# An Operational Semantics and a Compiler for Modechart Specifications

Carlos Puchol*, Douglas A. Stuart and Aloysius K. Mok
{cpg,dastuart,mok}@cs.utexas.edu
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188

**Keywords:** Formal specification, real-time systems, reactive systems, synchronous systems, operational semantics, compilers.

**Abstract**

The Modechart specification language is a formalism for specification of real-time systems. The original semantics of the language is described in an axiomatic style, with Real-Time Logic being the underlaying logic. We introduce a new formal semantics for it in an operational style in two steps. The semantics for the class of deterministic specifications is introduced first and then it is extended for the general class of non-deterministic specifications. The deterministic semantics leads very naturally to an implementation of a compiler, which is described as well. An enhancement to that compiler provides support for a limited, but very useful in practice, subset of the class of non-deterministic specifications.

## 1 Introduction

The Modechart specification language is a synchronous language especially amenable for specifying the behavior of real-time systems with absolute timing constraints [JM94]. It was designed within the SARTOR project [Mok85], an effort towards an environment for the formal design, analysis and implementation of real-time systems. It is based on the *synchrony hypothesis*. The synchrony hypothesis models the system as being theoretically infinitely faster than the environment —it assumes instantaneous broadcast of internal signals and immediate response to external inputs.

This paper introduces a new, formal operational semantics for pure Modechart specifications. Pure Modechart specifications are those not containing actions (non-synchronous functional objects attached to states of the specification). The execution model assumes that the internal events that take place in a reaction are infinitely fast. An integer model for time is assumed, where the system is said to take transitions or steps at integer intervals. Hence, the the semantics is built in two levels, namely, the "step" level and the "reaction" level, which is subsumed within the former. Two Kripke structures model these semantic levels. For simplicity, we present the development in two stages. The first stage describes the semantics for the class of deterministic specifications. The second stage takes the deterministic semantics and augments it to model all pure Modechart specifications.

The original formal semantics for Modechart is an axiomatic semantics defined in RTL (Real-Time Logic [JM86]). While the rigor of that definition allows for strong analytical capabilities, it does not address a computational approach to the language. The goal of introducing the operational semantics for Modechart is to provide a semantics which captures a more practical and intuitive, yet rigorous, notion of behavior, i.e. a semantics suitable for implementation as well as reference. The deterministic semantics defines a compiler for the class of specifications it describes. We present the compiler algorithm for the language after introducing the complete semantics.

This paper is organized as follows. Section 3 provides the notation and basic definitions used during the rest of the paper as well as the syntax of Modechart. Section 4 defines a two-level Kripke structure used in defining the first stage of the semantics covering the deterministic specifications. Section 5 essentially redefines the reaction structure,

*Supported by a Fulbright fellowship from Ministry of education, Spain.

adding the capabilities for non-deterministic features of Modechart. Section 6 goes over special cases of specifications that have been considered particularly problematic in the original axiomatic semantics. Section 7 shows that the newly introduced semantics is compatible with the axiomatic one. The compiler based on the deterministic semantics is described in Section 8. Section 9 offers the conclusions obtained from this work.

# 2 Modechart Overview

# 3 Modechart Syntax

Modechart is a graphical language, however we concentrate on laying out the formal syntax of the language capturing the graphical rules that govern it.

**Definition:** A *modechart* is a 5-tuple $\langle \mathcal{M}, \{S, P, initial\}, \sqsubseteq, \mathcal{T}, \mathcal{E} \rangle$ with the following components:

- $\mathcal{M}$ is a (finite) set of *modes*.
- $S \subset \mathcal{M}$ is a set of *serial* modes and $P \subset \mathcal{M}$ a set of *parallel* modes, with $S \cap P = \emptyset$ and $S \cup P \neq \mathcal{M}$. The total function *initial*: $S \rightarrow \mathcal{M}$ denotes a mode (the *default* or *initial* mode) corresponding to a serial mode.
- $\sqsubseteq$ is a partial order relation in $\mathcal{M}$ defining a tree among the elements of $\mathcal{M}$. We use often the relations $\sqsubset, \not\sqsubseteq$, and $\not\sqsubset$ naturally defined after $\sqsubseteq$. If $m \sqsubset n$ we say "$m$ is contained in $n$," "$m$ is a descendant of $n$" or "$n$ is an ancestor of $m$." It is required that $S \cup P = \{m \in \mathcal{M} \mid n \sqsubset m, \text{ for some } n \in \mathcal{M}\}$.
- $\mathcal{T} = \mathcal{M} \times \mathcal{M} \times MTE$ is a set of *transitions*. For a transition $(s, t, e)$, $s$ is the *source* mode of the transition, $t$ is the *target* mode of the transition and $e$ is the Modechart *transition expression* (MTE) associated with the transition, as defined by the grammar below.
- $\mathcal{E}$ is a (finite) set of external *input signals*.

**Definition:** The set of *external output signals* of a modechart specification is defined as a set of labels, and it is formed by the union of four sets of labels:

$$\mathcal{L} = \mathcal{M}^+ \cup \mathcal{M}^- \cup \mathcal{M}^\perp \cup \mathcal{M}^\rightarrow,$$

where a set of *mode entry* labels $M^+$ for $M \subseteq \mathcal{M}$ is defined as follows:

$$M^+ = \{\ m^+ \mid m \in M\ \},$$

and $m^+$ is an entry label for mode $m$. The sets of labels $M^-$ and $M^\perp$ are defined similarly, but for *exit mode* and *active mode* labels. Thus every mode $m$ in the system has three associated outputs which are meant to capture the evolution of the system, one denoting the system entering the mode, $m^+$, one denoting the exit of the mode, $m^-$, and one denoting the mode being active, $m^\perp$, each of which can be present or absent each time instant. Additionally, each transition in the system has a *transition label* associated with it:

$$\mathcal{M}^\rightarrow = \{s \rightarrow t \mid (s, t, e) \in \mathcal{T}, \text{for some } e \text{ in } MTE\}.$$

**Definition:** The valid Modechart *transition expressions* are defined according to the following BNF grammar, where $MTE$ is the initial symbol:

$$\begin{array}{rcl} MTE & ::= & TimingC \mid TrigC \\ TimingC & ::= & (\ lb\ ,\ ub\ ) \\ TrigC & ::= & Event\ \wedge\ TrigC \mid Event \\ Event & ::= & e \mid \overline{e} \mid \{\ MSet\ \} \mid m^+ \mid m^- \mid m \rightarrow n \\ MSet & ::= & m^\perp\ ,\ MSet \mid m^\perp \end{array}$$

Where $lb, ub \in \mathbb{N}$ are called the *lower bound* and *upper bound* of the transition, $e$ denotes the presence of external event $e$ at any time instant, $\overline{e}$ denotes the absence of external event $e$ at any time instant and $m, n \in \mathcal{M}$ are modes of the specification. The expression $m \rightarrow n$ denotes a transition being taken from $m$ to $n$.

Expressions whose form is $(lb, ub)$ are referred to as *timing* transition expressions while the rest of the expressions are referred to as *triggering* transition expressions.

**Notation:** For convenience, we denote the set of timing expressions as $E_{ti}$ and the set of triggering expressions as $E_{tr}$. We extend the set of triggering expressions as follows. Two transitions with identical source and destination modes with triggering expressions $e$ and $e'$ are merged into one transition with an expression $e \vee e'$ and we equally call it a triggering expression. We denote the set of possible input events in one time instant or input language with $\Sigma = 2^{\mathcal{E}}$ and the set of possible outputs or output language with $\Sigma_o = 2^{\mathcal{L}}$. We denote their *closure* or *Kleene star* with $\Sigma^*$ and $\Sigma_o^*$ respectively.

**Definition:** A mode $m$ is *atomic* iff $\forall n \in \mathcal{M} :: n \not\sqsubset m$. A mode $m$ is the *root* mode of $\mathcal{M}$ iff $\forall n \in \mathcal{M} :: n \sqsubseteq m$. The function *children*: $\mathcal{M} \to 2^{\mathcal{M}}$ is defined as follows:

$$children(m) = \{\ n \mid n \sqsubset m \wedge \ \nexists n' :: \ n \sqsubset n' \sqsubset m\ \}.$$

If $n \in children(m)$, we say that $n$ is an *immediate child* of mode $m$, and $m$ is the *immediate parent* of $n$. In other words, $\sqsubset$ captures the tree of all children (or all parents) of a mode, but the function *children* allows us to refer to the immediate relatives of modes.

**Definition:** The *least upper bound* of $m, n \in \mathcal{M}$ (with respect to the $\sqsubset$ relation), is defined as follows:

$$lub(m,n) = \{\ l \mid m \sqsubset l \ \wedge \ n \sqsubset l \wedge \ \nexists l' :: \ m \sqsubset l' \ \wedge \ n \sqsubset l' \ \wedge \ l' \sqsubset l\ \}.$$

**Definition:** Two modes $m, n \in \mathcal{M}$ are *concurrent,* written $m \parallel n$, iff $m \not\sqsubset n \ \wedge \ n \not\sqsubset m \ \wedge \ lub(m,n) \in P$. Two modes $m, n \in \mathcal{M}$ are *sequential,* written $m \sim n$ iff $m \not\sqsubset n \ \wedge \ n \not\sqsubset m \ \wedge \ lub(m,n) \in S$.

**Lemma 1** *Let $m, n \in \mathcal{M}$ be any two modes, $m \neq n$. Then the following holds:*

$$m \not\sqsubset n \ \ \wedge \ \ n \not\sqsubset m \quad \Rightarrow \quad m \sim n \equiv \neg(m \parallel n)$$

**Proof:** The proof is trivial, since the *lub* of a mode can only be either serial or parallel. ■

**Definition:** (Syntactic correctness of a modechart.) A *proper* or *syntactically correct* modechart is a modechart for which the following properties hold:

- $\forall s \in S :: \exists m \in children(s) :: initial(s) = m$. Every serial mode has one initial mode[1].
- $\forall (s,t,e) \in \mathcal{T} :: s \sim t$. Transitions can only take place between sequential modes.

This ends the definition of the syntax of Modechart and the notation that will be used in the remainder of the paper. The next section concentrates on the semantics of the language.

# 4 Operational Semantics for Deterministic Modechart Specifications

In order to define the mathematical semantics of Modechart specifications, we need to formalize a few concepts related to the external evaluation of their behavior as well as to the internal transitions taken by the system during instantaneous reactions. The latter need not be observable by external agents, but are essential for the understanding and design of Modechart specifications.

We accomplish this by defining a two-level Kripke structure (in Plotkin's structural operational semantics style [Plo81]). The top level transition system, $\langle \Gamma, \mapsto \rangle$ takes care of the aspects of the system that interface to the external environment and take time to carry out. At this level, the state of the system is defined as the elements of the carrying set, $\Gamma$, and are the possible states that the system can be in. Transitions in the system take a finite

[1] This was not required in the original definition of the language. We justify imposing it to get rid of unnecessary burdens, while not constraining the usefulness of the language.

amount of time, specifically one unit of time, or one step. The possible steps at this level are determined by the family of ($\Sigma$-indexed) relations $\stackrel{i}{\longmapsto}$.

The lower level structure, $\langle \Gamma, \rightarrow \rangle$, is subsumed by the top level structure and is used to provide the low level reactions of the system, which are not observable by external agents. This structure takes care of instantaneous reactions to external inputs, where no time is consumed in the process, in accordance with the synchrony hypothesis. The possible reactions are determined by the family of relations $\stackrel{i}{\longrightarrow}$.

We drop the input event label from $\rightarrow$ and $\mapsto$ when not ambiguous and we refer to them as just relations throughout the rest of the development, unless otherwise explicitly referred to as families of relations.

The top level structure is generic for the complete class of Modechart specifications, that is, only the low level structure will need to be redefined once the class of specifications that the semantics models is augmented to include non-deterministic specifications. In the first case, the low level transition system is deterministic and in the second, it is not. In general, the top level structure defines transitions as labeled pairs of the form $\gamma \stackrel{i}{\underset{o}{\longmapsto}} \gamma'$, which means that "if the system is in state $\gamma$, a possible result of external input event $i$ is the output event $o$ and a change in the state to $\gamma'$."

Technically, there are no outputs in Modechart specifications, however we consider the output of the system to be the trace of entered, exited and active modes, over time (i.e. the output labels are the outputs). In an implementation, the actual interaction with the environment is produced by the operations associated with entry and exit events as well as those associated with a mode being active or a transition being taken.

**Definition:** A set of modes $M \subseteq \mathcal{M}$ is *sound* iff

$$\forall m, m' \in M :: m \neq m' \Rightarrow (m \sqsubset m' \ \vee \ m' \sqsubset m \ \vee \ m \parallel m').$$

We assume that sound modes are non-empty in the remainder of the paper.

**Definition:** Let $M \subseteq \mathcal{M}$ be a sound set of modes. $M$ is *maximally sound* iff

$$\forall \ m \in (\mathcal{M} - M) :: \ M \ \cup \ \{m\} \ \text{ is not sound.}$$

**Definition:** Let $M \subseteq \mathcal{M}$ be a sound set of modes. The *closure* of $M$, $M^{\square}$, is constructed with the following rules:

- $M^{\square} \supseteq M$.
- $\forall m' \in M^{\square} :: \forall m \in \mathcal{M} :: m' \sqsubset m \Rightarrow m \in M^{\square}$.
- $\forall s \in M^{\square} \cap S :: (M \cap \mathit{children}(s) = \emptyset) \Rightarrow \mathit{initial}(s) \in M^{\square}$.
- $\forall p \in M^{\square} \cap P :: \mathit{children}(p) \subseteq M^{\square}$.
- No other modes are in $M^{\square}$ than those added by the above rules.

**Lemma 2** *Let $M \subseteq \mathcal{M}$ be a set of modes. Then the following holds:*

$$\exists m, m' \in M :: m \neq m' \ \wedge \ m \sim m' \ \Rightarrow \ M \ \textit{is not sound} .$$

**Lemma 3** *$M^{\square}$ is a unique maximally sound subset of $\mathcal{M}$, for each sound $M \in \mathcal{M}$*

**Proof:** From the definition of $M^{\square}$, we can see that since $M$ is sound, any two different elements in $M^{\square}$ either are related by $\sqsubset$ or are concurrent. Thus $M^{\square}$ is a sound subset of $\mathcal{M}$.

It is also the case that it is maximally sound. We prove it by showing that any addition of a mode $m \in \mathcal{M} - M^{\square}$ yields an unsound subset of modes. We know that

$$m \notin M^{\square} \Rightarrow \exists m' \in \mathcal{M} - M^{\square}, \exists n \in M^{\square} :: m' \in \mathit{children}(n) \ \wedge \ m \sqsubseteq m'.$$

We also know that $n$ must be a serial mode, since otherwise $m'$ would be in $M^{\square}$. In addition, we know that $\exists m'' \in \mathit{children}(n) \cap M^{\square} :: m' \neq m''$ by virtue of the definition of $M^{\square}$.

Now consider the set $N = M^{\square} \cup \{m\}$. From the definition of $M^{\square}$, all ancestors of $m$ must belong to $M^{\square}$, in particular $m'$. Since $m' \sim m''$, then by Lemma 3, $N$ is not sound.

It is also easy to see that it is unique. Assume that is not the case and there are two different closures, $M_1$ and $M_2$, of mode $M$. Assume $\exists m \in (M_1 - M_2)$. Then, since $M_1$ is a closure of $M$, it is the case that the set $M \cup \{m\}$ is sound. This implies that $m$ must belong to $M_2$ as well (by the definition of $M^{\square}$), contradicting the assumption $m$ exists. This can be argued for all elements of the difference $M_2 - M_1$, thus $M_1 = M_2$, i.e. they are in fact one and the same set, contradicting the assumption that the closure is not unique. ■

Maximally sound subsets form an integral part of the state of Modechart specifications. We shall refer to them as *mode configurations*. Each mode present in a mode configuration is said to be *active*.

We define the concept of state in the system at this point, which will be used as the carrying set of the Kripke structures. A state includes a mode configuration, a set of "active" transitions, each with an associated clock reading, and a set of labels, which carries the information of an output event to be constructed along the reaction.

**Notation:** We define the set of "extended" transitions as $\mathcal{T}' = \mathcal{T} \times \mathbb{Z}$, where the integer will denote a "clock reading," associated only to timing transitions.

**Definition:** A *state* of a proper modechart specification is a tuple $\gamma = (M, T, o) \in 2^{\mathcal{M}} \times 2^{\mathcal{T}'} \times \Sigma_o = \Gamma$ with the following properties:

- $M$ is a mode configuration of $\mathcal{M}$.
- $T$ is the set of *active* transitions: $\forall \tau = ((s,t,e),n) \in T, s \in M$. We call $\tau$ an *active* transition. We loosely treat it as a transition, when not ambiguous. When $e$ is a timing expression, we call $n$ the *local clock reading* of $\tau$.
- $o \in \Sigma_o$ is the set of internal events (or output).

**Definition:** A timing transition with clock reading $n$ is said to be *triggered* if $n = 0$. A triggering transition with expression $\epsilon$ is *triggered* in the *context* $R \subseteq \mathcal{E} \cup \mathcal{L}$, written $R \vdash \epsilon$ iff $\epsilon$ is valid, according to the following recursive set of rules:

$$\frac{R \vdash \epsilon \ \wedge \ R \vdash \epsilon'}{R \vdash (\epsilon \ \wedge \ \epsilon')} \qquad \frac{R \vdash \epsilon \ \vee \ R \vdash \epsilon'}{R \vdash (\epsilon \ \vee \ \epsilon')}$$

$$\frac{e \in R}{R \vdash e} \qquad \frac{e \notin R}{R \vdash \overline{e}}$$

$$\frac{m^+ \in R}{R \vdash m^+} \qquad \frac{m^- \in R}{R \vdash m^-}$$

$$\frac{m_0 \to m_1 \in R}{R \vdash m_0 \to m_1} \qquad \frac{\{\, {m_0}^{\perp}, {m_1}^{\perp}, \ldots, {m_n}^{\perp} \,\} \cap R \neq \emptyset}{R \vdash \{\, m_0, m_1, \ldots, m_n \,\}}$$

where $e$ is an external input signal and $m, m_0, \ldots, m_n$ are modes in $\mathcal{M}$.

**Definition:** Let $T$ be a set of active transitions and $M$ a set of modes. Then we define a function $texit : 2^{\mathcal{T}'} \times 2^{\mathcal{M}} \to 2^{\mathcal{T}'}$ as follows:

$$texit(T, M) = \{\, ((s,t,e),n) \in T \mid s \in M \,\}.$$

**Definition:** Let $M$ be a set of modes. Then we define a function $tenter : 2^{\mathcal{M}} \to 2^{\mathcal{T}'}$ as follows:

$$tenter(M) = \{\, ((s,t,e),n) \mid s \in M \}, \quad \text{with} \quad n = \begin{cases} lb & \text{if } e = (lb, ub) \in E_{ti} \\ \text{undefined} & \text{if } e \in E_{tr}. \end{cases}$$

**Definition:** Let $\tau = (s,t,e) \in \mathcal{T}$ be a transition. Then we define a function $mexit : \mathcal{T} \to 2^{\mathcal{M}}$ as follows:

$$mexit(\tau) = \{\ m \mid m \sqsubseteq n \in children(lub(s,t)) \ \wedge \ s \sqsubseteq n\ \}.$$

The function *mexit* defines the "tree" of modes contained or equal to the "source" side of the transition $\tau$.

**Definition:** Let $\tau = (s,t,e) \in \mathcal{T}$ be a transition. Then we define a function $menter : \mathcal{T} \to 2^{\mathcal{M}}$ as follows:

$$menter(\tau) = \{\ m \mid t \sqsubseteq m \sqsubseteq n \in children(lub(s,t))\ \}.$$

The function *menter* defines the "path" of modes in the "target" side of the transition $\tau$.

**Definition:** We define the *reaction* (family of $\Sigma$-indexed) binary relation(s), $\rightarrow \subseteq \Gamma \times \Sigma \times \Gamma$, using two rules for active transitions to be triggered, one for transitions with timing expressions and the other for transitions with triggering expressions. For $i \in \Sigma$,

- Timing transitions:

$$\frac{\tau = ((s,t,e),0) \in T \ \wedge \ (s,t,e) \in E_{ti}}{(M,T,o) \xrightarrow{i} (M',T',o')}$$

- Triggering transitions:

$$\frac{\tau = ((s,t,e),n) \in T \ \wedge \ (s,t,e) \in E_{tr} \ \wedge \ (i \cup o) \vdash e}{(M,T,o) \xrightarrow{i} (M',T',o')}$$

where:

$$\begin{array}{l} M_{aux} = M - mexit(\tau), \\ M_{out} = M \cap mexit(\tau), \\ M' = (M_{aux} \ \cup \ menter(\tau))^{\square}, \\ M_{in} = M' - M_{aux}, \end{array}$$

$$T' = (T - texit(T, mexit(\tau))) \ \cup \ tenter(M_{in}),$$

$$o_{aux} = {M_{in}}^{+} \ \cup \ {M_{out}}^{-} \ \cup \ {M'}^{\perp} \ \cup \ \{s \to t\} \quad \text{and}$$
$$o' = o \ \cup \ o_{aux}.$$

**Definition:** Let $\tau = (s,t,e), \tau' = (s',t',e') \in \mathcal{T}$ be two transitions. Let $l = lub(s,t)$ and $l' = lub(s',t')$. The pair of transitions are said to be *consistent* iff $l \parallel l'$, otherwise, they are said to be *in conflict*. A set of transitions is consistent iff no two distinct transitions in it are in conflict.

At this point, we have defined all the possible states of a transition. A transition is said to be *not active,* when it does not belong in a state, it is *active* otherwise. An active transition can be *triggered* if its transition expression is true in a state. One of the triggered transitions is said to be *taken* when the transition system $\langle \Gamma, \rightarrow \rangle$ selects it among the triggered transitions for a step in the structure.

**Definition:** The *reflexive, transitive closure* of the (family of binary) relation(s) $\xrightarrow{i}$ is defined as follows (for a given input $i$, which is omitted for simplicity):

$$\begin{array}{lll} \gamma \rightarrow^{0} \gamma' & \text{iff} & \gamma = \gamma' \\ \gamma \rightarrow^{n} \gamma' & \text{iff} & \exists \gamma'' :: \ \gamma \rightarrow \gamma'' \ \wedge \ \gamma'' \rightarrow^{n-1} \gamma' \\ \gamma \rightarrow^{*} \gamma' & \text{iff} & \exists n :: \ \gamma \rightarrow^{n} \gamma' \end{array}$$

**Definition:** A *fixpoint* of the reaction (family of binary) relation(s), $\xrightarrow{i}$, with respect of the state $\gamma$ and input event $i$ is $\gamma'$ and is defined as follows (for a given $i$):

$$\gamma \rightarrow^{\triangle} \gamma' \quad \text{iff} \quad \gamma \rightarrow^{*} \gamma' \ \wedge \ \nexists \gamma'' :: \ \gamma' \rightarrow \gamma''.$$

At this point, the low level structure $\langle \Gamma, \rightarrow \rangle$ is fully defined. We shall call the class of Modechart specifications that make this structure deterministic (with respect to the fixpoint relation) the class of deterministic specifications of the language. We now start building the top level structure, based on the low level structure just defined. We start by defining the function *tick*, which performs a "time step" in the state of the system, updating the transitions and resetting the output event after a reaction has taken place. We then define the step relation (or more specifically the family of binary relations) which forms the top level structure that causes the system to evolve over time. We thus associate a transition system $\langle \Gamma, \mapsto \rangle$ to every Modechart specification, which defines its semantics.

**Definition:** We define a function $tick : \Gamma \rightarrow \Gamma$ which given a state, returns the resulting state after one time instant has elapsed:

$$tick(M, T, o) = (M, T', M^{\perp})$$

where

$$T' = \{\ ((s,t,e), n') \mid ((s,t,e), n) \in T\ \}, \quad \text{with} \quad n' = \begin{cases} n-1 & \text{if } e \in E_{ti} \\ \text{undefined} & \text{if } e \in E_{tr}. \end{cases}$$

**Definition:** We define the *step* transition relation $\mapsto \subseteq \Gamma \times \Sigma \times \Sigma_o \times \Gamma$ as follows (for a given $i$):

$$\frac{\gamma \xrightarrow{i}{}^{\triangle}\ (M,T,o)\ \wedge\ \gamma' = tick(M,T,o)}{\gamma \underset{o}{\overset{i}{\longmapsto}} \gamma'}.$$

where $i \in \Sigma$ is a set of input events. We denote the transitive closure of $\mapsto$ with $\overset{*}{\mapsto}$ for sequences of input events. If $\gamma = \gamma'$ and no transitions in $T$ contain a reference to external events, the modechart is said to *halt* at $\gamma$.

**Definition:** Let $i_0 i_1 \ldots i_n \ldots = I \in \Sigma^*, n \geq 0,$ be an input event sequence. Let $\mathcal{S}$ be a modechart with $r$ as root mode. Let $p = \{r\}^{\square}$ and $\gamma_0 = (p,\ tenter(p),\ p^{+}\ \cup\ p^{\perp}\ )$. Then $\mathcal{S}(I) = o_0 o_1 \ldots o_n \ldots = O \in \Sigma_o^*$ is a *corresponding* output sequence for $\mathcal{S}$ and $I$ iff

$$\exists \gamma_1, \ldots, \gamma_n, \ldots \in \Gamma :: \ \gamma_0 \underset{o_0}{\overset{i_0}{\longmapsto}} \gamma_1 \underset{o_1}{\overset{i_1}{\longmapsto}} \gamma_2,\ \ldots\ , \gamma_n \underset{o_n}{\overset{i_n}{\longmapsto}} \gamma_{n+1},\ \ldots.$$

The initial state $\gamma_0$ is called the *start state* of $\mathcal{S}$.

**Definition:** Let $\mathcal{S}$ be a proper modechart. Then the *behavior* of $\mathcal{S}$ is the set $\mathcal{B}(\mathcal{S}) = \{\ \mathcal{S}(I) \mid I \in \Sigma^*\ \}$.

**Definition:** A modechart $\mathcal{S}$ is said to have a *zero-cycle* when $(o \cap o_{aux}) - \mathcal{M}^{\perp} \neq \emptyset$ for some transition $\gamma \xrightarrow{i} \gamma'$ in some derivation of some element of $\mathcal{B}(\mathcal{S})$. Intuitively, a zero-cycle-free modechart is one in which no mode can be entered (nor exited) more than once during one reaction.

**Lemma 4** *(Termination) Let $\mathcal{S}$ a zero-cycle-free modechart with a structure $\langle \Gamma, \mapsto, \rightarrow \rangle$. Let $\gamma \in \Gamma$ be some state of $\mathcal{S}$. Then there exists a $k \geq 0$ such that*

$$\exists \gamma' \in \Gamma \mid \gamma \rightarrow^k \gamma' \qquad \textit{and} \qquad \nexists \gamma'' \in \Gamma \mid \gamma' \rightarrow \gamma''.$$

*In other words, each reaction terminates.*

**Proof:** It is clear that each transition in the system can be taken at most once during one reaction, otherwise at least one mode would be entered more than once, contrary to the assumption the $\mathcal{S}$ does not have zero-cycles. Given than the set of transitions is finite, at most it can be exhausted within one reaction. Given an initial state $\gamma$, there must be a state $\gamma'$, such that $\gamma \rightarrow^* \gamma'$ and no transition is triggered in it. Thus there cannot exist a state $\gamma''$ such that $\gamma' \rightarrow \gamma''$ (for any given input). Hence $\gamma$ is a fixpoint of the relation $\rightarrow$, with initial state $\gamma$. ■

**Lemma 5** *Let $\mathcal{S}$ be a proper, zero-cycle free modechart specification and let $A = \langle \Gamma, \mapsto \rangle$ and $B = \langle \Gamma, \rightarrow^{\triangle} \rangle$. Then the following three statements are equivalent: $\rightarrow^{\triangle}$ is a function, $B$ is deterministic and $A$ is deterministic.*

**Proof:** If the fixpoint relation is a function for all inputs, $B$ must be obviously deterministic. If $B$ is deterministic then $A$ must be deterministic, since the fixpoint relation guarantees the determinism in $A$. From the definition of $A$, it is clear that the only possible element of non-determinism is the fixpoint relation, since the *tick* function does not introduce non-determinism. Therefore, if $A$ is deterministic then $\rightarrow^{\Delta}$ must be a function. ■

**Definition:** Let $\mathcal{S}$ be a proper, zero-cycle free modechart with a transition system $\langle \Gamma, \mapsto \rangle$. $\mathcal{S}$ is said to be *deterministic* iff $\langle \Gamma, \mapsto \rangle$ is deterministic, i.e. iff $\forall \omega \in \Sigma^*$,

$$\left.\begin{array}{l} \gamma_0 \xmapsto[\omega_o]{\omega} \gamma \\ \gamma_0 \xmapsto[\omega'_o]{\omega} \gamma' \end{array}\right\} \quad \Rightarrow \quad \omega_o = \omega'_o \ \wedge \ \gamma = \gamma'$$

where $\gamma_0 \in \Gamma$ is the start state of $\mathcal{S}$, $\gamma, \gamma' \in \Gamma$ and $\omega_o, \omega'_o \in \Sigma_o^*$. Note that by Lemma 5, the low level structure must be deterministic for a deterministic modechart.

**Lemma 6** *(Confluence) Let $\mathcal{S}$ be a deterministic modechart with a transition system $\langle \Gamma, \mapsto \rangle$. Then the fixpoint relation $\rightarrow^{\Delta}$ associated with $\mapsto$ is actually a function: it returns a unique state.*

**Proof:** It is possible that the system is in a reaction with a choice of several triggered transitions to be taken. The operational semantics calls for a choice to be made among them, not necessarily deterministic, and as long as there exists at least one triggered transition in the active transition set, one must be taken.

Since the sequence of output event sets generated by a reaction is non-decreasing, the clock readings of timing transitions are not decreased and the external input signal set is not changed during reactions, no triggered transition can become not triggered within a reaction. This ensures that all triggered transitions are taken within a reaction.

Furthermore, it is clear that the set of triggered transitions at any point in the reaction must be consistent, i.e. for each pair of triggered transitions, their source modes (as well as their target modes) must be parallel. Taking any of the transition must involve a disjoint set of modes with respect to the modes involved with the rest of triggered transitions. The net result of taking all the transitions is a single state, regardless of the order in which they are taken. This state is the result of the $\rightarrow^{\Delta}$ relation, thus it is in fact a deterministic function. From the definition of $\rightarrow^{\Delta}$, for any such state $\gamma$, it is obvious that $\gamma \rightarrow^{\Delta} \gamma$, i.e. it is a fixpoint. ■

## 4.1 Building Deterministic Modechart Specifications

The transition system $\langle \Gamma, \rightarrow \rangle$ defined allows more than one transition to be triggered simultaneously during the reaction. However, not all transitions should be allowed to be active simultaneously. Consistent sets of transitions do not present a problem for the semantics, since they can be taken in any sequence "in parallel," without affecting each other. On the other hand, conflicting transitions can present problems when triggered simultaneously.

Given that two triggered conflicting transitions in one state cannot have their source modes share a common immediate parent (no two immediate children can be active simultaneously by the definitions above), it is clear that their source modes must be at different "levels." Intuitively, there can be at most one transition triggered per mode level in the hierarchy. Obviously, when more than one of these transitions are active, a definite choice needs to be made in the reaction relation as to which one of them take.

As defined, computations involving conflicting transitions could either be deterministic or non-deterministic. However, we take a conservative approach to this problem by introducing the *strong preemption axiom:* "when a set of pairwise conflicting transitions are triggered, the highest level one must be taken first[2]." This axiom makes a choice of giving priority to "higher" transitions over related "lower" ones, which get preempted, once the higher transition is taken. This makes all such situations where a set of conflictive transitions is enabled deterministic, thus broadening the class of useful specifications that can be compiled in a deterministic way.

**Definition:** Let $\tau = ((s, t, e), n)$ be a triggered transition and let $T$ be a set of triggered transitions. We define the predicate *preempted* with domain $\mathcal{T}' \times 2^{\mathcal{T}'} \times \mathcal{L} \cup \mathcal{E}$ as follows:

$$\mathit{preempted}(\tau, T, o) = \ \tau \in T \ \wedge \ \exists \tau' = ((s', t', e'), n') \in T :: \ \mathit{triggered}(\tau', o) \ \wedge \ s \sqsubset s' \ \wedge \ \tau \text{ and } \tau' \text{ are in conflict.}$$

where

$$\mathit{triggered}(\tau', o) = \left\{ \begin{array}{ll} n' = 0 & \text{for } \tau' \in \mathcal{E}_{tr} \\ o \models e' & \text{for } \tau' \in \mathcal{E}_{ti} \end{array} \right.$$

[2] This is one of the options offered by the RTL semantics, but harder to implement there.

The strong preemption axiom then is formally introduced in the semantics. This is done by adding the clause $\neg preempted(\tau, T)$ with a conjunction to the antecedent of the two rules of the reaction relation, $\rightarrow$, to prevent transitions that are preempted by "higher level" transitions from being taken.

# 5 Operational Semantics for Non-Deterministic Modechart Specifications

We now proceed to extend the semantics just defined to capture the full set of non-deterministic specifications. In this section, we assume the definitions in the previous sections hold, only the new definitions presented redefine the previous structure. We generalize the definition of timing transitions and redefine the $\rightarrow$ relation to handle these extended expressions as well as the rest of expressions handled before. We then proceed to redefine the transitive closure of $\rightarrow$ and define the fixpoint of it as a non-deterministic function.

**Notation:** We make the following abbreviation: $\Pi = \{ 0, 1, 2 \}$. We introduce the symbol $\infty$ to denote a "large" positive integer, and define that $\forall n \in \mathbb{N} :: n < \infty \ \wedge \ \infty - n = \infty$. We use $\lambda$ to denote the empty string.

**Definition:** A *non-deterministic* timing transition is a timing transition such that $lb < ub$. Where $lb \in \mathbb{N}$ and $ub \in \mathbb{N} \cup \{ \infty \}$. A timing transition is *triggered* if its clock reading is at most zero.

The intuitive meaning of these transitions is that they may be taken at least $lb$ time units after becoming active or at most $ub$ time units after becoming active, at which time they must be taken (unless some other transition $\tau$ is taken and $\tau$ de-activates the original transition). To introduce this behavior into the formal semantics defined so far with minimal disruption, the semantics now allows the clock reading to become negative. The idea is that when a non-deterministic timing transition is initialized, the value of the clock reading is set to the lower bound and is decreased by one each time instant (as before). While its value is greater than zero, the transition relation does not allow the transition to be taken. When its value becomes zero, it may be taken non-deterministically, according to the reaction relation (as defined below). If it is not taken, its value keeps on decreasing to negative values each step. However, if by the time it reaches the value $(lb - ub)$ it has not been taken (i.e. its "window" is exhausted), the system must take it (if no other choice is available, as defined in the reaction transition).

Therefore, for non-deterministic specifications, the system has the choice of taking a transition if it has been active for at least a period of time equal to the lower bound and at most the upper bound. Furthermore, as it is characterized below, the definition of non-deterministic specifications does not exclude the system from the choice of taking a transition among any number of triggered transitions. Yet, if at least one triggering transition or one exhausted timing transition are triggered, one must be taken.

**Definition:** We define the labeled reaction relation, $\rightarrow \subseteq \Gamma \times \Sigma \times \Pi \times \Gamma$, as follows for an input event $i$:

0. Non-deterministic timing transitions:

$$\frac{((s, t, (lb, ub)), n) \in T \quad \wedge \quad n \leq 0 \quad \wedge \quad n > lb - ub}{(M, T, o) \xrightarrow{i,0} (M', T', o')}$$

1. Triggering transitions:

$$\frac{((s, t, e), n) \in T \quad \wedge \quad (i \cup o) \vdash e}{(M, T, o) \xrightarrow{i,1} (M', T', o')}$$

2. Exhausted timing transitions:

$$\frac{((s, t, (lb, ub)), n) \in T \quad \wedge \quad n = lb - ub}{(M, T, o) \xrightarrow{i,2} (M', T', o')}$$

where $M', T',$ and $o'$ have the same definitions as in the deterministic case.

**Definition:** The *transitive closure* of the relation $\xrightarrow{i}$ is defined as follows (for an input event $i$, which we explicitly show):

$$\gamma \xrightarrow{i,\lambda} \gamma' \qquad \text{iff} \qquad \gamma = \gamma'$$
$$\gamma \xrightarrow{i,u\omega} \gamma' \qquad \text{iff} \qquad \exists \gamma'', \exists u \in \Pi, \exists \omega \in \Pi^* :: \gamma \xrightarrow{i,u} \gamma'' \ \wedge \ \gamma'' \xrightarrow{i,\omega} \gamma'.$$

**Definition:** The *fixpoint* of the relation $\xrightarrow{i}$ with respect of the state $\gamma$ is $\gamma'$ and is defined as follows:

$$\gamma \xrightarrow{i}{}^{\triangle} \gamma' \quad \text{iff} \quad \exists \omega \in \Pi^* :: \gamma \xrightarrow{i,\omega} \gamma' \ \wedge \ \nexists u \in \{1,2\}, \exists \gamma'' :: \gamma' \xrightarrow{i,u} \gamma''.$$

**Definition:** Let $\mathcal{S}$ be a proper modechart. Then the *set of computations* corresponding to an input event sequence $I \in \Sigma^*$ is the set $\mathcal{S}'(I) = \{ O \mid O = \mathcal{S}(I) \}$.

**Definition:** Let $\mathcal{S}$ be a proper modechart. Then the *behavior* of $\mathcal{S}$ is the set:

$$\mathcal{B}(\mathcal{S}) = \bigcup_{I \in \Sigma^*} \mathcal{S}'(I).$$

**Definition:** A proper, zero-cycle-free modechart with a transition system $\langle \Gamma, \mapsto \rangle$ as defined above is said to be a *non-deterministic* Modechart specification.

# 6 Special Cases

In the past, a few semantic areas from the original definition of the Modechart language have been identified as being particularly difficult or problematic [Stu95]. These problems stem from the inherent power of RTL. The class of computations allowed by the axioms and formulas derived from some specifications can sometimes be larger than the intended intuitive semantics of said specifications, with sometimes obscure or non-intuitive behavior being allowed. We focus now on how the operational semantics addresses those issues.

A common semantic error in synchronous systems is related to the instantaneous nature of the reactions. Given that a reaction is considered to take place in zero time units, a cycle in the reaction could, in principle, produce an unbounded number of events. This phenomenon is defined in the context of Modechart as a zero-cycle. It is also called an instantaneous loop or zero loop in the context of other synchronous languages. The definition of zero-cycles in this semantics provides a compile-time checking procedure for ruling out those specifications with zero cycles (Section 4).

A second common semantic problem is the problem of preservation of causality in specifications (called non-linearizability in the Modechart literature, and causality errors in others). This problem arises in the axiomatic semantics due to the power of the axiomatic specifications. Computations which are not causally coherent can still satisfy the axioms formulated by a specification. In our semantics, this problem is solved by only allowing the system to react to events that have taken place. Transitions can only be triggered within the reaction relation when their expressions are true in the context of the current reaction input and accumulated output.

Other synchronous languages such as ESTEREL [BG92] are more aggressive in allowing the compilation procedures to generate transitions based on the impossibility of certain "future" conditions to falsify or make expressions true (e.g. negated events) however, this approach requires a (sometimes costly) "look-ahead" in the compilation. Allowing these expressions also leads to frequent obscure causality errors (this problem is enhanced by the presence of compile-time non-deterministic tests, which cause the tree of possible "future" computations within a reaction to grow).

The problem of simultaneous conflicting exits (when several conflicting transitions are triggered simultaneously) is taken care of in the semantics by way of the strong preemption axiom. The choice in the semantics is done by assigning relatively higher priority to "higher" transitions in the hierarchy (Section 4). This converts a class of non-deterministic specifications to a class of (more useful in practice) deterministic ones by establishing a priority among conflicting transitions.

The problem of implicit exit arises when a mode is restarted (exited and entered) in the same reaction and some of its *new* children are killed instead of *old* children being killed. This situation arises due to the lack of some mechanism to reflect the concept of "instances" of children in the RTL semantics. This problem does not arise in the semantics here defined because of the causal and cumulative nature of the reaction relation.

# 7 Compatibility of the Semantics

Defining an operational semantics for the Modechart specification language provides a framework for a practical implementation of the language. The definition introduced is now validated against the axiomatic formulation of the

semantics presented in the original definition using RTL (Real-Time Logic [JM86]).

## 7.1 An overview of the RTL semantics

The semantics in RTL essentially define, for every specification, a set of assertions in the logic of RTL. The behavior of the specification is the set of solutions of the conjunction of the set of assertions, for each possible input to the system. We now present succinctly how to derive the set of assertions from a given specification. Further details about the nature of the formulas displayed here are in [JM94].

0. For every (unique) triggering transition of the form $M \to N$ with triggering condition $C$ we add the formula:

$$\forall t :: M(t,t) \ \wedge \ C \Rightarrow \exists j :: @(M \to N, j) = t$$

1. For every (unique) timing transition of the form $M \to N$ with timing condition $(r, d)$ we add the formula:

$$\forall t :: M[t,t) \Rightarrow \exists t', \ \exists j :: @(M \to N, j) = t' \ \wedge \ B$$

   where $B = (t + r \leq t') \ \wedge \ (t' \leq t + d)$.

2. Let $e_1, e_2$ denote two mode transition expressions, a *mutual exclusion* constraint is a formula of the form:

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

   Every pair of transitions explicitly exiting a mode are subject to a mutual exclusion constraint.

3. If $M$ is a serial mode (with $M_i \in \mathit{children}(M)$) and the system exits $M$ at time $t$ then we add the following assertion:

$$\forall t :: M(t,t] \Rightarrow \bigwedge_{i=1}^{n} (M_i(t,t) \Rightarrow M_i(t,t]).$$

4. If $M$ is a parallel mode (with $M_i \in \mathit{children}(M)$) and the system exits $M$ at time $t$ then we add the following assertion:

$$\forall t :: M(t,t] \Rightarrow \bigwedge_{i=1}^{n} M_i(t,t].$$

5. Transition assertions for nested modes at level $M$ (where mode $M$ is serial). For each mode $M' \in \mathit{children}(M)$ the following formulas are added:

   - $\forall t :: M'(t,t) \Rightarrow \bigwedge_{i=1}^{n} (C_i \Rightarrow (T_i \ \vee \ M'(t,t]))$,
   - $\forall t :: M'[t,t) \Rightarrow \bigwedge_{i=1}^{m} [(T_{n+i} \ \wedge \ B_i) \ \vee \ (\exists t' :: M'[t,t'] \ \wedge \ t' \leq t + d_i)]$,
   - $\bigwedge_{i=1}^{n+m} T_i \Rightarrow R_i$,
   - $\forall t :: M'(t,t) \Rightarrow \{$ Level $M'$ formulas $\}$.

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

6. Transition assertions for nested modes at level $M$ (where mode $M$ is parallel). The following formulas are added:

   - $\forall t :: \bigwedge_{i=1}^{n} (C_i \Rightarrow (T_i \ \vee \ M(t,t]))$,
   - $\forall t :: M[t,t) \Rightarrow \bigwedge_{i=1}^{m} [(T_{n+i} \ \wedge \ B_i) \ \vee \ (\exists t' :: M[t,t'] \ \wedge \ t' \leq t + d_i)]$,
   - $\bigwedge_{i=1}^{n+m} T_i \Rightarrow R_i$,

and for each mode $M' \in children(M)$ the following formula is added:

$$\forall t :: M'(t,t) \Rightarrow \{ \text{ Level } M' \text{ formulas } \}.$$

**Definition:** Let $\mathcal{S}$ be a modechart. Then $\mathcal{RTL}(\mathcal{S})$ is the set of RTL assertions associated with $\mathcal{S}$ according to the semantics defined above. Given an input event set sequence $I = i_1, i_2, \ldots$, an output event set sequence $O = o_1, o_2, \ldots$ is a *computation* of S iff

$$\forall t :: @(i_t,t) \ \wedge \ @(o_t,t) \ \wedge \ \mathcal{RTL}(\mathcal{S}) \quad \text{is satisfiable,}$$

where $@(i_t,t)$ and $@(o_t,t)$ (loosely) denote that all events in the input and output event sets at position $t$ occur precisely at time $t$.

The semantics is completed by adding a clause to restrict the computations to consider, namely, the *linearizability axiom.* This axiom rules out all computations on which a total order relation capturing a causal relationship among events in a computation cannot be established. These computations are called *linearizable computations.* The order relation enforces a causality relationship and also eliminates the need for a series of auxiliary axioms (the accountability assertions) that were introduced in the original semantics to rule out computations with spontaneous generation of events. The following two theorems establish the equivalence of the two semantics.

**Theorem 1** *Let $\mathcal{S}$ be a modechart. Let $I \in \Sigma^*$ be an input set sequence and $O \in \Sigma_o^*$ be the corresponding output sequence for $I$. Then $O$ can be considered as a linearizable computation of $\mathcal{RTL}(\mathcal{S})$.*

**Theorem 2** *Let $\mathcal{S}$ be a modechart. Let $I \in \Sigma^*$ be an input set sequence and let $O$ be a linerizable computation of $\mathcal{RTL}(\mathcal{S})$. Then $O$ is the corresponding output sequence for input $I$ under the operational semantics definition.*

**Proof:** The premise is that $O$ is a linearizable computation, that is, it satisfies all axioms in $\mathcal{RTL}(\mathcal{S})$, with the output $o_t$ and input $i_t$ corresponding to time $t$. We will construct an output sequence $O' \in \Sigma_o^*$ satisfying the theorem. The assertions corresponding to the system entering the root mode in the first instant are formed by applying either rule 5 or 6 to the root mode (depending on it being serial or parallel). These rules in turn prompt the entry of some or all of their children recursively.

The structure of the formulas in rules 5 and 6 capture the hierarchy of $S$. That allows for the formulas of the children of the root mode to be (or not be) true properly at time 0: all children of parallel modes are entered and at most one of the children of a serial mode is entered. Furthermore, since $O$ is linearizable, there exists an order relation between all those entry events. This is precisely the order in which the closure operation is calculated in the operational semantics. This scenario described is the only difference between the computation at $t = 0$ and the rest of the computation.

If any transition belonging to the modes just entered is triggered and it is taken (rules 0, 1 and 2, and ignoring situations of self reinstatement such as self loops), it is taken at this point.

The process of constructing $O'$ is repeated just as described above for every time instant, creating each reaction's output. ■

# 8 Compiling Modechart Specifications

Once we have defined the operational behavior of the Modechart specification language, we introduce a compiler algorithm for deterministic modechart specifications that produces a finite state machine which is guaranteed to implement the semantics of the specification. This compiler algorithm is essentially an exhaustive symbolic execution of the two-level structure of the semantics, for all possible inputs. Reactions are compiled into states of a finite state machine and time-taking transitions are the directed edges among the states.

Section 8.2 describes an extension to the compiler to allow for a certain class of non-deterministic specifications which is useful in practice and still allows the compilation of the specification.

## 8.1 Deterministic Specification Compilation

The deterministic operational semantics presented in Section 4 introduces a relation that captures the essence of the execution of a deterministic Modechart, thus it yields itself naturally into the implementation of a compiler, presented below. In this presentation we do not keep track of the transitions for simplicity in the presentation. It is straightforward to add mechanisms to track them.

**Algorithm: CompileMC**
**Input:** MC (deterministic Modechart Spec.), E (input set);
**Output:** FSM (a Finite State Machine);

```
m := closure(root(MC));
% states is a list
% and current_state is a pointer in the list
states := insert (NIL, ⟨m, tenter(m)⟩);
current_state := head(states);
done := FALSE;
while (not done) do
    new_states := ∅;
    for each i in 2^E do
        new_states ∪ = fixpoint(current_state, i);
    end for
    if (new_states − states = ∅) and (current_state = last(states)) then
        done := TRUE;
    else
        states ∪= new_states;
        current_state := next_state(current_state, states);
    end if
end while
output_fsm(states);
```

**end Algorithm**

We next define the functions **fixpoint** and **next_state**. The function **fixpoint** computes the fixpoint of a reaction, returning the final state after all triggered transitions have been taken with the given input. The function **next_state** picks the next unexplored state out of the state list (the one right after the one from which **current_state** was produced in the state list) and "applies" a time instant to it, producing the new current state to explore.

**Function: fixpoint**
**Input:** s (state), i (input signal event)
**Output:** f (state)

```
f := s;
while (any_triggered(f)) do
    f := take(f, any_triggered(f));
end while
return f;
```

**end Function**

The function **take** picks one transition out of all the ones triggered. If any set of pairwise conflicting transitions is triggered, then **take** must pick the transition with highest priority, or otherwise any one of the other consistent transitions that may be left.

It is easy to see that the size of the state space is exponential on the size of the input signal set, the modes in the specification and linear on the timing transition expressions. However, the actual state space and input signal set, as well as the timing expressions, are user specifications, thus it is reasonable to assume that the user will be careful in providing an input to the compiler such that no state space blowup occurs.

Obviously, it is required that the input specification be deterministic so that the **fixpoint** function above is deterministic as well (Lemma 6). Assuming that, the function **fixpoint** is computed in $O(TN \log N)$ time, where $T$ is the size of the transition set, $N$ is the combined size of the state space (inputs times mode combinations).

We assume the set operations are performed with efficient data structures in $O(N \log N)$, including computing the closure of a set of modes, which is linear with the size of the mode set (function **taken**). Assuming an efficient implementation of lists, the **next_state** function has a time complexity of $O(T)$. Considering the above and the fact that $T \ll N$, the time complexity of the algorithm above is approximately $O(N^2 \log N)$.

### 8.2 Non-Deterministic Specification Compilation

In order to provide support for a larger and more practical class of specifications, we now add a particular type of non-deterministic specifications, the run-time deterministic class of specifications. These are introduced to model the concept of instantaneous non-deterministic decisions within a modechart. They model very practical situations such as loop counters, decisions over contents or types of messages received or, in general control flow *instantaneous* decisions based on data items whose value is determined out of the scope of the modechart being implemented and only available at run-time. It is the specifier's job to decide at which level these transitions are going to be used. Some may decide to model some of this behavior with external events and keep control modes running in parallel to the specification to constrain the possible situations and model the environment more closely, while others may decide to leave more of the data processing as external behavior.

Each immediate child of a serial mode is allowed to have a number of transitions whose expression involve instantaneous evaluations of predicates over *external variables*. These external variables are typed variables whose value can be changed at any time by the environment. The predicates must be exhaustive and mutually exclusive. Intuitively, all the predicates must cover all the range of the variables they test and only one of the predicates must be true at the moment of checking them. We assume they are side-effect free and that the environment remains invariant during the (instantaneous interval of) time in which they are invoked.

For verification purposes, these transitions are replaced by timing transitions of the form $(0, 0)$ before any property of the specification is attempted to be proved. For implementation, the functions **take** and **any_triggered** need to be redefined to generate calls to the instantaneous predicates in the transitions. It is clear that as long as the predicates take an amount of time significantly smaller than the synchronous period, the system will implement the right behavior. The transition expression syntax, the definition of triggered transitions and the reaction relation are augmented accordingly (from the deterministic semantics) to support this class of specifications very easily.

## 9 Discussion and Future Work

This paper has introduced a formal operational semantics for the class of real-time specifications expressed in the Modechart formalism. This semantics provides the framework for defining a compiler for a subclass of the Modechart language. The original semantics of the Modechart language was described axiomatically in terms of Real-Time Logic. Introducing this semantics provides a more intuitive reference to the language as well as an implementation guide for it. The compiler presented is an efficient implementation of the language. It is based on an exhaustive symbolic execution of the specification within the synchronous model, which compiles communication and concurrency away, providing a simple finite state machine.

This semantics and compiler are a basis for future work on native code generation for Modechart specifications in a number of parallel and distributed architectures within the SARTOR project. Other areas of future work include support of the complete set of transition expressions originally defined in the language, support for code generation in a variety of general purpose programming languages, more efficient implementation of timing specifications and support for a distributed correctness-preserving implementation of specifications.

## References


[BG92] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: design, semantics, implementation. *Science of Computer Programming*, 19:87–152, 1992.

[JM86] F. Jahanian and A.K. Mok. "Safety Analysis of Timing Properties of Real-Time Systems". *IEEE Transactions on Software Engineering*, 12(9):890–904, September 1986.

[JM94] F. Jahanian and A. Mok. Modechart: a specification language for real-time systems. *IEEE Transactions on Software Engineering*, pages 933–947, December 1994.

[Mok85] A. Mok. SARTOR –a design environment for real-time systems. In *Proceedings 9th IEEE COMPSAC*, 1985.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical report, Aarhus University, 1981.

[Puc95] C. Puchol. An operational semantics for Modechart specifications. Technical report, Computer Sciences Department, The University of Texas at Austin, 1995. http://www.cs.utexas.edu/users/cpg/OP-SEM.ps.Z.

[Stu95] Douglas A. Stuart. *"Specification and Analysis of Real-Time Systems"*. PhD thesis, The University of Texas at Austin, Department of Computer Sciences, 1995.