

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

- **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

Definition.

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

We call a relation

$$t \subseteq \left(\mathcal{D}(\mathcal{C}) \times (\Sigma_{\mathcal{D}} \times Eth) \right) \times (\Sigma_{\mathcal{D}} \times Eth)$$

a (system configuration) **transformer**.

Example:

- $t[u_x](\sigma, \varepsilon) \subseteq \Sigma_{\mathcal{D}} \times Eth$ is
 - the set (!) of the **system configurations**
 - which may result from **object** u_x
 - **executing** transformer t .
- $t_{\text{skip}}[u_x](\sigma, \varepsilon) = \{(\sigma, \varepsilon)\}$
- $t_{\text{create}}[u_x](\sigma, \varepsilon)$: 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**

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

- 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

$$Act_{\mathcal{S}} = \{\text{skip}\}$$

$$\cup \{\text{update}(expr_1, v, expr_2) \mid expr_1, expr_2 \in Expr_{\mathcal{S}}, v \in atr\}$$

$$\cup \{\text{send}(E(expr_1, \dots, expr_n), expr_{dst}) \mid expr_i, expr_{dst} \in Expr_{\mathcal{S}}, E \in \mathcal{E}\}$$

$$\cup \{\text{create}(C, expr, v) \mid C \in \mathcal{C}, expr \in Expr_{\mathcal{S}}, v \in V\}$$

$$\cup \{\text{destroy}(expr) \mid expr \in Expr_{\mathcal{S}}\}$$

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

Transformer: Skip

abstract syntax	concrete syntax
skip	<i>skip</i>
intuitive semantics	<i>do nothing</i>
well-typedness	<i>./.</i>
semantics	$t_{\text{skip}}[u_x](\sigma, \varepsilon) = \{(\sigma, \varepsilon)\}$
observables	$Obs_{\text{skip}}[u_x](\sigma, \varepsilon) = \emptyset$
(error) conditions	

Transformer: Update

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

abstract syntax

$update(expr_1, v, expr_2)$

concrete syntax

$expr_1.v := expr_2$

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

this does not change

semantics

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

change state of object u

where $\sigma' = \sigma[u \mapsto \sigma(u)[v \mapsto I[expr_2](\sigma, u_x)]]$ with

change value of this attr.

new value

$u = I[expr_1](\sigma, u_x)$.

object denoted by $expr_1$ (relative to u_x as self)

observables

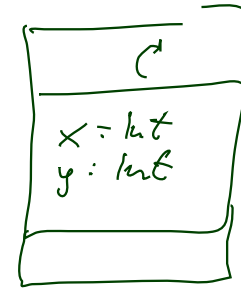
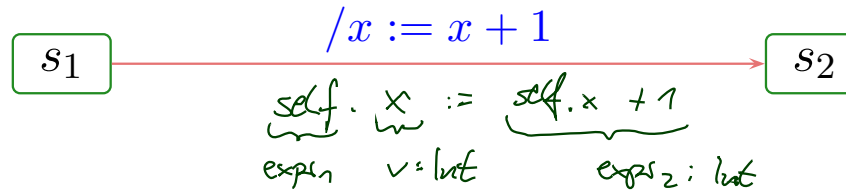
$Obs_{update(expr_1, v, expr_2)}[u_x] = \emptyset$

(error) conditions

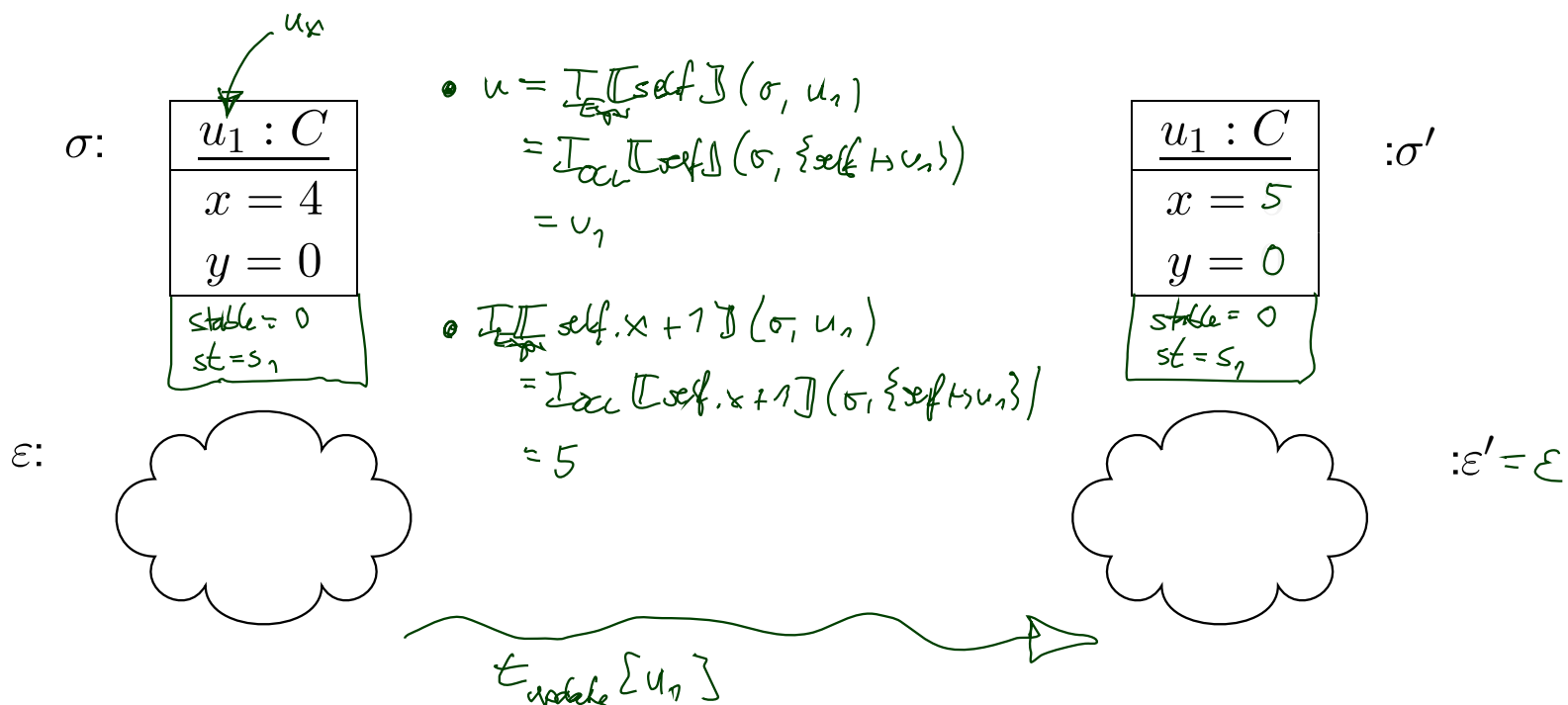
Not defined if $I[expr_1](\sigma, u_x)$ or $I[expr_2](\sigma, u_x)$ not defined.

Update Transformer Example

SMC:



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



Transformer: Send

abstract syntax

$\text{send}(E(\text{expr}_1, \dots, \text{expr}_n), \text{expr}_{dst})$

concrete syntax

$\text{expr}_{dst} ! E(\text{expr}_1, \dots, \text{expr}_n)$

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

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

semantics

$(\sigma', \varepsilon') \in t_{\text{send}(E(\text{expr}_1, \dots, \text{expr}_n), \text{expr}_{dst})}[u_x](\sigma, \varepsilon)$

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

if $u_{dst} = I[\llbracket \text{expr}_{dst} \rrbracket](\sigma, u_x) \in \text{dom}(\sigma); \quad d_i = I[\llbracket \text{expr}_i \rrbracket](\sigma, u_x)$ for
 $1 \leq i \leq n;$

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

② and where $(\sigma', \varepsilon') = (\sigma, \varepsilon)$ if $u_{dst} \notin \text{dom}(\sigma).$ $\left. \begin{array}{l} \} \text{ sending to a non-alive} \\ \} \text{ object: do nothing} \end{array} \right\}$

observables

$\text{Obs}_{\text{send}}[u_x] = \{(u_{\underline{E}}, u_{dst})\}$

(error) conditions

$I[\llbracket \text{expr} \rrbracket](\sigma, u_x)$ not defined for any $\text{expr} \in \{\text{expr}_{dst}, \text{expr}_1, \dots, \text{expr}_n\}$

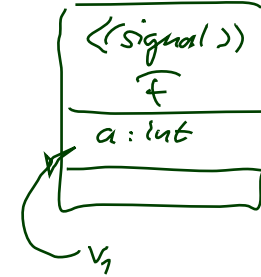
Send Transformer Example

SM_C :

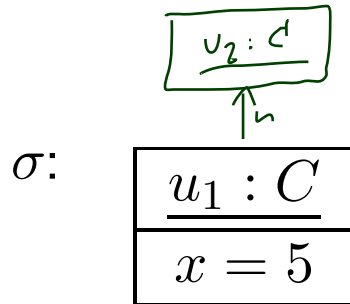
s_1

$/n! F(x+1)$

s_2

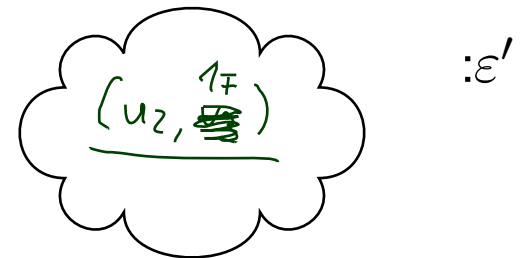
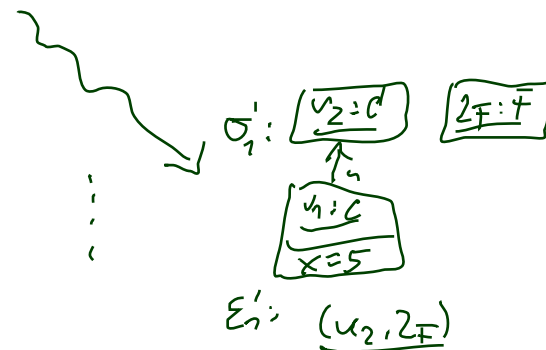
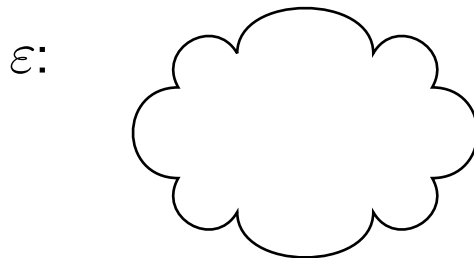
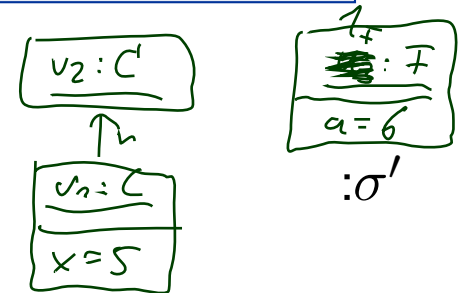


$t_{\text{send}}(\text{expr}_{\text{src}}, E(\text{expr}_1, \dots, \text{expr}_n), \text{expr}_{\text{dst}})[u_x](\sigma, \varepsilon) \ni (\sigma', \varepsilon')$ iff $\varepsilon' = \varepsilon \oplus (u_{\text{dst}}, u_{\varepsilon})$;
 ① $\sigma' = \sigma \dot{\cup} \{u_{\varepsilon} \mapsto \{v_i \mapsto d_i \mid 1 \leq i \leq n\}\}$; $u_{\text{dst}} = I[\text{expr}_{\text{dst}}](\sigma, u_x) \in \text{dom}(\sigma)$;
 $d_i = I[\text{expr}_i](\sigma, u_x), 1 \leq i \leq n$; $u_{\varepsilon} \in \mathcal{D}(E)$ a fresh identity;

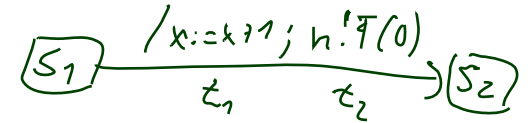


$t_{\text{send}}(\bar{F}, x+1, u)$

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



Sequential Composition of Transformers



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

$$(t_2 \circ t_1)[u_x](\sigma, \varepsilon) = t_2[u_x](\underbrace{t_1[u_x](\sigma, \varepsilon)}_{\text{wavy line}})$$

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

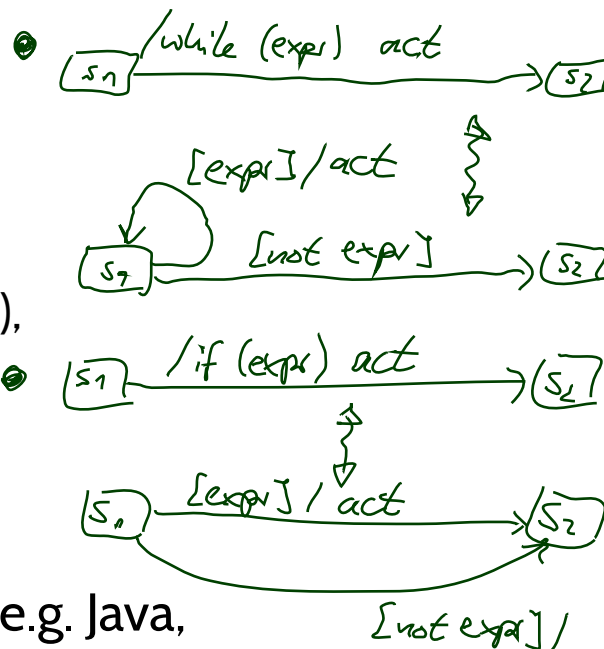
- empty statements, skips,
- assignments,
- conditionals (by normalisation and auxiliary variables),
- create/destroy (later),

but not **possibly diverging loops**.

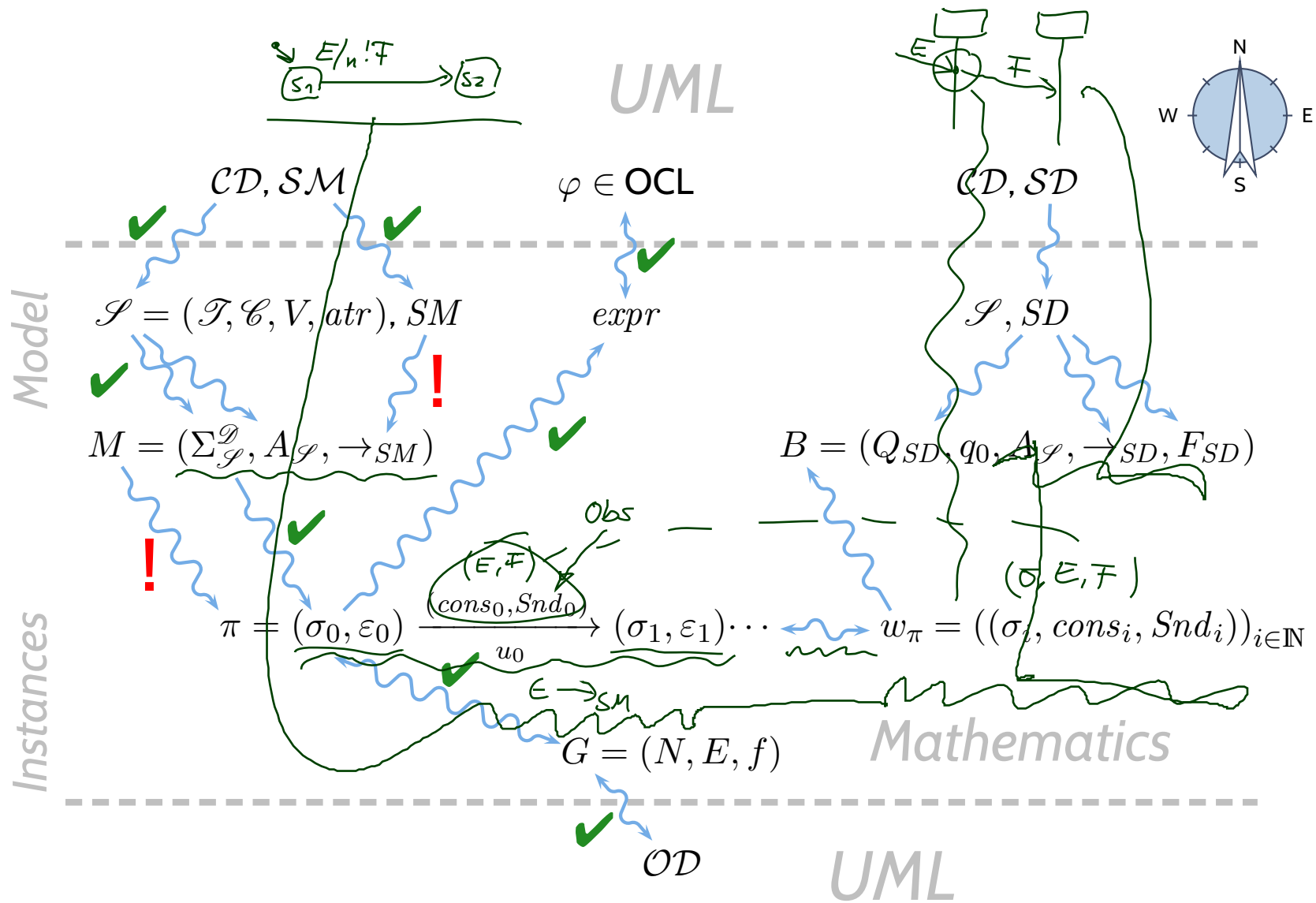
Our (Simple) Approach: if the action language is, e.g. Java, then (**syntactically**) forbid loops and calls of recursive functions.

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

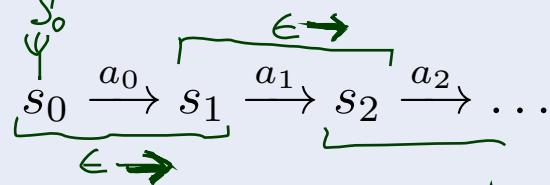
Transition Relation, Computation

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

$$\rightarrow \subseteq S \times A \times S$$

a (labelled) **transition relation**.

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



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

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

trans. relation
states *labels* *init. states*

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

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

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

error state
consumed sig. inst.
observation what has been sent out (plus create/destroy) by actions

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

who (which object) did do the transition

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

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 \mathcal{C} , i.e. if

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

- u is stable and in state machine state s , i.e. $\sigma(u)(\text{stable}) = 1$ 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)

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

and

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

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

where $b = 0$ if and only if there is a transition **with trigger $'_'$** enabled for u in (σ', ε') .

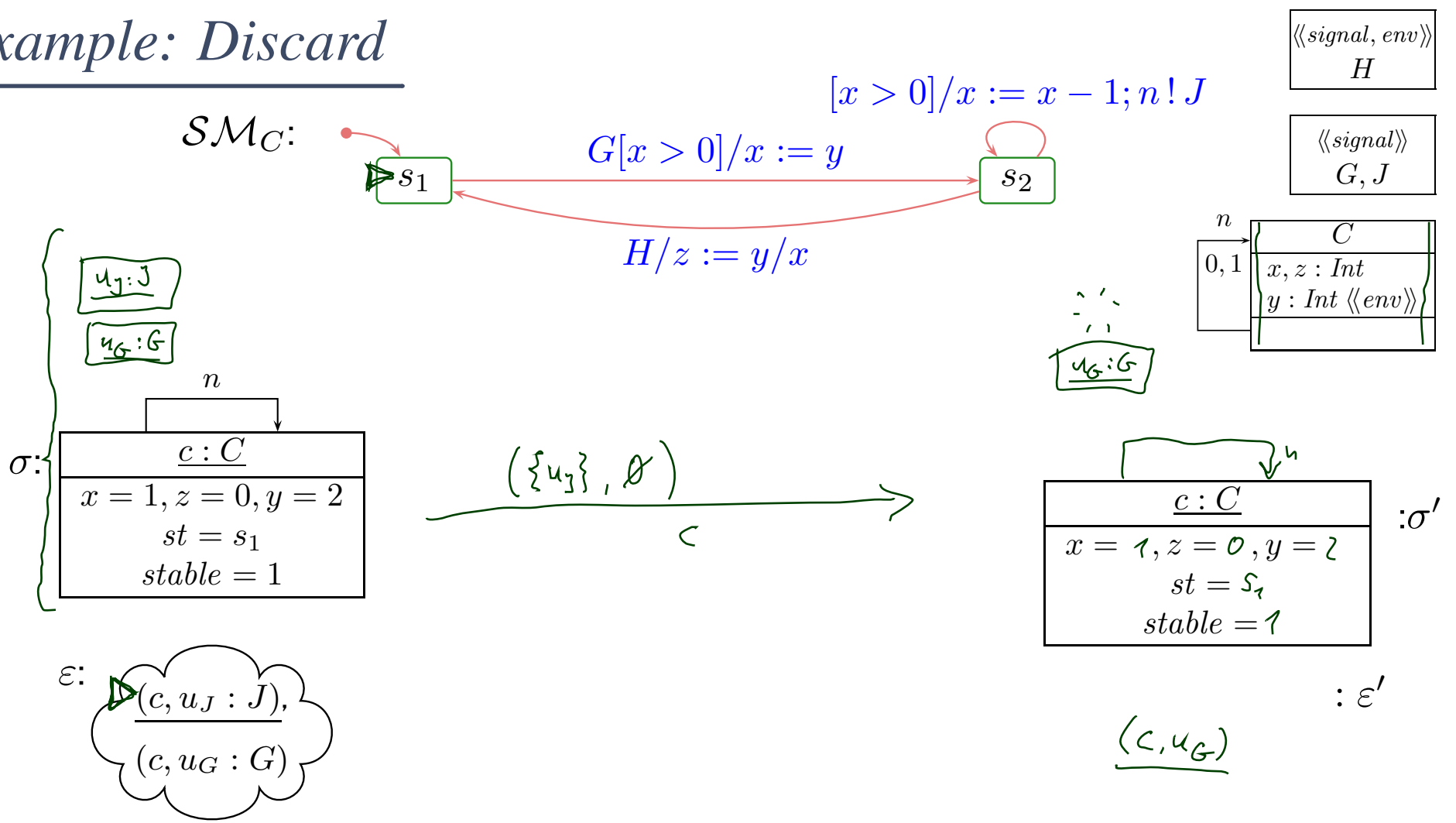
- the event u_E is removed from the ether, i.e.

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

- consumption of u_E is observed, i.e.

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

Example: Discard



- $u \in \text{dom}(\sigma) \cap \mathcal{D}(C)$
- $u_E \in \mathcal{D}(E), u_E \in \text{ready}(\epsilon, u)$
- $\forall (s, F, \text{expr}, \text{act}, s') \in \rightarrow (SM_C) : F \neq E \vee I[\text{expr}](\sigma, u) = 0$
- $\sigma(u)(\text{stable}) = 1, \sigma(u)(\text{st}) = s,$
- $\sigma' = \sigma[u.\text{stable} \mapsto _b] \setminus \{u_E \mapsto \sigma(u_E)\}$
- $\epsilon' = \epsilon \ominus u_E$
- $\text{cons} = \{u_E\}, \text{Snd} = \emptyset$

(ii) Dispatch

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

if

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

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

where $\tilde{\sigma} = \sigma[u.\text{params}_E \mapsto u_E]$.

e.g. $\boxed{S_1} \xrightarrow{E[\text{params}_E.x \geq 0] /} \boxed{S_2}$

and

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

$$(\sigma'', \varepsilon') \in t_{act}[u](\tilde{\sigma}, \varepsilon \ominus u_E),$$

$$\sigma' = (\sigma''[u.st \mapsto s', u.\text{stable} \mapsto b, u.\text{params}_E \mapsto \emptyset]) \upharpoonright_{\mathcal{D}(\mathcal{C}) \setminus \{u_E\}}$$

remove u_E

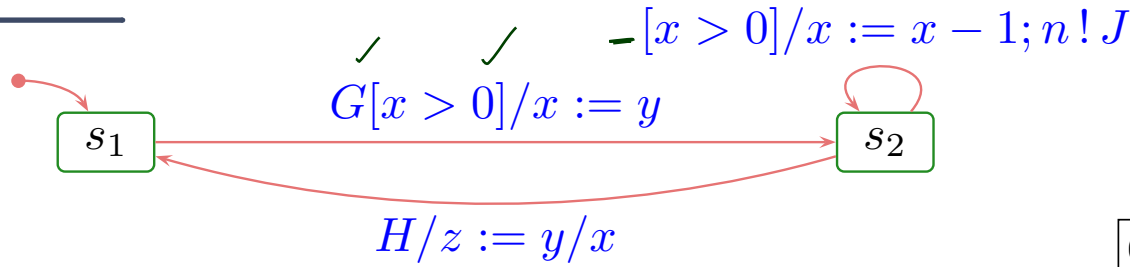
where b **depends** (see (i))

- Consumption of u_E and the side effects of the action are observed, i.e.

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

Example: Dispatch

SMC:



$\langle\langle \text{signal}, \text{env} \rangle\rangle$
 H

$\langle\langle \text{signal} \rangle\rangle$
 G, J

n
 C
 $0, 1$
 $x, z : \text{Int}$
 $y : \text{Int} \langle\langle \text{env} \rangle\rangle$

$u_G : G$

$\sigma:$
 n
 $c : C$
 $x = 1, z = 0, y = 2$
 $st = s_1$
 $stable = 1$

$(\{u_G\}, \emptyset)$
 c

n
 $c : C$
 $x = 2, z = 0, y = 2$
 $st = s_2$
 $stable = 0$
 $:\sigma'$

$\varepsilon:$
 $\langle (c, u_G : G) \rangle$

$:\varepsilon'$

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

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

(iii) Continue Run-to-Completion

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

if

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

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

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

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

and

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

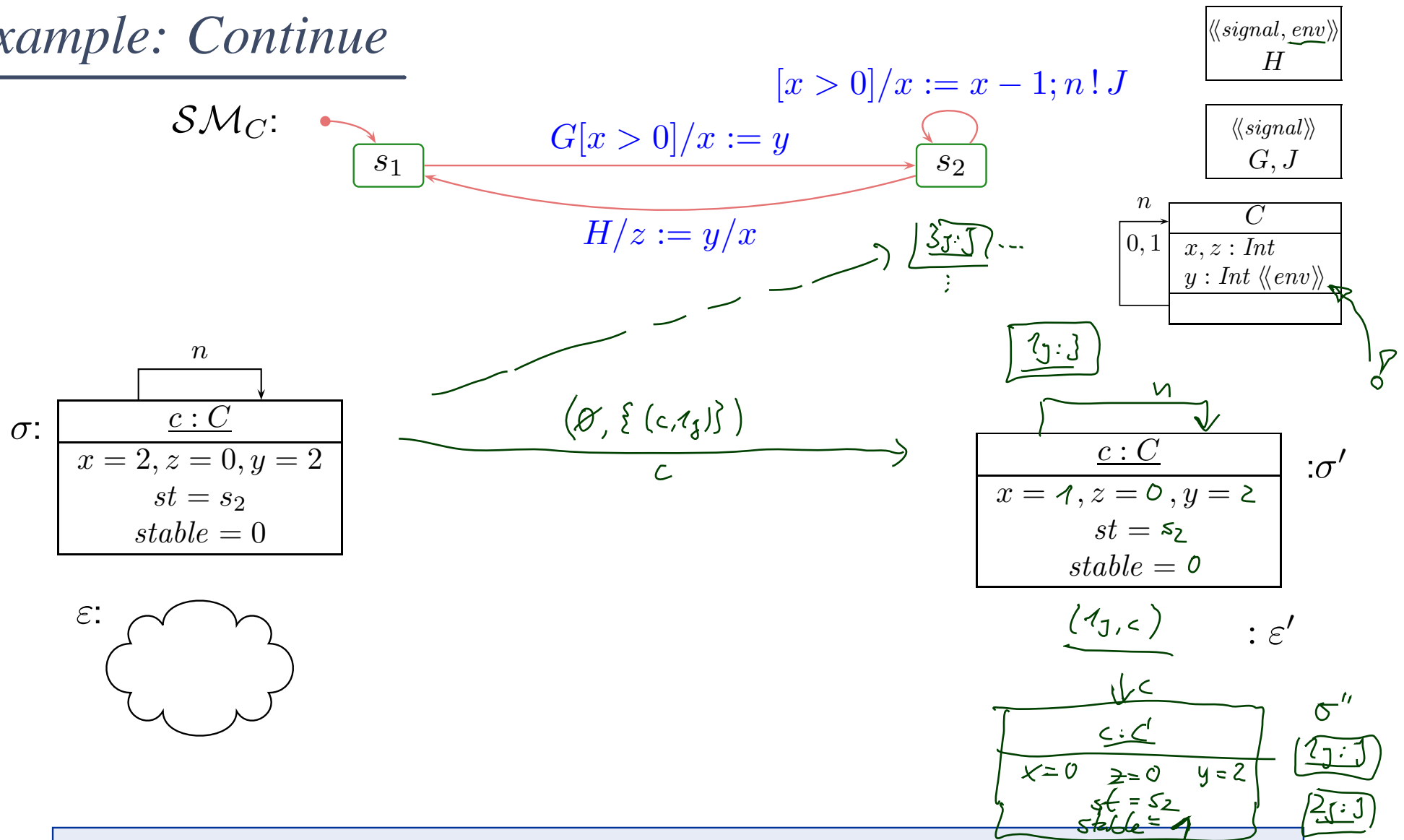
$$(\sigma'', \varepsilon') \in t_{act}[u](\sigma, \varepsilon), \quad \sigma' = \sigma''[u.st \mapsto s', u.stable \mapsto b]$$

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



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

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

(iv) Environment Interaction

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

Then

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

dedicated label

if either (!)

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

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

where $u_E \notin \text{dom}(\sigma)$ and $\text{atr}(E) = \{v_1, \dots, v_n\}$.

- Sending of the event is observed, i.e. $cons = \emptyset, Snd = \{u_E, \}$.

or

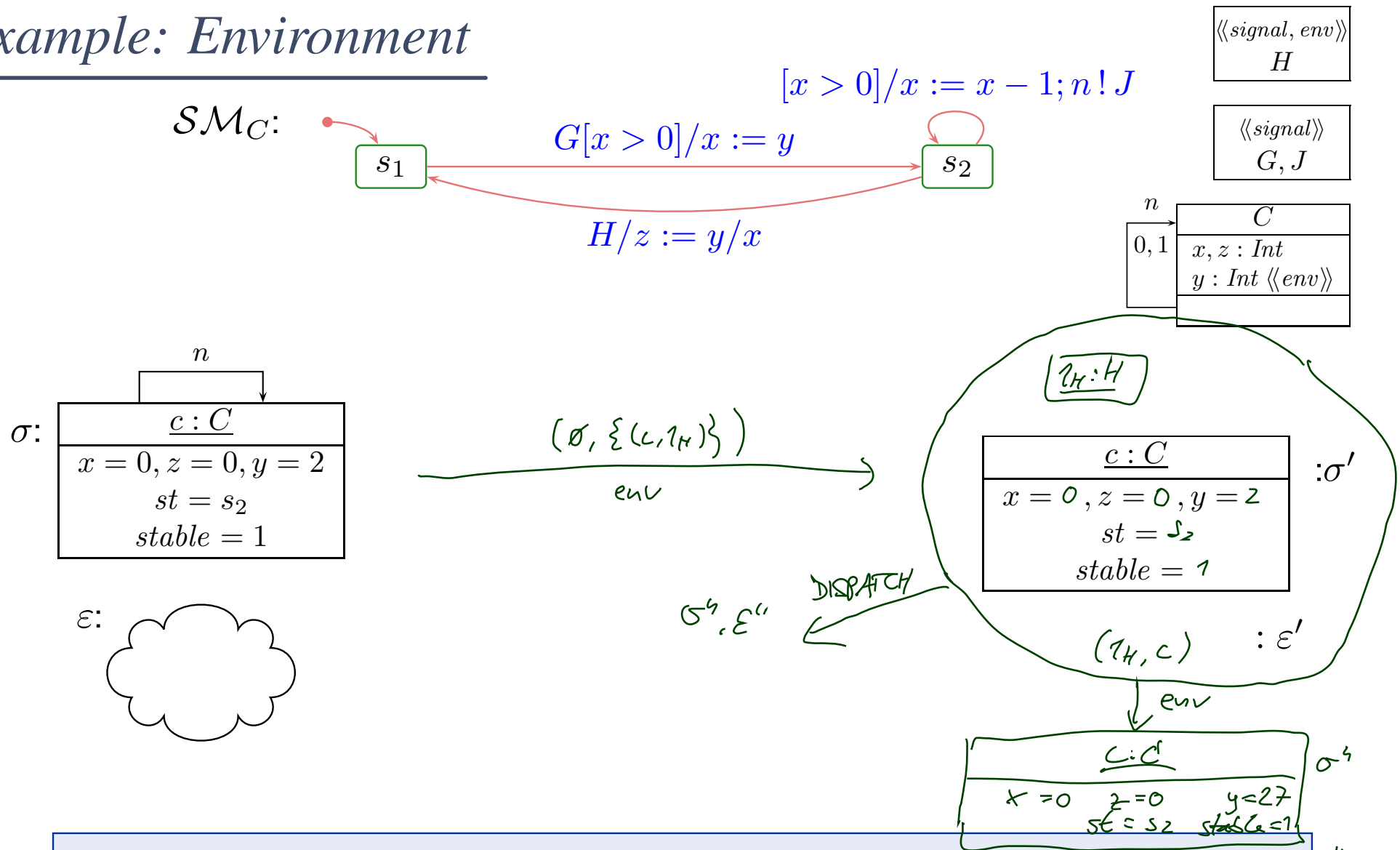
- Values of input attributes change freely in alive objects, i.e.

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

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

- $\varepsilon' = \varepsilon$.

Example: Environment



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

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

(v) Error Conditions

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

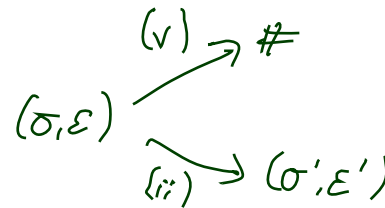
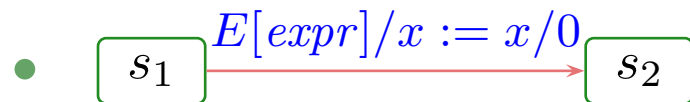
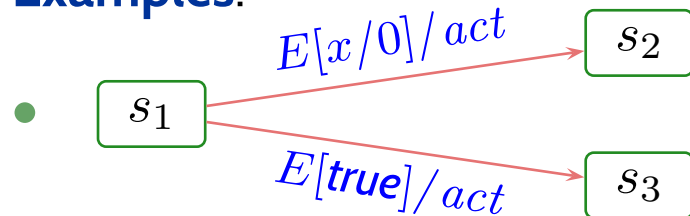
if, in (i), (ii), or (iii),

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

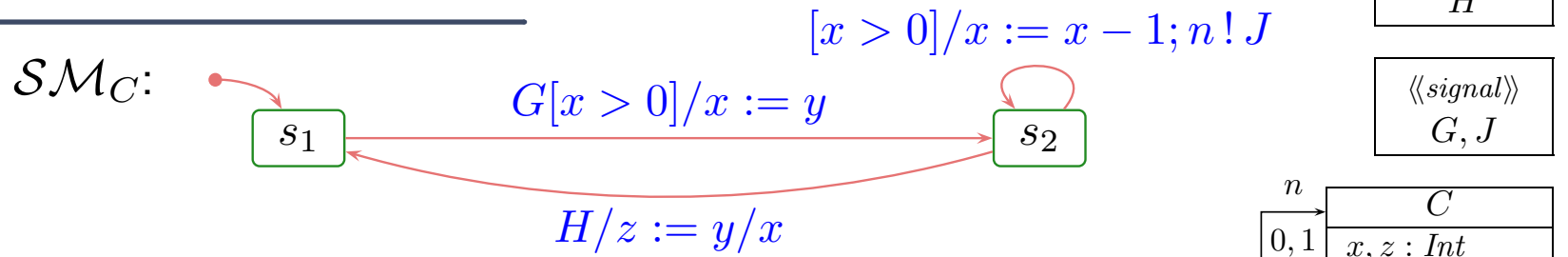
and

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

Examples:



Example: Error Condition



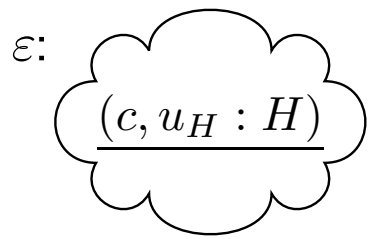
$\langle\langle signal, env \rangle\rangle$ H
$\langle\langle signal \rangle\rangle$ G, J
n
C
$0, 1$
$x, z : Int$ $y : Int \langle\langle env \rangle\rangle$

σ :

n
$c : C$
$x = 0, z = 0, y = 27$ $st = s_2$ $stable = 1$

$c : C$
$x = \quad, z = \quad, y =$ $st =$ $stable =$

$:\sigma'$



$:\varepsilon'$

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

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.