# Statecharts: From Visual Syntax to Model–Theoretic Semantics

Gerald Lüttgen[1] Michael Mendler[1]

[1]Department of Computer Science, The University of Sheffield, 211 Portobello Street, Sheffield S1 4DP, U.K.
{g.luettgen, m.mendler}@dcs.shef.ac.uk

## Abstract

*This paper presents a novel model–theoretic account of Harel, Pnueli and Shalev's original step semantics of the visual specification language Statecharts. The graphical syntax of a Statechart is read, directly and structurally, as a formula in propositional logic. This proposition captures all the logical constraints imposed by the diagram on the Statechart's semantics, i.e., the possible sets of transitions that can be taken together to perform a valid Statecharts step, and their effects on Statecharts configurations. The paper's main result shows that the correct semantics is uniquely described by the intuitionistic interpretation of Statecharts formulas, whereas the naive classical interpretation is insufficient. The advocated intuitionistic approach not only gives a correct, clear and direct logical account of Statecharts' semantics, but also permits the integration of Statecharts with formal validation tools, such as theorem provers.*

***Keywords:*** *Statecharts, model–theoretic semantics, intuitionistic logic*

## 1 Introduction

*Statecharts* is a visual formalism, introduced by Harel in the mid Eighties [4], for specifying the behavior of *reactive systems*, i.e., concurrent systems that are characterized by their ongoing interaction with their environment. It extends *finite–state machines* with concepts of *hierarchy* ("OR states"), so that one may speak of a state as having sub–states, *concurrency* ("AND states"), thereby allowing the definition of systems having simultaneously active subsystems, and *priority*, so that one may express that certain system events have precedence over others. Despite Statecharts' popularity with system engineers and an increasing wealth of related tools [7], its semantics remains an active and quite controversially debated field of research [5, 13, 14, 15, 18, 19], which led to the emergence of many Statecharts dialects [22]. The original Statecharts semantics, as conceived by Harel, Pnueli and Shalev [6, 18], is a two–level step semantics which reflects the behavior of globally synchronous, locally asynchronous systems in an intuitive way: a Statechart may react to an *event* by engaging in an enabled transition, thereby performing a so–called *micro step*, which may generate new events, as described by the transition's *action*, that may in turn *trigger* new transitions while disabling others. When this chain reaction comes to a halt, one execution step, a so–called *macro step*, is complete. The semantic principle underlying the maximality of a macro step is often referred to as *synchrony hypothesis* [2] and the one underlying the chain–reaction–style triggering of transitions as *causality*. In a seminal paper published in the early Nineties [18], Pnueli and Shalev formalized this step semantics obeying the synchrony hypothesis and causality in an operational and a declarative style, and showed both variants to coincide. Despite this mathematical elegance, their semantic accounts have not been adopted in tool implementations, mainly because of the complexity in the construction of Statecharts steps and the lack of compositionality.

This paper proposes, for the first time in the literature, a *model–theoretic approach* to defining Statecharts semantics. Related but quite different approaches have previously been investigated for *Modecharts* [9], a graphical language for specifying *real–time* systems, and *Esterel* [2], which is a *textual* language for modeling reactive systems. The idea suggested in this paper is to read the diagrammatic elements of a given Statechart, together with the implicit static semantics, as a formula in propositional logic. Intuitively, events, state names and transition names are taken to be atomic propositions and transition labels are logically interpreted as "*trigger implies action*," where a trigger is a conjunction of events and negated events describing under which condition a transition fires, and where an action corresponds to the conjunction of the events generated by firing the transition under consideration. In addition, the static semantics includes the structure of a Statechart and consists of rules such as the following: (i) an OR state is active if and only if one of its sub–states is active, (ii) an AND state is ac-

tive if and only if all of its sub–states are active, (iii) only mutually concurrent transitions may be included in the same macro step. Given a Statecharts formula, its models are expected to indicate which transitions might be taken together to perform a valid macro step, in the sense of Harel, Pnueli and Shalev. Our results, however, show that the straightforward approach of interpreting Statecharts formulas in classical logic, which is tempting to system engineers unaware of the subtleties of Statecharts' step semantics, does not work since some classical models give rise to invalid macro steps. The reason is that the two–valued classical logic is insufficient for reflecting Statecharts' principle of causality, where events cannot only be present or absent within a system step, but either causally present, non–causally present, or absent. This suggests interpreting Statecharts formulas intuitionistically, rather than classically. Indeed, within the intuitionistic setting we are able to characterize exactly those classical models of Statecharts formulas, which correspond to valid Statecharts macro steps.

The strength of our model–theoretic approach lies in the simplicity of structurally inferring the original semantics of a Statechart from its hierarchical diagrammatic notation. In addition to providing insights in the semantics of Statecharts, the results of this paper lend themselves for integrating the Statecharts formalism with declarative languages, such as Prolog, and formal verification tools, such as the theorem provers HOL [3], Isabelle [17] and PVS [16]. Indeed, our approach corresponds to a shallow encoding of Statecharts and, thus, for the manipulation of individual Statecharts designs, providing direct access to the algorithmic technology available in a theorem prover. In particular, automated tactics at the propositional level can be used, with the proviso that they preserve the underlying intuitionistic semantics. This direction for future work is made even more attractive by the fact that the intuitionistic framework also permits the adaptation of compositionality and full–abstractness results for Statecharts [11]; these properties are particularly important for facilitating component–based system design and validation within the Statecharts specification formalism.

## 2 Statecharts Formulas

In this section we show how a formula in propositional logic can be derived from the visual description of a given Statechart. The model–theoretic framework for interpreting such a Statecharts formula is provided in the next section, in such a way that the models of the formula directly correspond to the executions of those transitions which can fire together to perform a macro step of the Statechart under consideration.

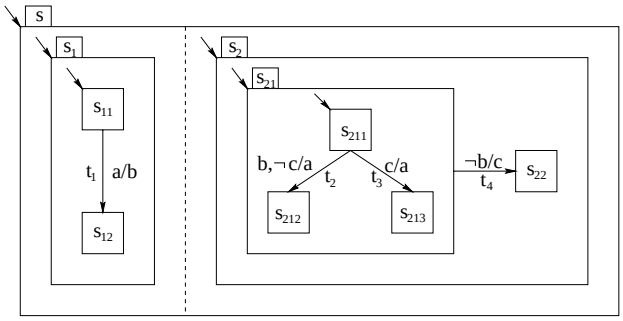


Figure 1: Example Statechart

Instead of presenting a formal account of Statecharts' visual syntax, we illustrate its diagrammatic notation by means of an example Statechart depicted in Figure 1. The state hierarchy of this Statechart is reflected in the nesting of states: The root $s$ is an *AND–state*, in signs $s \in AND$, which has the two *OR–states* $s_1$ and $s_2$ as children, in signs $s_1, s_2 \in OR$ and $child(s) = \{s_1, s_2\}$. The OR–state $s_1$ contains in turn the two states $s_{11}$ and $s_{22}$, which are connected via the transition $t_1$ and are referred to as *basic states* since they are not refined further. OR–state $s_2$ is refined by another OR–state $s_{21}$ and the basic state $s_{22}$, which are connected via transition $t_4$. Finally, $s_{21}$ describes a state machine with the three basic states $s_{211}$, $s_{212}$ and $s_{213}$, and the transitions $t_2$ and $t_3$.

From now on let *SC* be an arbitrary but fixed Statechart with a set $\mathbb{S}$ of states, a set $\mathbb{T}$ of transitions, and a set $\mathbb{E}$ of events. The state hierarchy of *SC* leads to an implicit semantic condition describing which states can be active together at any given time during the execution of the Statechart. Intuitively, the root $r$ of a Statechart is always active. An AND–state reflects that its children states are running in *parallel*, i.e., an AND–state is active if and only if *all* of its children are active. An OR–state denotes a *sequential* state machine which is active if and only if exactly *one* of its children states is active. In Statecharts, a configuration, i.e., a set of states, that obeys the above rules is referred to as *valid configuration*. Logically, these rules can be encoded by the formula $\text{CONFIG}(\underline{C}) =_{\text{df}}$

$$\underline{C}(r) \wedge \bigwedge_{s \in \mathbb{S} \cap AND} (\underline{C}(s) \equiv \bigwedge_{s' \in child(s)} \underline{C}(s')) \wedge$$

$$\bigwedge_{s \in \mathbb{S} \cap OR} (\underline{C}(s) \equiv \bigvee_{s' \in child(s)} (\underline{C}(s') \wedge \bigwedge_{s'' \in child(s) \setminus \{s'\}} \neg \underline{C}(s'')))$$

where $\underline{C}$ is a unary predicate which will later be interpreted by a configuration $C$ such that $\underline{C}(s')$ is valid if and only if $s' \in C$ holds. Since configurations are always finite, it is obvious that we could have written CONFIG, and similar the other formulas which we are going to present below, as a formula in propositional logic rather

than in predicate logic; however, we find that the use of predicates improves readability. In our example Statechart, the initial configuration consists of the set of states that are pointed to by the small arrows, namely $C_{init} = \{s, s_1, s_2, s_{11}, s_{21}\}$, which obviously validates the proposition $\text{CONFIG}(\underline{C}_{init})$.

Before introducing the logical encoding of transitions we present the architecture of the Statechart formula MACRO–STEP relative to some fixed Statechart *SC*: $\text{MACRO–STEP}(\underline{C}, \underline{E}, \underline{T}, \underline{A}, \underline{C}') =_{\text{df}}$

$$\text{CONFIG}(\underline{C}) \wedge \text{TRANS}(\underline{C}, \underline{E}, \underline{T}, \underline{A}, \underline{C}') \wedge \text{CONFIG}(\underline{C}')$$

which is parameterized in the predicates $\underline{C}$, $\underline{E}$, $\underline{T}$, $\underline{A}$, and $\underline{C}'$ standing for a configuration $C$, a set of events $E$, a set of transition names $T$, a set of events $A$, and a configuration $C'$, respectively. Intuitively, MACRO–STEP describes the macro steps of *SC*, i.e., the formula is expected to hold if (i) $C$ is a valid configuration of *SC*, (ii) $T$ corresponds to a macro step of *SC* in this configuration when the environment offers the events in $E$, as well as (iii) $A$ is the set of events generated by $T$, and $C'$ is the valid configuration reached when performing the macro step $T$.

We now turn to the logical formalization of a Statechart transition, which we illustrate using transition $t_2$ of our example Statechart in Figure 1. For notational convenience we denote the source state of a Statecharts transition $t$ by $\mathsf{source}(t)$ and its target state by $\mathsf{target}(t)$, i.e., $\mathsf{source}(t_2) = s_{211}$ and $\mathsf{target}(t_2) = s_{212}$. Moreover, a transition label in Statecharts is a pair of event sets, where the first component is referred to as *trigger* and may include *negated events*, and the second component is referred to as *action*. Furthermore, we write $\mathsf{trigger}^+(t)$ for the positive events included in the trigger of $t$ and $\mathsf{trigger}^-(t)$ for the negated ones, as well as $\mathsf{action}(t)$ for the events in the action of $t$. In our example, $\mathsf{trigger}^+(t_2) = \{b\}$, $\mathsf{trigger}^-(t_2) = \{c\}$, and $\mathsf{action}(t_2) = \{a\}$. Then, for a transition $t$ to fire, three conditions must be satisfied; the first two are determined by Statecharts' hierarchical structure, and the last one emerges from the transition's label of the form "*trigger implies action*."

- First, $t$ must exit a state that is active in the current configuration $C$, i.e., $t$ must be *relevant* [18]. Formally, $\text{RELEVANT}(t, \underline{C}) =_{\text{df}} \underline{C}(\mathsf{source}(t))$; in our example, $\text{RELEVANT}(t_2, \underline{C}) = \underline{C}(s_{211})$.

- Second, $t$ can only fire if all other transitions within the potential macro step $T$ are *consistent* with $t$, i.e., they are all located in different sub–states of the smallest enclosed AND–state in which $t$ is contained and, thus, are "orthogonal" or "concurrent" to $t$. For convenience, we refer to Pnueli and Shalev's notion of the set *consistent*$(t)$ which includes all transitions of *SC* that are consistent with $t$; it is defined along the state hierarchy of *SC*. This yields the formula

  $$\text{CONSISTENT}(t, \underline{T}) \quad =_{\text{df}} \bigwedge_{t' \notin \mathsf{consistent}(\{t\})} \neg \underline{T}(t')$$

  which states that transitions inconsistent with $t$ cannot be included in the considered macro step $T$. In our example, $\mathsf{consistent}(\{t_2\}) = \{t_1, t_2\}$, whence $\text{CONSISTENT}(t_2, \underline{T}) = \neg \underline{T}(t_3) \wedge \neg \underline{T}(t_4)$.

- Third, $t$ can only fire if the events in $\mathsf{trigger}^+(t)$, but not the events in $\mathsf{trigger}^-(t)$, are offered by the environment $E$ or are included in the generated event set $A$, i.e., are broadcasted by other transitions firing in the same macro step. This may be written in our logical formalism as $\text{TRIGGERED}(t, \underline{E}, \underline{A}) =_{\text{df}}$

  $$\bigwedge_{e \in \mathsf{trigger}^+(t)} (\underline{E}(e) \vee \underline{A}(e)) \;\wedge \bigwedge_{e \in \mathsf{trigger}^-(t)} (\neg \underline{E}(e) \wedge \neg \underline{A}(e)) \,.$$

  In our example, $\text{TRIGGERED}(t_2, \underline{E}, \underline{A})$ is the formula $(\underline{E}(b) \vee \underline{A}(b)) \wedge (\neg \underline{E}(c) \wedge \neg \underline{A}(c))$.

According to the synchrony hypothesis adopted in Statecharts, a transition $t$ satisfying the abovementioned three conditions *must* fire, thereby implying the following:

- First, firing $t$ generates the events in its action, i.e., $\mathsf{action}(t) \subseteq A$ or, as formula, $\text{ACTION}(t, \underline{A}) =_{\text{df}} \bigwedge_{e \in \mathsf{action}(t)} \underline{A}(e)$. Thus, $\text{ACTION}(t_2, \underline{A}) = \underline{A}(a)$ in our example.

- Second, $t$ must be included in the macro step $T$ under consideration; formally, $\underline{T}(t)$.

- Third, transition $t$ enters and initializes its target state $\mathsf{target}(t)$, which implies that $\mathsf{target}(t)$, as well as all states in the default configuration for the Statechart with root $\mathsf{target}(t)$ must be included in the target configuration $C'$. Formally, $\text{DEFAULT}(t, \underline{C}') =_{\text{df}} \bigwedge_{s \in \mathsf{default}(\mathsf{target}(t))} \underline{C}'(s)$, where $\mathsf{default}(s')$ denotes the default configuration of some state $s'$, as defined by Pnueli and Shalev in [18] along the state hierarchy of $s'$. In our example, $\mathsf{default}(\mathsf{target}(t_2)) = \mathsf{default}(s_{212}) = \{s_{212}\}$ since $s_{212}$ is a basic state, whence $\text{DEFAULT}(t_2, \underline{C}') = \underline{C}'(s_{212})$.

Note that each fired transition explicitly activates the default configuration of its target state. However, we also have to ensure that every state $s$ in the current configuration $C$, which is not deactivated by the firing of any transition in $T$, must also be included in the target configuration $C'$. In this case, we say that $s$ *idles* for $T$ and formally define $\text{IDLE}(\underline{C}, \underline{T}, \underline{C}') =_{\text{df}}$

$$\bigwedge_{s \in \mathbb{S}} (\underline{C}(s) \wedge \bigwedge_{t \in \mathbb{T}} (\underline{T}(t) \supset (s \perp \mathsf{target}(t)))) \supset \underline{C}'(s),$$

where $s_1 \perp s_2$ means as in Pnueli and Shalev's account [18] that state $s_1$ is orthogonal to $s_2$, i.e., both states are located in different sub–states of the smallest AND–state enclosing them. In the IDLE predicate, the sub–formula $\bigwedge_{t\in\mathbb{T}} \underline{T}(t) \supset (s \perp \mathsf{target}(t))$ encodes the condition under which state $s$ must remain idle, namely if the target states of all transitions included in the current step are orthogonal to $s$. We finally define the desired formula for $\textsc{trans}(\underline{C},\underline{E},\underline{T},\underline{A},\underline{C}') =_{\mathrm{df}}$

$$\begin{aligned}\bigwedge_{t\in\mathbb{T}} &(\textsc{relevant}(t,\underline{C}) \wedge \textsc{consistent}(t,\underline{T}) \wedge \\ &\quad \textsc{triggered}(t,\underline{E},\underline{A})) \supset \\ &\quad (\textsc{action}(t,\underline{A}) \wedge \underline{T}(t) \wedge \textsc{default}(t,\underline{C}')) \\ &\wedge\ \textsc{idle}(\underline{C},\underline{T},\underline{C}')\ .\end{aligned}$$

This formula directly reflects our interpretation of transition labels by reading "," in triggers and actions as event conjunction, "¬" in front of events as negation, and "/" as implication. In our example, the conjunct for $t_2$ is

$$\begin{aligned}&(C(s_{211}) \wedge (\neg\underline{T}(t_3) \wedge \neg\underline{T}(t_4)) \wedge \\ &\ ((\underline{E}(b) \vee \underline{A}(b)) \wedge (\neg\underline{E}(c) \wedge \neg\underline{A}(c)))) \supset \\ &(\underline{A}(a) \wedge \underline{T}(t_2) \wedge \underline{C}'(s_{212}))\ .\end{aligned}$$

Summarizing, we have shown how a Statecharts diagram can be intuitively read as and encoded by a formula in predicate logic or, in fact, propositional logic. The resulting Statecharts formula captures all constraints on the state hierarchy and macro–step transition semantics that can be derived structurally from the diagram. Although not shown in our example, our approach can also handle *interlevel transitions*, i.e., transitions crossing the borderlines of states, and can be easily adapted to cope with *state references*, i.e., trigger events of the form $\mathsf{in}_s$, for $s \in \mathbb{S}$, and *implicit priorities* on transitions imposed by the state hierarchy [5]. However, for incorporating the concept of *history states* included in some Statecharts dialects, our work would have to be extended.

# 3 Model–theoretic Semantics

In the following we define and analyze two semantic interpretations of Statecharts formulas, one using classical logic and the other one based on intuitionistic logic. Both rely on structures of the form $M = (C,E,T,A,C')$, where $C,C' \subseteq \mathbb{S}$, $E,A \subseteq \mathbb{E}$ and $T \subseteq \mathbb{T}$. Our goal is to characterize the macro steps of a given Statechart, in the sense of Pnueli and Shalev, as models of the step–construction predicate MACRO–STEP. Accordingly, we call a structure *M Pnueli–Shalev step*, or *PS–step* for

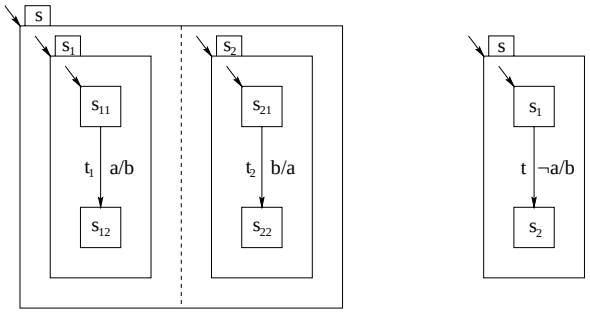


Figure 2: Insufficiency of classical logic

short, if (i) $C$ and $C'$ are valid configurations in the Statechart under consideration, (i) $T$ is a set of transitions resulting from the step–construction algorithm à la Pnueli and Shalev [18] with respect to the source configuration $C$ and the environment $E$, (iii) $A$ is the set of events generated by $T$, and (iv) $C'$ is the target configuration reached when executing the transitions $T$ in $C$. For a structure $M = (C,E,T,A,C')$ we use its components to interpret their corresponding predicates $(\underline{C},\underline{E},\underline{T},\underline{A},\underline{C}')$, i.e., $M$ becomes an *interpretation* of MACRO–STEP.

**Classical interpretation.** As usual, an interpretation $M$ is called a *classical model* of MACRO–STEP, if $M$ validates MACRO–STEP in the usual sense of classical logic. Unfortunately, from a classical point of view, MACRO–STEP is too loose, meaning that not every classical model corresponds to a PS–step. To see this, consider the example Statechart depicted in Figure 2 on the left, in its initial configuration $C_1 =_{\mathrm{df}} \{s,s_1,s_2,s_{11},s_{21}\}$. The configuration that would be obtained by the firing of both transitions $t_1$ and $t_2$ is $C_1' =_{\mathrm{df}} \{s,s_1,s_2,s_{12},s_{22}\}$. Although in the empty environment $E = \{\}$ neither of the two transitions is enabled, the associated transition predicate $\textsc{trans}(\underline{C},\underline{E},\underline{T},\underline{A},\underline{C}')$, which includes the conjuncts

$$\begin{aligned}(\underline{C}(s_{11}) \wedge (\underline{E}(a) \vee \underline{A}(a))) &\supset (\underline{A}(b) \wedge \underline{T}(t_1) \wedge \underline{C}'(s_{12})) \\ (\underline{C}(s_{21}) \wedge (\underline{E}(b) \vee \underline{A}(b))) &\supset (\underline{A}(a) \wedge \underline{T}(t_2) \wedge \underline{C}'(s_{22}))\end{aligned}$$

for transitions $t_1$ and $t_2$, respectively, possesses the classical model $(C_1,\{\},\{t_1,t_2\},\{a,b\},C_1')$. In fact, for every $T \supseteq \{t_1,t_2\}$ and $A \supseteq \{a,b\}$, the structure $(C_1,\{\},T,A,C_1')$ is a classical model of MACRO–STEP for this example. However, none of these is a PS–step. The actual PS–step of the example configuration, in the empty environment, is the empty step $(C_1,\{\},\{\},\{\},C_1)$ which is also a classical model. In general, every PS–step is a classical model but not vice versa. Thus, the question is how to distinguish PS–steps from other, non–desired classical models. The reason why classical models are too loose is that they do not enforce *causality* which is a basic ingredient of

Pnueli and Shalev's step semantics. Causality demands that every transition included in a macro step must be causally triggered by the effects of some other transition in an uninterrupted causal chain that can be traced back to the events $E$ offered by the environment. In the classical step $(C_1,\{\},\{t_1,t_2\},\{a,b\},C_1')$ above, none of the transitions $t_1$ and $t_2$ is forced by such a causal chain; instead, $t_1$ and $t_2$ trigger each other in a cyclic dependency.

Observing that the actual PS–step $(C_1,\{\},\{\},\{\},C_1)$ is the smallest classical solution in the example above, one might hypothesize that we should look for the *minimal* classical models. This, however, does not quite work, either. A counterexample is the Statechart in Figure 2 on the right. In this case the initial configuration is $D_1 =_{\mathrm{df}} \{s,s_1\}$, the target configuration $D_1' =_{\mathrm{df}} \{s,s_2\}$, and we consider again the empty environment. The conjunct for transition $t$ in the transition predicate TRANS reads

$$(\underline{C}(s_1) \wedge (\neg\underline{E}(a) \wedge \neg\underline{A}(a))) \supset (\underline{A}(b) \wedge \underline{T}(t) \wedge \underline{C}'(s_2)).$$

Here, both $(D_1,\{\},\{t\},\{b\},D_1')$ and $(D_1,\{\},\{\},\{a\},D_1)$ are minimal classical models, but only the first one is a valid PS–step. The event $a$ in the second model is consistent with the Statechart's step behavior but is spurious in that it is not generated causally from within the Statechart itself. We might say that $a$ is a non–causal or contingent event which is injected by the environment. Accordingly, it seems that we need at least three truth values to represent the membership relationship for the set $A$: an event $a$ may be (i) absent from $A$, and it may be present in two ways, either (ii) causally forced by the reaction of the Statechart, or (iii) non–causally if it is injected by the environment. An analogue statement is valid for the set $T$. Consequently, the classical principle of the excluded middle, i.e., $\underline{A}(a) \vee \neg\underline{A}(a)$ in our notation, must be given up.

**Intuitionistic interpretation.** The invalidity of the law of the excluded middle for Statecharts semantics motivates intuitionistic logic, which is a refinement of classical logic, as a candidate for our model–theoretic approach. Here, we consider the classical models as final structures $M_n$ of nonempty, strictly increasing sequences $M = (M_1,M_2,\ldots,M_n)$, for $n \in \mathbb{N}$, of classical structures, called *intuitionistic sequence structures*, or sequence structures for short. In this context, "strictly increasing" means that $M_i \subsetneq M_{i+1}$, for all $1 \leq i < n$, where the ordering is defined pointwise. Every such sequence ending in the classical model $M_n$ explicates the internal causality structure of the final structure $M_n$. The idea is that each proper inclusion $M_i \subsetneq M_{i+1}$ corresponds to a non–causal step in the construction of $M_n$, implying that some of the additional elements in $M_{i+1} \setminus M_i$, where "$\setminus$" is again defined pointwise, have been introduced due to some external effect and are not solely causally dependent on $M_i$. In particular, if there is no non–trivial sequence structure ending in $M_n$ other than $M_n$ itself, then all elements contained in the components of $M_n$ must be causally present. This gives the desired criterion for identifying the PS–steps among the classical models. In the following we make this intuition formally precise.

A sequence structure $M$ is an *intuitionistic sequence model* of MACRO–STEP, if each $M_i$ satisfies MACRO–STEP in the intuitionistic sense [21], i.e., when interpreting implication "$\supset$" and negation "$\neg$" as follows: (i) $M_i \models \phi \supset \psi$ if $\forall j \geq i.\ M_j \models \phi$ implies $M_j \models \psi$, and (ii) $M_i \models \neg\phi$ if $\forall j \geq i.\ M_j \not\models \phi$. Observe that in the special case where $M$ is a single structure, we have that $M$ is a sequence model of MACRO–STEP if and only if it is a classical model. This means that the classical models of MACRO–STEP are precisely the final structures of all sequence models of MACRO–STEP. As indicated before we are interested in those classical structures that are not final elements in any nontrivial sequence structure. Formally, a sequence model $M$ of length 1, i.e., a classical model, is called a *response model*, if there does not exist a sequence model $N = (N_1,N_2,\ldots,N_k)$ of length $k > 1$ such that $M = N_k$ and in which the components $E_i$ in all $N_i = (C_i,E_i,T_i,A_i,C_i')$, for $1 \leq i \leq k$, are identical; the condition that the $E_i$ are constant reflects the idea that the *initial* input events are causally present by definition. This refinement of the notion of a classical model paves the way to our main theorem which characterizes PS–steps in terms of response models.

**Theorem 3.1** *Let SC be a Statechart,* MACRO–STEP *be its Statecharts formula, and M be a structure. Then, M is a PS–step in SC if and only if M is a response model of* MACRO–STEP.

For deciding whether a classical model $(C,E,T,A,C')$ is a response model we only need to test the components $T$ and $A$: if $N = (N_1,N_2,\ldots,N_k)$ with $k > 1$ is a sequence model of MACRO–STEP such that $N_k = (C,E,T,A,C')$, then the first and last components of each $N_i$, for $1 \leq i \leq k$, must be identical to $C$ and $C'$, respectively. This is because $C_1 = C_2$ if $C_1 \subseteq C_2$ and if both CONFIG$(\underline{C}_1)$ and CONFIG$(\underline{C}_2)$ hold.

To conclude this section, let us revisit the two examples of Figure 2. For the Statechart on the left we find that none of the classical models $(C_1,\{\},T,A,C_1')$, with $T \supseteq \{t_1,t_2\}$ and $A \supseteq \{a,b\}$, is a response model. Indeed, the formula MACRO–STEP possesses the sequence model $((C_1,\{\},\{\},\{\},C_1'),(C_1,\{\},T,A,C_1'))$ of length 2, which shows that none of the elements in $T$ and $A$ are causal. On the other hand, $(C_1,\{\},\{\},\{\},C_1)$ is not only the smallest classical model but also a response model.

For the Statechart on the right in Figure 2 we find that only the minimal classical model $(D_1, \{\}, \{t\}, \{b\}, D_2)$ is a response model. The other one, $(D_1, \{\}, \{\}, \{a\}, D_1)$, is not a response model, since we have that the sequence model $((D_1, \{\}, \{\}, \{\}, D_1), (D_1, \{\}, \{\}, \{a\}, D_1))$ of length 2 witnesses that $a$ is non–causally introduced.

# 4 Discussion and Related Work

This section discusses our approach in the context of related work. Perhaps the most striking feature of our logical encoding of Statecharts' two–level step semantics is that it is essentially of a propositional nature. This is surprising since the explicit formalization in logic of both the operational and the declarative semantics given by Pnueli and Shalev [18] require second–order constructions. Regarding their operational semantics, the logical encoding of the *step–construction algorithm* would have to involve existential quantification over a–priori unbounded execution paths. In the declarative semantics, Pnueli and Shalev make use of the genuine second–order property of *inseparability* which encodes the principle of causality. In contrast, causality is captured in our intuitionistic setting by the notion of a response model, i.e., externally in the model theory rather than in Statecharts formulas.

In a previous paper [11], the authors have shown that the intuitionistic approach to Statecharts semantics not only provides for a compositional refinement of the macro–step construction of Pnueli and Shalev, but also leads to a fully–abstract semantics in terms of linear, intuitionistic Kripke structures. However, the work presented in [11] only focuses on single, fixed macro steps and abstracted from target configurations. In contrast, the encoding given above covers the full transition relation between arbitrary Statecharts configurations. However, a subtle question relating to the role of transition names remains. Notice that our encoding makes essential use of transition names as events and implicitly assumes that all transitions of a Statechart have distinct names. This, of course, is consistent with the standard presentation of the macro–step construction [18] which uses transition names, too. However, since transition names cannot be observed by the environment of a Statechart, it should be possible to do away with them. From the results in [11] it is clear that, within our setting, transition names are essential for capturing the choice between conflicting transitions in an OR–state. The reason for this is that our linear Kripke models cannot encode nondeterministic choice. Consequently, choices must be handled by auxiliary propositional constants that act as mutual exclusion events, for which simply the unambiguous transition names may be taken. Unfortunately, the straightforward approaches for abstracting from these names, such as via existential quantification, are incompatible with our notion of response model. However, we believe that by extending the model theory to branching Kripke structures instead of linear ones and by suitably modifying the definition of a response model, it should be possible to eliminate transition names in Statecharts formulas.

The approach of translating Statecharts into propositional formulas, as presented in this paper, falls into the class of *shallow embeddings*, which is distinguished from the class of *deep embeddings*. Both are possible strategies available for integrating a diagrammatic design language, such as Statecharts, with formal specification and validation techniques based on logic, such as provided by theorem provers [3, 16, 17]. In a deep embedding, the designs expressed in the diagrammatic language are considered as object–level terms in the logic, whose semantics is captured by axioms and rules. The advantage of this technique is that it allows one to use the logic formalism to derive not only properties of individual designs but also meta–theorems about the whole of the design language. However, when one is mainly interested in individual designs, the shallow–embedding strategy may be more beneficial. It translates designs not into the terms of the logic but into formulas that specify the designs' behaviors directly. In a shallow embedding, the rules and derivation mechanisms of the logic formalism manipulate designs *themselves* rather than just statements *about* designs. This has the advantage that the algorithmic technology built into the formal specification and validation framework, such as proof tactics within theorem provers, is immediately available to handle individual designs.

Other semantic dialects of Statecharts have been either defined operationally in process–algebraic settings [10, 12, 14, 20] or denotationally as in [8]. However, more closely related to our approach is research conducted for the diagrammatic real–time specification language *Modecharts* [9] and the textual reactive–system specification language *Esterel* [2]. Jahanian and Mok [9] gave a shallow embedding of Modecharts into a first–order predicate logic. The first–order setting is mainly used to express absolute timing properties, but also for defining causal chains of steps. This setting suffices since Modecharts has a flat micro–step semantics, rather than a layered micro– and macro–step semantics as the original Statecharts language. In contrast to Modecharts but similar to Statecharts, Esterel has a two–level step semantics encapsulating the synchrony hypothesis. Recently, Berry [1] has given an embedding of Esterel in propositional logic via a compilation into constructive circuits. Similar to our approach, his encoding follows the rules of a constructive logic rather than the ones of classical logic.

# 5 Conclusions and Future Work

This paper presented, for the first time in the literature, a model–theoretic account of Harel, Pnueli and Shalev's original Statecharts semantics [18], which showed that the propositional logic underlying Statecharts is intuitionistic rather than classical. The use of an intuitionistic approach to Statecharts is not accidental: it is embodied in the explicit notion of causality within Statecharts' step semantics and is also supported by recent compositionality and full-abstractness results for Statecharts obtained by the authors [11]. Equally important, the model–theoretic approach of this paper lays the foundation for integrating Statecharts with formal verification tools, in particular with theorem provers.

Regarding future work we plan to implement our approach within the theorem prover PVS [16] in order to facilitate formal reasoning about Statecharts. Moreover, we intend to extend our intuitionistic framework in a way that permits for the hiding of transition names, which requires the identification of an intuitionistic operator for encoding transition choices.

# References


[1] G. Berry. The constructive semantics of pure ESTEREL. Draft version 3.0, 1999.

[2] G. Berry. The foundations of ESTEREL. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 2000.

[3] M.J.C. Gordon and T.F. Melham. *Introduction to HOL*. Cambridge Univ. Press, 1993.

[4] D. Harel. Statecharts: A visual formalism for complex systems. *SCP*, 8:231–274, 1987.

[5] D. Harel and A. Naamad. The STATEMATE semantics of Statecharts. *ACM Trans. Softw. Eng.*, 5(4):293–333, 1996.

[6] D. Harel, A. Pnueli, J.P. Schmidt, and R. Sherman. On the formal semantics of Statecharts. In *LICS '87*, pp. 56–64. IEEE Comp. Soc. Press, 1987.

[7] D. Harel and M. Politi. *Modeling Reactive Systems with Statecharts: The* STATEMATE *Approach*. McGraw Hill, 1998.

[8] C. Huizing, R. Gerth, and W.-P. de Roever. Modeling Statecharts behavior in a fully abstract way. In *CAAP '88*, vol. 299 of *LNCS*, pp. 271–294. Springer-Verlag, 1988.

[9] F. Jahanian and A.K. Mok. MODECHART: A specification language for real-time systems. *IEEE Trans. on Softw. Eng.*, 20(12):933–947, 1994.

[10] F. Levi. *Verification of Temporal and Real-time Properties of Statecharts*. PhD thesis, Univ. of Pisa, 1997.

[11] G. Lüttgen and M. Mendler. Fully-abstract Statecharts semantics via intuitionistic Kripke models. In *ICALP 2000*, vol. 1853 of *LNCS*, pp. 163–174. Springer-Verlag, 2000.

[12] G. Lüttgen, M. von der Beeck, and R. Cleaveland. Statecharts via process algebra. In *CONCUR '99*, vol. 1664 of *LNCS*, pp. 399–414. Springer-Verlag, 1999.

[13] G. Lüttgen, M. von der Beeck, and R. Cleaveland. A compositional approach to Statecharts semantics. In *FSE 2000*. ACM Press, 2000.

[14] A. Maggiolo-Schettini, A. Peron, and S. Tini. Equivalences of Statecharts. In *CONCUR '96*, vol. 1119 of *LNCS*, pp. 687–702. Springer-Verlag, 1996.

[15] F. Maraninchi. Operational and compositional semantics of synchronous automaton compositions. In *CONCUR '92*, vol. 630 of *LNCS*, pp. 550–564. Springer-Verlag, 1992.

[16] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal verification for fault-tolerant systems: Prolegomena to the design of PVS. *IEEE Trans. on Softw. Eng.*, 21(2):107–125, 1995.

[17] L.C. Paulson. ISABELLE*: A Generic Theorem Prover*, vol. 828 of *LNCS*. Springer-Verlag, 1994.

[18] A. Pnueli and M. Shalev. What is in a step: On the semantics of Statecharts. In *TACS '91*, vol. 526 of *LNCS*, pp. 244–264. Springer-Verlag, 1991.

[19] P. Scholz. *Design of Reactive Systems and Their Distributed Implementation with Statecharts*. PhD thesis, Munich Univ. of Technology, 1998.

[20] A.C. Uselton and S.A. Smolka. A compositional semantics for Statecharts using labeled transition systems. In *CONCUR '94*, vol. 836 of *LNCS*, pp. 2–17. Springer-Verlag, 1994.

[21] D. van Dalen. Intuitionistic logic. In *Handbook of Philosophical Logic*, vol. III, ch. 4, pp. 225–339. Reidel, 1986.

[22] M. von der Beeck. A comparison of Statecharts variants. In *FTRTFT '94*, vol. 863 of *LNCS*, pp. 128–148. Springer-Verlag, 1994.