A natural language interface for man machine communication in electrical networks control centers

MARIANO GONZÁLEZ ROMANO
Dpto. Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda Reina Mercedes s/n.
41012 Sevilla. SPAIN

EDUARDO FERNÁNDEZ CAMACHO
Dpto. Ingeniería de Sistemas y Automática
Universidad de Sevilla
Camino de los Descubrimientos s/n.
41092 Sevilla. SPAIN

Abstract: - This paper describes the application of a natural language interface to man-machine interaction in the control centers of electrical networks. Natural language communication with complex systems is a major goal, since it simplifies the operation of such systems by allowing their operators to command the system in their own language. This communication has to be flexible enough to allow the operators to perform simple operations as well as high level tasks which involve multiple elements of the network. For this ones, an adequate representation of the knowledge about the network will allow for the creation of a plan of simple actions whose execution will result in the fulfillment of the requested task.

Key-Words: - automatic control, computer interfaces, natural language, man-machine systems, real time computer systems, electric power systems *IMACS/IEEE CSCC'99* Proceedings, pp.1291-1296

1 Introduction

Complex interactive systems are present in a high variety of fields. In these systems, sophisticated control techniques are used in combination with human operators in order to satisfy some specific goals. The system operators act from a control room in a double manner: on one hand, they are informed of the state of the system at every moment and, on the other hand, they perform actions over the system in order to satisfy some goals. The difficulty of the operation depends on the complexity of the system, which comes from three main sources: the complexity of the domain, the complexity of control and the complexity of interaction. The first one can hardly be reduced. However, the complexity of control and interaction are able to be reduced. In the first case, with the automation of the system. In the second case, with the design of an adequate interface for the communication between the operator and the system. As completely automated systems are highly problematic, it becomes even more important the design of friendly interfaces which decrease the complexity of interaction and allows the operators to act in a simple and quick way in every kind of situations.

Although it can be said that, in general, man-machine interfaces of control centers are quite friendly, they have some drawbacks, such as the need of an extensive training period for operators and the great quantity of information, some of it irrelevant, they tend to present the operator. Natural language interfaces can be used in control centers to help the operator to manipulate the system by giving commands in his own language. A natural language interface (NLI) requires a short training period for the operator and gives him flexibility to focus attention in specific system elements or consider wider parts of the system, modifying the degree of abstraction in which the system is viewed as a function of its current state. Natural language can also be used to generate explanations about the causes of a certain situation. In conclusion, natural language can be, en general, of great interest as a tool to simplify the task of control centers operators.

2 Related work

Natural language interfaces have been applied to a great variety of fields, such as database access, knowledge acquisition, interfacing with expert systems, human-computer dialog or tutorial systems.

In the field of interaction with complex systems, there are some works on the application of a NLI to communication with robots. Several researchers have developed robotic systems that are controlled by means of natural language sentences [9] [11]. The use of natural language allows operations to be performed on robots and their environment and specify goals to be satisfied without the need for previous knowledge about robots or computers.

A common feature in the majority of systems designed for natural language communication with complex systems is that they are designed for a specific kind of system. The one proposed in this paper, instead, is designed to work with generic complex systems. It has already been applied to two different complex systems. The first application was the operation of a pilot plant [4]. Through the NLI, the plant operator was able to perform simple actions such as open or close valves, as well as complex goals such as filling a tank up to a given level or maintaining the temperature of a tank. The system was later applied to the teleoperation of a NOMAD 200 mobile robot [6], allowing the user to perform high level operations on a robot navigating in a partially structured environment.

3 Description of the NLI

The NLI is a part of a more complex system aimed at communicating in natural language with complex interactive systems [5]. This system uses a NLI to acquire the knowledge of a generic complex system and another NLI to operate it. The system is composed of two different parts: the acquisition module and the operation module. The acquisition module allows the user to describe a system by introducing phrases in his own natural language and stores the acquired knowledge in a database. By using this knowledge the operation module allows to operate the system with natural language commands.

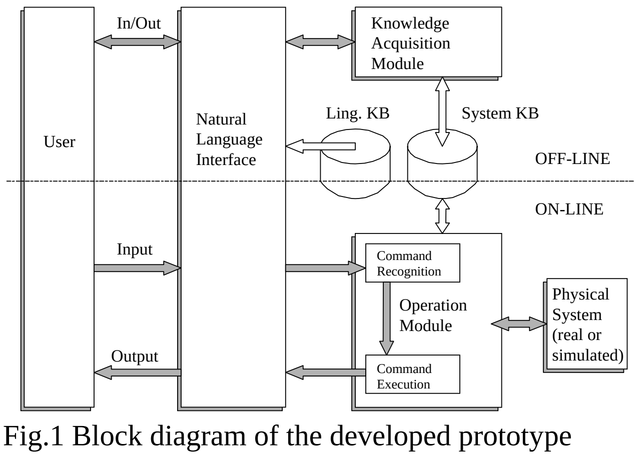


Fig.1 Block diagram of the developed prototype

Fig.1 shows the block diagram of the developed system, from which there exists a running prototype [3]. This prototype connects to the real system through its control system, which receives the commands from the prototype and executes them over the system. It is also possible to connect the prototype to a program which simulates the behaviour of the system when the connection is not possible or for operator training purposes.

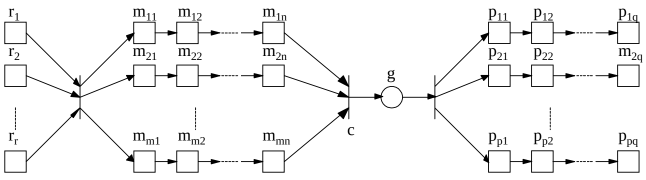
3.1 The Knowledge Database

The knowledge database consists of two parts: the linguistic knowledge database and the system knowledge database. The first one contains the linguistic knowledge that is necessary to analyze and extract the meaning of the user's phrases, and is composed of a dictionnary and a grammar. The second one contains all the necessary knowledge to describe the system, and can be divided into declarative and procedural knowledge. The first one concerns the different kinds of entities that can be found in the system: **classes**: (different kinds of objects present in the system), **objects** (particular instances of the classes), **connections** (topological relations between objects), **groupings** (groups of objects related to each other by their topology or their function) and **measurements**: (sensors allowing for some relevant magnitudes of the system to be known).

Procedural knowledge includes a set of functions that represent the means through which the goals of the system can be fulfilled. These functions are related to the different entities found in the system (classes, objects, groupings) or to the whole system. A function has the following components (Fig.2):

- **Goal** (g): is the goal fulfilled by the function.
- **Prerequisites** (r_i): are conditions that the system must accomplish before applying the means for the fulfillment of the goal. A prerequisite can be a condition on a state or the value of a property.
- **Means** (m_{ij}): are the operations whose execution results in the fulfillment of the goal of the function. They can be simple actions over objects or classes, or other functions. In general there will exist some sequences of means in parallel: some means will execute at the same time, as they are independent, while others will have to do it in sequence, as every means depends on the result of another.
- **Criteria** (c): is the criteria whose accomplishment implies that the goal of the function has been fulfilled. It is a condition over the value of some property, and will always be true provided that the means have been correctly executed.
- **Posterior actions** (p_{ij}): are operations that must be executed once the goal of the function has been fulfilled, in order to leave the system in a specific state. They can be simple actions or functions.

The execution of a function may need the previous execution of other functions that act as means of it, and may launch the execution of other functions that act as posterior actions of it. When executing a function, it will be decomposed into its constituents until there are any functions left, thus obtaining a network made of simple actions and conditions that willl be called the **actions network**.


r_1
r_2
r_r
m_{11} m_{12} m_{1n}
m_{21} m_{22} m_{2n}
m_{m1} m_{m2} m_{mn}
c
g
p_{11} p_{12} p_{1q}
p_{21} p_{22} p_{2q}
p_{p1} p_{p2} p_{pq}
Fig.2 Representation of a function

3.2 Knowledge Representation

The elected formalism for the representation of knowledge has been that of frames. This is due to the hierarchical organization of the knowledge, at the declarative (hierarchical structure of clases) and the procedural level (hierachical structure of functions). The knowledge database will be composed of class, object, grouping, measurement and function frames.

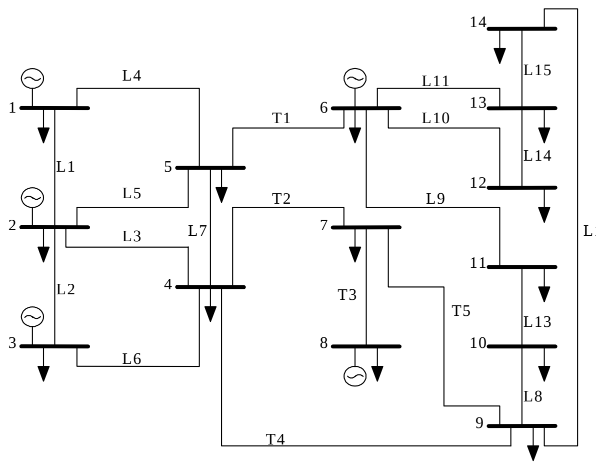


Fig.3 14 substations electrical network

4 Application to an Electrical Network

This prototype has been applied to the operation of an electrical network destined to the generation, transport and distribution of electric energy. The objective is to allow the operators of the control center of the network to perform through the NLI every operation they usually perform over the elements of the network: open/close a cell, feed/shed a load, etc. An interesting point of this application is its possibility of being used for operator training, simulating by software the behavior of the network.

The selected system is an electrical network made up of 14 interconnected substations (Fig.3) that is based on the IEEE 14 node net [7]. The substations have been specially designed for this work.

4.1 Network description

From the study of the typical components of electrical networks [1] [12] and electrical substations [8] results the structure of classes shown in Fig.4. The main class is `network_element`, which has two subclasses: `substation_element` and `power_element`. The `substation_element` class represents an element of the electrical network located inside a substation, and includes the subclasses `switching_element`, `cell` and `busbar`. The `power_element` class represents an element that consumes or generates power, and can be a `load`, a `generator` or a `capacitor`.

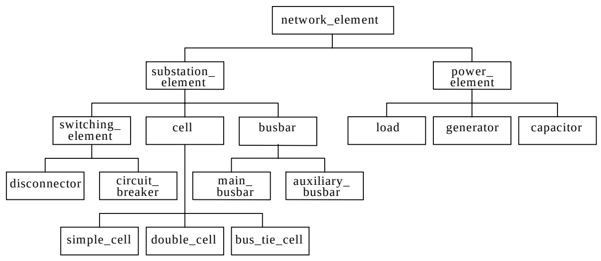


Fig.4 Class tree of the electrical network

Fig.5 shows the different connection classes in the network. There is a `connection_element` superclass with two subclasses: `main_connection`, which connects two objects from different substations, and `auxiliary _connection`, which connects two objects of the same substation. The first class includes the classes `line` and `transformer`.

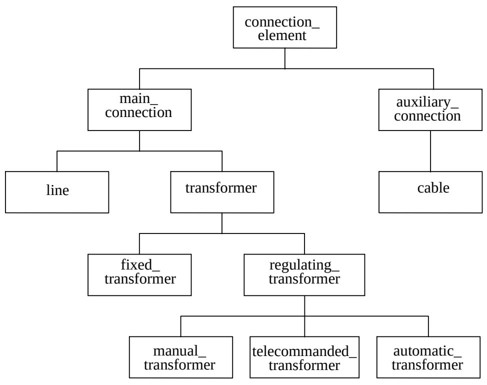


Fig.5 Class tree for the connections

Each substation is a grouping, so the system has 14 groupings that will be called `sub1`, ..., `sub14`. Finally, the functions that have been defined in the network are those related to its normal operation, and are shown in Fig.6. Level 0 functions correspond to simple actions defined over the classes. Higher level functions are based on those of the preceding levels.

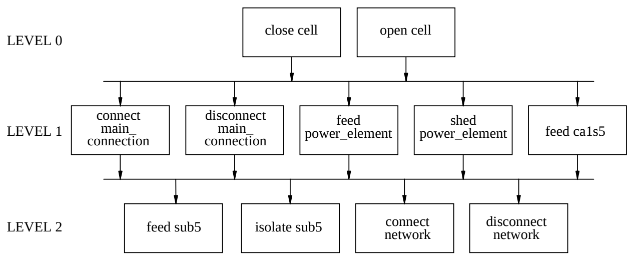


Fig.6 Structure of functions of the electrical network

4.2 Network operation

Owing to the difficulty of connecting the prototype to the real system it was decided to simulate the electrical network. In order to do this a simulation module that calculates the load flow of the network after analyzing the state of its elements was developed [2]. This module performs the following tasks:

- Topological analysis (correspondence between physical and electrical nodes, islands detection, dead islands elimination, optimal node ordering, computation of the node admittance matrix and initialization of node voltages).
- Load flow calculation (obtention of the load flow in every island, adjust of the transformer taps and surveillance of the reactive power limits).
- Inverse topological assignment (obtention of intensities in the lines, voltages in the busbars, consumed/generated power in loads/generators).

The simulation module is called for the first time at the begin of the session and is later activated every time the state of some switching element of the network changes, resulting in an updating of the network. The system operation is represented by the schema shown in Fig.7.

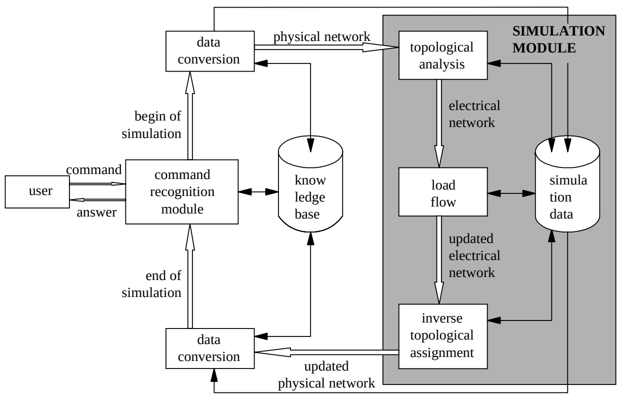


Fig.7 Schema of the electrical network operation

The control center operator can execute two kinds of commands over the network: simple and complex commands.

Simple commands are those which apply directly to an element or group of elements, and can be divided into two cathegories: action execution commands and information request commands. The first one includes commands with which the user requests the execution of a specific action over an object or set of objects. The following are examples of this kind of commands:

```
> open the cell C1S1.
> open the bus tie cell of generator G5.
> open the cell that joins G1 with busbar BC1S1.
> open the cells associated to line L1.
> close all the cells.
> open all the cells connected to the main busbar
of substation sub1.
> open all the cells from substation sub1.
```

Action execution commands imply the performing of a specific action over a specific object; the recognition process of these commands consists of identifying the action which has to be done and the object to which it has to be applied. Once this has been done, it should be checked whether the action can be applied to the object, and if the actual state of the object allows for the application of that action. If everything is correct, the action will be executed, and the object will be set to its new state.

Information request commands are those commands with which the user aks for a specific information about an object or a set of objects. For instance,

```
>show the voltage of the main busbar of sub1.
>show the active power of generator G1.
>show the state of all the cells of sub1.
>show the state of all the cells connected to the
main busbar BC1S1.
>show the active power of all the loads.
```

The requested information corresponds to the state or property value of an object or grouping. The recognition process of these commands consists of identifying the object or grouping whose state or property is to be known and, in this latter case, the corresponding property, which must be a valid property and must have an associated measurement.

Complex commands are those that imply the execution of a plan of actions in order to fulfill a specific goal. The plan of actions is a sequence of simple actions over some objects in a predetermined order, and is generated from the knowledge stored in the system knowledge database. Given a goal, there will exist in the knowledge database at least a function which will have this goal as its goal. The frame for this function will be the starting point to generate the plan of actions which allows to fulfill the command. Thus, the different means, prerequisites and posterior actions of this frame will be analysed one by one. Each means m_{ij} and posterior action p_{ij} from this function can be a simple or complex command. Every prerequisite r_i can be a complex command or a condition over a state or the value of a property. Simple and complex commands, as well as conditions, have a structure that is represented by a Petri Net [10]. Each structure has at least an input place and an output place. The input place will be marked when the net is activated whereas the marking of the output place will mean the ending of its traversal.

A simple command is represented by two places and one transition (Fig.8). The input place represents the action to be executed over the object to which the command is applied and will be marked when the command starts its execution. The transition represents the state to which the object is to be taken and will be fired when the action over it has been done. At this moment the output place representing the fulfillment of the simple command will be marked.

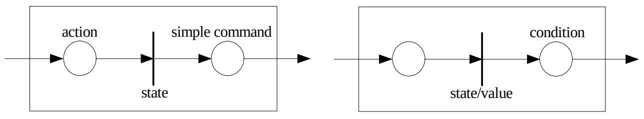


Fig.8 Structure of a simple command and a condition

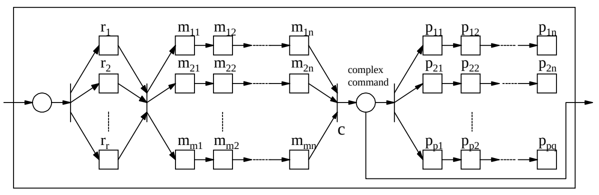


Fig.9 Structure of a complex command

For complex commands there will exist in the knowledge database another function whose goal corresponds to that command, and that will have its own means, prerequisites, criteria and posterior actions. The network representing this function will be a subnetwork of the one corresponding to the main function, and should in turn be expanded, should there have another complex command between its means, prerequisites and posterior actions. Its structure is shown in Fig.9. There is an input place which will be marked when the execution of the complex command begins, and this mark will propagate automatically to all the prerequisites. The output place of the net is the goal place. When the marks reaches this place the goal will be fulfilled.

Finally a condition type prerequisite is represented by two places and a transition, as it is shown in Fig.8. The input place will be marked when the evaluation of the prerequisite starts. The transition represents the desired state or property value, and will be fired when it has the adequate value, resulting in the marking of the output place.

If every means, posterior action and prerequisite is successively divided until there only are simple actions and conditions, a network, the actions network, will be obtained. Places in this network represent direct actions over objects and transitions represent conditions over the state of the objects or the value of their properties. The initial marking corresponds to the first simple actions to be executed and the first conditions to be checked. The marks will propagate as the transitions are fired, as a result of the accomplishment of the conditions, until the mark reaches the goal place. Fig.10 shows the generic structure of an actions network.

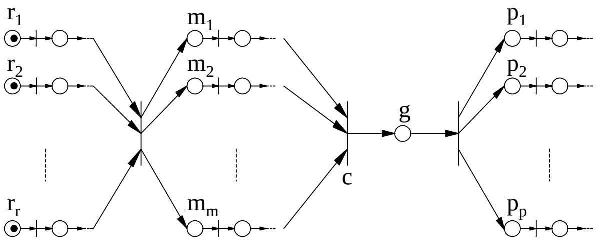


Fig.10 Structure of an actions network

The goal will be fulfilled when the place g is marked. In order for this to happen, condition c should be true when all of its input places are marked. This will happen once the corresponding means m_{ij} have been accomplished. The initial marking of these means will in turn depend on the marking of the ending places of the prerequisites r_i. On the other hand, once the goal g has been fulfilled its output transition will be fired marking the input places of the posterior actions p_{ij}. The initial marking of the network will correspond to the initial places of the prerequisites r_i, and will propagate towards the goal as transitions are fired.

To summarise, in order for a goal to be fulfilled there have to be accomplished, in the first place, all the requisites. These are, thus, necessary but not sufficient conditions for the fulfillment of the goal. Once the prerequisites are accomplished, the accomplishment of every means results in the fulfillment of the goal, provided that the criteria is true. The accomplishment of the means plus the criteria is, therefore, the necessary and sufficient condition for the main goal to be fulfilled.

5 Operation example

As an example of the execution of complex commands, it is shown the execution of the goal `feed_sub5`, which consist in feeding substation `sub5`. This is done by connecting lines `l4` and `l5`. As a previous requisite, substations `sub1` and `sub2` (from which lines `l4` and `l5` depart) must be alive. Table 1 shows the frame for this function and Fig.11 shows its representation.

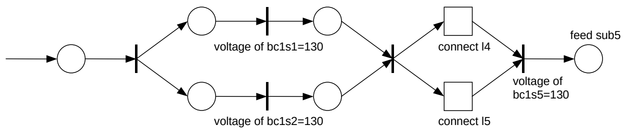


Fig.11 Structure of the function `feed_sub5`

NAME	feed_sub5
DESCRIPTION	Give voltage to substation sub5
ASOC_TYPE	Grouping
ASOC_NAME	sub5
PRE	(voltage of bc1s1 = 130 KV, voltage of bc1s2 = 130 KV)
MEANS	(connect line l4), (connect line l5)
CRITERIA	voltage of bc1s5 = 130 KV
POST	()

Table 1 Frame for the function `feed_sub5`

The execution of the goal `feed sub5`**,** provided that the prerrequisites have been accomplished, results in the creation and execution of the actions network shown in Fig.12. This network will later be reused if the goal is executed again.

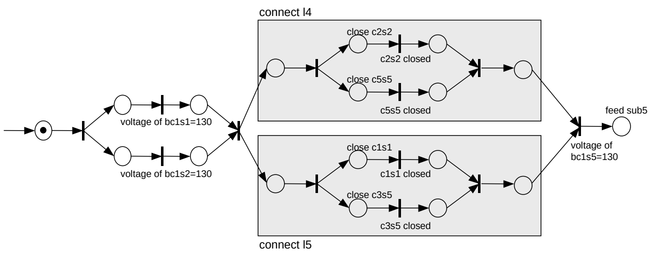


Fig.12 Actions network for the function `feed_sub5`

6 Conclusions

This paper has shown the application of a natural language interface to the operation of an electrical network. The adequate description of the system knowledge allows the user to perform high level operations, which are decomposed in other lower-level actions until there is a sequence of simple actions which are executed over the network, thus having a telescopic vision of the system. The developed prototype could be applied to simulate the execution of goals. In order to do this, it suffices to simulate the behaviour of the system through software and build the plan of actions corresponding to the goal. If the plan is successfully built, the goal can be achieved and could be executed over the real system. In other case, a different plan should be created. To summarise, it has been developed a tool which can simplify the operation of complex interactive systems, as the electrical network to which it has been applied in this paper.

References

[1] Franquelo, L.G. and A. Gómez Expósito, *Sistemas Eléctricos de Potencia*, E.S. Ingenieros Industriales de Sevilla, 1984

[2] Franquelo, L.G, A. Gómez, J.M. González Romano, J.L. Martínez, P.A. de Toro, Power System Simulation Module for a Dispatcher Training Simulator Project, *Int. J. of Energy Systems*, vol.11, no.3, 1991, pp.126-129.

[3] González Romano, J.M, J.A. Ternero and E.F. Camacho, Natural Language Interface for Process Control Centers. *Preprints 3rd IFAC Intnl. Workshop on Artificial Intelligence in Real Time Control*, Napa (California), 1991.

[4] González Romano, J.M, E.F.Camacho, Goal-Oriented Man Machine Interface in Control. Application to a Pilot Plant. *Prep. 12th IFAC World Congress*, Sydney, 1993, pp.455-458.

[5] González Romano,J.M. *Aplicación del lenguaje natural a la adquisición de conocimientos y operación de sistemas complejos*, Doctoral dissertation, Univ. Sevilla, 1997.

[6] González Romano, J.M, J. Gómez Ortega and E.F.Camacho, Application of a N.L.I. to the Teleoperation of a Mobile Robot. *Proc. IFAC Symposium on Intelligent Components for Vehicles*, Seville (Spain), 1998, pp. 413-418.

[7] Pai, M.A, *Computer Techniques in Power System Analysis*. Tata McGraw-Hill Publishing Co. Ltd, New Delhi, 1979.

[8] Raúll Martín, J, *Diseño de subestaciones eléctricas*, McGraw-Hill, 1987.

[9] Selfridge, M. and W. Vannoy, A Natural Language Interface to a Robot Assembly System, *IEEE Journal of Robotics and Automation*, vol. RA-2, no.3, 1986, pp.167-171.

[10] Silva, M, *Las Redes de Petri en la Automática y la Informática*, AC, 1985.

[11] Torrance, M.C, *Natural Communication with Robots* (doctoral dissertation), MIT, 1994.

[12] Weedy, B.M, *Sistemas eléctricos de gran potencia*, Reverté, 1978.