DN2 / V2

Notiztitel 04.03.2005

I.5. milli Process Algebra

A) Syntax of mPA: (milli PA)

$$0 \mid P+P \mid a.P \mid X \mid \text{rec } X:P \mid P \parallel_A P \mid P\backslash A \mid f(P)$$

"Renaming"

"Hiding"

Let $\tau \in Act$ be a distinguished "internal" action

where $f: Act \to Act$ with $f(\tau) = \tau$

B) Semantics

SOS as before, except:

$$\frac{P \xrightarrow{a} P'}{P\backslash A \xrightarrow{a} P'\backslash A} \quad a \notin A$$

$$\frac{P \xrightarrow{a} P'}{P\backslash A \xrightarrow{\tau} P'\backslash A} \quad a \in A$$

$$\frac{P \xrightarrow{a} P'}{f(P) \xrightarrow{f(a)} f(P')}$$

$\tau \notin A$

I.6. Syntactic and Semantic equality

$P + Q \stackrel{?}{=} Q + P$ ✓

$put.0 \stackrel{?}{=} put.0 + put.0$ ✓

$rec\,x.\, put.get.x \stackrel{?}{=} rec\,y.\, put.get.y$ ✓

$P + P \stackrel{?}{=} P$

Intermezzo:

Language: 0, 1, 2, 3, 4 ...

or: I, II, III, IV, ...

Semantics: $\{\}, \{\bullet\}, \{\bullet, \bullet\}, \{\bullet, \bullet, \bullet\}, \ldots$

$0, succ(0), succ(succ(0)) \ldots$

Calculus: $0 + x = x$

$x \cdot (y+2) = x \cdot 2 + y \cdot 2$

$x + y = y + x$

$rec\,x.(put.get.x + put.get.(x+x)) \stackrel{?}{=} rec\,y.\, put.get.y$

There are some natural structural laws one can usually agree on such as:

$$E+F = F+E \qquad E+0 = E \qquad E+E = E$$

$$(E+F)+G = E+(F+G)$$

How about the following laws?

$$\text{rec}X.E \overset{?}{=} \text{rec}Y.E\{Y/X\} \quad \text{provided } Y \text{ not free in } E$$ ✓

$$a.(E+F) \overset{?}{=} a.E + a.F$$ (⊥)

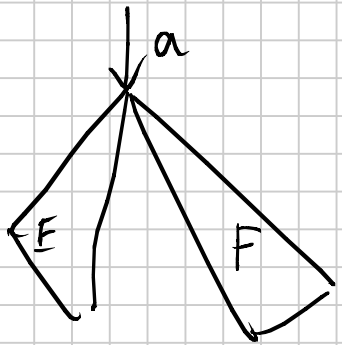
$$(E \between_A F) \between_B G \overset{?}{=} E \between_A (G \between_B F)$$

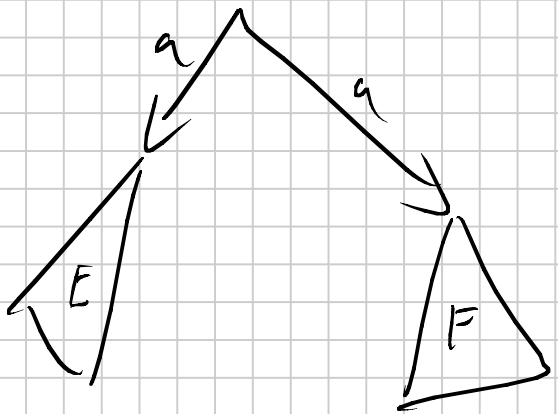
Note: Such a set of laws (axioms) induces a syntactic congruence notion owed to the following rules

$$\frac{E = F}{F = E} \qquad \frac{E = F \quad F = G}{E = G}$$

$$\frac{E = F \quad G = H}{E\{G/H\} = F\{G/H\}}$$

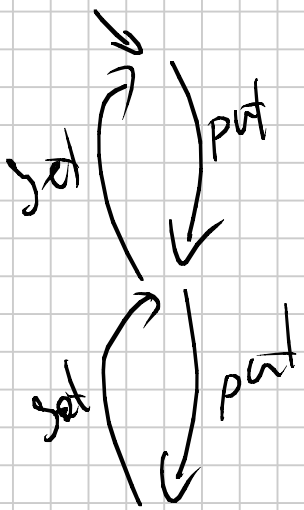
Compare $a.(E+F)$ and $a.E + a.F$

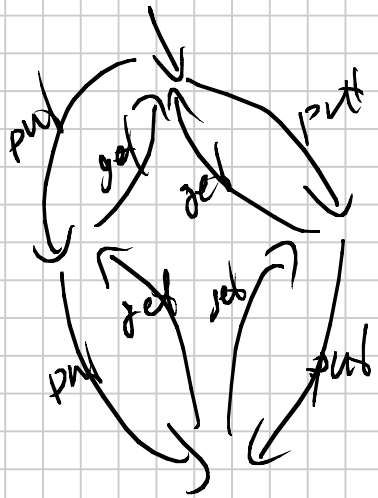


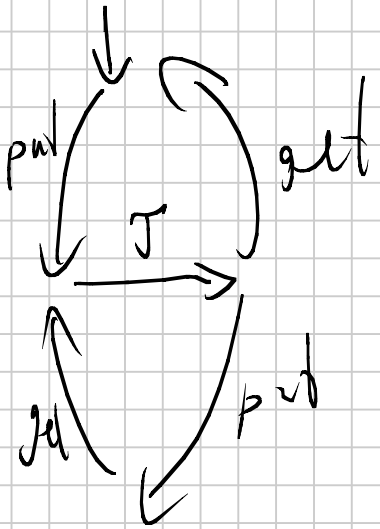


Deeper semantic issues:

We have seen 3 different specifications of a two-place buffer







So what is the true semantics of a "term"?

I.6. A Trace semantics

Let π be a (finite or infinite) path through an LTS rooted in E. $\pi = E \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_3 \cdots$

We call "$a_1 a_2 a_3 \ldots$" a trace of E.

We let "traces (E)" denote the set of all traces of E.

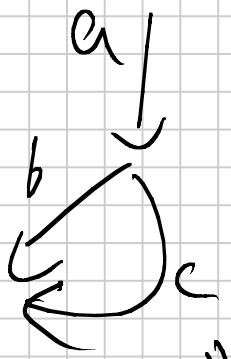
Def:

Two processes P and Q are trace-equivalent iff

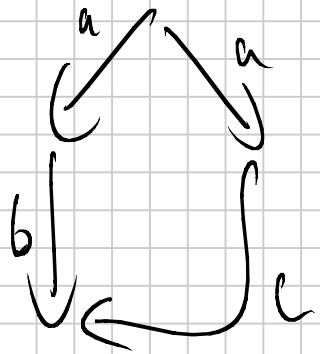
$$\text{Traces}(P) = \text{Traces}(Q).$$

Deficiency of trace semantics:

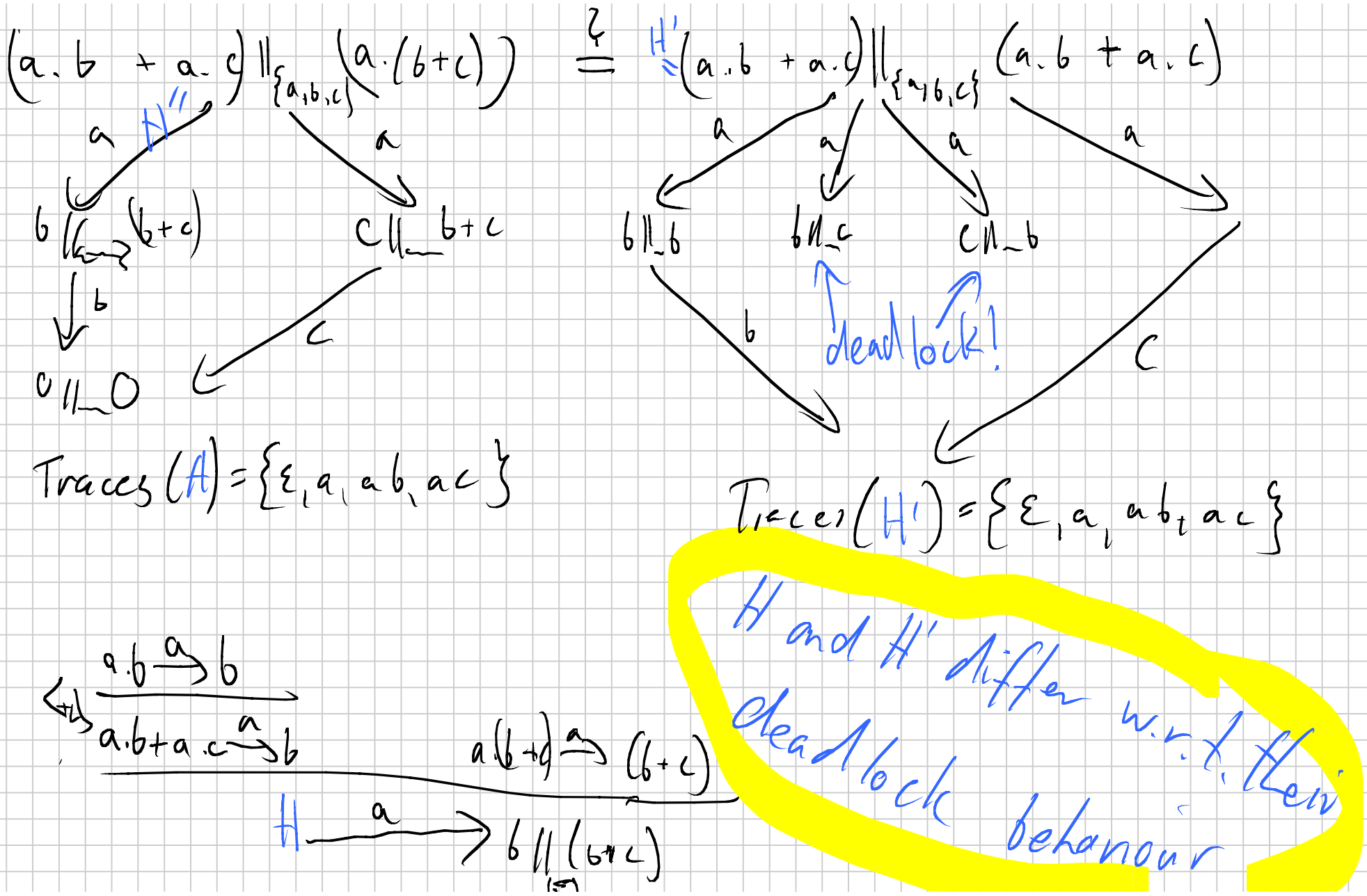
Are "$a.(b+c)$" and "$a.b + a.c$" trace-equivalent?



$Traces(a.(b+c)) = \{\varepsilon, a, ab, ac\}$



$Traces(a.b + a.c) = \{\varepsilon, a, ab, ac\}$


(a.b + a.c) ||{a,b,c} (a.(b+c))
≅
H''
H'
a
b ||_ (b+c)
c ||_ b+c
c ||_ 0
(a.b + a.c) ||{a,b,c} (a.b + a.c)
b ||_ b
b ||_ c
c ||_ b
deadlock!
Traces(H) = {ε, a, ab, ac}
Traces(H') = {ε, a, ab, ac}
a.b —a→ b
a.b + a.c —a→ b
a.(b+c) —a→ (b+c)
H —a→ b ||_ (b+c)
H and H' differ w.r.t. their deadlock behaviour

I.6.B. Requirements for a good semantics

What are we heading for?

- An equivalence relation on LTS
- A congruence relation w.r.t. the PA considered
- A relation which preserves "deadlock behaviour".

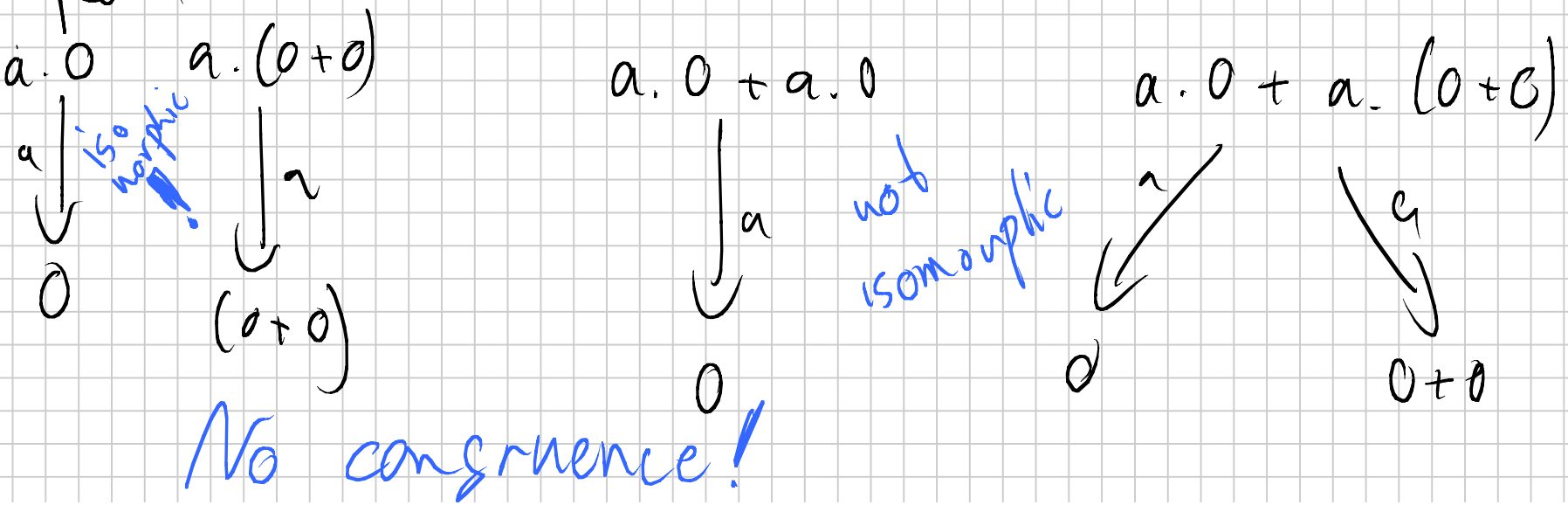
(• In principle we want to be as coarse as possible)

I.6.C Graph isomorphism semantics.

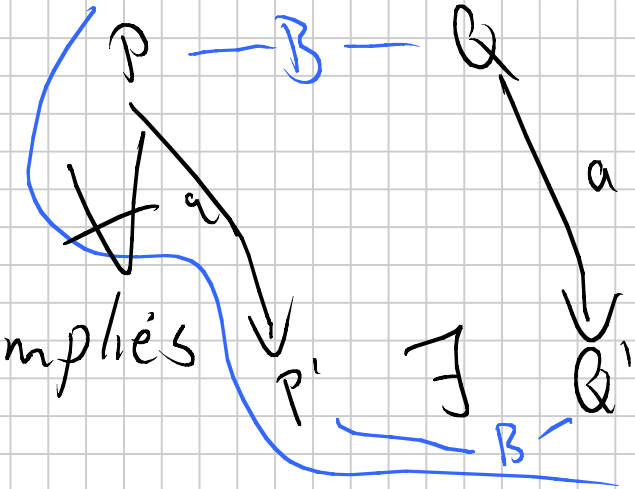
Def. Two processes P and Q are isomorphic if there is a bijective mapping $\gamma : mPA \to mPA$ such that $\gamma(P) = Q$ and

$$P' \xrightarrow{a} P'' \iff \gamma(P') \xrightarrow{a} \gamma(P'')$$

Example:



I. 6. D. Bisimulation Semantics



Def: A relation $B \subseteq nPA \times mPA$ is called a "bisimulation", if $(P, Q) \in B$ implies $\forall a \in Act$ that

i) $P \xrightarrow{a} P'$ implies $\exists Q' : Q \xrightarrow{a} Q' \wedge (P', Q') \in B$

ii) $Q \xrightarrow{a} Q'$ implies $\exists P' : P \xrightarrow{a} P' \wedge (P', Q') \in B$

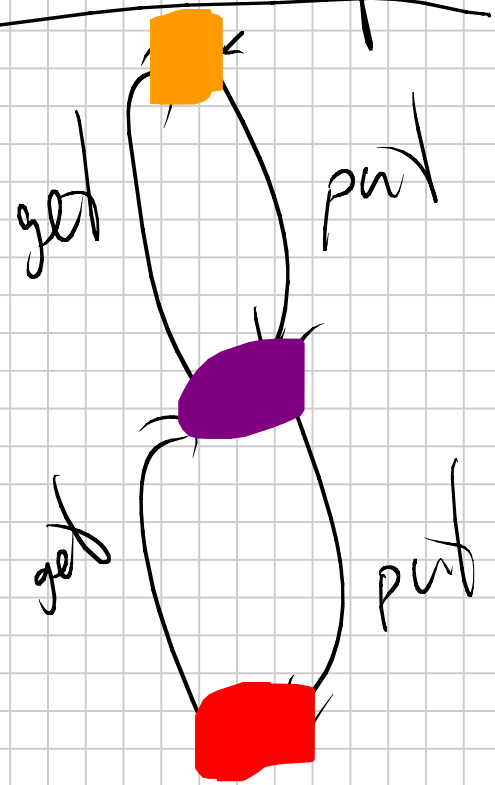
Two processe (P and Q) are called bisimilar, if they are contanicd in some bisimulation B,

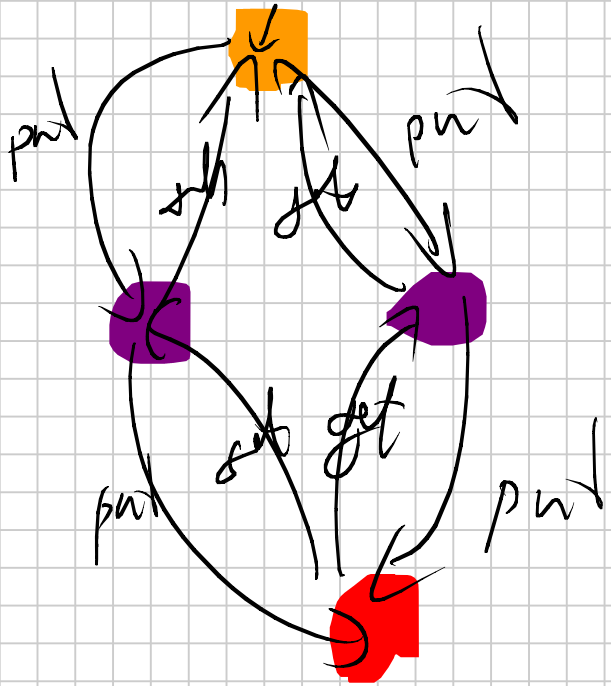
$\{(0, 0+0)\}$

we write $P \sim Q$ in this case

#7
xanples:

More examples:





Lemma 1: "$\sim$" is a bisimulation.

Proof: check conditions:

Let $(P, Q) \in \sim$, and $a \in Act$.

Assume $P \xrightarrow{a} P'$. We need to show: $\exists Q'$, $Q \xrightarrow{a} Q'$ where $(P', Q') \in \sim$

By assuption $\exists B$, such that $(P, Q) \in B$

By the def. of bisim, we have $Q \xrightarrow{a} Q''$, $(P', Q'') \in B$

so, we take $Q' = Q''$, and thus $(P', Q') \in \sim$.

Lemma 2: "$\sim$" is the largest bisimulation.

Lemma 3: "$\sim$" is an equivalence relation.

Lemma 4: "$\sim$" is a congruence relation.

Questions:

- Is each bisimulation B symmetric?
- Is $\{\}$ a bisimulation?
- Is $B \circ C$ a bisimulation if B and C are?

Proof of Lemma 4: Prove for each operator separately:

① $P \sim Q \Rightarrow a.P \sim a.Q$

② $P \sim Q \Rightarrow P + R \sim Q + R$ $\forall R$

③ $P \sim Q \Rightarrow R + P \sim R + Q$

④,⑤ $P \sim Q \Rightarrow P \|_A R \sim Q \|_A R \wedge R \|_A Q \sim R \|_A P$

⑦, ⑥ $P \sim Q \Rightarrow [P]_A \sim [Q]_A \wedge f(P) \sim f(Q)$

We only show ②

Choose R.

Define

$\mathcal{B} = \{(S+R, T+R) \mid S \sim T\} \cup \sim$

We only consider "processes" where rec-operators do not alter the behaviour. However congruence also holds for open expressions.

Claim: B is a bisimulation.

Let $(P, Q) \in B$ and $a \in Act$ and assume $P \xrightarrow{a} P'$

Note that P must have structure $P = S + R$ and also $Q = T + R$

Due to the operational rules, either $R \xrightarrow{a} P'$ or $S \xrightarrow{a} P'$

If $R \xrightarrow{a} P'$ we are ALMOST done, since $Q \xrightarrow{a} P'$. We have $P' \sim P'$ and thus we are done.

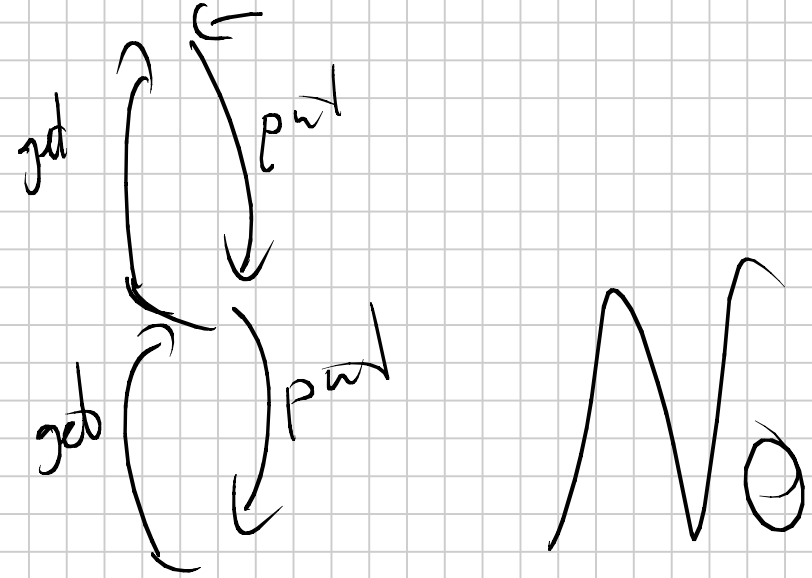
The only interesting case is that:

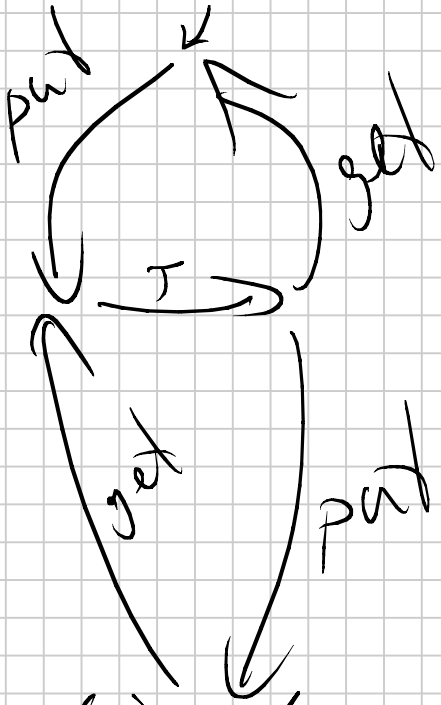
Otherwise: $S \xrightarrow{a} P'$. We need Q' such that $T \xrightarrow{a} Q'$ and $(P', Q') \in \sim$. Since $S \sim T$, there is a bisim. B such that $(S, T) \in B$. We get $T \xrightarrow{a} Q''$ with $(P', Q'') \in B$.

Choosing $Q' = Q''$ $(P', Q') \in \sim$ follows.

I.7. Abstract Semantics

Are those two bisimilar?





Bisimulation is still a little too fine, because it is not "τ-abstract".

I.7. A Branchy bisimulation

Def: A symmetric relation $B \subseteq mPA \times mPA$ is called a "branchy bisimulation" if $(P,Q) \in B$ implies for all $a \in Act$ that

$P \xrightarrow{a} P'$ implies

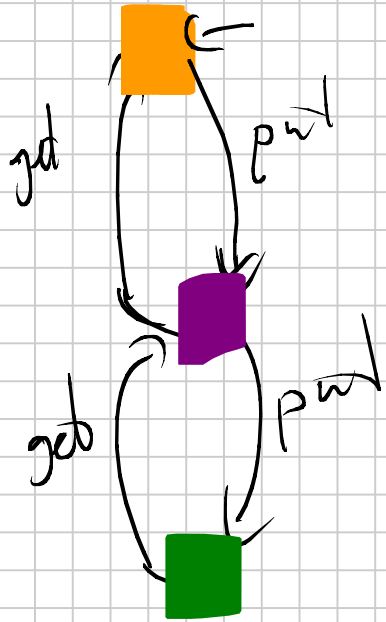
$$\left(a = \tau \text{ and } (P,P') \in B\right) \text{ or}$$

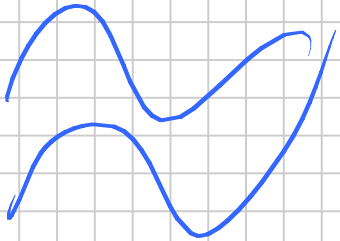
$$\left(\exists Q', Q'' : Q \xrightarrow{\tau^*} Q'' \xrightarrow{a} Q' \text{ and } (P,Q'') \in B \text{ and } (P',Q') \in B\right)$$

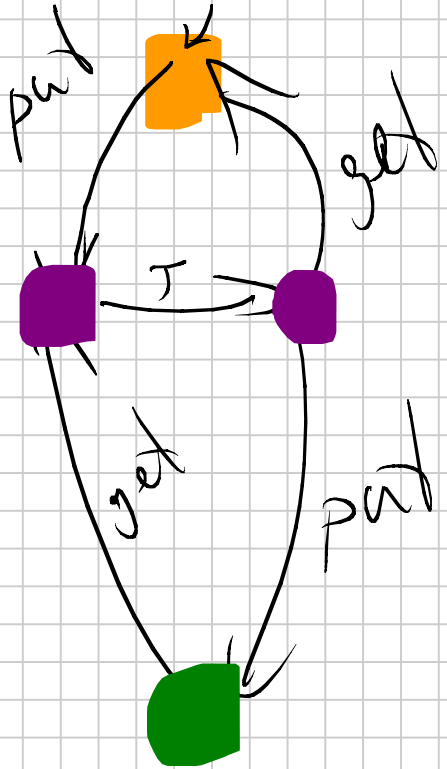
Two processes are called "branchy bisimilar" if they are contained in some branchy bisimulation.

Notation: $\approx$

Example:







Lemma 1 - 4 as before

Lemma 4 does not hold for "τ", but this can be "fixed".

Is $B \circ C$ a b. bisim. if B and C are.

$\Downarrow$

Lemma 3 needs different proof

$\exists.\ b \approx b$, but

$\forall.\ b + a \not\approx b + a$

counterexample

⇝ CADP-demo 01