Software Design, Modelling and Analysis in UML

Lecture 12: Core State Machines II

2016-12-13

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Content

- **Actions**
  - **transformer:**
    - **send** message
    - **create/destroy**: later
- **Labelled Transition System**
- **Transitions** of UML State Machines
  - **discard** event,
  - **dispatch** event,
  - **continue** RTC,
  - **environment** interaction,
  - **error** condition.
- **Example Revisited**

Transformer

Transformer

Definition.
Let $\Sigma_{\mathscr{S}}^{\mathscr{D}}$ the set of system configurations over some $\mathscr{S}_0, \mathscr{D}_0, Eth$.

We call a relation

$$t \subseteq \Big(\mathscr{D}(\mathscr{C}) \times (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times Eth)\Big) \times (\Sigma_{\mathscr{S}}^{\mathscr{D}} \times Eth)$$

a (system configuration) **transformer**.

Example:

- $t[u_x](\sigma, \varepsilon) \subseteq \Sigma_{\mathscr{S}}^{\mathscr{D}} \times Eth$ is
  - the set (!) of the **system configurations**
  - which **may** result from **object** u_x
  - **executing** transformer t.
- $t_{\texttt{skip}}[u_x](\sigma, \varepsilon) = \{(\sigma, \varepsilon)\}$
- $t_{\texttt{create}}[u_x](\sigma, \varepsilon)$: add a previously non-alive object to σ

Observations

- In the following, we assume that
  - each application of a transformer t
  - to some system configuration (σ, ε)
  - for object u_x

  is associated with a set of **observations**

  $$Obs_t[u_x](\sigma, \varepsilon) \in 2^{(\mathscr{D}(\mathscr{E}) \,\dot{\cup}\, \{*,+\}) \times \mathscr{D}(\mathscr{C})}.$$

- An observation

  $$(u_e, u_{dst}) \in Obs_t[u_x](\sigma, \varepsilon)$$

  represents the information that,
  as a "side effect" of object u_x executing t in system configuration (σ, ε),
  the event u_e has been sent to u_{dst}.

  Special cases: creation ('$*$') / destruction ('$+$').

A Simple Action Language

In the following we use

$$\begin{aligned} Act_{\mathscr{S}} = \{&\texttt{skip}\} \\ &\cup \{\texttt{update}(expr_1, v, expr_2) \mid expr_1, expr_2 \in Expr_{\mathscr{S}}, v \in atr\} \\ &\cup \{\texttt{send}(E(expr_1, ..., expr_n), expr_{dst}) \mid expr_i, expr_{dst} \in Expr_{\mathscr{S}}, E \in \mathscr{E}\} \\ &\cup \{\texttt{create}(C, expr, v) \mid C \in \mathscr{C}, expr \in Expr_{\mathscr{S}}, v \in V\} \\ &\cup \{\texttt{destroy}(expr) \mid expr \in Expr_{\mathscr{S}}\} \end{aligned}$$

and OCL expressions over $\mathscr{S}$ (with partial interpretation) as $Expr_{\mathscr{S}}$.

Transformer: Skip

abstract syntax **concrete syntax**

`skip` skip

intuitive semantics

do nothing

well-typedness

./.

semantics

$$t_{\texttt{skip}}[u_x](\sigma, \varepsilon) = \{(\sigma, \varepsilon)\}$$

observables

$$Obs_{\texttt{skip}}[u_x](\sigma, \varepsilon) = \emptyset$$

(error) conditions

Transformer: Update

(self.x := self.x + 1) x := x+1

abstract syntax **concrete syntax**

$\texttt{update}(expr_1, v, expr_2)$ expr₁.v := expr₂

intuitive semantics

Update attribute v in the object denoted by $expr_1$ to the value denoted by $expr_2$.

well-typedness

$expr_1 : T_C$ and $v : T \in atr(C)$; $expr_2 : T$;
$expr_1, expr_2$ obey visibility and navigability

semantics

either does not change

$$t_{\texttt{update}(expr_1, v, expr_2)}[u_x](\sigma, \varepsilon) = \{(\sigma', \varepsilon)\}$$

change local state of object u

change value of this attr.

where $\sigma' = \sigma[u \mapsto \sigma(u)[v \mapsto \underbrace{I[\![expr_2]\!](\sigma, u_x)}_{\text{new value}}]]$ with

$$u = \underbrace{I[\![expr_1]\!](\sigma, u_x)}.$$

object denoted by expr₁ (relative to uₓ as self)

observables

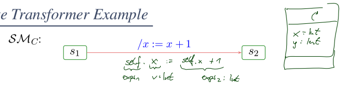
$$Obs_{\texttt{update}(expr_1, v, expr_2)}[u_x] = \emptyset$$

(error) conditions

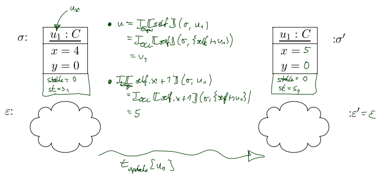
Not defined if $I[\![expr_1]\!](\sigma, u_x)$ or $I[\![expr_2]\!](\sigma, u_x)$ not defined.

Update Transformer Example

$\mathcal{SM}_C$:



$$t_{\texttt{update}(expr_1, v, expr_2)}[u_x](\sigma, \varepsilon) = (\sigma' = \sigma[u \mapsto \sigma(u)[v \mapsto \underbrace{I[\![expr_2]\!](\sigma, u_x)}]], \varepsilon), u = I[\![expr_1]\!](\sigma, u_x)$$



- $u = I_{Expr}[\![self]\!](\sigma, u_1)$
  $= I_{OCL}[\![self]\!](\sigma, \{self \mapsto u_1\})$
  $= u_1$
- $I_{Expr}[\![self.x + 1]\!](\sigma, u_1)$
  $= I_{OCL}[\![self.x+1]\!](\sigma, \{self \mapsto u_1\})$
  $= 5$

Transformer: Send

abstract syntax **concrete syntax**

$\texttt{send}(E(expr_1, ..., expr_n), expr_{dst})$ expr_dst ! E(expr₁, ..., exprₙ)

intuitive semantics

Object $u_x : C$ sends event E to object $expr_{dst}$, i.e. create a fresh signal instance, fill in its attributes, and place it in the ether.

well-typedness

$E \in \mathscr{E}; atr(E) = \{v_1 : T_1, \dots, v_n : T_n\}; expr_i : T_i, 1 \leq i \leq n;$
$expr_{dst} : T_D, C, D \in \mathscr{C} \setminus \mathscr{E};$
all expressions obey visibility and navigability in C

semantics

$$(\sigma', \varepsilon') \in t_{\texttt{send}(E(expr_1, \dots, expr_n), expr_{dst})}[u_x](\sigma, \varepsilon)$$

① if $\sigma' = \sigma \,\dot\cup\, \{u_E \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}\}$; $\varepsilon' = \varepsilon \oplus (u_{dst}, u_E)$;

if $u_{dst} = I[\![expr_{dst}]\!](\sigma, u_x) \in \mathrm{dom}(\sigma)$; $d_i = I[\![expr_i]\!](\sigma, u_x)$ for $1 \leq i \leq n$;

$u_E \in \mathscr{D}(E)$ a fresh identity, i.e. $u_E \notin \mathrm{dom}(\sigma)$,

② and where $(\sigma', \varepsilon') = (\sigma, \varepsilon)$ if $u_{dst} \notin \mathrm{dom}(\sigma)$. } sending to a non-alive object: do nothing

observables

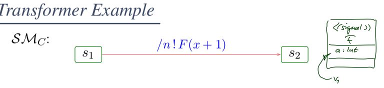
$$Obs_{\texttt{send}}[u_x] = \{(u_E, u_{dst})\}$$

(error) conditions

$I[\![expr]\!](\sigma, u_x)$ not defined for any $expr \in \{expr_{dst}, expr_1, \dots, expr_n\}$

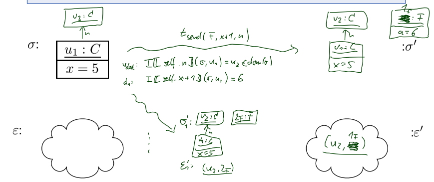
Send Transformer Example

$\mathcal{SM}_C$:



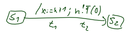
$$t_{\texttt{send}(\cancel{expr_{src}}, E(expr_1, \dots, expr_n), expr_{dst})}[u_x](\sigma, \varepsilon) \ni (\sigma', \varepsilon') \text{ iff } \varepsilon' = \varepsilon \oplus (u_{dst}, u_E);$$

① $\sigma' = \sigma \,\dot\cup\, \{u_E \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}\}; u_{dst} = I[\![expr_{dst}]\!](\sigma, u_x) \in \mathrm{dom}(\sigma);$
$d_i = I[\![expr_i]\!](\sigma, u_x), 1 \leq i \leq n; u_E \in \mathscr{D}(E)$ a fresh identity;



u_{dst}: $I[\![self.n]\!](\sigma, u_1) = u_2 \in \mathrm{dom}(\sigma)$
d_1: $I[\![self.x+1]\!](\sigma, u_1) = 6$

Sequential Composition of Transformers



- **Sequential composition** $t_1 \circ t_2$ of transformers t_1 and t_2 is canonically defined as

  $$(t_2 \circ t_1)[u_x](\sigma, \varepsilon) = t_2[u_x](t_1[u_x](\sigma, \varepsilon))$$

  with observation

  $$Obs_{(t_2 \circ t_1)}[u_x](\sigma, \varepsilon) = Obs_{t_1}[u_x](\sigma, \varepsilon) \cup Obs_{t_2}[u_x](t_1(\sigma, \varepsilon)).$$

- **Clear**: not defined if one the two intermediate "micro steps" is not defined.

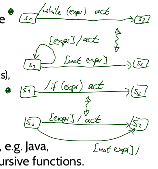
Transformers And Denotational Semantics

Observation: our transformers are in principle the **denotational semantics** of the actions/action sequences. The trivial case, to be precise.

Note: with the previous examples, we can capture

- empty statements, skips,
- assignments,
- conditionals (by normalisation and auxiliary variables),
- create/destroy (later),

but not **possibly diverging loops**.

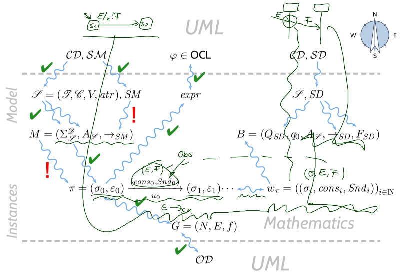


Our (Simple) Approach: if the action language is, e.g. Java, then (**syntactically**) forbid loops and calls of recursive functions.

Other Approach: use full blown denotational semantics.

No show-stopper, because loops in the action annotation can be converted into transition cycles in the state machine.

Course Map



Transition Relation

Transition Relation, Computation

Definition. Let A be a set of **labels** and S a (not necessarily finite) set of of **states**. We call

$$\to \; \subseteq S \times A \times S$$

a (labelled) **transition relation**.

Let $S_0 \subseteq S$ be a set of **initial states**. A (finite or infinite) sequence

$$\overset{\overset{S_0}{\Psi}}{s_0} \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \ldots$$

$\in \to$ $\in \to$

with $s_i \in S$, $a_i \in A$ is called **computation** of the **labelled transition system** $(S, A, \to, S_0)$ if and only if

trans. relation

states labels init. states

- **initiation**: $s_0 \in S_0$
- **consecution**: $(s_i, a_i, s_{i+1}) \in \to$ for $i \in \mathbb{N}_0$.

Active vs. Passive Classes/Objects

- **Note**: From now on, for simplicity, assume that all classes are **active**.

  We'll later briefly discuss the Rhapsody framework which proposes a way how to integrate non-active objects.

- **Note**: The following RTC "algorithm" follows Harel and Gery (1997) (i.e. the one realised by the Rhapsody code generation) if the standard is ambiguous or leaves choices.

From Core State Machines to LTS

Definition. Let $\mathscr{S}_0 = (\mathscr{T}_0, \mathscr{C}_0, V_0, atr_0, \mathscr{E})$ be a signature with signals (all classes in $\mathscr{C}_0$ **active**), $\mathscr{D}_0$ a structure of $\mathscr{S}_0$, and $(Eth, ready, \oplus, \ominus, [\,\cdot\,])$ an ether over $\mathscr{S}_0$ and $\mathscr{D}_0$. Assume there is one core state machine M_C per class $C \in \mathscr{C}$.

We say, the state machines **induce** the following labelled transition relation on states $S := \underbrace{(\Sigma_{\mathscr{S}}^{\mathscr{D}} \times Eth) \,\dot\cup\, \{\#\}}$ with labels $A := \underbrace{2^{\mathscr{D}(\mathscr{E})}}_{\text{consumed sig. inst.}} \times \underbrace{2^{(\mathscr{D}(\mathscr{E}) \,\dot\cup\, \{*,+\}) \times \mathscr{D}(\mathscr{C})}} \times \underbrace{\mathscr{D}(\mathscr{C})}$:

error state

- $(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$

  observation what has been sent out (plus create/destroy) by actions

  who (which object) did do the transition

  if and only if

  (i) an event with destination u is **discarded**, or

  (ii) an event is **dispatched** to u, i.e. stable object processes an event, or

  (iii) run-to-completion processing by u **continues**, i.e. object u is not stable and continues to process an event, or

  (iv) the **environment** interacts with object u, or

- $s \xrightarrow[u]{(cons, \emptyset)} \#$

  if and only if

  (v) an **error condition** occurs during consumption of $cons$, or $s = \#$ and $cons = \emptyset$.

(i) Discarding An Event

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

condition on (σ, ε)

and

conditions on (σ', ε')

(i) Discarding An Event

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- an E-event (instance of signal E) is ready in ε for object u of a class $\mathscr{C}$, i.e. if

$$u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C) \wedge \exists\, u_E \in \mathscr{D}(E) : u_E \in ready(\varepsilon, u)$$

- u is stable and in state machine state s, i.e. $\sigma(u)(stable) = 1$ and $\sigma(u)(st) = s$,
- but there is no corresponding transition enabled (all transitions incident with current state of u either have other triggers or the guard is not satisfied)

$$\forall\, (s, F, expr, act, s') \in\to (\mathcal{SM}_C) : F \neq E \vee I[\![expr]\!](\sigma, u) = 0$$

and

- in the system configuration, stability may change, u_E goes away, i.e.

$$\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$$

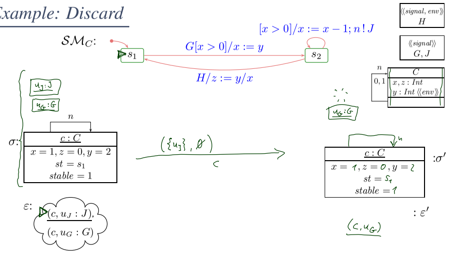
  where $b = 0$ if and only if there is a transition **with trigger '_'** enabled for u in (σ', ε').
- the event u_E is removed from the ether, i.e.

$$\varepsilon' = \varepsilon \ominus u_E,$$

- consumption of u_E is observed, i.e.

$$cons = \{u_E\}, \quad Snd = \emptyset.$$

Example: Discard



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$ ✓
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$ ✓
- $\forall\, (s, F, expr, act, s') \in\to (\mathcal{SM}_C)$:
  $F \neq E \vee I[\![expr]\!](\sigma, u) = 0$ ✓
- $\sigma(u)(stable) = 1$ ✓, $\sigma(u)(st) = s$, ✓
- $\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$
- $\varepsilon' = \varepsilon \ominus u_E$
- $cons = \{u_E\}, \quad Snd = \emptyset$

(ii) Dispatch

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C) \wedge \exists\, u_E \in \mathscr{D}(E) : u_E \in ready(\varepsilon, u)$
- u is stable and in state machine state s, i.e. $\sigma(u)(stable) = 1$ and $\sigma(u)(st) = s$,
- a transition is **enabled**, i.e.

$$\exists\, (s, F, expr, act, s') \in\to (\mathcal{SM}_C) : F = E \wedge I[\![expr]\!](\tilde{\sigma}, u) = 1$$

  where $\tilde{\sigma} = \sigma[u.params_E \mapsto u_E]$. e.g. $\boxed{s_1} \xrightarrow{E[params_E.x>0]/} \boxed{s_2}$

and

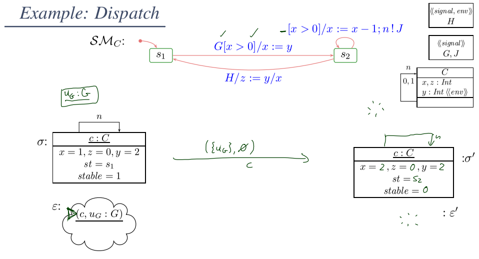
- (σ', ε') results from applying t_{act} to (σ, ε) and removing u_E from the ether, i.e.

$$(\sigma'', \varepsilon') \in t_{act}[u](\tilde{\sigma}, \varepsilon \ominus u_E),$$
$$\sigma' = (\sigma''[u.st \mapsto s', u.stable \mapsto b, u.params_E \mapsto \emptyset])|_{\overbrace{\mathscr{D}(\mathscr{C}) \setminus \{u_E\}}^{\text{remove } u_E}}$$

  where b **depends** (see (i))
- Consumption of u_E and the side effects of the action are observed, i.e.

$$cons = \{u_E\}, \quad Snd = Obs_{t_{act}}[u](\tilde{\sigma}, \varepsilon \ominus u_E).$$

Example: Dispatch



- $u \in \mathrm{dom}(\sigma)$ ✓ $\cap\, \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E)$ ✓, $u_E \in ready(\varepsilon, u)$ ✓
- ~~$\forall\, (s, F, expr, act, s') \in\to (\mathcal{SM}_C)$:~~
  ~~$F \neq E \vee I[\![expr]\!](\sigma, u) = 0$~~
- ~~$\sigma(u)(stable) = 1, \sigma(u)(st) = s,$~~
- ~~$\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$~~
- ~~$\varepsilon' = \varepsilon \ominus u_E$~~
- ~~$cons = \{u_E\}, \quad Snd = \emptyset$~~

(iii) Continue Run-to-Completion

$$(\sigma, \varepsilon) \xrightarrow[u]{(cons, Snd)} (\sigma', \varepsilon')$$

if

- there is an unstable object u of a class $\mathscr{C}$, i.e.

$$u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C) \wedge \sigma(u)(stable) = 0$$

- there is a transition without trigger enabled from the current state $s = \sigma(u)(st)$, i.e.

$$\exists\, (s, \_, expr, act, s') \in\to (\mathcal{SM}_C) : I[\![expr]\!](\sigma, u) = 1$$

and

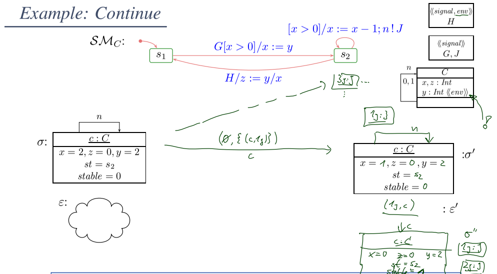
- (σ', ε') results from applying t_{act} to (σ, ε), i.e.

$$(\sigma'', \varepsilon') \in t_{act}[u](\sigma, \varepsilon), \quad \sigma' = \sigma''[u.st \mapsto s', u.stable \mapsto b]$$

  where b **depends** as before.
- Only the side effects of the action are observed, i.e.

$$cons = \emptyset, \quad Snd = Obs_{t_{act}}[u](\sigma, \varepsilon).$$

Example: Continue



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$
- ~~$\forall (s, F, expr, act, s') \in\rightarrow (\mathcal{SM}_C)$:
  $F \neq E \vee I[\![expr]\!](\sigma, u) = 0$~~

- ~~$\sigma(u)(stable) = 1, \sigma(u)(st) = s,$~~
- ~~$\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$~~
- ~~$\varepsilon' = \varepsilon \ominus u_E$~~
- ~~$cons = \{u_E\}, \quad Snd = \emptyset$~~

(iv) Environment Interaction

Assume that a set $\mathscr{E}_{env} \subseteq \mathscr{E}$ is designated as **environment events** and a set of attributes $V_{env} \subseteq V$ is designated as **input attributes**.

Then

$$(\sigma, \varepsilon) \xrightarrow[env]{(cons, Snd)} (\sigma', \varepsilon')$$

dedicated label

if either (!)

- an environment event $E \in \mathscr{E}_{env}$ is spontaneously sent to an alive object $u \in \mathrm{dom}(\sigma)$, i.e.

  $$\sigma' = \sigma \mathbin{\dot\cup} \{u_E \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}\}, \quad \varepsilon' = \varepsilon \oplus (u, u_E)$$

  where $u_E \notin \mathrm{dom}(\sigma)$ and $atr(E) = \{v_1, \ldots, v_n\}$.
- Sending of the event is observed, i.e. $cons = \emptyset$, $Snd = \{u_E,)\}$.

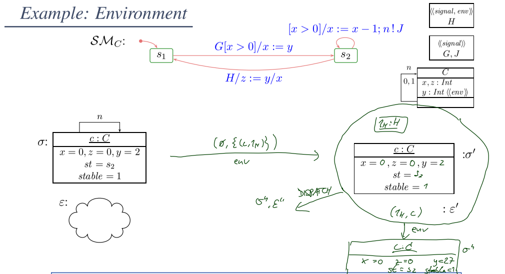
or

- Values of input attributes change freely in alive objects, i.e.

  $$\forall\, v \in V\ \forall\, u \in \mathrm{dom}(\sigma) : \sigma'(u)(v) \neq \sigma(u)(v) \implies v \in V_{env}.$$

  and no objects appear or disappear, i.e. $\mathrm{dom}(\sigma') = \mathrm{dom}(\sigma)$.
- $\varepsilon' = \varepsilon$.

Example: Environment



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$
- $\forall (s, F, expr, act, s') \in\rightarrow (\mathcal{SM}_C)$:
  $F \neq E \vee I[\![expr]\!](\sigma, u) = 0$

- $\sigma(u)(stable) = 1, \sigma(u)(st) = s,$
- $\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$
- $\varepsilon' = \varepsilon \ominus u_E$
- $cons = \{u_E\}, \quad Snd = \emptyset$

(v) Error Conditions

$$s \xrightarrow[u]{(cons, Snd)} \#$$

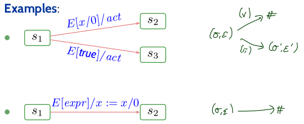
if, in (i), (ii), or (iii),

- $I[\![expr]\!]$ is not defined for σ and u, or
- $t_{act}[u]$ is not defined for (σ, ε),

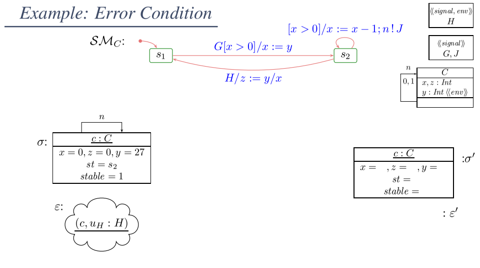
and

- $cons = \emptyset$, and $Snd = \emptyset$.

Examples:



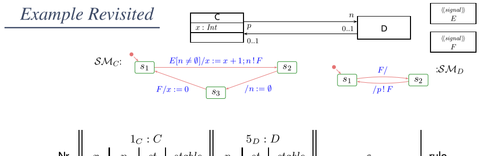
Example: Error Condition



- $u \in \mathrm{dom}(\sigma) \cap \mathscr{D}(C)$
  $u_E \in \mathscr{D}(E), u_E \in ready(\varepsilon, u)$
- $\forall (s, F, expr, act, s') \in\rightarrow (\mathcal{SM}_C)$:
  $F \neq E \vee I[\![expr]\!](\sigma, u) = 0$

- $\sigma(u)(stable) = 1, \sigma(u)(st) = s,$
- $\sigma' = \sigma[u.stable \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}$
- $\varepsilon' = \varepsilon \ominus u_E$
- $cons = \{u_E\}, \quad Snd = \emptyset$

Example Revisited



	$1_C : C$				$5_D : D$				
Nr.	x	n	st	$stable$	p	st	$stable$	ε	rule
0	27	5_D	s_1	1	1_C	s_1	1	$(3_F, 1_C).(2_E, 1_C)$	

Tell Them What You've Told Them…

- **State Machines** induce a **labelled transition system**.
- There are five kinds of transitions in the LTS:
  - **discard**: no matching state machine edge enabled, may change stability.
  - **dispatch**: a matching state machine edge is taken, i.e. actions are executed (according to transformers),
  - **continue**: a state machine edge without signal-trigger is enabled, and is taken,
  - **environment interaction**: dedicated environment signals are injected into the event pool,
  - **error condition**: a designated error state is assumed, maybe due to undefined action transformers.
- For now, we assume that all classes are active, thus steps of objects may interleave.

References

References

Harel, D. and Gery, E. (1997). Executable object modeling with statecharts. *IEEE Computer*, 30(7):31–42.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.