Outline

1 Background

2 Ingredients of Statecharts
   - Mealy Machines
   - State Hierarchy
   - Orthogonality
   - Broadcast Communication
   - Some Small Examples
   - Priority and Nondeterminism

3 Semantics of Statecharts

4 Formal Definition of UML Statecharts
Overview

1. Background

2. Ingredients of Statecharts
   - Mealy Machines
   - State Hierarchy
   - Orthogonality
   - Broadcast Communication
   - Some Small Examples
   - Priority and Nondeterminism

3. Semantics of Statecharts

4. Formal Definition of UML Statecharts
- MSCs are a visual modelling formalism for requirements

- Statecharts is a visual modelling formalism for describing the behaviour of discrete-event systems
  - automata + hierarchy + communication + concurrency

≠ CFM
MSCs are a visual modelling formalism for requirements

Statecharts is a visual modelling formalism for describing the behaviour of discrete-event systems
  - automata + hierarchy + communication + concurrency

Developed by David Harel in 1987
  - professor at Weizmann Institute (Israel); co-founder of I-Logix Inc.

Extensively used in embedded systems, automotive and avionics

Variants: UML Statecharts, Stateflow, hierarchical state machines
  - supported by Statemate toolset, and Matlab/Simulink
Overview

1 Background

2 Ingredients of Statecharts
   - Mealy Machines
   - State Hierarchy
   - Orthogonality
   - Broadcast Communication
   - Some Small Examples
   - Priority and Nondeterminism

3 Semantics of Statecharts

4 Formal Definition of UML Statecharts
Statecharts constitute a visual formalism for: [Harel, 1987]

- Describing states and transitions in a modular way reflects the system architecture
What are Statecharts?

Statecharts constitute a visual formalism for:

- Describing states and transitions in a modular way
- Enabling clustering of states

[Harel, 1987]
What are Statecharts?

Statecharts constitute a visual formalism for:

- Describing states and transitions in a modular way
- Enabling clustering of states
- Orthogonality, i.e., concurrency
- Refinement, and

[Harel, 1987]
What are Statecharts?

Statecharts constitute a visual formalism for:

- Describing states and transitions in a modular way
- Enabling clustering of states
- Orthogonality, i.e., concurrency
- Refinement, and
- Encouraging “zoom“ capabilities for moving easily back and forth between levels of abstraction

[Harel, 1987]
What are Statecharts?

Statecharts := Mealy machines
  + State hierarchy
  + Broadcast communication
  + Orthogonality
A Mealy machine $\mathcal{A} = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$ with:

- $Q$ is a finite set of states with initial state $q_0 \in Q$
- $\Sigma$ is the input alphabet
- $\Gamma$ is the output alphabet
- $\delta : Q \times \Sigma \to Q$ is the deterministic (input) transition function, and
- $\omega : Q \times \Sigma \to \Gamma$ is the output function

- no accept state
- but has output
**Definition (Mealy machine)**

A Mealy machine \( \mathcal{A} = (Q, q_0, \Sigma, \Gamma, \delta, \omega) \) with:

- \( Q \) is a finite set of states with initial state \( q_0 \in Q \)
- \( \Sigma \) is the input alphabet
- \( \Gamma \) is the output alphabet
- \( \delta : Q \times \Sigma \to Q \) is the deterministic (input) transition function, and
- \( \omega : Q \times \Sigma \to \Gamma \) is the output function

**Intuition**

A Mealy machine (or: finite-state transducer) is a finite-state automaton that produces output on a transition, based on current input and state.
Mealy machines [Mealy, 1953]

**Definition (Mealy machine)**

A Mealy machine $A = (Q, q_0, \Sigma, \Gamma, \delta, \omega)$ with:

- $Q$ is a finite set of states with initial state $q_0 \in Q$
- $\Sigma$ is the input alphabet
- $\Gamma$ is the output alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the deterministic (input) transition function, and
- $\omega : Q \times \Sigma \rightarrow \Gamma$ is the output function

**Intuition**

A Mealy machine (or: finite-state transducer) is a finite-state automaton that produces output on a transition, based on current input and state.

**Moore machines**

In a Moore machine $\omega : Q \rightarrow \Gamma$, output is purely state-based.
Mealy machines

- No final (accepting) states
- Transitions produce output
- Deterministic input transition function

⇒ Acceptance of input words is not important, but the **generation of output words** from input words is important

**Example**

![Example diagram](image-url)
Limitations of Mealy machines

- No support for hierarchy
  - all states are arranged in a flat fashion
  - no notion of substates

- Realistic systems require complex transition structure and huge number of states
  - scalability problems yields unstructured state diagrams

- No notion of concurrency
  - need for modeling independent components

- No notion of communication between automata.
Scalability

A bit unstructured Mealy machine

An equivalent statechart
Scalability

A bit unstructured Mealy machine

An equivalent statechart

State hierarchy yields modular, hierarchical and structured models.
Orthogonality

Two independent components

Mealy machine for $\textbf{Image} \parallel \textbf{Sound}$

Number of states is exponential in size of concurrent components
Orthogonality

Two independent components

Statechart for Image $\parallel$ Sound

Concurrency modeled by independence
Combined with state hierarchy

Switching on and off the television

name of AND state.

on

off

on

Standby
Output is broadcast that can be received by any other component.

When pushing button 1, channel switches to its state channel 1, while generating signal \( sm \) on which component \( SM \) switches off the sound.
Concurrent

Example concurrency in statecharts

As long as node $X$ is active, nodes $S$ and $T$ are active.

Node $S$ is active when either node $A$ or $B$ is active.

Node $T$ is active if one of $C$, $D$ or $E$ is active.
Concurrency

Example concurrency in statecharts

Exit behaviour

- When node $X$ exits, both nodes $S$ and $T$ exit
- When $Y$ exits, $X$ starts, $S$ starts in $A$, and $T$ starts in $C$
- On the occurrence of event $e$, node $X$ exits (regardless of current state in $S$ or $T$)
Swapping two variables

Swapping the value of variables $x$ and $y$

If nodes $A$ and $C$ are active, assume $x = 1$, $y = 2$

On occurrence of event $e$, $B$ and $D$ are active, and $x = 2$, $y = 1$

$\Rightarrow$ In Harel’s statecharts, memory is shared, i.e., concurrent components have access to shared variables.
What if event $e$ occurs when $A$ and $C$ are active?

Solution:

Add a **priority** mechanism that decides whether:

- **inter-level** transitions (such as $C \rightarrow E$), or
- **intra-level** transitions (such as $A \rightarrow B$)

prevail in case both are enabled.
Nondeterminism

What if event $e$ and $e'$ occur in $A$?

Solution:
Choice is resolved nondeterministically, i.e., the next state is either $B$ or $C$, but not both.
Overview

1 Background

2 Ingredients of Statecharts
   - Mealy Machines
   - State Hierarchy
   - Orthogonality
   - Broadcast Communication
   - Some Small Examples
   - Priority and Nondeterminism

3 Semantics of Statecharts

4 Formal Definition of UML Statecharts
Semantic problems with Statecharts

- Synchrony hypothesis (or: zero response time)
- Self-triggering
- Transition effect is contradicting its cause

Note: [von der Beeck, 1994]

Due to all these problems, hundred(s) (!) of different semantics for Statecharts have been defined in the literature.
Synchrony hypothesis

Event may yield chain of reactions

Note:

- If $A_1$, $B_1$ and $C_1$ are active and event $a$ occurs, a chain of reactions occurs: transition $t_1$ triggers $t_2$, and $t_2$ triggers $t_3$. 
Synchrony hypothesis

Event may yield chain of reactions

Note:
- If $A1$, $B1$ and $C1$ are active and event $a$ occurs, a chain of reactions occurs: transition $t_1$ triggers $t_2$, and $t_2$ triggers $t_3$
- But transitions $t_1$, $t_2$, $t_3$ occur at the same time as events do not take time (except for $after(d)$ events with real $d$)
Simplifications in UML statecharts

1. No shared variables

2. No negated and no compound events (like $e \land e'$)

3. Two-party communication rather than broadcast

4. No synchrony hypothesis:
   - events generated in step $i$ can only be consumed in step $i+1$,
   - and die otherwise, i.e., when they are not consumed in step $i+1$, events disappear
Overview

1 Background

2 Ingredients of Statecharts
   - Mealy Machines
   - State Hierarchy
   - Orthogonality
   - Broadcast Communication
   - Some Small Examples
   - Priority and Nondeterminism

3 Semantics of Statecharts

4 Formal Definition of UML Statecharts
**Definition (Statecharts)**

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

1. $N$ is a set of nodes (or: states) structured in a tree
2. $E$ is a set of events
   - pseudo-event $after(d)$ denotes a delay of $d \in \mathbb{R}_{\geq 0}$ time units
   - $\bot \not\in E$ stands for “no event available”
3. $Edges$ is a set of (hyper-) edges, defined later on.
Statecharts

Definition (Statecharts)

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

1. $N$ is a set of nodes (or: states) structured in a tree
2. $E$ is a set of events
   - pseudo-event $after(d)$ denotes a delay of $d \in \mathbb{R}_{\geq 0}$ time units
   - $\perp \notin E$ stands for “no event available”
3. $Edges$ is a set of (hyper-) edges, defined later on.

Definition (System)

A system is described by a finite collection of statecharts $(SC_1, \ldots, SC_k)$. 
Syntactic sugar

*This is an elementary form; the UML allows more constructs that can be defined in terms of these basic elements*

- Deferred events simulate by regeneration
- Parametrised events simulate by set of parameter-less events
- Activities that take time simulate by start and end event
- Dynamic choice points simulate by intermediate state
- Synchronization states use a hyperedge with a counter
- History states (re)define an entry point
Tree structure

**Function children**

Nodes obey a tree structure defined by function $\text{children} : N \rightarrow 2^N$ where $x \in \text{children}(y)$ means that $x$ is a child of $y$, or equivalently, $y$ is the parent of $x$.

**Partial order $\sqsubseteq$**

The partial order $\sqsubseteq \subseteq N \times N$ is defined by:

- $\forall x \in N. x \sqsubseteq x$
- $\forall x, y \in N. x \sqsubseteq y$ if $x \in \text{children}(y)$
- $\forall x, y, z \in N. x \sqsubseteq y \land y \sqsubseteq z \Rightarrow x \sqsubseteq z$

$x \sqsubseteq y$ means that $x$ is a descendant of $y$, or equivalently, $y$ is an ancestor of $x$. If $x \sqsubseteq y$ or $y \sqsubseteq x$, nodes $x$ and $y$ are ancestrally related.

**Root node**

There is a unique root with no ancestors, and $\forall x \in N. x \sqsubseteq \text{root}$. 
Functions on nodes

The type of nodes

Nodes are typed, \( \text{type}(x) \in \{ \text{BASIC, AND, OR} \} \) such that for \( x \in N \):

- \( \text{type}(\text{root}) = \text{OR} \)
- \( \text{type}(x) = \text{BASIC} \) iff \( \text{children}(x) = \emptyset \), i.e., \( x \) is a leaf
- \( \text{type}(x) = \text{AND} \) implies \( (\forall y \in \text{children}(x). \text{type}(y) = \text{OR}) \)

Default nodes

\( \text{default} : N \rightarrow N \) is a partial function on domain \( \{ x \in N \mid \text{type}(x) = \text{OR} \} \) such that

\[ \text{default}(x) = y \quad \text{implies} \quad y \in \text{children}(x). \]

The function \( \text{default} \) assigns to each OR-node \( x \) one of its children as \( \text{default} \) node that becomes active once \( x \) becomes active.
default (E) = A
default (F) = C
Definition (Edges)

An edge is a quintuple \((X, e, g, A, Y)\), denoted \(X \xrightarrow{e[g]/A} Y\) with:

- \(X \subseteq N\) is a set of source nodes with \(X \neq \emptyset\)
- \(e \in E \cup \{\bot\}\) is the trigger event
- \(A \subseteq \text{Act}\) is a set of actions
  - such as \(v := \text{expr}\) or local variable \(v\) and expression \(\text{expr}\)
  - or \(\text{send} \ j.e\), i.e., send event \(e\) to statechart \(\text{SC}_j\)
- Guard \(g\) is a Boolean expression over all variables in \((\text{SC}_1, \ldots, \text{SC}_k)\)
- \(Y \subseteq N\) is a set of target nodes with \(Y \neq \emptyset\)
Definition (Edges)

An edge is a quintuple \((X, e, g, A, Y)\), denoted \(X \xrightarrow{e[g]/A} Y\) with:

- \(X \subseteq N\) is a set of source nodes with \(X \neq \emptyset\)
- \(e \in E \cup \{\bot\}\) is the trigger event
- \(A \subseteq \text{Act}\) is a set of actions
  - such as \(v := \text{expr}\) or local variable \(v\) and expression \(\text{expr}\)
  - or \(\text{send \(j.e\)}\), i.e., send event \(e\) to statechart \(SC_j\)
- **Guard** \(g\) is a Boolean expression over all variables in \((SC_1, \ldots, SC_k)\)
- \(Y \subseteq N\) is a set of target nodes with \(Y \neq \emptyset\)

The sets \(X\) and \(Y\) may contain nodes at different depth in the node tree.
Example statechart

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

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

edge 1: \( \{A\} \xrightarrow{e[\text{true}]/\emptyset} \{B\} \)

edge 2: \( \{B\} \xrightarrow{\perp[\text{true}]/\{x:=1\}} \{\text{root}\} \)

reset root in node A.
Example statechart

edge : \{ A, B \} \longrightarrow \{ C \}