Priorities in Statecharts*

Andrea Maggiolo-Schettini, Massimo Merro

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

Abstract. The paper examines the specification language Statecharts from the point of view of its capability to express notions of priority. First a version of Statecharts that does not support priority is introduced, then various syntactic and semantics extensions are examined and compared from the point of view of expressing a general notion of priority. Finally, a special kind of priority, interrupt, is investigated.

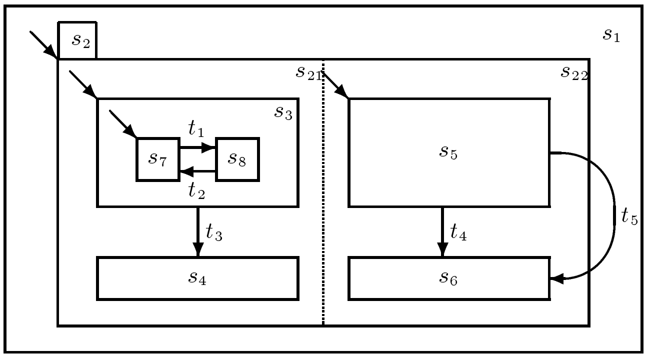
1 Introduction

Statecharts are a visual formalism, originally proposed in [6], for the specification of complex systems consisting of components running in parallel, reacting to prompts from an environment, and interacting with each other by broadcast. Statecharts have been successfully used to specify hardware components, communication networks, operating systems (e.g. [5, 7]).

Statecharts are state-transition diagrams enriched by a tree-like structuring of states, explicit representation of parallelism and communication among parallel components. States are either of type AND, *AND-states*, used to describe parallel behaviours, or of type OR, *OR-states*, used to describe sequential behaviours. The graphical convention is that states are represented by boxes, and the box of a substate of a state is drawn inside the area of the box of the state. The area of an AND-state is partitioned by dashed lines. Each element of the partition represents a parallel component. An OR- state has a privileged immediate substate (*default state*) that is pointed to by an arrow without source. A transition is labelled by a set of events and negated events (these to be interpreted as absence of events) enabling its performance (the *trigger*) and by a set of events that are communicated when the transition is performed (the *action*). An example of statechart is in Figure 1.

The semantics of a statechart is the following. It is assumed that the *environment* prompts the statechart by means of events that are related to a discrete time domain (for instance natural numbers). The history of the prompts of the environment is described as a sequence of sets of events $E_1, \ldots, E_i, \ldots$, and it is intended that the events in E_i are communicated at the $i-th$ instant of time. At the instant i the statechart "reacts" to the set of communicated events by performing a number of enabled transitions. The reaction (a *step*) is assumed to be immediate, namely input and corresponding output occur at the same time.

* Research partially supported by Esprit BRA 8130 LOMAPS.



$$
\begin{array}{l}
trigger_Z(t_1) = \{e_1\}\ action_Z(t_1) = \emptyset \\
trigger_Z(t_2) = \{e_2\}\ action_Z(t_2) = \{e_3\} \\
trigger_Z(t_3) = \{e_3\}\ action_Z(t_3) = \emptyset \\
trigger_Z(t_4) = \{e_1\}\ action_Z(t_4) = \{e_2\} \\
trigger_Z(t_5) = \{e_1\}\ action_Z(t_5) = \{e_2\}
\end{array}
$$

Fig. 1.

This assumption has been introduced in [3] as *synchrony hypothesis*. Now, in reality no action occurs simultaneously with the input that has caused it, but the synchrony hypothesis can be justified if it is considered an abstraction from the case that a reaction takes place in time but before the next input occurs. In this paper we adopt the microstep approach proposed first in ([8]), namely we assume that a step is divided in a number of *microsteps*. If the execution of a transition enables the execution of another, these two transitions will be taken in subsequent microsteps of the same step. In order to model the synchrony hypothesis, during the execution of microsteps, within a step, it is assumed that events from the environment are not sensed. In order to avoid the execution of an infinite number of transitions in a step, two feasible possibilities exist. A *step semantics* (like that in [8]) guarantees finiteness of a step by requiring that transitions in the step affect different components of an AND-state. An *instantaneous semantics* allows a step to contain chains of transitions in the same parallel component. In this case, finiteness of the step is guaranteed by requiring either that a state is not exited twice (see the semantics proposed in [11]) or that a transition is not performed twice (see the semantics proposed in [12]), in the step. In this paper we shall consider both step and instantaneous semantics.

Now, in a language for concurrent system specification, it is useful to express priorities between operations that can be (nondeterministically) chosen at an instant. So we investigate which syntactic and semantic extensions are needed in Statecharts to model transition based priorities. It is known that priorities can be modelled by negated events, but we show that this can be done only under some restrictive conditions. Therefore we consider an alternative way to express priorities, namely that of endowing statecharts by explicit priority relations between transitions. This approach allows us to do without negated events and to express, at the same time, a more general kind of priority. We believe that specifications without negated events are also more perspicuous.

Also the hierarchical structure of states induces forms of priority. Priorities of this type are those defined by Day ([4]) and by Pnueli and Shalev ([13]). We examine how priorities of this kind may be used to model interrupts. We define a priority relation motivated by this purpose, and we show that it is equivalent to Pnueli and Shalev's priority. This holds for both step and instantaneous semantics.

2 Statecharts: Syntax and Semantics

In this section we give a syntax for Statecharts in a restricted version with respect to [8]. Then we define a step stemantics and two instantaneous semantics. These three semantics are obtained by only varying a key definition.

We shall henceforward assume an infinite set $\mathcal{S}$ of states, an infinite set $\mathcal{T}$ of transitions and an infinite set $\mathcal{E}$ of events. We shall use $s, s', s_1, s_2, \ldots$ to range over $\mathcal{S}$, $t, t', t_1, t_2, \ldots$ to range over $\mathcal{T}$ and $e, e', e_1, e_2, \ldots$ to range over $\mathcal{E}$. Moreover we shall assume S as a general subset of $\mathcal{S}$, T as a general subset of $\mathcal{T}$ and E as a general subset of $\mathcal{E}$.

Definition 2.1 A *statechart* Z is a tuple $(S_Z, hrc_Z, type_Z, default_Z, E_Z, T_Z, lout_Z, lin_Z, hout_Z, hin_Z, label_Z)$, where:

1. $S_Z \subseteq \mathcal{S}$ is the finite non-empty set of *states.*
2. $hrc_Z : S_Z \to 2^{S_Z}$ is the *hierarchy function* that, for a state $s \in S_Z$, gives the set of immediate substates of s. A state $s \in S_Z$ is *basic* if $hrc_Z(s) = \emptyset$, *composite* otherwise. By hrc^*_Z and hrc^+_Z we denote the transitive-reflexive and the transitive-non-reflexive closure of hrc_Z, respectively, such that for $s \in S_Z$, the following predicates hold:
   (a) $s \in hrc^*_Z(s)$
   (b) $hrc_Z(s) \subseteq hrc^*_Z(s)$
   (c) $s' \in hrc_Z(s) \Rightarrow hrc_Z(s') \subseteq hrc^*_Z(s)$,
   (d) $hrc^+_Z(s) = hrc^*_Z(s) \setminus \{s\}$.
   There exists a unique state $s \in S_Z$, denoted $root_Z$, such that $hrc^*_Z(s) = S_Z$. Moreover for every $s \in S_Z$, $s \neq root_Z$, there exists a unique state $s' \in S_Z$, denoted $parent_Z(s)$, with $s \in hrc_Z(s')$, so that hrc_Z describes a tree-like structure. Finally, for a non-empty set of states $S \subseteq S_Z$, the *lowest common ancestor* (resp.: *strict lowest common ancestor*) of S, denoted $lca_Z(S)$ (resp.: $lca^+_Z(S)$), is the state $s \in S_Z$ such that:
   (a) $S \subseteq hrc^*_Z(s)$ (resp.: $S \subseteq hrc^+_Z(s)$)
   (b) $\forall s' \in S_Z \;\; S \subseteq hrc^*_Z(s') \Rightarrow s \in hrc^*_Z(s')$.
3. $type_Z : S_Z \to \{AND, OR\}$ is the *state-type function* that assigns a type to each composite state and is undefined for basic states. Immediate substates of AND-states cannot be AND-states.

4. $default_Z : S_Z \to S_Z$ is the *default function* that, for a state $s \in S_Z$ such that $hrc_Z(s) \neq \emptyset$ and $type_Z(s) = OR$, gives the default state of s, and is undefined otherwise.
5. $E_Z \subseteq \mathcal{E}$ is the finite set of *events*.
6. $T_Z \subseteq \mathcal{T}$ is the finite set of *transitions*.
7. $lout_Z : T_Z \to S_Z \setminus \{root_Z\}$ is a total function, called *lowest-source function*, s.t $lout_Z(t) = s$ iff transition t originates from state s. A transition cannot originate from an immediate substate of an AND-state, namely for every $t \in T_Z$ $type_Z(parent_Z(lout_Z(t))) = OR$.
8. $lin_Z : T_Z \to S_Z \setminus \{root_Z\}$ is a total function, called *lowest-target function*, s.t. $lin_Z(t) = s$ iff transition t ends in state s. Analogously to the lowest-source function, we have that for every $t \in T_Z$ $type_Z(parent_Z(lin_Z(t))) = OR$.
9. $hout_Z, hin_Z : T_Z \to S_Z \setminus \{root_Z\}$ are total functions, called *highest-source function* and *highest-target function* respectively, such that the following predicates hold:
   (a) $\forall\, t \in T_Z$ $lout_Z(t) \in hrc^*_Z(hout_Z(t))$
   (b) $\forall\, t \in T_Z$ $lin_Z(t) \in hrc^*_Z(hin_Z(t))$
   (c) $\forall t \in T_Z$ $parent_Z(hout_Z(t)) = parent_Z(hin_Z(t)) = s$ with $type_Z(s) = OR$.
10. $label_Z : T_Z \to 2^{E_Z} \times 2^{E_Z}$ is the *labelling function*. The two components of $label_Z(t)$ are called $trigger_Z(t)$ and $action_Z(t)$, respectively.

Intuitively $hout_Z(t) = lout_Z(t)$ (resp.: $hin_Z(t) = lin_Z(t)$) if t does not cross the border of any state, and $hout_Z(t) = s$ (resp.: $hin_Z(t) = s$) if t crosses the border of s with $lout_Z(t) \in hrc^+_Z(s)$ (resp.: $lin_Z(t) \in hrc^+_Z(s)$) and there is no $s' \in S_Z$ s.t. t crosses the border of s' and $s \in hrc^+_Z(s')$. The assumption that functions $hout_Z$ and hin_Z are total implies that transitions never cross the dashed line of an AND-state.
With reference to statechart Z in Figure 1 we have:

1. $S_Z = \{s_1, s_2, s_{21}, s_{22}, s_3, s_4, s_5, s_6, s_7, s_8\}$
2. $hrc_Z(s_2) = \{s_{21}, s_{22}\}$, $hrc_Z(s_3) = \{s_7, s_8\}$, $hrc^*_Z(s_2) = \{s_2, s_{21}, s_{22}, s_3, s_4, s_5, s_6, s_7, s_8\}$, $hrc^+_Z(s_2) = hrc^*_Z(s_2) \setminus \{s_2\}$, $parent_Z(s_{21}) = s_2$, $lca_Z(\{s_8, s_{22}\}) = s_2$, $lca^+_Z(\{s_8, s_{22}\}) = s_2$, $lca_Z(\{s_4, s_8, s_{21}\}) = s_{21}$, $lca^+_Z(\{s_4, s_8, s_{21}\}) = s_2$
3. $type_Z(s_2) = AND$, $type_Z(s_3) = OR$
4. $default_Z(s_{21}) = s_3$, $default_Z(s_3) = s_7$
5. $E_Z = \{e_1, e_2, e_3\}$
6. $T_Z = \{t_1, t_2, t_3, t_4, t_5\}$
7. $lout_Z(t_4) = hout_Z(t_4) = s_5$, $lin_Z(t_4) = hin_Z(t_4) = s_6$, $lout_Z(t_5) = s_5$, $hout_Z(t_5) = hin_Z(t_5) = s_2$, $lin_Z(t_5) = s_6$
8. $trigger_Z(t_1) = \{e_1\}, action_Z(t_1) = \emptyset, trigger_Z(t_2) = \{e_2\}, action_Z(t_2) = \{e_3\}$

We call $\mathcal{Z}$ the class of statecharts as in Definition 2.1, and we shall use $Z, Z', Z_1, Z_2, \ldots$ to range over $\mathcal{Z}$.

Now we are going to present a step semantics for $\mathcal{Z}$ closely related to the semantics of [8, 13]. From now on, if not specified otherwise, we assume a statechart $Z = (S_Z, hrc_Z, type_Z, default_Z, E_Z, T_Z, lout_Z, lin_Z, hout_Z, hin_Z, label_Z) \in \mathcal{Z}$.

Definition 2.2 Given a transition $t \in T_Z$, the *arena* of t, denoted $arena_Z(t)$, is the state $s \in S_Z$ such that $s = parent_Z(hout_Z(t)) = parent_Z(hin_Z(t))$.

With reference to statechart Z of Figure 1 we have $arena_Z(t_1) = s_3$, $arena_Z(t_4) = s_{22}$, $arena_Z(t_5) = s_1$. The concept of arena was introduced (but not formalized) in [13]. In the intention of the authors it represents the minimal context containing all the changes caused by the performance of transition. Note that for every $t \in T_Z$ $type_Z(arena_Z(t)) = OR$ by definition of $hout_Z$ and hin_Z.

Definition 2.3 Two transitions $t_1, t_2 \in T_Z$ are *structurally consistent* iff $type_Z(lca_Z(\{arena_Z(t_1), arena_Z(t_2)\})) = AND$. A set of transitions $T \subseteq T_Z$ is *structurally consistent* iff any two transitions $t_1, t_2 \in T$ are structurally consistent.

Two structurally consistent transitions are in two different parallel components of the same AND-state. This definition formalizes consistency of transitions as in [13]. It is more restrictive than that of [8], which could be obtained by substituting, in the definition above, $arena_Z(t)$ with $lca^+_Z(\{lout_Z(t), lin_Z(t)\})$. As an example, transitions t_3 and t_5 in Figure 1 are structurally consistent in the semantics of [8], but not in the semantics of [13] and in our semantics.

Definition 2.4 Two states $s_1, s_2 \in S_Z$ are *consistent*, if either $s_1 \in hrc^*_Z(s_2)$ or $s_2 \in hrc^*_Z(s_1)$ or $type_Z(lca_Z(\{s_1, s_2\})) = AND$. A set of states $S \subseteq S_Z$ is *consistent* iff any two states $s_1, s_2 \in S$ are consistent. A set of states $S \subseteq S_Z$ is *maximally consistent* iff for every $s \in S_Z \setminus S$, the set $S \cup \{s\}$ is not consistent.

With reference to the statechart of Figure 1, s_3 is consistent with s_5 and s_7 but not with s_4. The set $S = \{s_1, s_2, s_{21}, s_{22}, s_3, s_6, s_7\}$ is maximally consistent, $S \setminus \{s_{21}\}$ is only consistent and $S \cup \{s_4\}$ is not consistent.

Now we introduce the concepts of microconfiguration and microstep. A microstep within a step takes from a microconfiguration to a microconfiguration. A microconfiguration records the set of states exited and the set of transitions done in previous microsteps of the step, the set of current states and the set of events currently available.

Definition 2.5 A *microconfiguration* μC is a quadruple $(S_{exit}, T_{done}, S_{curr}, E_{curr})$ where:

1. $S_{exit} \subseteq S_Z$
2. $T_{done} \subseteq T_Z$
3. $S_{curr} \subseteq S_Z$ is maximally consistent
4. $E_{curr} \subseteq E_Z$.

Definition 2.6 Given a transition $t \in T_Z$ and a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$, t is *relevant* in μC iff $lout_Z(t) \in S_{curr}$, *triggered* in μC iff $trigger_Z(t) \subseteq E_{curr}$, and *consistent* in μC iff $T_{done} \cup \{t\}$ is structurally consistent.

Definition 2.7 A transition $t \in T_Z$ is *enabled* in a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ iff:

1. t is relevant in μC
2. t is triggered in μC
3. t is consistent in μC.

Definition 2.8 A *microstep* $\mu\Sigma$ from a microconfiguration μC is a non-empty set of transitions in T_z satisfying the following conditions:

1. $\forall\, t \in \mu\Sigma \quad t$ is enabled in μC
2. $\mu\Sigma$ is structurally consistent.

Definition 2.9 Given two states $s_1, s_2 \in S_Z$ with $s_2 \in hrc^*_Z(s_1)$, the *path* from s_1 to s_2, denoted by $path(s_1, s_2)$, is the set $\{s \in S_Z \mid s \in hrc^*_Z(s_1) \ \wedge \ s_2 \in hrc^*_Z(s)\}$.

With reference to the statechart in Figure 1, $path(s_1, s_8) = \{s_1, s_2, s_{21}, s_3, s_8\}$.

Definition 2.10 The *downward closure* of a set $S \subseteq S_Z$ denoted $downclos(S)$, is the least superset of S such that:

1. $(s \in downclos(S) \wedge type_Z(s) = AND) \ \Rightarrow \ hrc_Z(s) \subseteq downclos(S)$
2. $(s \in downclos(S) \wedge type_Z(s) = OR \wedge hrc_Z(s) \neq \emptyset \wedge hrc_Z(s) \cap S = \emptyset) \ \Rightarrow$ $default_Z(s) \in downclos(S)$.

With reference to the statechart of Figure 1, $downclos(\{s_2, s_6\}) = \{s_2, s_{21}, s_{22}, s_3, s_6, s_7\}$.

Definition 2.11 Given a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$, a transition $t \in T_Z$ relevant in μC and a microstep $\mu\Sigma \subseteq T_Z$, we define:

1. $exited(t, S_{curr}) = \{s \in S_Z \mid s \in hrc^*_Z(hout_Z(t)) \cap S_{curr}\}$
2. $entered(t) = downclos(path(hin_Z(t), lin_Z(t)))$
3. $next\mu C(\mu\Sigma, \mu C) = (S'_{exit}, T'_{done}, S'_{curr}, E'_{curr})$ with:
   (a) $S'_{exit} = S_{exit} \cup \bigcup_{t \in \mu\Sigma} exited(t, S_{curr})$
   (b) $T'_{done} = T_{done} \cup \mu\Sigma$
   (c) $S'_{curr} = (S_{curr} \setminus \bigcup_{t \in \mu\Sigma} exited(t, S_{curr})) \ \cup \ (\bigcup_{t \in \mu\Sigma} entered(t))$
   (d) $E'_{curr} = E_{curr} \cup \bigcup_{t \in \mu\Sigma} action_Z(t)$.

So, in a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$, $exited(t, S_{curr})$ and $entered(t)$ give the set of states exited and entered, respectively, when transition t is executed, and $next\mu C(\mu\Sigma, \mu C)$ gives the microconfiguration reached by the execution of the microstep $\mu\Sigma$ in μC.

Now we introduce the concepts of configuration and step, and define the semantics we have previously announced.

Definition 2.12 A *configuration* is a maximally consistent set of states. The set $downclos(\{root_Z\})$ is the *default configuration*.

Definition 2.13 Given an environment $E \subseteq E_Z$, an *admissible step* (also, briefly, a *step*) Σ from a configuration C to a configuration C' is a sequence of microsteps $\langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle$ such that:

1. $\mu C_0 = (\emptyset, \emptyset, C, E)$ is the initial microconfiguration
2. $\mu\Sigma_i$ is a microstep from $\mu C_i = (S^i_{exit}, T^i_{done}, S^i_{curr}, E^i_{curr})$, for $0 \leq i \leq n$
3. $\mu C_{i+1} = next\mu C(\mu\Sigma_i, \mu C_i)$, for $0 \leq i \leq n$
4. $\forall\, t \in T_Z$ t is not enabled in $\mu C_{n+1} = (S^{n+1}_{exit}, T^{n+1}_{done}, S^{n+1}_{curr}, E^{n+1}_{curr})$
5. $C' = S^{n+1}_{curr}$, denoted $nextC(\Sigma)$.

A step contains a finite number of transitions. In fact, by restriction 3 of definition 2.7, a step contains only transitions affecting different parallel components, and the number of these is finite. Furthermore, condition 4 of definition 2.13 ensures that a step is a maximal set of enabled transitions. We define now a semantic function.

Definition 2.14 The step semantic function $Sem^{step} : \mathcal{Z} \times 2^{\mathcal{S}} \times 2^{\mathcal{E}} \rightarrow 2^{(2^{\mathcal{T}})^*}$ is such that for a statechart $Z \in \mathcal{Z}$, a configuration C for Z and an environment $E \subseteq E_Z$, $Sem^{step}[\![Z]\!]CE = \{\Sigma_1, \ldots, \Sigma_k\}$, where $\Sigma_1, \ldots, \Sigma_k$ are all the admissible steps from the configuration C in the environment E.

We call this semantics *step semantics*. It differs from that of [8] only by the concept used of structural consistency of two transitions, as we have already pointed out. In a step semantics it is not possible, in a step, to exit a state entered in the same step (namely in the same instant of time). There are semantics, like those introduced in [11, 12], called instantaneous, that allow, in a step, chains of transitions inside the same parallel component. Our semantics can easily become instantaneous by weakening the condition for consistency of a transition t in a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ in Definition 2.6, namely, by replacing the condition either by $exited(t, S_{curr}) \cap S_{exit} = \emptyset$ or by $t \notin T_{done}$.

Proposition 2.1 For a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ of Z and a transition $t \in T_Z$ relevant in μC:

$$T_{done} \cup \{t\} \text{ structurally consistent } \Rightarrow\ exited(t, S_{curr}) \cap S_{exit} = \emptyset \ \Rightarrow\ t \notin T_{done}.$$

■

The converse implications does not hold.
So similarly to Sem^{step}, we can define semantic functions Sem^{ch1} and Sem^{ch2} by assuming the first and the second of the two new conditions of consistency of transitions, respectively. We call these new semantics *chain semantics of type 1* and *chain semantics of type 2*, respectively. In both the cases infinite steps are avoided, in the former because a state cannot be exited twice in a step, in the latter because a transition cannot appear twice in a step.
Now, let us consider the statechart Z of Figure 1 in the default configuration C_0 and in the environment $E = \{e_1\}$. In this case we have:
$Sem^{step}[\![Z]\!]C_0E = \{\langle\{t_1, t_4\}\rangle, \langle\{t_1\}, \{t_4\}\rangle, \langle\{t_4\}, \{t_1\}\rangle, \langle\{t_5\}\rangle\}$
$Sem^{ch1}[\![Z]\!]C_0E = \{\langle\{t_1, t_4\}, \{t_2\}\rangle, \langle\{t_1\}, \{t_4\}, \{t_2\}\rangle, \langle\{t_4\}, \{t_1\}, \{t_2\}\rangle, \langle\{t_1\}, \{t_5\}\rangle, \langle\{t_5\}\rangle\}$
$Sem^{ch2}[\![Z]\!]C_0E = \{\langle\{t_1, t_4\}, \{t_2\}, \{t_3\}\rangle, \langle\{t_1\}, \{t_4\}, \{t_2\}, \{t_3\}\rangle, \langle\{t_4\}, \{t_1\}, \{t_2\}, \{t_3\}\rangle, \langle\{t_1\}, \{t_5\}\rangle, \langle\{t_5\}, \{t_1\}, \{t_2\}, \{t_3\}\rangle\}$.

The step semantics allows, in a step, only transitions in parallel components. In the initial configuration, with the given environment, t_1, t_4 and t_5 are all enabled, but, while t_1 and t_4 are structurally consistent and can be performed in the same step, t_5 must be performed alone because t_5 and t_1 are not structurally consistent. Here we see that performances of t_4 and t_5, which have the same $trigger$, $action$, $lout$ and lin, but not the same $hout$ and hin, give rise to different behaviours. In fact, the performance of t_5, differently with respect to that of t_4, causes exiting and then reinitializing s_{21} together with exiting s_{22} and then reentering it in its substate s_6. The chain semantics of type 1 allows, in a step, chains of transitions in the same parallel component, such as t_1 and t_2 in state s_3, but it excludes the execution of transitions from states that have been reentered. In chain semantics of type 2 it is only required that a transition is not repeated in a step so that, as an example, t_3 can be performed after performing t_2.

3 Priority and negated events

Statecharts are, in general, nondeterministic descriptions. Nondeterminism arises from the fact that there may be more transitions that are contemporaneously enabled but cannot be performed contemporaneously because they are mutually exclusive, so that one of them is supposed to be chosen nondeterministically. Otherwise, a priority may be assigned to transitions to solve conflicts.

Definition 3.1 Two transitions $t_1, t_2 \in T_Z$ are *potentially conflicting* if

1. $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent
2. t_1, t_2 are not structurally consistent.

Definition 3.2 A *priority relation* on a statechart Z is a partial relation $\mathcal{R} \subseteq T_Z \times T_Z$ such that $(t_1, t_2) \in \mathcal{R}$ implies that t_1 e t_2 are potentially conflicting.

The meaning of $\mathcal{R}$ is that if transitions t_1, t_2 are both enabled and $(t_1, t_2) \in \mathcal{R}$ then t_1 has lower priority than t_2. Note that we do not assume either irreflexivity or antisymmetry or transitivity, and this may give rise to circularity. This situation is acceptable because in a configuration not all the transitions in conflict are necessarily triggered. Furthermore, note that a possible deadlock is limited to the current step.

Pnueli and Shalev ([13]) show how priority can be expressed by allowing a transition to be triggered by the absence of a specific event among the ones currently communicated. The absence of an event e is represented by its negation $\overline{e}$, and for $E \subseteq \mathcal{E}$, $\overline{E}$ denotes the set $\{\overline{e} \mid e \in E\}$. It also assumed that, for all $e \in \mathcal{E}$, $\overline{\overline{e}} = e$. So we extend the class of statecharts $\mathcal{Z}$ by assuming in definition 2.1, condition 10, $label_Z : T_Z \rightarrow 2^{E_Z \cup \overline{E_Z}} \times 2^{E_Z}$. We call the new class of statecharts $\mathcal{Z}_{neg}$. Consequently, in definition 2.6 we assume that a transition $t \in T_Z$ is triggered in $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ iff:

1. $trigger_Z(t) \cap E_Z \subseteq E_{curr}$

2. $\overline{(trigger_z(t) \cap \overline{E_Z})} \cap E_{curr} = \emptyset$.

So, by replacing $\mathcal{Z}$ with $\mathcal{Z}_{neg}$, we can define new semantic functions Sem^{step}_{neg}, Sem^{ch1}_{neg}, Sem^{ch2}_{neg}.
Now we show how it is possible to model a priority relation by negated events.

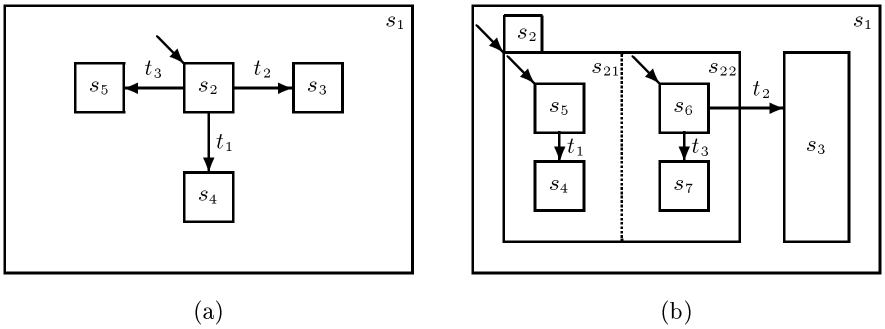


Fig. 2.

Let us consider the statechart of Figure 2(a) assuming $trigger_Z(t_1) = \{e_1\}$, $trigger_Z(t_2) = \{e_2\}$, $trigger_Z(t_3) = \{e_3\}$. Transitions t_1, t_2, t_3 are potentially conflicting with each other. Therefore if we want to model a priority relation $\mathcal{R}$ such that $(t_1, t_2) \in \mathcal{R}$ and $(t_2, t_3) \in \mathcal{R}$ but $(t_1, t_3) \notin \mathcal{R}$ (namely $\mathcal{R}$ is not transitive) we must change the triggers of transitions t_1 and t_2 in $\{e_1, \overline{e_2}\}$ and $\{e_2, \overline{e_3}\}$ respectively. So if the statechart is in configuration $\{s_1, s_2\}$ and events e_1, e_2 are supplied, then t_2 is performed, whereas if only events e_2, e_3 are supplied then transition t_3 is performed, and, finally, if only events e_1 and e_3 are supplied the choice between t_1 and t_3 is non-deterministic.
In our opinion this method does not always work. As an example, let us consider the statechart in Figure 2(b) with $trigger_Z(t_1) = \{e_1\}$ and $trigger_Z(t_2) = \{e_2\}$. Transitions t_1 and t_2 are potentially conflicting, and therefore in order to model a priority relation $\mathcal{R} = \{(t_1, t_2)\}$, we should only insert in $trigger_Z(t_1)$ the negated event $\overline{e_2}$. But, in this way, if we assume the configuration $C = \{s_1, s_2, s_{21}, s_{22}, s_5, s_7\}$ and the environment $E = \{e_1, e_2\}$, transition t_1 could not be executed because it would not be triggered, against our intention. This is due to the fact that we augment the trigger of t_1 by $\overline{e_2}$ independently of the fact that t_2 is relevant or not, whereas the priority between two transition plays a role only when both of them are relevant and triggered. So negated events can support correctly a priority relation $\mathcal{R}$ only if $(t_1, t_2) \in \mathcal{R}$ implies that, for every microconfiguration μC, if t_1 is relevant in μC also t_2 is relevant in μC.

We have seen how negated events allow us to implement a priority relation on transitions and with what limitations this can be done. On the other hand, statecharts with negated events are not always perspicuous, and one may prefer to endow a statechart directly with a relation expressing the priority wanted. So we consider statecharts $Z = (S_Z, hrc_Z, label_Z, default_Z, E_Z, T_Z, lout_Z, lin_Z, hout_Z, hin_Z, label_Z, \prec_Z)$, where S_Z, hrc_Z, $label_Z$, $default_Z$, E_Z, T_Z, $lout_Z$, lin_Z, $hout_Z$, $hin_Z, label_Z$ are as in definition 2.1, and $\prec_Z \subseteq T_Z \times T_Z$ is a priority relation where $t \prec_Z t'$ implies t and t' potentially conflicting. We call $\mathcal{Z}_{prior}$ the class of these statecharts. Moreover, if we add the following condition of enabling in definition 2.7 :

$$4. \; \forall\, t' \in T_Z \;\; t' \text{ relevant, triggered and consistent in } \mu\mathrm{C} \;\Rightarrow\; t \not\prec_Z t'$$

by replacing $\mathcal{Z}$ with $\mathcal{Z}_{prior}$, we can define new semantic functions Sem^{step}_{prior}, Sem^{ch1}_{prior} and Sem^{ch2}_{prior}.

Now we shall see that, under suitable hypotheses, statecharts with negated events can be transformed into statecharts with priority and vice versa, while preserving semantics. To this purpose we define two transformations $TRANS1 : \mathcal{Z}_{prior} \to \mathcal{Z}_{neg}$ and $TRANS2 : \mathcal{Z}_{neg} \to \mathcal{Z}_{prior}$.

Let us see first the transformation $TRANS1$. For $Z \in \mathcal{Z}_{prior}$, the idea of the construction of $Z' = TRANS1(Z)$, inspired by [13], is the following. If we have $t_1 \prec_Z t_2^{(j)}$ for $t_1, t_2^{(j)} \in T_Z$, $1 \leq j \leq m$, transition t_1 is splitted into k transitions $t_1^{(1)}, \ldots, t_1^{(k)} \in T_{Z'}$ such that at least one of them is triggered when all transitions $t_2^{(1)}, \ldots, t_2^{(m)}$ are not triggered, and all $t_1^{(1)}, \ldots, t_1^{(k)}$ are not triggered if there is at least a transition $t_2^{(j)}$ triggered.
Given a statechart $Z \in \mathcal{Z}_{prior}$ let us assume:

1. $REM_Z = \{t_1 \in T_Z \mid \exists\, t_2 \in T_Z \text{ s.t. } t_1 \prec_Z t_2,\, trigger_Z(t_2) \subseteq trigger_Z(t_1)\}$. If we assume that, for every two transitions $t_1, t_2 \in T_Z$, $t_1 \prec_Z t_2$ always implies that for every microconfiguration μC, when t_1 is relevant in μC also t_2 is relevant in μC, then transitions in REM_Z can be never executed, and so they can be removed.
2. $REP_Z = \{t_1 \in T_Z \setminus REM_Z \mid \exists\, t_2 \in T_Z \text{ s. t. } t_1 \prec_Z t_2\}$, the set of transitions that must be splitted.
3. $transsplit_Z(t) : REP_Z \to 2^{\mathcal{T}}$, the *transition splitting function* such that $transsplit_Z(t)$ is the set of transitions into which t is splitted.
4. $\mathcal{B}_Z$, the set of boolean expressions over E_Z constructed by means of the operators *and*, *or* and *negation*, such that, given a microconfiguration $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ and an event $e \in E_Z$, the atomic expression e is true iff $e \in E_{curr}$. We shall use $b, b', b_1, b_2, \ldots$ to range over $\mathcal{B}_Z$. Let us remember that in original Harel's statecharts triggers of transitions are boolean expressions over $\mathcal{E}$ and that in [13] Pnueli and Shalev give the equivalent representation of triggers by means of sets of events, that we have used in our syntax.
5. $lockexpr_Z(t) : REP_Z \to \mathcal{B}_Z$, the *lock expression function* such that *lock* $expr_Z(t) = b_1 \vee \ldots \vee b_{m(t)}$ is the disjunctive normal form of the logical formula

$$\bigwedge_{t \prec_Z t_1} (\overline{\bigwedge_{e \in trigger_Z(t_1)} e}).$$

So, given a microconfiguration μC, $lockexpr_Z(t)$ has value true iff for every $t_1 \in T_Z$ such that $t \prec_Z t_1$, t_1 is not triggered in μC, false otherwise.

6. $lockexprsplit_Z : REP_Z \to 2^{2^{\overline{E_Z}}}$, the *lock expression splitting function* s.t. $lockexprsplit_Z(t) = \{\overline{E_1}, \ldots, \overline{E_{m(t)}}\}$ with $\overline{E_i} = \{\overline{e} \in \overline{E_Z} \mid \overline{e}$ is a conjunct of b_i with b_i the i-th disjunct of $lockexpr_Z(t)\}$, for $1 \leq i \leq m(t)$. For $t \in REP_Z$ the set $lockexprsplit_Z(t)$ bijectively corresponds to $transsplit_Z(t)$ with $f_Z(t) : transsplit_Z(t) \to lockexprsplit_Z(t)$ such a bijection. The intuition is that each $\overline{E_i}$ contains the negated events that, together with $trigger_Z(t)$, constitute the trigger of the i-th of the transitions into which t is splitted.

The statechart $Z' = TRANS1(Z)$ is defined as follows:

1. $S_{Z'} = S_Z$
2. $hrc_{Z'} = hrc_Z$
3. $type_{Z'} = type_Z$
4. $default_{Z'} = default_Z$
5. $E_{Z'} = E_Z$
6. $T_{Z'} = ((T_Z \setminus REP_Z) \setminus REM_Z) \cup \bigcup_{t \in REP_Z} transsplit_Z(t)$
7. $lout_{Z'}(t') = lout_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
   $lout_{Z'}(t') = lout_Z(t)$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$
8. $lin_{Z'}(t') = lin_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
   $lin_{Z'}(t') = lin_Z(t)$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$
9. $hout_{Z'}(t') = hout_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
   $hout_{Z'}(t') = hout_Z(t)$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$
10. $hin_{Z'}(t') = hin_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
    $hin_{Z'}(t') = hin_Z(t)$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$
11. $trigger_{Z'}(t') = trigger_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
    $trigger_{Z'}(t') = trigger_Z(t) \cup f_Z(t)(t')$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$
12. $action_{Z'}(t') = action_Z(t')$ for $t' \in (T_Z \setminus REP_Z) \setminus REM_Z$
    $action_{Z'}(t') = action_Z(t)$ for $t \in REP_Z$, $t' \in transsplit_Z(t)$

As an example, let us consider the statechart $Z \in \mathcal{Z}_{prior}$ of Figure 3(a) with $trigger_Z(t_1) = \{e_1\}$, $trigger_Z(t_2) = \{e_2, e_3\}$, $trigger_Z(t_3) = \{e_3, e_4\}$, $action_Z(t_i) = \emptyset$, for $1 \leq i \leq 3$, $t_1 \prec_Z t_2$, $t_1 \prec_Z t_3$. We have $REM_Z = \emptyset$, $REP_Z = \{t_1\}$, $transsplit_Z = \{(t_1, \{t_1^{(1)}, t_1^{(2)}, t_1^{(3)}, t_1^{(4)}\})\}$, $lockexpr_Z = \{(t_1, (\overline{e_2} \wedge \overline{e_3}) \vee \overline{e_3} \vee (\overline{e_2} \wedge \overline{e_4}) \vee (\overline{e_3} \wedge \overline{e_4}))\}$, $lockexprsplit_Z = \{(t_1, \{\{\overline{e_2}, \overline{e_3}\}, \{\overline{e_3}\}, \{\overline{e_2}, \overline{e_4}\}, \{\overline{e_3}, \overline{e_4}\}\})\}$, $f_Z(t_1) = \{(t_1^{(1)}, \{\overline{e_2}, \overline{e_3}\}), (t_1^{(2)}, \{\overline{e_3}\}), (t_1^{(3)}, \{\overline{e_2}, \overline{e_4}\}), (t_1^{(4)}, \{\overline{e_3}, \overline{e_4}\})\}$. So, if we apply the transformation $TRANS1$ to Z, we obtain the statechart $Z' \in \mathcal{Z}_{neg}$ of Figure 3(b) with $trigger_{Z'}(t_1^{(1)}) = \{e_1, \overline{e_2}, \overline{e_3}\}$, $trigger_{Z'}(t_1^{(2)}) = \{e_1, \overline{e_3}\}$ $trigger_{Z'}(t_1^{(3)}) = \{e_1, \overline{e_2}, \overline{e_4}\}$, $trigger_{Z'}(t_1^{(4)}) = \{e_1, \overline{e_3}, \overline{e_4}\}$, $trigger_{Z'}(t_2) = \{e_2, e_3\}$, $trigger_{Z'}(t_3) = \{e_3, e_4\}$, $action_{Z'}(t') = \emptyset$, for every $t' \in T_{Z'}$.

We want to show that, under suitable hypotheses , $TRANS1$ is a semantics preserving transformation. It is immediate to prove the following proposition.

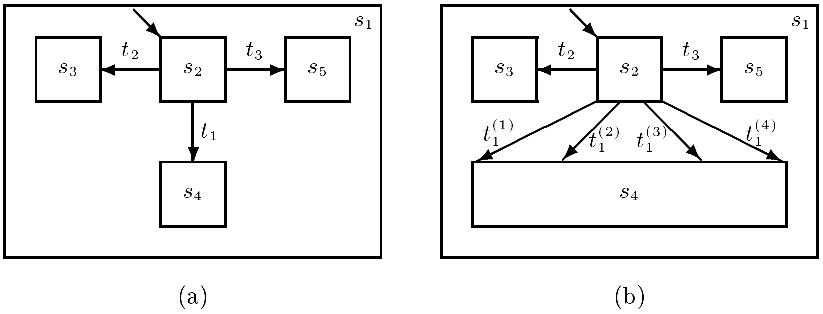


Fig. 3.

Proposition 3.1 Given transitions $t_1, t_2 \in T_Z$, for every microconfiguration μC t_1 is relevant in μC implies t_2 relevant in μC iff $lout_Z(t_1) \in hrc_Z^*(lout_Z(t_2))$. ■

We need a notation and a lemma.

Notation. Given $Z \in \mathcal{Z}_{prior}$ and $Z' = TRANS1(Z)$, if $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ is a microconfiguration for Z and $\mu C' = (S'_{exit}, T'_{done}, S'_{curr}, E'_{curr})$ is a microconfiguration for Z', we write $\mu C \doteq \mu C'$ iff:

1. $S'_{exit} = S_{exit}$
2. there exists a bijection $h_1 : T_{done} \rightarrow T'_{done}$ such that for $t \in T_{done}$ either $h_1(t) = t$ if $t \in (T_Z \setminus REP_Z) \setminus REM_Z$ or $h_1(t) = t'$ if $t \in REP_Z$ with $t' \in transsplit_Z(t)$
3. $S'_{curr} = S_{curr}$
4. $E'_{curr} = E_{curr}$.

Lemma 3.1 Given $Z \in \mathcal{Z}_{prior}$, s.t. $t_1 \prec_Z t_2$ implies $lout_Z(t_1) \in hrc_Z^*(lout_Z(t_2))$, and $Z' = TRANS1(Z)$, let us assume either the step semantics or the chain semantics of type 1. If Z and Z' are in two microconfigurations $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ and $\mu C' = (S'_{exit}, T'_{done}, S'_{curr}, E'_{curr})$, respectively, such that $\mu C \doteq \mu C'$, for every $t \in T_Z$:

1. t enabled in $\mu C \wedge t \notin REP_Z \Leftrightarrow t \in T_{Z'} \wedge t$ enabled in $\mu C' \wedge trigger_{Z'}(t) \subseteq E_{Z'}$
2. t enabled in $\mu C \ \wedge \ t \in REP_Z \ \Leftrightarrow \exists\, t^{(j)} \in transsplit_Z(t)$ enabled in $\mu C'$.

Proof.
See appendix. ■

Now we can enunciate the following theorem.

Theorem 3.1 Given $Z \in \mathcal{Z}_{prior}$, s.t. $t_1 \prec_Z t_2$ implies $lout_Z(t_1) \in hrc_Z^*(lout_Z(t_2))$, $Z' = TRANS1(Z)$, a configuration C and an environment E, there exists a total function $p : Sem_{prior}^{step}[\![Z]\!]CE \rightarrow 2^{Sem_{neg}^{step}[\![Z']\!]CE}$ such that:

1. $\forall\, \Sigma \in Sem_{prior}^{step}[\![Z]\!]CE \ \ \forall\, \Sigma' \in p(\Sigma)\ nextC(\Sigma) = nextC(\Sigma')$

2. $\forall\, \Sigma' \in Sem^{step}_{neg}[\![Z']\!]CE \;\; \exists\, \Sigma \in Sem^{step}_{prior}[\![Z]\!]CE \;\; \Sigma' \in p(\Sigma)$.

Proof.
See appendix. ∎

An analogous result can be obtained by assuming the chain semantics of type 1, but it does not hold if one assumes the chain semantics of type 2. Actually, in case of statecharts with cycles (a cycle is a sequence of transitions $t_1, \ldots, t_k$ where $lout_Z(t_{i+1}) \in entered_Z(t_i)$, $1 \leq i \leq k-1$, and $lout_Z(t_1) \in entered_Z(t_k)$), if a transition of the cycle is splitted by $TRANS1$ in more transitions, two or more of these might be executed in the same step and the equivalence would not hold anymore. Therefore we are able to prove equivalence for the chain semantics of type 2 only under the further assumption that, if a statechart has a cycle, any transition of the cycle has at most a transition with higher priority, and in this case such transition has at most one triggering event.
Note that when we have a statechart $Z \in \mathcal{Z}_{prior}$ with two transitions $t_1, t_2 \in T_Z$ s.t. $trigger_Z(t_2) \subseteq trigger_Z(t_1)$ and $t_1 \prec_Z t_2$ (under the hypothesis that $lout_Z(t_1) \in hrc^*_Z(lout_Z(t_2))$, t_1 will never be performed, because it will never be enabled. Correspondingly, in $Z' = TRANS1(Z)$ all the transitions in $transsplit_Z(t_1)$ have triggers containing both e and $\overline{e}$ for some $e \in E_{Z'}$. Obviously such transitions will never be triggered and, therefore, will never be performed, as we want.

Let us see now the transformation $TRANS2$. For $Z \in \mathcal{Z}_{neg}$, the idea of the construction of $Z' = TRANS2(Z)$ is the following. We replace every transition $t \in T_Z$, with negated events in its trigger, by a transition $t' \in T_{Z'}$ that differs from t only because the negative events of the trigger are dropped. Moreover, for every negated event $\overline{e} \in trigger_Z(t)$ we add a new transition $t'' \in T_{Z'}$ with $lout_{Z'}(t'') = lin_{Z'}(t'') = hout_{Z'}(t'') = hin_{Z'}(t'') = lout_Z(t)$, $trigger_{Z'}(t'') = \{e\}$ and $action_{Z'}(t'') = \emptyset$. In order that t' is not performed when the event e is in the environment, we give t' a lower priority than that of t''. Now, in the step semantics, the execution of t'' would not allow the execution of another transition in the same parallel component in the considered step. The idea is that of giving t'' a lower priority than that of itself, so it will never be executed, but we will reach the effect that also t' is not executed.
Given a statechart $Z \in \mathcal{Z}_{neg}$ let us assume:

1. $NEG_Z = \{t \in T_Z \mid trigger_Z(t) \cap \overline{E_Z} \neq \emptyset\}$, the set of transitions that contain negated events in their trigger.
2. $NEWT_Z$ a set of new transitions bijectively corresponding to NEG_Z with $r_Z : NEG_Z \to NEWT_Z$ such bijection.
3. $negevents_Z : NEWT_Z \to 2^{E_Z}$, the *negated events function* that for a transition $t' \in NEWT_Z$ gives the set of events appearing negated in $trigger_Z(t)$ with $r_Z(t) = t'$, i.e. such that $negevents_Z(t') = \overline{trigger_Z(r_Z^{-1}(t')) \cap \overline{E_Z}}$.
4. $loop_Z : NEWT_Z \to 2^{\mathcal{T}}$, the *loop function* such that for $t' \in NEWT_Z$, $loop_Z(t')$ is a set of new transitions bijectively corresponding to $negevents_Z(t')$, with $g_Z(t') : loop_Z(t') \to negevents_Z(t')$ such bijection.

The statechart $Z' = TRANS2(Z)$ is defined as follows:

1. $S_{Z'} = S_Z$
2. $hrc_{Z'} = hrc_Z$
3. $type_{Z'} = type_Z$
4. $default_{Z'} = default_Z$
5. $E_{Z'} = E_Z$
6. $T_{Z'} = (T_Z \setminus NEG_Z) \cup NEWT_Z \cup \bigcup_{t' \in NEWT_Z} loop_Z(t')$
7. $lout_{Z'}(t') = lout_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
   $lout_{Z'}(t') = lout_Z(t)$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
   $lout_{Z'}(t'') = lout_Z(t)$ if $t'' \in loop_Z(t') \wedge t \in NEG_Z \wedge r_Z(t) = t'$
8. $lin_{Z'}(t') = lin_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
   $lin_{Z'}(t') = lin_Z(t)$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
   $lin_{Z'}(t'') = lin_Z(t)$ if $t'' \in loop_Z(t') \wedge t \in NEG_Z \wedge r_Z(t) = t'$
9. $hout_{Z'}(t') = hout_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
   $hout_{Z'}(t') = hout_Z(t)$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
   $hout_{Z'}(t'') = hout_Z(t)$ if $t'' \in loop_Z(t') \wedge t \in NEG_Z \wedge r_Z(t) = t'$
10. $hin_{Z'}(t') = hin_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
    $hin_{Z'}(t') = hin_Z(t)$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
    $hin_{Z'}(t'') = hin_Z(t)$ if $t'' \in loop_Z(t') \wedge t \in NEG_Z \wedge r_Z(t) = t'$
11. $trigger_{Z'}(t') = trigger_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
    $trigger_{Z'}(t') = trigger_Z(t) \cap E_Z$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
    $trigger_{Z'}(t'') = \{g_Z(t')(t'')\}$ if $t'' \in loop_Z(t')$ for some $t' \in NEWT_Z$
12. $action_{Z'}(t') = action_Z(t')$ if $t' \in T_Z \setminus NEG_Z$
    $action_{Z'}(t') = action_Z(t)$ if $t' \in NEWT_Z \wedge r_Z(t) = t'$
    $action_{Z'}(t'') = \emptyset$ if $t'' \in loop_Z(t')$ for some $t' \in NEWT_Z$
13. $t' \prec_{Z'} t''$ if $t' \in NEWT_Z \wedge t'' \in loop_Z(t')$
    $t'' \prec_{Z'} t''$ if $t'' \in loop_Z(t')$ for some $t' \in NEWT_Z$.

As an example, let us consider the statechart $Z \in \mathcal{Z}_{neg}$ of Figure 4(a) with $trigger_Z(t_1) = \{\overline{e_1}, e_2\}$, $trigger_Z(t_2) = \{\overline{e_2}, \overline{e_3}\}$, $action_Z(t_i) = \emptyset$, for $1 \le i \le 2$. In this case we have $NEG_Z = \{t_1, t_2\}$, $NEWT_Z = \{t'_1, t'_2\}$, $r_Z = \{(t_1, t'_1), (t_2, t'_2)\}$, $negevents_Z = \{(t'_1, \{e_1\}), (t'_2, \{e_2, e_3\})\}$, $loop_Z = \{(t'_1, \{t_1^{(1)}\}), (t'_2, \{t_2^{(1)}, t_2^{(2)}\})\}$, $g_Z(t'_1) = \{(t_1^{(1)}, e_1)\}$, $g_Z(t'_2) = \{(t_2^{(1)}, e_2), (t_2^{(2)}, e_3).\}$. So if we apply the transformation $TRANS2$ to Z we obtain the statechart $Z' \in \mathcal{Z}_{prior}$ of Figure 4(b) with $trigger_{Z'}(t'_1) = \{e_2\}$, $trigger_{Z'}(t'_2) = \emptyset$, $trigger_{Z'}(t_1^{(1)}) = \{e_1\}$, $trigger_{Z'}(t_2^{(1)}) = \{e_2\}$, $trigger_{Z'}(t_2^{(2)}) = \{e_3\}$, $action_{Z'}(t') = \emptyset$, for every $t' \in T_{Z'}$, $t'_1 \prec_{Z'} t_1^{(1)}$, $t_1^{(1)} \prec_{Z'} t_1^{(1)}$, $t'_2 \prec_{Z'} t_2^{(1)}$, $t'_2 \prec_{Z'} t_2^{(2)}$, $t_2^{(1)} \prec_{Z'} t_2^{(1)}$, $t_2^{(2)} \prec_{Z'} t_2^{(2)}$.

Now we need a notation and a lemma.

Notation. Given $Z \in \mathcal{Z}_{neg}$ and $Z' = TRANS2(Z)$, if $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ is a microconfiguration for Z and $\mu C' = (S'_{exit}, T'_{done}, S'_{curr}, E'_{curr})$ is a microconfiguration for Z' we write $\mu C \hat{=} \mu C'$ iff:

1. $S'_{exit} = S_{exit}$

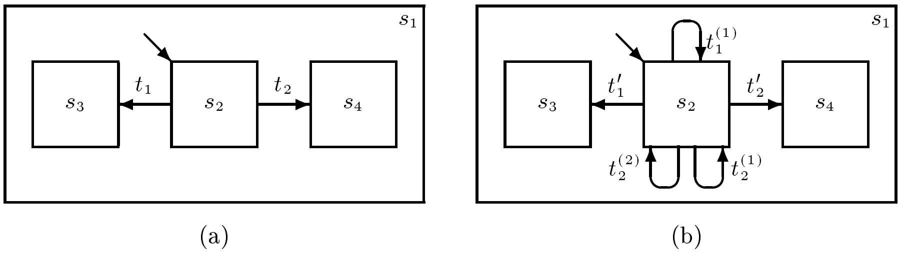


Fig. 4.

2. there exists a bijection $h_2 : T_{done} \rightarrow T'_{done}$ such that for $t \in T_{done}$ either $h_2(t) = t$ if $t \in (T_Z \setminus NEG_Z)$ or $h_2(t) = r_Z(t)$ if $t \in NEG_Z$
3. $S'_{curr} = S_{curr}$
4. $E'_{curr} = E_{curr}$.

Lemma 3.2 Given $Z \in \mathcal{Z}_{neg}$ and $Z' = TRANS2(Z)$, let us assume any of the three semantics considered. If Z and Z' are in microconfigurations $\mu C = (S_{exit}, T_{done}, S_{curr}, E_{curr})$ and $\mu C' = (S'_{exit}, T'_{done}, S'_{curr}, E'_{curr})$, respectively, such that $\mu C \hat{=} \mu C'$, for every $t \in T_Z$:

1. t enabled in $\mu C \ \wedge \ t \notin NEG_Z \Leftrightarrow t \in (T_Z \setminus NEG_Z) \ \wedge \ t$ enabled in $\mu C'$
2. t enabled in $\mu C \ \wedge \ t \in NEG_Z \Leftrightarrow r_Z(t)$ enabled in $\mu C'$.

Proof.
See appendix. ■

Now we can enunciate the following theorem.

Theorem 3.2 Given $Z \in \mathcal{Z}_{neg}$, $Z' = TRANS2(Z)$, a configuration C and an environment E, there exists a bijective function $q : Sem^{step}_{neg}[\![Z]\!]CE \rightarrow Sem^{step}_{prior}[\![Z']\!]CE$ s.t. for every step $\Sigma \in Sem^{step}_{neg}[\![Z]\!]CE$ it holds $nextC(\Sigma) = nextC(q(\Sigma))$.

Proof.
See appendix. ■

An analogous result can be obtained by assuming either the chain semantics of type 1 or the chain semantics of type 2.

Now for statecharts in $\mathcal{Z}_{neg}$ consistency problems may arise (see [13, 1]). As an example, let us consider the statechart $Z \in \mathcal{Z}_{neg}$ of Figure 5(a) assuming $trigger_Z(t_1) = \{\overline{e_1}\}$, $action_Z(t_1) = \{e_2\}$, $trigger_Z(t_2) = \{e_2\}$, $action_Z(t_2) = \{e_1\}$, and any of the three semantics defined for $\mathcal{Z}_{neg}$. Let us take as configuration the default configuration and $E = \emptyset$ as set of external events. In this case $\Sigma = \langle \{t_1\}, \{t_2\} \rangle$ is the only admissible step. This step is not consistent because, by the synchrony hypothesis, we have that in the instant when Σ is performed both

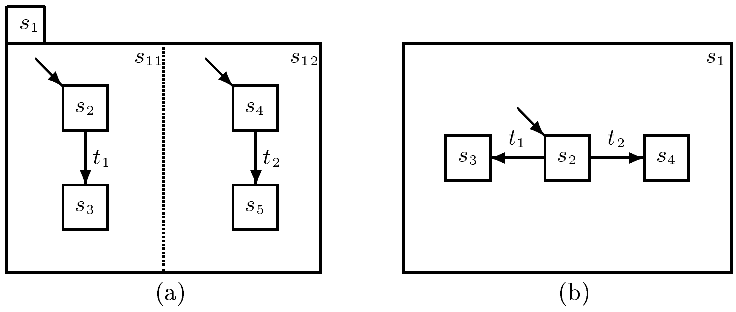


Fig. 5.

the event e_1 occurs and the transition t_1 is performed, whose triggering condition requires that e_1 does not occur in the instant. We formalize now the concept of global consistency of a step.

Definition 3.3 Given a statechart $Z \in \mathcal{Z}_{neg}$, an admissible step $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle$ of Z from a configuration C in an environment E is *globally consistent* iff it does not contain two (non-necessarily distinct) transitions $t \in \mu\Sigma_i$ and $t' \in \mu\Sigma_j$ such that $0 \leq i \leq j \leq n$, $\overline{e} \in trigger_Z(t)$ and $e \in action_Z(t')$ for some $e \in E_Z$.

A semantics for $\mathcal{Z}_{neg}$ is called globally consistent iff it only allows globally consistent admissible steps. Our semantics are not globally consistent. Two examples of globally consistent semantics are those in [13, 9].

Now one may wonder whether by replacing negated events with an explicit priority relation consistency problems of the kind seen above disappear. We shall see that this is not the case. Also for statecharts in $\mathcal{Z}_{prior}$ we may have consistency problems. As an example consider the statechart of Figure 5(b) assuming $trigger_Z(t_1) = \{e_1\}$, $action_Z(t_1) = \{e_2\}$, $trigger_Z(t_2) = \{e_2\}$, $action_Z(t_2) = \emptyset$, and $t_1 \prec_Z t_2$. Let us take as a configuration the default configuration and $E = \{e_1\}$ as environment. In this case $\Sigma = \langle \{t_1\} \rangle$ is the only admissible step. This step is not consistent because, by the synchrony hypothesis, in the instant when Σ is performed both the event e_2 occurs and transition t_1 is chosen, but, if e_2 occurs, t_2 should be chosen as $t_1 \prec_Z t_2$.

Definition 3.4 Given a statechart $Z \in \mathcal{Z}_{prior}$, an admissible step $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle$ of Z from a configuration C in an environment E is *consistent w.r. to priority* iff for every microstep $\mu\Sigma_i$ of Σ and for every transition $t \in \mu\Sigma_i$, $0 \leq i \leq n$, there is no transition $t' \in T_Z$ such that $t \prec_Z t'$, t' is relevant and consistent in μC_i and triggered in μC_j for $j > i$.

A semantics for $\mathcal{Z}_{prior}$ is called consistent w.r. to priority iff it only allows admissible steps consistent w.r. to priority. Our semantics are not consistent w.r. to priority.

Now we show that the transformations $TRANS1$ and $TRANS2$ preserve consistency in the sense that they neither introduce nor eliminate inconsistencies.

Theorem 3.3 Given $Z \in \mathcal{Z}_{prior}$, s.t. $t_1 \prec_Z t_2$ implies $lout_Z(t_1) \in hrc_Z^*(lout_Z(t_2))$, $Z' = TRANS1(Z)$, a configuration C and an environment E, for every step $\Sigma \in Sem_{prior}^{step}[\![Z]\!]CE$ it holds:

$$\Sigma \text{ consistent w.r. to priority} \Leftrightarrow \exists\, \Sigma' \in p(\Sigma) \;\; \Sigma' \text{ globally consistent.}$$

Proof.
See appendix. ■

An analogous result can be proved by assuming chain semantics of type 1.

Theorem 3.4 Given $Z \in \mathcal{Z}_{neg}$, $Z' = TRANS2(Z) \in \mathcal{Z}_{prior}$, a configuration C and an environment E, for every step $\Sigma \in Sem_{neg}^{step}[\![Z]\!]CE$ it holds:

$$\Sigma \text{ globally consistent} \Leftrightarrow q(\Sigma) \text{ consistent w.r. to priority.}$$

Proof.
It follows easily from Theorem 3.2 along the line of the proof of Theorem 3.3. ■

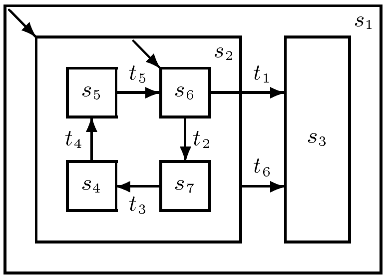
We can enunciate an analogous theorem for both the chain semantics of type 1 and the chain semantics of type 2.

4 Priority induced by the hierarchical structure of states

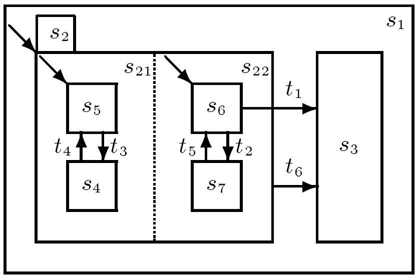
The hierarchical structure of states may be used to express several forms of priority. In particular, one may think of modelling interrupts, namely mechanisms to interrupt (under suitable conditions) system activities. In fact, in statecharts activities are modelled by states, and so interrupts can be modelled by transitions whose source state represents the (sequential or parallel) activity that should be interrupted. As an example, consider the statecharts Z_1 and Z_2 in Figure 6(a) and Figure 6(b), respectively. Transition t_6 may be interpreted as an interrupt for the sequential (resp. parallel) activity associated with state s_2 of Z_1 (resp. Z_2). Now, according to Harel's semantics ([8]), assuming that the statecharts are in both the case in the default configuration, with only transitions t_i, $2 \le i \le 6$, triggered, there would be a nondeterministic choice between transition t_6 and transitions t_2. Now, if we want that transition t_6 behaves as an interrupt, all the transitions in s_2 must have lower priority than transition t_6. In the literature there are various forms of priorities between transitions, that are induced by the hierarchical structure of states and support the concept of interrupt above.

In [4] a priority is defined that depends on the level of the source state of the transitions.

Definition 4.1 Given a statechart $Z \in \mathcal{Z}$ and two transitions $t_1, t_2 \in T_Z$, t_1 has lower priority than t_2 *à la Day*, denoted $t_1 \prec_{Day} t_2$, iff $lout_Z(t_1) \in hrc_Z^+(lout_Z(t_2))$.



(a)



(b)

Fig. 6.

So, assuming the class of statecharts $\mathcal{Z}$ we can define a semantic function Sem^{step}_{Day} (resp. Sem^{ch1}_{Day}, Sem^{ch2}_{Day}): $\mathcal{Z} \times 2^{\mathcal{S}} \times 2^{\mathcal{E}} \rightarrow 2^{(2^{\mathcal{T}})^*}$ that differs from Sem^{step} (resp. Sem^{ch1}, Sem^{ch2}) because a new enabling condition :

$$4'.\ \forall\, t' \in T_Z \ \ t' \text{ relevant, triggered and consistent in } \mu\mathrm{C} \ \Rightarrow\ t \not\prec_{Day} t'$$

is added to the condition in definition 2.7

From [13] another form of priority can be deduced. It is based on the concept of arena.

Definition 4.2 Given a statechart $Z \in \mathcal{Z}$ and two transitions $t_1, t_2 \in T_Z$, t_1 has lower priority than t_2 *à la Pnueli and Shalev*, denoted $t_1 \prec_{P\&S} t_2$, iff

1. $arena_Z(t_1) \in hrc^+_Z(arena_Z(t_2))$
2. $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent.

Note that condition 1 and 2 together ensure that we are considering potentially conflicting transitions. Semantic functions $Sem^{step}_{P\&S}$ (resp. $Sem^{ch1}_{P\&S}$, $Sem^{ch2}_{P\&S}$) can be defined analogously to what done before for priority à la Day. Note that $Sem^{step}_{P\&S}$ does not suffer the limitations due to the use of negated events that we have discussed in the previous section.
The following proposition shows that the two concepts of priority are equivalent if we limit ourselves to statecharts without interlevel transitions.

Proposition 4.1 Given a statechart $Z \in \mathcal{Z}$ such that for every $t \in T_Z$ $hout_Z(t) = lout_Z(t)$ and $hin_Z(t) = lin_Z(t)$, and two transitions $t_1, t_2 \in T_Z$, it holds that

$$t_1 \prec_{Day} t_2 \ \Leftrightarrow t_1 \prec_{P\&S} t_2.$$

Proof.
$\Longrightarrow$
If in Z there are no interlevel transitions and $lout_Z(t_1) \in hrc^+_Z(lout_Z(t_2))$ then $lin_Z(t_1) \in hrc^+_Z(lout_Z(t_2))$, and then $arena_Z(t_1) \in hrc^*_Z(lout_Z(t_2))$. This implies

that $arena_Z(t_1) \in hrc_Z^+(arena_Z(t_2))$. Moreover, as $lout_Z(t_1) \in hrc_Z^+(lout_Z(t_2))$, $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent, and so we have $t_1 \prec_{P\&S} t_2$.

$\Longleftarrow$

Assume $lout_Z(t_1) \notin hrc_Z^+(lout_Z(t_2))$; as $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent, there would be three possible cases:

1. $lout_Z(t_1) = lout_Z(t_2)$. In this case, as in Z there are no interlevel transitions, $arena_Z(t_1) = arena_Z(t_2)$, against the hypothesis that $arena_Z(t_1) \in hrc_Z^+(arena_Z(t_2))$.
2. $lout_Z(t_2) \in hrc_Z^+(lout_Z(t_1))$. In this case, as in Z there are no interlevel transitions, from the first part of the proof we have just seen that $arena_Z(t_2) \in hrc_Z^+(arena_Z(t_1))$, against the hypothesis that $arena_Z(t_1) \in hrc_Z^+(arena_Z(t_2))$.
3. $type_Z(lca_Z(\{lout_Z(t_1), lout_Z(t_2)\})) = AND$ and $lout_Z(t_2) \notin hrc_Z^*(lout_Z(t_1))$. In this case, as in Z there are no interlevel transitions, in $lca_Z(\{lout_Z(t_1), lout_Z(t_2)\}) = lca_Z(\{arena_Z(t_1), arena_Z(t_2)\})$ and so $type_Z(lca_Z(\{arena_Z(t_1), arena_Z(t_2)\})) = AND$, against the hypothesis that $arena_Z(t_1) \in hrc_Z^+(arena_Z(t_2))$.

■

The equivalence does not hold in general. In the case of the statechart of Figure 7(a) we have $t_1 \prec_{Day} t_2$ and $t_1 \not\prec_{P\&S} t_2$; moreover $t_3 \prec_{P\&S} t_1$ and $t_3 \not\prec_{Day} t_1$.

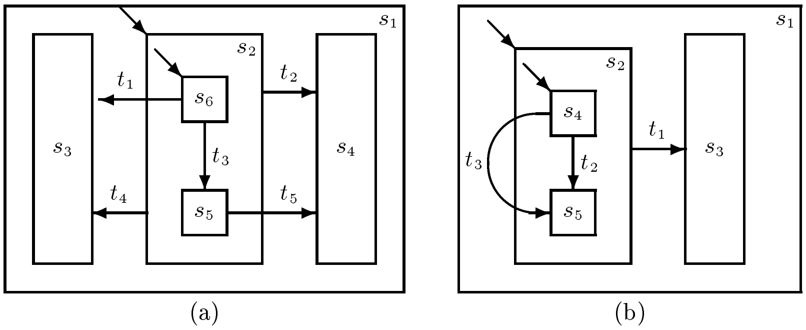


Fig. 7.

In [12] a transition is seen as an interrupt for the activity associated with its lowest source state. This is formalized by means of an implicite relation between a transition t and the transitions t_j inside $lout_Z(t)$, namely

$$t_j \prec_i t \text{ iff } lca_Z(\{lout_Z(t_j), lin_Z(t_j)\}) \in hrc_Z^*(lout_Z(t)).$$

If we consider the statechart in Figure 7(b), we see that with the definition above we would have $t_2 \prec_i t_1$ but also $t_3 \prec_i t_1$, and this is not reasonable because t_3

does not model an internal activity of state s_2. For this reason we consider now the relation

$$t_j \prec_i t \text{ iff } arena_Z(t_j) \in hrc^*_Z(lout_Z(t)).$$

Let us compare now the new relation $\prec_i$ with $\prec_{Day}$ and $\prec_{P\&S}$.

Proposition 4.2 Given a statechart $Z \in \mathcal{Z}$ and two transitions $t_1, t_2 \in T_Z$:

1. $t_1 \prec_i t_2 \;\Rightarrow\; t_1 \prec_{Day} t_2$
2. $t_1 \prec_i t_2 \;\Rightarrow\; t_1 \prec_{P\&S} t_2$.

Proof.

1. $t_1 \prec_i t_2 \;\Rightarrow\; arena_Z(t_1) \in hrc^*_Z(lout_Z(t_2)) \;\Rightarrow\; lout_Z(t_1) \in hrc^+_Z(lout_Z(t_2)) \;\Rightarrow\; t_1 \prec_{Day} t_2$. The vice versa does not hold. Consider, in fact, the statechart in Figure 7(a), for which it holds $t_5 \prec_{Day} t_4$ and $t_5 \not\prec_i t_4$.
2. $t_1 \prec_i t_2 \;\Rightarrow\; arena_Z(t_1) \in hrc^*_Z(lout_Z(t_2)) \;\Rightarrow\; arena_Z(t_1) \in hrc^+_Z(arena_Z(t_2))$ and $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent $\;\Rightarrow\; t_1 \prec_{P\&S} t_2$. The vice versa does not hold. Consider, in fact, the statechart in Figure 7(a), for which it holds $t_4 \prec_{P\&S} t_5$ and $t_4 \not\prec_i t_5$.

■

The proposition above shows that $\prec_{Day}$ and $\prec_{P\&S}$ may support a notion of interrupt more general than that supported by $\prec_i$. Now we want to investigate about a proper extension of the concept of interrupt.
Let us consider once more Figure 6. According to our concept of interrupt, in both the statecharts in figure, transition t_6 is an interrupt for transitions t_i, $2 \le i \le 5$, but not for transition t_1 (as $arena_Z(t_1) = arena_Z(t_6)$). In fact, transitions t_1 may be considered to be a "degenerate" interrupt, an interrupt because its execution forces to exit a composite state (namely state s_2), degenerate because its enabling depends on the internal status of s_2 (this is not the case of an interrupt like t_6). The formalization of *extended interrupt* that we give now covers also the case above.

Definition 4.3 Given a statechart $Z \in \mathcal{Z}$ and two transitions $t_1, t_2 \in T_Z$ we define $t_1 \prec^e_i t_2$ iff

1. $hout_Z(t_1) \in hrc^+_Z(hout_Z(t_2))$
2. $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent.

Obviously, given a statechart $Z \in \mathcal{Z}$, $\prec_i$ is a proper subset of $\prec^e_i$. Now we can prove that following proposition.

Proposition 4.3 Given a statechart $Z \in \mathcal{Z}$ and two transitions $t_1, t_2 \in T_Z$:

$$t_1 \prec_{P\&S} t_2 \;\Leftrightarrow\; t_1 \prec^e_i t_2.$$

Proof.

$\Longrightarrow$

We first prove that $arena_Z(t_1) \in hrc^*_Z(hout_Z(t_2))$. If $arena_Z(t_1) \notin hrc^*_Z(hout_Z(t_2))$, then, as $\forall\, t \in T_Z\;\; hout_Z(t) \in hrc_Z(arena_Z(t))$, there would exist a state $a \in hrc_Z(arena_Z(t_2))$, $a \neq hout_Z(t_2)$, such that $arena_Z(t_1) \in hrc^*_Z(a)$ against the hypothesis that $lout_Z(t_1)$ and $lout_Z(t_2)$ are consistent (remember that $\forall\, t \in T_Z\;\; type_Z(arena_Z(t)) = OR$). Finally, as it holds that for every $t \in T_Z\;\; hout_Z(t) \in hrc_Z(arena_Z(t))$, we have $t_1 \prec^e_i t_2$.

$\Longleftarrow$

If $hout_Z(t_1) \in hrc^+_Z(hout_Z(t_2))$, by definition of $hout_Z$ it holds $arena_Z(t_1) \in hrc^*_Z(hout_Z(t_2))$ and, as for every $t \in T_Z\;\; hout_Z(t) \in hrc_Z(arena_Z(t))$, we have $arena_Z(t_1) \in hrc^+_Z(arena_Z(t_2))$. ■

This shows that our formulation of extended interrupt is equivalent to Pnueli and Shalev's priority, while resembling that of Day as it is based on information regarding sources of transitions (the extra power comes from considering $hout_Z$ instead of $lout_Z$). It is immediate to see that the priorities considered in this section are transitive and therefore they are only a special case of the priorities of section 3.

5 Conclusions and future work

We have analyzed the power of the Statecharts language with respect to expressing transition based priority. Starting from the definition of a basic version of the language we have extended its syntax and semantics for expressing a general notion of priority. We have compared Statecharts with explicit priorities and Statecharts with negated events in triggers of transitions, and proved that the two formalisms can be translated one into the other under some restrictive conditions, showing that the formalism with explicit priorities is more powerful. We have also examined how a concept of interrupt can be modeled by a suitable interpretation of the hierarchical structure of states, and compared with priorities defined in terms of the structures of states by Day ([4]) and by Pnueli and Shalev ([13]).

We have assumed Statecharts with both a step semantics and instantaneous semantics that allow chain of transitions in the same component. In all the cases we have adopted a microstep approach in the definition of semantics. This gives rise to inconsistencies when basic statecharts are endowed with either negated events in triggers or explicit priorities. Among future developments, we consider to study priorities for Statecharts with consistent semantics (such as those of [13] and [9]).

The interrupts we have considered are preemptive, namely they cause immediate interruption of the activity of the subsystem associated. In the case of instantaneous semantics it would also be meaningful to study non-preemptive interrupts. Such interrupts, if enabled, only forbid the subsystem to continue its current activity at the next instant of time, while at the present instant the activity of the subsystem may be interrupted at every stage of its advancement (note that the language Esterel supports an analogous concept (see [2]).

Acknowledgement We thank two anonymous referees for constructive criticism.

References

1. von der Beeck, M.: A Comparison of Statecharts Variants, Proc. FTRTFT '94, Lecture Notes in Computer Science 863, Springer, Berlin, 1994, pp. 128–148.
2. Berry, G.: Preemption in Concurrent Systems, Proc. 13th FSTTCS Conference, Lecture Notes in Computer Science 761, Springer, Berlin, 1993, pp. 72-93.
3. Berry, G., Gonthier, G.: The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation, Science of Computer Programming **19** (1992), pp. 87-152.
4. Day, N.: A Model Checker for Statecharts (Linking CASE Tools with Formal Methods), Technical Report 93-35, University of British Columbia, Vancouver, Canada, 1993.
5. Drusinsky, D., Harel, D.: Using Statecharts for Hardware Description and Synthesis, IEEE Transactions on Computer-Aided Design **8** (1989), pp. 798–807.
6. Harel, D.: Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming **8** (1987), pp. 231–274.
7. Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., Trakhtenbrot M.: Statemate: A Working Environment for the Development of Complex Reactive Systems, IEEE Transations on Software Engineering **16** (1990), pp. 403–414.
8. Harel, D., Pnueli, A., Schmidt, J., P., Sherman, R.: On the Formal Semantics of Statecharts, Proc. 2nd IEEE Symposium on Logic in Computer Science, IEEE CS Press, New York, 1987, pp. 54–64.
9. Maggiolo-Schettini, A., Peron, A., Tini, S.: Equivalence of Statecharts, Proc. CONCUR '96, Lecture Notes in Computer Science 1119, Springer, Berlin, 1996, pp. 687-702.
10. Maraninchi, F.: Operational and Compositional Semantics of Synchronous Automaton Composition, Proc. CONCUR '92, Lecture Notes in Computer Science 630, Springer, Berlin, 1992, pp. 550-564.
11. Peron, A.: Synchronous and Asynchronous Models for Statecharts, Dipartimento di Informatica, Università di Pisa, PhD Thesis, TD 21/93, 1993.
12. Peron, A., Maggiolo-Schettini, A.: Transitions as Interrupts: A New Semantics for Timed Statecharts, Proc. TACS '94, Lecture Notes in Computer Science 789, Springer, Berlin, 1994, pp. 806–821.
13. Pnueli, A., Shalev, M.: What is a Step: On the Semantics of Statecharts, Proc. TACS '91, Lecture Notes in Computer Science 526, Springer, Berlin, 1991, pp. 244–264.

Appendix

Proof of Lemma 3.1

1. $\Longrightarrow$
   There is no $t_1 \in T_Z$ such that $t \prec_Z t_1$. In this case it is easy to see that t is not

affected by $TRANS1$ namely $t \in (T_Z \setminus REP_Z) \setminus REM_Z$. Therefore $t \in T_{Z'}$ and $trigger_{Z'}(t) = trigger_Z(t) \subseteq E_Z$. Moreover, as $\mu C \doteq \mu C'$, it is immediate to see that t is enabled in $\mu C'$.

$\Longleftarrow$

If $t \in T_{Z'}$ and $trigger_{Z'}(t) \subseteq E_Z$, t is not affected by $TRANS1$, and therefore $t \in (T_Z \setminus REP_Z) \setminus REM_Z$. Now, if t is enabled in $\mu C'$, as $\mu C \doteq \mu C'$, the enabling of t in μC follows immediately.

2. $\Longrightarrow$

For every transition $t_1 \in T_Z$ such that $t \prec_Z t_1$ (t_1 is relevant in μC by the hypothesis on $\prec_Z$) t_1 is not triggered in μC. In this case $lockexpr_Z(t)$ has value true in μC, namely there exists a disjunct b_j that is true. This implies that the corresponding transition $t^{(j)} \in transsplit_Z(t)$ is such that $\overline{f_Z(t)(t^{(j)})} \cap E'_{curr} = \emptyset$. So, as $trigger_{Z'}(t^{(j)}) = trigger_Z(t) \cup f_Z(t)(t^{(j)})$ and $E_{curr} = E'_{curr}$, $t^{(j)}$ is triggered in $\mu C'$. Moreover, as $S_{curr} = S'_{curr}$, $lout_Z(t) = lout_{Z'}(t^{(j)})$ and $hout_Z(t) = hout_{Z'}(t^{(j)})$, $t^{(j)}$ is relevant in $\mu C'$. Finally, because t and $t^{(j)}$ are in the same parallel component and have the same $lout$ and $hout$, $t^{(j)}$ is consistent in $\mu C'$, and therefore $t^{(j)}$ is enabled in $\mu C'$.

$\Longleftarrow$

If there exists a transition $t^{(j)} \in transsplit_Z(t)$ enabled in $\mu C'$, $t \in T_Z$ is relevant in μC because $S_{curr} = S'_{curr}$, $lout_Z(t) = lout_{Z'}(t^{(j)})$, $hout_Z(t) = hout_{Z'}(t^{(j)})$; in μC t is also triggered, because $E_{curr} = E'_{curr}$ and $trigger_Z(t) = trigger_{Z'}(t^{(j)}) \cap E_Z$, and consistent, because t and $t^{(j)}$ are in the same parallel component and have the same $lout$ and $hout$. Finally, if $t^{(j)}$ is triggered in $\mu C'$ the disjunct b_j of $lockexpr_Z(t)$ is true and so also $lockexpr_Z(t)$. This implies immediately there does not exist a transition $t_1 \in T_Z$, such that $t \prec_Z t_1$, triggered in μC. So $t \in T_Z$ is enabled in μC.

Proof of Theorem 3.1

1. Let us take $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle \in Sem^{step}_{prior}[\![Z]\!]CE$ and let $\mu\Sigma_i$ be a microstep from the microconfiguration $\mu C_i = (S^i_{exit}, T^i_{done}, S^i_{curr}, E^i_{curr})$ to μC_{i+1} $= (S^{i+1}_{exit}, T^{i+1}_{done}, S^{i+1}_{curr}, E^{i+1}_{curr})$, for $0 \leq i \leq n$. Let us construct a set $\Sigma' = \langle \mu\Sigma'_0, \ldots, \mu\Sigma'_n \rangle$ where each $\mu\Sigma'_i \subseteq T_{Z'}$ is such that:
   (a) $t \in \mu\Sigma_i \ \wedge \ t \notin REP_Z \ \Rightarrow \ t \in \mu\Sigma'_i$
   (b) $t \in \mu\Sigma_i \ \wedge \ t \in REP_Z \ \wedge \ \overline{f_Z(t)(t^{(j)})} \cap E^i_{curr} = \emptyset$ for some $t^{(j)} \in transsplit_Z(t)$ $\Rightarrow \ t^{(j)} \in \mu\Sigma'_i$.

   If we assume the configuration C, the environment E and the microconfiguration $\mu C'_0 = (\emptyset, \emptyset, C, E)$, we have that, by Lemma 3.1, all the transitions of $\mu\Sigma'_0$ are enabled in $\mu C'_0$ and, immediately, that they are pairwise structurally consistent, and so $\mu\Sigma'_0$ is a microstep from $\mu C'_0$ ($= \mu C_0$) to $\mu C'_1$ (with $\mu C_1 \doteq \mu C'_1$). Moreover, again by Lemma 3.1, each $\mu\Sigma'_i$ is a microstep from $\mu C'_i$ to $\mu C'_{i+1}$ with $\mu C_i \doteq \mu C'_i$ and $\mu C_{i+1} \doteq \mu C'_{i+1}$, for $1 \leq i \leq n$. So Σ' is a set of microsteps. It also easy to see that Σ' is maximal. If this were not true, there would be a transition $t' \in T_{Z'}$ enabled in $\mu C'_{n+1}$. Now, either t' would be a transition not affected by $TRANS1$, and the same transition would be enabled in μC_{n+1}, or $t' \in transsplit_Z(t)$ for some $t \in REP_Z$, and in this case t would be enabled in μC_{n+1}, in both the cases against the maximality of Σ. Therefore $\Sigma' \in Sem^{step}_{neg}[\![Z']\!]CE$. Now we define p such that, given a step $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle \in Sem^{step}_{prior}[\![Z]\!]CE$, all steps Σ', constructed from Σ, are in $p(\Sigma)$. Finally, as, for each $\Sigma' \in p(\Sigma)$, $\mu C_{n+1} \doteq \mu C'_{n+1}$, it holds $nextC(\Sigma') = nextC(\Sigma)$.

2. Let us take $\Sigma' = \langle \mu\Sigma'_0, \ldots, \mu\Sigma'_n \rangle \in Sem^{step}_{neg}[\![Z']\!]CE$ and let $\mu\Sigma'_i$ be a microstep from the microconfiguration $\mu C'_i$ to $\mu C'_{i+1}$, $0 \le i \le n$. Let us construct a set $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle$ where each $\mu\Sigma_i \subseteq T_Z$ is such that:
   (a) $t' \in \mu\Sigma'_i \ \wedge\ trigger_{Z'}(t') \cap \overline{E_{Z'}} = \emptyset \ \Rightarrow\ t' \in \mu\Sigma_i$
   (b) $t' \in \mu\Sigma'_i \ \wedge\ trigger_{Z'}(t') \cap \overline{E_{Z'}} \neq \emptyset \ \Rightarrow\ t \in \mu\Sigma_i$ for $t' \in transsplit_Z(t)$
   If we assume the configuration C, the environment E and the microconfiguration $\mu C_0 = (\emptyset, \emptyset, C, E)$, we have that, by Lemma 3.1, all the transitions of $\mu\Sigma_0$ are enabled in μC_0 and, immediately, that they are pairwise structurally consistent, and so $\mu\Sigma_0$ is a microstep from μC_0 ($= \mu C'_0$) to μC_1 ($\doteq \mu C'_1$). Moreover, again by Lemma 3.1, each $\mu\Sigma_i$ is a microstep from μC_i to μC_{i+1} with $\mu C_i \doteq \mu C'_i$ and $\mu C_{i+1} \doteq \mu C'_{i+1}$. So Σ is a set of microsteps. Its maximality follows immediately. Therefore, $\Sigma \in Sem^{step}_{prior}[\![Z]\!]CE$. Finally, by definition of p, $\Sigma' \in p(\Sigma)$.

Proof of Lemma 3.2

1. $\Longrightarrow$
   In this case t is not affected by $TRANS2$, and therefore $t \in (T_Z \setminus NEG_Z)$. Moreover, as $\mu C \hat{=} \mu C'$ and there is no $t' \in T_{Z'}$ such that $t \prec_{Z'} t'$ by definition of $TRANS2$, t is enabled in $\mu C'$.
   $\Longleftarrow$
   The proof follows a converse reasoning with respect to that of the direct implication.
2. $\Longrightarrow$
   In this case $r_Z(t) \in T_{Z'}$ and $\forall\, t'' \in loop_Z(r_Z(t)) \quad r_Z(t) \prec_{Z'} t''$. Now, as $\mu C \hat{=} \mu C'$, $trigger_{Z'}(r_Z(t)) = trigger_Z(t) \cap E_Z$, $lout_{Z'}(r_Z(t)) = lout_Z(t)$, $hout_{Z'}(r_Z(t)) = hout_Z(t)$, $lin_{Z'}(r_Z(t)) = lin_Z(t)$, $hin_{Z'}(r_Z(t)) = hin_Z(t)$, $r_Z(t)$ is trivially relevant, triggered and consistent in $\mu C'$. Moreover if t is triggered in μC, $negevents_Z(r_Z(t)) \cap E_{curr} = \emptyset$, and as $E_{curr} = E'_{curr}$, each $t'' \in loop_Z(r_Z(t))$ is not triggerd in $\mu C'$. Therefore there are no transitions with higher priority than $r_Z(t)$ and so $r_Z(t)$ is enabled in $\mu C'$.
   $\Longleftarrow$
   The proof follows a converse reasoning with respect to that of the direct implication.

Proof of Theorem 3.2

1. Let us take $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle \in Sem^{step}_{neg}[\![Z]\!]CE$ and let $\mu\Sigma_i$ be a microstep from the microconfiguration μC_i to μC_{i+1}, for $0 \le i \le n$. Let us construct a set $\Sigma' = \langle \mu\Sigma'_0, \ldots, \mu\Sigma'_n \rangle$ where each $\mu\Sigma'_i \subseteq T_{Z'}$ is such that:

(a) $t \in \mu\Sigma_i \ \wedge \ t \notin NEG_Z \ \Rightarrow \ t \in \mu\Sigma'_i$

(b) $t \in \mu\Sigma_i \ \wedge \ t \in NEG_Z \ \Rightarrow \ r_Z(t) \in \mu\Sigma'_i$.

If we assume the configuration C, the environment E and the microconfiguration $\mu C'_0 = (\emptyset, \emptyset, C, E)$, we have that, by Lemma 3.2, all the transitions of $\mu\Sigma'_0$ are enabled in $\mu C'_0$ and, immediately, that they are pairwise structurally consistent, and so $\mu\Sigma'_0$ is a microstep from $\mu C'_0$ ($= \mu C_0$) to $\mu C'_1$ (with $\mu C_1 \hat{=} \mu C'_1$). Moreover, again by Lemma 3.2, each $\mu\Sigma'_i$ is a microstep from $\mu C'_i$ to $\mu C'_{i+1}$ with $\mu C_i \hat{=} \mu C'_i$ and $\mu C_{i+1} \hat{=} \mu C'_{i+1}$. So Σ' is a set of microsteps. It also easy to see that Σ' is maximal. If this were not true there would be a transition $t' \in T_{Z'}$ enabled in $\mu C'_{n+1}$. Now, either t' would be a transition not affected by $TRANS2$, and the same transition would be enabled in μC_{n+1}, or if $t' \in NEWT_Z$ and $r_Z^{-1}(t')$ would be enabled in μC_{n+1}, in both the cases against the maximality of Σ. Therefore $\Sigma' \in Sem^{step}_{prior}[\![Z']\!]CE$. Now we define q such $q(\Sigma) = \Sigma'$. As $\mu C_{n+1} \hat{=} \mu C'_{n+1}$, it holds $nextC(\Sigma) = nextC(q(\Sigma))$.

2. Let us take $\Sigma' = \langle \mu\Sigma'_0, \ldots, \mu\Sigma'_n \rangle \in Sem^{step}_{prior}[\![Z']\!]CE$ and let $\mu\Sigma'_i$ be a microstep from the microconfiguration $\mu C'_i$ to $\mu C'_{i+1}$, $0 \leq i \leq n$. Let us construct a set $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle$ where each $\mu\Sigma_i \subseteq T_Z$ is such that:

   (a) $t' \in \mu\Sigma'_i \ \wedge \ t' \in (T_Z \setminus NEG_Z) \ \Rightarrow \ t' \in \mu\Sigma_i$

   (b) $t' \in \mu\Sigma'_i \ \wedge \ t' \in NEWT_Z \ \Rightarrow \ r_Z^{-1}(t') \in \mu\Sigma_i$.

   If we assume the configuration C, the environment E and the microconfiguration $\mu C_0 = (\emptyset, \emptyset, C, E)$, we have that, by Lemma 3.2, all the transitions of $\mu\Sigma_0$ are enabled in μC_0 and, immediately, that they are pairwise structurally consistent, and so $\mu\Sigma_0$ is a microstep from μC_0 ($= \mu C'_0$) to μC_1 ($\hat{=} \mu C'_1$). Moreover, again by Lemma 3.2, each $\mu\Sigma_i$ is a microstep from μC_i to μC_{i+1} with $\mu C_i \hat{=} \mu C'_i$ and $\mu C_{i+1} \hat{=} \mu C'_{i+1}$. So Σ is a set of microsteps. Its maximality follows immediately. Therefore, $\Sigma \in Sem^{step}_{neg}[\![Z]\!]CE$. Finally, by definition of q, $q(\Sigma) = \Sigma'$ for a unique step Σ, and hence q is a bijection.

Proof of Theorem 3.3

$\Longrightarrow$

Let us consider $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle \in Sem^{step}_{prior}[\![Z]\!]CE$ and let $\mu\Sigma_i$ be a μstep from the microconfiguration μC_i to μC_{i+1}, for $0 \leq i \leq n$. Let us assume Σ consistent w.r. to priority and let $t_1, \ldots, t_m$ be the sequence of all transitions in Σ such that:

1. $t_k \in REP_Z$ for $1 \leq k \leq m$
2. $t_k \in \mu\Sigma_{r_k} \Rightarrow t_l \in \mu\Sigma_{r_l}$, for $1 \leq k < l \leq m$, $0 \leq r_k \leq r_l \leq n$.

Let us consider t_1 with $t_1 \in \mu\Sigma_{r_1}$, $0 \leq r_1 \leq n$. All steps in $p(\Sigma)$, contain in their r_1-th microstep, a transition of $transplit_Z(t_1)$. Now there is at least one $t_1^{(j_1)} \in transsplit_Z(t_1)$ such that, in all steps of $p(\Sigma)$ with $t_1^{(j_1)}$ in the r_1-th microstep, events are not generated that contradict the trigger of $t_1^{(j_1)}$. If this were not true, by definition of $TRANS1$, Σ would be not consistent w.r. to priority (t_1 should be not executed) against the hypothesis. Let us call R_1 the set of steps in $p(\Sigma)$ with $t_1^{(j_1)}$ in the r_1-th microstep. Now, by induction, assume R_l, for $1 \leq l < m$,

the set of steps with $t_h^{(j_h)}$ in the r_h-th microstep, $1 \leq h \leq l$, $t_h^{(j_h)} \in transsplit_Z(t_h)$ such that, in all steps of R_l events are not generated that contradict the triggers of $t_h^{(j_h)}$, for each h. Now all steps in R_l contain in their r_{l+1}-th microstep a transition of $transsplit_Z(t_{l+1})$ and there is at least one $t_{l+1}^{(j_{l+1})} \in transsplit_Z(t_{l+1})$ such that, in all steps R_l with $t_{l+1}^{(j_{l+1})}$ in the r_{l+1}-th microstep, events are not generated that contradict the trigger of $t_{l+1}^{(j_{l+1})}$. If this were not true, by definition of $TRANS1$, Σ would be not consistent w.r. to priority. We define R_{l+1} the set of steps in R_l with $t_{l+1}^{(j_{l+1})}$ in the r_{l+1}-th microstep. So we can conclude that there is a non empty set $R_m \subseteq p(\Sigma)$ of globally consistent steps.
It is easy to see that, in general, not all steps in $p(\Sigma)$ are globally consistent.
$\Longleftarrow$
Let us consider $\Sigma = \langle \mu\Sigma_0, \ldots, \mu\Sigma_n \rangle \in Sem_{prior}^{step}[\![Z]\!]CE$ and let $\mu\Sigma_i$ be a μstep from the microconfiguration μC_i to μC_{i+1}, for $0 \leq i \leq n$. Let us assume a globally consistent step $\Sigma' \in p(\Sigma)$. Now if Σ were not consistent w.r. to priority there would be a transition $t \in \mu\Sigma_i$, for some i, and a transition $t_1 \in T_Z$ such that $t \prec_Z t_1$, t_1 relevant and consistent in μC_i but triggered in μC_j, with $j > i$. In this case, by definition of $TRANS1$, all steps in $p(\Sigma)$ should contain, in their i-th microstep, some transition of $transsplit_Z(t)$ whose trigger contains some negated event that is generated in the step. Therefore all the steps in $p(\Sigma)$ would not be globally consistent, against the hypothesis.

This article was processed using the LaTeX macro package with LLNCS style