Electronic Notes in Theoretical Computer Science 18 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume18.html 13 pages

Projectable semantics for Statecharts [1]

Andrea Maggiolo-Schettini and Simone Tini

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy.

Abstract

It has been proved that it is impossible to combine in one semantics for reactive systems the notions of modularity, causality and synchronous hypothesis. This limits bottom-up development of specifications. In this paper we introduce the notion of projectability, which is weaker than modularity, we define a non global consistent semantics for Statecharts that enforces projectability, causality and synchronous hypothesis, and we prove that no global consistent semantics for Statecharts can enforce these three notions.

1 Introduction

Synchronous languages [2,4] have been developed for the specification of *reactive systems* [7], namely systems that maintain an ongoing interaction with their environment at a rate controlled by this. Reactions to prompts from the environment are expected to happen in a bounded amount of time.

Synchronous languages are based on the *synchronous hypothesis* [3], namely the assumption that systems are able to react instantaneously to prompts from their environment. As a consequence, inputs from the environment and outputs of a system happen instantaneously.

In [10] properties of *causality* and *modularity* for formalisms that enforce the synchronous hypothesis, have been investigated. Causality means that for each event generated by a system at a particular moment there must be an event generated by its environment that directly or indirectly causes it. Causality ensures that reactive systems are really driven by their environment. Modularity means, firstly, that if two systems are put together to form a new one, they see each other behaviors as sequences of input-output pairs exactly as the environment sees them. No inner details of the execution of

[1] Research partially supported by CNR Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria" and by MURST Progetto "Tecniche Formali per la Specifica, l'Analisi, la Verifica, la Sintesi e la Trasformazione di Sistemi Software".

a system can be seen by the other. A second aspect is the uniformity of the view every subsystem has, namely that when an event is generated it is broadcast all around, and every subsystem has the same view at any moment. Finally, a reaction of the compound system is a combination of reactions of its subsystems. This means that the possible behaviors of a system are defined once and for all, and one can freely insert this in whatsoever context, being sure that it maintains its behaviors. This is needed to develop bottom-up specifications. Unfortunately, in [10] it is proved that synchronous hypothesis, causality and modularity cannot be combined in one semantics.

In this paper we introduce a notion weaker than that of modularity, the notion of *projectability*. Projectability does not require that the composition of subsystems is defined by abstracting from causality of their internal events, so one may combine synchronous hypothesis, projectability and causality. We investigate how these properties can be combined in the semantics of the synchronous formalism Statecharts [5].

2 Statecharts

Statecharts extend state-transitions diagrams with a tree-like structuring of states, explicit representation of parallelism, and broadcast communications among components. States at the bottom of the structure are basic states, states at intermediate levels are or-states and and-states. Or-states are states consisting of substates connected by directed edges, which represent activities to be performed in sequence. And-states are states representing activities to be performed in parallel. (In the diagram a dashed line separates the parallel substates.) The state at the top of the hierarchy is called the root-state. Let us consider the statechart in Fig. 1. The root-state $Central$ is an and-state consisting in two direct substates: $Left$ and $Right$. Both $Left$ and $Right$ are or-states. State $Left$ has the basic states L_Ready, L_Lock and L_Unlock as direct substates. Among the substates of an or-state, there is the default state, denoted with a dangling arrow. A transition between two states is labeled by a set of positive and negative signals, the trigger of the transition, and a set of positive signals, the action of the transition. Transition t_1 in Fig. 1 has both the positive signal $lock$ and the negative signal $\overline{unlock}$ as trigger and signal l_lock as action. Here we assume that source and target state of a transition are both immediate substates (in the tree-like structure) of the same or-state, namely transitions cannot cross borders of states.

Formally, a statechart z is a tuple

$$\langle S_z, \rho_z, \phi_z, \delta_z, T_z, in_z, out_z, \Pi_z, \chi_z \rangle$$

where:

(i) S_z is the non-empty, finite *set of states*.

(ii) $\rho_z : S_z \to 2^{S_z}$ is the *hierarchy function*; for $s \in S_z$, $\rho_z^*(s)$ denotes the least $S \subseteq S_z$ such that $s \in S$ and $\rho_z(s') \subseteq S$ for all $s' \in S$, and $\rho_z^+(s)$ denotes

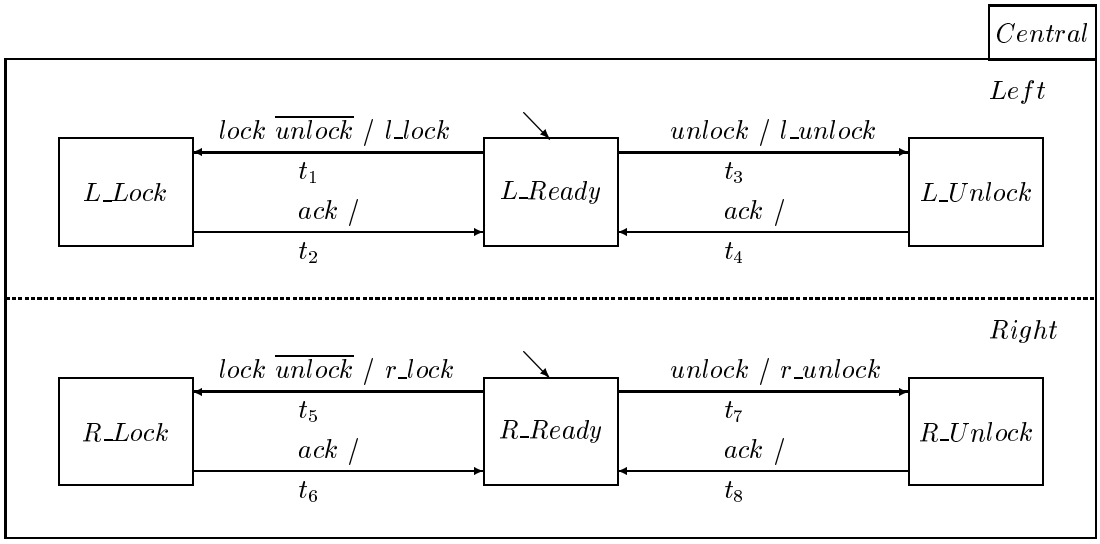

Fig. 1.

$\rho_z^*(s) - \{s\}$; ρ_z describes a tree-like structure, namely:
(a) There exists a unique $s \in S_z$, denoted $root_z$, s.t. $\rho_z^*(s) = S_z$.
(b) $s \notin \rho_z^+(s)$, for $s \in S_z$.
(c) If $\rho_z^*(s) \cap \rho_z^*(s') \neq \emptyset$, then either $s' \in \rho_z^*(s)$ or $s \in \rho_z^*(s')$, for $s, s' \in S_z$.
A state s is *basic* iff $\rho_z(s) = \emptyset$.

(iii) $\phi_z : S_z \to \{\text{OR, AND}\}$ is the (partial) *state type function* defined only for all non-basic states. States with type OR are called *or-states*, states with type AND are called *and-states*.

(iv) $\delta_z : S_z \to S_z$ is the (partial) *default function* defined only for or-states, so that $s' = \delta_z(s)$ implies that $s' \in \rho_z(s)$. For $s \in S_z$, $\delta_z^*(s)$ denotes the least $S \subseteq S_z$ s.t. $s \in S$, for each $s' \in S$ of type AND $\rho_z(s') \subseteq S$ and for each $s' \in S$ of type OR $\delta_z(s') \in S$.

(v) T_z is the finite *set of transitions*.

(vi) $in_z, out_z : T_z \to S_z - \{root_z\}$ are the *target* and the *source functions*. It is required that for each $t \in T_z$ there exists a state $s \in S_z$ such that $\phi_z(s) = OR$ and $in_z(t), out_z(t) \in \rho_z(s)$.

(vii) Π_z is the finite set of *signals*. For each $a \in \Pi_z$, $\overline{a}$ denotes the negation of a. For each $Y \subset \Pi_z$, $\overline{Y}$ is the set $\{\overline{a} | a \in Y\}$.

(viii) $\chi_z : T_z \to 2^{\Pi_z \cup \overline{\Pi_z}} \times 2^{\Pi_z}$ is the *labeling function*; the first component of $\chi_z(t)$ is denoted by $trigger(t)$ and is the *trigger* of t, the second component of $\chi_z(t)$ is denoted by $action(t)$ and is the *action* of t.

Given states s_1, s_2 of a statechart z, $lca_z(s_1, s_2)$ denotes the lowest common ancestor of s_1 and s_2, namely the state s such that $s_1, s_2 \in \rho_z^*(s)$, and for each $s' \neq s$ fulfilling the same requirement, $s \in \rho_z^+(s')$.

The limiting assumption that transitions do not cross borders of states

seems to be natural if one wants bottom-up development of specifications.

Given a state s, we denote by $trans(s)$ the set of all the transitions t such that both the source state and the target state of t are substates of s.

A transition is *triggered* by a set of broadcast signals if all positive signals of its trigger are broadcast and no signal appearing negated in the trigger is broadcast. A triggered transition may *fire* and broadcasts the signals in its action. Consider transitions t_1 and t_3 in Fig. 1. If signal *lock* is broadcast and signal *unlock* is not broadcast by the environment then t_1 is triggered, if *unlock* is broadcast then t_3 is triggered. The firing of t_1 (resp. t_3) implies the broadcasting of signal *l_lock* (resp. *l_unlock*).

The semantics of a statechart is given in terms of steps that take the statechart from a configuration to another.

A *configuration* of a statechart is a maximal set of states fulfilling the requirement that if an and-state is in the configuration then all its substates are in it, and if an or-state is in the configuration then exactly one of its substates is in it. The *default configuration* is the configuration such that for each or-state in it, its *default-state* is in the configuration. As an example, the set of states $\{Central, Left, L\_Ready, Right, R\_Ready\}$ is the default configuration of the statechart in Fig. 1. States in a configuration are said to be *active* in the configuration.

At each instant of time the environment prompts the statechart with a set of signals. Signals are assumed to be broadcast. The statechart reacts to a prompt from the environment by performing a set of transitions, called a *step*. According to the synchronous hypothesis principle a step is performed without consuming time. When a step $\mathcal{T}$ is performed from a configuration C, a new configuration $C' = (C - \bigcup_{t\in\mathcal{T}} \rho_z^*(out_z(t))) \cup \bigcup_{t\in\mathcal{T}} \delta_z^*((in_z(t))$ is entered. Configuration C' does not contain the source states of the transitions in $\mathcal{T}$ and contains the target states of the transitions in $\mathcal{T}$. In order to have finite reactions, it is required that for each pair of transitions t, t' in a step $\mathcal{T}$, t and t' are *consistent*, namely that there exist states s and s' with $t \in trans(s)$, $t' \in trans(s')$ and $\phi(lca(s, s')) = \text{AND}$. So, for each sequential component, at most one transition is in $\mathcal{T}$.

Now, since the introduction of the formalism, various semantics for Statecharts have been proposed. In [1] most of them are compared and related. The semantics proposed are either *non global consistent* (see the semantics in [8]) or *global consistent* (see the semantics in [15], [12], [13]), depending on the interpretation of negative signals.

In non global consistent semantics negation is interpreted as "not yet". Steps are computed as sequences of sets of transitions (microsteps) $\mathcal{T} = T_1, \ldots, T_k$. Note that all microsteps are performed in the same instant of time. This means that the sequence of microsteps does not correspond to a timing sequencing. Given a step $\mathcal{T} = T_1, \ldots, T_k$, it is required that transitions in $\mathcal{T}$ are pairwise consistent and transitions in T_{i+1} are triggered by signals broadcast by either the environment or by transitions in $T_1, \ldots, T_i$, for

$1 \leq i < k$. Now, transition t having $\overline{a}$ in its trigger and transition t' having a in its action can be in a step $\mathcal{T} = T_1, \ldots, T_k$, provided that t is in a microstep T_i and t' is in a microstep T_j, with $i < j$.

In global consistent semantics negation is interpreted as "never". Steps are computed as fixpoints of some equations and in a step there are never a transition t with $\overline{a}$ in its trigger and a transition t' broadcasting a.

In [9] it is argued that non global consistent semantics allow to distinguish clearly a cause from its effect, and therefore are more intuitive. The idea is that a sequence of microsteps defines a partial order among transitions, and this order reflects causality.

On the contrary, global consistent semantics allow to have a logical view of signals. Signals can be interpreted as boolean variables, and steps can be computed as solutions of sets of boolean equations. Causality is enforced by considering only minimal solutions. This approach needs rejecting programs giving rise to equation systems having no solution for some input.

We consider now the semantics in [8] and we explain the behavior of the statechart in Fig. 1. This statechart, which elaborates on a specification proposed in [14], describes a central locking system of a two-door car.

States $Left$ and $Right$ are the controllers of the left door and of the right door, respectively, and are active in parallel. We require that either both doors are locked or both doors are unlocked. Having one door locked and the other unlocked is considered to be an erroneous situation.

We explain the behavior of state $Left$. The behavior of $Right$ is analogous. Signals $lock$ and $unlock$ from the environment represent the request to lock and unlock doors, respectively. At the first instant of time states L_Ready and R_Ready are active. If signal $unlock$ is broadcast from the environment then transition t_3 is triggered and fires, so that signal l_unlock is broadcast and state L_Unlock is activated. We assume that l_unlock can be sensed by a motor that starts the unlocking operation of the left door. When this operation has been completed by the motor, then the motor broadcasts signal ack which triggers transition t_4. The firing of t_4 reactivates state L_Ready. Analogously, when L_ready is active, if signal $lock$ is broadcast by the environment and signal $unlock$ is not broadcast then transition t_1 fires and signal l_lock is broadcast. The motor can sense such signal and starts the locking operation of the left door.

We assume that the motor broadcast the signal ack only when it has completed all requested operations. This means that if the motor receives the request to lock both doors, then it will broadcast signal ack only when both doors have been locked.

Now, in the initial configuration, $Central$ can have only three possible reactions.

- If signal $unlock$ is broadcast, then both t_3 and t_7 fire, so that both doors are unlocked. Transitions t_4 and t_8 will be performed to react to the broad-

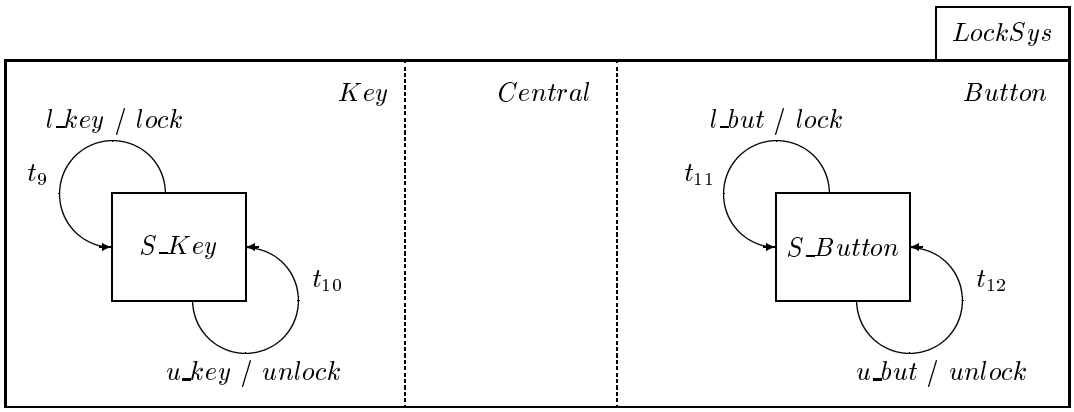

Fig. 2.

casting of signal *ack* by the motor, so that *L_Ready* and *R_Ready* will be reactivated in the same instant of time.

- If signal *lock* is broadcast and *unlock* is not broadcast, then both t_1 and t_5 fire, so that both doors are locked.
- If neither *lock* nor *unlock* is broadcast, then no transition fires.

Therefore we are sure that at each instant of time either both doors are locked or both doors are unlocked. It is reasonable requiring that this safety property is maintained when the statechart in Fig. 1 is inserted in a larger specification.

Let us consider the statechart in Fig. 2 obtained by composing in parallel the state *Central* in Fig. 1, state *Key* and state *Button*.

We assume here that doors can be locked and unlocked either from outside the car with a key or from inside the car by pushing a button. An attempt to lock (resp. unlock) the doors with the key implies the broadcasting of signal *l_key* (resp. *u_key*). Analogously, an attempt to lock (resp. unlock) the doors with the button implies the broadcasting of signal *l_but* (resp. *u_but*). States *Key* and *Button* are able to sense signals *l_key*, *u_key* and *l_but*, *u_but*, respectively, and to broadcast signals *lock* and *unlock*.

Now, let us suppose to have an attempt to lock the doors with the key and an attempt to unlock the doors with the button. In this case signals *l_key* and *u_but* are broadcast by the environment and are sensed by the statechart. One of the possible steps is the sequence of microsteps $\{t_9\}, \{t_1\}, \{t_{12}\}, \{t_7\}$. The set $\{t_9\}$ can be the first microstep as *l_key* is broadcast by the environment. Now, as transition t_9 broadcasts signal *lock*, transition t_1 is triggered and $\{t_1\}$ may be the second microstep. As *u_but* is in the environment, the third microstep may be $\{t_{12}\}$. The broadcasting of *unlock* by t_{12} triggers t_7, so that $\{t_7\}$ may be the last microstep. Therefore *Central* performs transitions t_1 and t_7, which means that the left door is locked and the right door is unlocked. Note that this is possible because signal *unlock* is broadcast by transition t_{12} during the computation of the step. If *Central* is considered in isolation, the fact cannot happen.

The example shows that we are not sure that when a statechart is inserted in a context, its behaviors are preserved. For the development of specifications

in a bottom-up fashion, one reasonably requires that subsystems perform the tasks for which they have been designed, and only these, when inserted in whatsoever context. If this requirement is satisfied, properties of safety proved for components are guaranteed to hold when such components are inserted into larger specifications. In the next section we formalize this idea by introducing the notion of projectability.

3 Projectability

Let us consider now the notions of causality and modularity of a reactive system, introduced in [10]. Let $S \stackrel{\langle I,O\rangle}{\longrightarrow} S'$ denote the fact that the reactive system S reacts instantaneously to input I by responding with output O, and by rewriting itself into S'. We denote by $S_1 \parallel S_2$ the parallel composition of S_1 and S_2.

A semantics is *causal* if for every reaction $S \stackrel{\langle I,O\rangle}{\longrightarrow} S'$ there exists a partial order $\leq$ over $I \cup O$ such that:

- if $I \neq \emptyset \neq O$ and $S \stackrel{\langle I',O\rangle}{\not\longrightarrow}$ for some $I' \subset I$, then there is at least one dependency between I and O, i.e. $\exists a \in I - I', b \in O$ with $a \leq b$;
- the ordering respects the composition of systems, i.e. if $S_1 \parallel S_2 \stackrel{\langle I,O\rangle}{\longrightarrow} S_1' \parallel S_2'$ with causal order $\leq$, then there exist systems $T_1, \ldots, T_n$ with $n \geq 2$ and causal orders $\leq_1, \ldots, \leq_n$ such that:
  - $T_1 \parallel \ldots \parallel T_n = S_1 \parallel S_2$,
  - $\leq \restriction (I_i \cup O_i) = \leq_i$,
  - $T_i \stackrel{\langle I_i,O_i\rangle}{\longrightarrow} T_i'$ for some T_i', and the union of these reactions is the reaction of $S_1 \parallel S_2$.

Causality means that for each event generated by the system there is a causal chain of events leading to it. The partial order $\leq$ that can be associated to a reaction $S \stackrel{\langle I,O\rangle}{\longrightarrow} S'$ gives this causal chain.

A semantics is *modular* iff the following condition holds:

$$(1) \qquad (S_1 \stackrel{\langle I\cup O_2,O_1\rangle}{\longrightarrow} S_1' \wedge S_2 \stackrel{\langle I\cup O_1,O_2\rangle}{\longrightarrow} S_2') \Leftrightarrow S_1 \parallel S_2 \stackrel{\langle I,O_1\cup O_2\rangle}{\longrightarrow} S_1' \parallel S_2'.$$

When S_1 and S_2 are composed in parallel, they see each other as a sequence of pairs $\langle I, O\rangle$, exactly as the environment sees them. The parallel composition of S_1 and S_2 is defined by considering only their input-output interface, namely both S_1 and S_2 are viewed as "black boxes", and no inner details of the execution of one of them is known by the other. Moreover, the output of a system is immediately available as input to the other. This implies the uniformity of the view every subsystem has of what is going on.

In [10] it is proved that modularity and causality cannot be combined with synchronous hypothesis. To see this fact, let us consider systems S_1 and S_2 such that $S_1 \stackrel{\langle\{a\},\{b\}\rangle}{\longrightarrow} S_1'$, $S_1 \stackrel{\langle\emptyset,\{b\}\rangle}{\not\longrightarrow}$, $S_2 \stackrel{\langle\{b\},\{a\}\rangle}{\longrightarrow} S_2'$ and $S_2 \stackrel{\langle\emptyset,\{a\}\rangle}{\not\longrightarrow}$. If causality holds

then there must be partial orders $\leq_1$ and $\leq_2$ such that $a \leq_1 b$, $b \leq_2 a$ and $\leq_i$ is associated to the reaction of S_i, $1 \leq i \leq 2$. By modularity we must have $S_1 \parallel S_2 \stackrel{\langle\emptyset,\{a,b\}\rangle}{\longrightarrow} S_1' \parallel S_2'$ and for this reaction no causal order exists that respects $\leq_1$ and $\leq_2$. Modularity implies a causal loop between a and b.

The semantics of Esterel [3] and Argos [11] are modular, and programs in which causality loops may occur are rejected.

Another aspect of modularity is that each reaction of the system $S_1 \parallel S_2$ is the union of a reaction of S_1 and a reaction of S_2. The consequence is that the semantics of S_1 (resp. S_2) viewed as a complete system is preserved when it runs in parallel with S_2 (resp. S_1). In this case we are sure that S_1 (resp. S_2) reaches configurations that are reachable also when it runs as a complete system.

The notion of projectability coincides with this aspect of modularity. Formally, a semantics is *projectable* iff the following condition holds:

$$S_1 \parallel S_2 \longrightarrow S_1' \parallel S_2' \Longrightarrow (S_1 \longrightarrow S_1' \wedge S_2 \longrightarrow S_2') \tag{2}$$

(which means, obviously, that a modular semantics is projectable but not viceversa).

In the case of Statecharts, we must take care of the hierarchy when defining the notion of projectability.

Definition 3.1 *A semantics for Statecharts is* projectable *iff given a statechart z and a step $\mathcal{T}$ from configuration C to configuration C', then for each state $s \in C \cap C'$, the set of transitions $\mathcal{T} \cap trans(s)$ is a step of the statechart having s as root-state.*

The definition above states that a step of a statechart consists in the union of steps of its components. According to this definition, the semantics in [8] is not projectable, as demonstrated with the example in Fig. 1 and Fig. 2.

4 Non global consistent semantics

In this section we define a non global consistent semantics for Statecharts enforcing projectability.

First of all we give our definition of microstep.

Definition 4.1 *For a statechart z in a configuration C, a sequence of (already) fired sets of transitions $\mathcal{T} = T_1, \ldots, T_k$ and a set of signals $\sigma \supseteq \bigcup_{t \in \mathcal{T}} action(t)$, a set T is a* microstep *iff:*

(i) *for each $t \in T$, t is triggered by σ;*

(ii) *for each $t, t' \in \mathcal{T} \cup T$, t and t' are consistent;*

(iii) *for each state $s \in C$, it holds that if $trans(s) \cap (\mathcal{T} \cup T) \neq \emptyset$ then there does not exist any transition t such that:*

(a) *$t \notin \mathcal{T} \cup T$ and t, t' are consistent for each $t' \in \mathcal{T} \cup T$;*

(b) *t is triggered by the set of signals $\Pi_z \cap (\bigcup_{t \in \mathcal{T} \cap trans(s)} (trigger(t) \cup$*

$action(t)) \cup \bigcup_{t \in T \cap trans(s)} trigger(t))$;

(c) $\exists a \in \Pi_z \mid (\overline{a} \in trigger(t) \wedge \exists t' \in \mathcal{T} \cap trans(s)$ *s.t.* $\overline{a} \in trigger(t') \wedge \exists t'' \in (\mathcal{T} \cup T) - trans(s)$ *s.t.* $a \in action(t''))$;

(iv) *for each state* $s \in C$, *it holds that if* $t' \in \mathcal{T} \cap trans(s)$, $t \in T \cap trans(s)$, $a \in \Pi_z \cap trigger(t)$, $\overline{a} \in trigger(t')$, *then there exists* $t'' \in \mathcal{T} \cap trans(s)$ *with* $a \in action(t'')$.

Configuration $C' = C - \{\rho_z^*(out_z(t)) \mid t \in T\} \cup \{\delta_z^*(in_z(t)) \mid t \in T\}$ *is* reached *from* C *by means of* T.

Condition *iii* ensures that given transitions $t, t' \in trans(s)$, both having $\overline{a} \in \overline{\Pi_z}$ in their trigger, $t' \in \mathcal{T} \cup T$, $t \notin \mathcal{T} \cup T$, t triggered by the set of signals $(\bigcup_{t \in \mathcal{T} \cap trans(s)}(trigger(t) \cup action(t)) \cup \bigcup_{t \in T \cap trans(s)} trigger(t)) \cap \Pi_z$, then there does not exist any transition $t'' \notin trans(s)$ with $t'' \in \mathcal{T} \cup T$ and $a \in action(t'')$. The reason is that if the statechart having s as root-state performs t' and transition t is triggered, then either a transition in $trans(s)$ broadcasts a signal that disallows t or t is performed. Condition *iv* ensures that given transitions $t, t' \in trans(s)$ with $t \in T$ and $t' \in \mathcal{T}$ and a signal $a \in \Pi_z$ with $a \in trigger(t)$ and $\overline{a} \in trigger(t')$, then signal a is broadcast by another transition of s. The reason is that if the statechart having s as root-state performs t', then it needs to perform a transition having a in its action in order to trigger t.

Let us consider the statechart in Fig. 2 in its initial configuration, the set of signals $\{l\_key, u\_but\}$ and the sequence of microsteps $\{t_9\}, \{t_1\}$. According to Def. 4.1, $\{t_{12}\}$ cannot be a microstep, contrarily to what happens if the semantics in [8] is assumed. The reason is that if we instantiate s, t, t', t'', a with $Central, t_5, t_1, t_{12}, unlock$ respectively, then condition *iii* is not respected. The new microstep must contain t_5, so that both doors are locked.

Definition 4.2 *Given configurations* $C_0, C_1, \ldots, C_n$, *a set of signals* σ, *sets of transitions* $T_1, \ldots, T_n$ *such that:*

(i) T_{i+1} *is a microstep for* z *in configuration* C_i, *sets of transitions* $T_1, \ldots, T_i$, *set of signals* $\sigma \cup \{actions(t) | t \in T_1 \cup \ldots \cup T_i\}$, $0 \leq i \leq n-1$;

(ii) C_{i+1} *is reached from* C_i *by means of* T_{i+1}, $0 \leq i \leq n-1$;

(iii) *there does not exist any microstep* $T \neq \emptyset$ *for* z *in configuration* C_n, *sets of transitions* $T_1, \ldots, T_n$, *set of signals* $\sigma \cup \{actions(t) | t \in T_1 \cup \ldots \cup T_n\}$,

$\mathcal{T} = T_1, \ldots, T_n$ *is a* step *for* z *in configuration* C_0, *and* C_n *is the configuration* reached *from* C_0 *by means of* $\mathcal{T}$.

In [10] it is proved that constructing steps as sequences of microsteps as in Def. 4.1 and Def. 4.2 ensures that the semantics enforces causality.

Proposition 4.3 *The semantics of definitions 4.1 and 4.2 is projectable.*

Proof. Let us suppose that $\mathcal{T} = T_1, \ldots, T_n$ is a step from configuration C of statechart z, with $s \in C$. Now, let us consider the set of signals $\sigma = \bigcup_{t \in \mathcal{T}} \{a \in (trigger(t) \cup action(t)) \cap \Pi_z \mid \nexists t' \in \mathcal{T} \cap trans(s). \overline{a} \in trigger(t')\}$. First of all

we prove that the set of transitions $T'_k = T_k \cap trans(s)$ is a microstep for the fired sets of transitions $\mathcal{T}'_k = T_1 \cap trans(s), \ldots, T_{k-1} \cap trans(s)$, for an arbitrary $1 \leq k \leq n$. The set T'_k satisfies condition i in Def. 4.1. In fact, if there exists a transition $t' \in T'_k$ such that t' is not triggered by $\sigma \cup \bigcup_{t \in \mathcal{T}'_k} action(t)$, then T_k does not satisfy condition iv of Def. 4.1. The fact that T_k satisfies conditions ii, iii, and iv of Def. 4.1 implies that T'_k satisfies the same conditions. Now, assume that the sequence of microsteps $T'_1, \ldots, T'_n$ is not maximal. Then there exists a transition $t \in trans(s)$, $t \notin \mathcal{T}$, which is triggered by $\sigma \cup \{a \mid \exists t' \in \mathcal{T} \cap trans(s). a \in action(t)\}$. So there must be $\overline{a} \in trigger(t) \cap \overline{\Pi_z}$ such that $a \in action(t')$ for some $t' \in \mathcal{T} - trans(s)$. Now there can be two cases:

(i) $\nexists t'' \in \mathcal{T} \cap trans(s)$ with $\overline{a} \in trigger(t'')$. In this case we put $\sigma = \sigma \cup \{a\}$ and reiterate the reasoning.

(ii) $\exists t'' \in \mathcal{T} \cup trans(s)$ with $\overline{a} \in trigger(t)$. In this case condition iii of Def. 4.1 is not satisfied for some microstep in $\mathcal{T}$. □

Following [16], we could easily give a compositional formalization of the semantics of definitions 4.1 and 4.2 by means of finite Labeled Transition Systems.

Note that in general compositionality does not imply projectability, as it is shown by the compositional semantics in [12], where a reaction of a compound system is obtained by combining "incomplete" reactions of its subsystems.

The non global consistent semantics in [6] can be easily shown to be projectable. However such semantics does not enforce the synchronous hypothesis as signals broadcast by transitions in a step can be sensed only in a successive step.

5 Global consistent semantics

In global consistent semantics a step $\mathcal{T}$ is computed as a fixpoint of an equation. We briefly explain the original global consistent semantics in [15].
Given a configuration C, let $Relevant(C)$ denote the set of transitions having source state in C.
Given a set of signals σ, let $Triggered(\sigma)$ be the set of transitions triggered by σ.
For a set of transitions T, let $Consistent(T)$ be the set of transitions consistent with each transition $t \in T$.
Now, given a configuration C and a set of signals σ, a step $\mathcal{T}$ is computed as the least set of transitions satisfying the following equation:

$$(3) \qquad \mathcal{T} = Consistent(\mathcal{T}) \cap Relevant(C) \cap Triggered(\sigma \cup \bigcup_{t \in \mathcal{T}} action(t)).$$

Note that all transitions in a step $\mathcal{T}$ must be triggered by signals broadcast by both the environment and transitions of $\mathcal{T}$.

z_1 z_2

Fig. 3.

As already noticed in [10] modularity lead to semantical problems if transitions can be triggered by negative signals. Assume to have systems S_1 and S_2 such that $S_1 \stackrel{\langle\emptyset,\{b\}\rangle}{\longrightarrow} S_1'$, $S_1 \stackrel{\langle\{a\},\{b\}\rangle}{\not\longrightarrow}$, $S_2 \stackrel{\langle\{b\},\{a\}\rangle}{\longrightarrow} S_2'$, and $S_2 \stackrel{\langle\emptyset,\{a\}\rangle}{\not\longrightarrow}$. If we have a modular semantics, then no reaction is defined when the compound system $S_1 \parallel S_2$ is prompted with the empty input. In fact, for $I = \emptyset$, there exist no O_1 and O_2 satisfying equation 1. In this case it is said that $S_1 \parallel S_2$ has a *non reactive* behavior, in the sense that the system is not able to respond to the environment.

Formally, we say that a semantics for reactive systems is *reactive* iff given a system S and an input I there exist S' and O such that $S \stackrel{\langle I,O\rangle}{\longrightarrow} S'$. In the case of Statecharts, reactivity means that given a statechart z in a configuration C and a set of signals σ, there exists a step from C for σ.

Esterel and Argos reject programs that may have non reactive behaviors.

The philosophy of Statecharts seems to be contrary to rejecting behaviors at the syntactical level. The original semantics of [15] does not enforce reactivity. Let us consider the statechart z obtained by composing in parallel statecharts z_1 and z_2 in Fig. 3. Let C be the initial configuration of z. Given the set of signals $\sigma = \emptyset$, there is no set of transitions satisfying equation 3. Note that statecharts having non reactive behaviors may be obtained composing statecharts having only reactive behaviors.

Two semantics have been proposed that enforce reactivity and global consistency (see [12] and [13]). According to the semantics of [12], statechart z as above would react to the empty input by performing step $\mathcal{T}_1 = \{t_1\}$. The approach in [13] implies that z reacts to the empty input by performing step $\mathcal{T}_2 = \emptyset$.

Note that transitions t_3 and t_4 are neither in $\mathcal{T}_1$ nor in $\mathcal{T}_2$, even though z_2 viewed in isolation must perform either t_3 or t_4. It follows that the semantics in [12] and [13] are not projectable.

We have a general negative result.

Proposition 5.1 *No global consistent semantics can enforce reactivity, causality, projectability and synchronous hypothesis.*

Proof. Let us consider the statecharts z_1 and z_2 in Fig. 3. If we consider z_1, for each input set of signals either t_1 or t_2 is triggered and therefore performed. Analogously, if we consider z_2, for each input set of signals either t_3 or t_4 is triggered and therefore performed. Now, let us consider the statechart z

obtained by composing z_1 and z_2 in parallel. Assume that z performs step $\mathcal{T}$ from its default configuration for the empty input set of signals. Projectability implies that each step $\mathcal{T}$ must satisfy the following condition: $\mathcal{T} \cap \{t_1, t_2\} \neq \emptyset$, $\mathcal{T} \cap \{t_3, t_4\} \neq \emptyset$. Global consistency implies that $\mathcal{T} \neq \{t_1, t_3\}$ and $\mathcal{T} \neq \{t_1, t_4\}$. Causality implies that $\mathcal{T} \neq \{t_2, t_3\}$ and $\mathcal{T} \neq \{t_2, t_4\}$. Therefore no step $\mathcal{T}$ exists and reactivity is not enforced. □

6 Conclusions

Semantics of formalisms for the specification of reactive systems must enforce causality, which ensures that systems are input driven, and reactivity, which ensures that systems are able to respond to external prompts. Modularity permits to compose systems considering only their input/output interface. As proved in [10], modularity and causality cannot be combined in semantics of synchronous formalisms.

In Esterel and Argos compositions of programs that may lead to non modular behaviors are rejected statically. In Statecharts modularity is sacrificed.

In this paper we have defined the property of projectability, which is weaker than modularity, and we have demonstrated with an example that it is needed for bottom-up development of specifications.

We have considered both global consistent and non global consistent semantics for Statecharts. Prop. 5.1 states that causality and projectability cannot be enforced by a global consistent semantics enforcing reactivity. We have defined a non global consistent semantics enforcing causality and projectability, as shown by Prop. 4.3. This semantics can be viewed as an improvement of the non projectable semantics originally proposed in [8]. The non global consistent semantics in [6], which can be proved to enforce projectability, does not allow instantaneous communications between components and therefore does not enforce the synchronous hypothesis.

References

[1] M. von der Beek: *A Comparison of Statecharts Variants.* In Proc. of FTRTFTS '94, Lecture Notes in Computer Science 863, pp. 128-148, Springer, Berlin, 1994.

[2] A. Benveniste and G. Berry, editors: *Another Look at Real-Time Systems.* Special Issue of Proceedings of the IEEE, September 1991.

[3] G. Berry and G. Gonthier: *The Esterel Synchronous Programming Language: Design, Semantics, Implementation.* Science of Computer Programming **19**, pp. 87-152, 1992.

[4] N. Halbwachs: *Synchronous Programming of Reactive Systems.* The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, 1993.

[5] D. Harel: *Statecharts: A Visual Formalism for Complex Systems.* Science of Computer Programming **8**, pp. 231-274, 1987.

[6] D. Harel and A. Naamad: *The Statemate Semantics of Statecharts.* ACM Transactions on Software Engineering Methods **5**:4, 1996.

[7] D. Harel and A. Pnueli: *On the Development of Reactive Systems.* In K.R. Apt, editor, Logic and Models of Concurrent Systems, NATO, ASI-13, pp. 477-498, Springer, Berlin, 1985.

[8] D. Harel, A. Pnueli, J.P. Schmidt and R. Sherman: *On the formal Semantics of Statecharts.* In Proc. Second IEEE Symp. on Logic in Computer Science, pp. 54-64, IEEE Computer Society Press, 1987.

[9] J.J.M. Hooman, S. Ramesh and W.P. de Roever: *A Compositional Axiomatization of Statecharts.* Theoretical Computer Science **101**, pp. 289-335, 1992.

[10] C. Huizing and R. Gerth: *Semantics of Reactive Systems in Abstract Time.* In Proc. of Real-Time: Theory and Practice, REX Workshop '91, Lecture Notes in Computer Science 600, pp. 291-314, Springer, Berlin, 1992.

[11] F. Maraninchi: *Operational and Compositional Semantics of Synchronous Automaton Composition.* In Proc. of CONCUR '92, Lecture Notes in Computer Science 630, pp. 550-564, Springer, Berlin, 1992.

[12] A. Maggiolo-Schettini, A. Peron and S. Tini: *Equivalence of Statecharts.* In Proc. of CONCUR '96, Lecture Notes in Computer Science 1119, pp. 687-702, Springer, Berlin, 1996.

[13] J. Philipps and P. Scholtz: *Compositional Specification of Embedded Systems with Statecharts.* In Proc. of TAPSOFT '97. Lecture Notes in Computer Science 1214, pp. 637-651, Springer, Berlin, 1997.

[14] J. Philipps and P. Scholtz: *Formal Verification of Statecharts with Instantaneous Chain Reactions.* In Proc. of TACAS '97, Lecture Notes is Computer Science 1217, Springer, Berlin, 1997.

[15] A. Pnueli and M. Shalev: *What is in a Step: on the Semantics of Statecharts.* In Proc. of TACS '91, Lecture Notes in Computer Science 526, pp. 244-264, Springer, Berlin, 1991.

[16] A.C. Uselton and S. A. Smolka: *A Process Algebraic Semantics for Statecharts.* In Proc. of CONCUR '94, Lecture Notes in Computer Science 836, pp. 2-17, Springer, Berlin, 1994.