What is in a Step: New Perspectives on a Classical Question*

Willem-Paul de Roever[1], Gerald Lüttgen[2], and Michael Mendler[2]

[1] Institute of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, Germany, wpr@informatik.uni-kiel.de

[2] Software Technologies and Informatics Theory Research Groups, Otto-Friedrich-University of Bamberg, Germany, gerald.luettgen@swt-bamberg.de, michael.mendler@uni-bamberg.de

Point of Departure: Pnueli & Shalev's 1991 paper
"What's in a Step: On the semantics of Statecharts"

- Pnueli and Shalev show how, while observing global consistency and causality, the synchronous language Statecharts can be given coinciding operational and declarative (i.e., fixed point) step semantics
- Over the past decade, this semantics has been supplemented with order-theoretic, fully abstract and compositional denotational, axiomatic and game-theoretic semantics and used to emphasize the close connection with Esterel and logic programming (subject of talk)
- This reveals the Pnueli-Shalev step semantics as a rather canonical interpretation of the synchrony hypothesis

Short intro to Statecharts

- A hierarchical, concurrent Mealy machine
- Basic states hierarchically refined by injecting other Statecharts
- Composite states of 2 possible sorts: and-states and or-states
- And-states permit parallel and or-states sequential decomposition
- An and-state is active if all its substates are active, an or-state if exactly one of its substates is active
- Set of active states during execution called a configuration

The synchrony hypothesis

- Statecharts belongs to the family of SYNCHRONOUS languages (s.a. Esterel, Signal, Lustre, Argos)
- Semantics based on a cycle-based reaction, in which events output by the system's env. are sampled first and pot. cause the firing of transitions that may produce new events
- Generated events output to the env. when the reaction ends
- SYNCHRONY HYPOTHESIS ensures that: this complex non-atomic step bundled into ONE ATOMIC STEP
- Justification: reactions computed quicker than time it takes for new events to arrive from the system's env

What exactly constitutes a step?

- Are generated events sensed only in the next step, or already in the current step, and thus trigger the firing of further transitions?
- First option: Harel's official non-compositional "semantics A" implemented in Statemate
- Second option: A step involves a causal chain of firing transitions:
- A transition fires if its positive triggers (offered by env or generated by a trans. fired previously in the same step) are present and its negative triggers are absent (i.e., not present)

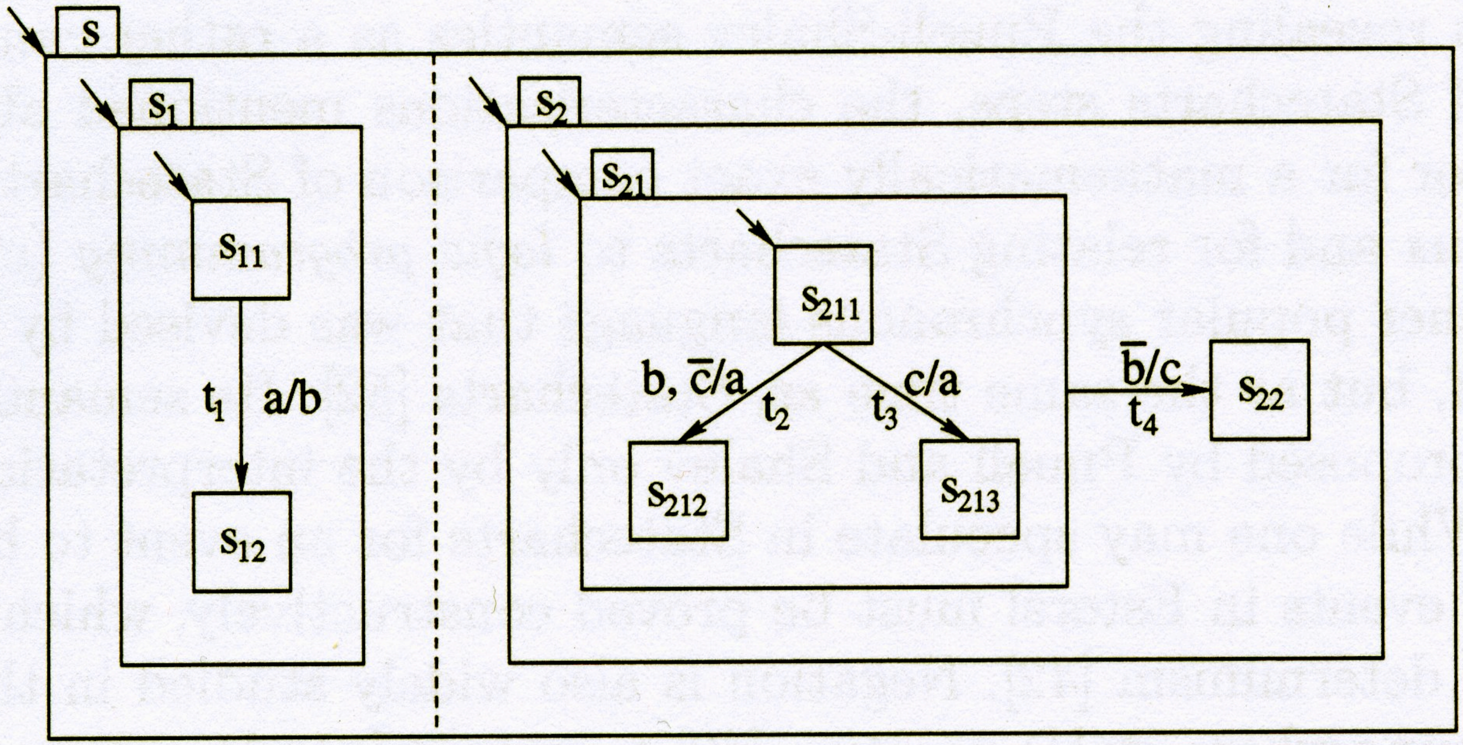

Fig. 1. Example Statechart.

What exactly constitutes a step (cont'd)?

- Thus, when it fires, a transition may, as part of its action, BROADCAST new events, which, by the principle of CAUSALITY, may trigger further transitions
- Only when this chain reaction of firing transitions comes to a halt is a step COMPLETE, and, acc. to the synchrony hypothesis, an atomic entity
- This semantics is NONCOMPOSITIONAL, since bundling a trans. into an atomic step implies forgetting the transition's causal justification
- Also, it is not GLOBALLY CONSISTENT, as it permits the same event to be both present and absent within the same step: an event that occurs negatively in the trigger of one firing transition MAY BE GENERATED BY A TRANS. THAT FIRES LATER IN THE SAME STEP

Pnueli & Shalev's contribution

- In Pnueli and Shalev's words, "a proven sign of healthy and robust understanding of the meaning of a programming or specification language is the possession of both an operational and declarative semantics, which are consistent with one another"
- They showed that adding global consistency is the key to achieving this ambitious goal for Statecharts
- The resulting operational semantics relies on an iterative FIXED-POINT CONSTRUCTION over a non-monotonic enabledness function for transitions
- This construction ensures causality but involves backtracking once a global inconsistency is introduced
- Their declarative semantics for Statecharts identifies the desired fixed point of the enabledness fu thru the notion of SEPARABILITY

Intro to Statecharts (cont'd)

- Statechart steps defined relative to a configation C and a set E of events given to the system by its environment
- Key to a step are transitions t each of which is labeled by two sets of events: a trigger trg(t) and an action act(t)
- Trigger $trg(t) = P, N^{co}$ split into positive events $P \subseteq \prod$ and negative events $N \subseteq \prod^{co}$.
- t is enabled and thus fires if the set $E \subseteq \prod$ is such that all events of P, but NONE of N, are in E, i.e., $P \subseteq E$ and $N \cap E = \varnothing$
- The effect of firing t is the generation of all events in the action act(t) of t, where a transition's action act(t) consists of positive events only

Transition t is *consistent* with set T of transitions, in signs $t \in \mathsf{consistent}(C, T)$, if t is not in the same "parallel component" as any $t' \in T \setminus \{t\}$. Formally,

$$\mathsf{consistent}(C,T) =_{\mathrm{df}} \{t \in \mathsf{trans}(C) \mid \forall t' \in T.\ t \triangle_C t'\}\ ,$$

where $t \triangle_C t'$ if (i) $t = t'$ or (ii) t and t' are in different substates of an enclosing and-state. Further, transition t is *triggered* by a set E of events, in signs $t \in \mathsf{triggered}(C, E)$, if the positive but not the negative trigger events of t are in E:

$$\mathsf{triggered}(C,E) =_{\mathrm{df}} \{t \in \mathsf{trans}(C) \mid \mathsf{trg}(t) \cap \Pi \subseteq E,\ \ \overline{(\mathsf{trg}(t) \cap \overline{\Pi})} \cap E = \emptyset\}\ .$$

Finally, transition t is *enabled* in C with respect to set E of events and set T of transitions, if $t \in \mathsf{enabled}(C, E, T)$ where

$$\mathsf{enabled}(C,E,T) =_{\mathrm{df}} \mathsf{consistent}(C,T) \cap \mathsf{triggered}(C, E \cup \bigcup_{t \in T} \mathsf{act}(t))\ .$$

Pnueli-Shalev Semantics

procedure *step–construction*(C, E);
  var $T := \emptyset$;
  while $T \subset$ enabled(C, E, T) do
    choose $t \in$ enabled$(C, E, T) \setminus T$;
    $T := T \cup \{t\}$
  od;
  if $T =$ enabled(C, E, T) then return T
  else *report failure*
end *step–construction*.

Operational semantics

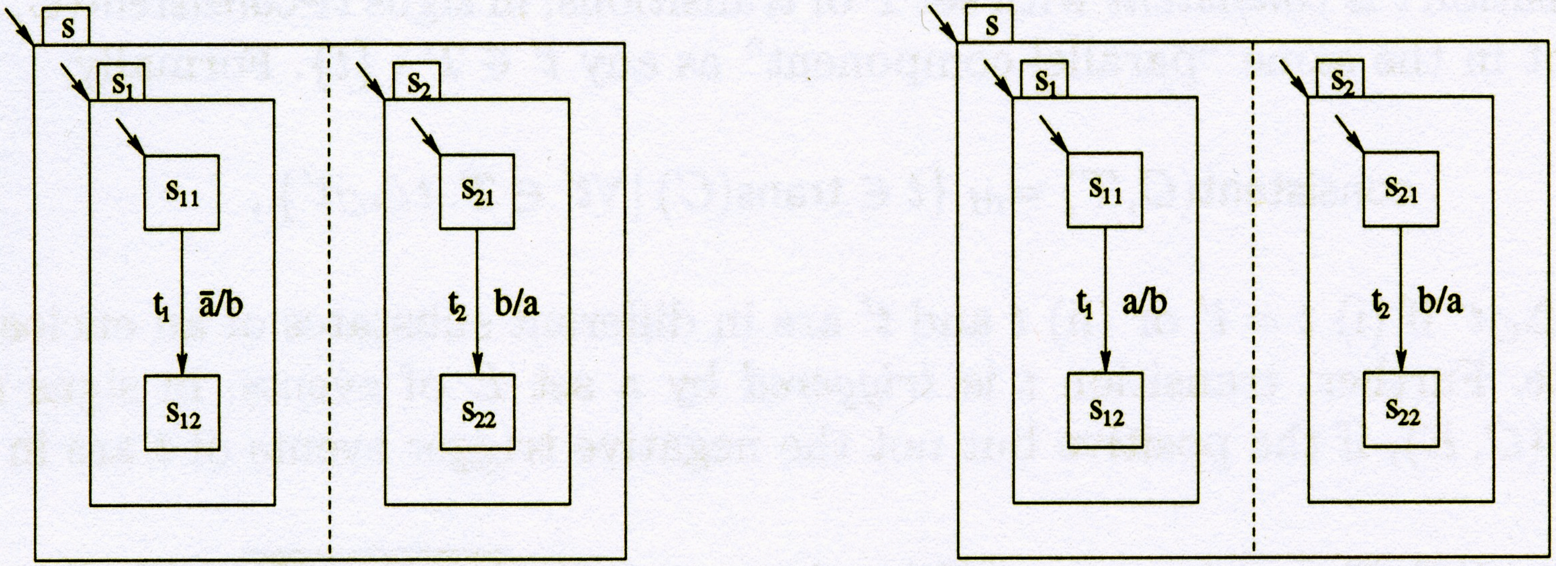

Fig. 2. Further example Statecharts.

Following Pnueli and Shalev's terminology, a set T of transitions is called *constructible* for a given configuration C and a set E of environment events, if it can be obtained as a result of successfully executing procedure *step-construction*. For each constructible set T, set $A =_{\mathrm{df}} E \cup \mathsf{act}(T) \subseteq \Pi$ is called the *(step) response* of C for E.