Verifying Properties of HMS Machine Specifications of Real-Time Systems*

A. Gabrielian and R. Iyer
Thomson-CSF, Inc.
Palo Alto, CA
U.S.A.

Abstract

A "hierarchical multi-state (HMS) machine" is an automaton in which multiple states may be true, multiple transitions may fire simultaneously, a state may be expanded into a lower-level HMS machine, and in which the transitions are "controlled" by predicates in a temporal interval logic called TIL. HMS machines provide a compact formalism for specifying and verifying the behavior of concurrent hard real-time systems. Two approaches to verification of properties of non-recursive HMS machines are presented: (1) an extension of tableau-based theorem proving that utilizes the logic TIL and the execution semantics of machines to verify safety properties, and (2) a variation of model checking that uses "interacting" parametric "computation graphs." The first verification approach avoids the construction of a complete computation graph and the second approach permits the analysis of behavior of certain types of HMS machines using multiple, relatively simple computation graphs.

1 Introduction

A variety of formalisms have been studied during the recent past for the specification and verification of real-time systems. Variations of finite-state automata, such as Büchi automata [AS89], temporal extensions of Petri nets [GM89], or modifications of standard temporal logic have been the key representation languages for many of these studies. The concept of "hierarchical multi-state (HMS) machine" is obtained by integrating hierarchical and parallel or "multi-state" automata with a temporal interval logic called TIL, resulting in a compact formalism for specifying concurrent hard real-time systems, in which complex temporal dependencies can be defined conveniently [GF88, FG89, GF90, GI91, Ga91].

In an HMS machine, multiple states may be true at a moment of time, multiple transitions may fire simultaneously, states may be expanded (recursively) into lower-level HMS machines, and transition enablement is defined by predicates in a temporal interval logic called TIL. The following are some of the key characteristics of HMS machines:

- Each transition in an HMS machine is specifically designated to be either *deterministic* or *nondeterministic*. At a particular moment of time, all enabled deterministic transitions and any subset of enabled nondeterministic transitions may fire. Thus, in contrast to Petri nets, multiple transitions from a single boolean state may fire simultaneously.

*This work was supported in part by the Office of Naval Research under Contract N00014-89-C-0022.

- The use of TIL predicates for defining conditions under which transitions are enabled provides a much richer language for controlling transitions than timed Petri nets [GM89] and time-based extensions of state models (see, e.g., [HMP91]). These TIL predicates may contain as a literal any state of the machine, thus allowing complex causal dependencies among states to be modeled quite naturally. The logic TIL allows a uniform treatment of time for the past, the present and the future, subsuming most of the common interval-based logics defined in the literature [DM88, Me88, PH88].

- A "multi-level specification" approach can be used to give a combination of axiomatic and executable specification of a system at multiple levels of abstraction that reduces the effort in specification and offers a high degree of reusability and modularity [GF90].

In [FG89] a method for verifying safety properties of HMS machines was presented that was based on the following scheme: (1) Given an HMS machine H and a safety property $\Box p$, an "extended" state Q for H is created that becomes true if and only if the safety property is violated. Thus, to prove that H satisfies $\Box p$, it is sufficient to show that Q is *unreachable*. (2) A set of correctness-preserving transformations are applied to H that modify the structure of H, without changing its behavior, such that the state Q becomes isolated (and unreachable) if the safety property is indeed satisfied by H. In [GI91], a branching time version of TIL called BTIL was defined and preliminary results on a model checking approach to verification of HMS machine properties were presented.

In this paper, two approaches to verification of real-time properties of HMS machines are presented. In the first approach, an extension of tableau-based theorem proving [Ab87, Fi90] is used to verify safety properties. The extension involves (1) the use of the interval-based TIL instead of standard temporal logic, and (2) the consideration of execution semantics of HMS machines, as well as logical alternatives, to derive facts about states and transitions. As in the correctness-preserving transformation technique [FG89], safety properties are represented as extended states and the refutation proceeds in a heuristic fashion. In the second approach, a variation of model checking is used on "interacting" parametric "computation graphs" to verify properties of HMS machines. This is possible when the associated machine can be partitioned into a set of components, such that separate but interacting computation graphs can be constructed for each component. The interactions are defined in terms of parametric delays that characterize the execution of nondeterministic transitions. Such a partitioning can reduce the size of a computation graph significantly and the use of parameters allows a simple encoding of nondeterministic behavior. The two verification techniques are illustrated for a railroad crossing example with nondeterministic transitions and real-time constraints.

In Section 2 we present an overview of HMS machines. We ignore multi-level specification and limit the discussion to non-recursive hierarchies. Formal definitions of multi-level specification and recursive hierarchies can be found in [GF90] and [GI91, Ga91], respectively. In Sections 3 and 4 we present our two verification approaches and in Section 5 we present a brief summary and the conclusions.

2 Overview of Hierarchical Multi-State (HMS) Machines

As indicated earlier, an HMS machine consists of a parallel or multi-state (hierarchical) automaton in which predicates in a temporal interval logic called TIL determine the conditions under which transitions are enabled. We assume that we are given a set S of "states" and we define a "marking" of S at the relative time t to be a mapping $M_t : S \rightarrow \{F, T\}$, where F and T represent false and true, respectively.

In addition, we call a sequence $\ldots, M_{-1}, M_0, M_1, \ldots$ of markings on a set S a "marking sequence." We consider a discrete model of time with 0 denoting the "current moment."

We now introduce the logic TIL, which is obtained by augmenting standard propositional logic with four temporal operators. In the sequel, given a marking M and a formula ψ, we denote the satisfiability of ψ in M by $M \models \psi$.

$\mathrm{O}(t')$	*At* relative time t'
	$M_t \models \mathrm{O}(t')\psi \Leftrightarrow M_{t+t'} \models \psi$
$[t_1, t_2]$	*Always* between t_1 and t_2
	$M_t \models [t_1, t_2]\psi \Leftrightarrow \forall t'\ t_1 \leq t' \leq t_2$ implies $M_t \models \mathrm{O}(t')\psi$
$<t_1, t_2>$	*Sometime* between t_1 and t_2
	$M_t \models <t_1, t_2> \psi \Leftrightarrow \exists t'$ such that $t_1 \leq t' \leq t_2 \wedge M_t \models \mathrm{O}(t')\psi$
$<t_1, t_2>!$	*Sometime-change* between t_1 and t_2
	$M_t \models <t_1, t_2>!\psi \Leftrightarrow \exists t'$ such that $((t_1 - 1) \leq t' < t_2) \wedge (M_t \models \mathrm{O}(t')\neg\psi) \wedge (M_t \models <t'+1, t_2> \psi)$

We note that standard temporal operators, including finite versions [DM88, PH88] are special cases of TIL operators. Thus, $\Box = [0, \infty]$, $\Diamond = <0, \infty>$, and $[t] = [0, t]$. The sometime-change operator $<>!$ is not normally defined in standard temporal logics. It plays an important role in the theorem proving technique to be introduced in Section 3.

We now turn to the definition of HMS machines and we introduce our visual notation for representing such machines. In this paper, we do not address recursive hierarchies [GI91, Ga91] and we assume that non-recursive hierarchies are expanded (or flattened) to yield a machine in which all transitions are between "primitive" states only. This involves no loss of generality in the absence of recursion.

Definition 2 An HMS machine is a triple $H = (S, \Gamma_D, \Gamma_N)$ such that

(a) S is a set of "states." If $\{A, B, \ldots\}$ a subset of S, we denote it by $(A, B, \ldots)$

(b) Γ_D and Γ_N consist, respectively, of "deterministic" and "nondeterministic" transitions of the form

$$\text{(PRIMARIES) (CONTROL)} \rightarrow \text{(CONSEQUENTS)},$$

where PRIMARIES $\subseteq S$, CONSEQUENTS $\subseteq S$ and CONTROL is a predicate in TIL with literals from the set S. For a particular transition $\gamma = (A, B, \ldots)\ (P) \rightarrow (E, F, \ldots)$ in $\Gamma_D \cup \Gamma_N$, we write PRIMS(γ) $= (A, B, \ldots)$, CNTRL(γ)$=P$, and CNSQS(γ)$=(E, F, \ldots)$.

Figure 2.1 presents the specification of a railroad crossing HMS machine example using our visual notation. Rectangles represent states, dark arrows denote transitions with nondeterminism indicated by an asterisk, and the short vertical bars connected to the transitions t_5 and t_7 denote infinite resources that are always true. TIL controls are indicated by a combination of VLSI notation and temporal operators next to the symbol Ⓣ. Thus, for example, for the deterministic transition t_4 and the nondeterministic transition t_5 in the figure, we have

PRIMS(t_4) $= (UP)$, CNTRL(t_4) $= LG \wedge [-20, 0]UP$, and CNSQS(t_4)$=DN$

PRIMS(t_5) $= (\ )$, CNTRL(t_5) $= [-10, 0]\neg TS$, and CNSQS(t_5)$=TS$.

Intuitively, the transition t_4 fires if its primary state UP is true and the following control condition holds: the state LG is true and UP has been true from -20 moments to the present (i.e., $[-20, 0]UP$). When the transition fires, UP becomes false and DN becomes true. Similarly, t_5 may fire in which case TS will become true (a new train starts), if $[-10, 0]\neg TS$ is true.

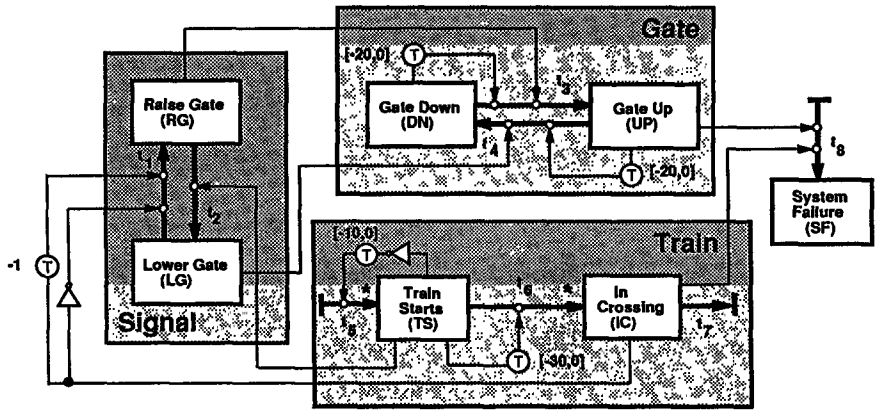

Figure 2.1. HMS Machine Specification of a Railroad Crossing

We note two major differences between the notion of nondeterminism in HMS machines and nondeterminism in traditional automata and Petri nets. In an HMS machine, a transition is explicitly designated as deterministic or nondeterministic. The firing of a nondeterministic transition is independent of the firing of all other transitions. In contrast, for traditional automata and Petri nets, nondeterminism is imposed by structural considerations. In addition, only one transition out of a set of related transitions may fire.

In this railroad crossing, the nondeterministic behavior of train arrivals is defined by the hierarchy Train, the signaling for raising and lowering the gate is defined by Signal, and the gate behavior is defined by Gate. The state SF is an "extended" state that defines the following "safety property" for the system: $\Box\neg(IC \wedge UP)$. This represents the requirement that the train should never be in the crossing when the gate is up. We verify this requirement in Sections 3 and 4 using two different verification techniques.

We consider next the semantics of execution for HMS machines. For a non-operational definition see [Ga91]. Here, we give an operational and logical definition of execution. For a transition u, by abuse of notation, we let (1) PRIMS(u)= $s_1 \wedge \ldots \wedge s_n$, where $\{s_1, \ldots, s_n\}$ is the set of primary states of u, and (2) CNSQS(u)= $q_1 \wedge \ldots \wedge q_m$, where $\{q_1, \ldots, q_m\}$ is the set of consequents of u. Also, we denote by $\Gamma_{in}(s)$ and $\Gamma_{out}(s)$ the sets of transitions into and out of s, respectively, and we use the symbol $\Rightarrow^*$ to represent a non-standard "nondeterministic implication," so that a formula $\lambda \Rightarrow^* \theta$ is interpreted as follows: if λ is true, θ *may* be true.

Definition 3 Given an HMS machine $H = (S, \Gamma_D, \Gamma_N)$, the set of transitions that will fire at the next moment is defined by the following two "Transition Firing" rules:

(TF-1) PRIMS(u) $\wedge$ CNTRL(u) $\Rightarrow$ O(1)u, where u is deterministic
(TF-2) PRIMS(u) $\wedge$ CNTRL(u) $\Rightarrow^*$ O(1)u, where u is nondeterministic.

The updating of values of states is defined by the following "State Updating" rule:

(SU) $\mathrm{O}(1)s \Leftrightarrow (s \wedge (\bigwedge_{u \in \Gamma_{out}(s)} \mathrm{O}(1)\neg u)) \vee (\bigvee_{v \in \Gamma_{in}(s)} \mathrm{O}(1)v)$,

where for a function ψ $\bigwedge_{x \in X} \psi(x) = T$ if $X = \{\}$ and $\bigvee_{x \in X} \psi(x) = F$ if $X = \{\}$.

At a moment of time, we first apply the Transition Firing rules to determine the set of transitions that will fire at the next moment ($t = 1$). We then apply the State Updating rule to determine the states of the machine that will be true at $t = 1$. If at $t = 0$ the control and all the primary states of a deterministic (nondeterministic) transition u are true then by TF-1 (TF-2) u will (may) fire. If u fires its consequent state will become true and a primary state s of u will become false unless some transition fires into s at the same moment. States that are literals in the TIL predicate CNTRL(u) are not affected by the firing of u.

3 Verifying Safety Properties by Theorem Proving

In this section, we introduce a tableau-based theorem proving method for verifying safety properties of HMS machines. A classical treatment of tableau-based theorem proving for propositional and first-order standard logic can be found in [Fi90]. Also, tableau-based theorem proving methods for standard temporal logic with the next operator (O) and mu-calculus appear in [Ab87] and [SW89], respectively.

The tableau method presented here is adapted from [Fi90] with the following two extensions: (1) facts are stated in the interval-based temporal logic TIL rather than propositional logic, and (2) rules based on the execution semantics of HMS machines are used to deduce additional facts during the theorem proving process. We limit our discussion to HMS machines with finite number of states and finite "past" controls which are inherently decidable. We note, however, that the state reachability problem for such machines has been shown to be NP-complete [FG89].

In Definition 3 of Section 2, we defined the semantics of execution of HMS machines in terms of two Transition Firing rules and a State Updating rule. We now introduce two additional sets of rules that are particularly useful in our theorem proving method.

Derived Execution Rules

The following three rules, which capture the key characterisitics of execution of HMS machines, are derived from a combination of the Transition Firing and State Updating rules of Section 2.

(DE-1) $<t_1, t_2>!\neg s \Rightarrow <t_1, t_2> (\bigvee_{u \in \Gamma_{out}(s)} u)$

(DE-2) $<t_1, t_2>!s \Rightarrow <t_1, t_2> (\bigvee_{v \in \Gamma_{in}(s)} v)$

(DE-3) $\mathbf{O}(t)u \Rightarrow \mathbf{O}(t-1)(\text{PRIMS}(u) \wedge \text{CNTRL}(u)) \wedge \mathbf{O}(t)\text{CNSQS}(u)$

Intuitively, rule DE-1 (DE-2) formalizes the following observation: if a state s becomes false (true) in an interval of time, then one or more transitions fire from (into) s in that interval. The rule DE-3 states that if a transition u fires at t, then its primary states and control were true at $t-1$ and its consequent states are true at t.

TIL Inference Rules

Let φ and ψ be TIL formulae, and let T stand for true. The following rules can then be derived using the definitions of TIL operators and propositional logic.

$$(\mathbf{O}(t)\varphi \wedge (\varphi \Rightarrow \psi)) \Rightarrow \mathbf{O}(t)\psi$$
$$<t_1, t_2>!\varphi \Rightarrow <t_1, t_2> \varphi$$
$$T \Rightarrow (<t_1, t_2> \varphi \vee [t_1, t_2]\neg\varphi)$$
$$[t_1, t_2]\varphi \Rightarrow (\mathbf{O}(t_1)\varphi \wedge \ldots \wedge \mathbf{O}(t_2)\varphi$$
$$<t_1, t_2> \varphi \Rightarrow (\mathbf{O}(t_1)\varphi \vee \ldots \vee \mathbf{O}(t_2)\varphi)$$

$$\mathrm{O}(t)[t_1, t_2]\varphi \Rightarrow [t_1 + t, t_2 + t]\varphi$$
$$\mathrm{O}(t) <t_1, t_2> \varphi \Rightarrow <t_1 + t, t_2 + t> \varphi$$

Given an HMS machine H, let $\Box p$ be a safety property of H where p is a "past" TIL formula (cf. [MP89]). Following the procedure introduced in [FG89], we represent this safety property by adding a deterministic transition w to H with an empty set of primary states (always true), an extended state Q as its consequent, and $\neg p$ as its TIL control ($Q = SF$ in Figure 2.1). It can be shown that *any* safety property about H can be represented in this manner. The proof that $\Box p$ holds for H in the theorem proving method to be introduced next is by refutation and is based on heuristics rather than exhaustive enumeration techniques. The refutation is accomplished by demonstrating that if Q has been false for a certain period of time, it can never become true.

Tableau-Based Theorem Proving Method

1. Define a "tableau" containing an "initial node" labeled with the formula $[-m, 0]\neg Q \wedge \mathrm{O}(1)Q$, which states the following assumption: machine H has been safe for $m+1$ moments and becomes unsafe at the next moment. The choice of m depends on the HMS machine considered and assumptions about the initial conditions. The initial node also defines a terminal node of a "path" to itself in the tableau.

2. For each terminal node N in the tableau, deduce a new fact φ using the Transition Firing, State Updating, Derived Execution and TIL Inference rules. For each disjunct φ_i, $i = 1, \ldots, k$ of φ extend the tableau by creating a branch i with a new terminal node N_i and label N_i with φ_i.

3. For each branch i, $i = 1, \ldots, k$ at N, define an "extension" of the path from the initial node to N that ends at the new node N_i. Remove N from the set of terminal nodes.

4. Repeat steps 2 and 3 until every path of the tableau is "closed." A path and all its extensions are "closed" if one of its nodes is labeled with F (false) or if one node is labeled with a TIL formula X while another one is labeled with $\neg X$. The process terminates with no conclusions about $\Box p$ if a formula of the form $\mathrm{O}(-m')s$ is deduced, where s is a state in H and $m' > m$ (indicating inadequate assumptions about the initial conditions).

The choice of m in Step 1 plays an important role in closing a tableau. In practice, we initially choose $m = \infty$ and we attempt to close the tableau. Let m' be the largest value such that $\mathrm{O}(-m')s$ is deduced in a closed tableau. Then, to complete the proof, we need to show by a simulation process that, starting from a desired initial set of markings, Q cannot be true for $m'+1$ moments, indicating that in Step 1 any finite m such that $m \geq m'$ is adequate. If the safety property is indeed true, this is usually relatively simple to verify, requiring only a fragment of steps necessary to prove safety by model checking (see, e.g., proof of Theorem 1 below). On the other hand, if the safety property is not true for H, then there exists some execution of length at most m' that will cause Q to be true.

As in traditional theorem proving, the choice of rules to be applied in Step 2 of our theorem proving method is not deterministic and the process is not guaranteed to terminate. Experience and heuristic guidelines are helpful in reducing the lengths of proofs. However, the method can be shown to be *complete* in the following sense: if a safety property is true about an HMS machine, there exists a tableau-based proof for it. This follows from the completeness of the transformational proof method of [FG89] since all the necessary transformations are derivable from the rules presented in this paper.

We now introduce a generic safety property that is often useful in closing the branches of a tableau in our theorem proving method. It is a variation of S-invariance for Petri nets [Re85].

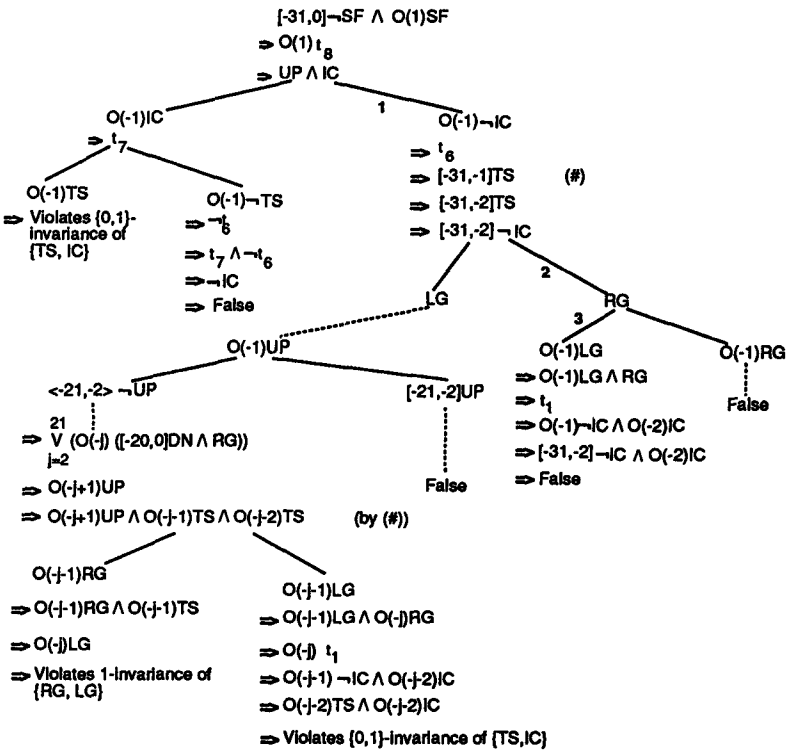

Figure 3.1. Partial Tableau for Theorem 1 (Safety of Railroad Crossing)

Definition 4 (K-**invariance**) Let K be a set of integers, S be a subset of states in an HMS machine H, and $\#T(S)$ be the number of states in S that are true. Then, S is said to be "K-invariant" if the following holds: If at anytime $\#T(S) \in K$, then $\Box(\#T(S) \in K)$. When K is a singleton set $\{n\}$, we say a set S is "n-invariant" if it is $\{n\}$-invariant.

We demonstrate in Theorem 1 the use of our theorem proving method by verifying the safety of the railroad crossing example of Figure 2.1. The proof of the theorem uses the following lemma that can be derived by a straightforward application of our theorem proving method.

Lemma 1 *$\{TS, IC\}$ is $\{0,1\}$-invariant and both $\{RG, LG\}$ and $\{UP, DN\}$ are 1-invariant.*

It follows that in the railroad example of Figure 2.1 if initially $RG = UP = T$ and $TS = IC = DN = LG = F$, then at most one of the two states TS and IC can be true and RG (UP) is true if and only if LG (DN) is false.

Theorem 1 *The HMS specification of the railroad crossing example is safe.*

Sketch of Proof: A partial tableau for the proof is shown in Figure 3.1, where $m = -31$ indicating

that the extended state SF is assumed to have been false for at least 31 moments. This can easily be verified by running the machine for 31 moments starting from any initial marking where $TS = IC = F$. Solid lines represent branches in the tableau, while broken lines indicate that a number of intermediate steps, possibly involving additional branches, have been skipped. As an illustration, we walk through branches 1, 2, and 3 of the tableau. We first create the initial node following step 1 of our method with the label $\neg SF \wedge \mathrm{O}(1)SF$. By Derived Execution rules DE-2 and DE-3 we deduce $UP \wedge IC$. The tautology $\mathrm{O}(-1)IC \vee \mathrm{O}(-1)\neg IC$ gives rise to the first pair of branches in the tableau. Along branch 1, we conclude that t_6 fires at $t = 0$ by Derived Execution rule DE-2 since $IC \wedge \mathrm{O}(-1)\neg IC$ holds. By Derived Execution rule DE-3 the control of t_6 was satisfied at $t = -1$, i.e., $[-31, -1]TS$ holds. Next, using the {0,1}-invariance of $\{TS, IC\}$ (Lemma 1), we conclude $[-31, -2]\neg IC$. At this point, we create two branches, with RG being true along branch 2. Along branch 3, we conclude that t_1 fires at $t = 0$ by Transition Firing rule TF-1 since $\mathrm{O}(-1)LG \wedge RG$ holds. By Derived Execution rule DE-3, the control of t_1 was satisfied at $t = -1$, i.e., $\mathrm{O}(-1)\neg IC \wedge \mathrm{O}(-2)IC$ holds. However, $\mathrm{O}(-1)\neg IC \wedge \mathrm{O}(-2)IC$ contradicts $[-31, -2]\neg IC$. This closes branch 3 of the tableau. Other branches can be closed in a similar manner. As indicated in Figure 3.1, the closing of certain branches involves the use of *parametric* TIL formulae containing the parameter j, for $j = 2, \ldots, 21$. In such cases, the refutation for an entire range of values of the parameters is accomplished in a single step.

The tableau-based theorem proving method presented here is particularly useful in proving safety properties of HMS machines with many *nondeterministic* transitions. Like the transformational approach of [FG89] and in contrast to traditional model checking approaches, it has the important advantage that the length of a proof does not directly depend on the size of the machine. The length of a proof depends only on the safety property being verified and the structure of the relevant portions of the associated machine.

4 Verifying Properties with Parametric Computation Graphs

In [GI91, Ga91] the basic ideas of a model checking approach for HMS machines were presented in which the set of all possible behaviors of a machine is represented in terms of a parametric "computation graph" and desirable behavior is defined in a branching time extension of TIL, called BTIL. In BTIL, the two additional operators $\underline{\exists}$ (there exists an execution) and $\underline{\forall}$ (for all executions) are used to characterize nondeterministic behavior for the past, the present and the future. Verification of a BTIL property is then achieved by the traversal of the associated computation graph.

In this section, we present some preliminary results on an approach to partitioning the computation graph of an HMS machine in terms of "interacting" parametric computation graphs that can be collectively much simpler than the computation graph of the entire machine. We limit our discussion to cases where the HMS machine consists of a "composition" [GI91, Ga91] of "cyclic" HMS machines, in which there is a partial order on controls between separate machines. As in the case of the "submachines" Signal, Gate, and Train in Figure 2.1, an HMS machine is cyclic if there is a circular path of transitions connecting all the states. In case of Train, the two infinite resources (vertical bars) are identified as a single state.

In more specific terms the HMS machines considered in this section can be characterized as follows: Given a machine $H = (S, \Gamma_D, \Gamma_N)$, S can be partitioned into disjoint subsets $S_1, \ldots, S_k$, such that for each $\gamma \in \Gamma_D \cup \Gamma_N$, there exists an $i \in \{1, \ldots, k\}$ with $\mathrm{PRIMS}(\gamma) \cup \mathrm{CNSQS}(\gamma) \in S_i$ and if s is a literal in $\mathrm{CNTRL}(\gamma)$ then $s \in S_1 \cup \ldots \cup S_i$. Each partition S_i gives rise to an HMS cyclic submachine H_i such that

the behavior of H_1 is independent of all the other submachines and the behavior of H_i for $i > 1$ depends only on $H_1, \ldots, H_{i-1}$. In our example of the railroad crossing of Figure 2.1, the HMS machine can be partitioned into the three submachines H_1 = Train, H_2 = Signal and H_3 = Gate. Train is independent of Signal and Gate, Signal depends only on Train, and Gate depends on Signal.

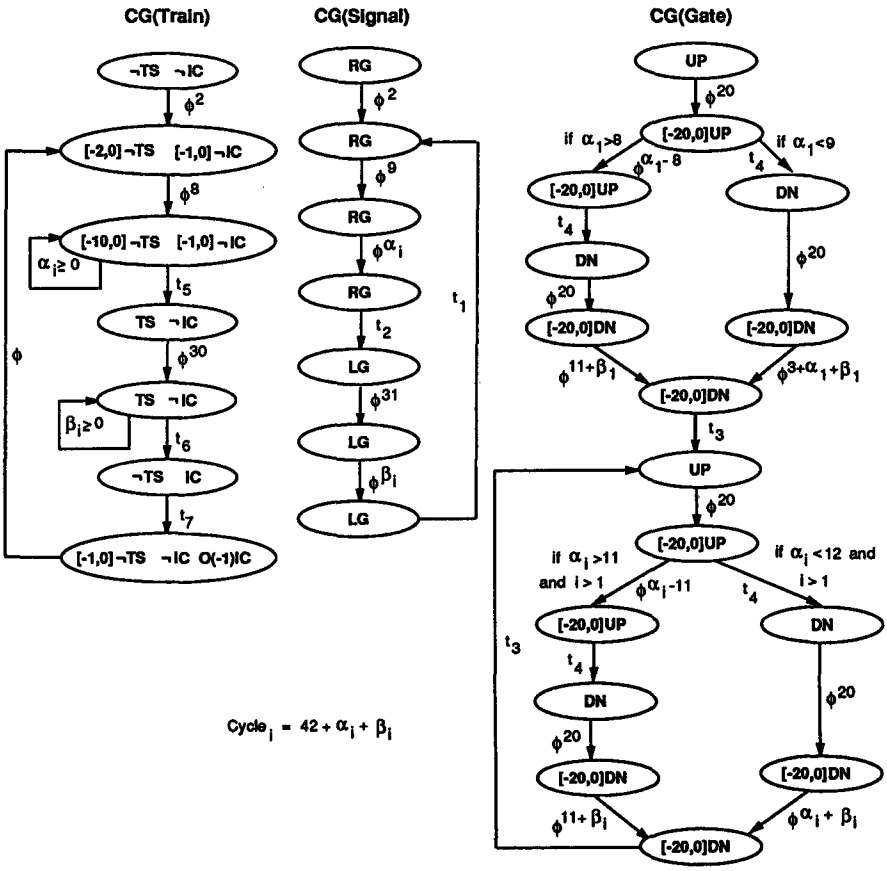

Figure 4.1. Interacting Parametric Computation Graphs for Railroad Crossing

The form of computation graph considered here consists of (1) nodes labeled with TIL formulae recording state marking histories, and (2) arcs between nodes labeled with sets of transitions or terms of the form ϕ^i. Here ϕ represents a "wait" or "no action" and i is either an integer constant or an integer variable with the interpretation that ϕ^i represents a parametric wait of i moments. Given a computation graph for an HMS machine H, assume that a node N_i is labeled with the set $\mu(N_i)$ of TIL formulae and an arc α_{ij} from node N_i to node N_j is labeled with $\gamma(\alpha_{ij})$. Then, if all the formulae $\mu(N_i)$ are true, the execution of $\gamma(\alpha_{ij})$ will make all the formulae in $\mu(N_j)$ true.

We note the following characteristics of our computation graphs: (1) nodes contain state histories defined in TIL for multiple states, (2) arcs record simultaneous firings of transitions or parametric waits, and (3) for each state a "maximum relevant history" [GI91, Ga91] can be defined that limits the length of history for the state that needs to be recorded. The key notion to be exploited in this section is the establishment of *interactions* among the computation graphs of submachines of an HMS machine in terms of the parametric waits. As a result, a significant reduction in the size of a computation graph can be achieved.

Figure 4.1 illustrates the concept of interacting parametric computation graphs for the railroad crossing HMS machine of Figure 2.1. We first construct the computation graph CG(Train) for the submachine Train which, as indicated earlier, does not depend on Signal or Gate. We begin with the assumption that TS and IC are both initially false so that after a wait of 10 moments $[-10, 0]\neg TS$ will be true. At this time, since t_5 is nondeterministic, it will fire after a parametric wait of α_1 moments for $\alpha_1 \geq 0$. The parameter α_i designates a similar wait during the ith traversal of CG(Train). Another arbitrary wait $\beta_i \geq 0$ occurs before the firing of the nondeterminisitic transition t_6. We then construct in sequence the computation graphs CG(Signal) and CG(Gate) for the submachines Signal and Gate, respectively. All three computation graphs turn out to have a cycle length of $42 + \alpha_i + \beta_i$ for the ith iteration. Note that the parametric waits α_i and β_i in CG(Train) determine uniquely the lengths of certain waits for certain branches of CG(Signal) and CG(Gate). In addition, for CG(Gate) alternative branches are chosen depending on the values of α_i and β_i.

The three relatively simple interacting computation graphs in Figure 4.1 permit a complete analysis of the behavior of our railroad crossing example. In contrast, a combined computation graph would have been quite complex, potentially involving the "cross product" of the individual computation graphs. As a result, the analysis of the behavior of the railroad crossing using the combined computation graph would have been much more difficult.

The proof of the safety property $\Box\neg(IC \wedge UP)$ of the railroad crossing can now be demonstrated *analytically* from the computation graphs of Figure 4.1 since $\mathbf{O}(k)IC$ for $k = 42i + \Sigma(\alpha_i + \beta_i), i \geq 1$, from CG(Train) and $\mathbf{O}(42i + \Sigma(\alpha_i + \beta_i)), i \geq 1$, from CG(Gate). Thus, whenever IC is true the gate is down (DN) and therefore UP is false by the 1-invariance of $\{DN, UP\}$.

5 Summary and Conclusions

The HMS machine concept provides a rich and compact formalism for the specification of systems with complex real-time constraints. Tableau-based theorem proving and interacting parametric computation graphs offer two complementary methods for verifying properties of HMS machine specifications. The former approach introduces the use of the interval-based temporal logic TIL and execution semantics of HMS machines into the theorem proving process. The latter approach simplifies the representation of computation graphs significantly. Current research is directed towards the generalization of the specification and verification techniques into an algebraic framework and the development of heuristic guidelines for the application of the verification methods discussed in this paper.

References

[Ab87] Abadi, M., "Temporal-logic theorem proving," Dissertation, Stanford Univ., 1987.

[AS89] Alpern, B., and F.B. Schneider, "Verifying temporal properties without temporal logic," *ACM Transactions on Programming Languages and Systems,* Vol. 11, No. 1, January 1989, pp. 147-167.

[DM88] Da-Hai, L., and T.S.E. Maibaum, "Developing a high level specification formalism," in *Formal Methods: Theory and Practice,* P.N. Scharbach (Ed.), CRC Press, Boca Baton, FL, 1988.

[Fi90] Fitting, M., *First-Order Logic and Automated Theorem Proving*, Springer-Verlag, 1990.

[FG89] Franklin, M.K., and A. Gabrielian, "A transformational method for verifying safety properties in real-time systems," *Proc. 10th Real-Time Systems Symposium,* Santa Monica, CA, December 5-7, 1989, pp. 112-123.

[Ga91] Gabrielian, A., "HMS machines: a unified framework for specification, verification and reasoning for real-time systems," *Foundations of Real-Time Computing: Formal Specifications and Methods*, A. van Tilborg (Ed.), Kluwer Academic Publishers, Norwell, Mass., 1991, to appear.

[GF88] Gabrielian, A., and M. K. Franklin, "State-based specification of complex real-time systems," *Proc. IEEE Real-Time Systems Symposium*, 1988, pp. 2-11.

[GF90] Gabrielian, A., and M.K. Franklin, "Multi-level specification and verification of real-time software," *12th Int. Conf. Software Engineering*, March 26-30, 1990, Nice, France, pp. 52-62. Revised version in *Communications of the ACM*, May 1991, pp. 50-60.

[GI91] Gabrielian, A., and R. Iyer, "Integrating automata and temporal logic: a framework for specification of real-time systems and software," *The Unified Computation Laboratory*, Inst. of Math. and its Applications, Stirling, Scotland, Oxford University Press, 1991.

[GM89] Ghezzi, C., D. Madrioli, et al., "A general way to put time in Petri nets," *Proc. Fifth Int. Workshop on Software Specif. and Design*, Pittsburgh, PA, May 1989, pp. 60-67.

[HMP91] Henzinger, T.A., Z. Manna, and A. Pneuli, "Temporal proof methodologies for real-time systems," *Proc. 18th Annual ACM Symposium on Principles of Programming Languages*, 1991.

[Me88] Melliar-Smith, P.M., "A graphical representation of interval logic," in *Proc. Concurrency 88*, LNCS 335, Springer-Verlag, 1988, pp. 106-120.

[MP89] Manna, Z., and Pnueli, A., "The anchored version of the temporal framework," *Proc. Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354, Springer-Verlag, 1989, pp. 201-284.

[PH88] Pnueli, A., and E. Harel, "Applications of temporal logic to the specification of real-time systems (extended abstract)," *Proc. Symp. Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 331, Springer-Verlag, 1988, pp. 84-98.

[Re85] Reisig, W., *Petri Nets*, Springer-Verlag, 1985.

[SW89] Stirling, C., and D. Walker, "CCS, liveness, and local model checking in the linear time mu-calculus," *Proc. Automatic Verification Methods for Finite State Systems*, Grenoble, 1989, LNCS 407, Springer-Verlag, 1990, pp. 166-178.