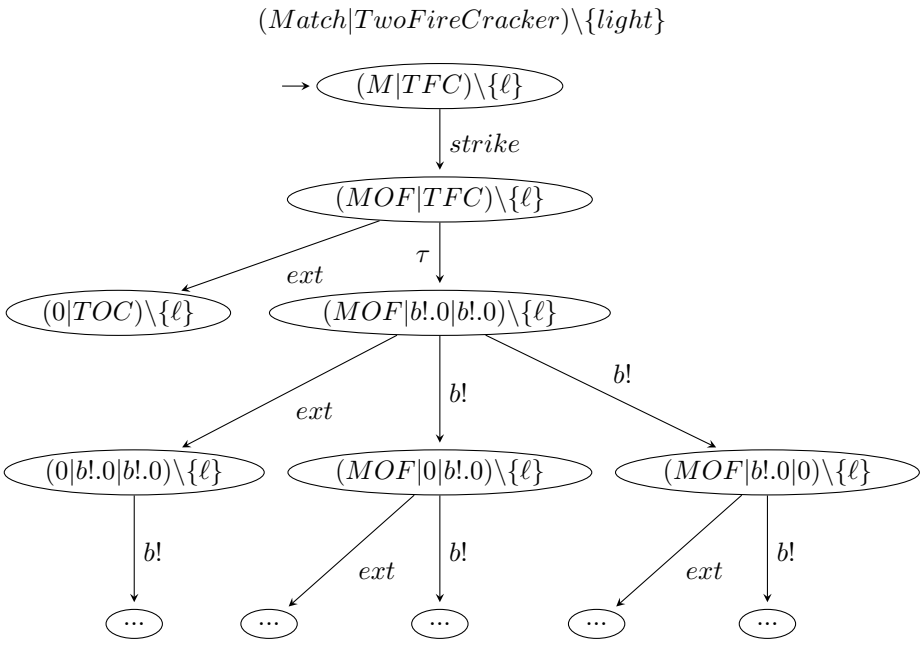
# Aula 6

### In PseuCo

$$
\begin{aligned}
Match &:= strike.MatchOnFire \\
MatchOnFire &:= light!.MatchOnFire + extinguish!.0 \\
TwoFireCracker &:= light?.(bang!.0|bang!.0)
\end{aligned}
$$




### CCS (quase) completo

Seja $Com = A^! \cup A^?$ conjunto de ações de comunicação, $Act = Com \cup \{\tau\}$ um conjunto de ações, e $Var$ um conjunto de nomes (variáveis). As expressões do $CCS$ são

$$P \quad ::= \quad 0 \mid X \mid P + P \mid \alpha.P \mid P|P \mid P\backslash H$$

onde $\alpha \in Act$, $X \in Var$ e $H \subseteq Com$. Supomos um conjunto $\Gamma$ de equações $X := P$.

### Semântica do CCS

A semântica das expressões do $CCS$ é então

$$[\![\_]\!] : (Var \to CCS) \to CCS \to LTS_{CCS}$$

tal que

$$[\![P]\!]_\Gamma = (CCS, \longrightarrow_\Gamma, P)$$

com

$$LTS_{CCS} = \{(CCS, T, s) \mid T \subseteq CCS \times Act \times CCS, \wedge s \in CCS\}$$

onde $\longrightarrow_\Gamma$ a mais pequena relação que satisfaz as regras de inferência.

$$\text{Pref}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\text{ParE}\ \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

$$\text{EscD}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\text{EscE}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\text{Sync}\ \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{ParD}\ \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\text{Res}\ \frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin H}{P\backslash H \xrightarrow{\alpha} P'\backslash H}$$

$$\text{Rec}\ \frac{P \xrightarrow{\alpha} P' \qquad \Gamma(X) = P}{X \xrightarrow{\alpha} P'}$$

**Mais regras de prioridade**

- $P|Q|R$ é $(P|Q)|R$
- $\alpha.P|Q$ é $(\alpha.P)|Q$
- $P + R|Q$ é $(P + R)|Q$

**Expressividade do CCS**

- $CCS_0$ só pode produzir sistemas de transição finitos acíclicos
- $CCS_0^\omega$ com $\Gamma$ finito só pode produzir sistemas de transição estado-finitos (a menos de isomorfismo)
- $CCS$ pode produzir sistemas e transição infinitos e com ramificação infinita
- Ex: $X := a.0|X$ tem ramificação infinita (*Verifica!*)
- com mais uma operação - renomeação $P[f]$ tem a potência duma Máquina de Turing.

**Operador de renomeação**

- Já vimos uma máquina de café em CCS
- $CM := coin?.coffee!.CM$
- Mas agora se quisermos um chocolate?
- Seria igual apenas mudando a accão de envio.
- $ChM := coin?.choco!.ChM$
- e o mesmo para outros produtos.
- Então podemos ter
- $VM := coin?.item!.VM$
- e definir

$$\begin{array}{rcl} CM & := & VM[coffee/item] \\ ChM & := & VM[choc/item] \\ \ldots & & \end{array}$$

**Operador de renomeação**

Seja $f : Act \to Act$ uma função de renomeação tal que

$$\begin{array}{rcl} f(\tau) & := & \tau, \\ f(\overline{a}) & := & \overline{f(a)} \ \ \forall a \in Com. \end{array}$$

A função $f$ pode representar-se da forma $[b_1/a_1, \ldots b_n/a_n]$ se $f(a_i) = b_i$.

Então, sendo $P$ um processo $P[f]$ é também um processo em que cada ação $a$ é substituida por $f(a)$ (assim como os complementos).

$$\begin{aligned}(a.B + b.B)[a/b, b/a] &= (b.B + a.B)\\ (a.0 + \overline{a}.A)[a/b] &= (a.0 + \overline{a}.A)\end{aligned}$$

**Regra de inferência para $P[f]$**

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

**Exemplo 6.1.** *Seja $A := a.b.B$, calcular*

$$[\![(A|b.a.B) + (b.A)[a/b]]\!]_\Gamma$$

**Operadores Estáticos e Dinâmicos**

- *Operadores dinâmicos*: $\cdot$ e $+$
- Desaparecem após se efectuar uma transição

$$\text{Pref}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\text{EscD}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

- *Operadores estáticos*: $|$ e $\setminus$ (também $[f]$)
- Não desaparecem após ser efectuada uma transição

$$\text{Sync}\ \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{ParD}\ \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\text{Res}\ \frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin H}{P\setminus H \xrightarrow{\alpha} P'\setminus H}$$

### CCS Regular

Não é permitida recursão sobre operadores estáticos.

$$\begin{aligned} P &::= 0 \mid X \mid P+P \mid \alpha.P \mid R \\ R &::= 0 \mid R+R \mid \alpha.R \mid R|R \mid R\backslash H \end{aligned}$$

**Proposição 6.1.** *Se $\Gamma$ é uma função parcial cujo contradomínio só tem expressões regulares então o Reach($[\![P]\!]_\Gamma$) é estados finito para todo $P \in CCS$ (i.e. o LTS atingível é estados-finito).*

Este é o cálculo que se usa mais na prática.

### Buffer Paralelos

- Calcular $[\![Buffer|Buffer]\!]_\Gamma$. para

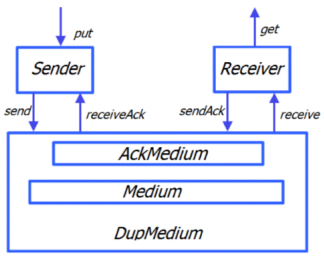
$$Buffer := put?.get?.Buffer$$

-

$$\begin{aligned} BufferL &:= put?.pass!.BufferL \\ BufferR &:= pass?.get?.BufferR \end{aligned}$$

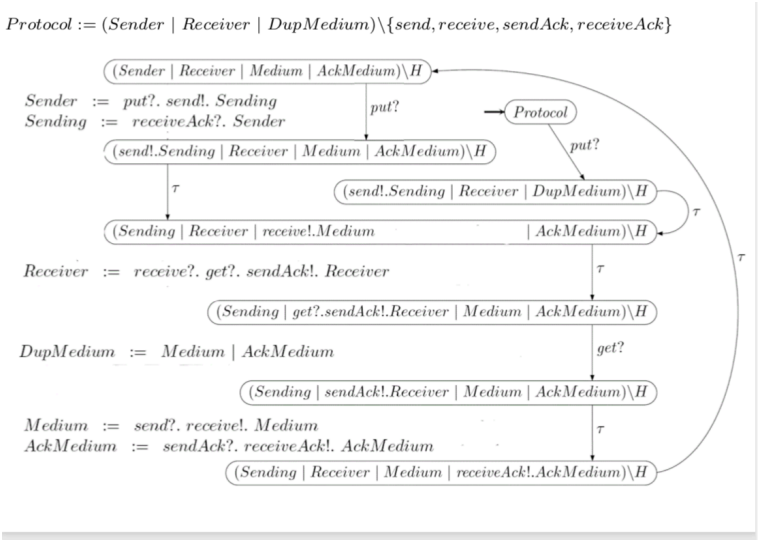
Calcular $[\![(BufferL|BufferR)\backslash\{pass!, pass?\}]\!]$

### Exemplo de um Protocolo com Ack (Pseuco)




$$\begin{aligned} Sender &:= put?.send!.Sending \\ Sending &:= receiveAck?.Sender \\ Receiver &:= receive?.get?.sendAck!.Receiver \\ Medium &:= send?.receive!.Medium \\ AckMedium &:= sendAck?.receiveAck!.AckMedium \\ DupMedium &:= Medium|AckMedium \\ ProtocolG &:= (Sender \mid Receiver \mid DupMedium)\backslash \\ &\quad \{send, receive, sendAck, receiveAck\} \end{aligned}$$

**Exemplo de um Protocolo com Ack (Pseuco, Holger H.)**

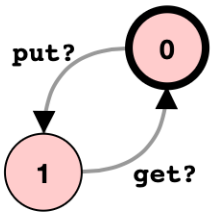



**Exemplo de um Protocolo com erro no meio –Pseuco**

$$
\begin{aligned}
Sender &:= put?.send!.Sending \\
Sending &:= receiveAck?.Sender + receiveNAck?.send!.Sending \\
Receiver &:= receive?.get?.sendAck!.Receiver + \\
&\quad gargled?.sendNAck!.Receiver \\
Medium &:= send?.(receive!.Medium + i.garbled!.Medium) \\
AckMedium &:= sendAck?.receiveAck!.AckMedium + \\
&\quad sendNAck?.receivedNAck!.AckMedium \\
DupMedium &:= Medium|AckMedium \\
Protocol &:= (Sender \mid Receiver \mid DupMedium)\backslash \\
&\quad \{send, receive, sendAck, receiveAck, \\
&\quad receiveNAck, sendNAck, garbled\}
\end{aligned}
$$

**Comportamento Externo**

Para um observador externo os processos $Buffer$, $Protocol$ e $ProtocolG$ têm o mesmo comportamento (mas LTSs diferentes, claro!).




# CCS em PSeuco -ver exemplos

If `P` and `Q` are valid CCS processes, then the following are valid CCS processes as well:

| Process | Semantics |
|---|---|
| `0` | cannot perform any action |
| `1` | terminated process – emits δ, behaves like `0` afterwards |
| `a.P` | performs action `a`, behaves like `P` afterwards |
| `P + Q` | non-deterministic choice between `P` and `Q` |
| `P \| Q` | concurrent execution of `P` and `Q` |
| `P \ {a,b,c}` | behaves like `P` except that actions `a`, `b`, `c` are **not** allowed |
| `P \ {*,a,b,c}` | behaves like `P` except that **only** actions `a`, `b`, `c` are allowed |
| `P; Q` | behaves like `P` until it terminates, then performs a τ-transition to `Q` |
| `X` | behaves like `P` if `X` was defined before as `X := P` |

A CCS action is any combination of letters starting with a lower-case letter. Some of them have a special meaning:

| Input Sequence | Action | Meaning |
|---|---|---|
| `i` | τ (tau) | represents in internal, non-observable action |
| `e` | δ (delta) | signals that a process has terminated |

CCS actions may contain a `!` or `?`. If both counterparts can be executed at the same time, they can synchronize, resulting in an internal τ transition. After `!`, a sending expression can be given, the value of which will be bound to the input variable given after `?`. Input variables may be bound to a range, defined as `range R := 0..0`, with `:R`. Ranges of string length can be specified, too: `$..$$$` allows strings of length 1 to 3.

A process name is any combination of letters starting with an upper-case letter.

Process definitions may contain an arbitrary number of variables in square brackets, which must be given when a process is called.