Transition orthogonality in statechart diagrams and inconsistencies in binary control system

Grzegorz Łabiak

Abstract: The paper concerns the problem of some inconsistencies between controller and controlled object in control system. These inconsistencies are a results of incorrectly built predicates where state of the controller does not conforms state of the object. In the paper two solutions removing this flaw has been presented: with concurrent transition and with local variables.

Key words: statechart diagrams, transition conflict, transition orthogonality, binary control system

1. INTRODUCTION

1.1. DIGITAL CONTROLLER DESIGN

The first step in digital binary controller design is formal specification of controller behavior [1]. Behavior of the controller can be specified as a Finite State Machine – for simple behavior, as a Petri nets – which because of using of concurrency explicitly is much more efficient in complex behavior description or as a statechart diagrams which traditional FSM enhance with concurrency, hierarchy and broadcast mechanism [1][2][6]. It seems that statechart diagrams in comparison with Petri nets features greater expressive power, especially when exception modeling is concerned [3][4]. Next, formally modeled behavior can by implemented in programmable devices (e.g. FPGA or CPLD) [1]. Every behavior specification formalism features transitions which in case of deterministic behavior transform activity to the next only one sequential state (place). The determinism of such models is assured by transition orthogonality (transitions are conflict-free), but, in case of statechart it turns out that transition orthogonality is not enough condition.

The paper presents case study where despite the fact that transition are conflict-free the control system can develop wrong behavior. This problem can be solved by introduction concurrent transitions crossing state border, or, in case of statechart modular semantic where transitions must not cross state border, by introduction additional variables. Both solutions are described in the paper.

1.1. BINARY CONTROL SYSTEM

Binary control system is a system where a controller which operates on binary values on its input and output interacts with controlled object (Fig. 1) [5]. Some signals can come to the controller from outside operator and some other signals can be visible to the outside world. The controller prompts controlled object and the controlled object responds with its state signal to the controller and so on. The controller is a reactive system that continuously have to changes its actions in response to stimuli. In this system both controller and controlled object must have synchronized its internal states or in other words the system as a whole must be consistent. This means that the controller forces changes in the object and the controller updates its internal states in responses to changes in the controlled object.

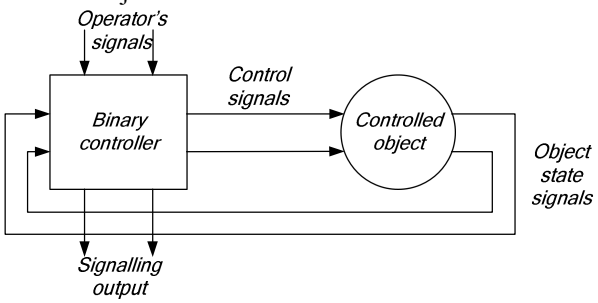

Fig. 1. Binary control system

1.2. TRANSITIONS IN CONFLICT

Two transitions are in conflict if there is some common state that would be exited if any one of them were to be fired [3]. In distinction from FSM and Petri nets statecharts conflicting transitions can be grouped into three categories:

- horizontal,
- vertical,
- mixed.

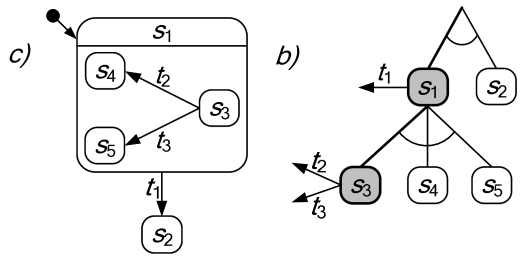

Fig. 2. Conflicts: a) diagram, b) hierarchy tree

The first case takes place when the transitions in conflict are on the same level of hierarchy tree. In Fig. 2 horizontally conflicting transitions are t_2 and t_3 and the common state is s_3. The second case holds when conflicting transitions are located on different levels of hierarchy tree. In Fig. 2 vertically conflicting transitions are transitions t_1 and t_2 and also t_1 and t_3, whereas the common state is again s_3. The case of transition t_1, t_2 and t_3

from Fig. 2 at the same time combines features horizontally and vertically conflicting transitions, so the conflict of those three is of mixed type at once.

For statechart-based controller to work correctly transitions being in potential conflicts must have predicates pair wise orthogonal (in the context of global state) [4], i.e. potentially conflicting transitions form compatibility classes, where compatibility means orthogonality relation, e.g. $\mathbf{T}_3=\{t_1,t_2,t_3\}$. The name of the set is $\mathbf{T}_3$ because potentially conflicting transitions come from the states which on hierarchy tree (Fig. 2b) belong to the path leading from the root to the leaf state s_3. For example for the diagram from Fig. 2 predicates imposed on transitions could be as follows: $t_1=a*!b*!c$, $t_2=b*!a*!c$, $t_3=c*!a*!b$. Then pair wise orthogonalities are as follows:

$t_1*t_2=a*!b*!c*b*!a*!c=0$,
$t_1*t_3=a*!b*!c*c*!a*!b=0$,
$t_2*t_3=b*!a*!c*c*!a*!b=0$

and hence the diagram is conflict-free.

Orthogonality of transition predicates is a necessary condition (but not sufficient) which must be met for a controller working correctly.

2. EXAMPLE

The chemical reactor is an example of industry technological process, in which reacting substances of two kinds are strictly measured out and next are mixed in water environment (Fig. 3) [1]. The process consists of three stages:

- water preparing and substrates weighting of given mass – state *FILLING*,
- ingredient stirring in main container for given period of time – state *PROCESS*,
- preliminary process preparing; this stage involves removing discards from scales and from main container – state *INITIATING*.

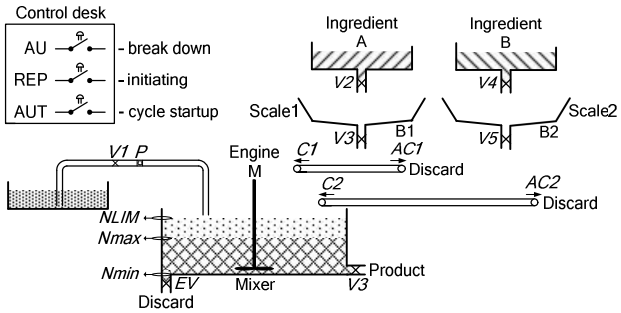

Fig. 3. Schematic diagram

The operator, who supervises course of process, has at his disposal a control desk which is capable of: break-down signalling (*AU* signal), init process requesting (*REP* signal), process starting (*AUT* signal). As it is can be seen in the Fig. 3, the operator is allowed to signal break-down in course of filling of containers and in course of chemical process execution. Incoming signals to the controller are signals from weight and level sensors (*B1*, *B2*, *NLIM*, *Nmax*, *Nmin*) deployed in chemical installation and signals from external clocks, which are assigned to measure given time intervals. Outgoing signals from controller are setting signals for pump valves, belt conveyors, mixer engine and for clocks (*V1*, *V2*, *V3*, *V4*, *V5*, *V6*, *EV*, *C1*, *AC1*, *C2*, *AC2*, *M.*, *TM1*, *TM2*). Fig. 4 presents reactor block diagram.

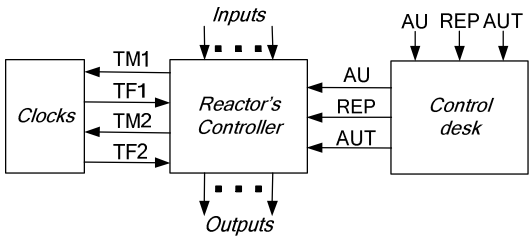

Fig. 4. Context of the controller

Functioning of a chemical plant is as follows (Fig. 5). System start from state START and next in case of lack of break-down signal moves into state INITIATING, where main container and belt conveyors are cleared out of previous cycle process remainders. Next, with signal AUT, preparing of process ingredient is started – state FILLING. In this state, break-down notification (signal AU) makes that state RESTART becomes active and after the failure is repaired active state of superstate FILLING become most recently active ones. Behaviour of this kind is achieved with the use of history attribute. After all the containers (main container and scale containers) are properly filled main chemical process is being started, where reaction time (state REACTION) is measured by external clocks. Start of an ensuing process is triggered after signal AUT is introduced from control desk, under that condition, that main container is emptied to the desired level (Nmin signal). Then system moves to the filling STATE.

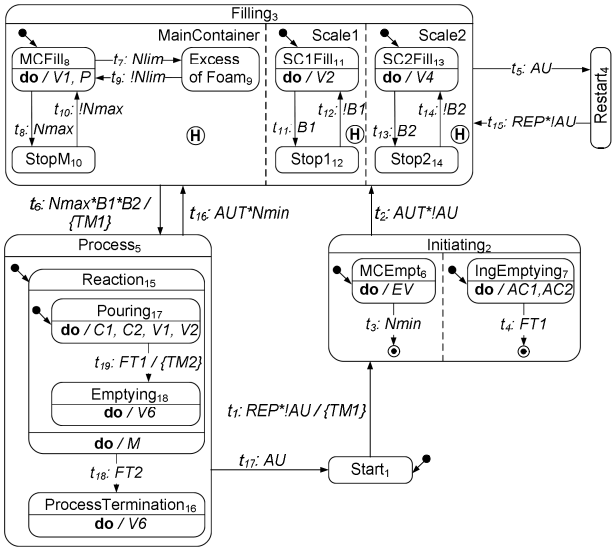

Fig. 5. Statechart of the controller

The diagram in Fig. 5 features nondeterminism, it means that transitions depicted there are potentially in conflict. The sets of transitions potentially in conflict are:

$\mathbf{T}_8=\{t_5,t_6,t_7,t_8\}$, $\mathbf{T}_9=\{t_5,t_6,t_9\}$,
$\mathbf{T}_{10}=\{t_5,t_6,t_{10}\}$,
$\mathbf{T}_{11}=\{t_5,t_6,t_{11}\}$, $\mathbf{T}_{12}=\{t_5,t_6,t_{12}\}$,
$\mathbf{T}_{13}=\{t_5,t_6,t_{13}\}$,
$\mathbf{T}_{14}=\{t_5,t_6,t_{14}\}$, $\mathbf{T}_{17}=\{t_{16},t_{17},t_{18},t_{19}\}$.

To solve conflicts orthogonal predicates can be used, but because of the diagram legibility, predicates imposed on transition consist of only these signals which are essential for proper understanding working of the reactor.

3. INCONSISTENCIES IN CONTROL SYSTEM

Binary controller and controlled object interact in control system. Current state of the controller must correspond to the state of the controlled object.

Let us take a closer look at compound state FILLING (Fig. 5) and let us simplify original diagram. Fig. 6 presents one of many possible sets of orthogonal transition predicates. Are the three processes (A, B, C which correspond to MainContainer, Scale1 and Scale2, respectively), that should be concurrent, really independent one of another? Can the transitions in these three concurrent regions fire freely and can states active independently one of another? Surprisingly, it turns out that they cannot.

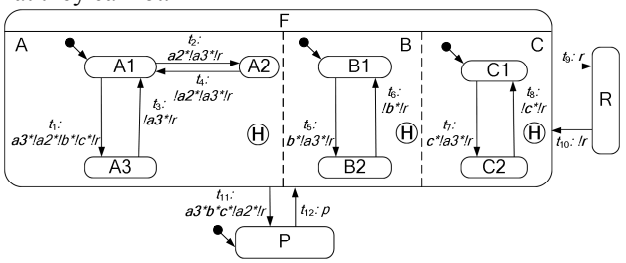

Fig. 6. Simplified diagram

For example, state *A3* and *B2* will never be active together. Simultaneous activity of these states could supposedly be reached in three case:

a) states *A1* and *B1* are simultaneously active and transitions t_1 and t_5 are firing simultaneously – this cannot happen, because of orthogonal predicates (events *b* and *a3*) imposed on t_1 and t_5,
b) states *A1* and *B2* are active and transitions t_1 fires – then predicate imposed on *c* (*a3**!*a2**!*b**!*c**!*r*) is met and it makes that predicate imposed on t_6 is also met (!*b**!*r*), so t_6 fires and activity from *B2* is removed; as a result active states are *A3* and *B1*,
c) states *A3* and *B1* are active and transition t_5 fires – then predicate imposed on t_5 (*b**!*a3**!*r*) is met and it makes that predicates imposed on t_3 is also met (!*a3**!*r*), so t_3 fires and activity from *B3* is removed; as a result active states are *A1* and *B2*.

In cases b) and c) it is as if activities in concurrent regions *A* and *B* pass each other. Similar considerations can be apply to regions *A* and *C*.

Changes in a controller are not a results of respective changes in controlled object. They appear as an unintentional effects and lead to inconsistencies in the control system. What these inconsistencies means to the chemical reactor control system? The case a) means only that in controller main container and scales will never be full simultaneously, that is of no great significance for this system but is absurdity. In the case b) in the controlled object both main container and scale 1 are full, but controller is in local states *StopM* (*A3*) and *SC1Fill* (*B1*) – what is contradictory. The case c) corresponds to situation when, in the object, scale 1 and main container are full again, but controller is in states *Stop1* (*B2*) and *MCFill* (*A1*) – what is contradictory again.

A question arises what makes that under conditions of orthogonal predicates a control system is inconsistent? The answer is that predicate transitions in concurrent regions must not be in implication relation, i.e. firing transition in one region could never entail firing transition in other region. In case of an investigated example (Fig. 6) this undesired relation takes place between transitions: t_1 and t_6, t_5 and t_3, t_1 and t_8, t_7 and t_3. This stems from the fact that predicate of one transition is an implicant of predicate of other transition, e.g.: $t_1 \rightarrow t_6$ means:

$$a3^*!a2^*!b^*!c^*!r \rightarrow !b^*!r$$

In this particular chemical reactor control system occurring inconsistencies do not have serious consequences for the controlled object, but in case of live- or safety-critical system this can be heavily dangerous.

4. ASSURANCE OF CONSISTENCY

The problem is how to avoid in control system inconsistencies brought about by orthogonal transitions. The answer is that for every pair of transition located in two different concurrent regions their predicates must not implicate one another. To say it in other way: for two predicates of transitions t_1 and t_2 following two formulas must be satisfiable: $!(t_1 \rightarrow t_2)$ and $!(t_2 \rightarrow t_1)$. To put it formally:

$$\exists_M M \models !(t_1 \rightarrow t_2) \text{ and } \exists_M M \models !(t_2 \rightarrow t_1),$$

where *M* is an interpretation satisfying formulas (model). This is sufficient condition (non-implication) related to transition predicates which must be met for a controller to work correctly. For the diagram from Fig. 6 examples of pair non-implicable transitions are: $\{t_1,t_5\}$, $\{t_1,t_6\}$, $\{t_1,t_7\}$.

Assurance of consistency comes down to fulfillment of two conditions: orthogonality (necessary) and non-implicability (sufficient). This can be achieved by appropriate selection of predicate formulas. However, the task is computationally complex and rather unfeasible for the human, especially in case of complex diagrams, therefore, it is necessary to use some syntactical graphic structures. Compound concurrent transition and special variables feature of great clarity and can be applied.

4.1. CONCURRENT TRANSITION

Concurrent transition may have many input states. It represents a synchronization of concurrent threads [6]. The transition is enabled when all the source states are active and also may have imposed predicates. Graphically concurrent transition is a heavy bar with many arrows coming from source states.

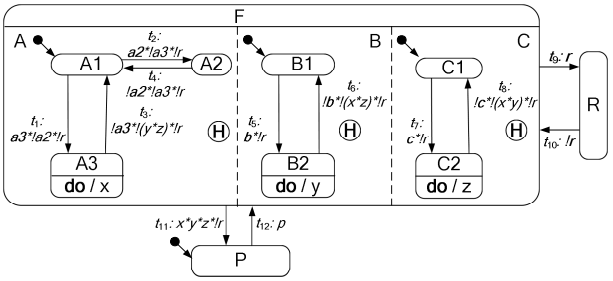

Fig. 7. Synchronization with concurrent transition

Fig. 7 presents improved diagram from Fig. 6 and Fig. 8 presents improved reactor controller. Implicitly is assumed that concurrent transition has higher priority over hierarchy lower level transitions (t_3, t_6, t_8). If it is necessary

priority can be changed by changing predicate, e.g. transition t_{11} predicate could be $a3*b*c*!r$.

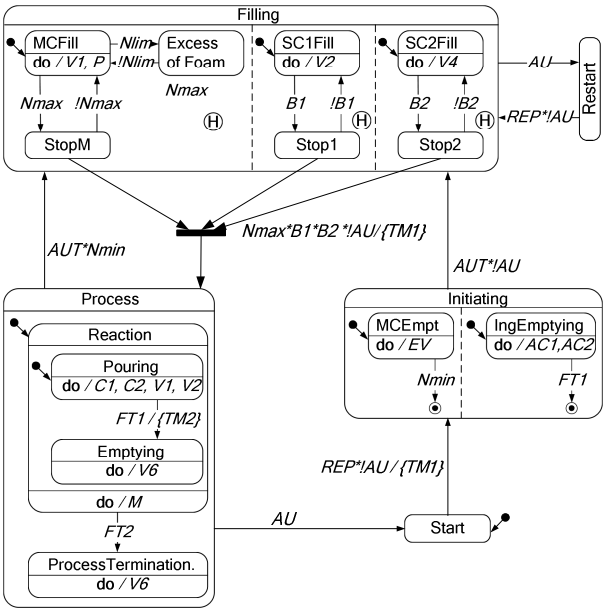

Fig. 8. Statechart diagram

4.2. SYNCHRONIZATION VARIABLES

Synchronization variable is normal variable declared locally, but used in special way. When activity has to leave state *F* (Fig. 9, transition t_{11}) during states *A3*, *B2* and *C2* are active, variables can be associated with the states (action *do*) and form predicate imposed on transition t_{11} ($x*y*z$). For the transitions t_3, t_6 and t_8 to resolve conflicts with t_{11} their respective predicates must take into account activities of the states *A3*, *B2* and *C2*. This is done by presence of the proper synchronization variables (e.g. for t_3 the condition is $!(y*z)$). It is worth noting that synchronization local variable, unlike input variable, can only be changed by activities of the diagram (e.g. states or transitions) and not by event coming from outside world.

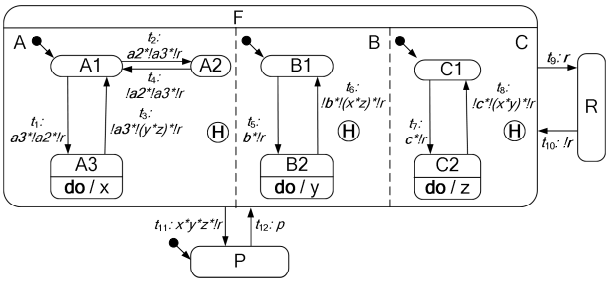

Fig. 9. Synchronization with variables

5. SUMMARY

The first necessary condition concerning transitions is orthogonality. Predicates imposed on transitions must be orthogonal. But this is not sufficient condition for correctly working controller. The only orthogonal transitions can make bring about inconsitencies in control system. The state of the controller does not conform the state of controlled object. Their transition predicates must meet second sufficient condition – non-implication. Predicates pair wise must not implicate each other. Both conditions can be met by appropriate construction of predicates what is very complex. Therefore statechart graphic structures: concurrent transition and special variable can be applied. It is worth to note that introduction synchronization with variable (Fig. 10) reduced the number of global sates from 162 to 41.

Figs. 8 and 10 present improved diagram of reactor controller. Predicates presented in the diagrams are incomplete, they include only signals which are essential for grasp of controller working by the designer. Complete predicates would have obscured the diagrams and their full version can be obtained by analogy to Figs. 7 and 9.

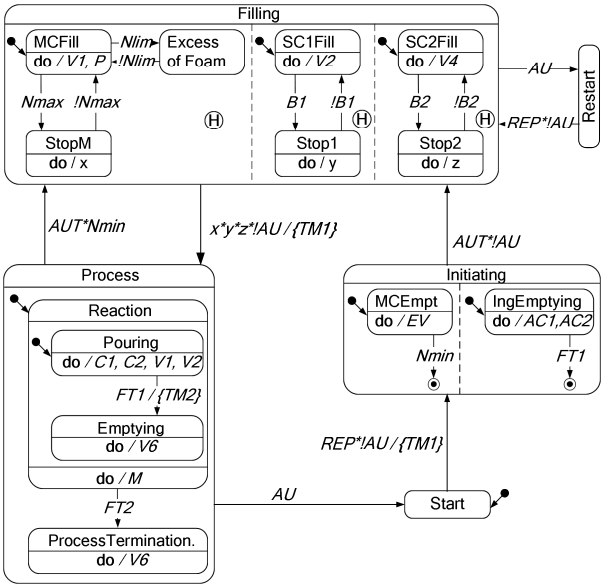

Fig. 10. Statechart diagram

LITERATURA

[1] *Design of Embedded Control Systems*, ed.ed. M. A. Adamski, Karatkevich A., and Węgrzyn M., Springer 2005

[2] Harel D., *Statecharts: A Visual Formalism for Complex Systems*, Science of Computer Programming 8, pp. 231–274

[3] Harel, D. and Naamad A.. *The STATEMATE Semantics of Statecharts*. ACM Trans. Soft. Eng. Method, 1996

[4] Łabiak G. *Transition conflicts detection in binary modular statecharts diagrams*, Proc. of IFAC workshop – PDS'04, Poland 2004, pp. 192-197

[5] Misiurewicz P., *Układy automatyki cyfrowej*, WSiP, Warszawa 1987.

[6] *UML, Unified Modeling Language Specification*. Version 1.4.2, ISO/IEC 19501, 2005

dr inż. Grzegorz Łabiak
University of Zielona Góra
Faculty of Electronic Engineering, Computer Science and Telecommunications
Computer Engineering. & Electronics Department
ul. prof. Z. Szafrana 2
65-246 Zielona Góra
tel.: 68 328 26 16