Formal Semantics of CCS

Moonzoo Kim

CS Dept. KAIST

Review of the Previous Class

- Sequential system v.s. Reactive system
  - Ex1. Mathematical functions with given inputs generate outputs
    - Usually no environment consideration and timing consideration.
  - Ex2. Ad-hoc On-Demand Vector routing protocol
    - Should model multiple concurrent nodes (environment)
    - Should model communication among the nodes
    - Should model timely behavior (e.g. time-out, etc)
- Modeling of a complex system
  - Concurrency => interleaving semantics
  - Communication => synchronization
  - Hierarchy => refinement

Process Algebra

- A process algebra consists of
  - a set of operators and syntactic rules for constructing processes
  - a semantic mapping which assigns meaning or interpretation to every process
  - a notion of equivalence or partial order between processes
- Advantages: A large system can be broken into simpler subsystems and then proved correct in a modular fashion. Also, correctness can be checked
  - A hiding or restriction operator allows one to abstract away unnecessary details.
  - Equality for the process algebra is also a congruence relation; and thus, allows the substitution of one component with another equal component in large systems.

Notations (1/2)

- A system is described as a set of communicating processes
  - Each process executes a sequence of actions
  - Actions represents either inputs/outputs or internal computation steps

- A set of actions/events *Act = L U L' U {τ}*
  - *L* ={a,b,…} is a set of *names* and *L'* ={a',b',…} is a set of *co-names*
    - a∈ *L* can be considered as the act of receiving a signal
    - a'∈ *L'* can be considered as the act of emitting a signal
    - *τ* is a special action to represent internal hidden action
  - *Act* – {*τ* } represents the set of externally visible actions:

Notations (2/2)

- Operational (transitional) semantics of CCS process
  - Define the “execution steps” that processes may engaged in
  - P –a-> P’ holds if a process P is capable of engaging in action a and then behaving like P’
  - Define –a-> inductively using inference rules for operators
    - premises
      -------------- *(side condition)*
      conclusion

Example 1:

$$\text{Choice}_R \quad \frac{Q \text{ -}\alpha\text{-> } Q'}{P+Q \text{ -}\alpha\text{-> } Q'}$$

Example 2:

$$\text{Prefix} \quad \frac{}{\alpha.P \text{ –}\alpha\text{-> } P}$$

Operators for Sequential Process

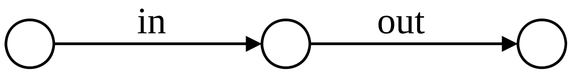
The idea: 7 elementary ways of producing or putting together labelled transition systems

1.Nil 0 No transitions (deadlock)

Prefix Prefix

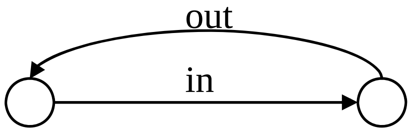
2.Prefix $\alpha.P$ ($\alpha \in Act$) in.out.0 –*in*-> out.0 –*out*-> 0

$$\text{Prefix}\ \frac{\text{(empty)}}{\alpha.P \xrightarrow{\alpha} P}$$



3.Defn $A = P$ Buffer = in.out.Buffer

Buffer-*in*->out.Buffer-*out*->Buffer



Operators for Sequential Process (cont.)

4.Choice *P* + *Q*

P -α->P'

Choice$_L$ ----------------

P+Q -α-> P'

Q -α-> Q'

Choice$_R$ ---------------

P+Q -α-> Q'

BadBuf = in.(τ.0 + out.BadBuf)

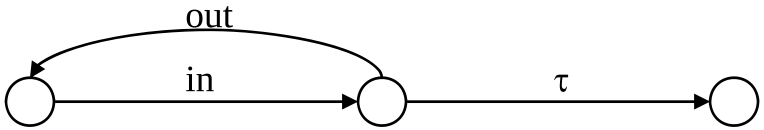
Prefix

BadBuf –*in*-> τ.0 + out.BadBuf

Choice$_L$

Choice$_R$

-τ-> 0 **or** –*out*-> BadBuf



Obs: No priorities between τ's, a's or a's !

May use Σ notation to comactly represent sequential process

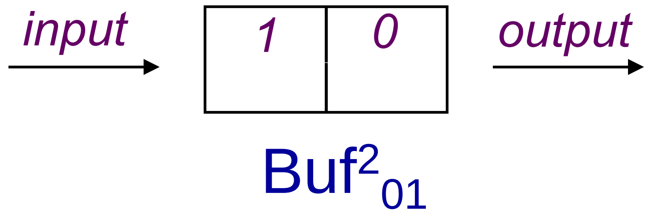
$$P = \sum_{i \in I} \alpha_i . P_i$$

Example: Boolean Buffer of Size 2

Action and Process Def.

in_0 :0 is coming as input
in_1 :1 is coming as input
out_0 :0 is going out as output
out_1 :1 is going out as output

Buf^2 : Empty 2-place buffer
Buf^2_0 : 2-place buffer holding 0
Buf^2_{01}: 2-place buffer holding 0 at head and 1 at tail



$$Buf^2 = in_0.Buf^2_0 + in_1.Buf^2_1$$

$$Buf^2_0 = out_0.Buf^2 + in_0.Buf^2_{00} + in_1.Buf^2_{01}$$

$$Buf^2_1 = out_1.Buf^2 + in_0.Buf^2_{10} + in_1.Buf^2_{11}$$

$$Buf^2_{00} = out_0.Buf^2_0$$

$$Buf^2_{01} = out_0.Buf^2_1$$

$$Buf^2_{10} = out_1.Buf^2_0$$

$$Buf^2_{11} = out_1.Buf^2_1$$

Operators for Concurrent Process

5. Composition

$$\text{Par}_L \quad \frac{P \text{ -}\alpha\text{->} P'}{P|Q \text{ -}\alpha\text{-> } P'|Q}$$

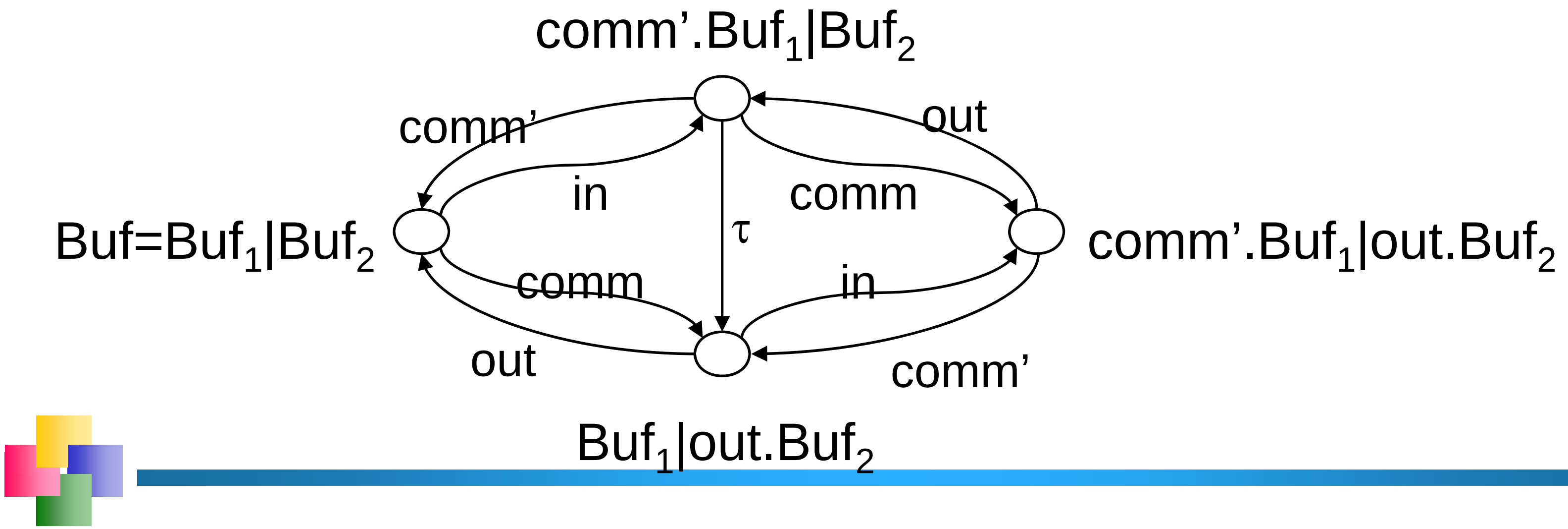
$$\text{Par}_R \quad \frac{Q \text{ -}\alpha\text{-> } Q'}{P|Q \text{ -}\alpha\text{-> } P|Q'}$$

$$\text{Par}\,\tau \quad \frac{P\text{-}a\text{->}P',\ Q\text{–}a'\text{->}Q'}{P|Q \text{ -}\tau\text{-> } P'|Q'}$$

Buf_1 = in.comm'.Buf_1
Buf_2 = comm.out.Buf_2
Buf = Buf_1 | Buf_2

Buf
Par_L -*in*-> comm'.Buf_1 | Buf_2
Par_τ -τ> Buf_1 | out.Buf_2
Par_R -*out*-> Buf_1 | Buf_2

Buf
Par_R -*comm*-> Buf_1 | out.Buf_2
Par_R -*out*-> Buf_1 | Buf_2

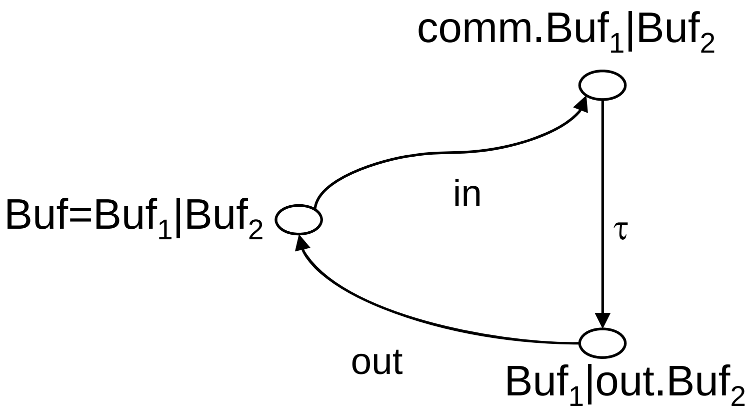


Operators for Concurrent Process (cont.)

6. Restriction *P\L*

$$\text{Res}\quad \frac{P \text{ -}\alpha\text{->} P'}{P\backslash L \text{ -}\alpha\text{-> } P'\backslash L}\quad \alpha \notin L \cup L'$$

Buf_1 = in.comm.Buf_1
Buf_2 = comm'.out.Buf_2
Buf=(Buf_1 | Buf_2)\{comm}



Buf
-*in*-> (comm.Buf_1 | Buf_2)\{comm}
-τ-> (Buf_1 | out.Buf_2)\{comm}
-*out*-> (Buf_1 | Buf_2)\{comm}

Buf
~~-*comm'*-> Buf_1 | out.Buf_2~~

(Buf1 | Buf2)\{comm} : a design for buffer with separated input/output ports
ReqBuf = in.out.ReqBuf : a requirement for buffer design
(Buf1 | Buf2)\{comm} == ReqBuf means that buffer design satisfies the requirement

Operators for Concurrent Process (cont.)

7. Relabelling ***P*[*f*]**

$$\text{Rel}\quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\begin{aligned}
Buf &= in.out.Buf \\
Buf_1 &= Buf[comm/out] \\
&= in.comm.Buf_1 \\
Buf_2 &= Buf[comm'/in] \\
&= comm'.out.Buf_2
\end{aligned}$$

Relabelling function *f* must preserve complements:

$f(a') = f(a)'$

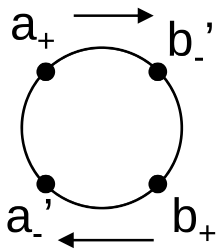
Relabelling function often given by name substitution as above

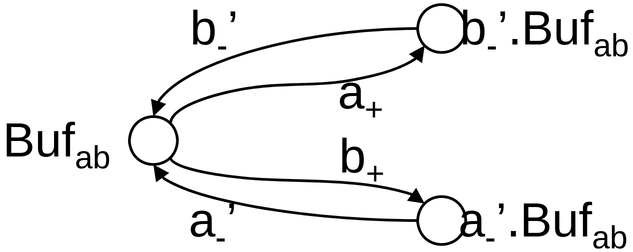
Example: 2-way Buffers

1-place 2-way buffer:

$Buf_{ab} = a_+.b_-'.Buf_{ab} + b_+.a_-'.Buf_{ab}$

LTS:



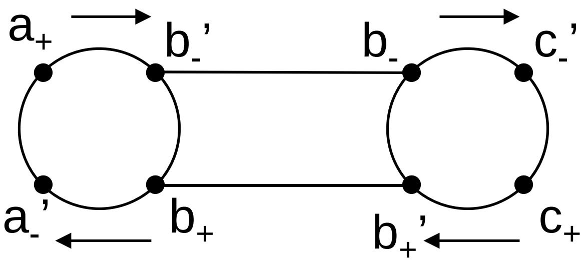


$Buf_{bc} =$

$Buf_{ab}[c_+/b_+,c_-/b_-,b_-/a_+,b_+/a_-]$

(Obs:simultaneous substitution!)

$Sys = (Buf_{ab} \mid Buf_{bc})\backslash\{b_+,b_-\}$



But what's wrong? Deadlock occurs

In other words, $Sys == Buf_{ac}$?

Summary of CCS Semantics

$$\mathit{Act}\quad \frac{}{\alpha.P \text{ –}\alpha\text{-> } P}$$

in.P -in-> P

$$\mathit{Choice}_L\quad \frac{P \text{ -}\alpha\text{->} P'}{P+Q \text{ -}\alpha\text{-> } P'},\qquad \mathit{Choice}_R\quad \frac{Q \text{ -}\alpha\text{->} Q'}{P+Q \text{ -}\alpha\text{-> } Q'}$$

in.P + out.Q -in-> P *or* –out-> Q

$$\mathit{Par}_L\quad \frac{P \text{ -}\alpha\text{->} P'}{P|Q \text{ -}\alpha\text{-> } P'|Q}\qquad \mathit{Par}_R\quad \frac{Q \text{ -}\alpha\text{->} Q'}{P|Q \text{ -}\alpha\text{-> } P|Q'}$$

in.P|in’.Q -in->P|in’.Q *or* –in’-> in.P|Q

$$\mathit{Par}\tau\quad \frac{P\text{-}a\text{->}P',\ Q\text{–}a'\text{->}Q'}{P|Q \text{ -}\tau\text{-> } P'|Q'}$$

in.P | in’.Q -τ-> P|Q

$$\text{Res}\quad \frac{P \text{ -}\alpha\text{->} P'}{P\backslash L \text{ -}\alpha\text{-> } P'\backslash L}\ \alpha \notin L \cup L'$$

(in.P | in’.Q)\{in} -τ-> (P|Q)\{in} only

$$\text{Rel}\quad \frac{P \text{ -}\alpha\text{->} P'}{P[f] \text{ –}f(\alpha)\text{-> } P'[f]}$$

in.P [out/in] -out-> P[out/in]