

Modechart: A Specification Language for Real-Time Systems

Farnam Jahanian, *Member, IEEE*, and Aloysius K. Mok

Abstract—In this paper, we present a specification language for real-time systems called Modechart. The semantics of Modechart is given in terms of RTL (Real Time Logic [8]) that is especially amenable to reasoning about the absolute (real-time clock) timing of events. The semantics of Modechart has an important property that the translation of a Modechart specification into RTL formulas will result in a hierarchical organization of the resulting RTL assertions. This gives us significant leverage in reasoning about properties of a system by allowing us to filter out assertions that concern lower levels of abstraction. Some results about desirable properties of Modechart specifications will be given. A graphical implementation of Modechart has been completed.

Index Terms—Real-time systems, specification, rapid prototyping, timing constraints, RTL.

I. INTRODUCTION

AS SOFTWARE CONTROL of safety-critical functions in embedded systems becomes more common, a means for precise and concise specification of their behavior becomes increasingly important. The specification problem for these real-time applications is more complex since the absolute timing behavior (the timing of events measured by a real-time clock) and not only the functional behavior of a system is important. In addition to being a valuable aid to the maintenance of complex real-time systems, a precise and concise specification language can also be used to provide answers to queries about system behavior without a full implementation (rapid prototyping). In this paper, we shall present a specification language for real-time systems called Modechart. The semantics of Modechart is defined in terms of RTL [8] which is a logic especially amenable to reasoning about the absolute timing of events. A graphical implementation of Modechart has also been completed.

Substantial work has been done by a number of researchers in the specification and prototyping of complex real-time systems, e.g., [13], [9], [1], [2]. A notable experiment in

the application of specification methods is the Software Cost Reduction project at the Naval Research Laboratory where a systematic methodology was applied to document the software requirements of the A-7E aircraft (see [5], [12]). More recently, an important advance was made by Harel who proposed a visual language called Statechart [3] which is an extension of conventional state-transition diagrams. Harel's language provides a succinct way to represent large systems since it supports hierarchical and modular decomposition of state machines. A formal semantics of Statechart in terms of temporal logic and a general theory of reactive systems are also being investigated by its inventors [4].

Unlike previous work, our work emphasizes the specification of absolute timing properties of systems. In Modechart, we make use of the concept of modes from the work of Parnas *et al.* at the Naval Research Laboratory. Modes can be thought of as partitions of the state space of a system and are an effective way for modular specification of large state machines. Modechart also borrows from Statechart the very appealing compact representation of large state machines. Our main contribution is in providing a semantics which explicitly deals with the absolute timing of events and avoids some of the potential semantic anomalies of Statechart. More importantly, the translation of a Modechart specification into RTL formulas will result in a hierarchical organization of the resulting RTL assertions. This gives us significant leverage in reasoning about properties of a system by allowing us to filter out assertions that concern lower levels of abstraction. The ability to avoid considering all the assertions defining the behavior of a large system is a prerequisite to practical applications of verification technology.

Modechart has been developed as a graphical specification tool in SARTOR (Software Automation for Real-Time Operations), a design environment for hard-real-time software currently under development at the University of Texas at Austin. The goal of SARTOR is to mechanize the analysis and synthesis of real-time software from systems specification. An implementation of Modechart serves as a front-end for SARTOR which provides a suit of tools to analyze a specification for satisfaction of safety requirements. A design can be synthesized (rapid prototyping) if it is determined that there are sufficient resources to do so. A review of SARTOR can be found in [10].

The rest of this paper is organized as follows. Section II gives the syntactic description of modes. Section III gives a brief overview of RTL (Real Time Logic). Section IV gives the RTL semantics of mode transitions. Section V gives the

Manuscript received February 1989; revised July 1989. This paper is a substantially revised version of a paper presented at the 21st Hawaiian Conference on System Sciences, January 1988. It contains major changes to the semantics of the specification language reported herein and complete proofs of results that are not in the conference paper [6]. This work was supported in part by a research grant from the Office of Naval Research under ONR Contract N00014-85-K-0117, by a University Research Initiative Grant under ONR Contract N00014-86-K-0763, and also by a DoD-University Research Instrumentation Program Grant under ONR Contract N00014-86-G-0199.

F. Jahanian is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

A. K. Mok is with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA.

IEEE Log Number 9406405.

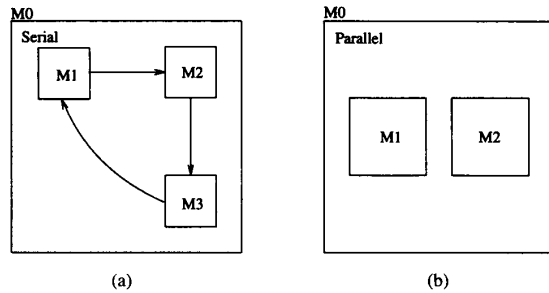


Fig. 1. Serial and parallel modes.

RTL semantics of actions in modes. Section VI deals with mode transitions that span more than one level of the mode hierarchy. Section VII presents some results about desirable properties of mode specifications using Modechart. Section VIII is the conclusion.

II. HIERARCHICAL DECOMPOSITION: SERIAL AND PARALLEL MODES

Intuitively, modes may be viewed as control information that impose structure on the operation of a system. Modes are arranged hierarchically. Furthermore, modes which are peers in this hierarchy can be related in one of two ways: in *series* or in *parallel*.

The serial relationship among several modes indicates that the system operates in, at most, one of these modes at any time. For instance, Fig. 1(a) illustrates a two-level hierarchy where M_0 is the parent mode and M_1 , M_2 , and M_3 are embedded in M_0 . If the system is in mode M_0 , then it must also be in either one of M_1 , M_2 , or M_3 . M_0 is said to be a serial mode, and M_1 , M_2 , and M_3 are said to be in series. A transition allows the system to go from one mode to another. Since a specification may require a transition to go across different levels of the hierarchy, transitions may not always be between modes at the same level in a hierarchy. Also, entry into a serial mode M requires the designation of one of the child modes of M to be the default mode the system will be in. If the transition arrow crosses into the serial mode, then the child mode that the transition arrow points at is entered. Otherwise, a child mode which is labeled as the initial mode is entered. As an example, mode M_0 of Fig. 1(a) could model the flight path monitoring mechanism for an aircraft. Modes M_1 , M_2 , and M_3 represent Monitor, Signal, and Correction modes, respectively. When the plane is detected to be off course, the monitor mechanism moves from the Monitor mode into the Signal mode. In the signal mode, the pilot is warned that the plane is off course. The pilot corrects the flight of the aircraft, taking the system into the Correction mode. After making the correction, the pilot informs the system (e.g., by pressing a button), returning the system to the monitor mode. Monitor, Signal, and Correction modes are in series because the monitor mechanism can only be in one of the three at a time.

The parallel relationship among several modes indicates that a system operates in all of these modes simultaneously. For instance, Fig. 1(b) illustrates a mode M_0 with two embedded modes, M_1 and M_2 . If the system is in mode M_0 , then it

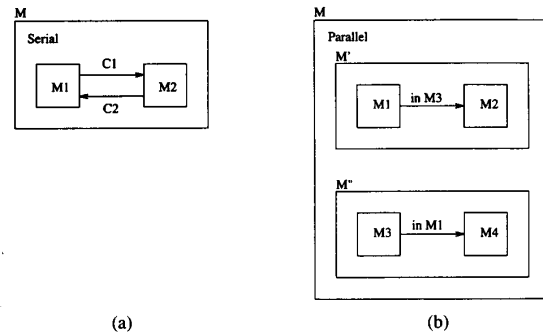


Fig. 2. Semantic problems of modes.

also must be in both M_1 and M_2 . (M_0 is said to be a parallel mode, and M_1 and M_2 are said to be in parallel.) Transitions between modes in parallel are not allowed. Entry into a parallel mode requires entry into all of its immediate child modes. Hence, no member of a set of modes in parallel should be specified as the initial mode. Similarly, a transition out of one mode requires exit out of all the modes in parallel to it. Returning to the aircraft example, mode M_0 of Fig. 1(b) could model the plane's instrument display system. Modes M_1 and M_2 might represent the Airspeed and Altitude display modes, respectively. The airspeed and altitude displays are updated independently of each other and so are in parallel under mode M_0 .

The preceding informal description of serial and parallel modes might give a deceptively simple view of Modechart. In fact, there are many ambiguities which must be resolved in providing a formal semantics for mode transitions. For example, if transition between two modes is instantaneous, then a cyclic sequence of transitions occurring at the same instant of time will lead to an anomaly. Consider the serial mode M in Fig. 2(a). If the system is in mode M_1 and condition C_1 is true at time t , then the corresponding transition is taken and the system exits from mode M_1 and enters mode M_2 at that instant of time. However, if condition C_2 is also true at time t , then the system can exit M_2 and enter M_1 at the same instant of time. Hence, the modes in the cycle would be entered and exited an infinite number of times at an instant in time. To avoid this anomaly, one might require mode transitions to take finite time. The problem is that system behavior will then be ambiguous when an event occurs in between modes and the response to that event is supposed to depend on which mode the system is in, and any bound put on the duration of a transition is likely to be ad hoc at the requirements specification stage. We shall model mode transitions as being instantaneous. In Section VII, we shall present a condition for preventing anomalies due to cyclic transitions.

Ambiguities can also arise due to the way one may model concurrency. For example, consider the parallel mode M in Fig. 2(b). The system is in modes M_1 and M_3 initially. If the behavior of a system specified in Modechart is viewed as a sequence of nonoverlapping events (i.e., following the interleaving model of concurrency), then the system in Fig. 2(b)

cannot end up in both modes $M2$ and $M4$ even though one may expect simultaneous transitions from $M1$ to $M2$ and from $M3$ to $M4$ to take the system into modes $M2$ and $M4$. This is due to the fact that two simultaneous events are regarded as equivalent to the occurrence of the two events in either order.¹ However, if concurrency in a system is modeled by a partial ordering among events (i.e., two events are concurrent if neither precedes the other one), then the two transitions in Fig. 2(b) can be taken simultaneously. Hence, the system can be in both modes, $M2$ and $M4$. In defining the semantics of Modechart, we shall model concurrency as a partial ordering on event occurrences.

Before providing a formal semantics for modes and transitions, we first give a syntactic description of well-formed modes in Section II-A below.

A. Well-Formed Modes

A mode with no internal modes is the primitive with which compound (serial and parallel) modes are built. A compound mode is constructed from other modes in three ways:

- constructing a serial mode from existing modes,
- constructing a parallel mode from existing modes, and
- connecting two modes via a mode transition.

Informally, a well-formed mode is one which either contains no other mode, or the modes contained in it are nested properly and all the transitions in that mode do not connect parallel modes. Furthermore, a well-formed serial mode may have at most one initial mode and a well-formed parallel mode may not have an initial mode. It should be noted that well-formedness is a syntactical property which by itself does not guarantee that all initial modes are properly designated. A well-formed mode may still be ambiguous in the sense that its initial mode designations may be underspecified. We shall return to this point shortly.

Definition (Containment Partial Order): Let \mathbf{M} be a set of modes $\{M_1, M_2, \dots, M_m\}$. The binary relation containment, denoted by \subset , defines a partial ordering of the elements of the set \mathbf{M} . Graphically, $M_i \subset M_j$ iff the box representing mode M_i is contained entirely inside the box representing mode M_j .

Definition:

- A mode M_i is a primitive mode iff $\forall j M_j \not\subset M_i$.
- A mode M_i is a compound mode iff $\exists j M_j \subset M_i$.
- A mode M_i is a root mode iff $\forall j M_i \not\subset M_j$.
- A mode M_i is an immediate child of a mode M_j (and conversely M_j is the parent of M_i) iff

$$M_i \subset M_j \wedge (\forall k j \neq k \wedge M_i \subset M_k \rightarrow M_j \subset M_k)$$

Graphically, a primitive mode contains no other mode; a compound mode contains at least one other mode; a root mode is one which is not contained in any other mode. Furthermore, the box representing a child mode is immediately surrounded by the box representing the parent. We now define a well-formed mode.

¹In this example, it is possible to allow the system to end up in both $M2$ and $M4$ in the interleaving model by stipulating additional causality rules governing events that occur "at the same time." The semantics of this approach is rather complicated [11].

Definition: Well-formed modes are defined recursively as follows:

- 1) A primitive mode is well-formed.
- 2) Suppose M_1, M_2, \dots, M_m are well-formed root modes, and at most one mode M_i is labeled as the initial mode, then adding a serial mode M consisting of only these modes as its immediate children makes M well-formed.
- 3) Suppose M_1, M_2, \dots, M_m are well-formed root modes, and none is labeled as an initial mode, then adding a parallel mode M consisting of only these modes as its immediate children makes M well-formed.
- 4) Let M be a well-formed mode, and $M_i \subset M$ and $M_j \subset M$. Adding a transition from M_i to M_j preserves M as a well-formed mode if the least upper bound of M_i and M_j with respect to the \subset relation is a serial mode.²

Informally, the above condition requires that the first mode containing both M_i and M_j to be a serial mode. Observe that the least upper bound of the two modes is either M or a mode inside M . Also notice that it is not necessary for the children of a serial mode to be connected by any transition, since they can be entered by distinct transitions from outside the parent. None of the children in this case needs to be designated an initial mode.

As mentioned earlier, a well-formed mode can still be ambiguous if designation of the initial modes is underspecified. As an example, consider the case where a transition enters a mode in parallel to a serial mode M , but none of the immediate children of M is labeled as the initial mode. We first present a definition which is useful in formulating a condition for ensuring proper designation of the initial modes.

Definition: A mode M is a landing mode if any one of the following conditions holds:

- if M is the root mode,
- if a transition ends at mode M ,
- if its parent M' is a serial mode, M' is a landing mode, and M is the initial mode,
- if its parent M' is a parallel mode, and M' is a landing mode
- if its parent M' is a parallel mode, and there is a transition ending at or crossing into a mode in parallel with M .

In a system whose root mode is well-formed, we define the UDIM condition (for Unambiguous Designation of Initial Modes) as follows:

For each serial mode which is a landing mode, there is exactly one initial mode.

To illustrate the above condition, assume that the system enters a serial mode M . If M is not a landing mode, the only way to enter M is to have the corresponding transition arrow cross into it. In this case, the transition identifies a unique child of M to be entered, i.e., there is no ambiguity. However, if M

²The least upper bound of M_i and M_j is a mode M' such that $M_i \subset M' \wedge M_j \subset M' \wedge$ (for each mode M'' immediate child of M' , $M_i \not\subset M'' \vee M_j \not\subset M''$)

is a landing mode, the UDIM condition requires M to have an initial mode, thus ensuring against ambiguity.

Before presenting the formal semantics of Modechart, we provide an overview of Real Time Logic (RTL) in the next section. (For a more detailed discussion, refer to the report in [8].)

III. REAL TIME LOGIC

Real Time Logic is a first order predicate logic invented primarily for reasoning about timing properties of real-time systems. It provides a uniform way for the specification of both relative and absolute timing of events. In RTL, we reason about individual occurrences of events where an event occurrence marks a point in time which is of significance to the behavior of the system. Unlike some other logics, there is an important distinction between an action and an event in RTL. An action is an operation which requires a nonzero but bounded amount of system resources. However, events serve only as temporal markers. An occurrence of an event defines a time value, its time of occurrence, and imposes no requirement on system resources. Every action is characterized by two events: one denoting its initiation and the other denoting its completion. In [8], we distinguish between four classes of events: (1) external event, e.g., an operator pushes the red button, (2) start event which marks the initiation of an action, (3) stop event which marks the completion of an action, and (4) state variable transition event which marks a change in a certain attribute of the system, e.g., airplane becomes airborne. In the next section, we shall introduce new classes of events to capture the formal semantics of modes.

Events have unique names. Any name in capital letters prefixed by the special letter Ω (Omega) is used to denote an external event. For instance, $\Omega\text{BUTTON1}$ denotes the external event associated with pressing button #1. Execution of an action is represented by the corresponding start and stop events. We use the notation $\uparrow A$ to represent the event marking the initiation of action A , and $\downarrow A$ to denote the event marking the completion of action A . For instance, $\uparrow\text{SAMPLE}$ and $\downarrow\text{SAMPLE}$ represent the events corresponding to the start and the stop of action SAMPLE , respectively.

A state variable may describe a physical aspect or certain property of a system, e.g., an autopilot switch which is either ON or OFF. The execution of an action may cause the value of one or more state variables to change. A state attribute, S is a predicate which asserts that a state variable takes on a certain value in its domain. For example, S may denote the predicate: the autopilot is ON. The corresponding state variable transition events, represented syntactically by $(S := T)$ and $(S := F)$, denote respectively the events that mark the turning on and off of the autopilot switch. Whenever it is unambiguous, we shall use state variable and state attribute interchangeably.

An *occurrence function*, denoted by the character “@,” is introduced to capture the notion of real time by assigning time values to event occurrences. Formally, the “@” function is a mapping from the space (E, Z^+) to N such that E , Z^+ , and N are respectively the set of event constants, the set of positive integers, and the set of natural numbers.

Definition:

$$\@(e, i) \equiv \text{time of the } i\text{th occurrence of event } e$$

where i is an integer.

For instance, $\@(\uparrow\text{SAMPLE}, 1)$ denotes the time of the first occurrence of the event marking the start of action SAMPLE .

The notion of an occurrence function is central to RTL. In particular, timing requirements imposed by a system specification and a property under investigation are restrictions on the “@” function. A timing property (an RTL assertion) of a system can be established by showing that there is no occurrence function which is consistent with the system specification and the negation of that timing property. The occurrence function is in general not a total function since some sporadic events may occur only a finite number of times or even not at all.

RTL predicates are formed from the arithmetical relations ($=, <, \leq, >, \geq$) and algebraic expressions allowing integer constants, variables, addition, subtraction, multiplication by constants, and the occurrence function. RTL formulas are constructed using the equality/inequality predicates, universal and existential quantifiers, and the first-order logical connectives ($\neg, \wedge, \vee, \rightarrow$).³ The following is an example of an RTL formula.⁴

Example 1:

$$\forall i \@(\Omega\text{BUTTON1}, i) < \@(\uparrow\text{SAMPLE}, i) \wedge \\ \@(\downarrow\text{SAMPLE}, i) \leq \@(\Omega\text{BUTTON1}, i) + 20.$$

The preceding RTL formula is interpreted as: action SAMPLE is executed only after push-button #1 is pressed, and every execution of SAMPLE must complete within 20 time units of the corresponding pressing of button #1.

Recall that a transition event marks a change in the value of a state variable. RTL uses a notational device, called a state predicate for asserting the truth value of a state attribute (e.g., the autopilot switch of an airplane being in the ON position) during an interval. We omit giving the full syntax for state predicates in this paper. It suffices to know that the state predicate $S(x, y)$ asserts that S becomes true before or at time x and it remains true at least until time y . When both arguments of a state predicate are the same, a useful notation is obtained: $S(x, x)$ asserts that the state attribute S holds true in an interval surrounding time x , and the state predicate $\bar{S}(t, t)$ denotes that the state attribute S is false during an interval around time t . The reader is assured that a state predicate is simply a shorthand for algebraic relations involving the time of occurrence of the relevant state transition events.

³The standard precedence order is assumed for these connectives. \neg has the highest precedence, \wedge and \vee the next highest precedence, and \rightarrow the lowest precedence.

⁴The RTL syntax presented here is the functional form described in [8]. To express event instances that may not happen, we use an occurrence relation as described in [7]. In particular, the “@” function here is used in place of an occurrence relation Θ on the set $E \times Z^+ \times N$, where $\Theta(e, i, t)$ denotes that the i th occurrence of event e happens at time t . We use the “@” notation in this paper as an abbreviated form of the Θ relation. For example, the formula $\forall i \exists j \@(\Omega(e_1, i) + 10 \leq \@(e_2, j))$ corresponds to the formula

$$\forall i \forall t \Theta(e_1, i, t) \rightarrow \exists j \exists t' \Theta(e_2, j, t') \wedge t + 10 \leq t'.$$

IV. SPECIFYING THE SEMANTICS OF MODES AND TRANSITIONS IN RTL

In this section, we introduce additional classes of events to help capture the formal semantics of modes in RTL. After presenting the syntax for the conditions on mode transitions, we shall illustrate how mode transitions are specified in RTL.

A. Comparison of Modes and State Variables

Since modes represent control information about the behavior of a system, it may seem natural to use state variables to model modes in RTL. In fact, we use a notation similar to state variables to represent modes in RTL. The semantic distinctions between modes and state variables will be discussed shortly.

To capture the formal semantics of modes in RTL, we introduce two events denoting mode entry and exit. For a mode M , the event $(M := T)$ denotes entering M and the event $(M := F)$ represents exiting the mode. (To distinguish these events from transition events for state variables, we shall refer to them as the mode entry and the mode exit events.) Staying in a mode during an interval is described through the use of a notation similar to state predicates. The notation, referred to as the mode predicate, also has nine variations for each mode M :

$$M[x, y], M[x, y), M[x, y >, M(x, y), M(x, y), \\ M(x, y >, M < x, y >, M < x, y), \text{ and } M < x, y).$$

Each mode predicate qualifies the timing of two events, one denoting the entry to a mode and the other denoting the exit from a mode. The two arguments, x and y , in a mode predicate are used in conjunction with the symbols “[,” “],” “(,” “),” “<,” and “>” to denote an interval over which the system remains in a mode.

The convention we use for arriving at this syntax requires some explanation. Suppose $(M := T)$ and $(M := F)$ are the mode entry and the mode exit events for a mode M , respectively. Informally,

- “ x ” denotes that $(M := T)$ occurs at time x ,
- “ $(x$ ” denotes that $(M := T)$ occurs before or at time x ,
- “ $< x$ ” denotes that $(M := T)$ occurs before time x ,
- “ $y]$ ” denotes that $(M := F)$ occurs at time y ,
- “ $y)$ ” denotes that $(M := F)$ does not occur before time y ,
- “ $y >$ ” denotes that $(M := F)$ does not occur before or at time y .

For example, the mode predicate $M[x, y]$ indicates that the system enters mode M at time x and exits mode M at time y . Precisely, the mode entry event $(M := T)$ occurs at time x , the system remains in mode M during the interval between x and y , and the mode exit event $(M := F)$ occurs at time y . For another example, $M[x, y)$ indicates that the system enters mode M at time x and it remains in this mode at least until time y . We define the notation $M(x, x)$ to denote that the system is in mode M at time x . Due to the above definition, when a transition from a mode M_1 to another mode M_2 is taken, the system is in both modes at that instant of time. However, as it will be shown in Section VII, the semantics of Modechart prevents the system to be in two in series over a finite time interval. We stress that mode predicates are formally

defined in terms of the corresponding mode entry and exit events.

Despite the similarities in their notations, there are important semantic distinctions between modes and state variables. Intuitively, a state variable represents information about data whereas a mode represents some control information about the system. The value of a state variable is changed explicitly by completing the execution of an action which takes nonzero units of time to perform. Consequently, a state attribute S cannot become true and then become false at the same instant of time, i.e., the two events $(S := T)$ and $(S := F)$ cannot happen at the same instant of time. A mode entry or exit is implicit in that it does not require the execution of an action; a mode transition is taken when a certain condition is satisfied. Hence, a mode M can be entered and exited simultaneously if the condition for exit is also satisfied, i.e., we allow the two events $(M := T)$ and $(M := F)$ to happen at the same instant of time.

B. Transitions

Transition between two modes represents a change in the control information of the system. A mode transition is an instantaneous event which takes zero time units. To capture the formal semantics of a transition, we introduce the mode transition event to denote the occurrence of a transition from one mode to another. Specifically, we use the notation $(M_i - M_j)$ to indicate the mode transition event from mode M_i to mode M_j .

We associate a condition with each transition. The condition for a mode transition is of the form

$$c_1 \vee c_2 \vee \cdots \vee c_n$$

where each disjunct c_k is either (1) a triggering condition for taking the transition, or (2) a lower/upper bound restriction on when the transition may be taken.

1) *Triggering Condition*: A disjunct c_k denoting a triggering condition is of the form

$$p_1 \wedge p_2 \wedge \cdots \wedge p_m$$

where the p_j 's specify a condition for taking the transition depending on the occurrence of an event and/or the truth values of certain predicates. In particular, each subcondition p_j is of one of the three forms shown below. (E denotes an event, S is a state variable, and t denotes the time at which the transition is taken.)

a)

$$S(\text{or } \bar{S})$$

Enabling subcondition is a state variable S being true (or false) at time t .

b)

$$\{M_1, M_2, \cdots, M_m\}$$

Enabling subcondition is the system being in at least one of the specified modes at a finite interval up to time t .

c)

 E

Enabling subcondition is the occurrence of an event E at time t where E can be an

- a) external event, e.g., ΩE ,
- b) event denoting start of an action, e.g., $\uparrow A$,
- c) event denoting completion of an action, e.g., $\downarrow A$,
- d) event setting a state variable to true, e.g., $(S := T)$,
- e) event setting a state variable to false, e.g., $(S := F)$,
- f) event denoting entry into mode, e.g., $(M1 := T)$, or
- g) an event denoting exit from mode, e.g., $(M1 := F)$.

A transition from a mode is taken at a time t iff all the subconditions are satisfied at time t when the system is in that mode. Each of the three forms of a subcondition p_j can be expressed as an RTL predicate:

a)

$$S(t, t) \text{ (or } \bar{S}(t, t))$$

b)

$$M_1 < t, t) \vee M_2 < t, t) \vee \dots \vee M_m < t, t)$$

c)

$$@ (E, i) = t$$

In the above, t is the time at which the triggering condition for the transition holds, i.e., the transition is taken at that instant of time. We will shortly illustrate how these predicates are used in RTL formulas to express a mode transition.

2. *Lower/upper bound condition:* A condition c_k denoting a lower/upper bound restriction is of the form

$$(r, d)$$

where r is a nonnegative integer denoting a delay and d is a positive integer or ∞ denoting a deadline. If a lower/upper bound is specified as the condition for a transition from a mode, the transition can be taken after r time units and before d time units has elapsed since entering the mode. Three special forms of lower/upper bound condition are of particular interest. The condition

alarm r

is used to represent the case where the delay and the deadline on a transition are equal. In this case, the transition is taken exactly after r time units has elapsed since entering the mode. The condition

delay r

specifies a lower bound on when the transition can be taken without specifying an upper bound, i.e., the condition (r, ∞) . The condition

deadline d 

Fig. 3. Conditions on mode transitions.

specifies a finite upper bound and no lower bound on the transition, i.e., the condition $(0, d)$. The lower/upper bound restriction imposed on a transition from a mode M can be expressed as RTL predicates:

$$t + r \leq t' \wedge t' \leq t + d$$

where t is the time at which mode M is entered and t' is the time at which the transition is taken.

Having discussed the two forms of conditions on mode transitions, the triggering condition and lower/upper bound condition, we can show how a mode transition is formally specified in RTL. Suppose there is a transition from mode $M1$ to mode $M2$ and this transition is the only one out of mode $M1$. A triggering condition c_k on the transition arrow from mode $M1$ to mode $M2$ is captured by the following RTL formula:

$$\forall t M1(t, t) \wedge C \rightarrow \exists j @ ((M1 - M2), j) = t$$

where $(M1 - M2)$ is the mode transition event from $M1$ to $M2$, C is the conjunction of RTL predicates specifying the subconditions (the p_j s) in c_k , and t is the time at which the transition is taken. As an example, consider the two modes $M1$ and $M2$ in Fig. 3(a). The condition on the arrow states: while in mode $M1$, if the external event ΩE occurs when state variable S is true, the transition from mode $M1$ to mode $M2$ is taken. The corresponding RTL formula representing this mode transition is as follows:

$$\forall t \forall i M1(t, t) \wedge @ (\Omega E, i) = t \wedge S(t, t) \rightarrow \exists j @ ((M1 - M2), j) = t.$$

A lower/upper bound condition c_k on a transition arrow from $M1$ to $M2$ is expressed by the following RTL formula:

$$\forall t M1[t, t) \rightarrow \exists t' \exists j @ ((M1 - M2), j) = t' \wedge B$$

where $(M1 - M2)$ is the mode transition event from $M1$ to $M2$, B is the conjunction of RTL predicates specifying the lower/upper bound restrictions on the transition, t is the time of entering mode $M1$, and t' is the time at which the transition is taken. For example, consider the transition from $M1$ to $M2$ in Fig. 3(b). The condition on the arrow states that the transition is taken only after 20 time units and before 50 time units has elapsed since entering mode $M1$.

$$\forall t M1[t, t) \rightarrow \exists t' \exists j @ ((M1 - M2), j) = t' \wedge t + 20 \leq t' \leq t + 50.$$

In the preceding formulas, it is assumed that the transition from $M1$ to $M2$ is the only way to exit $M1$. In the case of multiple transitions and nested serial/parallel modes, a transition out of a mode is captured by a larger formula which is composed of formulas like the ones shown above. This will be dealt with in Section VI.

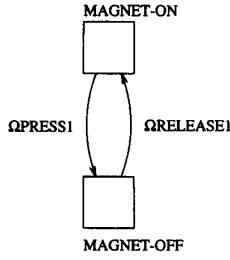


Fig. 4. Action upon mode entry/exit.

V. ACTIONS AND TIMING CONSTRAINTS

A real-time system may be required to execute certain actions while operating in some mode, or upon transition from one mode to another. Furthermore, the initiation and completion of actions are often subject to timing constraints. This section discusses the RTL semantics of actions in relation to mode transitions.

A. Actions Upon Mode Transitions

A system designer may wish to specify an action to be performed when an event triggering a mode transition occurs. For example, an action may be required to set a state variable upon exiting a mode and entering another, as shown in the system in Fig. 4. In this system, there are two modes in series: MAGNET-ON and MAGNET-OFF. The transition from MAGNET-ON to MAGNET-OFF happens when push-button #1 is pressed (external event ΩPRESS1); the transition from MAGNET-OFF to MAGNET-ON happens when push-button #1 is released (external event ΩRELEASE1). The state variable MAG must be set to false when ΩPRESS1 (triggering one transition) occurs and set to true when ΩRELEASE1 (triggering the other transition) occurs. We refer to the above two actions as A and B, respectively.

Since mode transitions are instantaneous in Modechart, an action to be performed upon a transition is performed in the destination mode. In other words, the action is triggered by the transition and will be performed upon entering the destination mode. For the example in Fig. 4, action A is performed after entering MAGNET-OFF mode; the state variable MAG remains true even after entering MAGNET-OFF until action A is completed (see Fig. 5).

In Modechart, we require each action in a system to be associated with a mode. Furthermore, at most one action may be associated with a mode. The action, referred to as the native action of the mode, may be initiated some time after entry into the mode. The exact time of execution is determined by the timing constraint imposed on the action. (In this paper, a timing constraint on an action is either sporadic or periodic.) The requirement of at most one entry action per mode may seem overly restrictive, since two or more actions with different timing constraints may need to be performed when a system is in a certain mode. Our approach is to create a child mode for each action. This way, no additional mechanism needs to be introduced in case execution of actions in the same mode is also subject to precedence or mutual exclusion

Action A: MAG:=F
Action B: MAG:=T

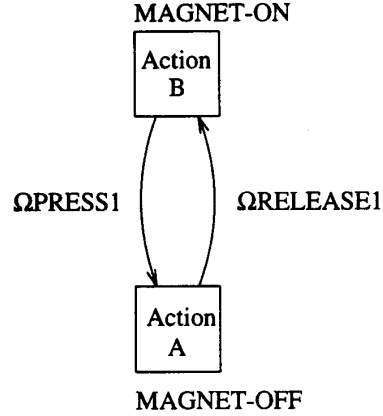


Fig. 5. Execution of actions.

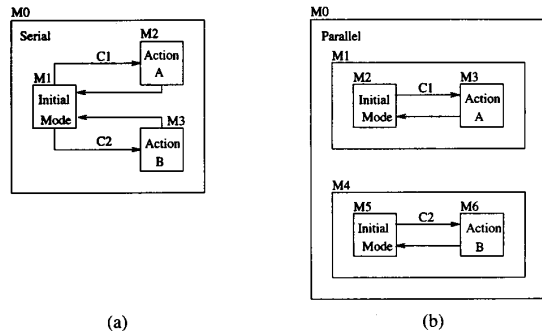


Fig. 6. Order of actions in a mode.

constraints.

For example, consider two actions, A and B which may be executed when a system is in mode M0. Action A is executed subject to a certain timing constraint when condition C1 becomes true and action B is executed when condition C2 becomes true, subject to another timing constraint. An application may require actions A and B to be performed in series, or the two actions must be executed in parallel. Both choices can be specified unambiguously by designating actions A and B as entry actions of two modes inside mode M0. Fig. 6 shows how mode M0 should be constructed to correctly reflect each choice.

B. Actions in Modes

Since mode transitions are instantaneous, there is a need to be precise about system behavior with respect to actions which are being executed when the condition for taking a transition becomes true. Consider the two modes, M1 and M2 shown in Fig. 7. Suppose execution of action A is started in mode M1. The transition from M1 to M2 should occur when C becomes true. If the execution of A has not completed, there are three different ways the execution of A may be terminated: (1) The transition cannot be taken until A is completed. (2) The transition aborts the action A. (3) The transition is taken and A is completed in the next mode. A scrutiny reveals that case (3) is unnecessary if the action is instead associated with

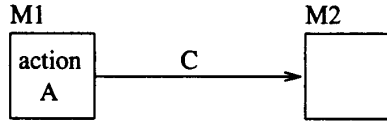


Fig. 7. Mode exit during an action.

a mode at a higher level. Intuitively, if the transition from $M1$ to $M2$ is taken when action A is in progress, the control information on the execution of action A will be lost. In other words, if action A can be in progress while the system is in $M1$ or $M2$, then action A should be associated with a mode parallel to $M1$ and $M2$. Of course, appropriate conditions must be specified in the new mode to ensure that the action is invoked only when the system enters mode $M1$.

Cases (1) and (2), however, must be formally captured in the formulas expressing mode transitions and the timing constraints imposed on actions. The remainder of this section discusses the modeling of sporadic and periodic timing constraints in Modechart. In each case, we give the RTL formulas capturing the semantics of mode transitions and timing constraints.

1) *Sporadic Timing Constraint*: The syntax for a sporadic timing constraint in a mode $M1$ is:

When in mode $M1$, if condition $C1$ is true,

execute action A with deadline = d , separation = s .

If condition $C1$ becomes true while the system is in mode $M1$ or if $C1$ is true when the system enters mode $M1$, the timing constraint is invoked requiring action A to be performed before the deadline d . We say that the timing constraint is *in effect* from the time of its invocation (when in $M1$ and $C1$ holds) to the time when action A is completed (before the deadline d). The separation parameter s is the minimum time that must pass between two successive invocations of the sporadic timing constraint. Fig. 8(a) illustrates how a sporadic timing constraint can be modeled in Modechart. Mode $M1$ contains two modes $M1'$ and $M1''$. The system is initially in mode $M1'$ upon entering $M1$. When condition $C1$ is true, the transition from $M1'$ to $M1''$ is taken. The timing constraint is in effect while the system is in mode $M1''$. Upon entering mode $M1''$, action A is performed with the deadline d . When action A is completed, the transition is taken back to the initial mode $M1'$. The separation parameter specifies the minimum time that must elapse between two successive entries to mode $M1''$.

The transition arrow from $M1$ to $M2$ represents the transition out of mode $M1$. The label C on the transition denotes an arbitrary condition specified by the system designer for exiting mode $M1$. We now consider the effect of a mode transition on the action A while the timing constraint on A is in effect.

a) *Transition Upon Completion of Action A*: If the timing constraint is in progress, one may wish to prevent a transition from $M1$ unless the execution of A is completed. In this case, Fig. 8(a) can be refined to Fig. 8(b), where $M1''$ is further expanded into two modes: $M3$ and $M4$. $M3$ represents the mode after invocation of the timing constraint and prior to the start of the execution of action A . $M4$ represents the

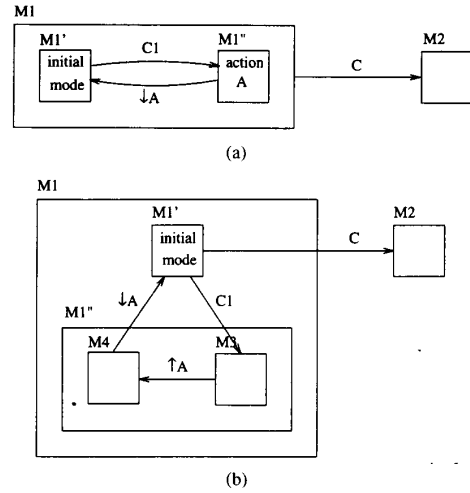


Fig. 8. Action in a sporadic timing constraint.

mode during the execution of action A . The motivation for expanding $M1''$ is to illustrate that one can model a more detailed execution of action A by identifying the control information denoted by modes $M3$ and $M4$. However, the most important change from Fig. 8(a) to (b) is the source of transition into mode $M2$. In particular, Fig. 8(b) shows that the transition from $M1$ to $M2$ is taken if condition C is true and the timing constraint to perform action A is not in progress. The following RTL formula expresses the timing constraint on the execution of action A upon entering $M1''$:

$$\forall t M1''[t, t] \rightarrow \exists i t \leq @(\uparrow A, i) \wedge @(\downarrow A, i) \leq t + d$$

The following assertion ensures the minimum separation as required by the timing constraint:

$$\forall i \forall t \forall t' @((M1'' := T), i) = t \wedge @((M1'' := T), i + 1) = t' \rightarrow t + s \leq t'$$

The mode transition assertion for the transition from $M1$ to $M2$ follows from the discussion in Section IV-B. In particular, if the condition on the transition arrow is a triggering condition, the following RTL formula captures the mode transition:

$$\forall t M1'(t, t) \wedge C \rightarrow \exists j @((M1' - M2), j) = t$$

where C in the above formula is the corresponding RTL predicates specifying the transition condition. However, if the condition on the transition arrow is a lower/upper bound restriction, the following RTL formula is applicable:

$$\forall t_1, t_2 M1[t_1, t_2] \wedge M1'(t_2, t_2) \wedge t_1 + r \leq t_2 \rightarrow \exists t_3 \exists j @((M1' - M2), j) = t_3 \wedge t_1 + r \leq t_3 \leq t_1 + d$$

where r and d are the lower and upper bounds specified on the transition.

b) *Transition Aborting Action A*: One may wish to terminate action A or not start it when a transition to a new mode occurs. In Fig. 8(a), if condition C on the transition arrow becomes true while the system is in mode $M1$, the transition from $M1$ to $M2$ is taken even if the timing constraint is in

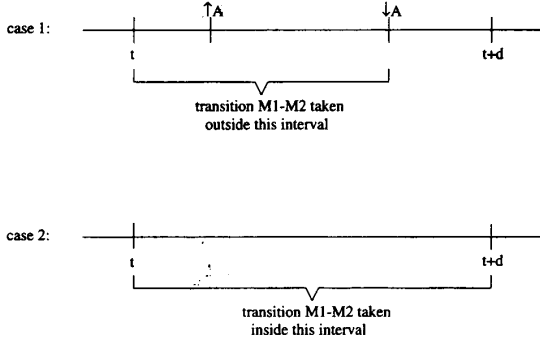


Fig. 9. Timing of actions that may be aborted.

effect. Hence, the mode transition assertion is very similar to case (a), except that the formula must now specify the transition from $M1$ to $M2$.

The RTL formula which expresses the timing constraint imposed on the execution of action A in mode $M1''$ must now reflect the fact that action A may be aborted due to a mode transition from $M1$ to $M2$. As shown in Fig. 9, there are two cases: (1) transition from $M1$ to $M2$ is not taken until after the completion of action A , or (2) transition from $M1$ to $M2$ is taken prior to the deadline imposed on action A , i.e., the action is not performed.

The following formula expresses the timing constraint on the execution of action A which may be aborted. We assume that action A is atomic. Hence, if it is aborted due to a transition from $M1$, it is up to the implementation to ensure that no inconsistencies exist in the system state. The two cases mentioned above are captured by the two disjuncts in the formula:

$$\begin{aligned} & \forall t M1''[t, t) \rightarrow \\ & (\exists i t \leq @(\uparrow A, i) \wedge @(\downarrow A, i) \leq t + d \wedge \\ & (\forall j @((M1 - M2), j) < t \vee @(\downarrow A, i) < @((M1 - M2), j))) \\ & \vee \\ & (\exists j t \leq @((M1 - M2), j) \leq t + d \wedge \\ & (\forall i @(\downarrow A, i) \leq t \vee @((M1 - M2), j) \leq @(\uparrow A, i))). \end{aligned}$$

2) *Periodic Timing Constraint*: A periodic timing constraint in a mode $M1$ is of the form:

When in mode $M1$, if condition $C1$ is true, execute A with period = p and deadline = d .

If condition $C1$ is true while in mode $M1$, the timing constraint requires an action A to be executed at fixed intervals with period p and deadline d . After its invocation (when condition $C1$ is true), the timing constraint is in effect during each period from the beginning of the period to the time when an instance of action A is completed before the deadline. Fig. 10(a) illustrates how a periodic timing constraint can be modeled in Modechart. Mode $M1$ contains two modes $M1'$ and $M1''$. The system is initially in mode $M1'$ upon entering $M1$. When condition $C1$ is true, the transition from $M1'$ to $M1''$ is taken denoting the invocation of the timing constraint. Upon entering mode $M1''$, the action is executed once every

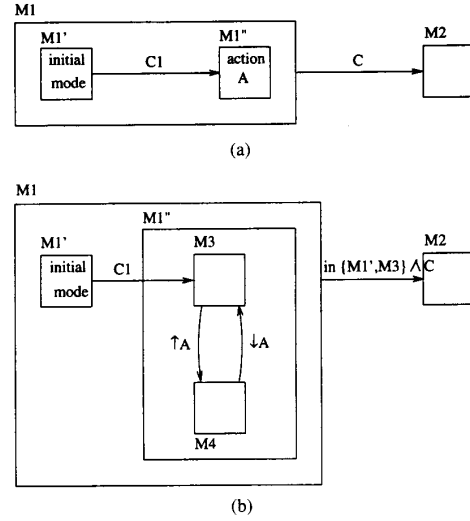


Fig. 10. Action in a periodic timing constraint.

p time units, as long as the system remains in mode $M1''$. The transition arrow from $M1$ to $M2$ represents the transition out of mode $M1$ when condition C is true.

As before, we consider the two possible cases where a periodic timing constraint is in effect and the condition for taking a transition out of the mode is true.

a) *Transition Upon Completion of the Action in the Current Period*: If the instance of action A in the current period has started, the transition is intended to occur after the execution of action A is completed. In this case, we refine Fig. 10(a) to (b) where $M1''$ is further expanded into two modes: $M3$ and $M4$. $M3$ represents the mode after the beginning of a period and prior to the start of action A ; $M4$ represents the mode during the execution of action A . The following RTL formula captures the timing constraint imposed on the execution of action A upon entering mode $M1''$:

$$\begin{aligned} & \forall t \forall t' \forall n M1''[t', t) \wedge n * p \leq t - t' \rightarrow \\ & \exists i t' + n * p \leq @(\uparrow A, i) \wedge @(\downarrow A, i) \leq t' + n * p + d. \end{aligned}$$

The mode transition assertion for the transition from mode $M1$ to $M2$ is as follows:

$$\forall t [M1'(t, t) \vee M3(t, t)] \wedge C \rightarrow \exists j @((M1' - M2), j) = t.$$

b) *Transition Aborting Action A*: In this case, the action in the current period is aborted or not started when a transition to a new mode occurs. In Fig. 10(a), if condition C becomes true while the system is in mode $M1$, the transition from $M1$ to $M2$ is taken even if the timing constraint is in effect for the current period. Hence, the mode transition assertion for the transition from $M1$ to $M2$ is as described in Section IV-B. However, the RTL formula expressing the timing constraint imposed on the execution of action A in mode $M1''$ must now reflect the fact that action A may be aborted due to a mode transition from $M1$ to $M2$ shown at the top of the page. The motivation of this formula is similar to that of the sporadic case.

$$\begin{aligned}
& \forall t \forall t' \forall n \geq 0 \ M1''[t', t] \wedge n * p < t - t' \rightarrow \\
& (\exists i \ t' + n * p \leq @(\uparrow A, i) \wedge @(\downarrow A, i) \leq t' + n * p + d \wedge \\
& (\forall j \ @((M1 - M2), j) < t' \vee @(\downarrow A, i) \leq @((M1 - M2), j))) \\
& \vee \\
& (\exists j \ t' + n * p < @((M1 - M2), j) \leq t' + n * p + d \wedge \\
& (\forall i \ @(\downarrow A, i) \leq t' + n * p \vee @((M1 - M2), j) \leq @(\uparrow A, i))).
\end{aligned}$$

VI. ENTERING AND EXITING NESTED MODES

We now give RTL formulas to capture the meaning of transitions out of modes which are nested inside other modes. Firstly, entering and exiting nested modes must be precisely defined:

Definition (explicit and implicit entry): A transition *explicitly* enters a mode *M* if

- the arrow ends at *M*, or
- the arrow crosses the boundary into *M*.

A transition *implicitly* enters a mode *M* if

- 1) *M* is an initial mode, *M'* the immediate parent of *M* is a serial mode, and
 - a) the arrow ends at *M'*, or
 - b) the transition implicitly enters *M'*.
- 2) *M'* the immediate parent of *M* is a parallel mode, and
 - a) the arrow ends at *M'*, or
 - b) the transition implicitly enters *M'*, or
 - c) the transition explicitly enters a sibling of *M*.

Definition (explicit and implicit exit): A transition *explicitly* exits a mode *M* if

- the arrow originates from *M*, or
- the arrow crosses the boundary out of *M*.

A transition *implicitly* exits a mode *M* if

- 1) *M'* the immediate parent of *M* is a serial mode, the system is in mode *M*, and
 - a) the arrow originates from *M'*, or
 - b) the transition implicitly exits *M'*.
- 2) *M'* the immediate parent of *M* is a parallel mode and
 - a) the arrow originates from *M'*, or
 - b) the transition implicitly exits *M'*, or
 - c) the transition explicitly exits a sibling of *M*.

For example, in Fig. 11, the transition from *M4* to *M3* has explicit exits from modes *M4* and *M1*, an explicit entry into *M3*, and implicit entries into modes *M6* and *M7*. The transition from *M3* to *M1* has explicit exits from *M3*, implicit exits from modes *M6* and *M7*, an explicit entry into *M1*, and an implicit entry into *M4*.

Definition (at level): Suppose mode *M'* is an immediate child of mode *M*, then mode *M'* is said to be at level *M*.

In Fig. 11, modes *M1*, *M2*, and *M3* are at level *M0*, and *M4* and *M5* are at level *M1*. Modes *M6* and *M7* are at level *M3*.

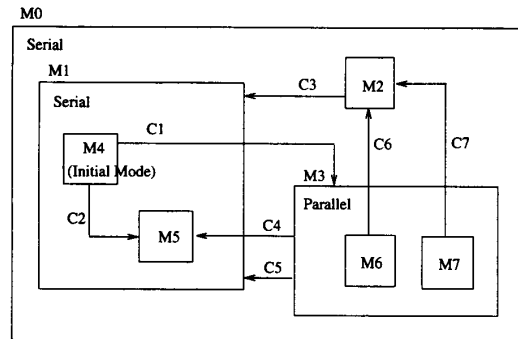


Fig. 11. Entry/exit of nested modes.

Secondly, since a Modechart system can be in more than one mode simultaneously and multiple transitions can originate from a mode, it is possible that more than one transition can be taken at an instant of time. We identify a set of rules which identify simultaneous transitions that violate the semantics of Modechart. We motivate the discussion by considering three simple examples shown in Fig. 12. In Fig. 12(a), two transitions (*M1 - M2*) and (*M1 - M3*) originate from the same mode, thus both transitions cannot be taken at the same instant of time. However, in Fig. 12(b), a transition inside *M1* can happen simultaneously with a transition inside *M2*, because *M1* and *M2* are in parallel. In contrast to the first two examples, Fig. 12(c) illustrates a potential problem due to simultaneous transitions at different levels of hierarchy. Consider the two transitions (*M2 - M3*) and (*M0 - M1*). On one hand, one can view the transition from *M0* to *M1* as an abbreviation for a transition from *M2* to *M1*. Then, if the system is in *M2*, transitions (*M2 - M3*) and (*M0 - M1*) cannot happen at the same time because both originate from *M2*. On the other hand, one can view the transition from *M0* to *M1* as an abbreviation for a transition from *M3* to *M1*. Then, simultaneous transitions (*M2 - M3*) and (*M0 - M1*) can be seen as the system moving from *M2* to *M3* and then to *M1*. We define the semantics of transitions from nested modes so that both computations are allowed. Hence, if both transitions in Fig. 12(c) can be taken at an instant of time, then either (*M0 - M1*) is taken, or both (*M2 - M3*) and (*M0 - M1*) are taken. The following rule applies to transitions exiting a mode:

- Two transitions explicitly exiting a mode are subject to a mutual exclusion constraint.

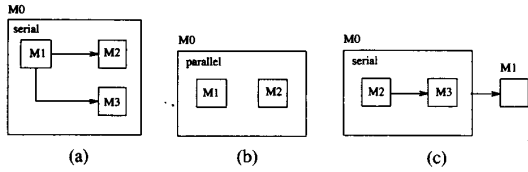


Fig. 12. Simultaneous transitions in nested modes.

If a mutual exclusion constraint is imposed on two distinct transitions that can be taken simultaneously, then either (but not both) are taken. Otherwise, either or both may be taken. In Fig. 12(a), the two transitions explicitly exit mode $M1$, thus they cannot occur simultaneously. In Fig. 12(c), although transition ($M2 - M3$) explicitly exits $M2$, transition ($M0 - M1$) does not explicit exit $M2$. Therefore, the two transitions can occur at the same instant of time. In Section VI-D, we will discuss an alternative semantics such that a transition originating from a mode has higher precedence over a transition from a mode nested inside it.

A. Mode Transition Exclusion

When a mutual exclusion constraint is imposed on two transitions, an assertion is introduced which prevents the simultaneous occurrence of the corresponding transition events. Let e_1 and e_2 be mode transition events corresponding to two distinct transitions explicitly exiting a mode, the exclusion assertion is:

$$\forall i \forall j @ (e_1, i) \neq @ (e_2, j).$$

For each pair of transitions subject to a mutual exclusion constraint, an exclusion assertion is specified.

B. Implicit Mode Exits

When a transition is taken, a set of modes may be explicitly and implicitly exited and another set of modes explicitly and implicitly entered. As will be shown in Section VI-C, a formula capturing a mode transition will specify explicit and implicit mode entries and explicit mode exits. The reason for including the explicit entries and implicit entries as part of a single mode transition assertion (Section VI-C) is that they cannot be specified independent of the specific transition taken to enter a compound mode. For example, in Fig. 11, entering mode $M1$ could mean entering $M4$ or $M5$ depending on the transitions. For implicit mode exits, however, the corresponding RTL formula can be written independent of the transitions.

Suppose a system exits at time t from a mode M , which has as its immediate children the modes M_1, M_2, \dots, M_n . If M is a serial mode, the following assertion captures the exit from its immediate children.

$$\begin{aligned} M(t, t] &\rightarrow (M1(t, t) \rightarrow M1(t, t]) \wedge \\ &\quad (M2(t, t) \rightarrow M2(t, t]) \wedge \\ &\quad \vdots \\ &\quad (Mn(t, t) \rightarrow Mn(t, t]). \end{aligned}$$

If M is a parallel mode, the assertion to capture implicit exits is simply:

$$M(t, t] \rightarrow M1(t, t] \wedge M2(t, t] \cdots \wedge Mn(t, t].$$

Finally, specifying explicit exits as part of the mode transition assertion allows us to use the above assertion for exiting all modes which are exited implicitly due to the transition.

C. The Transition Assertion For Nested Modes

We shall use a single assertion to capture the transitions in a set of nested modes. The mode transition assertion is structured to reflect the serial and parallel structure of the system. Intuitively, a formula at a level M describes the system behavior specified by the modes nested in M . At a level M , we include all the formulas describing the transitions originating from each immediate child M' of M . In turn, the level M' formulas describe the modes inside M' , their transitions and timing constraints. (A complete specification will also include the formulas capturing the timing constraints imposed on the actions in M' . For the purpose of this section, we shall not be concerned with timing constraints. To deal with sporadic and periodic timing constraints, the transition assertion given here can be modified in a straightforward manner in accordance with the discussion in Section V-B.)

Suppose M is a serial mode and M' is an immediate child of M . Let n be the number of transitions from M' with a triggering condition on the arrow and let C_1, C_2, \dots, C_n be the corresponding triggering conditions. Let m be the number of transitions from M' with a lower/upper bound condition on the arrow and let B_1, B_2, \dots, B_m be the corresponding bounds on the transitions. (The C_i 's and B_i 's are RTL predicates as described in Section IV-B.) The following formulas capture the transitions originating from an immediate child M' of mode M :

$$\begin{aligned} \{ \text{Level } M \text{ transition assertion is the conjunction of formulas} \\ \text{below for each } M' \} \\ M'(t, t) \rightarrow (C_1 \rightarrow T_1 \vee M'(t, t]) \wedge \\ (C_2 \rightarrow T_2 \vee M'(t, t]) \wedge \\ \dots \\ (C_n \rightarrow T_n \vee M'(t, t]) \end{aligned} \quad (1)$$

$$\begin{aligned} M'[t, t) \rightarrow [(T_{n+1} \wedge B_1) \vee (M'[t, t'] \wedge t' \leq t + d_1)] \wedge \\ [(T_{n+2} \wedge B_2) \vee (M'[t, t'] \wedge t' \leq t + d_2)] \wedge \\ \dots \\ [(T_{n+m} \wedge B_m) \vee (M'[t, t'] \wedge t' \leq t + d_m)] \end{aligned} \quad (2)$$

$$\begin{aligned} T_1 &\rightarrow R_1 \\ T_2 &\rightarrow R_2 \\ &\dots \\ T_{n+m} &\rightarrow R_{n+m} \\ M'(t, t) &\rightarrow \{ \text{Level } M' \text{ formulas} \} \end{aligned} \quad (3)$$

where T_i is a predicate denoting the occurrence of the mode transition event corresponding to the i th transition from M' , and each R_i is the conjunction of

- the mode predicates denoting explicit exits for the i th transition, and
- the mode predicates denoting explicit and implicit mode entries for the i th transition.

Equation (1) specifies the transitions from M' with triggering conditions. Equation (2) specifies the transitions from M' with lower/upper bounds. Equation (3) specifies the condition for level M' formulas by asserting that if the system is in M' and none of the transitions originating from M' has been taken at time t , then the assertions about the mode transitions nested in M' will hold.

For example, assume the conditions on the transitions in Fig. 13(a) are all triggering conditions. The level $M0$ mode transition assertion is:

$$\begin{aligned} & \{\text{Level } M0 \text{ transition assertion}\} \\ & M1(t, t) \wedge C1 \rightarrow \exists i@((M1 - M2), i) = t \vee M1(t, t) \\ & M1(t, t) \wedge C2 \rightarrow \exists i@((M1 - M3), i) = t \vee M1(t, t) \\ & \forall i@((M1 - M3), i) = t \rightarrow M1(t, t) \wedge M0(t, t) \wedge M3[t, t) \\ & \forall i@((M1 - M2), i) = t \rightarrow M1(t, t) \wedge M2[t, t) \\ & M1(t, t) \rightarrow \{\text{Level } M1 \text{ formulas}\} \\ & M2(t, t) \wedge C3 \rightarrow \exists i@((M2 - M3), i) = t \vee M2(t, t) \\ & \forall i@((M2 - M3), i) = t \rightarrow M2(t, t) \wedge M0(t, t) \wedge M3[t, t) \\ & M2(t, t) \rightarrow \{\text{Level } M2 \text{ formulas}\}. \end{aligned}$$

If M is a parallel mode, the following formulas capture the transitions originating from all immediate children of mode M .

$$\begin{aligned} & \{\text{Level } M \text{ transition assertion is the conjunction of} \\ & \text{the following formulas}\} \\ & C_1 \rightarrow T_1 \vee M(t, t) \\ & C_2 \rightarrow T_2 \vee M(t, t) \\ & \dots \\ & C_n \rightarrow T_n \vee M(t, t) \\ & M[t, t) \rightarrow \\ & [(T_{n+1} \wedge B_1) \vee (M[t, t'] \wedge t' \leq t + d_1)] \wedge \\ & [(T_{n+2} \wedge B_2) \vee (M[t, t'] \wedge t' \leq t + d_2)] \wedge \\ & \dots \\ & [(T_{n+m} \wedge B_m) \vee (M[t, t'] \wedge t' \leq t + d_m)] \\ & T_1 \rightarrow R_1 \\ & T_2 \rightarrow R_2 \\ & \dots \\ & T_{n+m} \rightarrow R_{n+m} \\ & M'(t, t) \rightarrow \{\text{Level } M' \text{ formulas where } M' \text{ is an immediate} \\ & \text{child of } M\} \end{aligned}$$

where C_i , B_i , T_i , R_i , and d_i are as described in the serial case earlier. Observe that the above formulas hold when the system is in mode M at time t . Since M is a parallel mode,

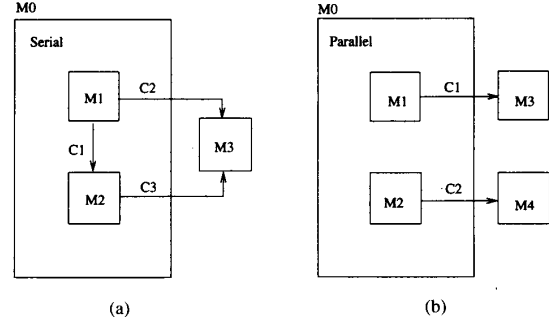


Fig. 13. Pertaining to mode transition axiom.

it is not necessary to specify the mode M' in the antecedents of the implications of the above formulas.

As an example, assume both conditions on the transitions in Fig. 13(b) are triggering conditions. The level $M0$ mode transition assertion is as follows:

$$\begin{aligned} & \{\text{Level } M0 \text{ transition assertion}\} \\ & C1 \rightarrow \exists i@((M1 - M3), i) = t \vee M0(t, t) \\ & C2 \rightarrow \exists i@((M2 - M4), i) = t \vee M0(t, t) \\ & \forall i@((M1 - M3), i) = t \rightarrow M1(t, t) \wedge M0(t, t) \wedge M3[t, t) \\ & \forall i@((M2 - M4), i) = t \rightarrow M2(t, t) \wedge M0(t, t) \wedge M4[t, t) \\ & M1(t, t) \rightarrow \{\text{Level } M1 \text{ formulas}\} \\ & M2(t, t) \rightarrow \{\text{Level } M2 \text{ formulas}\}. \end{aligned}$$

We conclude this section by discussing how the semantics of Modechart deals with the potential ambiguities caused by simultaneous transitions. The precedence rules and the transition exclusion assertions handle the case where two transitions can (explicitly or implicitly) exit a mode. If the two transitions have the same precedence, either one (but not both) can be taken. Otherwise, the transition with higher precedence is taken. However, the semantics of Modechart allows that any two concurrent transitions inside two modes in parallel to be taken simultaneously. Recall the parallel mode M in Fig. 2(b). The following assertion expresses the two mode transitions in M :

$$\begin{aligned} & \forall tM(t, t) \rightarrow \\ & \quad \{\text{Level } M' \text{ transition assertion}\} \\ & \quad [M1(t, t) \wedge M3(t, t) \rightarrow \exists i@((M1 - M2), i) = t] \wedge \\ & \quad \{\text{Level } M'' \text{ transition assertion}\} \\ & \quad [M3(t, t) \wedge M1(t, t) \rightarrow \exists i@((M3 - M4), i) = t]. \end{aligned}$$

Assume that the system is in mode $M1$ and $M3$ initially. The preceding assertion allows the two transitions, from $M1$ to $M2$ and from $M3$ to $M4$, to occur at the same time. If there are lower/upper bounds on the transitions, then the semantics of Modechart does not require both transitions to be taken.

D. Transitions with Precedence

An alternative semantics for transitions from nested modes is to define a set of precedence rules on transitions which can be taken at the same instant of time. Recall the example in

Fig. 12(c). The semantics of transitions from nested modes can be defined so that transition $(M0 - M1)$ has a higher precedence than transition $(M2 - M3)$. Hence, the transition events $(M0 - M1)$ and $(M2 - M3)$ are prevented from occurring at the same instant of time. The precedence among transitions (explicitly or implicitly) exiting a mode is defined below:

- 1) Transitions originating from the same mode have the same precedence.
- 2) Transitions originating from modes in parallel and at the same level have the same precedence.
- 3) Transitions originating from inside two distinct modes in parallel at level M and exiting out of M have the same precedence.
- 4) Transitions originating from a mode M have precedence over transitions originating from the children of M .
- 5) Transitions originating from a mode M have precedence over transitions originating from children of a mode M' , where M and M' are in parallel.

Referring to Fig. 11, $(M4 - M3)$ has the same precedence as $(M4 - M5)$ because of precedence rule (1). By rule (2), both $(M6 - M2)$ and $(M7 - M2)$ have the same precedence. $(M3 - M1)$ has precedence over both $(M6 - M2)$ and $(M7 - M2)$ by rule (4). Any transitions coming out from inside $M6$ and $M7$ would have had the same precedence by rule (3). Lastly, by rule (5), $(M6 - M2)$ would have had precedence over transitions coming from inside $M7$ and $(M7 - M2)$ would have had precedence over transitions coming from inside $M6$.

Having defined the notion of precedence among transitions from a mode, we can now specify rules for simultaneous transitions. If two transitions out of a mode must be taken at the same instant of time, then the transition with the higher precedence is taken. If their precedence is the same, then either (but not both) is taken. If no precedence is defined between the transitions, then both may be taken. The transition assertions for nested serial and parallel modes, as presented in Section VI-C, can be rewritten to reflect the above precedence rules.

VII. ABSENCE OF ANOMALOUS BEHAVIOR

In the preceding sections, we described the syntax and provided a formal semantics for Modechart in terms of Real Time Logic. In this section, we give a theorem which shows that the well-formedness requirement and the UDIM condition ensure the proper behavior of serial and parallel modes as described in Section II. We also state a condition and give a theorem for preventing the anomaly which arises from cycles of transitions.

Lemma 1: If the root mode of a system is well-formed and the UDIM condition holds, then

- a) if a transition (explicitly or implicitly) enters a serial mode M , it enters exactly one of the immediate children of M .
- b) if a transition (explicitly or implicitly) enters a parallel mode M , it enters all of the immediate children of M .
- c) if a transition exits from a (serial or parallel) mode M , it exits from all of the immediate children of M .

Proof: The proof follows directly from the well-formedness assumption, the UDIM condition, and the precedence rules described in the previous section. \square

Theorem 2: Suppose the root mode of a system is well-formed, and the UDIM condition holds for the system. Suppose M is a compound mode and M_i, M_j are two of its immediate children.

- a) If M is a serial mode, then if the system is in mode M_i for a nonzero interval of time, it cannot be in another mode M_j for nonzero units of time in the same interval.
- b) If M is a parallel mode, then if the system is in mode M_i for a nonzero interval of time, it is also in each mode M_j during the same time interval.

Proof: We prove this theorem by induction. We start with the top-most mode as the base case. We then show that if properties (a) and (b) hold for all ancestors of a mode, they also hold for the mode.

Base Case: (Top-Most Mode)

- A) Consider the case where the top-mode mode M is a serial mode. Let M_i and M_j be two immediate children of M . Lemma 1 implies that the system will be in exactly one mode upon system start-up.

Assume that the system is in mode M_i during the interval from t_i to t'_i where $t_i < t'_i$. If the system enters another mode M_j at time t_j where $t_i \leq t_j < t'_i$, it cannot remain in mode M_j for any nonzero time interval. Otherwise, the system must have exited M_i at time t_j and stayed out of M_i for a nonzero time interval.

- B) Consider the case where the top-most mode M is a parallel mode. Let M_i and M_j be two immediate children of M . We know by Lemma 1 that the system starts in both M_i and M_j . Furthermore, the well-formedness property implies that there is no transition connecting the two modes M_i and M_j . Hence, the system will remain in both modes.

Induction Step:

- A) Let M be a serial mode such that properties (a) and (b) in the statement of the theorem hold for ancestors of M . We need to show that property (a) holds for M . Again, let M_i and M_j be two immediate children of M .

Assume the system is in mode M_i during the interval from t_i to t'_i where $t_i < t'_i$. Furthermore, assume the system enters another mode M_j at time t_j where $t_i \leq t_j < t'_i$. The system cannot remain in mode M_j for any nonzero time interval for the following reason. M_j is entered because of the occurrence of a sequence of transitions at time t_j . The first mode in this sequence of transitions can be either one of the two cases:

- a) The sequence of transitions originated from mode M_i . But this is impossible, because it means that the system is not in M_i from t_j to $t'_i + 1$.
- b) The sequence of transitions originated from a mode M' outside mode M . From the well-formedness assumption, we know that M (or

its ancestor) is in series with M' (or its ancestors). This fact implies that the system was simultaneously in M and M' right before time t_j for a nonzero interval of time. This is a contradiction to the hypothesis of the induction step.

- B) Let M be a parallel mode such that properties (a) and (b) in the statement of the theorem hold for ancestors of M . We need to show that property (b) holds for mode M as well.

Assume the system is in mode M_i during the interval from t_i to t'_i where $t_i < t'_i$. Furthermore, assume that the system is in mode M_j until it exits M_j at time t_j where $t_i \leq t_j < t'_i$. If the system remains outside M_j in another mode M' for some nonzero time interval, then by the well-formedness assumption we know that M (or its ancestor) is in series with M' (or its ancestor). But this is a contradiction to the hypothesis of the induction step, because it implies that the system is in two modes (in series) simultaneously during the interval from t_j to $t_j + 1$. \square

In Section II, we described an anomaly which may result from a cycle of transitions. Clearly, it is too restrictive to disallow transition cycles in a specification. However, if we can ensure that there is a positive delay or an action is executed in one of the modes involved in a cycle of transitions, the anomaly is avoided. The following definitions will be used in a sufficient condition for preventing the transition cycle anomaly.

Definition (Cycle of Transitions): Let T_1, T_2, \dots, T_m denote a sequence of transitions. The transitions form a cycle on a set of modes M_1, M_2, \dots, M_m if

- each transition T_i (implicitly or explicitly) exits M_i and (implicitly or explicitly) enters M_{i+1} for all $1 \leq i < m$, and
- transition T_m exits mode M_m and enters M_1 .

Definition (Blocked Transition): A transition exiting mode M and entering mode M' is blocked if

- there is an action associated with mode M such that the transition cannot be taken until that action completes, or
- there is a positive delay imposed on the transitions.

Theorem 3: A Modechart specification is free from the transition cycle anomaly if in each cycle of transitions, at least one transition t exiting mode M and entering M' is blocked by M .

Proof: If a transition in a cycle is blocked by an action or a delay, the cycle cannot be traversed instantaneously. Hence, the anomaly is avoided. \square

For each transition cycle, the preceding theorem ensures that the system remains in at least one mode for nonzero units of time. This is possible because either at least one action is executed in one of the modes involved in the cycle, or there is a positive delay on one of the transitions in the cycle. We remark that the above condition is sufficient but not necessary to prevent the transition cycle anomaly.

VIII. CONCLUSION

In this paper, we have presented a language called Modechart for the specification of real-time systems. Modechart owes its origin to the *mode* concept of Parnas *et al.* in their systems requirement work on the A-7E aircraft and also to the Statechart language of Harel *et al.*. The emphasis of Modechart, however, is in the specification of absolute timing properties. A formal semantics of Modechart was given in terms of RTL (Real Time Logic).

The concept of modes is familiar to designers of process control systems. Hence, Modechart should be an easy specification language to use for defining real-time systems. More importantly, the specification of a system in terms of modes gives us more leverage in the verification of timing properties. Serial and parallel modes provide a way to organize the assertions about system behavior (RTL formulas) in a hierarchical and compartmental organization. Proof techniques can take advantage of this organization to focus on the set of relevant assertions to establish a timing property. The ability to avoid considering all the assertions defining the behavior of a large system is a prerequisite to practical applications of verification technology.

A preliminary version of a software tool which allows a user to describe a system specification in Modechart and a translator for generating the corresponding RTL formulas has been developed as part of the SARTOR project. The graphics-based tool for generating a Modechart specification, the Modechart Constructor consists of two components: (1) an icon-driven user interface for creating, displaying and modifying a specification, and (2) a database manager from which the information is retrieved by the user interface through message passing.

The software tool, TRANS for generating RTL formulas from a Modechart specification performs the translation in two stages. First, TRANS converts the Modechart specification in the database into an intermediate ASCII format. Then it generates the appropriate RTL formulas as queried by the user. The primary advantage of a two-stage translation is that it allows the user to bypass the Modechart Constructor tool when a bit-map workstation is not available. A system specification in the ASCII format can also be used directly as input to the TRANS software for generating the corresponding RTL formulas.

Ongoing work investigates the theoretical and practical issues in designing a verifier which takes advantage of the organizational structure of a system as specified by the use of Modechart.

REFERENCES

- [1] M. W. Alford, "A requirements engineering methodology for real-time processing requirements," *IEEE Trans. Software Eng.*, vol. SE-3, Jan. 1977.
- [2] C. G. Davis and C. R. Vick, "The software development system," *IEEE Trans. Software Eng.*, vol. SE-3, pp. 69-84, Jan. 1977.
- [3] D. Harel, "Statecharts: A visual formalism for complex systems," The Weizmann Inst. of Sci., Israel, Tech. Rep., July 1986.
- [4] D. Harel, A. Pnueli, J. P. Schmidt, and R. Sherman, "On the formal semantics of statecharts," in *Proc. 2nd Symp. Logic in Comput. Sci.*, June 1987, pp. 54-64.

- [5] K. L. Heninger, "Specifying software requirements for complex systems: New techniques and their application," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 2-13, Jan. 1980.
- [6] F. Jahanian, R. S. Lee, and A. K. Mok, "Semantics of Modechart in real-time logic," in *Proc. 21st Hawaii Int. Conf. Syst. Sci.*, Jan. 1988.
- [7] F. Jahanian, A. K. Mok, and D. A. Stuart, "Formal specification of real-time systems," Dep. of Comput. Sci., Univ. of Texas at Austin, Tech. Rep. TR-88-25, June 1988.
- [8] F. Jahanian and A. K. Mok, "Safety analysis of timing properties in real-time systems," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 890-904, Sept. 1986.
- [9] Luqi and V. Berzins, "Rapid prototyping of real-time systems," *IEEE Software*, Sept. 1988.
- [10] A. K. Mok, "SARTOR—A design environment for real-time systems," in *Proc. 9th IEEE COMPSAC*, Chicago, IL, Oct. 1985, pp. 174-181.
- [11] A. Pnueli, private communication, 1987.
- [12] D. Parnas, K. L. Heninger, J. W. Kallander, and J. E. Shore, "Software requirements for the A-7E aircraft," NRL memo. rep. 3876, Washington, DC, Nov. 1978.
- [13] P. Zave, "A distributed alternative to finite-state-machine specifications," *ACM Trans. Prog. Lang. Syst.*, pp. 10-36, July 1985.

Aloysius K. Mok received the B.S. and M.S. degrees in electrical engineering and computer science, and the Ph.D. degree in computer science in 1983, all from the Massachusetts Institute of Technology.

He is currently an Associate Professor of Computer Science at the University of Texas at Austin, where he holds a Faculty Fellowship in Computer Science. His current interests include design problems of robust, distributed real-time systems, real-time database and performance issues of real-time rule-based programs. He has consulted for government and industry on real-time system design.



Farnam Jahanian (S'82-M'87) received the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin in 1987 and 1989, respectively.

He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. Prior to joining the faculty at the University of Michigan last year, he had been a Research Staff Member at the IBM T. J. Watson Research Center where he led several experimental research projects in distributed and fault-tolerant systems. His research work has been successfully transferred to several development labs on projects including a highly-available automated manufacturing system and a commercial parallel server. His current research interests include specification and analysis of real-time systems and fault-tolerant computing.