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

Willem-Paul de Roever¹, Gerald Lüttgen², and Michael Mendler²

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 function through the notion of SEPARABILITY
Intro to Statecharts (cont’d)

- Statechart steps defined relative to a configuration \( C \) 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 \( \text{trg}(t) \) and an action \( \text{act}(t) \).

- Trigger \( \text{trg}(t) = P, N \) split into positive events \( P \subseteq \Pi \) and negative events \( N \subseteq \Pi \).

- \( t \) is enabled and thus fires if the set \( E \subseteq \Pi \) is such that all events of \( P \), but NONE of \( N \), are in \( E \), i.e., \( P \subseteq E \) and \( N \cap E = \emptyset \).

- The effect of firing \( t \) is the generation of all events in the action \( \text{act}(t) \) of \( t \), where a transition’s action \( \text{act}(t) \) consists of positive events only.
Transition $t$ is consistent with set $T$ of transitions, in signs $t \in \text{consistent}(C, T)$, if $t$ is not in the same “parallel component” as any $t' \in T \setminus \{t\}$. Formally,

$$\text{consistent}(C, T) = \text{df} \{ t \in \text{trans}(C) | \forall t' \in T. t \not\in C t' \},$$

where $t \not\in C t'$ 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 $t \in \text{triggered}(C, E)$, if the positive but not the negative trigger events of $t$ are in $E$:

$$\text{triggered}(C, E) = \text{df} \{ t \in \text{trans}(C) | \text{trg}(t) \cap \Pi \subseteq E, \overline{\text{trg}(t) \cap \Pi} \cap E = \emptyset \}.$$

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

$$\text{enabled}(C, E, T) = \text{df} \ \text{consistent}(C, T) \cap \text{triggered}(C, E \cup \bigcup_{t \in T} \text{act}(t)).$$
procedure step-construction(C, E);
    var T := ∅;
    while T ⊆ enabled(C, E, T) do
        choose t ∈ enabled(C, E, T) \ T;
        T := T ∪ {t}
        od;
    if T = enabled(C, E, T) then return T
    else report failure
    end step-construction.
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 $A =_{df} E \cup \text{act}(T) \subseteq \Pi$ is called the (step) response of $C$ for $E$. 

**Fig. 2.** Further example Statecharts.