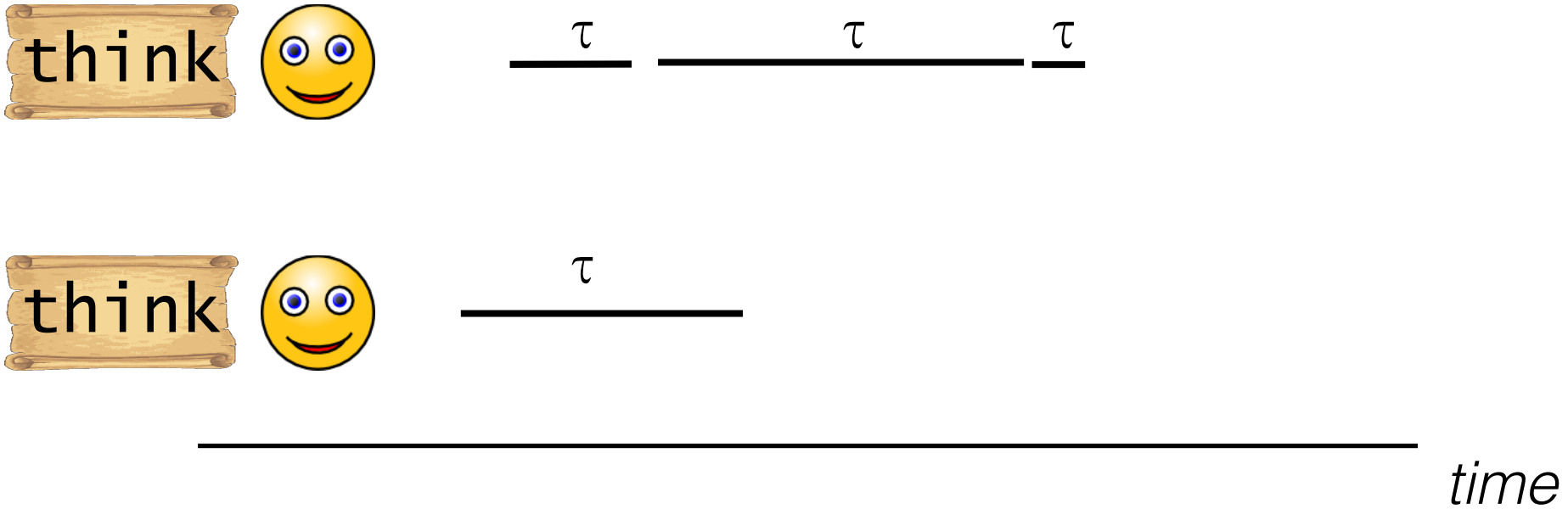
π-calculus

Introduction

Open discussion on the words “parallelism” and ‘’concurrency’’ and/or other related concepts

Synchronisation

unobservable transition

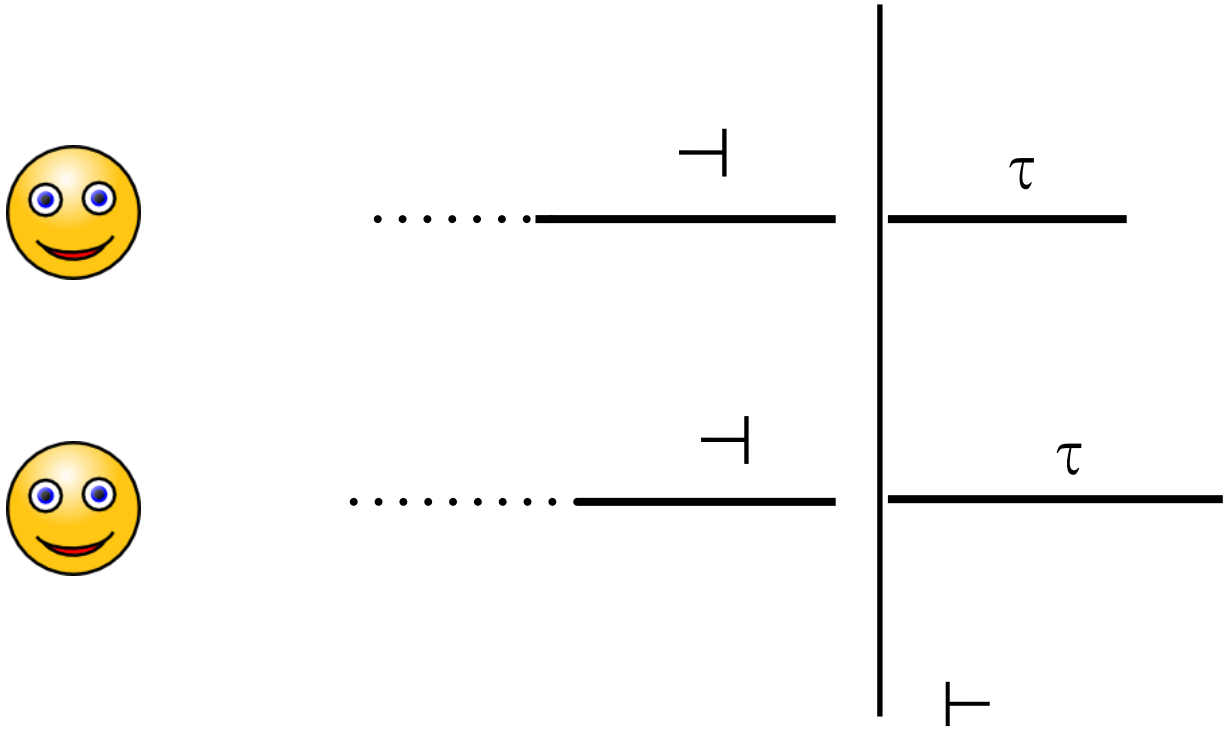


think = τtransition = unit of computation

unobservable from outside

4

synchronisation



⊣ = pause / cooperate

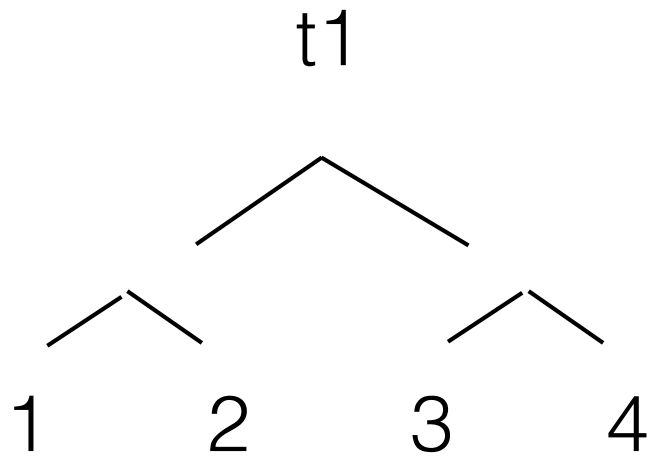
same fringe

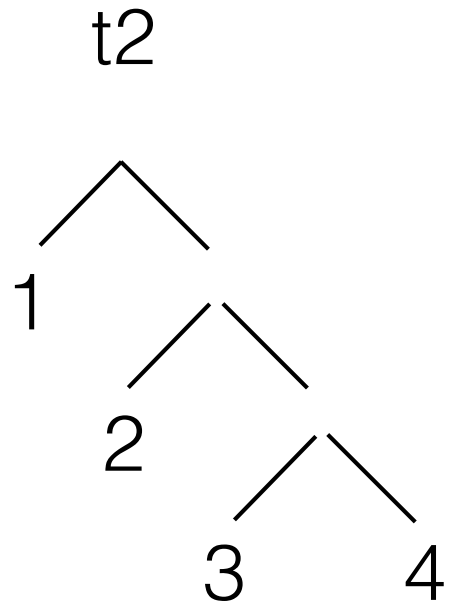
```
type tree =
 | Leaf of int
 | Node of tree * tree
```

```
let rec dfs f t = match t with
 | Leaf n -> (f n)
 | Node (l,r) -> (dfs f l); (dfs f r)
```

```
dfs print_int t1;
print_newline();
dfs print_int t2
```

```
1234
1234
```

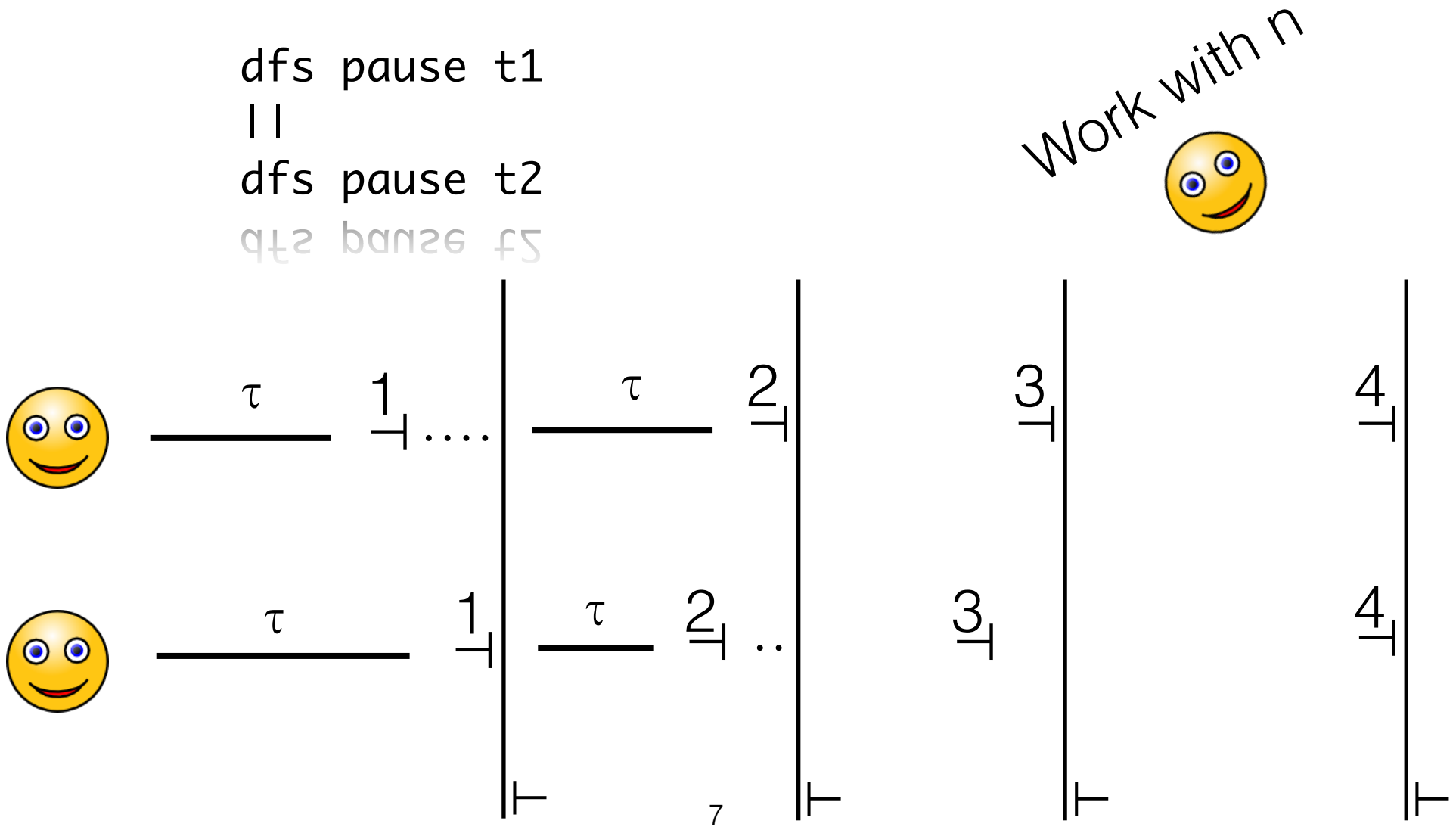




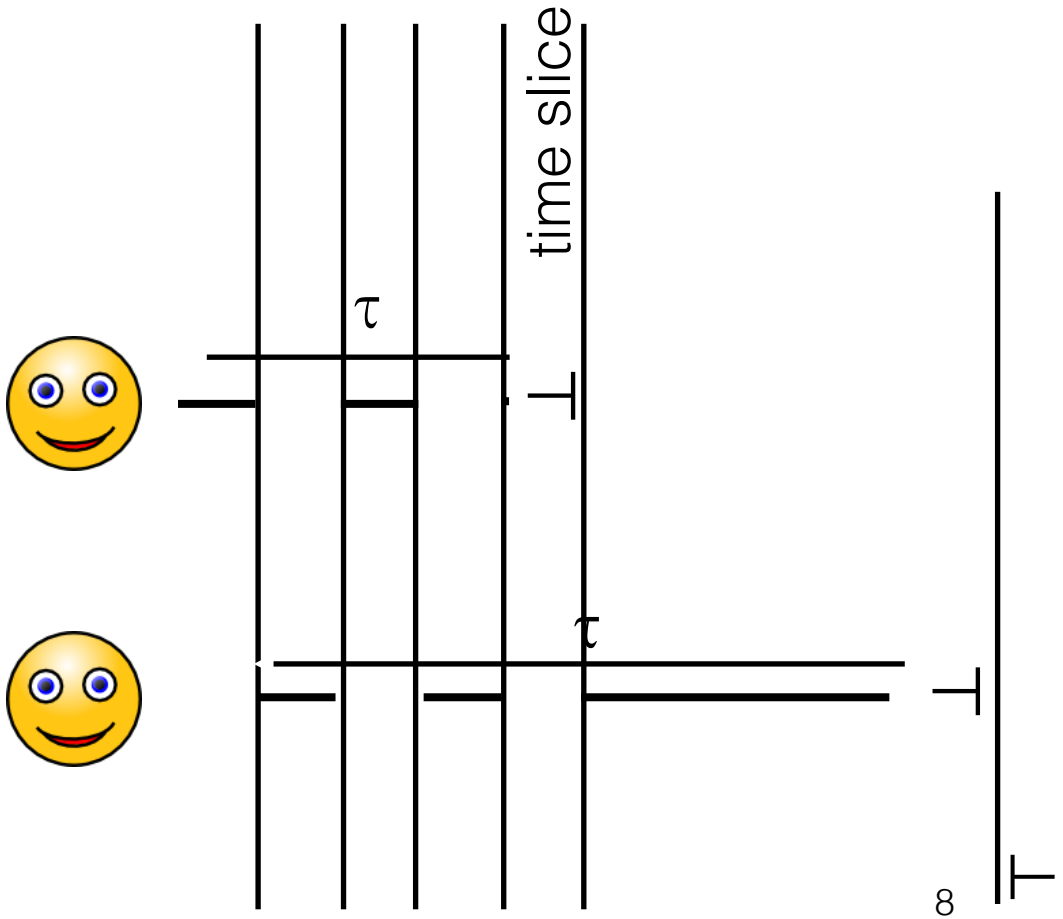
6

synchronised same fringe

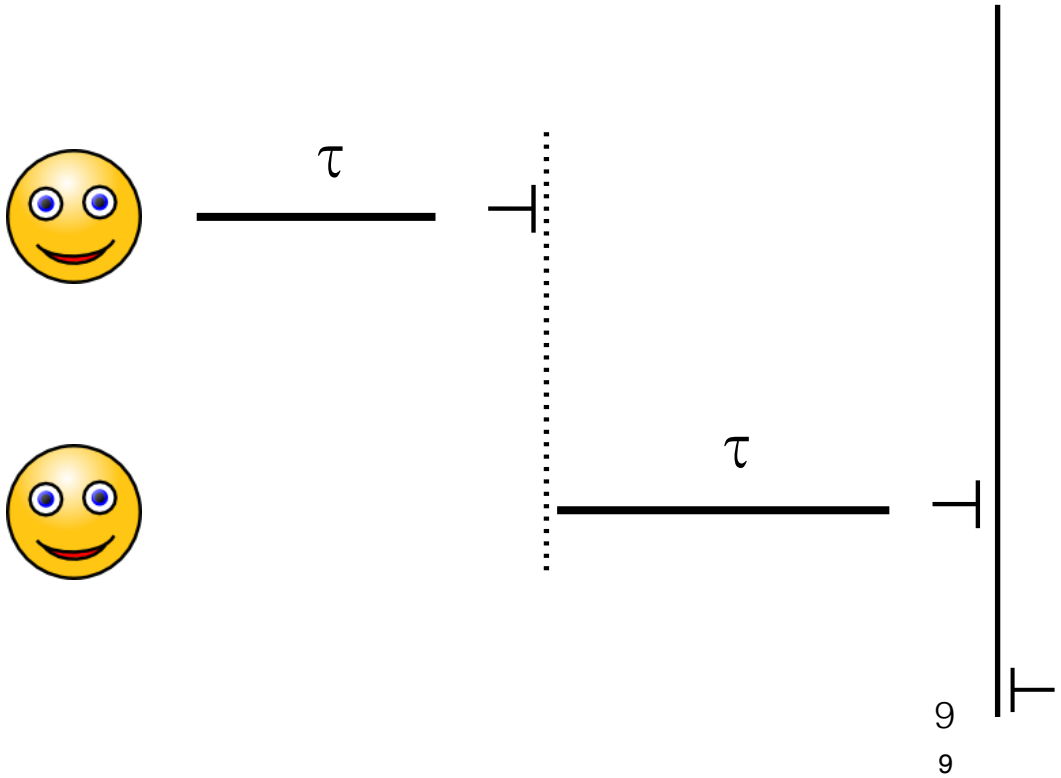
```
dfs pause t1
||
dfs pause t2
```



Time sharing (*Preemptive*)

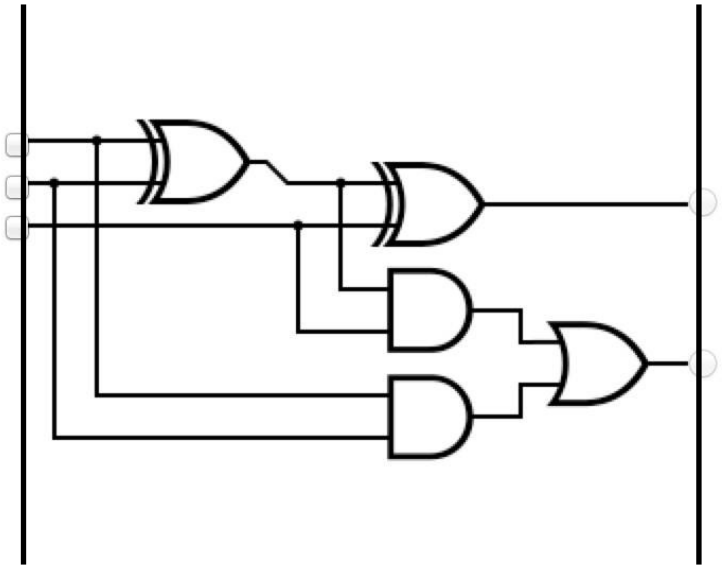


Time sharing (*Non-Preemptive*)



Gate level

Lets-Increment-I-Logic-Gate-Style



clock speed

Global Synchronisation

11

Abstract syntax

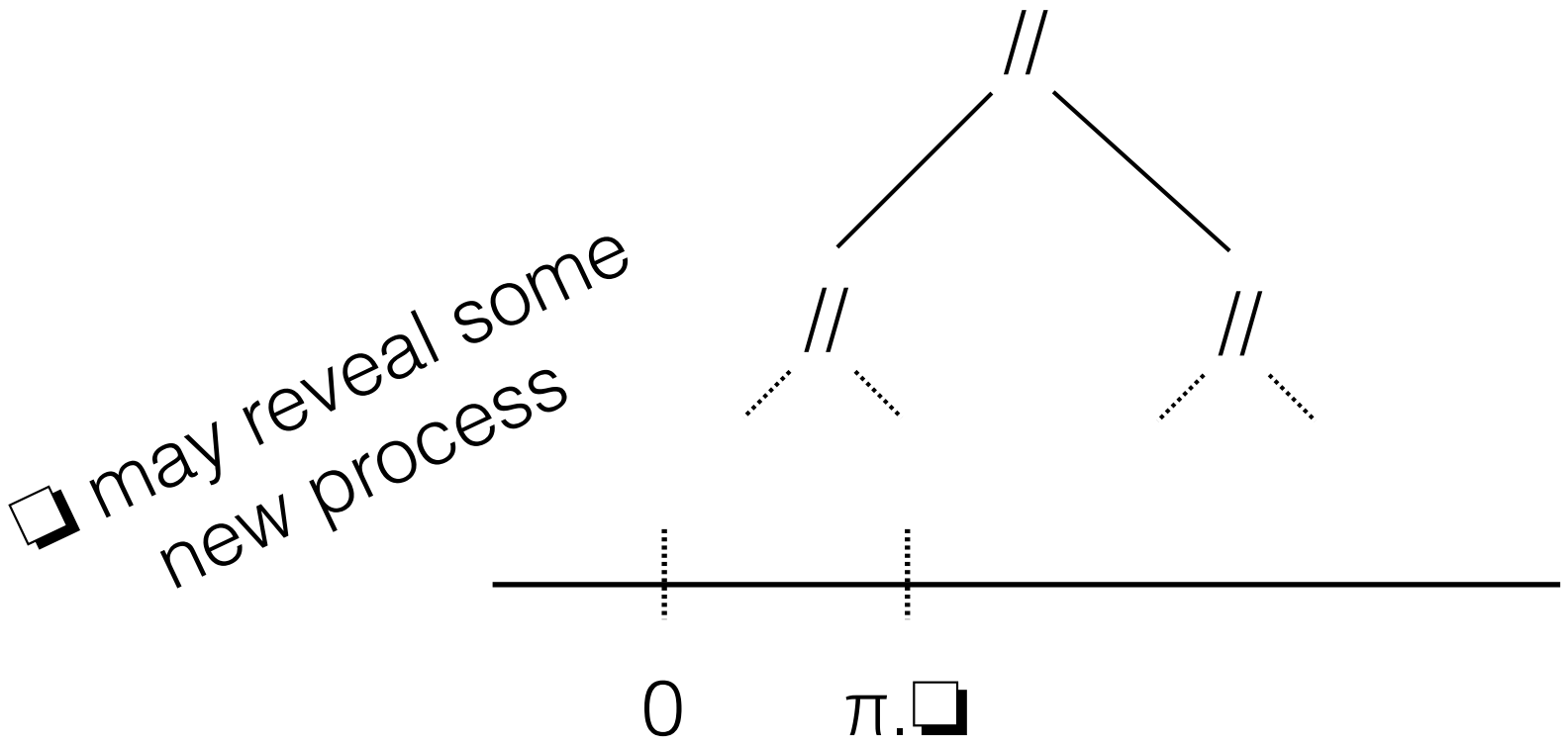
$$P := \quad 0 \mid \pi.P \mid P \parallel P$$

$$\pi := \quad \tau \mid \dashv$$

π = prefix

12

Tree of processes



When all leaves are 0 or ⊣.❏ then one synchronisation is done

Montrer la forme normal pour synchro

Context

$$
\begin{aligned}
C \;=\;& [\,] \\
\mid\;& P \parallel C \\
\mid\;& C \parallel P
\end{aligned}
$$

Well adapted to skip a tau : $C[\tau.\square] \rightarrow C[\square]$

Also adapted to remove 0s : $C[0 \parallel P] \rightarrow C[P]$

But also need the symmetric rule : $C[P \parallel 0] \rightarrow C[P]$

Not easy to specify that all leaves are in $\dashv.\square$ form

But the normal form is in synchronised form

15

Annotated SOS

SOS = Structural Operational Semantics

$\tau.P \longrightarrow^{\tau} P$

$\dashv.P \longrightarrow^{\dashv} P$

$P \longrightarrow^{a} P'$

$0 \,||\, P \longrightarrow^{\tau} P$

tau are not in //

$$\frac{P \longrightarrow^{\tau} P'}{(P||Q) \longrightarrow^{\tau} (P'||Q)}$$

and symmetric!

$$\frac{Q \longrightarrow^{\tau} Q'}{(P||Q) \longrightarrow^{\tau} (P||Q')}$$

$$\frac{P \longrightarrow^{\dashv} P' \quad Q \longrightarrow^{\dashv} Q'}{(P||Q) \longrightarrow^{\dashv} (P'||Q')}$$

0 remains on trees

M-tau->* —|-> M’

Equivalence/Congruence

$$\frac{P' \equiv P \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$$

Pourquoi = a droite?

$$P \equiv P$$

$$\frac{P \equiv Q}{Q \equiv P}$$

$$\frac{P \equiv Q \quad Q \equiv R}{P \equiv R}$$

$$\frac{P \equiv Q}{C[P] \equiv C[Q]}$$

$$P \parallel Q \equiv Q \parallel P \qquad P \parallel 0 \equiv P$$

$$P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$$

17

Point to point synchronization

18

Calculus of Communicating Systems

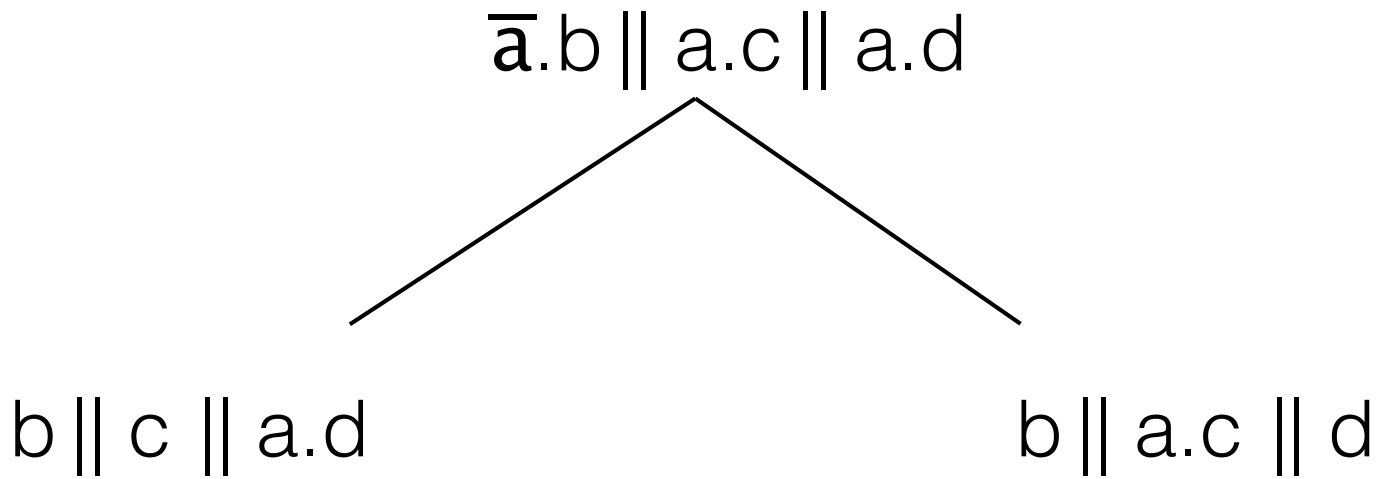
$$P := \quad 0 \;\mid\; \pi.P \;\mid\; P \parallel P$$

$$\pi := \quad \tau \;\mid\; \overline{a} \;\mid\; a$$

$$\ldots \parallel \overline{a}.P_1 \parallel \ldots \parallel a.P_2 \parallel \ldots \longrightarrow \ldots \parallel P_1 \parallel \ldots \parallel P_2 \parallel \ldots$$

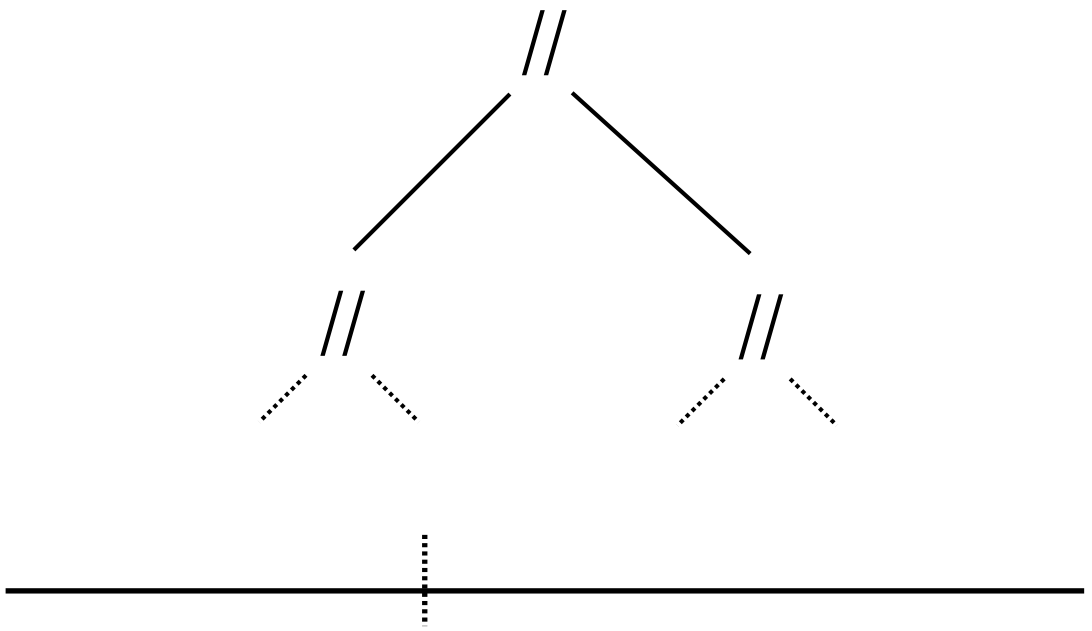
19

Non confluent



20

Tree of processes



We want to synchronise some a.❑ $\overline{a}$.❑

21

Bihole context

$C^2 = C^2 \parallel P \mid P \parallel C^2 \mid C^1 \parallel C^1$

$C^1 = [] \mid P \parallel C^1 \mid C^1 \parallel P$

$C^2[\overline{a}.P_1][a.P_2] \longrightarrow C^2[P_1][P_2]$

and symetry!

22

With annotations

$\tau.P \longrightarrow^{\tau} P$

$a.P \longrightarrow^{a} P$

$\overline{a}.P \longrightarrow^{\bar{a}} P$

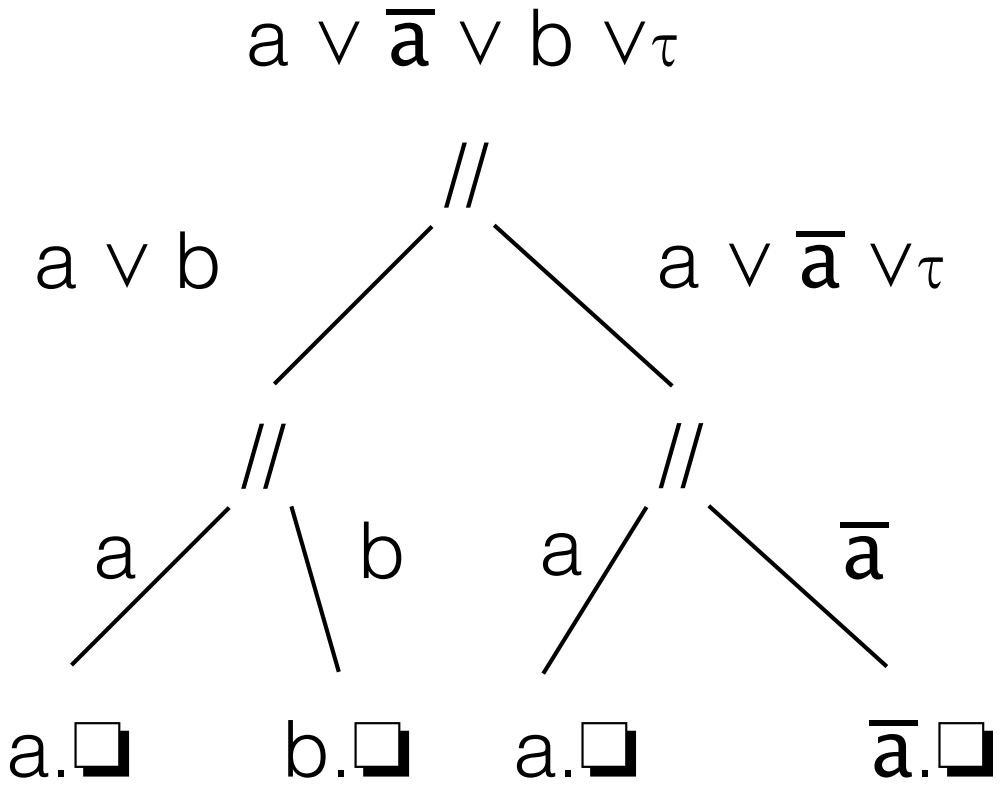
$$\frac{P \longrightarrow^{a} P' \qquad Q \longrightarrow^{\bar{a}} Q'}{(P \,\|\, Q) \longrightarrow^{\tau} (P' \,\|\, Q')}$$

$$\frac{P \longrightarrow^{\alpha} P'}{(P \,\|\, Q) \longrightarrow^{\alpha} (P' \,\|\, Q)}$$

Can we have two sync in the same step?

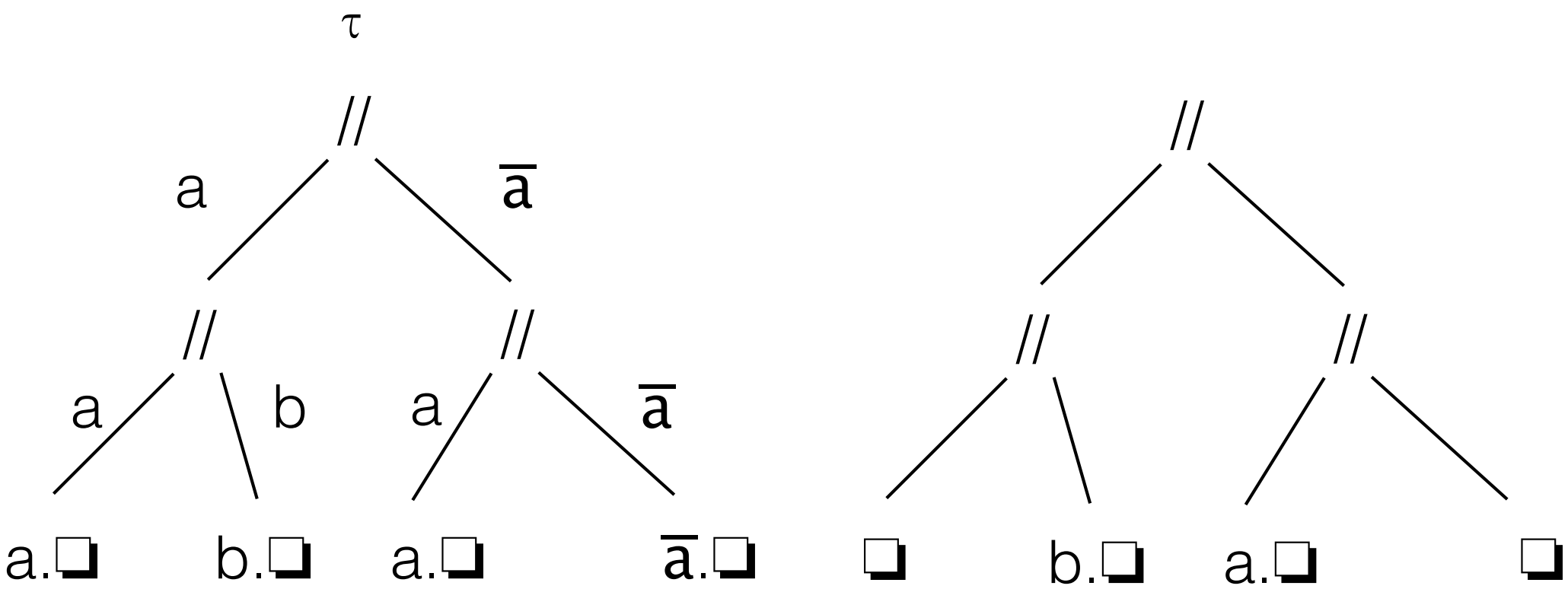
M-tau->M'

Example



24

One choice



without annotation

$$(\tau.P_1 \parallel P_2) \quad \longrightarrow \quad (P_1 \parallel P_2)$$

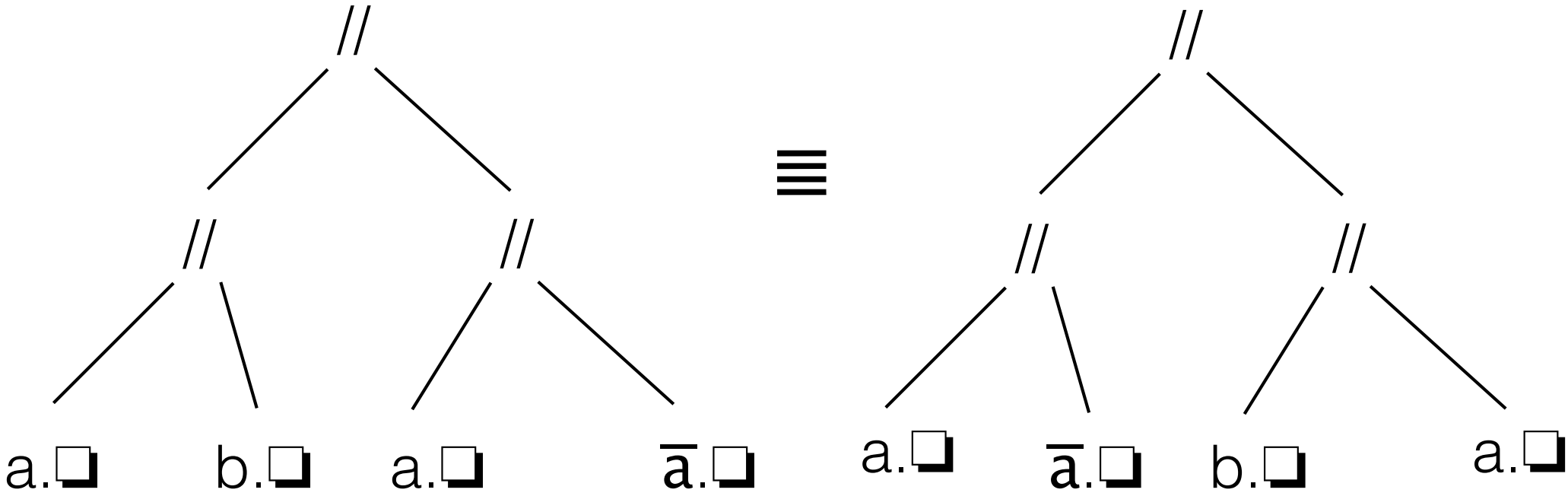
$$((a.P_1 \parallel \overline{a}.P_2) \parallel P_3) \quad \longrightarrow \quad ((P_1 \parallel P_2) \parallel P_3)$$

Congruence can moves the candidates to the top left, adding a 0 if needed

The simplest way, but it is the equivalence which decide the synchronisation

26

Example



27

Alternative

$$\tau.P \longrightarrow P$$

$$(a.P_1 \parallel \overline{a}.P_2) \longrightarrow (P_1 \parallel P_2)$$

$$\frac{P \longrightarrow P'}{(P \parallel Q) \longrightarrow (P' \parallel Q)}$$

28

Full synchronisation

Puzzle : How many channels are needed to synchronise N processes

Result for 3 : $\tau.a.\overline{b} \parallel \tau.b.\overline{c} \parallel \tau.\overline{a}.c$

Bad result if one process ends while another still in it's τ

29

π-calculus

30

Value transmission

$$P := \; 0 \mid \pi.P \mid P \,\|\, P \qquad \pi := \; \tau \mid \overline{a}\, x \mid a(y)$$

$$\ldots \,\|\, \overline{a}\, x.P_1 \,\|\, \ldots \,\|\, a(y).P_2 \,\|\, \ldots$$

$$\longrightarrow$$

$$\ldots \,\|\, P_1 \,\|\, \ldots \,\|\, P_2\{y:=x\} \,\|\, \ldots$$

a(y).P acts as a binder of the name y in P

Values are channel names

31

Substitution

$$(P \parallel Q)\{y:=x\} \quad = \quad (P\{y:=x\} \parallel Q\{y:=x\})$$

$$(\tau.P)\{y:=x\} \quad = \quad \tau.(P\{y:=x\})$$

$$(a(y).P)\{y:=x\} \quad = \quad a(y).P$$

$$(a(x).P)\{y:=x\} \quad = a(z).(P\{x:=z\}\{y:=x\}) \quad z \notin Free(P)$$

Definition of Free and Bound

32

Bihole context

$C^2 = C^2 \parallel P \mid P \parallel C^2 \mid C^1 \parallel C^1$

$C^1 = [] \mid P \parallel C^1 \mid C^1 \parallel P$

and symetry!

$C^2[\overline{a}\ x.P_1][a(y).P_2] \longrightarrow C^2[P_1][P_2\{y:=x\}]$

Still elegant

code in Java Definition of fill

33

With equivalence

$$\overline{a}\, x.P_1 \parallel a(y).P_2 \quad \longrightarrow \quad P_1 \parallel P_2\{y:=x\}$$

$$\tau.P \quad \longrightarrow \quad P$$

34

With annotations

Late substitution

$$a(y).P \xrightarrow{a(y)} P \qquad \overline{a}\,x.P \xrightarrow{\overline{a}\,x} P$$

$$\frac{P \xrightarrow{a(y)} P' \qquad Q \xrightarrow{\overline{a}\,x} Q'}{(P \parallel Q) \xrightarrow{\tau} (P'\{y:=x\} \parallel Q')}$$

Failure

Early substitution

$$a(y).P \xrightarrow{a(x)} P\{y:=x\} \qquad \overline{a}\,x.P \xrightarrow{\overline{a}\,x} P$$

$$\frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}\,x} Q'}{(P \parallel Q) \xrightarrow{\tau} (P' \parallel Q')}$$

Restriction

$$
\begin{aligned}
P := \quad & 0 \\
\mid \quad & \pi.P \\
\mid \quad & P \parallel P \\
\mid \quad & \nu \mathbf{a}\ P
\end{aligned}
\qquad\qquad
\begin{aligned}
\pi := \quad & \tau \\
\mid \quad & \overline{\mathbf{a}}\ x \\
\mid \quad & a\ (y)
\end{aligned}
$$

Intuition $\overline{\mathbf{a}}\ x.P \parallel \nu a\ (a(y).Q)$ doesn't communicate

ν is also a binder but **a** is not a parameter

Restriction equivalence

$$\nu a\ 0 \equiv 0$$

$$\nu a\ (P \parallel Q) \equiv (\nu a\ P) \parallel Q \qquad a \notin \mathbf{Free}(Q)$$

$$\nu a\ \nu b\ P \equiv \nu b\ \nu a\ P$$

Prove that $a \notin \mathbf{Free}(P)$ implies $\nu a\ P \equiv P$

Reduction under restriction

$$\frac{P \longrightarrow P'}{\nu a\, P \longrightarrow \nu a\, P'}$$

Looks like strong reduction

$(\nu x\ \bar{a}\ x) \quad \| \quad a\,(y).P$

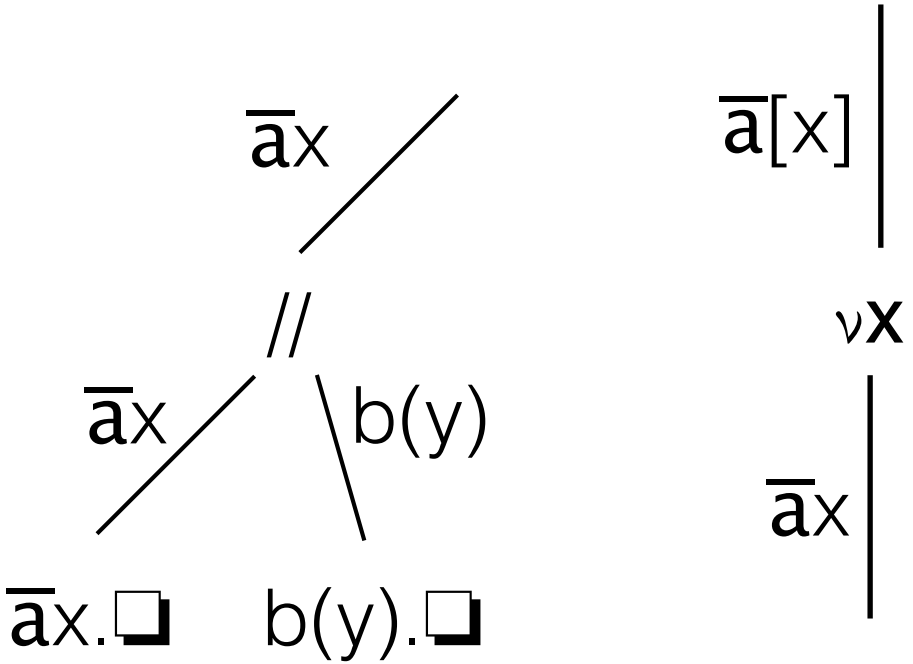
$\equiv (\nu f\ \bar{a}\ f) \quad \| \quad a\,(y).P \qquad\qquad f \notin \mathrm{Free}(P)$

$\equiv \nu f\,(\bar{a}\ f \quad \| \quad a\,(y).P)$

$\longrightarrow \nu f\,(0 \quad \| \quad P\{y:=f\})$

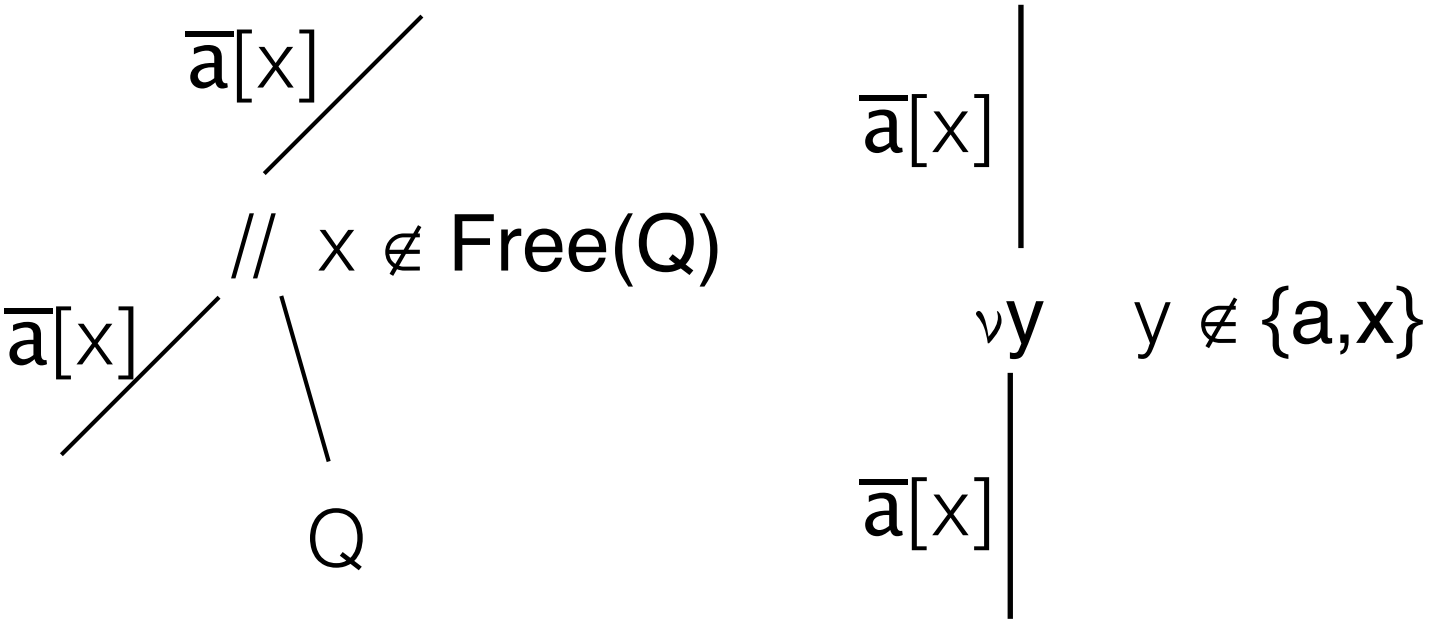
38

Annotation : moving up a restriction Bound output rule



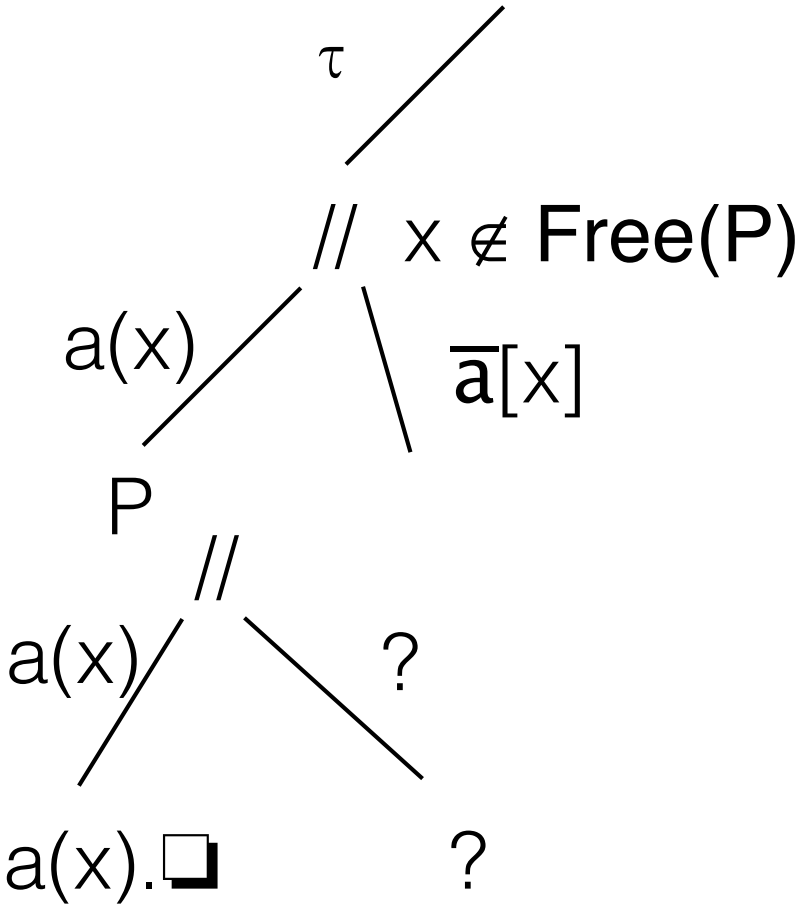
$$\frac{P \xrightarrow{\bar{a}\,x} P'}{\nu x\, P \xrightarrow{\bar{a}[x]} P'}$$

moving up


$\overline{a}[x]$
// $x \notin$ Free(Q)
$\overline{a}[x]$
Q
$\overline{a}[x]$
νy $y \notin \{a,x\}$
$\overline{a}[x]$

40

fixing the restriction



$$\frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\bar{a}[x]} Q'}{(P \parallel Q) \xrightarrow{\tau} \nu x\,(P' \parallel Q')}$$

Big step

Using “real” process

No substitution

Reveal the channel structure

David N Turner’s thesis

42

Running a term

```
public abstract class Term {
    public abstract void run(Env e);

    public static void addProcess(final Term p, final Env env) {
      new Thread() {
        public void run() {
          p.run(env);
        }
      }.start();
      return;
   }
…
```

running a ||

```
public class Par extends Term {
    Term p1;
    Term p2;

   public void run(Env env) {
      Term.addProcess(p2, env);
      p1.run(env);
   }
…
```

Restriction = channel creation

```
public class New extends Term {
    Var channel;
    Term p;

   public void run(Env env) {
       p.run(new Env(channel, new Channel(channel.name), env));
   }
…
```

Sequence : π.P

```
public class Seq extends Term {
    Op op;
    Term p;

   public void run(Env env) {
     p.run(op.run(env));
    }
…
```

```
public abstract class Op {
    public abstract Env run(Env env);
}
```

read/write

```
 public class Read extends Op {
     Var channel;
     Var var;

    public Env run(Env env) {
      return(new Env(var, env.get(channel).read(), env));
     }
 …
```

```
public class Write extends Op {
    Var channel;
    Var value;

    public  Env run(Env env) {
     env.get(channel).write(value==null ? Value.NULL : env.get(value));
     return(env);
    }
…
```

Channels are values

```
public class Channel extends Value {
    BlockingQueue<Value> q;

    public Value read() {
      try {
       return(q.take());
      } catch(Exception e) {
       throw(new RuntimeException(e));
      }
    }

    public void write(Value val) {
      q.offer(val);
    }
…
```

Recursion

$$
\begin{aligned}
P := \quad & 0 \\
\mid \quad & \pi.P \\
\mid \quad & P \parallel P \\
\mid \quad & \nu \mathbf{a}\ P \\
\mid \quad & !P
\end{aligned}
\qquad\qquad
\begin{aligned}
\pi := \quad & \tau \\
\mid \quad & \overline{\mathbf{a}}\ x \\
\mid \quad & a\ (y)
\end{aligned}
$$

Intuition $!P \equiv P \parallel !P$

it is the equivalence which unfold

but not $\dfrac{P \longrightarrow P'}{!P \longrightarrow !P'}$ and not $!(\nu \mathbf{a}\ P) \equiv \nu \mathbf{a}\ !P$

Unfolding by rule

$$\frac{P \parallel !P \longrightarrow^{\alpha} P'}{!P \longrightarrow^{\alpha} P'} \qquad \text{better} \qquad \frac{P \longrightarrow^{\alpha} P'}{!P \longrightarrow^{\alpha} P' \parallel !P}$$

$$\frac{P \longrightarrow^{a(x)} P' \qquad Q \longrightarrow^{\bar{a}\,x} Q'}{(P \parallel Q) \longrightarrow^{\tau} (P' \parallel Q')} \qquad \frac{P \longrightarrow^{a(x)} P' \qquad P \longrightarrow^{\bar{a}\,x} P''}{!P \longrightarrow^{\tau} (P' \parallel P'' \parallel !P)}$$

$$\frac{P \longrightarrow^{a(x)} P' \qquad Q \longrightarrow^{\bar{a}[x]} Q'}{(P \parallel Q) \longrightarrow^{\tau} \nu a\,(P' \parallel Q')} \qquad \frac{P \longrightarrow^{a(x)} P' \qquad P \longrightarrow^{\bar{a}[x]} P''}{!P \longrightarrow^{\tau} (\nu a\,(P' \parallel P'')) \parallel !P}$$

Example

51

Big step

The first action of a repeat must be a read

```
public class Repeat extends Term {
    Read op;
    Term p;

   public void run(Env e) {
     Env nenv = op.run(e);
     Term.addProcess(this, e);
     p.run(nenv);
    }
…
```

Could we start with a write?

52

Non deterministic choice

$$
\begin{aligned}
P ::= \quad & M \\
\mid \quad & P \parallel P \\
\mid \quad & \nu a\, P \\
\mid \quad & !P
\end{aligned}
$$

$$
\begin{aligned}
M ::= \quad & 0 \\
\mid \quad & \pi.P \\
\mid \quad & M + M
\end{aligned}
$$

Intuition : (P + Q) will choice to do P or Q
P and Q must begin with an action (or 0)
If chosen, P **must** do its first action

Equivalence

$$P + Q \equiv Q + P \qquad P + 0 \equiv P$$

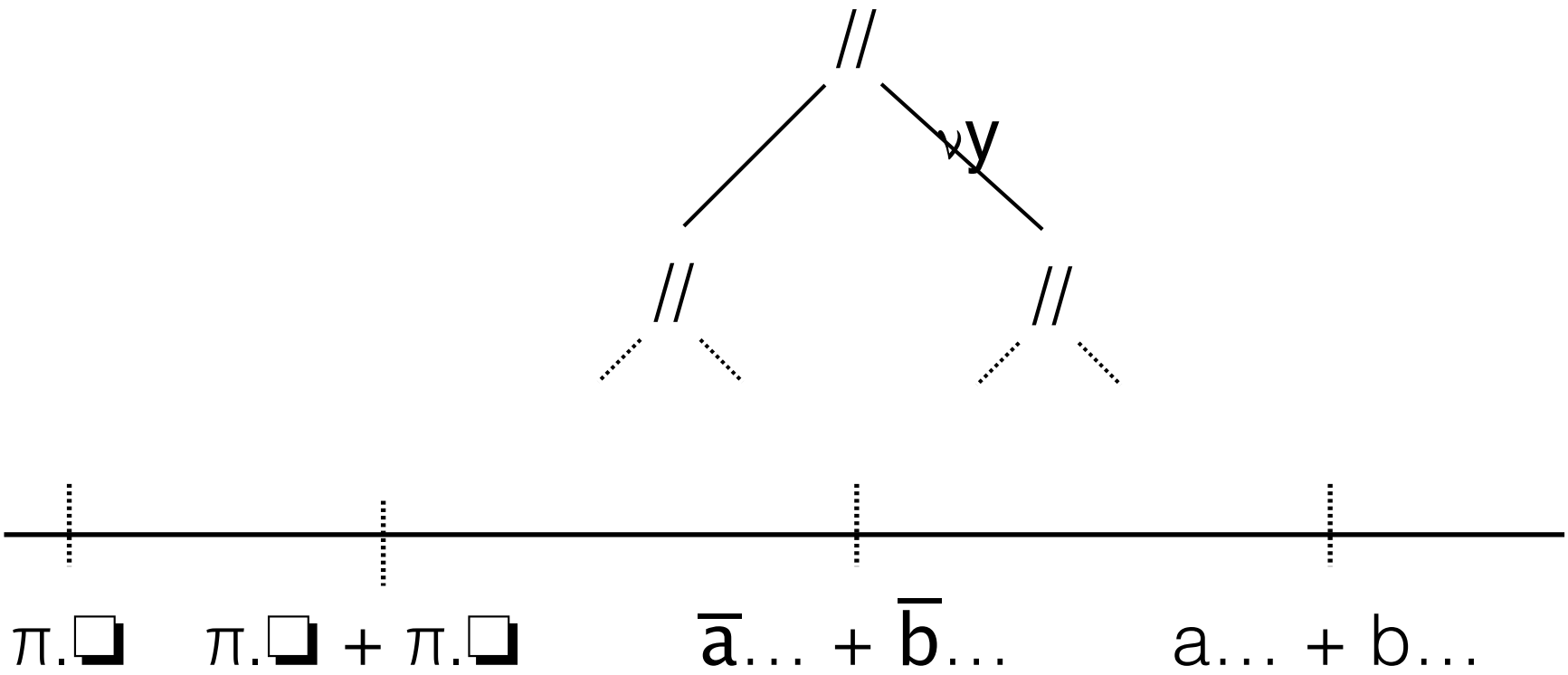
$$P + (Q + R) \equiv (P + Q) + R$$

$$(\overline{a}\, x.P_1 + M_1) \parallel (a(y).P_2 + M_2) \longrightarrow P_1 \parallel P_2\{y:=x\}$$

$$\tau.P + M \longrightarrow P$$

$$\frac{P \xrightarrow{\alpha} P'}{(P + Q) \xrightarrow{\alpha} P'}$$

Process



SynchronousQueue cannot be used

Variations

Guarded choice $\pi := \ldots \mid [x=y]\,\pi$

Intuition: $[x=x]\,\pi \equiv \pi$

Polyadic

$\overline{a}\langle x_1 \ldots x_n \rangle.P_1 \parallel a(y_1 \ldots y_n).P_2 \longrightarrow P_1 \parallel P_2\{y_1:=x_1\}\ldots\{y_n:=x_n\}$

Asynchronous

Intuition: no wait on write

56