FOUNDATIONS OF THE UML

Lecture 9:

Statecharts Semantics

Joost-Pieter Katoen

MOVES

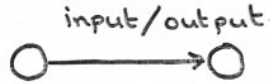
moves.rwth-aachen.de/i2/192

WS 2007/2008

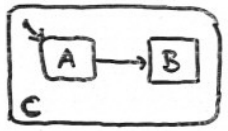
January 7, 2008

STATECHARTS INGREDIENTS

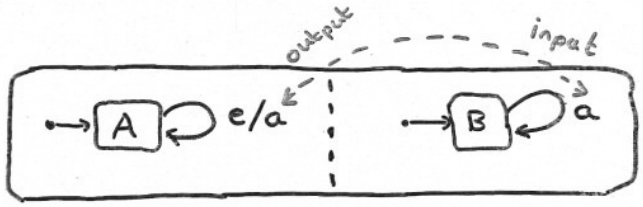
- Mealy machines



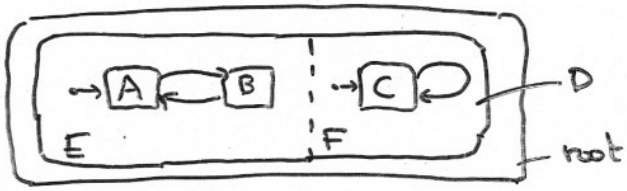
- Hierarchy



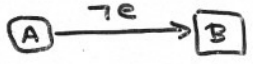
- Communication



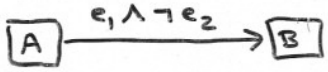
- Concurrency (orthogonality)



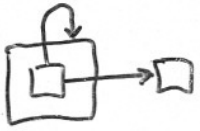
- Negated events



- Compound events



- Edges: intra-level and inter-level



SOME PROBLEMS

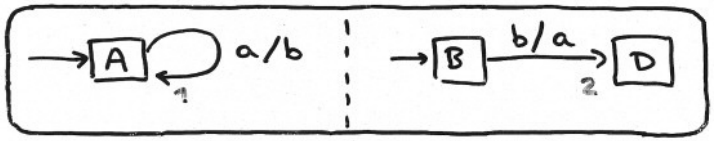
[von der Beeck 1994]

The rich possibilities of statecharts yield a number of problems such as, for instance:

- zero response time assumption

  input and corresponding output occur at the same time

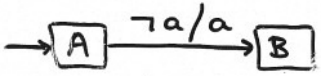
- self-triggering



edge 1 requires event a that (by zero response time) is however generated once edge 1 executes

⟶ edge 1 "triggers" itself

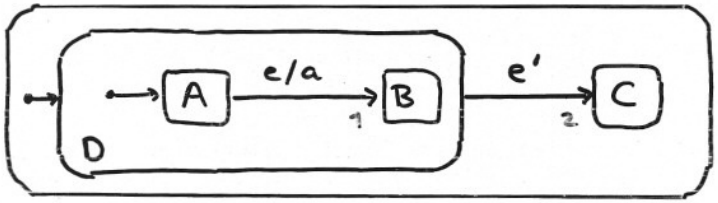
- effect of edge may contradict its cause



event a is generated if it is absent

(see also paradox of last lecture)

SOME PROBLEMS



if "A" is active and events e and e' occur:

1) does edge 2 prevent edge 1 from happening ?
   (i.e., action a does not occur)

or 2) do both edges 1 and 2 occur ?
   (i.e., action a occurs)

or 3) does only edge 1 occur ?

To avoid these problems, certain restrictions are imposed on statecharts, and interpretation is formalized by a formal semantics.

SIMPLIFICATIONS

- No global variables
- No compound events ~~$e \wedge e' \wedge e''$~~
- No negated events ~~$\neg e$~~ ~~$\neg e'$~~
- Communication is biliteral (no broadcast)
- Events generated in "step" i can only be consumed in "step" $i+1$ (and "die" otherwise)

STATECHARTS FORMALLY

A statechart (SC) is a triple (N, E, Edges) with :

- N, a set of nodes structured in a tree
- E, a set of events
- ~~Edges~~, a set of edges (= arrows)

Nodes are sometimes called states, or state nodes

Edges are sometimes called transitions

A system is typically described by a collection of statecharts $(SC_1, SC_2, \ldots, SC_k)$

NODES

Nodes obey a tree structure, defined by function

$$children : N \longrightarrow 2^N$$

$x \in children(y)$ means "x is a child of y"
or "y is the parent of x"

This function induces a partial order $\leq$ on N:

- $x \leq x$
- $x \leq y$ if $x \in children(y)$
- $x \leq y \wedge y \leq z$ implies $x \leq z$

$x \leq y$ means "x is a descendant of y"
or "y is an ancestor of x"

if $x \leq y$ or $y \leq x$, nodes x and y are ancestrally related

There is a unique root node, a node without parent and all nodes are descendats from it (usually simply called "root")

FUNCTIONS ON NODES

The set N is equipped with two functions:

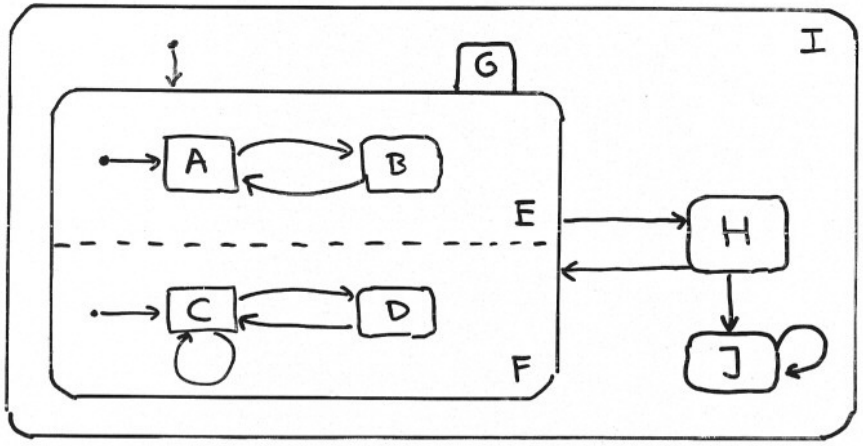
- type: $N \rightarrow \{basic, and, or\}$ with
  - type (root) = or
  - type (x) = basic iff $\underbrace{children(x) = \emptyset}_{\text{Leaf}}$
  - type (x) = AND implies
    $\forall y \in children(x).\ type(y) = OR$

- default: $N \rightarrow N$ is a partial function with $dom(default) = \{x \in N \mid type(x) = or\}$ defined by:
  $default(x) = y \rightsquigarrow y \in children(x)$

type assigns to each node its type

default identifies for each or-node x one of its children as default node that becomes active once x is entered

EXAMPLE



I is the root node

$N = \{A, \ldots, I\}$

children (I) = {G, H, J}

children (G) = {E, F}

.....

default (E) = A

default (F) = C

default (I) = G

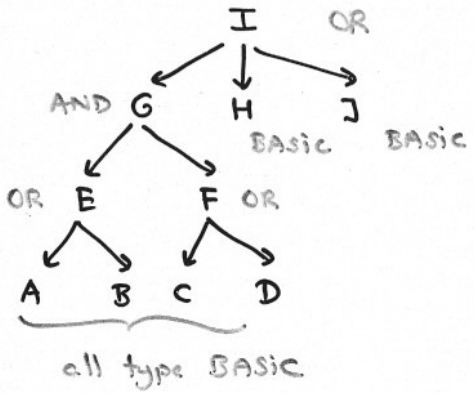
$A \leq I$

$F \leq I$

$C \not\leq H$

.....

Node hierarchy as tree:



EDGES

notation: $X \xrightarrow{e[g]/A} Y$

An edge is a quintuple (X, e, g, A, Y) with:

- $X \subseteq N$ a set of source nodes $X \neq \emptyset$
- $e \in E \cup \{\perp\}$ the trigger event

  ($\perp$ denotes "no event")
- $A \subseteq Act$ a set of actions

  (such as $v := expr$ for local variable v and expression $expr$, or send $j.e$, i.e. send event e to statechart SC_j)
- $g \in Cond(Var)$ a guard

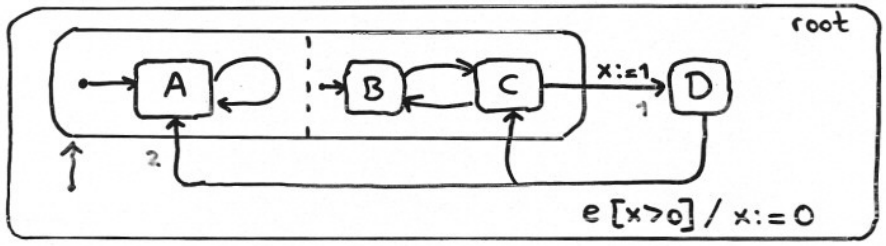
  (a Boolean expression over all variables in $(SC_1, \ldots, SC_j)$)
- $Y \subseteq N$ a set of target nodes $Y \neq \emptyset$

Note: X, Y may contain nodes at different levels (i.e., depths in node tree).

Notation:

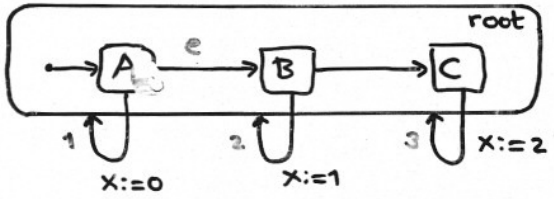
- if guard g holds we may omit it
- if $e = \perp$ we may omit it
- if $A = \emptyset$ we may omit it

EXAMPLE



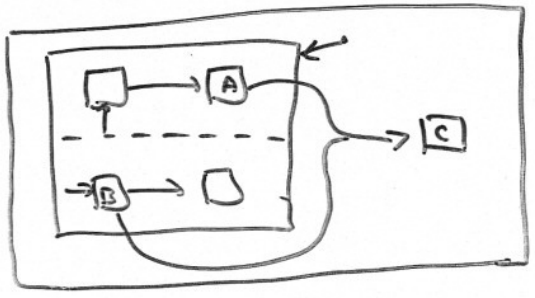
edge 1: $\{C\} \xrightarrow{\perp\,[true]\,/\,x:=1} \{D\}$

edge 2: $\{D\} \xrightarrow{e\,[x>0]\,/\,x:=0} \{A,C\}$



edge 2: $\{B\} \xrightarrow{\perp\,[true]\,/\,x:=?} \{root\}$

Hyperedge:



$\{A,B\} \longrightarrow \{C\}$

SEQUENTIAL STEP SEMANTICS

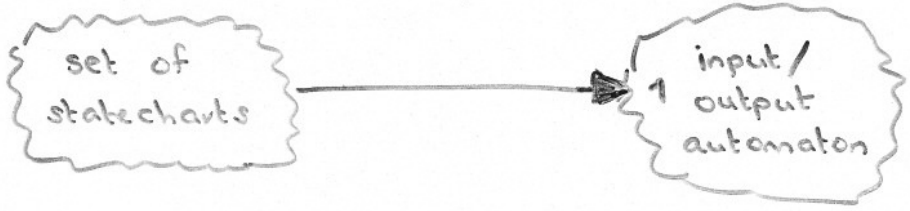
Two notions of a step:

- macro steps : observable "time" steps
- micro steps : a set of edges

A macro step is subdivided in an arbitrary, finite number of micro steps that cannot be prolonged (i.e., extended).

Events generated in macro step i are only available in the next macro step, i.e., $i+1$. If event e is generated in macro step i, but not consumed in macro step $i+1$, it dies, i.e., it is not available in macro steps $i+2$, $i+3$, ...

Semantics:



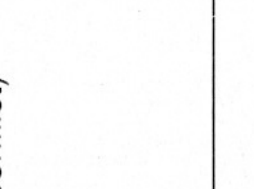
ASSUMPTIONS

- Input to a macro step is a set of events
  - order of event generation is ignored
  - if e,e' are generated in step i, their order is not of relevance in step i+1
- A macro step reacts to all available events
  - events can only be used in macro step immediately succeeding their generation
- Instantaneous edges and actions
- Unlimited concurrency
  - there is no limit on the number of events that can be consumed in a macro step
- Perfect communication
  - messages are not lost

What does a single StateChart mean?

Intuitive semantics as a transition system:

- State = one or more nodes ("current control") + the values of variables
- Edge is enabled if guard holds in current state
- Executing an edge = perform actions A, consume event e,
  - leave source node and switch to target node
  - ⇒ events are unordered, and considered as a set



- Principle: execute as many edges at once (without conflict)
  - ⇒ this is called a *step*

How to compute a step?

STATES AND CONFIGURATIONS

A configuration of SC $(N, E, Edges)$ is a set of nodes $C \subseteq N$ such that:

- root $\in C$
- $x \in C$ and type $(x) =$ OR implies $|\ children(x) \cap C\ | = 1$
- $x \in C$ and type $(x) =$ AND implies $children(x) \subseteq C$

$Conf_i$ is the set of configurations of SC_i;

A state of SC $(N, E, Edges)$ is a triple (C, I, V) with

- $C \in Conf$
- $I \subseteq E$
- V a valuation of the variables

ENABLING OF AN EDGE

Edge $X \xrightarrow{e[g]/A} Y$ is enabled in state (C, I, V) if :

- $X \subseteq C$ all nodes in X are "active"
- $e \in I$ event e is "alive"
- $\big((C_1, \ldots, C_n), (V_1, \ldots, V_n)\big) \models g$ ← guard

  $(C_1, \ldots, C_n)$: configurations of $SC_1, \ldots, SC_n$

  $(V_1, \ldots, V_n)$: valuations of $SC_1, \ldots, SC_n$

i.e., guard g is satisfied

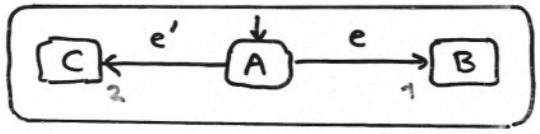
The set of enabled edges in state (C, I, V) is denoted by $En(C, I, V)$

CONSISTENCY

Events that may be performed in a single macro step need to be consistent

Example:

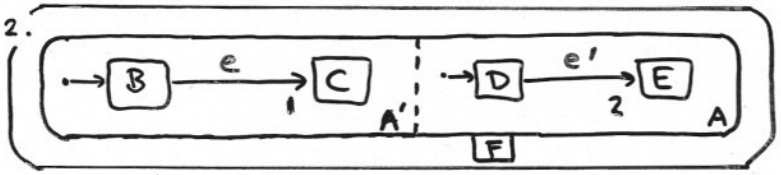
1.



in configuration $\{root, A\}$ and events $\{e, e'\} \in I$ "alive"

only event e or e' can happen in the next macro step. e and e' are inconsistent

2.



in configuration $\{root, F, A, A', B, D\}$ and events $\{e, e'\} \in I$ both events e and e' can happen in next macro step.

e and e' are consistent

CONSISTENCY

- To define consistency we need orthogonality
- For $X \subseteq N$, the least common ancestor, denoted $lca(X)$, is the node $y \in N$ such that:
  - $\forall x \in X.\ x \leq y$
  - $\forall z \in N.\ (\forall x \in X.\ x \leq z) \longrightarrow y \leq z$

In words: y is an ancestor of all nodes in X, and is a descendant of all nodes for which this holds

- Nodes $x, y \in X$ are orthogonal, denoted $x \perp y$, if: $x \not\leq y$ and $y \not\leq x$ and $type(lca(\{x, y\})) = AND$

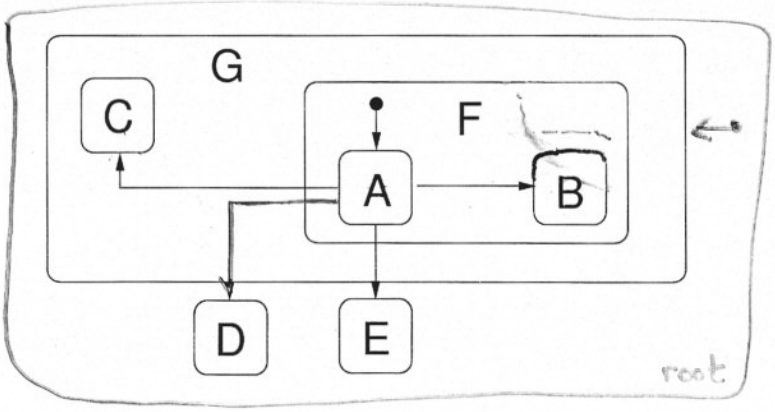
- The scope of edge $X \longrightarrow Y$ is the most nested OR-node that is an ancestor of X and Y

Node z is an ancestor of X if $\forall x \in X.\ x \leq z$

Scope of an edge

Scope of edge $X \longrightarrow Y$ = *most nested* OR-node containing X *and* Y

The most nested node that is *unaffected* by executing the edge



scope$(A \to D) =$ *root* and *scope*$(A \to C) = G$ and *scope*$(A \to B) = F$

CONSISTENCY

Edges $ed, ed' \in$ Edges are consistent if:

$ed = ed'$ or (scope $(ed) \perp$ scope (ed') — their scopes are orthogonal)

A set $T \subseteq$ Edges is consistent if

$\forall\, ed, ed' \in T.$ (ed, ed') is consistent

i.e., if the edges are pairwise consistent.

Examples:

on the black board