On the Relationship between Finite State Machine and Causal Network Representations for Discrete Event System Modeling: Initial Results

Gregory Provan[†] and Yi-Liang Chen
Rockwell Science Center
1049 Camino Dos Rios, Thousand Oaks, CA 91360, USA
{*gmprovan, ylchen*}*@rsc.rockwell.com*

Abstract

This paper shows the relationship between two discrete event system representations, finite state machines and causal networks. Finite state machine models have been used extensively for the supervisory control of logical (and timed, with some extension) discrete event systems. On the other hand, Causal Networks have been applied mainly to the diagnosis of discrete event systems. Recent advances in finite-state-machine-based diagnosis and causal-network-based control have prompted an interest in understanding the relationship between these two representations. We describe initial findings concerning the mappings between these two representations for modeling synchronous system components, and discuss the implications of their relationships. We demonstrate the relationship using an example of a factory conveyor system.

1 Overview

This document provides a means for understanding the relationship between two discrete event system (DES) modeling methologies: the standard finite state machine (FSM) approach and a model-based reasoning approach, causal networks (CNs) [4]. Both representations have been used for the control ([1] for FSM and [6] for CN) and diagnosis ([7, 2] for FSM and [3] for CN) of DESs. Whereas the FSM approach is well-understood with respect to control modeling and less well-understood with respect to diagnostic inference, the converse is true for the CN approach. This article provides the initial step toward examining the correspondences between these two representations. This could result in cross-fertilization between these two approaches, thereby creating a representation that is equally well-understood and adequate for both control and diagnositics of DESs.

In this document we present our preliminary mappings between FSMs and CNs for modeling synchronous *system components*. The existence of such mappings provides a number of contributions to DES studies in general. First, mapping an FSM plant model into a CN model provides a means for plant model verification using CN model-verification techniques. Second, a CN representation is a more compact representation that is closer to system physical configurations/properties, which may be more convenient for certain applications. Third, given a CN simulation model, diagnostics can be generated directly through simple extensions to the simulation model and the application of CN diagnostics algorithms; this approach does not require the generation of additional system-specific diagnoser structures, as required in [7]. Fourth, the well-understood supervisory control theory of the FSM approach can be applied to CN models of DESs, thereby providing a clear semantics to CN control models.

We organize the remainder of the document as follows. Section 2 introduces the notations used for FSM and CN modeling. In Section 3, we present our preliminary mappings between FSM and CN representations for modeling system components, and proves that the languages generated by both representations through such mappings are equivalent. Finally, Section 4 discusses the implications of these results, and proposes future study of these relationships.

2 Two Modeling Approaches

2.1 Causal Network Modeling

We first describe how to specify DES (plant) behavior in terms of a causal network (plant) model Φ [3]. The (plant) model describes the *physics* of the system, such as distances, speeds, and their relationships which can be viewed as an actuator-to-sensor map. Note that, for diagnosis purposes, the plant model Φ simulates the system behavior under both normal and abnormal *component modes* (*i.e.*, operating modes of the components) [3]; in this document we focus solely on normal behaviors. We can augment the plant model to specify the control of the plant using the same causal model [6].

[†]Corresponding author. Tel: +1 (805) 373-4726. Fax: +1 (805) 373-4383.

A causal network model is a tuple $\Phi = (\mathbf{V}, \mathcal{G}, \Delta)$, where $\mathbf{V}$ is a set of variables, $\mathcal{G}$ is a directed graph called a *causal structure* whose nodes are members in $\mathbf{V}$ and edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$, and Δ is a set of *sentences* constructed from members in $\mathbf{V}$ based on the topological structure of $\mathcal{G}$.

Variables in $\mathbf{V}$ represent the components, modules, or system properties, such as sensors, motors, or power level. Each variable in $\mathbf{V}$ has a finite set of discrete values. We define $\mathbf{V}_{obs}$, the set of *observables*, as the subset of variables in $\mathbf{V}$ whose values can be either directly measured (sensor) or set (actuator). Similarly, $\mathbf{V}_{unobs}$ (the set of *unobservables*) consists of the remaining variables in $\mathbf{V}$ whose values cannot be directly detected. For control and discrete event applications, the observables are usually sensors and actuators while the unobservables represent internal components (*e.g.*, motors) or properties (*e.g.*, velocity and position).

The causal structure $\mathcal{G}$ is a directed acyclic graph that specifies the causal relationships over the variables. Each node V_i of $\mathcal{G}$ represents a unique element in $\mathbf{V}$ and each directed arc from V_1 to V_2 represents the causal influence of V_1 on V_2. The *family* of a variable V, $\mathcal{F}(V)$, consists of the union of V and the k parents of V in $\mathcal{G}$, which we denote $Pa_i(V),\ i = 1, .., k$.

We formally describe the syntax and semantics for the sentences in Δ that specify how the values of different variables interact as follows. Given a set of variables $\mathbf{V}$, a *sentence* is defined recursively as follows: (i) $[V = v]$ is a sentence, where $V \in \mathbf{V}$, and v is a value of V; and (ii) $\neg\alpha, \alpha \vee \beta$, or $\alpha \wedge \beta$ is a sentence where α and β are sentences. We call $[V = v]$ a *V-literal*.

For each variable $V \in \mathbf{V}$, a set of equations Δ_V, called the *database* of V, is defined. These equations are of the form $\beta \supset \alpha$, where α and β are sentences, that must satisfy modularity, locality, and consistency conditions: (Locality) Δ_V can only mention elements in $\mathcal{F}(V)$; (Modularity) every non-vacuous clause entailed by Δ_V must include variable V; (Consistency) Δ_V is logically consistent for every V in $\mathbf{V}$. The *database* Δ is then the union of Δ_V for all V in $\mathbf{V}$. The semantics of Δ is simply that of a multivalued propositional logic.

2.2 Finite State Machine Modeling

An FSM is defined as $G = (X, \Sigma, \delta, x_0)$, where X is the state space, Σ is the set of events, δ is the partial transition function $\delta : X \times \Sigma \rightarrow X$ (and defines the transitions between states in X), and x_0 is the initial state of the system. More precisely, equation $\delta(x_1, \sigma) = x_2$ means that there is a transition of event σ from state x_1 to state x_2 in G. We can also define an FSM using a graphical representation, in which a node representing x_1 is connected to a node representing x_2 by a directed edge labeled with transition σ iff $\delta(x_1, \sigma) = x_2$. We further assume all the FSMs in this article are deterministic.

2.3 Illustrative Example

Consider the simple conveyor system shown in Figure 1. This system consists of a conveyor, two sensors, a motor, and an actuator/controller. There are three positions on the conveyor, X, Y, and Z; we assume that all variable domains are discrete. The sensor turns on whenever there is an object at its corresponding position. The motor can move the conveyor belt forward (from X to Z, denoted as $+$) or not move at all, according to the actuator command. The actuator/controller will, upon the "on" signal of *sensor-x* (which means there is an object at X), issue the actuator command to move the object to Z and, upon the "on" signal of *sensor-z*, issue the actuator command to stop the conveyor. We assume that the object is removed from position Z, at which point *sensor-z* goes "off", and the cycle can restart. For simplicity of exposition we assume that the system is atemporal; we can capture the notion of a variable changing state, e.g., the position Pos of an object having moved from X to Y, by using previous and current variables for position, $[Pos_{Prev} = X]$ and $[Pos = Y]$ respectively. For a similar approach to failure diagnosis of the linear-axis system when time is considered, refer to [2, 3].

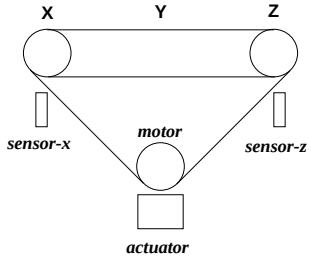


Figure 1: A Simple Linear-Axis System.

To model this simple system directly as a causal network, we construct a graphical model as shown in Figure 2. This model shows a directed acyclic graph with nodes for the actuator (ActM), motor (Motor), position (Pos), and sensors (Sensor-X, Sensor-Z). Directed arcs show the causal influences, e.g., the motor's state (on or off) is causally influenced by the actuator's state (on or off), etc. We show feedback control using dashed arcs from each sensor to the actuator.

Note that this CN model explicitly shows physical structural relationships. For example, it shows how the motor's state (on or off) is causally influenced by the actuator's state (on or off), but is not directly causally related to the sensor outputs.

For the FSM approach, this example is best described by the FSM models of its components (*e.g.*, conveyer belt, sensors, and motor). The complete system behavior can then be obtained through parallel synthesis

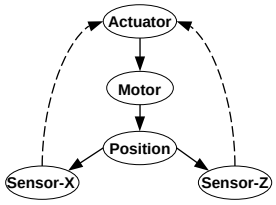


Figure 2: A Causal Network model for the simple conveyor system. The arcs for the physical causal links are solid, and the arcs for the control causal links are dashed.

of these components [1]. We will present some of the FSM component models (and their corresponding CN models) for this example in Section 3.3.

3 Relationships between Causal Networks and Finite State Machines

This section describes our initial steps toward understanding the relationships between finite state machine models and causal network models for *system components*. Note that the entire system model can then be composed from the component models through standard procedures in the respective approaches as described in [3, 1].

A **component causal network model** for a component C is fully specified by the family of C, $\mathcal{F}(\mathcal{C})$, and the equations for C, Δ_C. In other words, the only variables mentioned in Δ_C are in $\mathcal{F}(\mathcal{C})$. In logical terms, we can define the equations in Δ_C as a set of equations of the form:

$$[Pa_1(C) = \xi_i] \wedge \cdots \wedge [Pa_q(C) = \nu_j] \Rightarrow [C = c_i],$$

where ξ_i, ν_j and c_i are values of $Pa_1(C)$, $Pa_q(C)$ and C respectively. By the locality and modularity conditions, Δ_C describes all equations governing the behavior of C.

For many applications, we augment the above equational form with a previous-state parent: every variable can depend on its previous state C_{prev} in addition to its other parents. Note that the domain of C_{prev} is the same as the domain of C. Consequently, the full set of equations takes the form:

$$[Pa_1(C) = \xi_i] \wedge \cdots \wedge [Pa_q(C) = \nu_j] \wedge [C_{prev} = c_k] \Rightarrow [C = c_i],$$

where ξ_i, ν_j, c_k and c_i are values of $Pa_1(C)$, $Pa_q(C)$, C_{prev}, and C respectively. We call all parent nodes of a node C, exclusive of the previous-state parent C_{prev}, *conditional parents* of C.

It turns out that an FSM corresponds to a restricted form of CN, which we call Boolean canonical form. A CN model for a component C is in a *Boolean canonical* form if the domain of all its conditional parent variables is Boolean (*i.e.*, $\{true, false\}$), and every equation for C is either of the form $([V_n = true] \wedge [C_{prev} = c_k] \Rightarrow [C = c_i])$ or $([V_n = true] \Rightarrow [C = c_i])$ for some conditional parent V_n.

Given a causal network model for a component C, we can always transform it into a Boolean canonical form using the following steps:

1. Convert each equation $[Pa_1(C) = \xi_i] \wedge \cdots \wedge [Pa_q(C) = \nu_j] \wedge [C_{prev} = c_k] \Rightarrow [C = c_i]$, into $[Pa_1(\xi_i) \circ \cdots \circ Pa_q(\nu_j) = true] \wedge [C_{prev} = c_k] \Rightarrow [C = c_i]$, where the symbol $\circ$ denotes the concatenation of two value strings into a single variable symbol;
2. Remove all original conditional parent nodes, create a conditional parent node for every distinguished variable symbol (except C and C_{prev}) in the converted equations.

As can be observed, the semantics of the new variable $[Pa_1(\xi_i) \circ \cdots \circ Pa_q(\nu_j) = true]$ is $[Pa_1(C) = \xi_i] \wedge \cdots \wedge [Pa_q(C) = \nu_j]$. We assume that all the causal network component models are in Boolean Canonical form, unless specified otherwise.

An *event* of label V_i when the value of C is c_j can be *generated* by a causal network component model (of Boolean canonical form), if either of the equations $[V_i = true] \wedge [C_{prev} = c_j] \Rightarrow [C = c_k]$ or $[V_i = true] \Rightarrow [C = c_k]$ exists in Δ for some value c_k of C. The value of C becomes c_k after the occurrence of the event V_i. We denote this occurrence of the event and its result on the value of C by the function $\tau(c_j, V_i) = c_k$. Similarly, a string of events $s = V_1 \ldots V_n$ can be generated when C has the value c_1, if $\tau(c_i, V_i) = c_{i+1}$, for some c_{i+1} of C, for $i = 1, \ldots, n$. We then denote $\tau(c_1, s) = c_{n+1}$. We further define that for an empty string ϵ, $\tau(c_j, \epsilon) = c_j$, for all value c_k of C. Finally, the *language* generated by a causal network component model Φ when C has an initial value of c_0, denoted as $L(\Phi, c_0)$, contains all the strings that can be generated by Φ when C has the value of c_0.

A **component FSM** G for a component C consists of the states in X defining the behavior of the component, together with the transitions of events in Σ relating those states. In logical terms, we can define the tuples corresponding to δ using a set of equations of the form $x_i \wedge \sigma_m \Rightarrow x_j$, where $(x_i, \sigma_m, x_j) \in \delta$. In this equational form, we can think of x_i as the previous state (or value of the component C), and x_j as the current state (or value of the component C).

3.1 FSM to CN Mapping

We now define an algorithm φ for mapping a component FSM model into a component CN model. Given a component FSM model $G = (X, \Sigma, \delta, x_0)$, as speci-

fied above, we map this to a component causal network model $\Phi = (\mathbf{V}, \mathcal{G}, \Delta)$ as follows:

1. Create a CN node for variable C with domain X. That is, each state labeled x in X corresponds to a unique value x of C;
2. Create a CN node for variable C_{prev} with domain X. This node is the previous state parent node of C. Define the initial value of C_{prev} as x_0;
3. For each event label $\sigma \in \Sigma$, create a CN node for variable σ that has a Boolean domain (*i.e.*, $\{true, false\}$). Each CN node for σ is a conditional parent of C; and
4. Generate the set of CN equations Δ_C, with an equation for each transition in δ, using the transformations for state space and transitions described above. For example, for transition $\delta(x_i, \sigma_i) = x_j$, we generate a CN equation $[\sigma_i = true] \wedge [C_{prev} = x_i] \Rightarrow [C = x_j]$.

To show the CN model create by the above procedure is equivalent to the FSM model, we show that they generate the same language.

Theorem 1 $L(G) = L(\Phi, x_0)$

Proof: We prove by induction on the length of the strings generated by the two languages. First, by definition, $\epsilon \in L(G)$ and $\epsilon \in L(\Phi, x_0)$; $\delta(x_0, \epsilon) = x_0$ and $\tau(x_0, \epsilon) = x_0$. Then, we assume that for all strings s and some positive integer n, such that $|s| \leq n$, we have $s \in L(G) \Leftrightarrow s \in L(\Phi, x_0)$. Moreover, $\delta(x_0, s) = x_n$ and $\tau(x_0, s) = x_n$, for some $x_n \in X$. On one hand, for all $\sigma \in \Sigma$ such that $s\sigma \in L(G)$ and $\delta(x_n, \sigma) = x_{n+1}$ for some x_{n+1}, we know from steps 1 to 3 of our mapping, x_{n+1} is in the domain of both C and C_{prev} and σ is a conditional parent node of C. Meanwhile, from step 4, there exists an equation $[\sigma = true] \wedge [C_{prev} = x_n] \Rightarrow [C = x_{n+1}]$. Hence, $\tau(x_n, \sigma) = x_{n+1}$ and $s\sigma \in L(\Phi, x_0)$. On the other hand, for all conditional parents V of C such that $sV \in L(\Phi, x_0)$ and $\tau(x_n, V) = x_{n+1}$ for some value x_{n+1} of C, we know from steps 1 to 3 that $x_{n+1} \in X$ and $V \in \Sigma$. Meanwhile, from step 4, there exists a transition $\delta(x_n, V) = x_{n+1}$ such that $sV \in L(G)$. Therefore, $L(G) = L(\Phi, x_0)$. ∎

Corollary 1 *The component FSM to component CN mapping φ is homomorphic.*

The result of Corollary 1 comes directly from Theorem 1. Note that the above mapping algorithm will lead to a CN in Boolean canonical form. This is not a compact CN; in particular, mapping every transition into a binary CN node is not a compact representation, as several transitions can usually be view as effectively just different values of a single CN node, rather than as multiple nodes as required in Boolean canonical form.

3.2 CN to FSM Mapping

Given a component causal network model (in a Boolean canonical form) $\Phi = (\mathbf{V}, \mathcal{G}, \Delta)$ for a component C, as specified above, we map this to a component FSM model $G = (X, \Sigma, \delta, x_0)$ as follows:

1. Map the domain of the CN (child) node C into the state space X. That is, for each possible value c_i of C, create a state $c_i \in X$.
2. Create the set of event labels in Σ by assigning each equation Δ_C^i in Δ_C for node C an event label $\sigma_i \in \Sigma$.
3. Map the set of CN equations in Δ_C into δ, using the transormations as outline below. (1) for an equation Δ_C^i of the form $[V_k = true] \wedge [C_{prev} = c_m] \Rightarrow [C = c_n]$ we create a transition $\delta(c_m, \sigma_i) = c_n$, where σ_i is the event label assigned to this equation. (2) for an equation Δ_C^i of the form $[V_k = true] \Rightarrow [C = c_n]$, we create transitions $\delta(c, \sigma_i) = c_n$, for all $c \in X$.
4. Map the initial value of C, c_0, into the initial state of G.

Theorem 2 *The component CN to component FSM mapping is homomorphic.*

Proof: Similar to the proof of Theorem 1. ∎

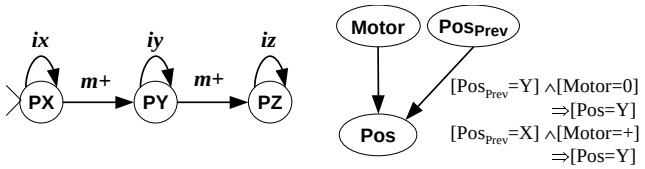
3.3 Conveyor Example

This section shows how the mappings work for the simple conveyor system shown in Figure 1. We describe how we would model two "components" of this system *i.e.*, the conveyor and the sensor-x, directly as individual CN and FSM component models. Then we outline how the CN-FSM mappings work on these models.

Conveyor: We now examine the model for the belt itself, *i.e.*, the model for the physics of the conveyor belt transporting an object from X to Y to Z.

We model a conveyor using the FSM approach as follows. Figure 3(a) shows the state machine model G_{conv} for the conveyor. Starting from the initial state PX as indicated by the wedge, the self-loop transition of event ix indicates the existence of an object at position X of the conveyor. Event $m+$ represents the forward movement of the motor that brings the object from one position to the next position on the conveyor. Hence, a transition of $m+$ brings the system from PX to PY ("object at position Y"), and another transition of $m+$ will further bring the object to position Z (*i.e.*, state PZ), where event iz occurs to indicate the existence of an object at Z. Finally, the event iy occurs when the object remains at position Y (due to the motor being off).

We model a conveyor using the CN approach as follows. We base our model on a qualitative physical specification of the conveyor. Following are the equations for



(a) FSM Model ***(b) CN Model***

Figure 3: (a)G_{conv}: state machine for the conveyor; (b) Causal network for the conveyor with equations for Position= Y

the *position (Pos)*, which are based on the physical law of motion $Pos = Pos_{prev} + vt$, where the motor action (either forward (+) or 0) plays the role of velocity v, and Pos and Pos_{prev} denote the distance and previous distance respectively. In a qualitative sense, Pos is causally dependent on Pos_{prev} and the Motor values, ignoring the actual value of time t, as given by the function $Pos = f(Pos_{prev}, Motor)$. The causal graph for the conveyor is shown in Figure 3(b), and the equations for the graph are depicted in Table 1. In the third column of Table 1, we denote the transition corresponding to the portion of the equation. Note that we can suppress the explicit depiction of a node corresponding to Pos_{Prev} in a causal network graphical model, since in the graph we need to represent only a single node for the position (Pos), such that the possible values of this node include bindings for Pos_{Prev} as well as Pos. In subsequent CN models we use this more compact depiction. This compactness of representation can be useful in certain circumstances, as it can suppress many details that may be unnecessary for particular applications.

Consequent variable	Antecent variables	Assoc. Trans.
$(Pos = X)$	$(Pos_{Prev} = X) \wedge (Motor = 0)$	ix
$(Pos = Y)$	$((Pos_{Prev} = Y) \wedge (Motor = 0))$	iy
	$((Pos_{Prev} = X) \wedge (Motor = +))$	m_x^+
$(Pos = Z)$	$((Pos_{Prev} = Z) \wedge (Motor = 0))$	iz
	$((Pos_{Prev} = Y) \wedge (Motor = +))$	m_y^+

Table 1: Causal network equations for conveyor (of the form *antecedent* $\supset$ *consequent*), with associated FSM transitions in column 3.

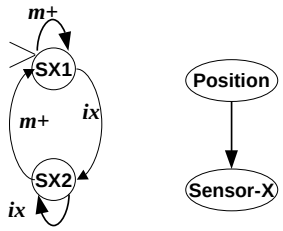
We perform the CN↔FSM mappings as follows:

M1: FSM→CN (1) create a CN node C with domain given by the state space of the corresponding FSM variable; noting that all FSM variables in Figure 3(a) are Position variables, we create the domain $\mathbf{X} = \{X, Y, Z\}$. (2) create the variable C_{prev} with domain $\mathbf{X}$ and initial value X. (3) create Boolean event variables E_{ix}, E_{m+}, E_{iy}, E_{iz}. (4) create CN equations: three equations, for $\vartheta = X, Y, Z$: $[E_{i\vartheta} = true] \wedge [C_{prev} = \vartheta] \supset [C = \vartheta]$, plus $[E_{m+} = true] \wedge [C_{prev} = X] \supset [C = Y]$; $[E_{m+} = true] \wedge [C_{prev} = Y] \supset [C = Z]$;

M2: CN→FSM (1) create the FSM state space from the Position node Pos: $\mathbf{X} = \{X, Y, Z\}$. (2) create event labels based on CN equations, as given by Table 1: $\Sigma = \{ix, iy, iz, m_x^+, m_y^+\}$. (3) create event transitions from CN equations, as given by Table 1: $\delta(X, ix) = X$, $\delta(X, m_x^+) = Y$, $\delta(Y, m_y^+) = Z$, $\delta(Y, iy) = Y$, $\delta(Z, iz) = Z$. (4) create the initial state of the FSM, $x_0 = X$.

It is trivial to verify that the CN created by mapping $M1$ is identical to the hand-created CN of Figure 3(b), and that the FSM created by mapping $M2$ is very similar to the hand-created FSM of Figure 3(a).

Sensor-x: The state machine model for sensor-x, $G_{sen-x}$is shown in Figure 4(a). Upon the arrival of an object at position x (*i.e.*, event ix occurs), the sensor-x moves from initial state $SX1$ to state $SX2$, indicating that the sensor turns on (*i.e.*, outputs an "on" signal). Any motor actions (*i.e.*, $m+$) will then bring the sensor from state $SX2$ to its initial state.



(a) FSM Model ***(b) CN Model***

Figure 4: G_{sen-x}: state machine for Sensor-x.

The CN component model for sensor-x is straightforward. As shown in Figure 4(b), the position of the object on the conveyor is the only variable that affects the output of sensor-x (assuming normal sensor operation). We list sensor-x causal equations in Table 2.

Consequent variable	Antecent variables	Assoc. Trans.
$(xSensor = on)$	$(Pos = x)$	ix
$(xSensor = off)$	$(Pos = y)$	iy
	$(Pos = z)$	iz

Table 2: Causal network equations for sensor-x.

CN↔FSM mappings of the sensor-x models can be performed similar to those for the conveyor described previously. However, we note that, though sematically similar, the models created by the mapping procedures we present in this paper are not similar to the hand-created models topologically. We will follow up this observation with further discussions in the next section.

4 Discussion

This article documents our first attempt to establish the relationship between the tranditional finite state machine approach and the emerging causal network approach in modeling discrete event systems. We presented a set of mappings between system components modeled in FSM and CN representations, showing the existence of an "equivalence" relationship between these two modeling methodologies. However, we believe that the mappings between FSM and CN are not unique, and the mappings shown here are rather primitive. In pointing out the problems and shortcomings of our mappings in this section, we hope to be able to shed some light on the difficulties in establishing such a relationship, which can lead to further development of more sophisticated/"practical" mappings.

Our mapping from an FSM to a CN preserves the correctness of the underlying model, but it may not produce a compact or efficiently-computed CN model. If we perform this mapping on a component-wise basis, as done in our example, we do recover a CN with some physical structure. Note, however, that each component-wise CN is just a two-level CN. If we perform this mapping on a system FSM, the resulting equivalent CN will still be two-level, and will not preserve the physical structure integral to a CN representation. In contrast, an FSM model encodes no notion of system structure in a model. As a consequence, in the conversion of FSM to CN, unless we add extra information to the FSM, such as tagging nodes with a "structure tag", the CN that is created is just a 2-level CN (in a Boolean canonical form) with no real structural information. Further research is needed to define how to amend the structural information during the mapping from FSM to CN, and how to aggregate conditional parent nodes from the resulting Boolean canonical component CN into a compact yet structurally meaningful CN. These issues are especially important if we want to investigate the relationships of the system models (composed from component models) described in the two representations.

The reverse mapping, of CN to FSM, does not suffer from loss of structural detail. However, while the variables of a child node in a component CN are converted into states, the variables of the parent nodes have become events in the resulting FSM. Hence, it virtually denies the possibility to further synthesize the component FSM models generated through this CN-to-FSM mapping into a complete system model. This results from the primary difference between these modeling methodologies; whereas both states and transitions are *explicit* in an FSM, both states and transitions are *implicit* in a CN, and must be inferred from variable bindings. In this mapping, we proposed a simple yet limited way of inferring states from variables of the child node of a component CN model. To more properly infer the underlying states in CN models, more computationally involved methods (along the lines of the preliminary analysis we proposed in [5]) may be required. The task of properly inferring the underlying event transitions is even more difficult yet urgently needed, and correctly synchronizing transitions between different component models during model synthesis might require certain "event tags" to be introduced and appended to the original CN models.

Proper mappings between FSMs and CNs are important and very beneficial to the advancement of DES technologies in general. As a consequence of such mappings, it will be possible to map an FSM to a CN, and have the control properties of the FSM carry over to the CN, an important issue since control properties of CNs have not yet been fully investigated. A major benefit of using CNs is that diagnostic reasoning can be done much more compactly than using the diagnoser approach of FSMs. Second, the CN can be used to prove the correctness of plant models. Conversely, if we start with a CN control model, mapping it to a FSM means that we can use the already-developed techniques and theory for proving control properties, without having to develop this explicitly in the CN framework.

References

[1] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, 1999.

[2] Y.-L. Chen and G. Provan. Modeling and disgnosis of timed discrete event systems – a factory automation example. In *Proc. 1997 American Control Conf.*, pages 31–36, Albuquerque, NM, June 1997.

[3] A. Darwiche and G. Provan. Exploiting system structure in model-based diagnosis of discrete-event systems. In *Proc. 7th Intl. Workshop on Principles of Diagnosis*, pages 95–105, 1996.

[4] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[5] G. Provan and Y.-L. Chen. Modeling, diagnosis, and control of timed discrete event systems using temporal causal networks. In *Proc. 1998 Intl. Workshop on Discrete Event Systems (WODES'98)*, pages 152–154, Cagliari, Italy, August 1998.

[6] G. Provan and Y.-L. Chen. Model-based diagnosis and control reconfiguration for discrete event systems: An integrated approach. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 1762–1768, December 1999.

[7] M. Sampath, R. Sangupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosibility of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.