STUDIA UNIV. "BABEŞ–BOLYAI", INFORMATICA, Volume XLII, Number 2, 1997

Discrete Event Dynamic Systems Modeling

ALEXANDRU CICORTAŞ

Abstract. Specific tools can model dynamical systems. The formalisms developed allow to model a wide area of systems. The discrete event systems used frequently in modeling of manufacturing, communication networks, can also be formalized. Is developed a particular formalism starting from general dynamical systems, the Atomic Discrete EVent (ADEV), which is illustrated with an example, the system assembly using Virtual Assembly Cell.

1. Review of General Dynamical Systems

Discrete event modeling can be used in representation of dynamical systems which has piecewise constant input and output segments. This class is called by Zeigler [13] as *DEVS representable.* In particular, Differential Equation Specified Systems (DESS) are usually used to represent the system under control in hybrid systems and are controlled by high-level, symbolic, event-driven control schemes. Closure under coupling is a desirable property for subclasses of dynamical systems since it guarantees that coupling of class instance results in a system in the same class. The class of DEVS representable dynamical systems is closed under coupling. This justifies hierarchical, modular construction of both DEVS models and the (continuous or discrete) counterpart systems they represent. The combination of DEV and DESS allow to model and simulate the discrete and continuous systems. It provide a means of specifying systems with interacting continuous and discrete trajectories. The formalism is fully expressive of hybrid systems in that it is shown to be closed under coupling.

The followings are from [13]and [14].

Definition 1.1. *A general dynamical system is*

$$DS = (T, X, Y, \Omega, Q, \Delta, \Lambda).$$

Received by the editors: March 23, 1998.

1991 *Mathematics Subject Classification.* 68U20, 65C99.

1991 *CR Categories and Descriptors.* I.6.8 **[Simulation and Modeling]**: Types of Simulation – *Discrete event*

Additional Key Words and Phrases. Discrete event systems, Modeling and simulation, Dynamic systems modeling.

Where

- T is the time base;
- X is the set of input values;
- Y is the set of output values;
- Ω is the set of admissible input segments $\omega :< t_1, t_2 >\rightarrow X$ over T and X and Ω is closed under concatenation as well as under left segmentation;
- Q is the set of states;
- $\Delta : Q \times \Omega \rightarrow Q$ is the global state transition function;
- $\Lambda : Q \times X \rightarrow Y$ is the output function.

The global state transition function of general dynamical system DS has the following properties:

- consistency: $\Delta(w, \omega_{<t,t>}) = 0$;
- semigroup property: $\forall \omega :< t_1, t_2 >\rightarrow X \in \Omega, t \in< t_1, t_2 >$: $\Delta(q, \omega_{<t_1,t_2>} = \Delta(\Delta(q, \omega_{<t_1,t>}), \omega_{<t,t_2>})$;
- causality: $\forall \omega, \overline{\omega} \in \Omega$ if $\forall t \in< t_1, t_2 >: \omega(t) = \overline{\omega}(t)$ then
  $$\Delta(q, \omega_{,t_1,t_2>}) = \Delta(q, \overline{\omega}_{<} t_1, t_2 >).$$

Based on the causality, the semigroup property and closure of admissible segments under left segmentation the state trajectory resulting from every initial state $q \in Q$ and input segment $\omega :< t_1, t_2 >\in X \in \Omega$ can be defined as follows:

$$\mathrm{STRAJ}_{q,\omega} :< t_1, t_2 >\in Q$$

where $\forall t \in< t_1, t_2 >$ $\mathrm{STRAJ}_{q,\omega}(t) = \Delta(q, \omega_{<t_1,t>})$. The output trajectory $\mathrm{OTRAJ}_{q,\omega} :< t_1, t_2 >\rightarrow Y$ is defined as follows. For every initial state $q \in Q$, input segment $\omega :< t_1, t_2 >\rightarrow X \in \Omega$ and $t \in< t_1, t_2 >$

$$\mathrm{OTRAJ}_{q,\omega}(t) = \Lambda(\mathrm{STRAJ}_{q,\omega}(t), \omega(t)).$$

The input/output behavior R_{DS} of the dynamical system is given by

$$R_{DS} = \{(\omega, \mathrm{OTRAJ}_{q,\omega}) : \omega \in \Omega, q \in Q\}.$$

2. Discrete Event Systems

In the following is proposed a particular formalism developed from [14]. An Atom of a Discrete EVent (ADEV) system can be specified with following formalism.

Definition 2.1.

$$ADEV = (X_a, Y_a, S_a, \delta_a, \lambda_a).$$

Where

- X_a is the set of inputs;
- Y_a is the set of outputs;
- S_a is the set of states;
- $\delta_a : S_a \rightarrow S_a$ is the state transition, which refers only to internal states;

- $\lambda_a : S_a \to Y_a$ is the output function.

The whole system is composed from atomic ADEVs that co-operate one another in order to improve the real system behavior and specified time evolution given in the δ_a by some algorithm and the *event timestamp* is the instant when an event appears or *state timestamp* is the instant when the system enters in a specified state. A set of ADEV specifies a Dynamical System DS in the following way:

- the time base T is the set of real numbers R;
- $X = \cup X_a \cup \{\emptyset\}$, i.e., the input set of the dynamical system is the union of the input sets of the ADEVSs together with the $\emptyset$ specifying the non-event;
- $Y = \cup Y_a \cup \{\emptyset\}$, the output set of the dynamical system is the union of the output sets of the ADEVs together with the $\emptyset$;
- $Q = \{(s, ts(s)) : s \in S, ts(s) - \text{timestamp of s}\}$ the set of states of dynamical system consists of the states of the ADEVs paired with the timestamp of the state ts_s, which is a real number and it can be assimilated with the $ta(s)$;
- the admissible input segments is the set of all ADEV-segments over X and T that are characterized by the fact that for any $\omega :< t_1, t_2 > \to X \in \Omega$ there is only a finite number of event times $\{\tau_1, \cdots, \tau_n\}, \tau_i \in < t_1, t_2 >$, with $\tau_i \neq \emptyset$;
- for any ADEV input segment $\omega :< t_1, t_2 > \to X \in \Omega$ and state $q = (s, ts(s))$ at time t_1 the global state transition function $\Delta(q, \omega_{<} t_1, t_2 >)$ can be defined but this definition is not given now;
- The output function Λ of the dynamical system is given by $\Lambda(s, ts(s), x) = \lambda(s)$.

For the input segment ω must define its position over time related to the events that appear before it, into it and after it.

The property of piecewise constant of trajectories can be transformed into ADEV-segments. If we have a dynamical system whose all input trajectories $\omega \in \Omega$ and associated output trajectories $\text{OTRAJ}_{q,\omega}$ are piecewise constant, such system can be represented in the ADEV formalism. We give without proof the following theorem [14].

Theorem 2.2. *The ADEV constructed for a dynamical system whose all trajectories are piecewise constant and the dynamical system are behaviorally equivalent, i.e., they have the same input/output behavior when started in the same state.*

The models constructed from components (and especially similar components, [4]), can be formalized in the modified ADEV formalism as follows.

$$CM =< X, Y, M, EIC, EOC, IC, SEL >$$

where

- X : input events set;
- X : input events set;

- M : ADEV component set;
- EIC $\subset$ CM.OUT $\times$ M.IN: external input coupling relation;
- EOC $\subset$ M.OUT $\times$ CM.IN: external output coupling relation;
- IC $\subset$ M.IN $\times$ M.OUT: internal coupling relation:
- SEL : $2^M - 1 \to M$: tie-breaking selector.

The CM.IN, CM.OUT, M.IN and M.OUT refer to the input queues and output queues of the new model constructed and of the component models respectively. The EIC,EOC and IC specify the connection between the set of models M and input queues and output queues X, Y. The SEL function acts as a tie-breaking selector. The closure under coupling of ADEV-representable systems can be stated as:

Theorem 2.3. *A modular coupled system whose components are ADEV-representa and which does not contains loops in the graph sense, is an ADEV-representable dynamical system.*

This theorem was given and its proof was done by Praehofer and Ziegler, for DEVS-representable dynamical systems in which differ the definitions of EIC, EOC and ID from the above definitions.

3. Example

Let a system, which is composed from, cells that assembles the products and their compound parts from component parts. Such cell can be defined as an ADEV and the system as previous CM. In [1] and [2] the cell was defined in a model for a distributed computer architecture and was named Virtual Assembly Cell (VAC). The ADEV can be stated for the current VAC VAC_{crt} as follows.

- $X_{crt} = \{re_i, i \in \{i_1, \cdots, i_n\}, ac_j, j \in \{j_1, \cdots, j_m\}\}$ is the set of all:
  - re_i requirements made by other VACs to the VAC_{crt} and
  - ac_j achievements from other VACs to the VAC_{crt};
- $Y_{crt} = \{nd_j, j \in \{j_1, \cdots, j_m\}, as_i, i \in \{i_1, \cdots, i_n\}\}$ is the set of all:
  - nd_j requirements made for the VAC_{crt} to other VACs for the component parts that are necessary for assembling its own compound part (as needs);
  - as_i achievements made by the VAC_{crt} to other VACs as response to requirements made by these VACs (as assembly result);
- S_{crt} is the set of states for the VAC_{crt}, where the states are:
  - s_{in} initial state;
  - s_w waits for achievements of component parts that are necessary for a requirement;
  - s_a assembles its own compound part, when all component parts were received from the appropriate VACs, this state is time consuming and the assembly time interval will increase the time;

   - s_f final state, when all requirements were assembled;
- $\delta_{crt}(re_i, ac_j, ts_j, tas_i, v)$ is the state transition which applies to all events that appear, whose details will be done in the following;
- $\lambda_{crt}(re_i, ts_i, nd_k)$ is the output function.

The state transition $\delta_{crt}(re_i, ac_j, ts_j, tas_i, v)$ has the following arguments:

- $re_i, i \in \{i_1, \cdots, i_n\}$ is a requirement for its own compound part of the VAC_{crt} made by some VAC; all requirements are indexed by the order of its arrival to the VAC_{crt};
- $ac_j, j \in \{j_1, \cdots, j_m\}$ is an achievement of a needed component part sent by the appropriate VAC;
- $ts_j, j \in \{j_1, \cdots, j_m\}$ is the timestamp of the achievement ac_j;
- $tas_i, i \in \{i_1, \cdots, i_n\}$ is the timestamp of finishing the assembly process for the re_i;
- v is zero while $\forall i$ the re_i has not received yet all the needed component parts and is one while $\exists i$ such as for the re_i were received all the achievements.

Denote that the re_i, nd_k, ac_j contains a list of needed attributes that are not detailed here. The state transition acts as follows:

- $$\delta_{crt}(re_i,,,,0) : \begin{cases} s_{in} \rightarrow s_w & \text{or} \\ s_w \rightarrow s_w \end{cases}$$

  a requirement re_i arrived from a VAC, then the appropriate list is updated with it and is estimated the necessary amount for every component part:
   - updates the appropriate list of requirements with the requirement re_i in the VAC_{crt} appropriate list;
   - computes the amount for every component part $nd_k, \in \{j_1, \cdots, j_m\}$
   - updates the appropriate list with these needs;
   - pass in the state s_w;
- $$\delta_{crt}(re_i, ac_j, ts_j,,0) : \begin{cases} s_w \rightarrow s_w & \text{or} \\ s_w \rightarrow s_a \end{cases}$$

  an achievement is just arrived and is updated the appropriate list:
   - an achievement arrived and updates the appropriate list with it;
   - if $\exists j, \in \{j_1, \cdots, j_m\}$ for the requirement re_i such as for it was not received the achievement, then remains in the same state s_w, else passes to the state s_a for the requirement re_i;
- $$\delta_{crt}(re_i,,,,1) : \begin{cases} s_a \rightarrow s_w & \text{or} \\ s_a \rightarrow s_f \end{cases}$$

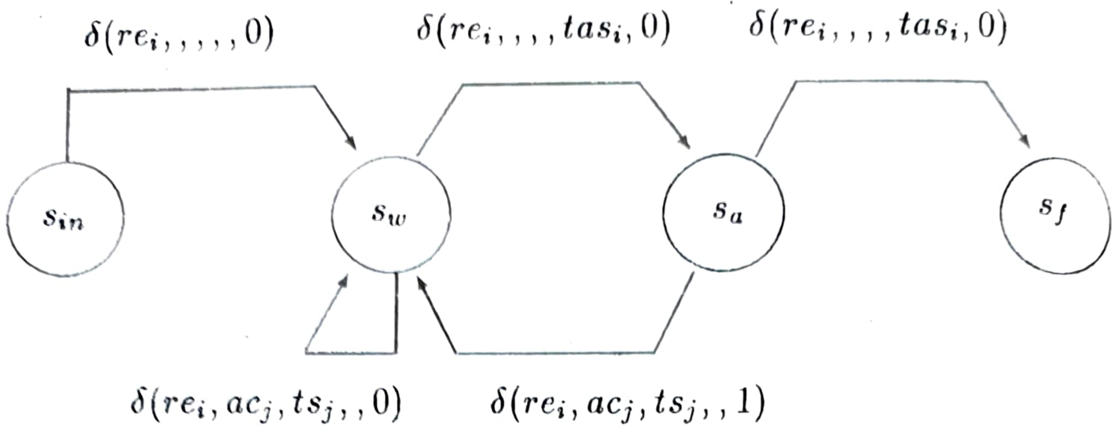
$\delta(re_i,,,,,0)$
$\delta(re_i,,,,tas_i,0)$
$\delta(re_i,,,,tas_i,0)$
s_{in}
s_w
s_a
s_f
$\delta(re_i,ac_j,ts_j,,0)$
$\delta(re_i,ac_j,ts_j,,1)$

FIGURE 1. The state transition function

for the requirement re_i all the achievements arrived and the VAC_{crt} is in the state s_a, then:

- is computed tas_i = LVT = $\max_j(js_j, LVT) + ass(re_i)$ where LVT is the Local Virtual Time, $ass(re_i)$ is the time interval necessary to assemble the compound for the requirement re_i;
- is updated the appropriate list with the assembly result for the re_i;
- pass in the state s_w or in the state s_f if there all requirements were assembled;

In the Figure 1 is illustrated the state transition function for the proposed model.

Remark 3.1. *The time evolution in the proposed model is supplied by δ function. At simulation beginning, the $LVT = 0$ for all VACs. The time in a VAC is the LVT, which is computed with the formula*
$$tas_i = LVT = max_j(js_j, LVT) + ass(re_i).$$

The output function $\lambda(re_i, tas_i, nd_k)$ has the following attributes that are positional. Some of them can be omitted.

- re_i is the achievement of the VAC_{crt} and has like destination the VAC which made the requirement;
- tas_i is the timestamp of the end of achievement for the re_i requirement;
- nd_k is the requirement of the VAC_{crt} made for the component part k which has like destination the VAC that assembled the required part.

The λ function acts as follows.

- $\lambda(re_i, tas_i,)$ sends the achievement of the requirement re_i to the appropriate VAC that made the request;

- $\lambda(,,nd_k)$ sends the requirement made by the VAC_{crt} to the appropriate VAC.

The coupling is another main feature of model. Strictly looking, the coupling between the VACs is given by the structure of products and compound parts. This coupling not allow the circuits in the graph sense.

4. Conclusions

The ADEV formalism is a powerful tool in modeling of systems that have component parts, which have a similar behavior [4].

In such formalism the time advancè function is supplied by the transition state function due to system particularity. The coupling is a main feature of the system which can be exploited. The VAC model is very useful in manufacturing modeling, taking into account the degradable VACs [3], it is a very powerful tool.

References

[1] Cicortaş, A, Distributed Modeling of Discrete Events, Ph.D. Thesis, West University of Timişoara, Mathematics Faculty, May 1997.

[2] Cicortaş, A., Conservative Simulation for Discrete Event Systems, INFORMATICA International,Tome 9, Issue 3, Vilnius, 1998, pp. 297-314.

[3] Cicortaş, A., Modeling with Fault Tolerant and Degradable Virtual Assembly Cells, International Symposium Conti'98, Technical University of Timisoara, 29-30 Oct. 1998.

[4] Cicortaş, A., *Modeling of Systems with Similar Components* accepted for publication in Analele Universităţii din Timişoara, Seria Matematică-Informatică, 1997/1998.

[5] Fishwick, P.A. and Ziegler, B.P., A Multimodel Methodology for Qualitative Model Engineering, ACM Transactions on Modeling and Computer Simulation, Vol. 2, No. 1, 1992, pp. 52-81.

[6] Fishwick, P.A., A Simulation Environment for Multimodeling, Discrete Event Dynamic Systems, Theory and Applications, Vol. 3, 1993, pp. 151-171.

[7] Fujimoto, R.M., Parallel Discrete Event Simulation, Communications of the ACM, Vol. 33, No. 10, Oct. 1990, pp. 31-53.

[8] German, R.,van Moorsel, A.P.A.,Qreshi, M.A., Sanders, W.H., Expected Impulse Rewards in Markov Regenerative
Stochastic Petri Nets, TR, Computer Science Department, University of Illinois at Urbana Campaign, 1996.

[9] Meyer, J.F., Sanders, W. H., Specification and Construction of Performability Models, Proceedings of the Second International Workshop on Performability Modeling of Computer and Communication Systems, Mont Saint-Michel, France, June 28-30, 1993.

[10] Sanders, W.H., Meyer, J.F., A Unified Approach for Specifying Measures of Performance, Dependability and Performability, Dependable Computing for Critical Applications, Vol. 4: of Dependable Computing and Fault-Tolerant Systems (ed., A. Avizienis and J. Laprie), Springer-Verlag, 1991, pp. 215-237.

[11] Peterson, J.L., Petri Net Theory and the Modeling of the Systems, Prentice-Hall, N.Y., 1981.

[12] Lin, Yi-Bing, Fishwick, P.A., Asynchronous Parallel Discrete Event Simulation, IEEE Transactions on Systems, Man and Cybernetics, Vol. XX, No. Y, Part A, 1996.

[13] Ziegler, B.P., *Object-Oriented Simulation with Hierarchical, Modular Models*, Academic Press, San Diego, Ca., 1990.

[14] Ziegler, B.P., Song, H.S., Kim, T.G., Prachofer, H., *DEVS Framework for Modeling, Simulation, Analysis, and Design of Hybrid Systems*, TR, Electrical and Computer Engineering, The University of Arizona Tucson,1998.

MATHEMATICS FACULTY, WEST UNIVERSITY OF TIMIŞOARA