A Discrete Event Systems Modeling Formalism
Based on Event Occurrence Rules
and Precedences

Vigyan Chandra and Ratnesh Kumar, Senior Member, IEEE

Abstract—The analysis (verification, diagnosis, control, etc.) of discrete event systems requires a correct model of the system and of its specifications. In this paper, we present a new modeling formalism for generating valid models of complex systems. The class of systems this applies to is one which consists of signals that take binary values. The technique presented here makes the task of modeling considerably less cumbersome and less error-prone and is user-friendly. Another advantage of using this modeling formalism is that the size of the system model is polynomial in the number of signals, whereas the number of states in the automata models is exponential in the number of signals. We present automated techniques for deriving an automaton-based model from the model in the proposed formalism. We illustrate the modeling formalism using examples drawn from manufacturing systems and process control.

Index Terms—Automata model, discrete event systems, rules-based model.

I. INTRODUCTION

MOST man-made systems are discrete event systems (DESs) owing to the manner in which they evolve: in response to events that are spontaneous, instantaneous, asynchronous (thus discrete in nature). Ramadge and Wonham [9] introduced the theory of supervisory control of DESs [8], [3] where they employed an automaton-based model of the system, called a plant, and studied how another automaton, called a supervisor, can be employed to restrict its behavior. The control specifications which express the constraints that one wishes to impose on the system’s behavior are modeled as automata or equivalently as formal languages. A supervisor exercises control over the system by dynamically disallowing a minimal set of controllable events so as to achieve the desired specifications.

There exist numerous types of models of DESs which include automaton-based models [9], Boolean models [1], and polynomial representations [6]. In [5], the authors provide a survey of modeling and analysis methods such as Temporal Logic, Calculus of Events, Petri Nets and Minimax Algebra. Logic-based systems and automatic theorem proving have been used for the development of a logic controller in [2].

Models are primarily classified as timed or untimed with the former having the ability to answer questions related to real-time and performance properties, and the latter for having the ability to answer questions that are only of a qualitative or logical nature. Both of these types can be further classified into stochastic and nonstochastic models.

Modeling of DESs continues to pose a challenge to system analysts. If one attempts to include all the details of the system in one single model, it is generally prone to errors as the size of the state space grows exponentially with the number of signals in the system. Thus deriving a single model for real world systems is a hard task. For this reason, modular construction is attempted where submodels are first built. It is naturally desired that all the relevant detail is present in the system model obtained from the composition of submodels, and additionally no extraneous behavior results. Since a formal analysis of a system relies primarily on automata for representing both the system and its specifications, developing easier techniques for obtaining accurate model for them is necessary.

A pliable modeling formalism is proposed here, which provides a methodology for constructing correct system models that makes the modeling process considerably simpler. We start by establishing the initial conditions of the system signals. Next weighted sensor event occurrence rules involving Boolean constraints over input (actuator) signals for each of the output (sensor) signals in the system are obtained. The weights are used to resolve issues of contention that can potentially arise amongst the weighted sensor event occurrence rules. Lastly, precedence relations amongst the sensor events in the system are modeled. We show how the rule and relation representations can be utilized for automatic derivation of an equivalent automaton model of the system. The model in our formalism is polynomial in the size of signals, in contrast to the automata models which are exponential in the size of signals. This compactness of representation, together with its intuitive nature, makes the model user-friendly, less error-prone, more flexible, easily scalable, and provides canonicity of representation. Since the time this paper was written, our further research has led to additional simplification of the modeling formalism, where the precedence relations are captured as part of the event occurrence rules. Moreover, this simplification has also led to a more direct and simplified procedure for automated derivation of an equivalent automaton model [4].

Manuscript received March 27, 2001. This paper was recommended for publication by Associate Editor S. Revianiis and Editor N. Viswanadhman upon evaluation of the reviewers’ comments. This work was supported in part by the National Science Foundation under Grant NSF-ECS-9709796 and Grant NSF-ECS-0099851, a DoD-EPSCoR grant through the Office of Naval Research under Grant N000140110621, and by a KYDEPSCoR grant.

The authors are with Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506 USA (e-mail: vigyan@engr.uky.edu; kumar@engr.uky.edu).

1042--296X/01$10.00 © 2001 IEEE
II. NOTATION AND PRELIMINARIES

An overview of the automata-based model of DESs follows. Let $\Sigma$ denote the finite set of events. A concatenation of events forms a string of events or a trace. A language is a collection of traces. Let $\Sigma^*$ be the set of all finite strings (traces) of events of $\Sigma$ including the empty string $\epsilon$. A language is thus a subset of $\Sigma^*$. Abstractly, a DES can be viewed as an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where $Q$ is the set of states, $\Sigma$ is the set of events, $\delta : Q \times \Sigma \rightarrow Q$ is the partial state transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked or final states. The behavior of the DES modeled by $G$ is described by its generated language: $L(G) := \{ s \in \Sigma^* | \exists (q_0, s) \}$. Where by induction the transition function has been extended from events to traces $\delta : Q \times \Sigma^* \rightarrow Q$. The generated language of $G$ is the set of all traces that it can execute starting from its initial state. The marked language of $G$ contains those generated traces which terminate in a final state and signify task completion: $L_m(G) := \{ s \in L(G) | \exists (q_0, s) \in Q_m \}$.

Synchronous composition [7] of automata is used to represent the concurrent behavior of two automata. Given two automata $S_1 := (X_1, \Sigma_1, \delta_1, x_{01}, X_{m1})$ and $S_2 := (X_2, \Sigma_2, \delta_2, x_{02}, X_{m2})$, the synchronous composition of $S_1$ and $S_2$, which is denoted by $S_1||S_2 := (X, \Sigma, \delta, x_0, X_m)$, is defined as: $X := X_1 \times X_2$, $\Sigma := \Sigma_1 \cup \Sigma_2$, $x_0 := (x_{01}, x_{02})$, $X_m := X_{m1} \times X_{m2}$, and for each $x = (x_1, x_2) \in X$ and $\sigma \in \Sigma$:

$$\delta(x, \sigma) := \begin{cases} \langle \delta_1(x_1, \sigma), \delta_2(x_2, \sigma) \rangle & \text{if } \delta_1(x_1, \sigma), \delta_2(x_2, \sigma) \text{ defined, } \sigma \in \Sigma_1 \cap \Sigma_2 \\ \langle \delta_1(x_1, \sigma), x_2 \rangle & \text{if } \delta_1(x_1, \sigma) \text{ defined, } \sigma \in \Sigma_1 - \Sigma_2 \\ \langle x_1, \delta_2(x_2, \sigma) \rangle & \text{if } \delta_2(x_2, \sigma) \text{ defined, } \sigma \in \Sigma_2 - \Sigma_1 \\ \text{undefined otherwise} \end{cases}$$

When $S_1$ and $S_2$ are composed, the common events occur synchronously, while the other events occur asynchronously. Note that, when $\Sigma_1 = \Sigma_2 = \Sigma$, then $L(S_1||S_2) = L(S_1) \cap L(S_2)$ and $L_m(S_1||S_2) = L_m(S_1) \cap L_m(S_2)$ since all events must occur synchronously.

For control purposes, the set of system events is partitioned into two disjoint sets: the set of controllable events, and the set of uncontrollable events. A controllable event is one which can be allowed to occur or prevented from occurring by an external agent, whereas no such control is possible for an uncontrollable event. The event set is also partitioned into the union of two other sets, namely, the observable ones and the unobservable ones.

In our context, we take an input–output view of the system. In fact, we think of events as either an input or an output event. Actuator events are examples of input events that are observable and controllable, whereas sensor events are examples of output events that are observable and uncontrollable. It is possible to have a system where certain input events are uncontrollable, and certain outputs are unobservable. In the present exposition we assume all inputs are actuator inputs and hence controllable, and all outputs are sensor outputs and hence observable.

III. EXAMPLE: MATERIAL HANDLING SYSTEM

A transporter, shown in Fig. 1, moves both forward and backward over guide rails, between the home and extended positions. It consists of a fixture that is connected to one end of a rack that is moved by a pinion powered from a gear box motor, $M_1$. With each circular rotation of the drive shaft, the transporter advances or retracts linearly.

An angle sensor, $A_1$, mounted on the same shaft as that of the pinion counts off the number of rotations of the axle through it in order to determine the position of the fixture. Position 0 in Fig. 1 is the home position of the transporter, while position 1 is a particular extended position, corresponding to a certain number of rotations of the drive shaft in the forward direction. When the transporter leaves the home position, it enters an intermediary position, which is a collection of all those positions whose values are unimportant from the positioning point of view. The slide continues to be in the intermediary position, until it reaches the extended position. An electric drive delivers voltage to the gear-box motor for movement in either direction. The forward/reverse direction commands to the drive appear on it independent of each other, and they cancel each other’s effect when they are both simultaneously on.

The events that can occur in the material handling system are: $T_{fon}$, $T_{fof}$, $T_{ron}$, $T_{rfo}$, $iup$, $udn$, $cup$, $cdn$. $T_{fon}/of$ refers to the gear-box motor being turned on/off in the forward direction, while $T_{ron}/of$ is for the reverse direction. Assume that the power associated with the forward direction command signal $T_{fon}$ has the value of $+1$, while that of the reverse command $T_{ron}$ is also $+1$. $iup$ (up) corresponds to the transporter arriving/leaving the intermediary position, from/to the home position; and $udn$ (down) corresponds to its arriving/leaving the extended position, from/to the intermediate one.

In order to construct an automaton model of the system, the sequencing of all the events of the system need to be considered. The material handling system has two actuators (two input signals) and two sensors (two output signals). Each signal is binary valued and hence possesses two transitions (low–high, high–low). Thus associated with each signal there are two events.

In fact, we think of events as either an input or an output event. Actuator events are examples of input events that are observable and controllable, whereas sensor events are examples of output events that are observable and uncontrollable. It is possible to have a system where certain input events are uncontrollable, and certain outputs are unobservable. In the present exposition we assume all inputs are actuator inputs and hence controllable, and all outputs are sensor outputs and hence observable.
none of the other states. Now, starting from this initial state, the automaton model is constructed by considering all the possible state transitions which can occur and connecting them to the resulting states, iteratively. Controllable transitions in the model are indicated by a short line drawn across them, indicating that the transition can be initiated or inhibited by an external controller.

In actual practice, the development of automata models can turn out to be unmanageable, as the size of the models for real systems frequently exceeds a million states. This also makes it easy for the designer to make errors. Instead of starting with a complex and potentially unmanageable model for the system, we derive it from the proposed modeling formalism. In particular, the automaton model of the material handling system shown in Fig. 2, can be derived from the model of our formalism given in Fig. 3.

**IV. PROPOSED MODELING FORMALISM**

The proposed modeling formalism applies to systems with the following criteria.

1) The actuator signals (system inputs) and sensor signals (system outputs) are binary valued.
2) The actuator signals can be toggled at any time.
3) All the system states are determined by the values of signals in the system.

The system starts out with certain initial values of all its signals which is captured by the *initial conditions* in our modeling formalism. The occurrence of sensor events is initiated by prior occurrence of certain actuator events. For example, in Fig. 1, it is only after the forward movement command $T_f$ is given that it can reach either the intermediate $i$ or final $e$ positions on the slide. Such dependencies are captured through *weighted sensor event occurrence rules* in our modeling formalism. There is one such rule per sensor event. Binary valued weights are associated with the degree to which actuator events influence sensor events. These weights are then used to resolve contention between the two sensor events of the same sensor signal as explained later in Section IV-B.

The only restriction on actuator events in the system is that on/off alternate, while for sensor events, in addition to this default restriction, further constraints due to the physical structure of the system need to be taken into account, which has a bearing on the order in which the sensor signals transition.

In the material handling system shown in Fig. 1, initially when the transporter is at the home position, the intermediate and extended position sensors register a binary value of zero. When the transporter moves forward toward the extended position, first the intermediate position sensor turns high, followed by the extended position sensor turning high. Here we see a strong coupling effect between the way sensors are arranged physically and the order in which their sensed values change. We use *precedence relations* over sensor events to capture this.

In summary, a model in our formalism contains three components:

1) initial conditions;
2) weighted sensor event occurrence rules;
3) precedence relations over sensor events.

Remark 1: If the second modeling assumption does not hold (i.e., if the actuator signals cannot be toggled at any time), then the modeling formalism can be extended by including occurrence rules for actuator events (besides those for sensor events).

For the sake of illustration, we present the model of the material handling system in Fig. 3.

**A. Initial Conditions**

The DESs we consider for modeling purposes have a fixed initial state corresponding to the initial values of actuators and sensors connected up in the system. This initial state is commonly the state when the parts in the system are least/lowest and all the actuators are turned off.

For example, the initial conditions for the material handling system in Fig. 1 is when the forward–reverse motors are turned off and the slide is in its home position. Thus, we have $T_f = T_r = \text{off}$ and $i = e = \text{down}$. This is shown in Fig. 3-(1).

**B. Weighted Sensor Event Occurrence Rules**

The most significant interaction from the perspective of modeling is that which occurs between conditions on actuator signals and the sensor events they directly influence.

The system model is constructed out of *rules* which govern how the sensor events in the system are influenced by the values of actuator signals. Boolean formulae over actuator signals that
induce sensor events in the system constitute the sensor event occurrence rules in our modeling formalism.

Each actuator signal has associated with it a binary valued weight with regards to the sensor events in the system that it influences. This indicates the degree to which the actuator signal can exercise influence over the sensor signal. In the material handling system, for example, the weight of the actuator signal \( T^f \) with regard to the sensor signal transitions \( \text{up} \) and \( \text{dn} \) is taken to be the binary value 1.

Each rule has an antecedent and a consequent. The right-hand side (consequent) of the rule corresponds to a particular sensor event, either an \( \text{up} \) or \( \text{dn} \) going transition, while the left-hand side (antecedent) consists of a Boolean condition over actuator signals. The weight of a rule is a function of the Boolean formula in the antecedent of that rule and the weights of the individual actuator signals within the antecedent. For a system with \( m \) sensor signals, there will be \( 2^m \) such rules corresponding to each of the \( 2^m \) sensor events in the system.

The rules take on the following form for a system with \( m \) sensor signals, and \( p \) actuator ones:

\[
\begin{align*}
\text{Rule}_{1}^{\text{up}} : f_{1}^{\text{up}}(a_{1}^{\text{up}}, \ldots, a_{p}^{\text{up}}) & \Rightarrow s_{1}^{\text{up}} \\
\text{Rule}_{1}^{\text{dn}} : f_{1}^{\text{dn}}(a_{1}^{\text{dn}}, \ldots, a_{p}^{\text{dn}}) & \Rightarrow s_{1}^{\text{dn}} \\
\vdots & \\
\text{Rule}_{m}^{\text{up}} : f_{m}^{\text{up}}(a_{1}^{\text{up}}, \ldots, a_{p}^{\text{up}}) & \Rightarrow s_{m}^{\text{up}} \\
\text{Rule}_{m}^{\text{dn}} : f_{m}^{\text{dn}}(a_{1}^{\text{dn}}, \ldots, a_{p}^{\text{dn}}) & \Rightarrow s_{m}^{\text{dn}} 
\end{align*}
\]

where \( f_{1}^{\text{up/dn}}, \ldots, f_{m}^{\text{up/dn}} \) are the Boolean formulae consisting of those combinations of the actuator events \( a_{1}, \ldots, a_{p} \), which can result in the right-hand side of each formula to become true, and \( w_{1}, \ldots, w_{p} \) denote the weights of \( a_{1}, \ldots, a_{p} \), respectively.

For example, the weighted sensor event occurrence rules for the material handling system are shown in Fig. 3-(2). In it, \( \text{Rule}_{1}^{\text{up}} : T^f \rightarrow \text{up} \) means that when the transporter drive is not connected to the slide though the rack and pinion, then eventually the transporter will reach the intermediate position (\( \text{up} \)). The weight of the actuator signal \( T^f \) is taken to be +1. (The weights are binary valued for a “pure” DES, but can take other nonnegative real values for general systems for which an abstract DES model is sought, as is illustrated by the process control example of Section IV-B.)

If there is no rule for a sensor event in the system, say for the \( \text{dn} \) event, then implicitly such an event never occurs. As an example, in the material handling system of Fig. 1, if the transporter drive is not connected to the slide though the rack and pinion, then under no condition can the intermediate or the extended position go high, regardless of the power applied to the transporter drive. This situation may also be represented as \( \text{false} \Rightarrow \text{up} \) and \( \text{false} \Rightarrow \text{dn} \).

We compute the weight of a rule at each combination of actuator signal values where its antecedent is true. The weights corresponding to the actuator signals are combined using either a minimum or a maximum operation depending on whether the combination of the actuator signals is an AND or an OR combination. It should be noted, however, that the operations for combining the weights would more generally be application-dependent. When the composition is based upon minimum and maximum operations, the weight of a rule is defined inductively as follows.

For the AND composition, it is given by \( w(a_{1}^{\text{up}}, a_{2}^{\text{up}}) = \min(w(a_{1}), w(a_{2})) \) and for the OR composition as \( w(a_{1}^{\text{up}} \lor a_{2}^{\text{up}}) = \max(w(a_{1}), w(a_{2})) \). Thus, for example, \( w((a_{1}^{\text{up}} \land a_{2}^{\text{up}}) \lor a_{3}^{\text{up}}) = \max((\min(w(a_{1}), w(a_{2})), w(a_{3}))) = \max((\min(w_1, w_2), w_3)) \).

C. Precedence Relations

Precedence relations capture the order in which sensor events can occur in the system. Since all of the system signals can take on only binary values, it is always the case that up- and down-transitions alternate, which is the default precedence relation. In addition, the layout of the system places restrictions on the order in which the sensor events can occur in relation to the other. This behavior can be captured by a partial order.

Consider the material handling system shown in Fig. 1. If there were no constraints on the order of occurrence for the sensor events, then, starting from an initial state, when the transporter is in the home position, that is, \( i \) and \( c \) are low (\( \text{ch} \) and \( \text{cn} \), respectively), the sensor events that can occur are either \( \text{up} \) or \( \text{dn} \). The arrangement of the transporter slide sensors, however, make it impossible for the extended position sensor, \( c \), to go high (\( \text{cp} \)) before the intermediate position sensor, \( i \), is already high (\( \text{ip} \)). In a similar way, while the slide is retracting from the extended position, it is not possible for \( \text{dn} \) to occur prior to \( \text{cn} \). The precedence relations for the material handling system are shown in Fig. 3(a) and (b).

Consider next Fig. 4-(1), in which the tank shown has two level sensors \( n \) and \( h \). The event transitions of these two sensors are \( \text{up} / \text{dn} \) and \( \text{hp} / \text{hd} \), respectively. As in the material handling system, if there were no constraints on the order of occurrence for the sensor events, then, starting from an initial state, when the tank is drained, that is \( n \) and \( h \) are low, the events that can occur are shown in the four-state automaton of Fig. 4-(2). This suggests that it is possible for the level to become high (\( \text{hp} \)) without first having to rise above the nominal level (\( \text{mp} \)). Clearly the physical structure of the tank disallows such a sequence of events from occurring, and this is shown by the region within the dotted border in Fig. 4-(2), while the only possible sensor event sequence is shown within the solid border. For this system, the two additional constraints other than the de-
fault ones which have to be considered are of the order in which np/lnp and hdn/ndn occur. Assuming that initially both the level sensors are low, the precedence relations for this system are: np precedes lnp and hdn precedes ndn.

Remark 2: Note that the number of rules and relations is polynomial in the number of signals of the system. Thus, the size of the model in our formalism is polynomial in the number of the signals. This is in contrast to automata models which have a state space that is exponential in the number of signals.

V. DERIVATION OF AN EQUIVALENT AUTOMATON MODEL

We introduce a few definitions, which are referred to in the algorithm used for automatically deriving an automaton model out of the model in the proposed formalism:

Definition 1: \( \Sigma_\alpha \)-diagram: This is an automaton which consists only of all possible interactions that can occur amongst the actuator events in the system, starting from a given initial state. If there are \( p \) actuator signals in a system, then the \( \Sigma_\alpha \)-diagram has \( 2^p \) states capturing all possible values of the \( p \) signals.

Definition 2: \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram: This automaton results when the \( \Sigma_\alpha \)-diagram is selectively augmented by self-loops on sensor events in the system using the weighted sensor event occurrence rules. Self-loops are those transitions in an automaton which originate and end at the same state. Their presence at a state indicates that the events associated with those transitions are permitted to occur in that state.

In certain states of the automaton, both transitions of a sensor signal may become possible. For example, in the material handling system shown in Fig. 1, this condition arises in the states where both the forward and reverse movement commands are enabled (\( \text{for and Iron} \)). In the states of the \( \Sigma_\alpha \)-diagram where this condition exists, both of the events, \( \text{np} \) and \( \text{idn} \), will be permitted to occur, resulting in contention as to which will actually occur. This contention is resolved using dominance of one rule over the other, as determined by their weights.

Definition 3: Refined \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram: Resolution of contention by the removal of those self-loops on sensor events of a signal, from the \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram, that cannot occur due to dominance of one over the other defines this diagram.

Definition 4: \( \Sigma_s \)-diagram: This is the automaton model obtained by imposing the constraints of precedence relations on the sequences of sensor events.

Algorithm 1: The overall system automaton model is derived using the following algorithm.

1) Construct the \( \Sigma_\alpha \)-diagram by drawing from the initial state, determined by the initial conditions of actuator events, all possible actuator signal states and transitions. This has \( 2^p \) states for a system having \( p \) actuator signals in it, since the actuator events in the system can always be switched on/off at will.

2) Augment the \( \Sigma_\alpha \)-diagram by adding self-loops of appropriate sensor events at appropriate states, as specified by the weighted sensor event occurrence rules, thereby creating a \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram. (This diagram tells when a sensor event may occur, but retains potential contentions, and lacks information on precedence of occurrence of sensor events.)

3) Refine the \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram by deleting those self-loops of sensor events which are deemed infeasible, after determining which sensor event dominates its counterpart event of the same signal.

4) Model the precedence relations amongst the sensor events in the system by the \( \Sigma_s \)-diagram. This is done by restricting the automaton modeling the default precedence relations.

5) Take a synchronous composition of the automaton of step 4 and 5 in order to obtain the final automaton model. The weights attached to the sensor event transitions may be removed, or they may be retained for future use.

Let us revisit the schematic of the material handling system shown in Fig. 1. The system description using our proposed modeling methodology has been given in Fig. 3. Next, we obtain an automaton model of the system using the algorithm presented.

An equivalent automaton model of the material handling system shown in Fig. 1, is derived using the algorithm described above as follows:

- **Step 1**: Draw the \( \Sigma_\alpha \)-diagram for the actuator events \( T_f \), \( T_r \) starting from the initial state. The resulting automaton model has four states and is shown in Fig. 5.

- **Step 2**: Augment the \( \Sigma_\alpha \)-diagram by self-loops of sensor events as specified by the weighted sensor event occurrence rules. From Rule\( \text{idn} \) it can be seen that forward movement occurs when the forward command \( T_f \) is on, with a power of +1, and from Rule\( \text{cdn} \) it follows that reversing occurs when the reverse command \( T_r \) is on, with a power of +1. So we add self-loops on \( \text{np} \), \( \text{esp} \) with weight +1, in states where \( T_f \) is on, and on \( \text{idn} \), \( \text{cdn} \) with weight +1, in states where \( T_r \) is on. The resulting \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram is shown in Fig. 6.

- **Step 3**: Refine the \( \Sigma_s \)-augmented \( \Sigma_\alpha \)-diagram by deleting at least one of the self-loops from those states where both sensor events of the same signal are present. Determine which sensor event of the same signal is feasible by comparing the weights of the two corresponding sensor event rules, in order to resolve contention. If the weights of both events are equal, delete both the self-loops, since neither would be possible.

Contention for the \( \text{np/idn} \), \( \text{esp/cdn} \) events can occur in certain states of the system owing to the presence of

\[ \text{Diagram} \]

Fig. 4. Automaton for representing precedence relations.
commands to move the motor in both directions. Rule\textsuperscript{op}\textsubscript{2} dominates Rule\textsuperscript{op}\textsubscript{1} in those states where the weight of the Rule\textsuperscript{op}\textsubscript{2} exceeds the weight of Rule\textsuperscript{op}\textsubscript{1}, and vice versa. In states where both are equal, neither of the sensor events \textit{iup}/\textit{idn} can occur. This also holds for the rules Rule\textsuperscript{op}\textsubscript{2} and Rule\textsuperscript{op}\textsubscript{1} of the events \textit{eup}/\textit{edn}. Contention between \textit{iup} and \textit{idn} arises only in those states where \textit{Tf} and \textit{Tg} are both on, and for those states, the automaton of Step 2 is refined by deleting the infeasible transitions, namely all self-loops on \textit{iup}, \textit{idn}, \textit{eup}, and \textit{edn}. The resulting automaton is shown in Fig. 7.

- **Step 4**: The structure of the material handling system needs to be taken under consideration while obtaining the precedence relations. Since the material handling system has two binary-valued sensors, \(i\) and \(e\), using the physical layout of the system, the precedence relations between the sensor events \textit{iup}/\textit{idn} and \textit{eup}/\textit{edn} is given in Fig. 3(a) and (b). This, along with the default restriction that the \textit{iup}/\textit{idn} and \textit{eup}/\textit{edn} events alternate, are used to draw the \(\Sigma_a\)-diagram shown in Fig. 8.

- **Step 5**: Generate the overall automaton model of the system by taking a strict synchronous composition of the automaton models of steps 4 and 5. This final automaton model of the transporter, is shown in Fig. 2. It can be verified to be the correct model of the system.

**Remark 3**: For a class of systems having binary-valued actuator and sensor signals, whose states are given by values of actuator and sensor signals, and that are modeled using automata, it is possible to extract the initial conditions, sensor event occurrence rules, and precedence relations between sensor events in the system as follows.

1) The initial condition can be read off from the initial state of the automaton.

2) To obtain the occurrence rule for a sensor event, determine the states where this event is executable, and compute the disjunction of the terms representing the values
of the actuator signals in those states. The weight of each actuator signal term present in the disjunction is taken to be $+1$.

3) To obtain the precedence relation between a pair of sensor events, relabel the transitions on all other events by the label “epsilon,” and determine the precedence relation, if any, from the resulting relabeled state machine.

In other words, we can get back the model of our formalism starting from an automaton model of the system, establishing the equivalence of the two modeling formalisms.

VI. MODELING SYSTEMS WITH DES ABSTRACTIONS

The modeling methodology of the discrete event system in our formalism can be extended to model systems involving nondiscrete variables for which a discrete event system abstraction is being sought, such as process control systems. In such systems, the values of the continuous variables (flow rates, temperature measurements) are abstracted for the purpose of discrete event system modeling.

A. Proposed Modeling Formalism

The proposed model in our formalism for systems which possess DES abstractions is similar to that for a “pure” discrete event system; the only difference being that the weights attached to actuator signals are allowed to be nonnegative real values, i.e., they are no longer restricted to be binary valued.

1) Initial conditions: Initial conditions summarize the chosen initial state of the system from where it begins to evolve under the action of actuator signals and feedback in the form of sensor signals.

2) Weighted sensor event occurrence rules: The system model has rules, one per sensor event, which govern how the sensor events in the system are influenced by the values of actuator signals. The antecedents of these rules are Boolean formulae over actuator signals.

Weights are associated with each actuator signal indicating the degree to which they influence the sensors in the system. These are nonnegative real values, which in contrast to “pure” DES’s where the weights were binary valued.

We compute the weight of a rule for each combination of actuator signal values which causes an antecedent of a sensor event occurrence rule to become true. The weights corresponding to the actuator signals are combined using either a minimum or a summation operation depending on whether the combination of the actuator signals is an AND or an OR combination. When the composition is based upon minimum and summation operations, the weight of a rule is defined inductively as follows. For the AND composition, it is given by: $w(a_1 \land a_2) = \min (w(a_1), w(a_2))$. For the OR composition: $w(a_1 \lor a_2) = w(a_1) + w(a_2)$. Thus, for example, $w((a_1 \land a_2) \lor a_3) = \min (w(a_1), w(a_2)) + w(a_3) = \min (w(a_1), w(a_2), w(a_3))$.

At a certain value of actuator signals, the weights are used to determine whether an upgoing or a downgoing transition of any particular sensor transition can occur, i.e., the dominance of one rule over the other, for any of the sensor signals’ two conflicting sensor events, is established.

B. Algorithmic Derivation of an Automaton Model

The overall automaton model is obtained in a manner similar to the algorithm in Section V. An example drawn from a process control system illustrating the steps involved in deriving an automaton model from a model in our formalism is provided. The tank has one filling tap $t_2$, one draining tap $t_1$, and a nominal level sensor $n$, as shown in Fig. 9. The events that can occur in this system are $t_1$, $t_2$, $\overline{t_1}$, and $\overline{t_2}$. Assume that the filling rate of tap $t_2$ has the flow value of $10$, while that of the draining is $1$. The model of the tank system in our formalism is shown in Fig. 10.

An equivalent automaton model of the tank system is derived using the algorithm described above as follows.

1. Initial conditions: $t_1 = t_2 = \text{off}$; $n = \text{down}$.

2. Weighted sensor event occurrence rules: Since there is 1 sensor signal, $n$, which has 2 sensor events associated with it, $\text{nup}$/$\text{ndn}$, there are 2 rules for this system: Filling on $\Rightarrow$ level goes up; Draining on $\Rightarrow$ level goes down.

   $\text{Rule}_{\text{nup}}^\text{up}: t_2^{+10} \Rightarrow \text{nup},$
   $\text{Rule}_{\text{ndn}}^\text{up}: t_1^{+1} \Rightarrow \text{ndn}$.

3. Precedence relations over sensor events: None, since only one sensor $n$. The only restriction on sensor events is the default one that $\text{nup}$/$\text{ndn}$ alternate.
when the tap \( t_2 \) is on, with a flow rate of +10, and, from Rule \( \text{s}^{dn}_2 \), it follows that draining occurs when the tap \( t_1 \) is on, with a flow rate of +1. So we add self-loops on \( \text{nup} \) with weight +10, in states where \( t_2 \) is on, and on \( \text{ndn} \) with weight +1, in states where \( t_1 \) is on. The resulting \( \Sigma_s \)-augmented \( \Sigma_a \)-diagram is shown in Fig. 12.

- **Step 3**: Refine the \( \Sigma_s \)-augmented \( \Sigma_a \)-diagram by deleting at least one of the self-loops from those states, where both sensor events of the same signal are present. Determine which sensor event of the same signal is feasible by comparing the weights of the two corresponding sensor event rules, in order to resolve contention. If the weights of both events are equal delete both the self-loops, since neither would be possible. In the present example, Rule \( \text{s}^{up}_1 \) dominates Rule \( \text{s}^{dn}_1 \), i.e., \( \text{ndn} \) is not possible when both the filling tap \( t_2 \) and the draining tap \( t_1 \) are on. This is because the weight of Rule \( \text{s}^{up}_1 \), which is +10, is larger than the weight of Rule \( \text{s}^{dn}_2 \), which is +1. Since contention between \( \text{nup} \) and \( \text{ndn} \) arises only in those states where both \( t_1 \) and \( t_2 \) are on, and for those states, the automaton of Step 2 is refined by deleting the infeasible transition, namely the self-loop on \( \text{ndn} \). The resulting automaton is shown in Fig. 13.

- **Step 4**: The structure of the system needs to be taken under consideration while drawing the precedence relations. Since the tank has only one binary-valued sensor, \( n \), there are no precedence relations over sensor events, and the only restriction is that the \( \text{nup/ndn} \) events of this sensor alternate. This is shown in Fig. 14.

- **Step 5**: Generate the overall model of the system by taking a strict synchronous composition of the automaton models of steps 4 and 5. The resulting automaton model of the overall system is shown in Fig. 15, which is the correct model of the tank system shown in Fig. 9.

### VII. An Additional Process Control Example

We illustrate the model of our formalism described in Section IV through more complicated examples taken from process control systems. These involve the use of AND and OR configurations of actuator signals in the layout of the system. Obtaining accurate automaton models of the system is not neces-
sarily straightforward. Our modeling formalism makes the modeling simpler.

Consider a tank with ANDed filling taps, and a single draining tap shown in Fig. 16. This system has the following sensors and actuators:

- Tap $t_1$ for draining the tank. Its being open/closed is denoted by $t_{1on}/t_{1of}$.
- Tap $t_2$ for filling the tank. Its being open/closed is denoted by $t_{2on}/t_{2of}$.
- Tap $t_3$ for filling the tank. Its being open/closed is denoted by $t_{3on}/t_{3of}$.
- Level sensors for sensing fluid nominal ($n$), and high ($h$) levels in the tank. The nominal level sensor being high/low is denoted by $nup/ndn$, and the high level sensor being high/low by $hup/hdn$.

The events that can occur in this system are $t_{1on}, t_{1of}, t_{2on}, t_{2of}, t_{4on}, t_{4of}, nup, ndn, hup, hdn$. The weights associated between the filling taps $t_2$ and $t_3$ and $nup$ are $+1, +10$; while that between the draining tap $t_1$ and $ndn$ is $+10$.

Proceeding as before the model in our formalism is given in Fig. 17. Since the taps $t_1$ are in an ANDed configuration, the weight for $nup$ and also that of $hup$ evaluates to:

1) $\min(+1, +10) = +1$. when $t_2$ and $t_3$ are both on;

2) $0$, when at least one of the filling taps $t_2$ and $t_3$ is off.

The weight of $ndn/hdn$ always evaluates to $+10$ whenever the draining tap $t_1$ is turned on. Contention for the $nup/ndn$ or the $hup/hdn$ event can occur in certain states of the system, owing to the presence of both filling and the draining taps. The overall automaton model obtained after combining the models shown in Fig. 18 consists of 24 states, and is shown in Fig. 19.
VIII. CONCLUSION

Circumventing the excessive hardship involved in modeling discrete event systems (DESs) have been examined, and a new modeling formalism has been proposed. This new formalism presents a scalable as well as flexible alternative to the modeling of discrete event systems. Such models may be automatically converted to their equivalent automata models. The technique has been demonstrated to work for a class of discrete event systems comprised of Boolean valued input/output signals. Using the proposed formalism, the manner in which the system is constructed provides a solution for addressing the problem of state space explosion which arise while obtaining automata model of DESs. The model we propose, being polynomial in the size of the systems signals, greatly aids rapid reconstruction and error-checking of the automaton model, especially when elements are added, removed, or configured differently within the system. Since the time this paper was written, ongoing research has further simplified the modeling formalism, absorbing the precedence relations into the event occurrence rules. As a consequence of this, the procedure for automated derivation of an equivalent automata model of the system from the rules has been simplified as well [4]. Further research will include introduction of temporal guards in order to capture information regarding the area of discrete event simulation, system modeling, lean manufacturing, failure diagnosis, and in modern classroom teaching techniques.

Mr. Chandra is the recipient of the Research Challenge Trust Fund (RCTF) fellowship at the University of Kentucky, Lexington. He is a registered engineer intern (EI) in the state of Kentucky and holds certifications from Sun Microsystems in Java (SCJP2), Microsoft professional (MCP 2000) and office (MOUS), Cisco networking (CCNA 2.0), and ComTIA Network+, iNet+. He is a student member of the Instrumentation, Systems and Automation Society (ISA).

Prof. Kumar was a recipient of the Microelectronics and Computer Development (MCD) Fellowship from the University of Texas at Austin and was awarded the Lalit Narain Das Memorial Gold Medal for the Best EE Student and the Ratan Swarup Memorial Gold Medal for the Best All-rounder Student from the Indian Institute of Technology at Kanpur, India. He is a recipient of the NSF Research Initiation Award, NASA-ASEE summer faculty fellowship award, ARL Sabbatical Fellowship, and the ANL summer faculty fellowship. He has served on the program committee for the IEEE Control Systems Society, the International Workshop on Discrete Event Systems, and the International Symposium on Intelligent Control. He is an Associate Editor of the SIAM Journal on Control and Optimization, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, and is a member of the IEEE Control Systems Society.

REFERENCES


Vigyan Chandra

Vigyan Chandra received the B.S. degree in electrical and electronics engineering from Birla Institute of Technology, Ranchi, India, in 1991 and the M.S. degree in electrical engineering from the University of Kentucky, Lexington, in 1997. He is currently working toward the M.S. degree in manufacturing systems engineering and the Ph.D. degree in electrical engineering at the University of Kentucky.

After completing his undergraduate work, he worked with Hindustan Motors, Ltd. (HML), Pithampur, India, in the electrical maintenance department until 1995, leaving as a Senior Engineer. His academic interests are in the area of discrete event simulation, system modeling, lean manufacturing, failure diagnosis, and in modern classroom teaching techniques.

Ratnesh Kumar

Ratnesh Kumar (S’90–M’92–SM’00) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology at Kanpur, India, in 1987, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, TX, in 1989 and 1991, respectively.

In 1991 he joined the faculty of University of Kentucky, Lexington, where he is an Associate Professor of Electrical and Computer Engineering. He has held visiting position at the Institute of Systems Research at the University of Maryland at College Park, the Applied Research Laboratory at the Pennsylvania State University, the NASA Ames Research Center, and the Argonne National Laboratory—West, Idaho-falls.

His primary research interest is discrete event systems and its applications to communication networks, manufacturing, formal verification, and intelligent control systems. He is co-author of the book Modeling and Control of Logical Discrete Event Systems (Boston, MA: Kluwer, 1995).

Prof. Kumar was a recipient of the Microelectronics and Computer Development (MCD) Fellowship from the University of Texas at Austin and was awarded the Lalit Narain Das Memorial Gold Medal for the Best EE Student and the Ratan Swarup Memorial Gold Medal for the Best All-rounder Student from the Indian Institute of Technology at Kanpur, India. He is a recipient of the NSF Research Initiation Award, NASA-ASEE summer faculty fellowship award, ARL Sabbatical Fellowship, and the ANL summer faculty fellowship. He has served on the program committee for the IEEE Control Systems Society, the International Workshop on Discrete Event Systems, and the International Symposium on Intelligent Control. He is an Associate Editor of the SIAM Journal on Control and Optimization, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, and is a member of the IEEE Control Systems Society.