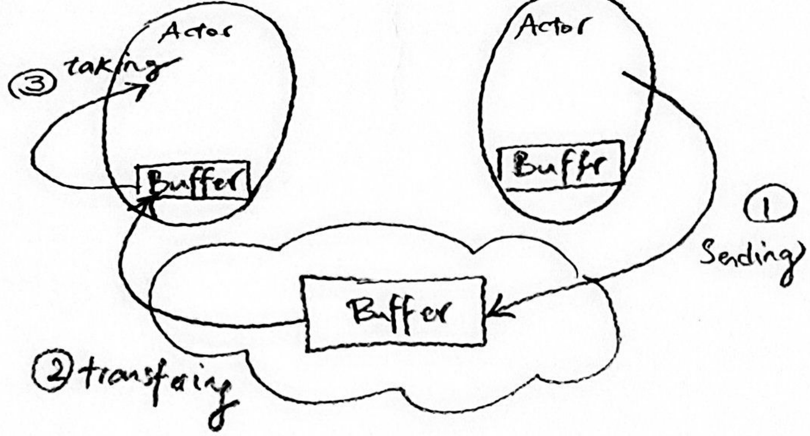
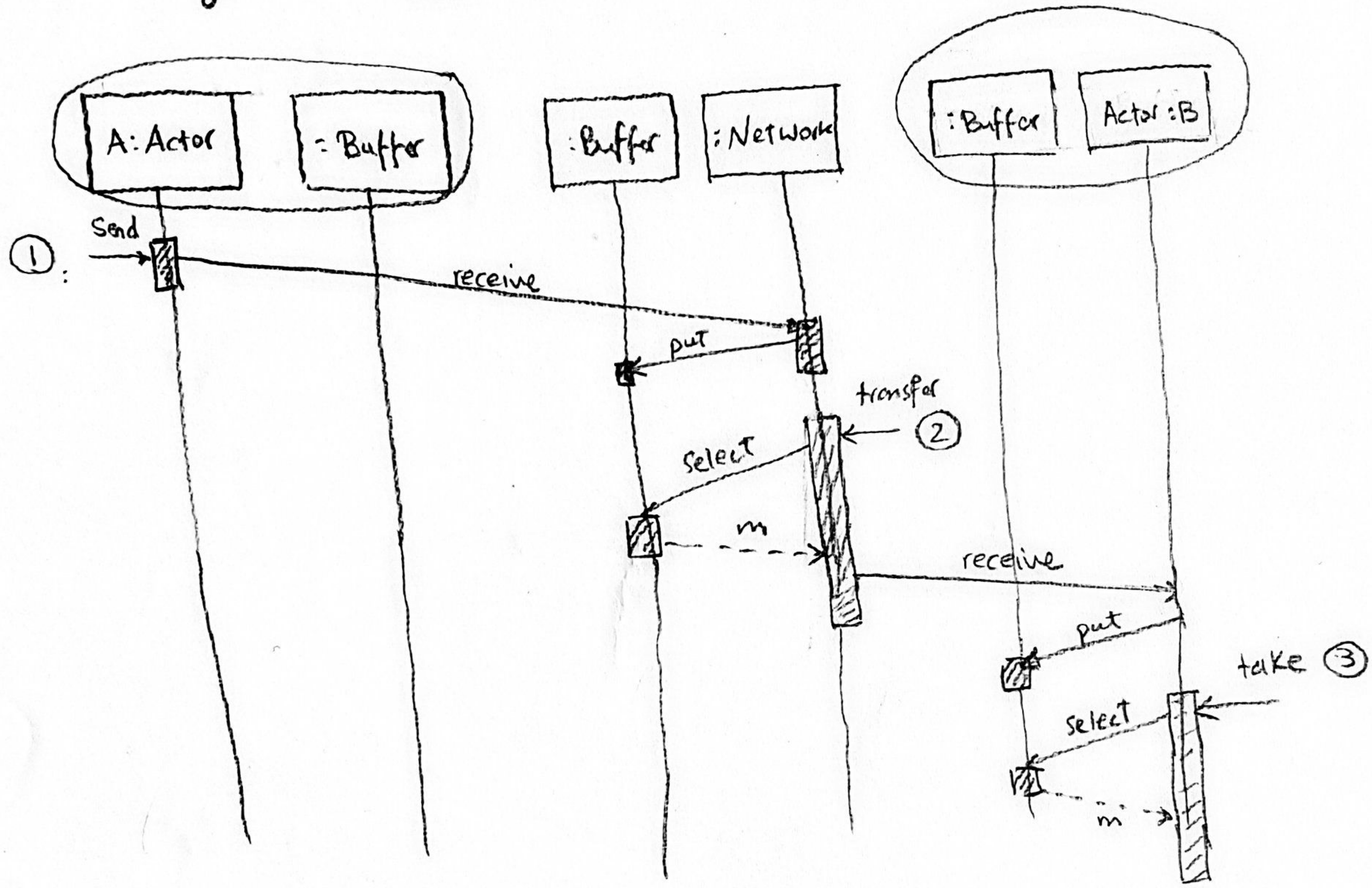
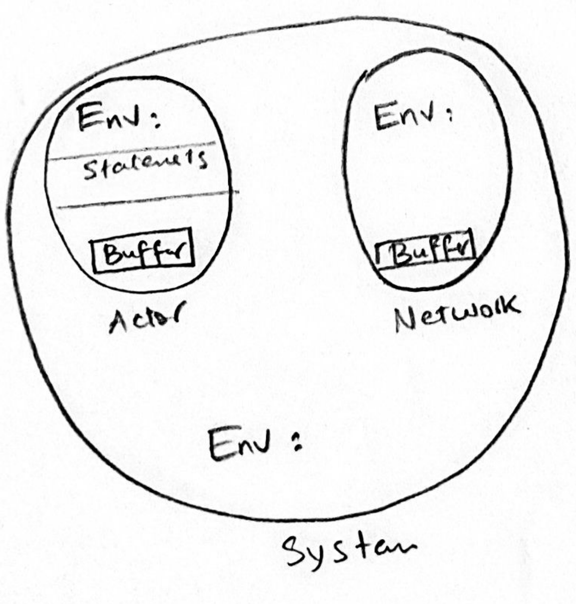


Actor
Actor
③ taking
Buffer
Buffer
① Sending
Buffer
② transfering



A: Actor
:Buffer
:Buffer
:Network
:Buffer
Actor :B
Send
①
receive
put
transfer
②
select
m
receive
put
take ③
select
m



Env:
Statements
Buffer
Actor
Env:
Buffer
Network
Env:
System

Buffers implements policies for put / select

put : What if the buffer is full ?

Select : base on priority / times / ...

user should provide the SOS rule for put / select regarding the structure of buffer

put : $$\frac{?}{b \xrightarrow{m?} b'}$$

Select : $$\frac{?}{b \xrightarrow{m!} b'}$$

b : Buffer

m : Msg

Actor-level SOS rules

e is empty in classic Rebeca

receive : $$\frac{b \xrightarrow{m?} b'}{e, b, \sigma \xrightarrow{m?} e, b', \sigma}$$

actor is input-enabled if its buffer accepts

internal progress : $$\frac{e, \sigma \xrightarrow{\alpha} e', \sigma' \quad \overset{?}{\boxed{\;\;}}}{e, b, \sigma \xrightarrow{\alpha} e', b, \sigma'}$$

$\alpha = \{\tau, m!\}$

a condition indicating when an actor can internally progress

take : $$\frac{b \xrightarrow{m!} b' \quad \overset{?}{\boxed{\;\;}}}{e, b, \epsilon \xrightarrow{\tau} e, b', body(m)}$$

Statements-level

Send : $$e, snd(x, m, y) \xrightarrow{m!} e, T$$

Seq : $$\frac{e, \sigma_1 \xrightarrow{\alpha} e, T}{e, \sigma_1 ; \sigma_2 \xrightarrow{\alpha} e, \sigma_2}$$

network-level

recieve : $\dfrac{b \xrightarrow{m?} b'}{e, b \xrightarrow{m?} e, b'}$: network is input-enabled

transfer : $\dfrac{b \xrightarrow{m!} b'}{e, b \xrightarrow{m!} e, b'}$

user can put condition on messages

e is empty in classic Rebeca

System-level

$\beta = \{\tau, m?\}$

actor-progress $\dfrac{s(x) = (e^*, b, \sigma) \quad e, b, \sigma \xrightarrow{\beta} e', b', \sigma'}{e, s, ns \xrightarrow{\beta} s[x \mapsto (e b', \sigma')], ns}$

$\cup e^*$

$e' \setminus e$

network ← local state

$ns = (e^{**}, b)$

Communication I: Actor-network

$\dfrac{s(x) = (e^*, b^*, \sigma) \quad (e, b^*, \sigma \xrightarrow{m!} e', b^{*\prime}, \sigma' \quad ns \xrightarrow{m?} ns'}{e, s, ns \longrightarrow s[x \mapsto (e^{*\prime}, b^{*\prime}, \sigma')], ns'}$

$\cup e^*)$

$e^{**} \cup e, b \xrightarrow{m?} e^{**\prime}$

$\{e\},$

$e' \setminus e$

Communication II: network-Actor

$\dfrac{s(y) = (e^*, b^*, \sigma) \quad e, b^*, \sigma \xrightarrow{m?} e, b^{*\prime}, \sigma' \quad ns \xrightarrow{m!} ns'}{e, s, ns \longrightarrow s[x \mapsto (e', b^{*\prime}, \sigma')], ns'}$

$* \cup e$

environment-progress

$\dfrac{e \longrightarrow e' \quad \boxed{?}}{e, b, s \longrightarrow ?}$

user should define how the enviroment progresses?

① Sending

$$\dfrac{\dfrac{e^*, snd(x,m,y) \xrightarrow{m!} e, \top}{e^*, snd(x,m,y);\sigma \xrightarrow{m!} e^*, \sigma} : \text{send}}{e^*, snd(x,m,y);\sigma \xrightarrow{m!} e^*, b, \sigma} : \text{internal progress}$$

$$\dfrac{e^*, snd(x,m,y);\sigma \xrightarrow{m!} e^*, b, \sigma \qquad s(x) = (\overset{e^*}{b^*}, snd(x,m,y);\sigma) \qquad \dfrac{b \xrightarrow{m?} b'}{e, b \xrightarrow{m?} e, b'} : \text{recieve}}{s, \underbrace{(b, e)}_{ns} \longrightarrow s[x \mapsto (b^*, \sigma)], \underbrace{(b', e)}_{ns'}} : \text{Communication I}$$

② transfering

$$\dfrac{\dfrac{b^* \xrightarrow{m?} b^{*\prime}}{e^*, b^*, \sigma \xrightarrow{m?} e^*, b^{*\prime}, \sigma} : \text{receive}}{e^*, b^*, \sigma \xrightarrow{m?} e^*, b^{*\prime}, \sigma} : \text{internal progrers}$$

$$\dfrac{e^*, b^*, \sigma \xrightarrow{m?} e^*, b^{*\prime}, \sigma \qquad s(y) = (e^*, b^*, \sigma) \qquad \dfrac{b \xrightarrow{m!} b'}{e, b \xrightarrow{m!} e, b'} : \text{transfer}}{s, \underbrace{(b, e)}_{ns} \longrightarrow s[y \mapsto (b^{*\prime}, \sigma)], \underbrace{(b', e)}_{ns'}} : \text{Communication II}$$

③ taking

$$\dfrac{s(x) = (e^*, b^*, \varepsilon) \qquad \dfrac{b^* \xrightarrow{m!} b^{*\prime}}{e^*, b^*, \varepsilon \xrightarrow{\tau} e^*, b^{*\prime}, body(m)} : \text{take}}{s, ns \xrightarrow{\tau} s[x \mapsto (b^{*\prime}, body(m))], ns} : \text{actor-progress}$$

Buffer type for Timed Rebecs

ActorBuffer <: Buffer

ActorBuffer : Bag(Msg)

$\oplus, \ominus$: $Bag \times Msg \to Bag$

$\in$: $Msg \times Bag \to Bool$

put: $\dfrac{}{b \xrightarrow{m?} b \oplus m}$ select: $\dfrac{m \in b \quad \forall m' \in b\,(m.ar \leq m'.ar)}{b \xrightarrow{m!} b \ominus m}$

Actor-level SOS

$\underline{e}$: $ActorEnv = (\underset{\text{local time}}{\overset{now}{\mathbb{N}}} \times \overset{rt}{\mathbb{N}})$ ↳ resume time ← we can also include variable valuations in case of having variables

time progress I $\dfrac{e.now < e.rt \qquad e.now < t \leq e.rt}{e, b, \sigma \xrightarrow{t} e[now \mapsto t], b, \sigma}$

time progress II $\dfrac{m \in b \quad \forall m' \in b\,(m.ar \leq m'.ar) \quad e.now < t < m.ar}{e, b, \varepsilon \xrightarrow{t} e[now \mapsto t], b, \varepsilon}$

internal progress $\dfrac{e, \sigma \xrightarrow{\alpha} e', \sigma' \qquad e.rt = \perp}{e, b, \sigma \xrightarrow{\alpha} e', b, \sigma'}$ → it can internally progress when it is not resumed

resuming actor $\dfrac{e.rt = e.now}{e, b, \sigma \xrightarrow{I} e[rt \mapsto \perp], b, \sigma}$

We can also resume actor with time progress

Statement-level

$$\dfrac{t = e.now}{e, delay(d) \xrightarrow{\tau} e[\underset{rt}{\cancel{now}} \mapsto t+d], T}$$

network-level

networkBuffer :< Buffer

networkBuffer = ID ⟶ ActorBuffer

NetworkEnv : $\overset{now}{\mathbb{N}}$ indicating local time of network

put : $$\frac{}{b \xrightarrow{m?} b \oplus m}$$

select : $$\frac{\exists x \in ID\,(\exists m \in b(x) \wedge \forall m' \in b(x) \cdot (m.ar \leq m'.ar))}{b \xrightarrow{m!} b[x \mapsto b(x) \ominus m]}$$

transfer : $$\frac{b \xrightarrow{m!} b' \quad m.ar = e.now}{e, b \xrightarrow{m!} e, b'}$$

time progress : $$\frac{\exists x \in ID(\exists m \in b(x) \cdot (\forall x' \in ID(\forall m' \in b(x')(m.ar \leq m'.ar))) \wedge e.now < m.ar \wedge t \in (e(now), m.ar]))}{e, b \xrightarrow{t} e[now \mapsto t], b}$$

System-level

SystemEnv : $\overset{now}{\mathbb{N}}$ indicating the global time of system

$s(x) = (e_x, b.)$

time progress

$$\frac{\exists t\ (t > e.now \wedge \forall x \in ID(s(x) \xrightarrow{t} e', b', \sigma') \wedge ns \xrightarrow{t} ns' \wedge \not\exists t' > t\ (\forall x \in ID(s(x) \xrightarrow{t'} \ldots \wedge ns \xrightarrow{t'} ns'')\ )}{e, s, ns \xrightarrow{t} e[now \mapsto t], update(s,t), ns'}$$

where $update(s,t)(x) = e[now \mapsto t], b, \sigma$

and $s(x) = e, b, \sigma$

Hybrid Rebeca

Only we should define the statements and physical actors.

other parts are intact

physical-Actor-level SOS

$$e : \text{PActorEnv} = (\underbrace{\mathbb{N}}_{now} \times \underbrace{(Var \to Value)}_{val} \times \underbrace{Mode}_{mode})$$

$$\text{Mode} = \underbrace{Guard}_{guard} \times \underbrace{Flow}_{flow} \times \underbrace{Inv}_{inv} \times \underbrace{Triggers}_{trigs}$$

time progress

$$\frac{f \in e.mode.flow \quad f(0) = e.val \quad f(t) = v' \quad t \geq 0 \qquad \forall 0 \leq t' \leq t \quad f(t') \models e.mode.inv}{e, b, \varepsilon \xrightarrow{t} e[val \mapsto v'], b, \varepsilon}$$

time only can progress when it has no statement to execute

end of mode

$$\frac{e.v \models e.mode.guard \quad \sigma = e.mode.trigs}{e, b, \varepsilon \to e, b, \sigma}$$

Statement-level

$$e, setmode(m) \xrightarrow{\tau} e[mode \mapsto m], T$$

network-level

NetworkBuffer : ID $\longrightarrow$ Actor Buffer ; Similar to Timed Setting

Env

$$\text{NetworkEnv} : \underbrace{\text{IN}}_{\text{now}} \times \underbrace{\text{CANConfig}}_{\text{CAN}} \qquad : \text{Name} \longrightarrow \text{Value}$$

DELAY : ID $\times$ MNAME $\times$ ID $\longrightarrow$ IN   delay between two end points regarding message name

PRIORITY : $\underline{\text{MNAME}}$ $\longrightarrow$ IN   assigns priority to message name

$$\text{CANConfig} = \underbrace{\text{DELAY}}_{\text{NetDelay}} \times \underbrace{\text{PRIORITY}}_{\text{priority}}$$

$$\text{transfer} : \frac{\begin{array}{l} \nexists m' \,(b \xrightarrow{m'} \;\wedge\; e.\text{CAN}.\text{priority}(m'.\text{name}) > e.\text{CAN}.\text{priority}(m.\text{name}) \;\wedge \\ \qquad e.\text{now} = m.\text{ar} + e.\text{CAN}.\text{NetDelay}(m')) \;\wedge \\ b \xrightarrow{m!} b' \quad e.\text{CAN}.\text{NetDelay}(m.\text{sender}, m.\text{mname}, m.\text{reciever}) = e.\text{now} + m.\text{ar} \end{array}}{e, b \xrightarrow{m!} e, b'}$$

This transfer the message with the highest priority and delaying the message

$$\text{ID} \times \text{ID} \times \text{IN}$$

$$\text{CAN} \longmapsto \underline{\text{MName}} \longrightarrow \text{IN}$$

$$\underbrace{\text{CAN}}_{\text{IN} \text{ Name}}$$