A CONCISE COMPOSITIONAL STATECHARTS SEMANTICS DEFINITION

Michael von der Beeck
Department of Computer Science
Munich University of Technology
Arcisstr. 21, D-80290 München, Germany
beeck@in.tum.de

Abstract Statecharts is a well-known specification language for modeling system behavior. Its intuitive graphical appeal for specifying hierarchical, concurrent state machines conceals the difficulties in formalizing its semantics. The causes are in Statecharts' two-level semantics when demanding compositionality, as is necessary in order to obtain easy comprehensibility and modularity of system specifications. This paper suggests a compositional approach for formalizing the Statecharts semantics directly on sequences of micro steps using labeled transition systems as semantical domain. Though we consider causality, global consistency, as well as the synchrony hypothesis, our approach results in a very short and concise semantics definition of Statecharts.

Keywords: Statecharts, operational semantics, compositionality

1. INTRODUCTION

Statecharts is a visual language widely accepted for specifying the behavior of *reactive* and *embedded systems* [4]. It constitutes an essential part of many well-known object-oriented specification notations like UML [2] and ROOM [19]. The Statecharts language extends traditional *finite-state machines* by concepts of *hierarchy* and *concurrency*. Hierarchy is achieved by embedding one Statechart in a state of another Statechart – resulting in an Or-state. Concurrency is supported by a complex Statechart (an And-state) composed of several simultaneously active sub-Statecharts communicating with each other via a sort of broadcasting mechanism. Furthermore, in addition to traditional finite-state machines the Statecharts language allows transitions labeled by pairs of events, where the first component is referred to as *trigger* and may include *negated events*, and the second component is referred to as

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35533-7_26

Tommaso Bolognesi and Diego Latella (Eds.), *Formal Methods for Distributed System Development.*

action. Intuitively, if the environment offers the positive events in the trigger, but not the negated ones, then the transition is triggered and can be executed, thereby performing a so-called *micro step* and generating the events contained in the action which in turn can trigger new transitions while disabling others. When this chain reaction comes to a halt, one execution step – also referred to as a *macro step* – is complete.

On the one hand the Statecharts language has been proven very successful for specification purposes due to its intuitive syntax and semantics, on the other hand precisely defining its semantics has been a long-term challenge. Many proposals for defining the Statecharts semantics [3, 5, 6, 9, 10, 11, 12, 14, 17, 18, 20, 21] have emerged [22, 7]. Semantical intricacies arise if the following wishful properties – Statecharts specific as well as quite general semantic ones – shall be simultaneously achieved by the Statecharts semantics to be defined:

- synchrony hypothesis
  This hypothesis [1] requires that the reaction time of a system is short in comparison with the environment, i.e. the system reacts to an input, before the environment sends the next input.

- causality principle
  Causality holds if there exists a (causal) ordering among the transitions of a macro step such that each transition t of the macro step is triggered by the events which are offered by the environment or which are generated by transitions occuring before t in that ordering.

- global consistency
  Global consistency requires that every transition of a macro step should not only be enabled by the events generated in previous micro steps of this macro step, but by the events generated in all micro steps of this macro step.

- compositionality
  Compositionality ensures that the semantics of a Statechart is only defined in terms of the semantics of its components. This general semantic property provides essential means supporting development and verification in a structural way.

Several Statecharts semantics definitions circumvent difficulties by ignoring some of these properties or by only having defined the semantics in an informal way. Other approaches combine these properties, but at the expense of resulting in quite a lengthy and complex Statecharts semantics definition.

Therefore it is the aim of this paper to define a Statecharts semantics which offers all of the aforementioned features, but which nevertheless is as concise and easily comprehensible as possible.

We present an operational Statecharts semantics of minimal length directly defined on sequences of micro steps. We use labelled transition systems as the semantic domain and SOS rules (Structured Operational Semantics [16]) in order to achieve a compositional semantics.

The remainder of this paper is organized as follows. The next section offers a brief introduction to Statecharts and the way their semantics are traditionally defined. In Section 3 we present our definition of Statecharts semantics. Related work is considered in Section 4, while Section 5 provides our conclusions.

2. TRADITIONAL STATECHARTS

In this section we present the syntax and traditional semantics of Statecharts. As the starting point for our new semantics definition we define Statecharts terms, which constitute a textual notation providing the abstract syntax of (visual) Statecharts. Then we briefly describe our reference semantics. The presentation of syntax and semantics is supported by an example, which we will later consider again for illustrating our new semantics definition.

2.1. SYNTAX

Example. Consider the Statechart in Fig. 1. It consists of an *And-state*, labeled by n_1, which denotes the parallel composition of the two Statecharts labeled by n_2 and n_3, both of which are *Or-states*, i.e. both describe a sequential state machine. Or-state n_2 is further refined by Or-state n_4 and basic state n_5 which are connected via transitions t_1 and t_2. Label c of t_1 specifies that t_1 is triggered by c, i.e., by the occurence of event c. Or-state n_4 is further refined by basic states n_8 and n_9, connected by transition t_4 with label $a \wedge \neg c$ specifying that t_4 is triggered if event a occurs, but not event c. Or-state n_3 consists of two basic states n_6 and n_7 connected via transition t_3 with label a/c, so that upon occurence of event a transition t_3 can execute thereby generating event c.

Statecharts Terms. Statecharts is a visual language. However, for our purposes it is convenient to represent Statecharts not visually but by terms. This is also done in related work [9, 12, 20], and our approach closely follows the one described by Maggiolo-Schettini et al. [12]. For-

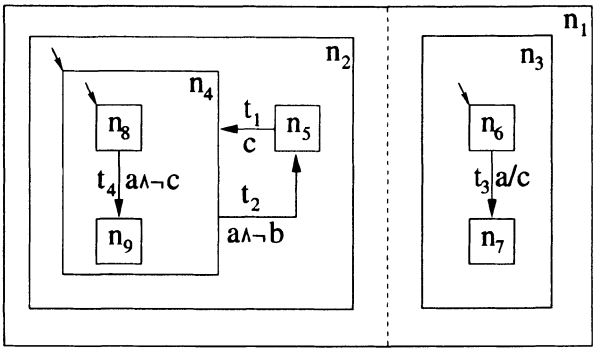

Figure 1 Statechart Example

mally, let $\mathcal{N}$ be a countable set of names for Statecharts states, $\mathcal{T}$ be a countable set of names for Statecharts transitions, and Π be a countable set of Statecharts events. Moreover, we associate with every event $e \in \Pi$ its negated counterpart $\neg e$. We also lift negation to negated events by defining $\neg\neg e =_{\mathrm{df}} e$. Furthermore, we write $\neg I$ for $\{\neg e \mid e \in I\}$. Then, the set SC of Statecharts terms is defined to be the least set satisfying the following rules.

1 **Basic state:** If $n \in \mathcal{N}$, then $s = [n]$ is a Statecharts term.

2 **Or-state:** If $n \in \mathcal{N}$, $s_1, \ldots, s_k$ are Statecharts terms for $k > 0$, $\rho = \{1, \ldots, k\}$, $l \in \rho$, and $T \subseteq \mathsf{TR} =_{\mathrm{df}} \mathcal{T} \times \rho \times 2^{\Pi \cup \neg\Pi} \times 2^{\Pi} \times \rho$, then $s = [n : (s_1, \ldots, s_k); l; T]$ is a Statecharts term. Here, $s_1, \ldots, s_k$ are the sub-states of s, T is the set of transitions between these states, s_1 is the default state of s, and s_l is the currently active sub-state of s.

3 **And-state:** If $n \in \mathcal{N}$ and $s_1, \ldots, s_k$ are Statecharts terms for $k > 0$, then $s = [n : (s_1, \ldots, s_k)]$ is a Statecharts term. Here, $s_1, \ldots, s_k$ are the (parallel) sub-states of s.

We define $\mathsf{root}(s) =_{\mathrm{df}} n$. If $\hat{t} = (t, i, I, O, j) \in T$ is a transition of Or-state $[n : (s_1, \ldots, s_k); l; T]$, then we define $\mathsf{name}(\hat{t}) =_{\mathrm{df}} t$, $\mathsf{source}(\hat{t}) =_{\mathrm{df}} s_i$, $\mathsf{trg}(\hat{t}) =_{\mathrm{df}} I$, $\mathsf{act}(\hat{t}) =_{\mathrm{df}} O$, and $\mathsf{target}(\hat{t}) =_{\mathrm{df}} s_j$. $\mathsf{trg}(\hat{t})$ is called the *trigger part* and $\mathsf{act}(\hat{t})$ is called the *action part* of $\hat{t}$. Furthermore, we define $\mathsf{trg}^+(\hat{t}) =_{\mathrm{df}} \mathsf{trg}(\hat{t}) \cap \Pi$ and $\mathsf{trg}^-(\hat{t}) =_{\mathrm{df}} \mathsf{trg}(\hat{t}) \cap \neg\Pi$.

We assume the following for SC: (i) all state names and transition names are mutually disjoint, (ii) no transition t produces an event that contradicts its trigger, i.e., $\mathsf{trg}(t) \cap \neg\mathsf{act}(t) = \emptyset$, and (iii) no transition t produces an event that is included in its trigger, i.e., $\mathsf{trg}(t) \cap \mathsf{act}(t) =$

$\emptyset$. As a consequence of (i), states and transitions in Statecharts terms are uniquely referred to by their name. For convenience, we sometimes identify a Statecharts state s and a transition t with its name $\mathsf{root}(s)$ and $\mathsf{name}(t)$, respectively. Furthermore, let $\mathsf{trans}(s)$ be the set of transitions within a Statecharts term s.

The Statecharts term corresponding to the Statechart example presented in Fig. 1 is term s_1, which is defined as follows.

$$
\begin{array}{ll}
s_1 = [n_1 : (s_2, s_3)] & s_2 = [n_2 : (s_4, [n_5]); 1; \{\hat{t}_1, \hat{t}_2\}] \\
s_3 = [n_3 : ([n_6], [n_7]); 1; \{\hat{t}_3\}] & s_4 = [n_4 : ([n_8], [n_9]); 1; \{\hat{t}_4\}] \\
\hat{t}_1 = (t_1, 5, \{c\}, \emptyset, 4) & \hat{t}_2 = (t_2, 4, \{a, \neg b\}, \emptyset, 5) \\
\hat{t}_3 = (t_3, 6, \{a\}, \{c\}, 7) & \hat{t}_4 = (t_4, 8, \{a, \neg c\}, \emptyset, 9)
\end{array}
$$

The Statecharts variant considered here does not include all "features" present in some other variants. In particular, we prohibit *interlevel transitions* and – only for space limitations – do not treat the *history mechanism.*

2.2. TRADITIONAL SEMANTICS

In this section, we sketch the semantics of Statecharts terms adopted from Maggiolo-Schettini, Peron, and Tini [12] which will serve for us as a reference semantics and which is a slight variant of the "traditional" Statecharts semantics, as proposed by Pnueli and Shalev [17].[1]

As mentioned before, a Statechart s reacts to the arrival of some external events by triggering enabled micro steps, possibly in a chain-reaction–like manner, thereby performing a macro step. More precisely, a macro step comprises a maximal set of micro steps, or transitions, that

- are *triggered* by events which are either offered by the environment or generated by other micro steps of the same macro step,
- are mutually *consistent*,
- are mutually *compatible*,
- are *relevant*, and
- obey the principle of *causality.*

The Statecharts principle of *global consistency*, which prohibits an event to be present and absent in the same macro step, is subsumed by the notions of *triggered* and *compatible.*

[1]The variation was motivated by the fact that Pnueli and Shalev's step-construction procedure – which will be presented in the following – can fail.

Figure 2 Step-construction algorithm

```
procedure step-construction(s, E);
var T := ∅;
while T ⊂ enabled(s, E, T) do
  choose t ∈ enabled(s, E, T) \ T;
  T := T ∪ {t}
od;
return T
```

Now, we briefly introduce these notions. Let $s \in \mathsf{SC}$, $t \in \mathsf{trans}(s)$, $T \subseteq \mathsf{trans}(s)$, and $E \subseteq \Pi$.

Transition t is *consistent* with all transitions in T, denoted by $t \in \mathsf{consistent}(s,T)$, if t is not in the same parallel component as any transition in T. Transition t is *compatible* to all transitions in T, denoted by $t \in \mathsf{compatible}(s,T)$, if no event produced by t appears negated in the trigger of a transition in T. Transition t is *relevant* for Statechart term s, denoted by $t \in \mathsf{relevant}(s)$, if the source state of t is currently active. Transition t is *triggered* by event set E, denoted by $t \in \mathsf{triggered}(s,E)$, if the positive, but not the negative trigger events of t are in E. Finally, we say that transition t is *enabled* in s with respect to event set E and transition set T, if $t \in \mathsf{enabled}(s,E,T)$, where $\mathsf{enabled}(s,E,T) =_{\mathrm{df}} \mathsf{relevant}(s) \cap \mathsf{consistent}(s,T) \cap \mathsf{triggered}(s, E \cup \bigcup_{t \in T} \mathsf{act}(t)) \cap \mathsf{compatible}(s,T)$.

Observe that the maximality of each macro step implements the synchrony hypothesis of Statecharts.

Macro Step. Function enabled constitutes the essential means to define a macro step. Unfortunately, for given Statechart s and event set E a solution T^* of equation $T = \mathsf{enabled}(s,E,T)$ need not constitute a macro step, since causality is neglected here. However, Maggiolo-Schettini et al. [12] provide an operational approach for causally justifying the triggering of each transition of a macro step using the nondeterministic *step-construction* algorithm presented in Fig. 2.

Given Statecharts term s and set E of events, $\mathsf{step\text{-}construction}(s,E)$ nondeterministically computes one macro step, i.e. a set of transitions, out of the set of all possible macros steps. For event set I offered by the environment and set T of transitions computed by $\mathsf{step\text{-}construction}(s,I)$, the Statecharts term s may evolve in a single *macro step* $s \overset{I}{\underset{O}{\Longrightarrow}} s'$ to Statecharts term s', thereby executing the transitions in T and producing the events $O =_{\mathrm{df}} \bigcup_{t \in T} \mathsf{act}(t)$.

In the following we consider some macro steps of our Statechart example from Fig. 1. For convenience, we abbreviate a Statechart term by its

active basic states - e.g. writing (n_8, n_6) for term s_1 defined before. Let $\Pi = \{a, b, c\}$ and assume that the environment only offers event a. Then transition t_4 can execute resulting in macro step $(n_8, n_6) \xRightarrow[\emptyset]{\{a\}} (n_9, n_6)$. Note that though transition t_3 is also triggered, it can not be executed together with t_4 in the same macro step, because this would violate global consistency, since t_3 would generate event c whose negative conterpart $\neg c$ is contained in the trigger of t_4. But t_2 and t_3 can execute in the same macro step, since both are triggered by event a and since global consistency is fulfilled resulting in macro step $(n_8, n_6) \xRightarrow[\{c\}]{\{a\}} (n_5, n_7)$. Note, that the execution of t_3 alone does not constitute a valid macro step in the considered situation, because a macro step has to be maximal. However, if the environment offers the event set $\{a, b\}$ or $\{a, b, c\}$, then t_3 constitutes a complete macro step.

3. NEW SEMANTICS DEFINITION

3.1. AIM

We aim at the development of the definition of a Statecharts semantics which satisfies the following properties:

- general properties (i.e. not specific for the Statecharts language): The semantics shall be formal, as concise as possible, comprehensible, modular, and easily modifiable. These properties suggest to define a compositional semantics.

- Statecharts specific properties: The semantics shall fulfill causality, the synchrony hypothesis, and global consistency, as already provided by our (non-compositional) reference semantics.

3.2. APPROACH

We directly define the semantics on sequences of micro steps. This is in contrast to our other approaches [10, 11], where we defined a fine-grained semantics on single micro steps. Due to the direct approach a very concise Statecharts semantics results. We use labelled transition systems (LTS) as the semantic domain, where LTS-states model Statechart terms and LTS-transitions model sequences of Statechart micro steps. Considering micro step sequences – which may not be of maximal length – instead of macro steps is a prerequisite to achieve a compositional semantics, because a non-maximal micro step sequence of a Statecharts term s constitutes a potential macro step (or parts thereof) of an And-state with sub-state s.

Our semantics combines causality and synchrony on the macro step level, and meets compositionality on the level of micro step sequences. In contrast to the work of Maggiolo-Schettini et al. [12] and Uselton and Smolka [20] our semantics gets along with quite simple labels.

Note that due to a result of Huizing and Gerth [7] it is impossible to define a single-level LTS semantics that combines causality, synchrony, and compositionality on the level of macro steps if the labels are simply pairs of event sets in the style "trigger/action".

3.3. FORMAL SEMANTICS DEFINITION

Let LTS be the set of *labeled transition systems.* Then the semantics $[\![s]\!]$ of a Statecharts term $s \in \mathsf{SC}$ is given by the labeled transition system $(\mathsf{SC}, 2^{\Pi} \times 2^{\neg\Pi} \times 2^{\Pi} \times 2^{\mathcal{T}}, \longrightarrow, s) \in \mathsf{LTS}$, where

- SC is the set of states,
- $2^{\Pi} \times 2^{\neg\Pi} \times 2^{\Pi} \times 2^{\mathcal{T}}$ the set of labels,
- $\longrightarrow \subseteq \mathsf{SC} \times (2^{\Pi} \times 2^{\neg\Pi} \times 2^{\Pi} \times 2^{\mathcal{T}}) \times \mathsf{SC}$ the transition relation, and
- s the start state.

For the sake of simplicity, we abbreviate $(s, (I^+, I^-, O, L), s') \in \longrightarrow$ by $s \xrightarrow[O]{(I^+,I^-)}_L s'$. We say that *Statechart term s may change with label (I^+, I^-, O) to Statechart term s'*, where s and s' are called the *source* and the *target states* of this transition, respectively, and I^+ and I^- are called the *positive* and *negative input part* of the label, respectively, whereas O is called the *output part* of the label.[2] If we are not interested in the target state, we write $s \xrightarrow[O]{(I^+,I^-)}_L$ instead of $\exists s'.\ s \xrightarrow[O]{(I^+,I^-)}_L s'$ and say that Statechart s *may engage in a transition labeled with* (I^+, I^-, O). We define $(s_1, \ldots, s_k)_{[l \mapsto s']} =_{\mathrm{df}} (s_1, \ldots, s_{l-1}, s', s_{l+1}, \ldots, s_k)$ for $1 \leq l \leq k$ and $s' \in \mathsf{SC}$, and define $\vec{s} =_{\mathrm{df}} (s_1, \ldots, s_k)$. We write $s - t \rightarrow s'$ instead of $(\mathsf{source}(t) = s \ \wedge\ \mathsf{target}(t) = s')$ for $t \in \mathsf{TR}$ and $s, s' \in \mathsf{SC}$.

Furthermore, we need function $\mathsf{default} : \mathsf{SC} \longrightarrow \mathsf{SC}$ which for a given Statecharts term s sets the default state of every Or-state s' within s to the currently active sub-state of s'.

$$
\begin{array}{ll}
\mathsf{default}([n]) & =_{\mathrm{df}} [n] \\
\mathsf{default}([n : (s_1, \ldots, s_k); l; T]) & =_{\mathrm{df}} [n : (\mathsf{default}(s_1), s_2, \ldots, s_k); 1; T] \\
\mathsf{default}([n : (s_1, \ldots, s_k)]) & =_{\mathrm{df}} [n : (\mathsf{default}(s_1), \ldots, \mathsf{default}(s_k))]
\end{array}
$$

[2] Here we ignore the set L of transition names, because L is not necessary for the semantics definition itself. L is only used to compute macro steps out of the semantics.

Transition relation $\longrightarrow$ is now defined by Table 1 using SOS rules. Due to space limitations we use the rule formats

$$\textit{name}\frac{\textit{premise}}{\begin{array}{c}\textit{conclusion}\\ \textit{(side condition)}\end{array}} \quad \text{as well as } \textit{name}\frac{\textit{premise}}{\textit{conclusion}} \ \textit{(side condition)}.$$

Table 1 SOS rules

$$\text{OR-1}\ \frac{-}{[n:\vec{s};i;T]\xrightarrow[\mathsf{act}(t)]{(\mathsf{trg}^+(t),\mathsf{trg}^-(t))}_{\{\mathsf{name}(t)\}}[n:\vec{s}_{[l\mapsto\mathsf{default}(s_l)]};l;T]}\ \left(\begin{array}{c}t\in T,\\ s_i-t-s_l\end{array}\right)$$

$$\text{OR-2}\ \frac{s_l\xrightarrow[O]{(I^+,I^-)}_L s_l'}{[n:\vec{s};l;T]\xrightarrow[O]{(I^+,I^-)}_L[n:\vec{s}_{[l\mapsto s_l']};l;T]}$$

$$\text{AND}\ \frac{\left(\forall m\in M: s_m\xrightarrow[O_m]{(I_m^+,I_m^-)}_{L_m}s_m'\right)\ \wedge\ \left(\forall i,j\in M:(\neg I_i^-)\cap O_j=\emptyset\right)}{[n:\vec{s}]\xrightarrow[\bigcup_{m\in M}O_m]{\left(\bigcup_{l\in N}\left(I^+_{f(l)}\setminus\left(\bigcup_{j=1}^{l-1}O_{f(j)}\right)\right),\ \bigcup_{m\in M}I_m^-\right)}_{\bigcup_{m\in M}L_m}[n:\vec{s'}]}$$

$$\left(\begin{array}{c}M\subseteq K=\{1,\ldots,k\},\ N=\{1,\ldots,|M|\},\\ f:N\to M\ \text{bijection}\ ,\ \forall p\in K\setminus M:\ s_p'=s_p\end{array}\right)$$

Explanation of the SOS rules:

- OR-1 (Statechart transition execution)
  This rule describes the execution of a Statechart transition $t\in T$ of an Or-state $[n:\vec{s};i;T]$. It defines that the Or-state with currently active sub-state s_i may change with label $(\mathsf{trg}^+(t),\mathsf{trg}^-(t),\mathsf{act}(t))$ to Or-state $[n:\vec{s}_{[l\mapsto\mathsf{default}(s_l)]};l;T]$ with currently active sub-state s_l, if $t\in T$, $\mathsf{source}(t)=s_i$, and $\mathsf{target}(t)=s_l$. The definition of

the transition label is motivated as follows: In order to execute the trigger, all positive events $\mathsf{trg}^+(t)$ in $\mathsf{trg}(t)$ must be offered by the environment, and all negative events $\mathsf{trg}^-(t)$ in $\mathsf{trg}(t)$ must be absent. Then t can execute by offering the output $\mathsf{act}(t)$.

- OR-2 (Propagation of a change of a sub-state)
  If Statechart term s_l (with $l \in \{1, \ldots, k\}$) may change with label (I^+, I^-, O) to Statechart term s'_l, then Or-state $[n : \vec{s}; l; T]$ with currently active sub-state s_l may change with the same label to Or-state $[n : \vec{s}_{[l \mapsto s'_l]}; l; T]$ with currently active sub-state s'_l.

- AND (Synchronization of parallel states)
  This rule defines a sort of synchronization between (parallel) sub-states of an And-state. More concretely, AND defines for all such sub-states s_m of And-state $[n : \vec{s}]$, which may engage in a transition, the interaction between s_m and the other sub-states s_j of $[n : \vec{s}]$, where $1 \leq j \leq k$, $j \neq m$. (Remember that $\vec{s} =_{\mathrm{df}} (s_1, \ldots, s_k)$.) If $M \subseteq \{1, \ldots, k\}$, such that

  1 for all $m \in M$ sub-state s_m may change with label (I^+_m, I^-_m, O_m) to sub-state s'_m and

  2 the global consistency condition $\forall i, j \in M : (\neg I^-_i) \cap O_j = \emptyset$ is fulfilled,

  then And-state $[n : \vec{s}]$ may change to And-state $[n : \vec{s'}]$, where $\forall p \in K \setminus M : s'_p = s_p$.

  The label of this state change is justified as follows:

  - To explain the definition of the positive input part, we consider every parallel component $s_{f(l)}$ of $[n : \vec{s}]$ separately for $l \in N$ (or equivalently for every $f(l) \in M$).[3] $I^+_{f(l)}$ is the set of positive events which have to be offered by the environment as a precondition for the state change from $s_{f(l)}$ to $s'_{f(l)}$ if no events have been generated within the corresponding sequence of micro steps. But since we can assume that the state change from $s_{f(l)}$ to $s'_{f(l)}$ is the l-th state change, i.e. the l-th micro step, within the micro step sequence from $[n : \vec{s}]$ to $[n : \vec{s'}]$, we can further assume that exactly all the state changes from $s_{f(j)}$ to $s'_{f(j)}$ for $j \in \{1, \ldots, l-1\}$ have

[3] Note, that we use the bijection $f : \{1, \ldots, |M|\} \to M$ in order to express an ordering of micro step executions within a sequence of micro steps which is independent of the order of substates s_m in $[n : \vec{s}]$ for $(1 \leq m \leq k)$.

already taken place in this micro step sequence. Due to these state changes the set $\bigcup_{j=1}^{l-1} O_{f(j)}$ of events has been generated, before a possible state change from $s_{f(l)}$ to $s'_{f(l)}$ occurs. Taking these generated events into account, the set of positive events which still have to be provided by the environment, so that the state change from $s_{f(l)}$ to $s'_{f(l)}$ can occur, is the set $I^+_{f(l)} \setminus \left(\bigcup_{j=1}^{l-1} O_{f(j)}\right)$. Thus the positive input part for the micro step sequence, i.e. for the state change from $[n : \vec{s}]$ to $[n : \vec{s'}]$, is given by the set $\bigcup_{l \in N} \left(I^+_{f(l)} \setminus \left(\bigcup_{j=1}^{l-1} O_{f(j)}\right)\right)$.

- The negative input part and the output part are simply given by the union of all I^-_m and O_m for $m \in M$, respectively.

Rule AND reveals that our semantics is defined on the granularity of micro step sequences instead of on a single micro step.

We define a notion of *macro step* for our semantics as follows:

Macro Step. Let $s, s' \in \mathsf{SC}$, $I, O \subseteq \Pi$. Then: $s \overset{I}{\underset{O}{\Longrightarrow}} s'$ *if and only if* $(\exists I^+ \subseteq \Pi, I^- \subseteq \neg\Pi, L \subseteq \mathcal{T}.(s \xrightarrow[O]{(I^+,I^-)}_L s' \wedge I^+ \subseteq I \wedge (\neg I^-) \cap I = \emptyset)) \wedge (\nexists \hat{I}^+, \hat{O} \subseteq \Pi, \hat{I}^- \subseteq \neg\Pi, \hat{L} \subseteq \mathcal{T}.(s \xrightarrow[\hat{O}]{(\hat{I}^+,\hat{I}^-)}_{\hat{L}} \wedge \hat{I}^+ \subseteq I \wedge (\neg\hat{I}^-) \cap I = \emptyset \wedge L \subsetneq \hat{L}))$

In this case we say that Statechart term s may perform a *macro step* with input I and output O to Statechart term s'.

This macro step definition employs the information about the set L of Statechart transitions executed if a (semantical) transition $s \xrightarrow[O]{(I^+,I^-)}_L s'$ takes place by requiring that L must be maximal. Thus a macro step consists of a maximal number of micro steps, whereas the AND-rule of our Statecharts semantics allows micro step sequences of non-maximal length. Therefore the semantics considers all potential macro steps which can arise if the considered Statechart is put in parallel with another Statechart context. This approach realizes a compositional Statecharts semantics on sequences of micro steps. But note that as soon as macro steps are computed, i.e. when maximality of micro step sequences is required, compositionality is lost.

To sum up, our semantics approaches the combination of causality, synchrony, and compositionality by satisfying the first two properties on the macro step level and by approaching macro step compositionality due to the semantics' compositionality on micro step sequences.

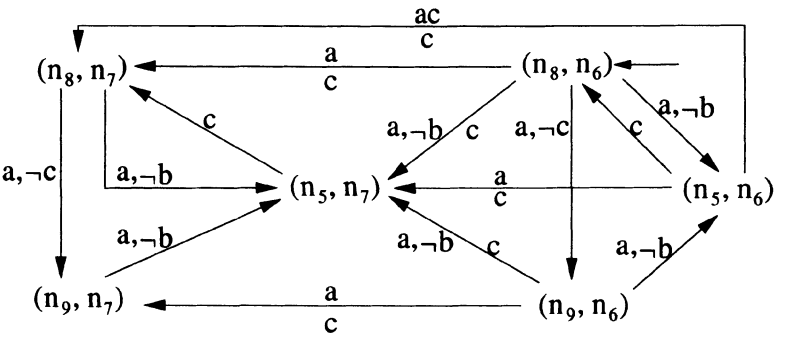

Figure 3 Semantics for the Statechart Example from Fig. 1

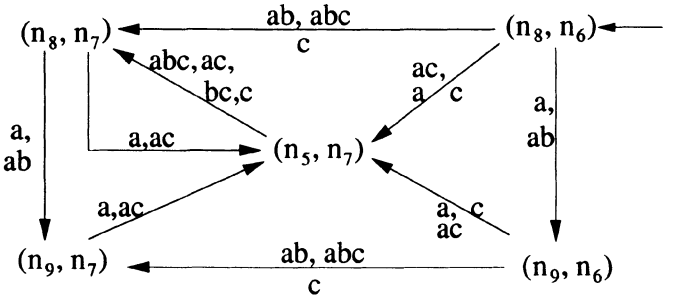

Figure 4 Macro Steps for the Statechart Example from Fig. 1

3.4. EXAMPLE

Consider again our Statecharts example from Fig. 1. Its semantics, i.e. a labeled transition system, and the possible macro steps are graphically represented as state transition diagrams in Fig. 3 and Fig. 4.[4] For example, the macro steps $(n_8, n_6) \overset{\{a\}}{\underset{\emptyset}{\Longrightarrow}} (n_9, n_6)$ and $(n_8, n_6) \overset{\{a\}}{\underset{\{c\}}{\Longrightarrow}} (n_5, n_7)$ already considered before can be found in Fig. 4, where we represent a macro step $\ldots \overset{I}{\underset{O}{\Longrightarrow}} \ldots$ by writing I to the left of or above, whereas O to the right of or below the arrow. Furthermore, we write e.g. ab instead of $\{a, b\}$. Different possibilities for I at the same arrow are separated by ",". Therefore, the arrow from (n_8, n_6) to (n_5, n_7) also represents macro step $(n_8, n_6) \overset{\{ac\}}{\underset{\{c\}}{\Longrightarrow}} (n_5, n_7)$. Note, that we have skipped empty output sets.

[4] A Statecharts term is again abbreviated by its active basic states.

The above-mentioned macro steps from (n_8, n_6) to (n_5, n_7) both result from the semantical transition $(n_8, n_6) \xrightarrow[\{c\}]{(\{a\},\{\neg b\})} (n_5, n_7)$, which can be found in Fig. 3. A semantical transition is represented analogously to a macro step, but both – positive and negative – input parts occur to the left of or above the arrow, where e.g. an input $(\{a\}, \{\neg b\})$ is abbreviated by $a, \neg b$ and an input $(\{a\}, \emptyset)$ is abbreviated by a. The semantical transition considered before results from one of the following two sequences of micros steps: $(n_8, n_6) \xrightarrow[\{c\}]{(\{a\},\emptyset)} (n_8, n_7) \xrightarrow[\emptyset]{(\{a\},\{\neg b\})} (n_5, n_7)$ and $(n_8, n_6) \xrightarrow[\emptyset]{(\{a\},\{\neg b\})} (n_5, n_6) \xrightarrow[\{c\}]{(\{a\},\emptyset)} (n_5, n_7)$.

3.5. EQUIVALENCE OF SEMANTICS

We can now formalize our intuition of the semantic relation between the traditional Statecharts semantics from Section 2.2 and our Statecharts semantics from Section 3.3 by relating their corresponding macro step notions as follows:

$$\forall s, s' \in \mathsf{SC},\ \forall I, O \subseteq \Pi: \quad s \overset{I}{\underset{O}{\Rightarrow}} s' \text{ if and only if } s \overset{I}{\underset{O}{\Longrightarrow}} s'.$$

4. RELATED WORK

We compare work concerning the formal semantics of Statecharts and quite similiar languages along the *three semantic dimensions of Statecharts*: causality, synchrony (hypothesis), and compositionality. Huizing and Gerth [7] have been the first considering these three properties simultaneously.

The starting point is provided by Harel et al. [6] developing a formal Statecharts semantics satisfying causality and synchrony, but ignoring compositionality. Subsequent work of Pnueli and Shalev [17] follows the same approach, but also considers global consistency.

In order to achieve synchrony and compositionality, some work neglects causality. So Scholz [18] uses streams as semantic domain resulting in a very concise, but non-causal fixed point semantics. Maraninchi [13] presents *Argos*, a language with a visual appeal very similar to Statecharts and a solid algebraic foundation. However, its compositional semantics – defined via SOS-rules as labelled transition systems – significantly differs from classical Statecharts semantics. For example, Argos is deterministic, abstracts from "non-causal" specifications, and allows a sequential component to fire more than once within a macro step.

Combining causality and compositionality, but at the expense of synchrony, has been the motivation for Mikk et al. [14] as well as Damm et al. [3]. However, in these cases one can not argue that they ignored synchrony, because they started from the Statecharts variant used in the STATEMATE tool [5]. In this variant an event generated within a macro step will not be sensed in the same, but the next macro step. The work of Latella et al. [8] as well as Paltor and Lilius [15] formalizes the semantics of UML-Statecharts [2] – the latter ignoring compositionality. Also UML-Statecharts do not fulfill the synchrony hypothesis.

From our point of view, the most interesting work concerns formalizing Statecharts semantics combining all three dimensions – causality, synchrony, and compositionality. Here, two classes can be distinguished:

The first class is based on a process algebra approach. At first the Statecharts language is embedded in a process algebra, for which then a structured operational semantics based on labelled transition systems is defined. Uselton and Smolka [21] and Levi [9] follow this approach by explicitly defining a causality relation on events and event sets, respectively. In our earlier work [11] we also present a process algebra approach, but do so without an explicitly defined causal ordering.

The second class is characterized by a "direct" approach defining an operational Statecharts semantics using labelled transition systems. Uselton and Smolka [20] again require a causality relation on events, whereas Maggiolo-Schettini et al. [12] use labels consisting of four-tuples which include complex and intricate information about causal orderings, global consistency, and negated events. Our present work also fits into this class. But in contrast to the previous work [20] and [12], our semantics does without an explicit causality relation and, moreover, is very concise.

5. CONCLUSIONS

The definition of a formal semantics for the Statecharts language has been a long-term challenge. A considerable number of proposals has been developed which differ in the Statecharts syntax to be supported and the set of semantical properties to be fulfilled.

We have presented a formal operational Statecharts semantics definition which combines causality and synchrony on the macro step level with compositionality on the level of micro step sequences. The essential additional property of this semantics definition is provided by its very conciseness.

References

[1] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

[2] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley, 1999.

[3] W. Damm, B. Josko, H. Hungar, and A. Pnueli. A compositional real-time semantics of STATEMATE designs. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *Lecture Notes in Computer Science*, pages 186–238. Springer-Verlag, September 1997.

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

[5] D. Harel and A. Naamad. The STATEMATE semantics of Statecharts. *ACM Transactions on Software Engineering*, 5(4):293–333, October 1996.

[6] D. Harel, A. Pnueli, J. Schmidt, and R. Sherman. On the formal semantics of Statecharts. In *Symposium on Logic in Computer Science (LICS '87)*, pages 56–64, Ithaca, NY, USA, June 1987. IEEE Computer Society Press.

[7] C. Huizing and R. Gerth. On the semantics of reactive systems. Technical report, Eindhoven University of Technology, 1990.

[8] D. Latella, I. Majzik, and M. Massink. Towards a formal operational semantics of UML Statechart diagrams. In *Formal Methods for Open Object-based Distributed Systems.* Chapman & Hall, 1999.

[9] F. Levi. *Verification of Temporal and Real-Time Properties of Statecharts.* PhD thesis, University of Pisa-Genova-Udine, Pisa, Italy, February 1997.

[10] G. Lüttgen, M. von der Beeck, and R. Cleaveland. A Compositional Approach to Statecharts Semantics. submitted for publication.

[11] G. Lüttgen, M. von der Beeck, and R. Cleaveland. Statecharts via process algebra. In J. C. M. Baeten and S. Mauw, editors, *Concurrency Theory (CONCUR '99)*, volume 1664 of *Lecture Notes in Computer Science*, Eindhoven, The Netherlands, August 1999. Springer-Verlag.

[12] A. Maggiolo-Schettini, A. Peron, and S. Tini. Equivalences of Statecharts. In U. Montanari and V. Sassone, editors, *CONCUR '96 (Concurrency Theory)*, volume 1119 of *Lecture Notes in Computer Science*, pages 687–702, Pisa, Italy, August 1996. Springer-Verlag.

[13] F. Maraninchi. Operational and compositional semantics of synchronous automaton compositions. In R. Cleaveland, editor, *CONCUR '92 (Concurrency Theory)*, volume 630 of *Lecture Notes in Computer Science*, pages 550–564, Stony Brook, NY, USA, August 1992. Springer-Verlag.

[14] E. Mikk, Y. Lakhnech, and M. Siegel. Hierarchical automata as model for Statecharts. In *Proceedings of Asian Computing Science Conference (ASIAN '97)*, volume 1345 of *Lecture Notes in Computer Science*. Springer-Verlag, December 1997.

[15] I. Paltor and J. Lilius. Formalising UML state machines for model checking. In R. France and B. Rumpe, editors, *UML'99 - The Unified Modeling Language. Beyond the Standard.*, volume 1723 of *LNCS*. Springer, 1999.

[16] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Department, Aarhus University, Denmark, 1981.

[17] A. Pnueli and M. Shalev. What is in a step: On the semantics of Statecharts. In T. Ito and A. Meyer, editors, *Theoretical Aspects of Computer Software (TACS '91)*, volume 526 of *Lecture Notes in Computer Science*, pages 244–264, Sendai, Japan, September 1991. Springer-Verlag.

[18] P. Scholz. *Design of Reactive Systems and their Distributed Implementation with Statecharts.* PhD thesis, Munich University of Technology, Munich, Germany, August 1998.

[19] B. Selic, G. Gullekson, and P. Ward. *Real-time Object Oriented Modeling and Design.* J. Wiley, 1994.

[20] A. Uselton and S. Smolka. A compositional semantics for Statecharts using labeled transition systems. In B. Jonsson and J. Parrow, editors, *CONCUR '94 (Concurrency Theory)*, volume 836 of *Lecture Notes in Computer Science*, pages 2–17, Uppsala, Sweden, August 1994. Springer-Verlag.

[21] A. Uselton and S. Smolka. A process-algebraic semantics for Statecharts via state refinement. In *Proceedings of PROCOMET '94.* North Holland/Elsevier, 1994.

[22] M. von der Beeck. A comparison of Statecharts variants. In H. Langmaack, W. Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT '94)*, volume 863 of *Lecture Notes in Computer Science*, pages 128–148, Lübeck, Germany, September 1994. Springer-Verlag.