

LANGUAGES, LOGICS, TYPES AND TOOLS FOR CONCURRENT SYSTEM MODELLING

Ramūnas Gutkovas

ramunas@fct.unl.pt

2016-12-14 NOVA LINCS



UPPSALA
UNIVERSITET

2011 MSc

2016 PhD



TITLE ABOUT MYSELF



2007 BSc

Startup
2007-2009



Uppsala, Sweden

Kaunas, Lithuania



BUGS

OR, WHEN MACHINES GO WRONG

Windows

An error has occurred. To continue:

Press Enter to return to Windows, or

Press CTRL+ALT+DEL to restart your computer. If you do this, you will lose any unsaved information in all open applications.

Error: 0E : 016F : BFF9B3D4

Press any key to continue _



Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: HAL_INITIALIZATION_FAILED

EXPENSIVE BUGS



Pentium FDIV bug, 1994

\$475 million worth of recalls



Ariane 5 went “poof”, 1996

Integer Overflow

\$ 500 million loss

goo.gl/bU36B2

KILLER BUGS

Therac-25, 1980s



Due to a race condition
produced a lethal radiation burst

5 killed

Toyota, 2010



Unintended acceleration
Software bug

89 killed

Many more:

goo.gl/GVQIIC

Microsoft calls them 1 million dollar bugs!

SECURITY

Heartbleed



OpenSSL

A bug that allows to obtain keys
Most of the internet affected

SSL is foundation for ecommerce.

CVE-2016-5195



Data race in the linux kernel
since 2007 allows to escalate
privileges

Millions of Android devices
vulnerable

UNPRECEDENTED IMPLICATIONS

Celebgate

Celebrity Apple's iCloud
accounts hacked

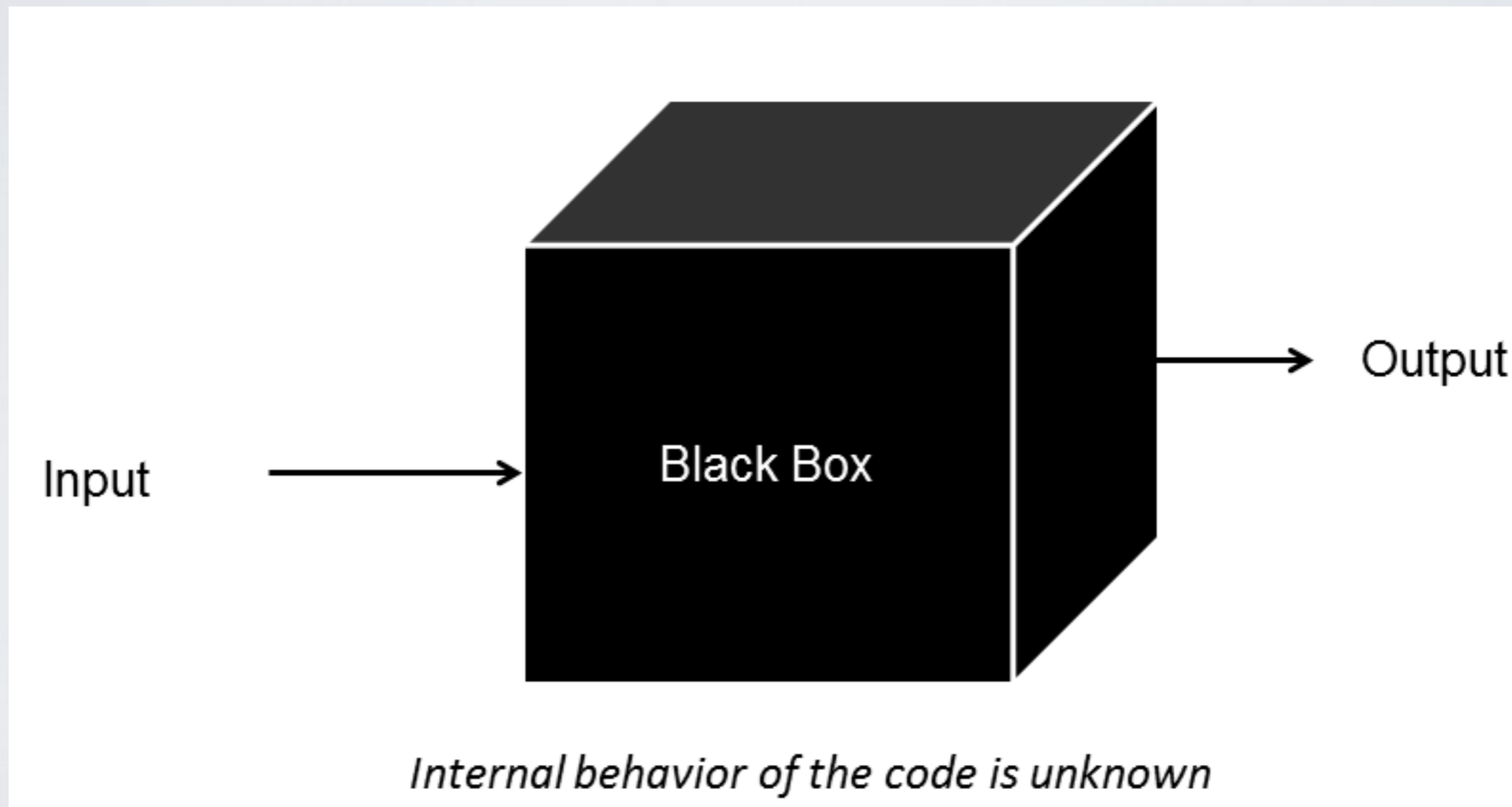


Influence foreign
government elections



Voldemord

STANDARD APPROACH: TESTING



TESTING CAN'T BE COMPLETE

Testing is **essential**, however, it is not **sufficient!**

Suppose `int` is 32 bits

```
int multiply (int x, int y)
```

Thus, there are 2^{64} inputs

Intel Core i7 5960X (8 core)

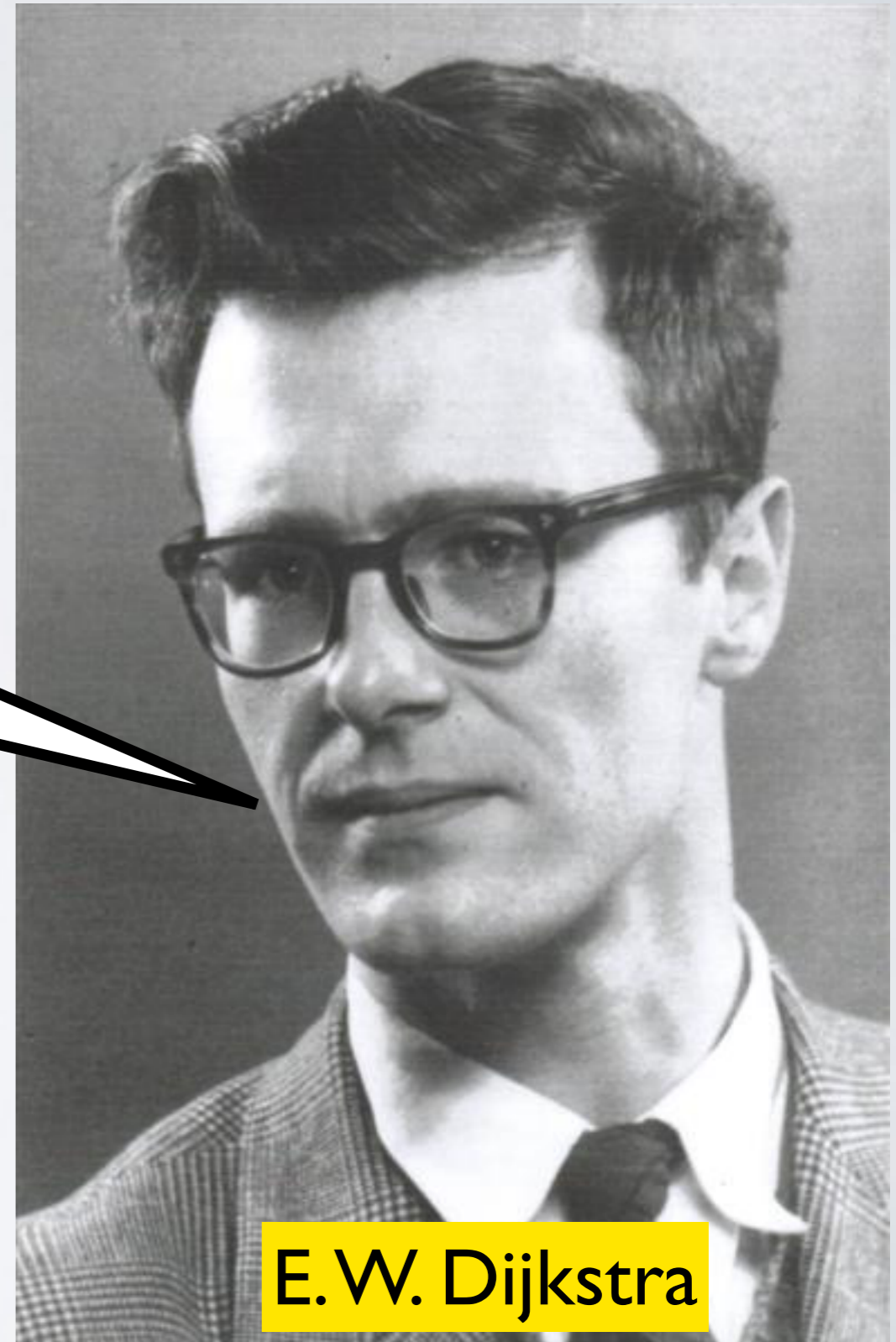
can do about 2^{38} instructions per second

It would take 2^{26} sec \sim 18641 hours \sim **2 years to test**

TESTING

...program testing can be used to show the presence of bugs, but never to show their absence!

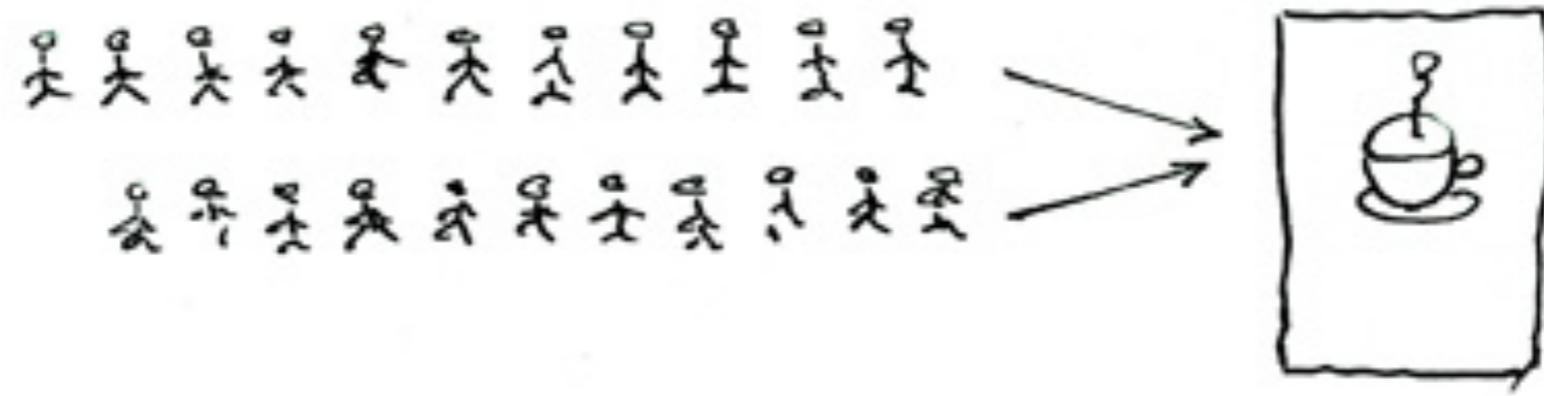
[EWD303]



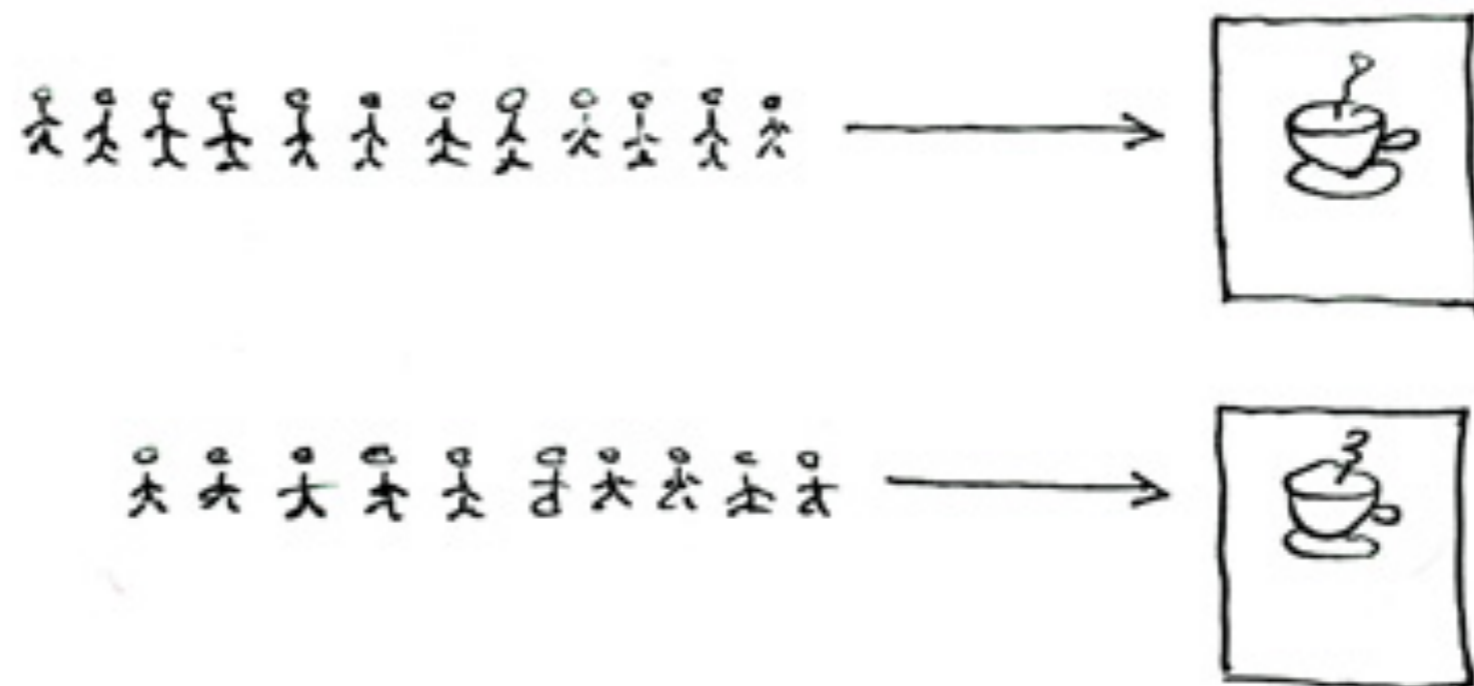
E.W. Dijkstra

CONCURRENT SYSTEMS

Concurrent = Two Queues One Coffee Machine



Parallel = Two Queues Two Coffee Machines



THE CHALLENGE

find

An adequate language for describing concurrent systems

A mathematical theory for capturing dynamics, i.e. semantics

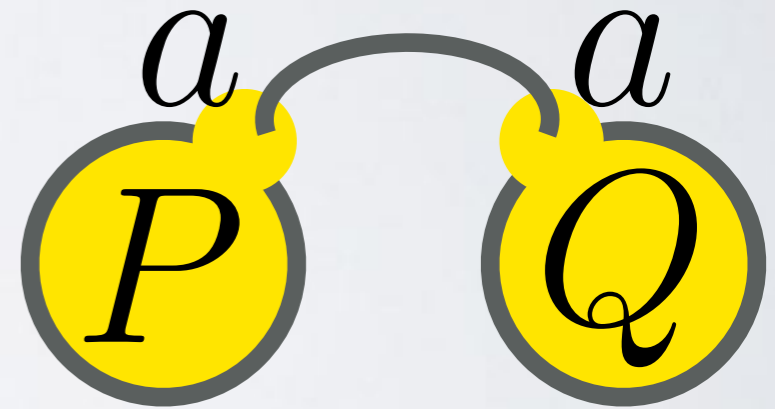
Well-founded Verification Technique

CALCULUS OF COMMUNICATING SYSTEMS [Milner 1980]

a	\in	\mathcal{A}	action
P, Q	$::=$	$a.P$	input
		$\bar{a}.P$	output
		0	inaction
		$\tau.P$	silent
		$P \mid Q$	parallel
		$P + Q$	sum/choice
		$!P$	replication

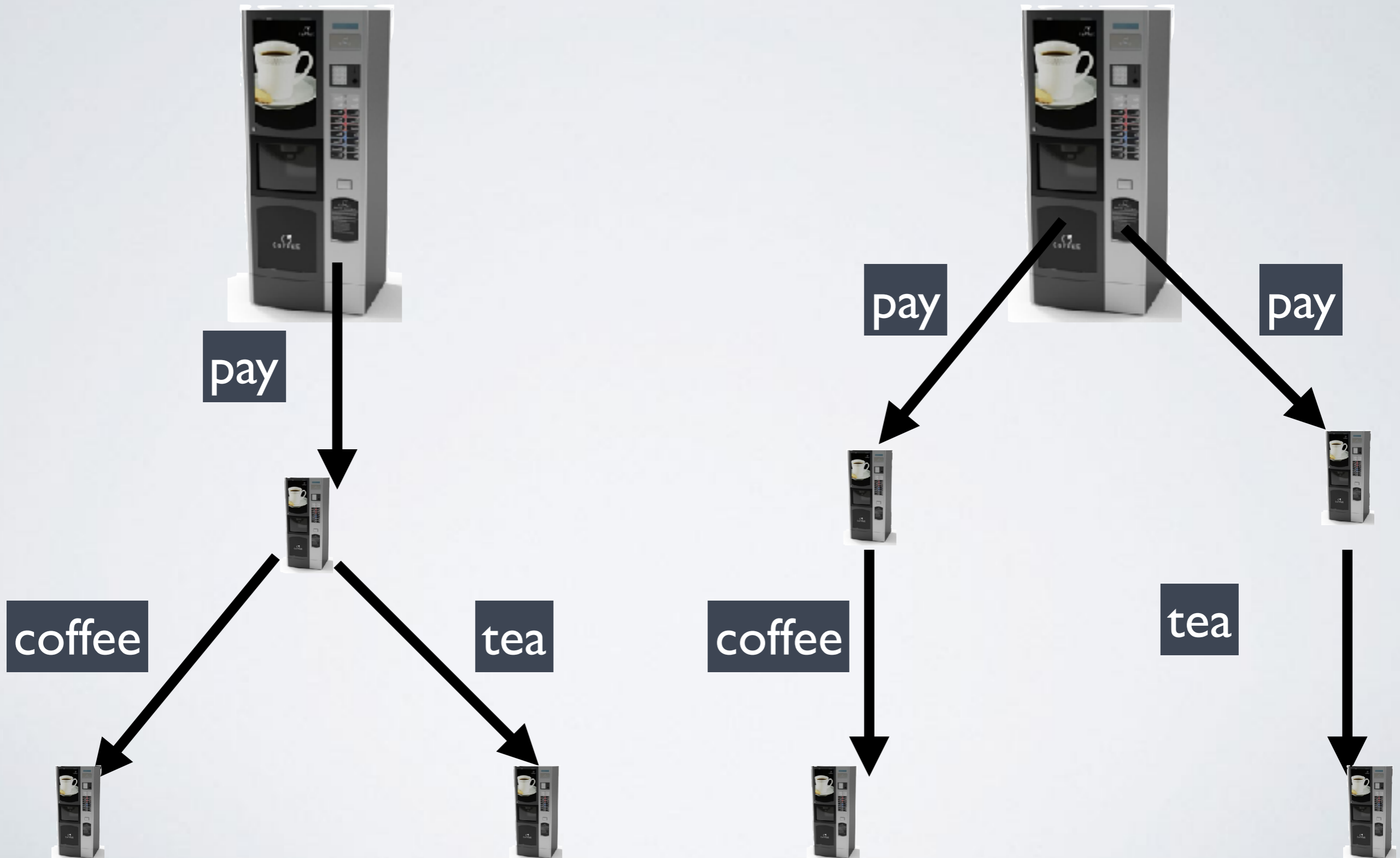
Ex.

$a.P \mid \bar{a}.Q$

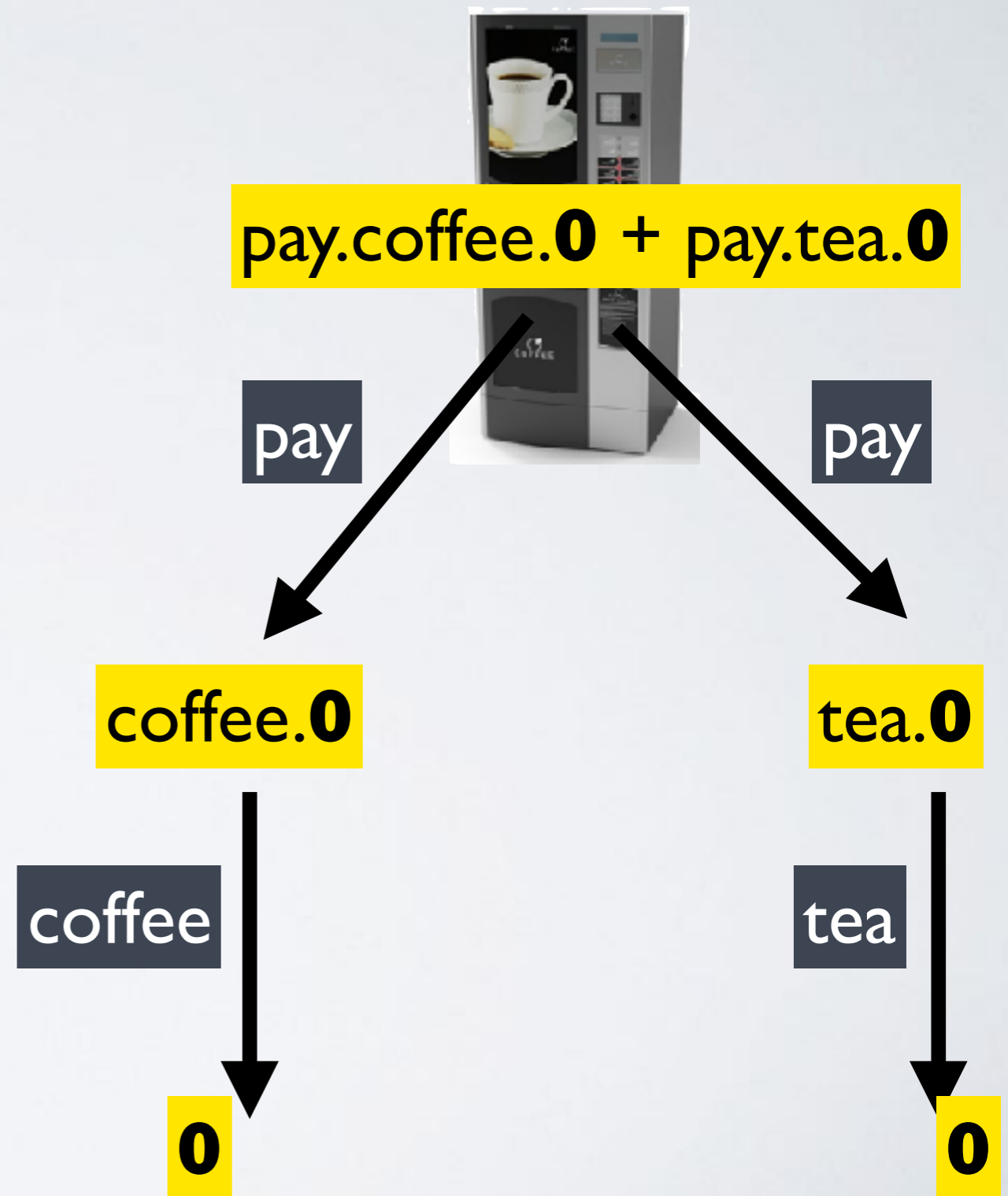
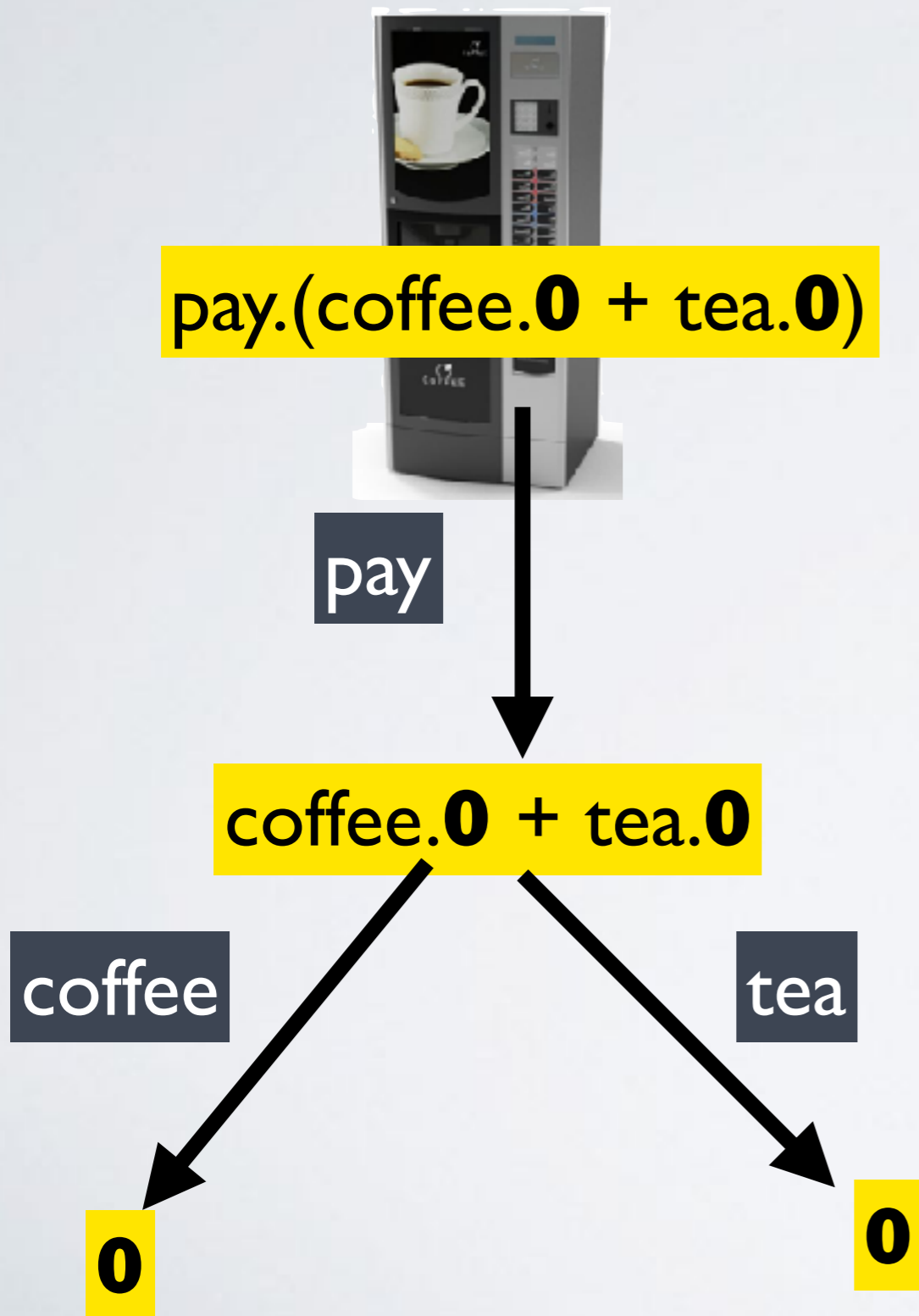


(can be extend with value passing)

OBSERVABLE BEHAVIOUR

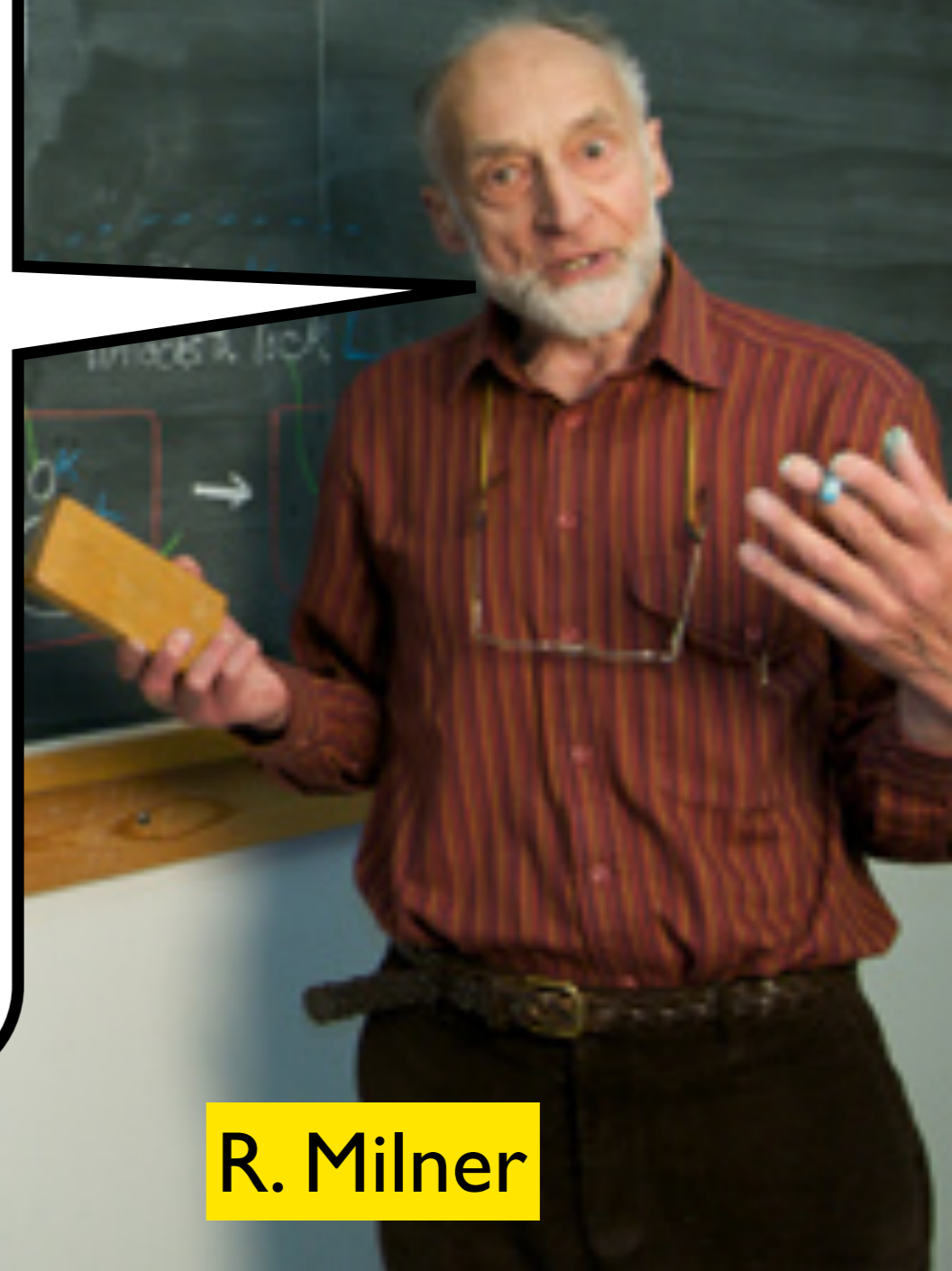


OBSERVABLE BEHAVIOUR



[Milner 1980]

There is nothing canonical about the choice of the basic combinators, even though they were chosen with great attention to economy. What characterises our calculus is not the exact choice of combinators, but rather the choice of interpretation and of mathematical framework.



R. Milner

ALGEBRA OF PROCESSES

Equivalence based on the observable behaviour

Bisimilarity

$P \sim Q$ at each state P can perform all the actions of Q , and vice versa, and states continue to be bisimilar

Alg. properties

$$P|Q \sim Q|P \quad P|(Q|R) \sim (P|Q)|R$$

$$0|P \sim P \quad \text{etc.}$$

$$0 + P \sim P \\ P + Q \sim Q + P$$

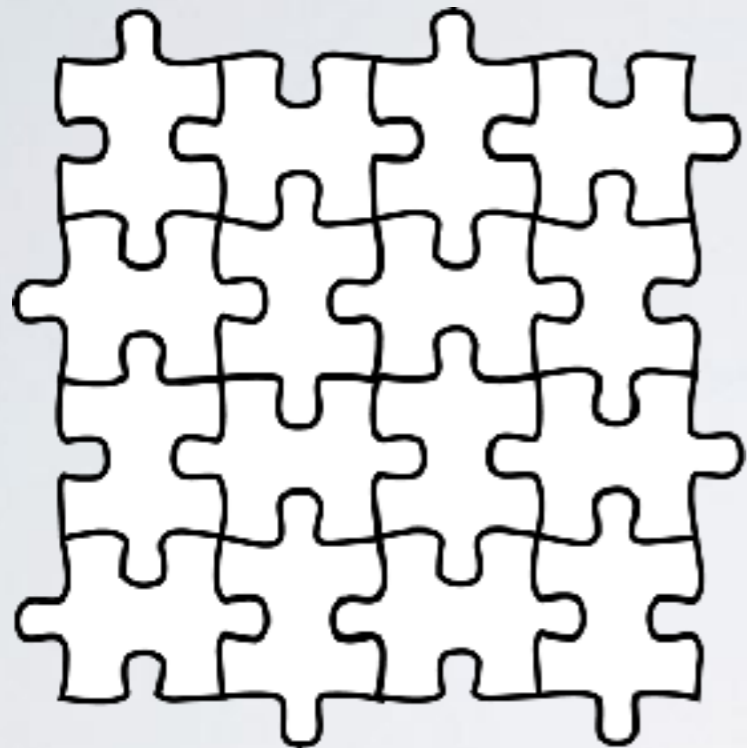
Weak bisimilarity

roughly, ignoring silent actions

$$P \approx \tau.P$$

COMPOSITIONALITY

(Frege's principle)



Systems built from smaller systems

Component Modularity

Under all contexts a processes behaviour is indistinguishable (ie. bisimilar)

A congruence relation

Equivalence (bisimulation)
preserved under all operations

Ex.

if $P \sim Q$
then $R|P \sim R|Q$

VERIFICATION TECHNIQUE

Specification \approx Implementation
(weakly) bisimilar

Specification = $\text{pay}.\text{coffee.0} + \text{tea.0}$

Implementation =

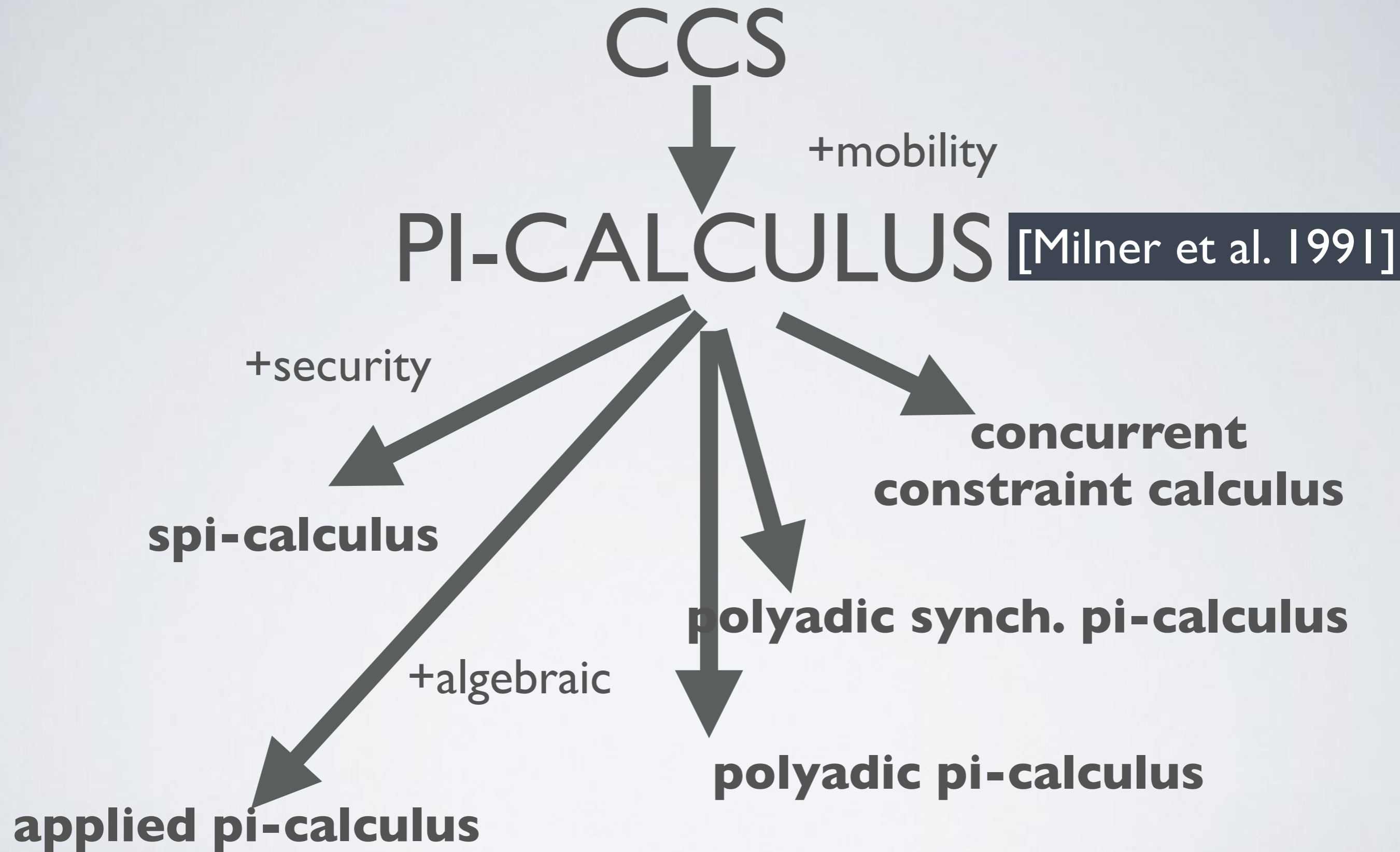
$\text{pay}.\nu\text{internal}(\text{internal}(\text{amount}).$

$\text{if amount} = 50$

$\text{then coffee.0} + \text{tea.0}$

$\text{else coffee.0} + \text{tea.0}$

$| P)$



... and myriad of other 'small' extensions of pi

CCS

+mobility

PI-CALCULUS [Milner et al. 1991]

+security

spi-calculus

your pi-calculus

+algebraic

applied pi-calculus

+ your extension

concurrent pi-calculus

radic synch. pi-calculus

polyadic pi-calculus

... and myriad of other 'small' extensions of pi

Appendix

In this Appendix we outline the proofs of some of the results stated in the text; most of the proofs are by case analysis, and we give the argument for a few crucial or typical cases. Full proofs may be found in [3].

Proof of Lemma 1: The proof is by induction on depth of inference. We consider in turn each transition rule as the last rule applied in the inference of the antecedent $P \xrightarrow{\alpha} P'$. We give two cases.

(INPUT-ACT) Then $\alpha = x(y)$ and $P \equiv x(z).P_1$ with $y \notin \text{fn}(\{z\}P_1)$ and $P' \equiv P_1\{y/z\}$, so (i) holds and (ii) $\text{fn}(P') \subseteq (\text{fn}(P_1) - \{z\}) \cup \{y\} \subseteq \text{fn}(P) \cup \{y\}$.

(CLOSE) Then $\alpha = \tau$ and $P \equiv P_1 \mid P_2$ with $P_1 \xrightarrow{x(y)} P'_1$, $P_2 \xrightarrow{x(y)} P'_2$ and $P' \equiv (y)(P'_1 \mid P'_2)$, so (i) holds, and $\text{fn}(P'_1) \subseteq \text{fn}(P_1) \cup \{y\}$ and $\text{fn}(P'_2) \subseteq \text{fn}(P_2) \cup \{y\}$, so $\text{fn}(P') = (\text{fn}(P'_1) \cup \text{fn}(P'_2)) - \{y\} \subseteq \text{fn}(P)$. \square

Lemmas 2-5 are all similarly proved by induction on depth of inference. Theorem 1 follows easily from the lemmas.

Proof of Lemma 6: Let $\mathcal{S} = \bigcup_{n < \omega} \mathcal{S}_n$ where

$$\begin{aligned} \mathcal{S}_0 &= \sim \\ \mathcal{S}_{n+1} &= \{(P\{w/z\}, Q\{w/z\}) \mid PS_nQ, w \notin \text{fn}(P, Q)\} \end{aligned}$$

We show that \mathcal{S} is a strong bisimulation by showing by induction on n that if PS_nQ then

- if α is a free action and $P \xrightarrow{\alpha} P'$ then for some $Q', Q \xrightarrow{\alpha} Q'$ and $P'SQ'$,
 - if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{x(y)} P'$ then for some $Q', Q \xrightarrow{x(y)} Q'$ and for all $v, P'\{v/y\}SQ'\{v/y\}$,
 - if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and $P'SQ'$.
- If $n = 0$ then 1, 2 and 3 hold since $\mathcal{S}_0 = \sim$.

Suppose $n > 0$ and that $P\sigma S_n Q$ where $PS_{n-1}Q$ and $\sigma = \{w/z\}$ where $w \notin \text{fn}(P, Q)$. We consider only 3.

Suppose that $P\sigma \xrightarrow{\tau} P'$ where $y \notin \text{fn}(P\sigma, Q\sigma)$. Choose $y' \notin \text{fn}(P, Q, w, z)$. Then $P\sigma \xrightarrow{\tau} P'' \equiv P'\{y'/y\}$. Hence by Lemma 4 for some P'' and x' with

33

$P''\sigma \equiv P''$ and $x'\sigma = x$, $P \xrightarrow{x'} P''$. Since $PS_{n-1}Q$ and $y' \notin \text{fn}(P, Q)$ for some $Q'', Q \xrightarrow{x'} Q''$ and $P''SQ''$. Hence $Q\sigma \xrightarrow{x'} Q'' \equiv Q'\sigma$, and so $Q\sigma \xrightarrow{\tau} Q' \equiv Q''\{y'/y\}$. Then

$$\begin{aligned} P' &\equiv P''\{w/z\}\{y'/y\} \\ \mathcal{S} &\equiv Q''\{w/z\}\{y'/y\} \text{ since } y \notin \text{fn}(P''\{w/z\}, Q''\{w/z\}) \\ &\equiv Q' \end{aligned}$$

\square

Proof of Lemma 7: Let $\mathcal{S}^* = \bigcup_{n < \omega} \mathcal{S}_n$ where

$$\begin{aligned} \mathcal{S}_0 &= \mathcal{S} \\ \mathcal{S}_{n+1} &= \{((w)P, (w)Q) \mid PS_nQ, w \in \mathcal{N}\} \end{aligned}$$

The proof involves showing that \mathcal{S}^* is a strong bisimulation. First we note that by induction on n , if PS_nQ and $w \notin \text{fn}(P, Q)$, then $P\{w/z\}S_nQ\{w/z\}$. For $n = 0$ this is immediate from the definition. Suppose $n > 0$ and $(v)PS_n(v)Q$ where $PS_{n-1}Q$ and $w \notin \text{fn}((v)P, (v)Q)$. Then $((v)P)\{w/z\} \equiv (u)P\{w/z\}\{v/y\}$ and $((v)Q)\{w/z\} \equiv (u)Q\{w/z\}\{v/y\}$ where $u \notin \text{fn}((v)P, (v)Q, w)$ and $u\{w/z\} = u$, so $(v)P\{w/z\}S_n(v)Q\{w/z\}$.

Next we show by induction on n that if PS_nQ then

- if α is a free action and $P \xrightarrow{\alpha} P'$ then for some $Q', Q \xrightarrow{\alpha} Q'$ and $P'S^*Q'$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{x(y)} P'$ then for some $Q', Q \xrightarrow{x(y)} Q'$ and for all $v, P'\{v/y\}S^*Q'\{v/y\}$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and $P'S^*Q'$.

For $n = 0$ this is immediate from the fact that \mathcal{S}_0 is a strong bisimulation up to restriction and the definition of \mathcal{S}^* . The remaining details are omitted. \square

Proof of Theorem 2:

- (a) That \sim is both reflexive and symmetric is clear. For transitivity it suffices to show that $\sim \sim$ is a strong bisimulation. The proof uses Lemma 2. We give one case.

Suppose that $y \notin \text{fn}(P, R)$ and $P \xrightarrow{x(y)} P'$. Choose $z \notin \text{fn}(P, Q, R)$. Then $P \xrightarrow{x(z)} P'' \equiv P'\{z/y\}$, so for some $Q', Q \xrightarrow{x(z)} Q'$ and for all $w, P''\{w/z\} \sim Q'\{w/z\}$. Hence for some $R', R \xrightarrow{x(z)} R'$ and for all $w, Q'\{w/z\} \sim R'\{w/z\}$. Then $R \xrightarrow{x(y)} R'' \equiv R'\{y/z\}$ and for all $w, P''\{w/y\} \sim R''\{w/y\}$.

34

(b) For the congruence properties note that:

- $\{(\alpha.P, \alpha.Q) \mid P \sim Q\} \cup \sim$ is a strong bisimulation.
- $\{(P + R, Q + R) \mid P \sim Q\} \cup \sim$ is a strong bisimulation.
- $\{([x=y]P, [x=y]Q) \mid P \sim Q\} \cup \sim$ is a strong bisimulation.
- Let $\mathcal{S} = \{(P|R, Q|R) \mid P \sim Q\}$. It suffices by Lemma 7 to show that \mathcal{S} is a strong bisimulation up to restriction. To see this note first that if $P \sim Q$ and $w \notin \text{fn}(P, Q)$ then by Lemma 6, $P\{w/z\} \sim Q\{w/z\}$ and so $(P|R)\{w/z\}S(Q|R)\{w/z\}$. It is routine to check that the clauses concerning transitions hold. The only rules applicable are PAR, COM and CLOSE.
- It follows from Lemma 6 that \sim is a strong bisimulation up to restriction. Hence by the proof of Lemma 7, if $P \sim Q$ then $(w)P \sim (w)Q$.

(c) Note that $\{(x(y).P, x(y).Q) \mid \text{for all } w \in \text{fn}(P, Q, y), P\{w/y\} \sim Q\{w/y\}\}$ is a strong bisimulation. This follows easily using Lemma 6. \square

Proof of Theorem 8: The proofs of Theorem 8 (a) and Theorem 8 (b) are straightforward. In contrast, the proofs of Theorem 8 (c) and (d) are not short.

Proof of Theorem 8 (c): In the proof we make use of the idea of a *strong bisimulation up to \sim and restriction*. For completeness we introduce first the following concept.

Definition 25 A relation \mathcal{S} is a *strong simulation up to \sim* iff whenever PSQ then

- if α is a free action and $P \xrightarrow{\alpha} P'$ then for some $Q', Q \xrightarrow{\alpha} Q'$ and $P' \sim \mathcal{S} \sim Q'$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{x(y)} P'$ then for some $Q', Q \xrightarrow{x(y)} Q'$ and for all $w, P'\{w/y\} \sim \mathcal{S} \sim Q'\{w/y\}$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and $P' \sim \mathcal{S} \sim Q'$.

\mathcal{S} is a *strong bisimulation up to \sim* iff both \mathcal{S} and \mathcal{S}^{-1} are strong simulations up to \sim .

Lemma 9 If \mathcal{S} is a strong bisimulation up to \sim then $\mathcal{S} \subseteq \sim$.

Proof: Let $\mathcal{S}^* = \bigcup_{n < \omega} \mathcal{S}_n$ where

$$\begin{aligned} \mathcal{S}_0 &= \sim \mathcal{S} \sim \\ \mathcal{S}_{n+1} &= \{(P\{w/z\}, Q\{w/z\}) \mid PS_nQ, w \notin \text{fn}(P, Q)\} \end{aligned}$$

Then by an argument very similar to that in the proof of Lemma 6 it can be shown that \mathcal{S}^* is a strong bisimulation. We omit the details. \square

Combining this concept with that of a strong bisimulation up to restriction we obtain the following.

Definition 26 A relation \mathcal{S} is a *strong simulation up to \sim and restriction* iff whenever PSQ then

- if $w \notin \text{fn}(P, Q)$ then $P\{w/z\}SQ\{w/z\}$,
- if $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and $P' \sim \mathcal{S} \sim Q'$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{x(y)} P'$ then for some $Q', Q \xrightarrow{x(y)} Q'$ and for all $w, P'\{w/y\} \sim \mathcal{S} \sim Q'\{w/y\}$,
- if $y \notin \text{fn}(P, Q)$ and $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and $P' \sim \mathcal{S} \sim Q'$,
- if $P \xrightarrow{\tau} P'$ then for some $Q', Q \xrightarrow{\tau} Q'$ and either $P' \sim \mathcal{S} \sim Q'$ or for some P'', Q'' and $w, P' \sim (w)P'', Q' \sim (w)Q''$ and $P''SQ''$.

\mathcal{S} is a *strong bisimulation up to \sim and restriction* iff both \mathcal{S} and \mathcal{S}^{-1} are strong simulations up to \sim and restriction.

We have the following result.

Lemma 10 If \mathcal{S} is a strong bisimulation up to \sim and restriction then $\mathcal{S} \subseteq \sim$.

Proof: Let $\mathcal{S}^* = \bigcup_{n < \omega} \mathcal{S}_n$ where

$$\begin{aligned} \mathcal{S}_0 &= \sim \mathcal{S} \sim \\ \mathcal{S}_{n+1} &= \sim \{((w)P, (w)Q) \mid PS_nQ, w \in \mathcal{N}\} \sim \end{aligned}$$

Then by an argument similar to that in the proof of Lemma 7 it may be shown that \mathcal{S}^* is a strong bisimulation. We omit the details. \square

Returning to the main proof of Theorem 8 (c), we prove that the relation

$$\mathcal{S} = \{((y)P_1 \mid P_2, (y)(P_1 \mid P_2)) \mid P_1, P_2 \text{ agents, } y \notin \text{fn}(P_2)\} \cup \mathbf{Id}$$

is a strong bisimulation up to \sim and restriction. Thus, for each P and Q such that PSQ and each transition $P \xrightarrow{\alpha} P'$, we must find a “simulating” transition $Q \xrightarrow{\alpha} Q'$ satisfying the requirements of a strong simulation up to restriction and equivalence, and vice versa. Clearly, if $P \equiv Q$ this is trivial, so we assume that $P \equiv (y)P_1 \mid P_2$, $Q \equiv (y)(P_1 \mid P_2)$, and $y \notin \text{fn}(P_2)$.

The proof that there always exists an appropriate transition $Q \equiv (y)(P_1 \mid P_2) \xrightarrow{\alpha} Q'$ is by a case analysis on how the transition $P \equiv (y)P_1 \mid P_2 \xrightarrow{\alpha} P'$ is

derived, and vice versa. There are sixteen cases in all from which we draw a sample of two.

For $\alpha = x(y)$ and $P \equiv x(z).P_1$ with $y \notin \text{fn}(\{z\}P_1)$ and $P' \equiv P_1\{y/z\}$ in the following case:

$$\text{RES: } \frac{}{(y)(P_1 \mid P_2) \xrightarrow{x(y)} P'}$$

We then have to prove three things:

- (\Downarrow): that the premises of the upper derivation imply the premises of the lower derivation;
- (\Uparrow): conversely that the premises of the lower derivation imply the premises of the upper derivation;
- (S): that the derivatives P' and Q' satisfy the requirement of a strong bisimulation up to \sim and restriction.

Note that by the definition of strong simulation we only have to consider α such that $y \notin \text{bn}(\alpha)$, since y occurs in the agents P and Q .

Case :

$$\text{RES: } \frac{P_1 \xrightarrow{x(z)} P'_1 \quad x, z \neq y}{(y)P_1 \xrightarrow{x(z)} (y)P'_1} \quad P_2 \xrightarrow{\tau} P'_2$$

$$\text{COM: } \frac{}{(y)P_1 \mid P_2 \xrightarrow{\tau} ((y)P'_1\{v/z\}) \mid P'_2}$$

\Downarrow

$$\text{COM: } \frac{P_1 \xrightarrow{x(z)} P'_1 \quad P_2 \xrightarrow{\tau} P'_2}{P_1 \mid P_2 \xrightarrow{\tau} P'_1\{v/z\} \mid P'_2}$$

$$\text{RES: } \frac{}{(y)(P_1 \mid P_2) \xrightarrow{x(z)} (y)(P'_1\{v/z\}) \mid P'_2}$$

37

(\Downarrow): Trivial.

(\Uparrow): From $y \notin \text{fn}(P_2)$ and Lemma 1 we get that $x \neq y$. We cannot prove that $z \neq y$, but if $z = y$ then we use a fresh z' instead of z to get a simulating transition as follows: from Lemma 2 we get that $P_1 \xrightarrow{x(z')} P'_1\{z'/y\}$. The simulating transition then is:

$$(y)P_1 \mid P_2 \xrightarrow{x(z')} ((y)P'_1\{z'/y\})\{v/z'\} \mid P'_2 \quad (*)$$

(S): From $v, z \neq y$ it follows that $((y)P'_1)\{v/z\} \equiv (y)P'_1\{v/z\}$, and Lemma 1 with $y \notin \text{fn}(P_2)$ gives that $y \notin \text{fn}(P'_2)$, so

$$((y)P'_1)\{v/z\} \mid P'_2 \mathcal{S} (y)(P'_1\{v/z\} \mid P'_2)$$

as required. For the simulating transition (*) we know that $z = y$, so it holds (since $v \neq y$ and z' is chosen fresh) that

$$((y)P_1\{z'/y\})\{v/z'\} \mid P'_2 \equiv (y)P'_1\{v/y\} \mid P'_2 \mathcal{S} (y)(P'_1\{v/y\} \mid P'_2)$$

Case :

$$\text{RES: } \frac{P_1 \xrightarrow{\tau} P'_1 \quad x, v \neq y}{(y)P_1 \xrightarrow{\tau} (y)P'_1} \quad P_2 \xrightarrow{x(z)} P'_2$$

$$\text{COM: } \frac{}{(y)P_1 \mid P_2 \xrightarrow{\tau} (y)P'_1 \mid P'_2\{v/z\}}$$

- (\Downarrow): Trivial.
- (\Uparrow): From Lemma 1 and $y \notin \text{fn}(P_2)$ we get $x \neq y$. The situation when $v = y$ is treated in another case (see [3]).
- (S): From Lemma 1 and $y \notin \text{fn}(P_2)$ we get that $y = z$ or $y \notin \text{fn}(P'_2)$, so from $(y)P_1 \mid P_2 \xrightarrow{\tau} (y)P'_1 \mid P'_2\{v/z\}$ it follows $y \notin \text{fn}(P'_2\{v/z\})$. This proves as required

$$((y)P_1 \mid P_2) \xrightarrow{\tau} ((y)P'_1 \mid P'_2\{v/z\})$$

Proof of Theorem 8 (d): The proof involves showing that the relation

[Milner et al. ECS-LFCS-89-86]

The proof that there always exists an appropriate transition $Q \xrightarrow{\alpha} Q'$ is by a case analysis on how the transition $P \xrightarrow{\alpha} P'$ is derived, and vice versa. There are 30 cases in total. We present one sample case in the same style as in the proof of Theorem 8 (c).

Case :

$$\text{CLOSE: } \frac{P_1 \xrightarrow{x(z)} P'_1 \quad P_2 \xrightarrow{x(z)} P'_2}{P_1 \mid P_2 \xrightarrow{x(z)} (z)(P'_1 \mid P'_2)}$$

$$\text{PAR: } \frac{}{(P_1 \mid P_2) \mid P_3 \xrightarrow{x(z)} (z)(P'_1 \mid P'_2) \mid P_3}$$

\Downarrow

$$\text{PAR: } \frac{P_2 \xrightarrow{x(z')} P'_2\{z'/z\} \quad z' \notin \text{fn}(P_3)}{P_2 \mid P_3 \xrightarrow{x(z')} P'_2\{z'/z\} \mid P_3} \quad P_1 \xrightarrow{x(z)} P'_1\{z'/z\}$$

$$\text{CLOSE: } \frac{}{P_1 \mid (P_2 \mid P_3) \xrightarrow{x(z')} (z')(P'_1\{z'/z\} \mid (P'_2\{z'/z\} \mid P_3))}$$

(\Downarrow): By Lemma 2 there exists a fresh z' such that $P_1 \xrightarrow{x(z')} P'_1\{z'/z\}$ and $P_2 \xrightarrow{x(z')} P'_2\{z'/z\}$.

(S): Note that z' is a fresh name. By alpha-converting z to z' and then applying Theorem 8 (c) we get that

$$(z)(P'_1 \mid P'_2) \mid P_3 \equiv (z')(P'_1\{z'/z\} \mid P'_2\{z'/z\}) \mid P_3 \sim (z')((P'_1\{z'/z\} \mid P'_2\{z'/z\}) \mid P_3)$$

so the condition for a simulation up to \sim and restriction is satisfied:

$$(P'_1\{z'/z\} \mid P'_2\{z'/z\}) \mid P_3 \mathcal{S} P'_1\{z'/z\} \mid (P'_2\{z'/z\} \mid P_3)$$

\square

39

Proof of Theorem 17: We first state some immediate consequences of the definition of replacement. If E is an agent expression and σ a substitution of names, then $E\sigma$ is defined to be the agent expression obtained in the way analogous to Definition 3. Then substitutions of names as expected commute with replacements in the following way: $E(A_1, \dots, A_n)\sigma \equiv E\sigma(A_1, \dots, A_n)$. Also, since replacement clearly distributes over the operators we have that Theorem 2 generalizes to agent expressions. These facts will be used freely in what follows.

We will only prove the theorem for $I = \{1\}$. The proof of the general case is similar and only notationally more cumbersome. We write E, F, A, B, X, \tilde{x} for $E_1, F_1, A_1, B_1, X_1, \tilde{x}_1$. Assuming the premises of the theorem, define the relation \mathcal{S} by

$$\mathcal{S} = \{(G(A), G(B)) : G \text{ has only the schematic identifier } X\}$$

We show that \mathcal{S} is a strong bisimulation up to \sim . By Lemma 9 it follows that $\mathcal{S} \subseteq \sim$. By choosing $G \equiv X(\tilde{y})$ we then get that $A(\tilde{y}) \sim B(\tilde{y})$; since this holds for any names \tilde{y} it implies that $A(\tilde{x})\sigma \sim B(\tilde{x})\sigma$ for any σ , which amounts to $A(\tilde{x}) \sim B(\tilde{x})$.

To prove \mathcal{S} a strong bisimulation up to \sim it is clearly enough to prove the following properties, which we will call (*):

- If $G(A) \xrightarrow{\alpha} P'$ and α is a free action or bound output action with $\text{bn}(\alpha) \cap \text{n}(G(A), G(B)) = \emptyset$, then $G(B) \xrightarrow{\alpha} Q''$ with $P'S \sim Q''$.
- If $G(A) \xrightarrow{x(y)} P'$ and $y \notin \text{n}(G(A), G(B))$ then $G(B) \xrightarrow{x(y)} Q''$ such that for all $u, P'\{u/y\}S \sim Q''\{u/y\}$.

So assume $G(A) \xrightarrow{\alpha} P'$; we will prove (*) by induction on the depth of the inference of this transition. We argue by cases on how the last step in this transition is inferred. We give two sample cases.

10 pages of proof appendix + 30 main text and proofs

$G(B) \equiv B(\tilde{y}) \xrightarrow{\alpha} Q''$. Since \sim is transitive, $P'S \sim Q''$ as required. Consider next the subsubcase where $\alpha = x(y)$ is an input action. We only have to consider $y \notin \text{n}(G(A), G(B))$. By definition, then $y \notin \text{n}(E\{y/\tilde{x}\}(A), E\{y/\tilde{x}\}(B))$.

40

(\Downarrow): Trivial.

(\Uparrow): From $y \notin \text{fn}(P_2)$ and Lemma 1 we get that $x \neq y$. We cannot prove that $z \neq y$, but if $z = y$ then we use a fresh z' instead of z to get a simulating transition as follows: from Lemma 2 we get that $P_1 \xrightarrow{x(z')} P_1\{z'/y\}$. The simulating transition then is:

$$(y)P_1 \mid P_2 \xrightarrow{\tau} ((y)P_1\{z'/y\})\{v/z'\} \mid P_2' \quad (*)$$

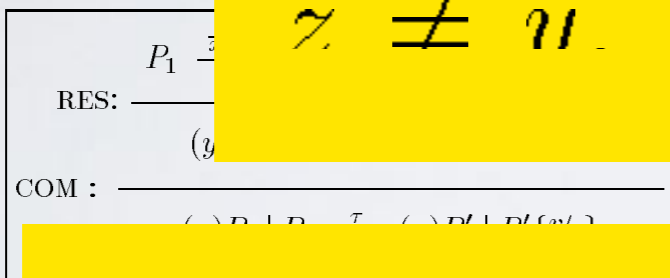
(\mathcal{S}): From $v, z \neq y$ it follows that $((y)P_1')\{v/z\} \equiv (y)P_1'\{v/z\}$, and Lemma 1 with $y \notin \text{fn}(P_2)$ gives that $y \notin \text{fn}(P_2')$, so

$$((y)P_1')\{v/z\}$$

as required. For the simulation (since $v \neq y$ and z' is chosen

$$((y)P_1\{z'/y\})\{v/z'\} \mid P_2'$$

Case :



(\Downarrow): Trivial.

(\Uparrow): From $y \notin \text{fn}(P_2)$ and Lemma 1 we get $z \neq y$, but if $z = y$ then we use a fresh

With cheats!

In this Appendix we outline the proofs of some of the results stated in the text; most of the proofs are by case analysis, and we give the argument for a few crucial or typical cases. Full proofs may be found in [3].

(\Downarrow): Trivial.

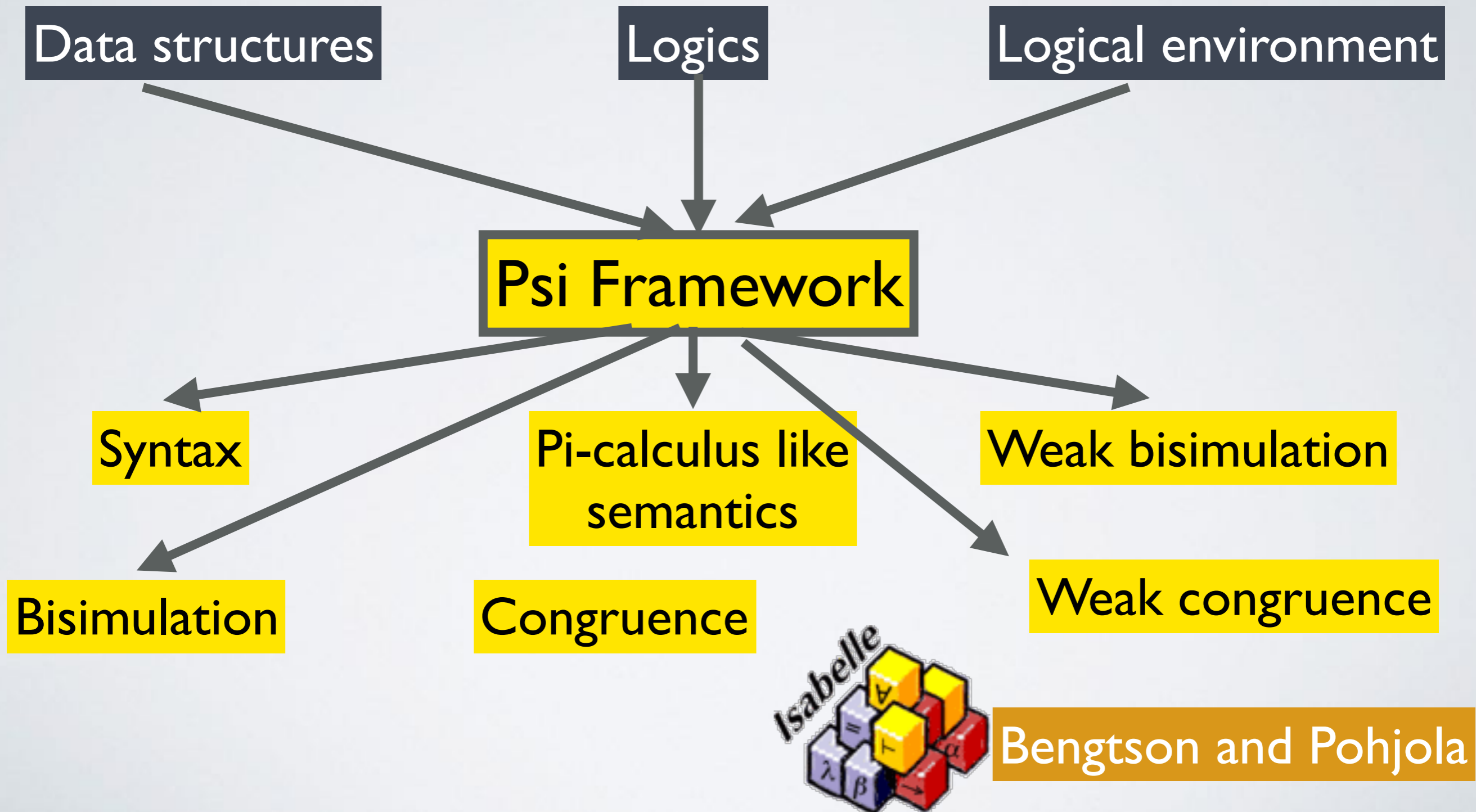
(\Uparrow): From Lemma 1 and $y \notin \text{fn}(P_2)$ we get that $y = z$ or $y \notin \text{fn}(P_2')$, so from $v \neq y$ it follows $y \notin \text{fn}(P_2'\{v/z\})$. This proves as required

(\mathcal{S}): From Lemma 1 and $y \notin \text{fn}(P_2)$ we get that $y = z$ or $y \notin \text{fn}(P_2')$, so from $v \neq y$ it follows $y \notin \text{fn}(P_2'\{v/z\})$. This proves as required

$$(y)P_1' \mid P_2'\{v/z\} \xrightarrow{\mathcal{S}} (y)(P_1' \mid P_2'\{v/z\})$$

[3] is self-reference.

PSI-CALCULUS FRAMEWORK



Parameters

x	\in	\mathcal{N}	name
M, N	\in	\mathbf{T}	term
$\varphi_1, \dots, \varphi_n$	\in	\mathbf{C}	condition
Ψ	\in	\mathbf{A}	assertion
\leftrightarrow	\in	$\mathbf{T} \times \mathbf{T} \rightarrow \mathbf{C}$	channel equivalence
\otimes	\in	$\mathbf{A} \times \mathbf{A} \rightarrow \mathbf{A}$	assertion composition
\vdash	\subseteq	$\mathbf{A} \times \mathbf{C}$	entailment
$[\tilde{x} := \tilde{N}]$	\in	$\mathbf{T} \rightarrow \mathbf{T}$	substitution function

P, Q	$::=$	$M(\lambda\tilde{x})N.P$	input
		$\overline{M}N.P$	output
		$P \mid Q$	parallel
		$!P$	replication
		$(\nu x)P$	restriction
		(Ψ)	assertion
		case $\varphi_1 : P_1 \square \dots \square \varphi_n : P_n$	case
		$\mathbf{0}$	inaction

Syntax

MAC

$(\nu \text{ secret})($
 $(\parallel \text{hash}(\langle \text{secret}, \text{message} \rangle) = x \parallel \mid$
 $\bar{a} \langle \text{message}, x \rangle \mid$
 $a(y).$
 case $\text{hash}(\langle \text{secret}, \text{fst}(y) \rangle) = \text{snd}(y) \quad : \bar{b} \text{ YES}$
 $\parallel \text{hash}(\langle \text{secret}, \text{fst}(y) \rangle) \neq \text{snd}(y) \quad : \bar{b} \text{ NO}$
 $)$

generate a key
sign a message
send MAC
receive MAC
Verify



UPPSALA
UNIVERSITET

Languages, Logics, Types and Tools for Concurrent System Modelling

RAMŪNAS GUTKOVAS



Expanding generality of Psi-calculi
with a type-system

Providing a verification calculus for
psi-calculus, and others

Tool support for psi-calculi

SORT SYSTEM FOR PSI

REPRESENTATION

A direct encoding of a process calculus to a Psi-calculus

No elaborate encodings

No superfluous data terms

No superfluous behaviour

Many calculi were not representable

Unsorted polyadic pi-calculus Sorted polyadic pi-calculus

LINDA pattern matching Polyadic synchronisation pi-calculus

Value-passing CCS

Goal: extend psi-calculi to be capable of representing new calculi!

SYMMETRIC CRYPTO

Computation

$$\text{dec}(\text{enc}(M, K), K) \rightarrow M$$

makes sense when it is typed

$$\begin{aligned} &(\nu a, k)(\bar{a} \text{ "foobar" } . \mathbf{0} \quad | \quad \underline{a}(\lambda y)y. \bar{c} \text{ dec}(y, k). \mathbf{0}) \\ &\xrightarrow{\tau} (\nu a, k)(\mathbf{0} \quad | \quad \bar{c} \text{ dec}(\text{"foobar"}, k). \mathbf{0}) \end{aligned}$$

$$\begin{aligned} &(\nu a, k)(\bar{a} \text{ enc}(M, k). \mathbf{0} \quad | \quad \underline{a}(\lambda y)y. \bar{c} \text{ dec}(y, k). \mathbf{0}) \\ &\xrightarrow{\tau} (\nu a, k)(\mathbf{0} \quad | \quad \bar{c}M) \end{aligned}$$

SORT SYSTEM

Set of sorts

\mathcal{S}

Sort assigning to params function

$\text{SORT}(X) \in \mathcal{S}$

Sorting relations for substitution and processes:

can send

can receive

can restrict

can substituted

Consider only well sorted substitutions

Sanity check: A well-sorted substitution preserves well-sortedness of a process.

RESULTS

All the standard algebraic laws of bisimulation are preserved

Weak bisimulation

Weak congruence

Bisimulation

Congruence

All the mentioned calculi are **directly representable**

Unsorted polyadic pi-calculus LINDA pattern matching

Sorted polyadic pi-calculus Polyadic synchronisation pi-calculus

Value-passing CCS

MODAL LOGICS FOR PSI

MODAL LOGICS

Find grained properties of a system

== Process

Deadlock freedom

Eventually coffee machine produces **coffee**

== Modal
Logic
Formula

A malicious message is eventually rejected

Process

is a model

Modal logical formula

$$P \models \varphi$$

formula φ is true for P

MODAL LOGICS

Concurrent System Models

CCS

Value-Passing CCS

Spi-calculus

Applied pi-calculus

Fusion calculus

Multi-labelled Nominal transition systems

Psi-calculi framework

Concurrent constraint calc.

Possibly others

Logics

Hennesy, Milner 1985

Hennesy, Liu 1995

Frendrup et al.2002

Hüttel, Pedersen et al. 2007

Haugstad et al. 2006

De Nicola, Loreti 2008

???

???

NOMINAL MODAL LOGIC

Formulas depend on finite number of names

$$P \models \varphi \quad \mathbf{iff} \quad P \vdash \varphi$$

$$P \models \neg A \quad \mathbf{iff} \quad \mathbf{not} \ P \models A$$

$$P \models \bigwedge_{i \in I} A_i \quad \mathbf{iff} \quad (\forall i \in I) \ P \models A_i$$

$$P \models \langle \alpha \rangle A \quad \mathbf{iff} \quad (\exists P') \ P \xrightarrow{\alpha} P', \ P' \models A$$

Thm. Adequate for strong bisimilarity.

What's new: finitely supported formulas

EXPRESSIVENESS

Next step

for any action there is a state

Quantifiers

for every value of a domain

Fresh/New

for a state where a name does not appear

Recursion in Logic

$\text{rec } X.A$

Ex.

Eventually get **coffee** :=
 $\text{rec } X. \langle \text{coffee} \rangle \text{true} \vee \text{next step, recurse on } X$

RESULTS

Adequate Modal Logic for many transition systems

The main proofs are machine checked



Adequate for many variants of bisimilarity:
hyper, open, early, late, weak

Provide an adequate modal logic for

**psi-calculi, concurrent constraint calculus,
and others**

TOOL SUPPORT

AUTOMATED TOOLS

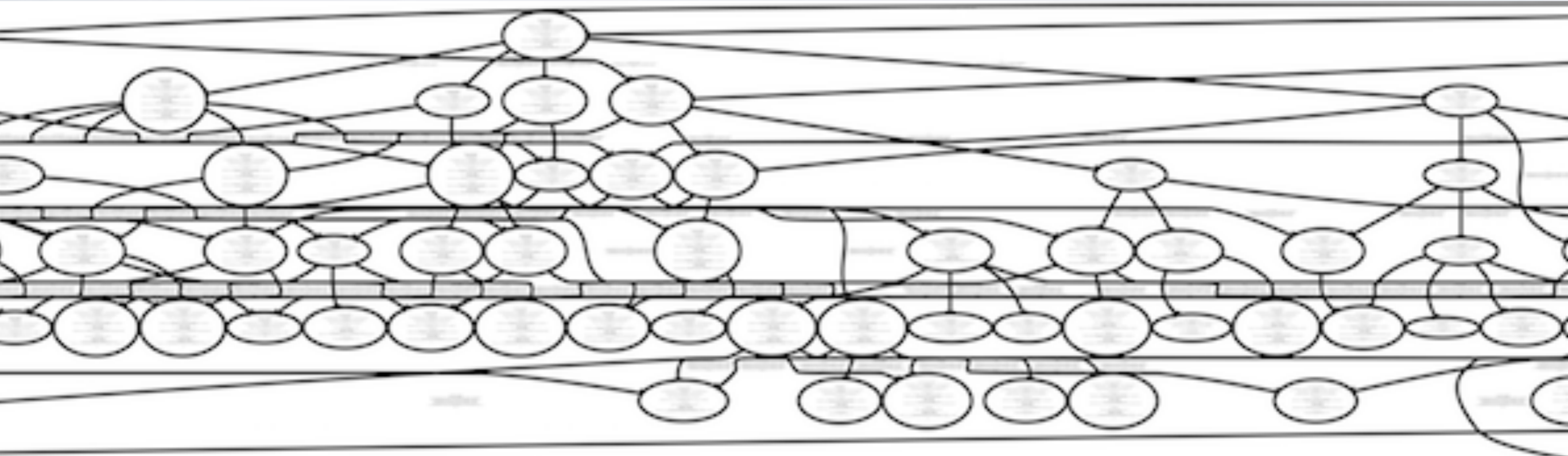
Small specification:

WSN secure aggr.
Small spec. in Pwb
20 LOC
only 3 nodes

Property

There is no tempered data
that the network accepts

Results in



PSI-CALCULI WORKBENCH

Tool factory: define your own tool!



Based on the parametric psi-calculi framework

PARAMETRIC

Data Structures

e.g., Names, Bits, Vectors,
ADTs, Trees, ...

Logics

e.g., EUF, FOL, Equational
Theory, ...

Logical Assertions

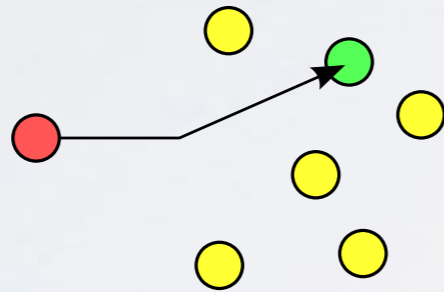
e.g., Knows a secret,
Connectivity, ...

FEATURES

Communication
Primitives

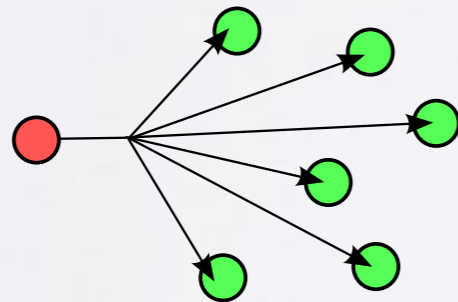
Execution of Processes

Unicast



(Weak) Bisimulation
Checking

Unreliable
Broadcast



Pluggable
Architecture

[Borgström et
al. 2011]

EXAMPLE: WSN AGGREGATION

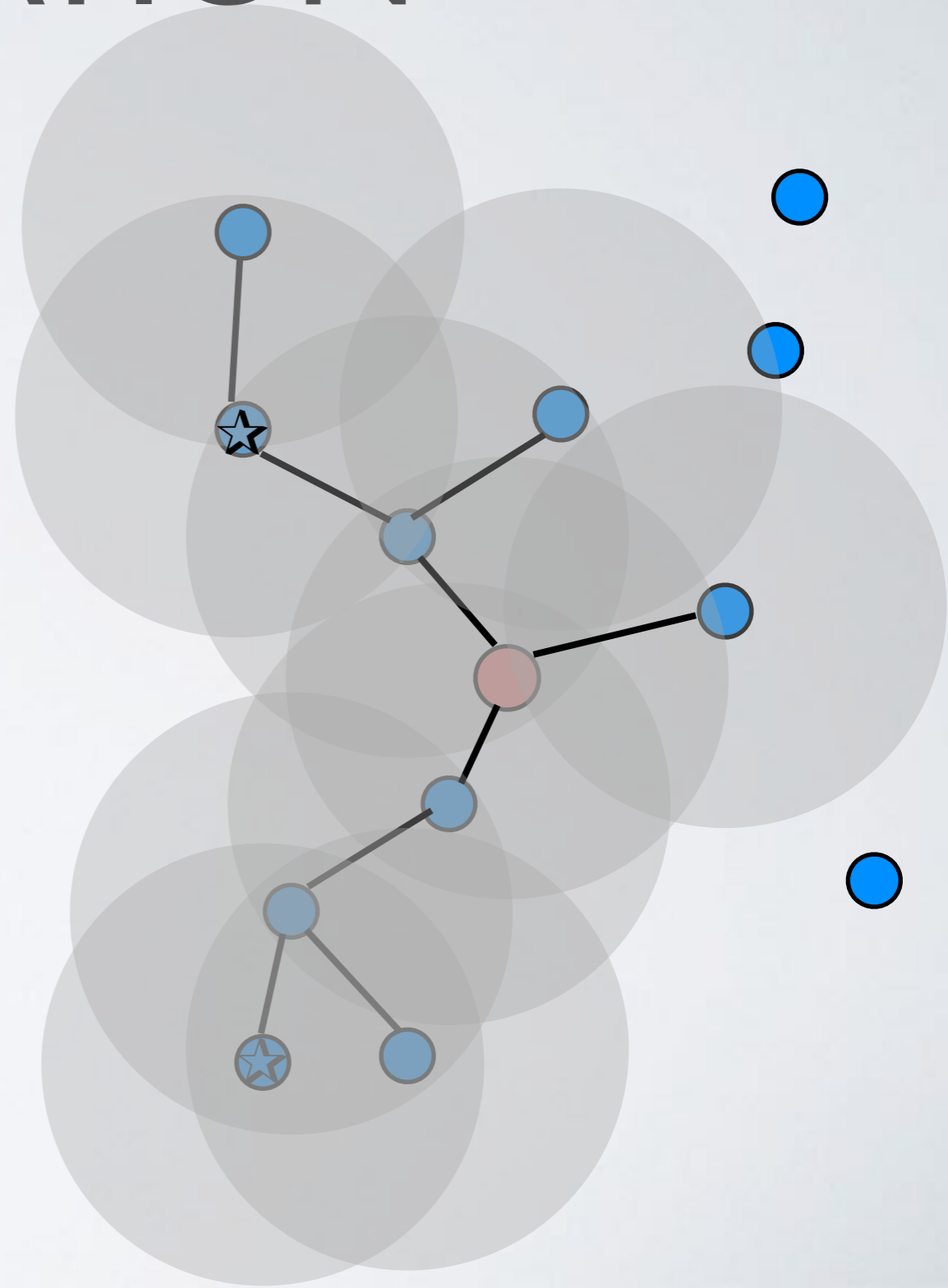
Spatially distr. nodes

Wireless communication

Protocol:

Establish routing tree

Forward data



WORKBENCH MODEL

```
Sink (nodeId , bsChan) <=  
  ' "init (nodeId)" !<bsChan> .  
  ! "data (bsChan)" (x) ;
```

```
Node (nodeId , nodeChan , datum) <=  
  "init (nodeId)" ?(pChan) .  
  ' "init (nodeId)" !<nodeChan> .  
  ' "data (pChan)" <datum> .  
  NodeForwardData<nodeChan , pChan> ;
```

```
NodeForwardData (nodeChan , pChan) <=  
  ! "data (nodeChan)" (x) . ' "data (pChan)" <x> ;
```

SYMBOLIC EXECUTION

generated action

```
--|gna!(new bsChan)bsChan|-->
```

Source:

```
System3<d1, d2>
```

system with 3 nodes

Constraint:

```
(new chan1, chan2, chanS) { | "init(0)<gna" | } ^  
(new chanS, chan2, chan1) { | "gna>init(1)" | } ^  
(new chanS, chan1, chan2) { | "gna>init(2)" | }
```

constraints

Solution:

```
([gna := "init(0)"], 1)
```

solution

Derivative:

```
(!("data(chanS)"(x))) |  
  ((new chan1)(  
    "'init(1)'"!<chan1>.  
    "'data(chanS)'"<d1>.  
    NodeForwardData<chan1, chanS>  
  )) |  
  ((new chan2)(  
    "'init(2)'"!<chan2>.  
    "'data(chanS)'"<d2>.  
    NodeForwardData<chan2, chanS>  
  )))
```

**Execution:
derived
process**

ARCHITECTURE

Pwb

Command Interpreter

Symbolic Equivalence gen.

Symbolic Execution

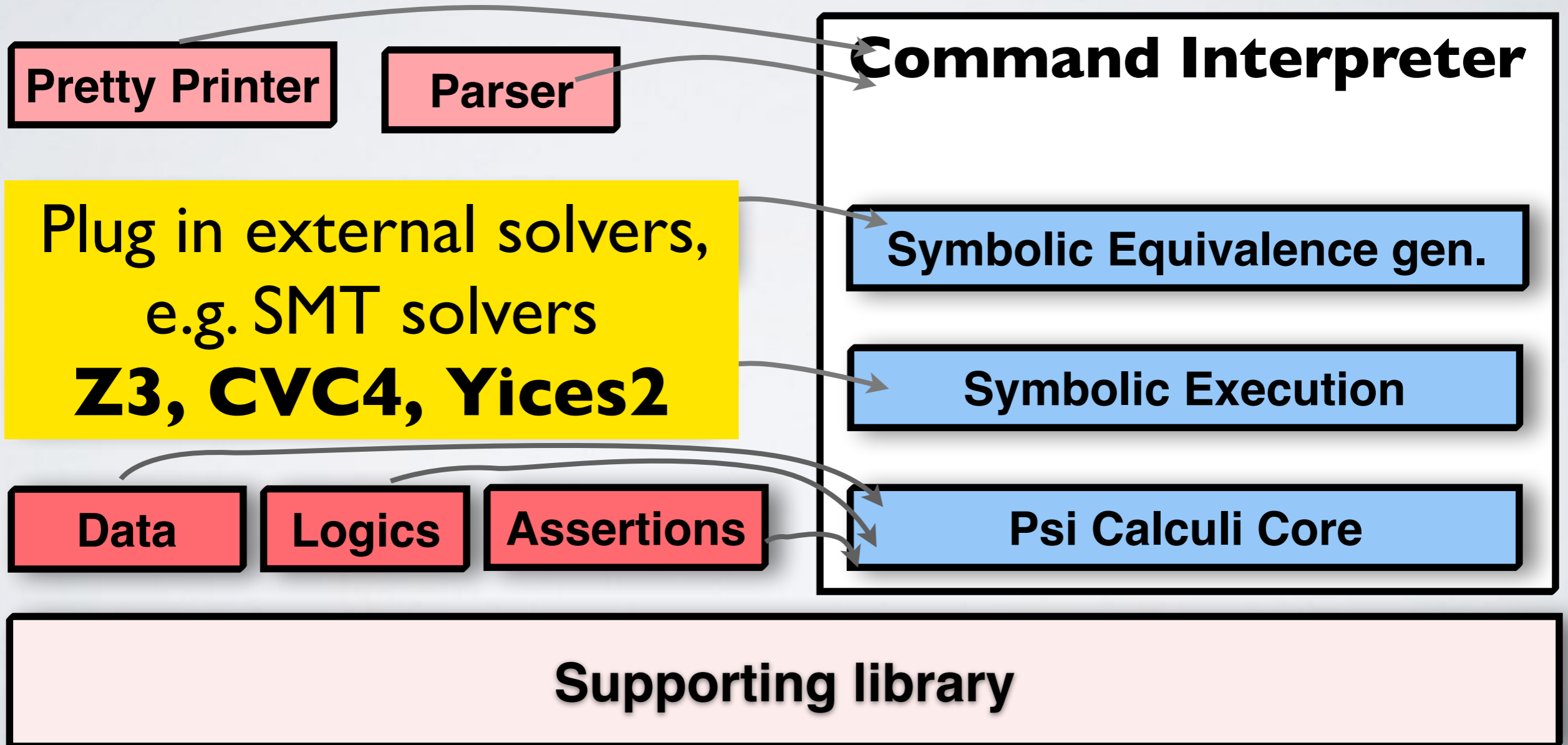
Psi Calculi Core

Supporting library

ARCHITECTURE

Parameters

Pwb



CONCLUSION

A widened applicability of psi-calculi via a type system

A general and powerful modal logic that is applicable to systems such as psi-calculi

Tool support for psi

QUESTIONS