Production Systems and Information Engineering
Volume 3 (2006), pp. 109-119

HYPOTHESIS-BASED SEARCH IN PARTLY-OBSERVABLE SYSTEMS

TAMÁS BÁKAI
University of Miskolc, Hungary
Department of Information Engineering
iitrifle@gold.uni-miskolc.hu

[Received November 2005 and accepted May 2006]

Abstract. Nowadays the growing demands are the dominant concept in most parts of life. To satisfy these demands, planning of more complex and flexible problem-solving systems is required. In the last decades many new technology and technique were developed to handle the growing demands [6][7][8][9]. Apparently the object-oriented modelling methodology was the most efficient between them. However, nowadays the growing demands of the market gradually outgrow the abilities of the pure object-oriented concepts. One of the main reasons of this is that the decomposition techniques can not handle efficiently the numerous sub-systems with varying objective functions and constraints. The common problems appear in the untreatable complexity and the missed deadlines. The use of artificial intelligence means new concepts in the field the development processes. The agent based programming gives the possibility to describe the functionality of the required system not only by using actions-reactions but by defining the goals and constraints in the system. The machine-learning helps to determine the connections, relations and logical behaviour in the dynamism of the modelled system and helps to reveal the effects of the non-modelled systems into the modelled system. This paper shows a method for revealing and handling the effects of a non-modelled system according to the observed behaviour of the modelled system.

Keywords: learning systems, partly-observable systems, identification, cause-effect relations

1. Modelling the Observable-Systems

The basic concept of the machine learning is based on the observations of a functioning system and the knowledge gathering from the observed data. The observed systems according to their perceptibility can be categorized as observable and partly-observable systems [1]. In the learning system developed at the University of Miskolc, the observed system can be modelled using the object-oriented concept. The objects model the main and separate elements of the

observed system. The attributes describe the properties of the objects and the values represent the concrete cases of the attributes. The activities model the connections between the objects of the system. In this system each object has to have one or more attributes and zero or more activities and each attribute has to have two or more values [2]. The structure of the connections between the objects, attributes, values and activities can be described by a tree model. Each element represents a node of the tree.

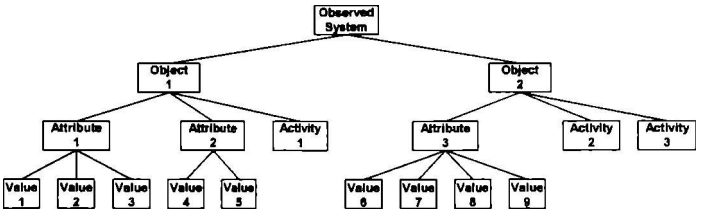


Figure 1. Structure of the connections between the objects, attributes, values and activities of the observed system

The activities and the values in this concept have no sub-elements therefore these are named leaf-nodes. Each leaf-node has one parent-node. The parent-node of a leaf-node represents the node the actual leaf-node belongs to directly. The parent-node of a value is the attribute the value belongs to and the parent-node of an activity is the object the activity belongs to. Each attribute has one marked value among its values at each time moment. This marked value represents the state of that attribute at that time moment. So the marked value is named active-value. The state of the observed system at time t can be described by the set of its active-values at time t.

The observed system has one marked activity among its activities at each time moment. This activity represents the event in the observed system of that time moment. The marked activity is named active-activity.

In the practice, the sampling frequency of the observing system is much higher than the frequency of the changes of states of the observed system. Therefore those complex events that generate changes of state can be separated into the sequence of single events. The event which occurs at time t_k represents the transient signal for the t_k change of state process and after reaching the t_{k+1} state of the observed system the active-event becomes inactive again.

Each leaf-node has one super-parent-node, too. The super-parent-node of a leaf-node represents the node whose one possible value the actual leaf-node represents. The super-parent-node of a value represents a possible value of the attribute the actual value belongs to; therefore the super-parent of each value is the attribute the actual value belongs to. The super-parent-node of an activity represents a possible

value of the events in the observed system therefore the super-parent-node of each activity is the observed system itself.

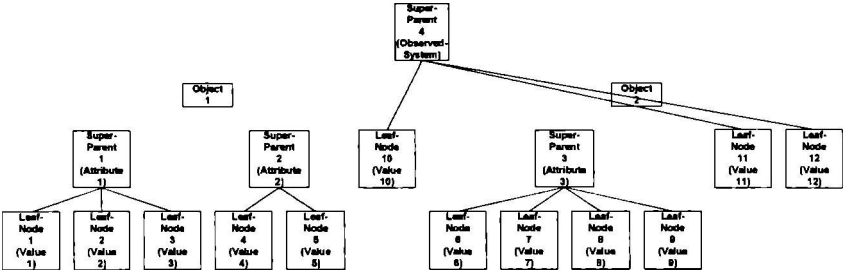


Figure 2. Structure of the super-parent and the leaf-node elements

The state of the observed system at time t can be described by the set of active-values at time t and the event which occurs at time t can be described by the active-activity at time t. The dynamic behaviour of the observed system can be modelled by its changes of state. The t_k change of state contains three parts:

1, the state of the observed system at time t (source-state of the change of state),
2, the event at time t,
3, the state of the observed system at time $t+1$ (destination-state of the change of state).

The changes of state are stored in a history database in chronological order. The t-th entry of the history contains the state of the observed system at time t_k and the event which occurred at time t_k.

2. Meaning of inconsistence in the changes of state processes

Denote each state of the observed system with uppercase letters and each event of the observed system with lowercase letters of the alphabet. Denote same letters the states with the same set of active-values and the same events and denote different letters the states with different set of active-values and the different events. According to this a possible state of the history can be as follows:

Table 1. Example of a possible pattern of the history database

time sequence	history
k	A, a
k+1	B, c
k+2	C, a
k+3	A, b
k+4	B

Four types of changes of state can be distinguished:

Table 2. The main different types of the changes of state

1. same source-state and same event result the same destination-state	*2.* same source-state and same event result different destination-state
3. different source-state or different event result the same destination-state	*4.* different source state or different event result different destination-state

In the changes of state of *types 1*, *3* and *4* the source-state and the event determine unambiguously the destination-state. The changes of state of these types are consistent. In the changes of state of type *2* the source-state and the event do not determine unambiguously the destination-state. The changes of state of these types are inconsistent.

If an observed system contains inconsistent changes of state then the cause-effect relations in the observed system can not be revealed unambiguously. This case indicates that the observed system is partly observable only.

3. Properties of the inconsistent changes of states

If in the changes of state processes of the observed system two changes of states become inconsistent then this proves that the observed system contains non-modelled elements. Modelling of these non-modelled elements can distinguish the same source states or the same events of the changes of states of type *2*. Therefore these changes of states could be converted into the changes of states of type *4*. According to the behaviour of the non-modelled elements the inconsistent changes of states can be divided into two main groups namely the Non-Observable-State-Space (NOSS) based and the sequential characteristic based inconsistent changes of states. For example if the observed system is a logical gate-circuit with an existing but non-modelled enabling input [3][10] then the observed changes of

states contain NOSS-based inconsistence. If the observed system is a container filling-emptying system which has one input and one output tap and three sensors which indicates the level of liquid at full, half and empty states, then the observed changes of states contain Sequential-Characteristic based inconsistence. To eliminate the different types of inconsistence two elimination methods was developed.

To eliminate the NOSS-based inconsistence, an automatically generated state-space with objects, attributes, values and activities is needed. The dynamic behaviour of this NOSS-based system can be determined indirectly according to the observable changes of state. The correct static structure and dynamic behaviour of the NOSS-based system can not be determined correctly at any time. The only thing the learning system can do is to determine the NOSS-based system in order to eliminate the inconsistence of the changes of states until the actual state of the system. Each newly observed change of state may modify the complete structure and behaviour of the automatically generated NOSS-based system.

To eliminate the sequential based inconsistence retrospection is required in the sequence of the observed changes of states. For example regarding to the container filling-emptying system the destination state of the changes of states can be determined unambiguously taking into consideration not only the actual state of the system but some sequence of states in the past from the actual state. This method is named *n-steps-deep-retrospection*. In the previous container filling-emptying example only one retrospection step is sufficient to eliminate the inconsistence in the changes of states.

$$\begin{aligned} (empty, half) &\Rightarrow full \\ (full, half) &\Rightarrow emty \end{aligned} \tag{1}$$

Detailed analysis is required to determine which method gives the best performance for eliminating the inconsistent changes of states of the actual observed system. This kind of research gives the developing possibilities of this system today. According to the latest results both methods can eliminate both types of inconsistence but they are different from each other according to their need for additional information.

4. The inconsistence elimination ability of The NOSS-based method

The NOSS-based elimination method assumes that the inconsistence in the changes of states is based on non-modelled objects, attributes, values or activities of the observed system. According to this concept the actual state of the system contains an observable and a non-observable part. In this point of view two states are equivalent with each other if not only their observable but also their non-

observable parts are equivalent with each other. Using this concept gives the possibility to distinguish the equivalent source-states of the changes of states of *type 2* with non-equivalent non-observable state space parts associated to their source states. Therefore the changes of states of *type 2* can be converted into the changes of states of *type 4*.

There is a dominant difference between the observable and non-observable parts of the states according to their knowableness. Reaching the t-th state of the observed system the observable part of this state defines correctly and invariably the observed state. On the other hand, generating the non-observable part for the t_k state gives nothing else than the changes of states will be consistent until the t_k state. However, many other non-observable state space combinations can give the same result. Choosing between them means an optimization task after each change of state step of the observed system. The constraint of this optimization is to define a non-observable state space for each state of the observed system in order to eliminate the inconsistence of the changes of states. The objective function of this optimization is to minimize the additional information need for the definition of the non-observable state space. To satisfy the constraint of this optimization the forbidden states of the non-observable state space has to be defined for each state of the observed system. The forbidden states represent the permanent information about the behaviour of the non-observable parts of the states. The forbidden states for each state are defined in the Non-Observable-State-Space-Constraint (NOSSC) of each state.

Denote with $!i$ in the NOSSC associated to the j-th state of the observed system that the non-observable part of the j-th state can not be equivalent with the non-observable part of the i-th state.

Similarly to the observable part of the observed system, the static structure of the non-observable part is modelled by objects, attributes, values and activities. These elements may be attached to an observable or a non-observable element of the modelled system. To determine the concrete need for each type of these elements and to reveal their necessary connections with each other requires deeper analysis in the static structure and dynamic behaviour of the observed system. At this time the non-observable part of each state is modelled using only one object (namely: *NOObject*), one attribute (namely: *NOAttribute*) and as many values (namely: *0*, *1*, *x*) as many non-equivalent states of the non-observable parts require.

5. The forbidden equivalences of the Non-Observable Parts

The elimination of the inconsistent changes of states of the observed system using the NOSS-based method means that in the changes of states with equivalent source-state and event but with non-equivalent destination-state the equivalence of

the source-states has to be eliminated. The source-states of the inconsistent changes of states are in forbidden equivalence. Because the observable part of each state holds permanent information and can not be modified, therefore a non-observable part is needed for eliminating the equivalence of these source-states. Figure 3 shows an example of the NOSS-based inconsistence elimination.

time	observable state space
...	...
i	A, a
i+1	B, ?
...	...
j	A, a
j+1	C

$$A \xrightarrow{a} B$$

$$A \xrightarrow{a} C$$

3.a. Example of inconsistent changes of states

time	observable state space	NOSSC	non-observable state space
...	...	...	...
i	A, a	?	0
i+1	B, ?	?	?
...	...	....	...
j	A, a	!i	1
j+1	C	....	...

$$A0 \xrightarrow{a} B$$

$$A1 \xrightarrow{a} C$$

3.b. Example of the elimination of the inconsistent changes of states

Figure 3. Example of NOSS-based inconsistence elimination

In this example the changes of states (*i*-th and the *j*-th) are inconsistent with each other and the properly generated non-observable part eliminates their inconsistence by taking their source states different. The *?* sign denotes the unimportant elements of the state according to this example.

Because the destination state of each changes of state is the source state of the next changes of state simultaneously, therefore each NOSS-based inconsistence elimination step may cause a side effect. A side effect appears each time if the previous states of the modified states are equivalent with each other. If the states *i*-

1 and *j-1* are equivalent with each other then before the non-observable part of states *i* and *j* become modified, the type of these previous changes of states were of *type 1*. After the non-observable part of states *i* and *j* will be modified the type of these previous changes of states become of *type 2*. These mean that these previous changes of states inconsistent with each other. To eliminate the occurred side effect, the equivalence of states *i-1* and *j-1* has to be eliminated too. This step may cause a new side effect too, if the previous states of the modified states are equivalent with each other. To eliminate all of the side effects occurred, the equivalence elimination method has to be performed until either the previous states of the modified states are non-equivalent with each other or one of the modified states is the first state of the observed system. Table 3. shows an example of side effects elimination steps.

Table 3. Example for eliminating the inconsistence and its side effects

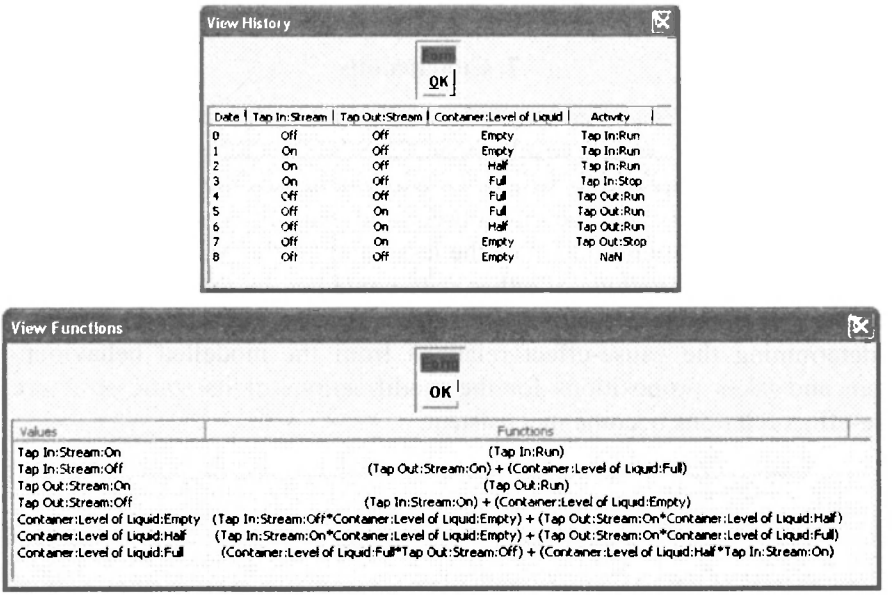
time	observable state space	NOSSC
0	?	?
1	?	?
...	...	...
i-k-1	B, a	?
i-k-1	C, ?	?
i-k	F, ?	?
i-2	D, c	?
i-1	B, a	?
i	A, a	?
i+1	B, ?	?
...	...	...
...	...	!0
...	...	...
j-k-1	B, a	!i-k-1
j-4	A, ?	!i-k-1
j-k	E, ?	!i-k
j-2	D, c	!i-2
j-1	B, a	!i-1
j	A, a	!i
j+1	C, ?	?

Because the state *j-k* (which belongs to the *k*-th inconsistence elimination step) is not equivalent with the state *I-k* no more side effects occur and the inconsistence elimination steps could be finished. If the state *j-k-1* (which belongs to the *k-1*

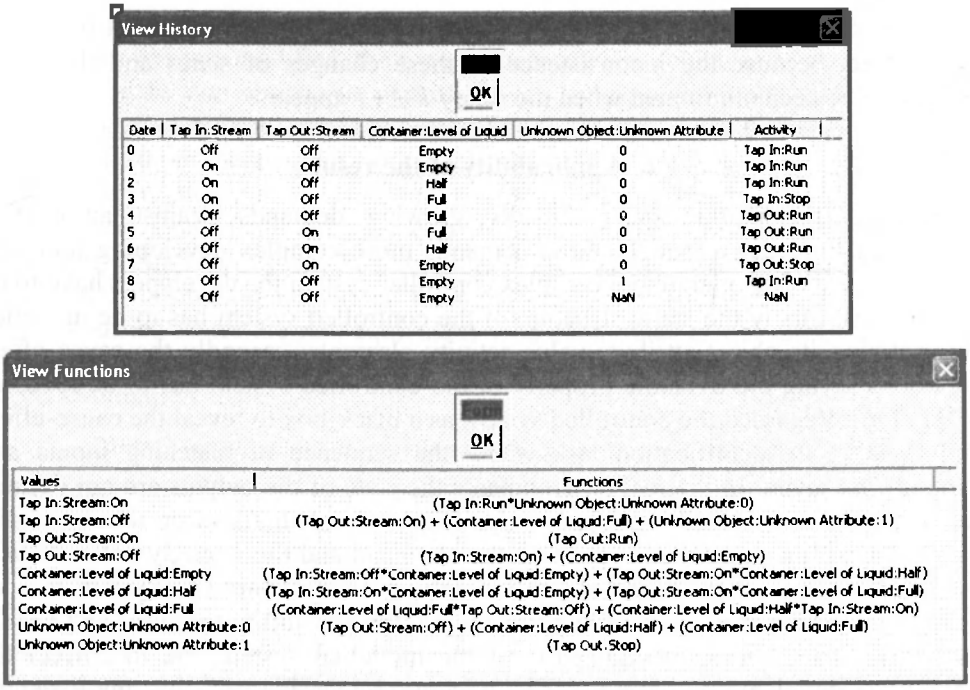
inconsistence elimination step) is equivalent with the state *I-k-1* then it results no side effect because the inconsistence of these changes of states and their side effects have been eliminated when the state *j-k-1+1* appears.

6. Applicability of the results

Nowadays the market has gradually growing demands against automation processes. In order to satisfy these demands more complex developing tools are required [7]. When planning a complex controller system the developers have to do two things. Firstly the static structure of the controlled system has to be modelled by defining its object-attribute-value-activity elements; secondly the cause effect relations giving the dynamic property of the controlled system has to be revealed [4] [5] [6]. Regarded the controlled system as a black-box to reveal the cause-effect relations is an identification task where the sequence of matching inputs and outputs are given. In the real environments the state of the outputs are not defined by the state of its matching inputs only. The cause of this lays on the sequential behaviour of the controlled system on the one hand and the wrongly defined static structure on the other. To divide the modelling of the controlled system into an observable and a non-observable part can give the possibility to reveal and handle the logical and the sequential parts of the modelled system and to correct the structural deficiencies. Figure 4 shows some screenshots of this implemented modelling tool.



4.a, Logical functions of the controlled system contains no non-observable elements



4.b, Logical functions of the controlled system contains non-observable elements

Figure 4. The revealed logical functions of the system identification method

7. Conclusions

The use of the NOSSC-based method for eliminating the inconsistent changes of states of the observed system results in an NOSSC for each state. The NOSSC of a state contains the states from which the equivalency of the actual state has to be eliminated. To determine the non-observable part of a state according to its NOSSC means the determination of the less value of the *NOAttribute* which is not the value of the *NOAttribute* of either state contained by the NOSSC of the actual state. This inconsistence elimination method gives a helpful tool for the modeller for determining the cause-effect relations from the modelled behaviour of the system and takes propositions for the modifications of its static structure if the cause-effect relations become inconsistent.

REFERENCES

[1] RUSSELL, S. J. NORVIG, P.: *Artificial Intelligence in Modern Approach.* Panem Könyvkiadó, Budapest, 2000. in Hungarian)

[2] KONDOROSI, K. LÁSZLÓ, Z., SZIRMAY-KALOS, L.: *Object Oriented Software Development.* ComputerBooks, Budapest, 1999. (in Hungarian)

[3] BÁNHIDI, L., OLÁH, M.: *Automation for Engineers.* Tankönyvkiadó, Budapest, 1992. (in Hungarian)

[4] VENKATESH, K., ZHOU, M.: *Object-Oriented Design of FMS Control Software Based on Object Modeling Technique Diagrams and Petri Nets*, Journal of Manufacturing Systems, pp. 118-136, 1998.

[5] CASTILLO, L., SMITH, J. S.: *Formal Modeling Methodologies for Control of Manufacturing Cells*: *Survey and Comparsion.* Journal of Manufacturing Systems, Vol 21/No 1, pp. 40-57, 2002.

[6] GAERTNER, N., THIRION, B.: *GRAFCET an Analysis Pattern for Event Driven Real-time Systems*, Plop Conference, p. 11, 1999.

[7] TÓTH, T.: *Design and Planning Principles, Models and Methods in Computer Integrated Manufacturing.* Miskolci Egyetemi Kiadó, 1998. (in Hungarian)

[8] BÁKAI, T.: *New Methods and Tools for Supporting the Logical Control Design of Machine Lines in the Manufacturing Industry*, 6th International Carpathian Control Conference, Miskolci Egyetemi Kiadó, pp. 279-286, 2005.

[9] BÁKAI, T.: *Identification of PLC Based Control Systems*, microCAD 2005 International Scientific Conference, pp. 13-18, Miskolci Egyetemi Kiadó, 2005.

[10] AJTONYI, I., GYURICZA, I.: *Programmable Control Systems, Networks and Systems.* Műszaki Könyvkiadó, Budapest, 2002. (in Hungarian)