connection is a good tool, to build, and approximate solutions of NP-hard [14] problems because uses the combined advantages of the logic and objectual programming. Such a connection was used, for example, to offer a solution to the problem of generating an universitary timetable in [13].
In order to manage the steps in building and running the application we use **Apache Ant** [12] which is also, free, open-source and cross platform.
The documentation and regarding the java-swi prolog connection can be found in swi-prolog installation directory, for example on windows systems the path is *pl/doc/packages/examples/jpl*.
The locations of the java compiler and the swi-prolog executable must be found by adding them to the path variable of the operating system. On windows systems this can can be achieved by editing the AUTOEXEC.BAT file.

The part of the implementation written in swi-prolog application is modular consisting in:

- **cfg.pl** - It's a configuration file which containing the name of the knowledge base used in the application.
- **facts.pl** - Is the knowledge base containing the semantic schemas. A semantic schema is encoded through predicates which reflects the entities which express it: $\{X, A_0, A, R, Ob, ob, \{Alg_u\}_{u \in A}\}$
- **sem.pl** - Contains the predicates which verifies the conditions (1), (2), (3) and compute $Out_{\mathcal{I}}$

. A user may want to modify the the informations contained in the knowledge base files as long the rules from sem.pl remains true.

The java part consist from several packages as follows:

- *gui* - graphic user interface
  SemSchFrame - the principal class that manages the interface
  ChooseSch - implement a customized dialog. Instances of this class are created whenever the confirmation and, also, other informations from the user are needed.
- *sui* - speech user interface
  Speech - contains methods to transform a text into speech using the FreeTTS java package [7] which is an open-source implementation of the Java Speech Api [15]
- *semsch* - the principal package of the java part of the application
  SemSch - the main class containing an instance of a object of type gui.SemSchFrame
  Join - implements the necessary actions when the supremum of semantic schemas is computed
  JoinUtil - implements adjuvant methods to be called from the Join class
  SemSchPl- the class which contains the methods that make the calls from java to prolog logic program in order to extract the conclusions from the logic derivation using java prolog connection [10]

OpenKB - read the content of the knowledge base into java environment
SaveKB - save the modified content of the knowledge base (for example, after the supremum of semantic schemas is computed)
CloseKB - save and close the knowledge base
RUtil - methods for extracting elements of R in the java environment

- *util* - contains utility classes for workings with strings and the elements of a semantic schema
  StringUtil - utility methods over strings
  ParseOut and **ParseROut** - implements utility methods in the process of transforming the prolog of semantic schema entities representation to java

Through the use of this interface the advantages of a logic environment are combined with the use of a object oriented platform.
The application functionalities are the following:

- the semantic schema proposed by the user is verified to be consistent, in the sense of conditions (1),(2),(3), and, in order to compute $Out_{\mathcal{I}}$ (Ţăndăreanu [6]),the interpretation of semantic schema is employed together with the necessarily formulaes. The computed conclusions are displayed in a text area of the user interface (Fig. 5).
- user choose two or more semantic schemas and the application computes their supremum (Fig. 7)

By pressing the *Speech text* button (Fig. 4) **the finite subset** of the output space, corresponding to the results from the text area, is spoken.

5 Tests and results

5.1 Distributed knowledge and reasoning by analogy

The use of distributed knowledge and reasoning by analogy is presented, for example, in [3].
In the following example we consider two semantic schemas and we build their supremum putting in evidence the use of distributed knowledge and the reasoning by analogy.
First we consider semantic schema $\mathcal{S}_1$ from **Figure 1**:

$$X^1 = \{x_1, x_3, x_4, x_5\}$$
$$A_0^1 = \{a_1, a_2, a_3\}$$
$$A^1 = \{a_1, a_2, a_3, \theta(a_1, a_2), \theta(\theta(a_1, a_2), a_3)\}$$
$$R^1 = R_0^1 \bigcup R_1^1 \bigcup R_2^1$$
$$R_0^1 = \{(x1, a1, x3), (x3, a2, x4), (x4, a3, x5)\}$$
$$R_1^1 = \{(x1, \theta(a1, a2), x4)\}$$

$$R_2^1 = \{(x1, \theta(\theta(a1, a2), a3), x5)\}$$
$$\mathcal{F}_{comp}(\mathcal{S}_1) = \mathcal{P}_0^1 \bigcup \mathcal{P}_1^1 \bigcup \mathcal{P}_2^1$$
$$\mathcal{P}_0^1 = \{h(x1, a1, x3), h(x3, a2, x4), h(x4, a3, x5)\}$$
$$\mathcal{P}_1^1 = \{\sigma(h(x1, a1, x3), h(x3, a2, x4))\}$$
$$\mathcal{P}_2^1 = \{\sigma(\sigma(h(x1, a1, x3), h(x3, a2, x4)), h(x4, a3, x5))\}$$

where h and σ are symbols of arity 1 and, respectively 2 (Ţăndăreanu [3]), used for computing the intermediary set $F_{comp}(\mathcal{S}_1)$ in order to determine $Out_{\mathcal{I}^1}$.
We consider the following interpretation:
$\mathcal{I}^1 = (Ob^1, ob^1, \{Alg_u^1\}_{u \in A^1})$
$Ob^1 = \{John, Michelle, Dan, sportcar\}$
$ob^1 = \{(x_1, John), (x_3, Michelle), (x_4, Dan), (x_5, sportcar)\}$

- **Algorithm** $Alg_{a_1}^1(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of "+ob_2
- **Algorithm** $Alg_{a_2}^1(ob_1 : String, ob_2 : String)$
  return ob_1+" drives on the same car model of "+ob_2
- **Algorithm** $Alg_{a_3}^1(ob_1 : String, ob_2 : String)$
  return ob_1+" drives a "+ob_2
- **Algorithm** $Alg_{\theta(a_1,a_2)}^1(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of a person which drives on the same car model of "+ob_2
- **Algorithm** $Alg_{\theta(\theta(a_1,a_2),a_3)}^1(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of a person which drives on the same car model of a person which drives a "+ob_2

Remark:

the symbol "+" represents the binary operator of *concatenation* over strings

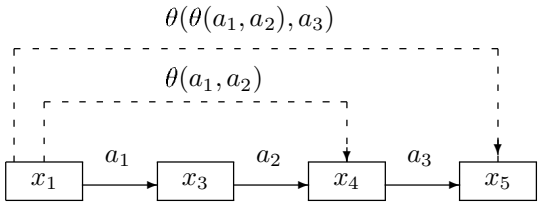

Figure 1 :Schema $\mathcal{S}_1$

The second semantic schema ($\mathcal{S}_2$) is presented in **Figure 2**:

$X^2 = \{x_1, x_2, x_4\}$

$A_0^2 = \{a_1, a_2\}$
$A^2 = \{a_1, a_2\}$
$R^2 = R_0^2$
$R_0^2 = \{(x1, a1, x2), (x2, a2, x4)\}$
$\mathcal{F}_{comp}(\mathcal{S}^2) = \mathcal{P}_0^2$
$\mathcal{P}_0^2 = \{h(x_1, a_1, x_2), h(x_2, a_2, x_4)\}$

If we consider the following interpretation $\mathcal{I}^2 = (Ob^2, ob^2, \{Alg_u^2\}_{u \in A^2})$
$Ob^2 = \{John, Clark, Dan\}$
$ob^2 = \{(x_1, John), (x_2, Clark), (x_4, Dan)\}$

- **Algorithm** $Alg_{a_1}^2(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of "+ob_2
- **Algorithm** $Alg_{a_2}^2(ob_1 : String, ob_2 : String)$
  return ob_1+" drives on the same car model of "+ob_2

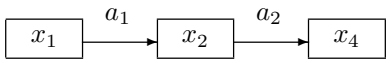

Figure 2 :Schema $\mathcal{S}_2$

In **Figure 3** the semantic schema ($\mathcal{S}_3$), corresponding to the supremum of $\mathcal{S}_1$ and $\mathcal{S}_2$ is presented:

$X = \{x_1, x_2, x_3, x_4, x_5\}$
$A_0 = \{a_1, a_2, a_3\}$
$A = \{a_1, a_2, a_3, \theta(a_1, a_2), \theta(\theta(a_1, a_2), a_3)\}$
$R = R_0 \bigcup R_1 \bigcup R_2$
$R_0 = \{(x1, a1, x2), (x1, a1, x3), (x2, a2, x4), (x3, a2, x4), (x4, a3, x5)\}$
$R_1 = \{(x1, \theta(a1, a2), x4)\}$
$R_2 = \{(x1, \theta(\theta(a1, a2), a3), x5)\}$
$\mathcal{F}_{comp}(\mathcal{S}) = \mathcal{P}_0 \bigcup \mathcal{P}_1 \bigcup \mathcal{P}_2$
$\mathcal{P}_0 = \{h(x_1, a_1, x_2), h(x_1, a_1, x_3), h(x_2, a_2, x_4), h(x_3, a_2, x_4), h(x_4, a_3, x_5)\}$
$\mathcal{P}_1 = \{\sigma(h(x_1, a_1, x_2), h(x_2, a_2, x_4)), \sigma(h(x_1, a_1, x_3), h(x_3, a_2, x_4))\}$
$\mathcal{P}_2 = \{\sigma(\sigma(h(x_1, a_1, x_2), h(x_2, a_2, x_4)), h(x_4, a_3, x_5))\}$

The corresponding interpretation is $\mathcal{I} = (X, ob, \{Alg_u\}_{u \in A})$
$Ob = \{John, Clark, Michelle, Dan, sportcar\}$
$ob = \{(x_1, John), (x_2, Clark), (x_3, Michelle), (x_4, Dan), (x_5, sportcar)\}$

- **Algorithm** $Alg_{a_1}(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of "+ob_2
- **Algorithm** $Alg_{a_2}(ob_1 : String, ob_2 : String)$
  return ob_1+" drives on the same car model of "+ob_2
- **Algorithm** $Alg_{a_3}(ob_1 : String, ob_2 : String)$
  return ob_1+" drives a "+ob_2
- **Algorithm** $Alg_{\theta(a_1,a_2)}(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of a person which drives on the same car model of "+ob_2
- **Algorithm** $Alg_{\theta(\theta(a_1,a_2),a_3)}(ob_1 : String, ob_2 : String)$
  return ob_1+" is a friend of a person which drives on the same car model of a person which drives a "+ob_2

$Out_{\mathcal{I}} = \{$"John is a friend of Clark", "John is a friend of Michelle", "Clark drives on the same car model of Dan", "Michelle drives on the same car model of Dan", "Dan drives a sport car", "John is a friend of a person which drives on the same car model of Dan", "John is a friend of a person which drives on the same car model of a person which drives a sport car"$\}$
We denote the fact that the last two conclusions are not obtained in a unique way.

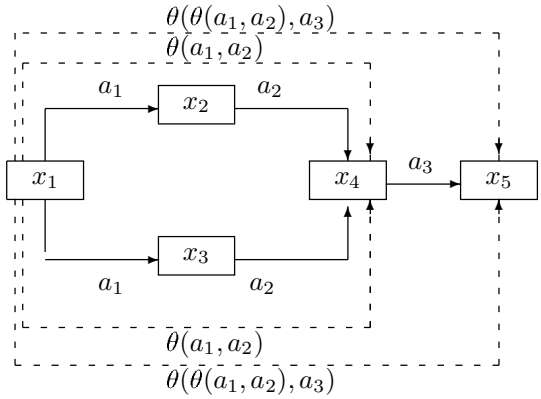

Figure 3 :Schema $\mathcal{S}_3$

The main functionalities of the expert system are presented in the following Figures:

- **Figure 4** - Represents the the dialog which ask for the the id of the semantic schema in order to compute $Out_{\mathcal{I}}$
- **Figure 5** - Shows the content of the kb file, in the upper text area, and the elements of the $Out_{\mathcal{I}}$, in the bellow text area.
- **Figure 6** - Presents the dialog which asks the number of semantic schemas, and their id's, for which the supremum is computed.
- **Figure 7** - Express the state of the interface after the supremum of the schemas is computed.

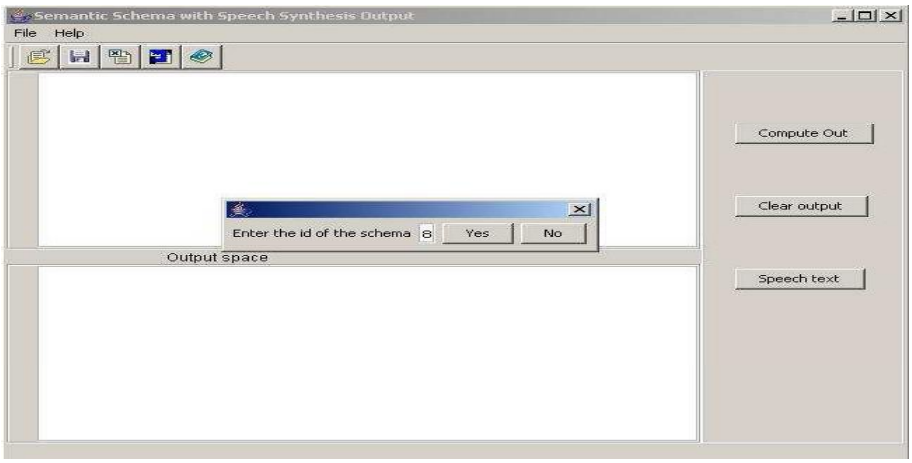

Figure 4

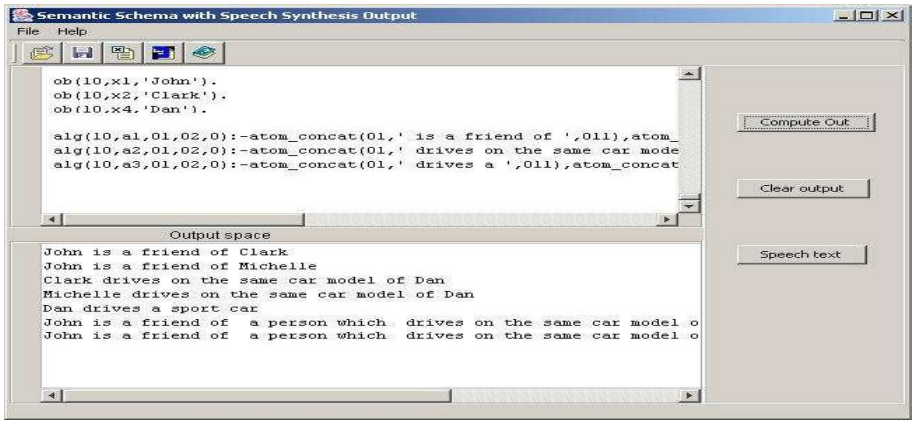

Figure 5

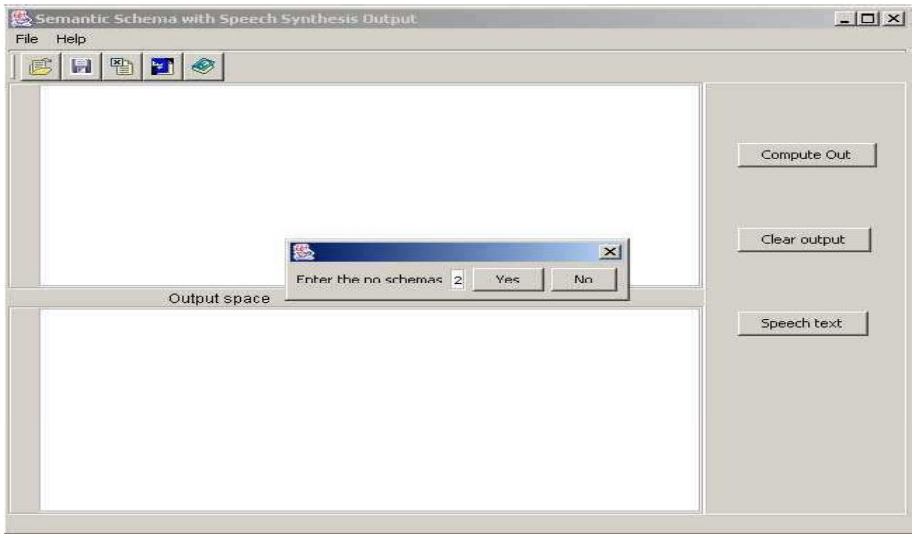

Figure 6

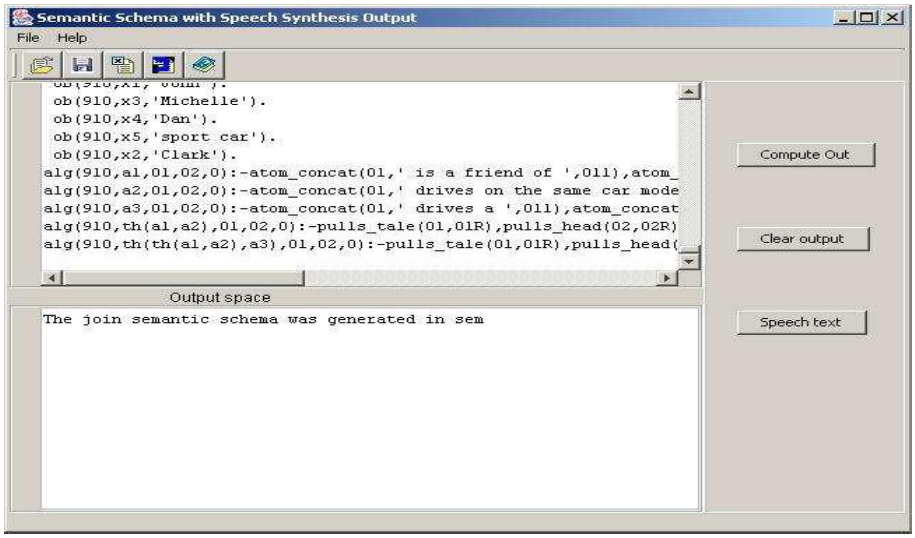

Figure 7

6 Conclusions and future work

The presented approach has proved her functionality. However, further implementations, must consider other functionalities like exporting in other formats the inference engine conclusions and the use as an output of a semantic schema the romanian natural language.
As part of future work we consider the interrogation by voice of the expert system which produces the output of the semantic schema in order to obtain a complete system of communication by voice.

References

[1] **N. Ţăndăreanu**:- Semantic Schemas and Applications in Logical Representation of Knowledge, Proceedings of The International Conference on Cybernetics and Information Technologies,Systems and Applications: CITSA 2004, Orlando, Florida.
[2] **N. Ţăndăreanu, M.Ghindeanu** :- Properties of derivations in a Semantic Schema, Annals of University of Craiova, Math. Comp. Sci. Series, 2006
[3] **N. Ţăndăreanu** :- Transfer of knowledge via semantic schemas, 9^{th} World MultiConference on Systemics, Cybernetics and Informatics, July 10-13, Vol. IV, 2005, 70-75
[4] **N. Ţăndăreanu, M.Ghindeanu** :- A three-level distributed knowledge system based on semantic schemas, 16^{th} Int. Workshop on Database and Expert Systems Applications, Proceedings of DEXA '05,Copenhagen, 2005, 423-427
[5] **V. Boicescu, A. Filipoiu, G. Georgescu, S. Rudeanu**:- Lukasiewicz-Moisil Algebras, Annals of Discrete Mathematics, 49, North-Holland, 1991
[6] **N. Ţăndăreanu**:- Knowledge Representation by Semantic Schemas, Technical Report, Research Center for Artificial Intelligence, Department of Computer Science, University of Craiova, 2006
[7] ***:- FreeTTS, http://freetts.sourceforge.net/docs/index.php
[8] ***:- Swi-Prolog, http://www.swi-prolog.org/
[9] ***:- Java Language, http://www.java.com/en/download/index.jsp
[10] ***:- JPL, http://www.swi-prolog.org/packages/jpl/java_api/index.html
[11] **N. Ţăndăreanu**:- VoSys: A System by Voice to Answer in Inheritance-Based Knowledge Systems, 6^{th} International Conference on Artificial Intelligence and Digital Communications,Thessaloniki, Greece, Vol. 106, August 2006, 91-101
[12] ***:- Ant, http://ant.apache.org/
[13] **I. Iorga**:- Genetic algorithms, Logic programming and Java - Prolog connection Applied to the University Timetable Scheduling, 5^{th} International Conference on Artificial Intelligence and Digital Communications, Craiova, Romania, Vol. 105, September 2005, 96-107
[14] ***:- http://en.wikipedia.org/wiki/NP-hard
[15] ***:- Java Speech Api, http://java.sun.com/products/java-media/speech/