Statemate Course

W.-P. de Roever
K. Baukus

CAU Kiel

Semantics of Statecharts — Lesson II

We discuss the central concepts and decisions for various possible semantics for Statecharts (and the "real" implemented one).

[HdR91] Introduction to Design Choices in the Semantics of Statecharts, C. Huizing, W.-P. de Roever, Information Processing Letters 37, p. 205-213, 1991

[HP98] Modeling Reactive Systems with Statecharts: The STATEMATE Approach, D. Harel, M, Politi. McGraw-Hill, 1998.

Real-Time Systems

Real-Time Systems

- the environment can deliver data continuously, for example via temperature sensor.
- data can be delivered from different sources simultaneously and must therefore be processed in parallel.
- the time scale is fast by human standard (milli seconds instead of seconds),
- the system must react in time and accurately on input from the environment.

Characteristics

- the environment of a real-time systems contains often equipment that act as sensors of the systems (a typical example are optical scanners, that collect a continuous stream of relative unstructured data) and also equipment that can change the environment physical (like 'actuators' that continuously change the position of a tap).
- real-time systems must often simultaneously process input from different sources (in an industrial process control system, for example, the values of temperature, pressure and liquid concentration must be correlated simultaneously to execute adjustments of heating-apparatus and taps)

Characteristics

- the time scale of much real-time systems is fast considered to human standards (in an automated systems for the control of the speed of a car, for instance, the real speed must be measured many times a second to insure a comfortable ride; although this is fast for human standards, it is slow for real-time standards where the speed of operation is near the speed of the available implementation technology)
- real-time systems must react punctual and accurate on the input from the environment (a valid value on the wrong moment is often more worse that no value at all)
- the data in a real-time system has often a limited period of validity (think of for instance a green traffic light for an automated car control system)

Characteristics

- the observation of the environment by the real-time system is always behind so that there is an inconsistency between the perception and the real state of that environment
- a real-time system must be prepared for every possible situation that can happen in the environment and must be safe with respect to appearance of critical fault situations
- a real-time system must satisfy high reliability requirements: a guaranteed MTBF (Mean Time Between Failure) and MTTR (Mean Time To Repair), for instance, telephone exchanges are considered not to be out of order for more than 2 hours in 40 years.

Time in Statecharts

Time

- The elementary unit of observation in a reactive system is the **event**
- The **environment** sends events to the system to **trigger computations**, the system reacts to the environment by sending, or **generating**, events.
- Events are also means of **communication** between parts of a system.

Time (cont'd)

- Because one wants to specify reactive systems at the **highest level of abstraction** in a discrete fashion, events are **discrete signals**, occurring at a **point in time.**
- Events have **no duration**; they are generated from one state to another. Hence, transitions have a discrete uninterruptable nature and **all time is spent in states.**

Reason

In a real system new inputs may arrive at any moment. Therefore the current state it is in should be always clear. Since transitions have no duration, there are no "transient" periods in between states.
Therefore, the reaction on a possible input is always well defined.

> Of course this is an abstraction from reality. (At deep levels of electronic implementations, one encounters levels where discrete reasoning makes no sense anymore)

Statecharts is meant to be a high level specification language, where this abstraction can be maintained and is appropriate.

Reaction Time

- We know that transitions have no duration, but **when** do they take place, relative to the trigger? And:

  HOW LONG DOES IT TAKE THE SYSTEM TO COMPUTE A REACTION UPON AN EXTERNAL EVENT?

- For transformational systems this is easy — the only important distinction is between finite and infinite values (corresponding to a final state or no final state)
- For reactive systems this is not enough:
  We have to know when an output occurs relative to the events in the input sequence (see Brock-Ackermann paradox) $\Longrightarrow$
  One has to determine **the reaction time of a sequence.**

What's the reaction time of a reactive systems upon an external event in the high level Specification Language Statecharts?

Possibility 1

Specify a concrete amount of time for each situation. This forces us to quantify time right from the beginning. **Clumsy**, and not appropriate at this stage of specification where one is only interested in the relative order and coincidence of events.

Possibility 2

Fix reaction time between trigger **a** and corresponding action **a** within **e**/**a** (the label of a transition) upon 1 time unit.
Doesn't work: Upon refining **question**/**answer** to a **question**/**consult** and a **consult**/**answer** transition, there's a change of time, which may have far reaching effects (because of **tm(n)**-events, e.g.)
$\Longrightarrow$

A fixed execution time for syntactic entities (transitions, statements, etc.) is not **flexible** enough.

Possibility 3

Leave things open: say only that execution of a reaction takes some positive amount of time, and see at a later stage (closer to the actual implementation) how much time things take.

Clumsy, introduces far too much nondeterminism.

Solution

Summary : We want the execution time associated to reactions to have following properties:

- It should be accurate, but not depending on the actual implementation.
- It should be as short as possible, to avoid artificial delays.
- It should be abstract in the sense that the timing behavior must be orthogonal to the functional behavior.

$\Longrightarrow$

Only choice that meets all wishes is **zero reaction time.**

Problems Disappear

As a result all objections raised w.r.t. the possibilities mentioned on the previous page are met!

- Now, for instance, upon refining transition **question/answer** from previous page into two transitions, the reaction time of this refinement is the same as that of the original transition.
- Objection 3 on the previous transparency is resolved, too.
- Finally, also objection 1 (on previous transparency) is met, because $0 + 0 = 0$!

Berry's synchrony hypothesis

This choice, that the reaction time between a trigger and its event is zero, is called **Berry's synchrony hypothesis.**
Is this implementable? No, a real computation takes time.
But in actual implementation this means:

The reaction comes before the next input arrives,

or, so to say,

Reactions are not infinitely fast but fast enough.

Example

See the following figure:

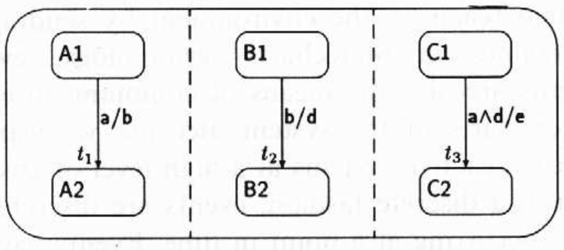


Fig. 6.

A consequence is that transition t_3 is taken!!

Negations and paradoxes

- Idea of immediate reaction works fine as long as transitions only triggered by primitive events, or or conjunctions and disjunctions of them.
- However, one also needs **negations of events** to trigger a transition. E.g.: to specify priority:

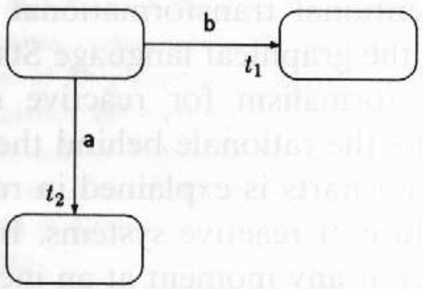


Fig. 7.

Problem

What semantics to give to Statecharts in the next figure?

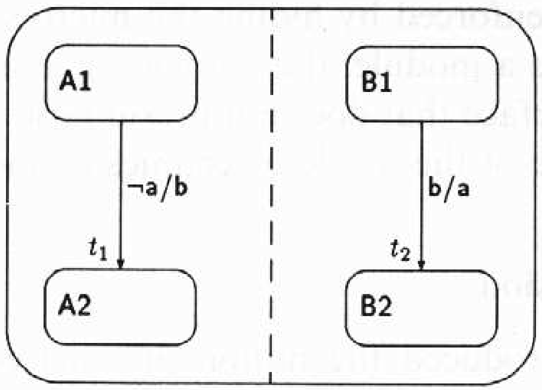


Fig. 8.

Grandfather Paradoxon

If **a** is absent, i.e., ¬**a** holds as condition, transition t_1 is taken, i.e., **b** is generated, and hence t_2, i.e., **b/a** is taken, generating **a within the same time unit**, i.e., in zero time, hence transition t_1 should not be taken.
But that means that event **b** is not generated, and hence event **a** is not generated, so transition t_1 should be taken, etc.
This is called the "Grandfather paradox".
It's solution is to **order** event occurrences causally, with **later events not influencing earlier events**:$\neg a \leq b \leq a$

> Note here: this causal order has nothing to do with the passage of time; it merely refers to causal chains **within** one time step.

Solution

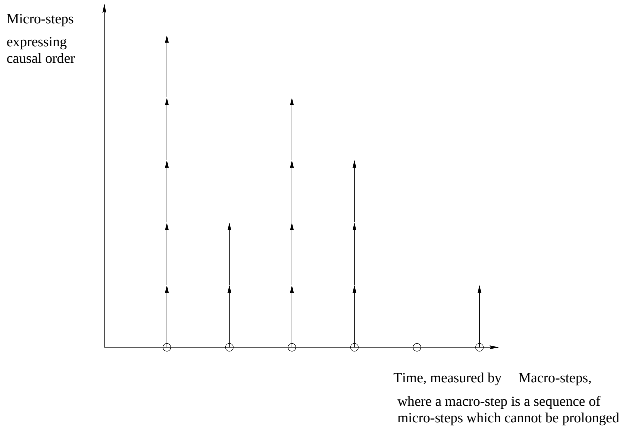
Introduce two levels of time

- **Macro steps**, for counting time, (these are observable) time steps, and
- **Micro steps**, which describe the causal chain within reactions.
  Every macro-step is then divided in an arbitrary but finite number of micro-steps.

This sequence of micro-steps has only an operational meaning.

The New Semantics

This leads to a semantics of the following form:



The New Semantics (cont'd)

- Macro-steps are observable steps $\Longrightarrow^{O}_{I}$
- Each macro-step is a sequence of micro-steps, that are ordered causally; one micro-step can never influence previous micro-steps.
- In Statecharts as implemented by Statemate causality is trivially obtained because in Statemate events generated in one step are only available in the next step, and only for that one. I.e., there is no causality within one step.