“Observing States”

Nadine Karsten
Uwe Nestmann

OPCT
Bertinoro
2023-06-27

Technische Universität Berlin

Modelle und Theorie Verteilter Systeme

Motivation

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

"pseudo" code
incomplete
informal
...
shows "code" for just one participant
...
underlying communication mechanism by textual explanation

"pseudo" proofs !
handwaving
"intuitive"

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

A Fault Tolerance Bisimulation Proof For Consensus (Extended Abstract)

Adrian Francalanza[1] and Matthew Hennessy[2]

[1] Imperial College, London SW7 2BZ, England, adrianf@doc.ic.ac.uk
[2] University of Sussex, Brighton BN1 9RH, England, matthewh@sussex.ac.uk

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

Validity

every ***decision*** *value* ***must have been proposed*** *by one of them*

Agreement

no two *(correct) processes* ***decide differently***

Termination

every *correct process* ***eventually decides***

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

For this algorithm, Agreement requires *sufficiently reliable failure detection*.

Weak Accuracy (Chandra/Toueg):
Some correct process will never be suspected.

Distributed Consensus

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

For this algorithm, Agreement requires *sufficiently reliable failure detection*.

Weak Accuracy (Chandra/Toueg):
Some correct process will never be suspected.

Fuzzati/N. call this process "trusted immortal". Let ti refer to it.

Informal Correctness Argument

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

Before round ti, "anything goes".

In such a round, any process may receive the value proposed by any coordinator of the rounds until then. Or not. No guarantees …

Informal Correctness Argument

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

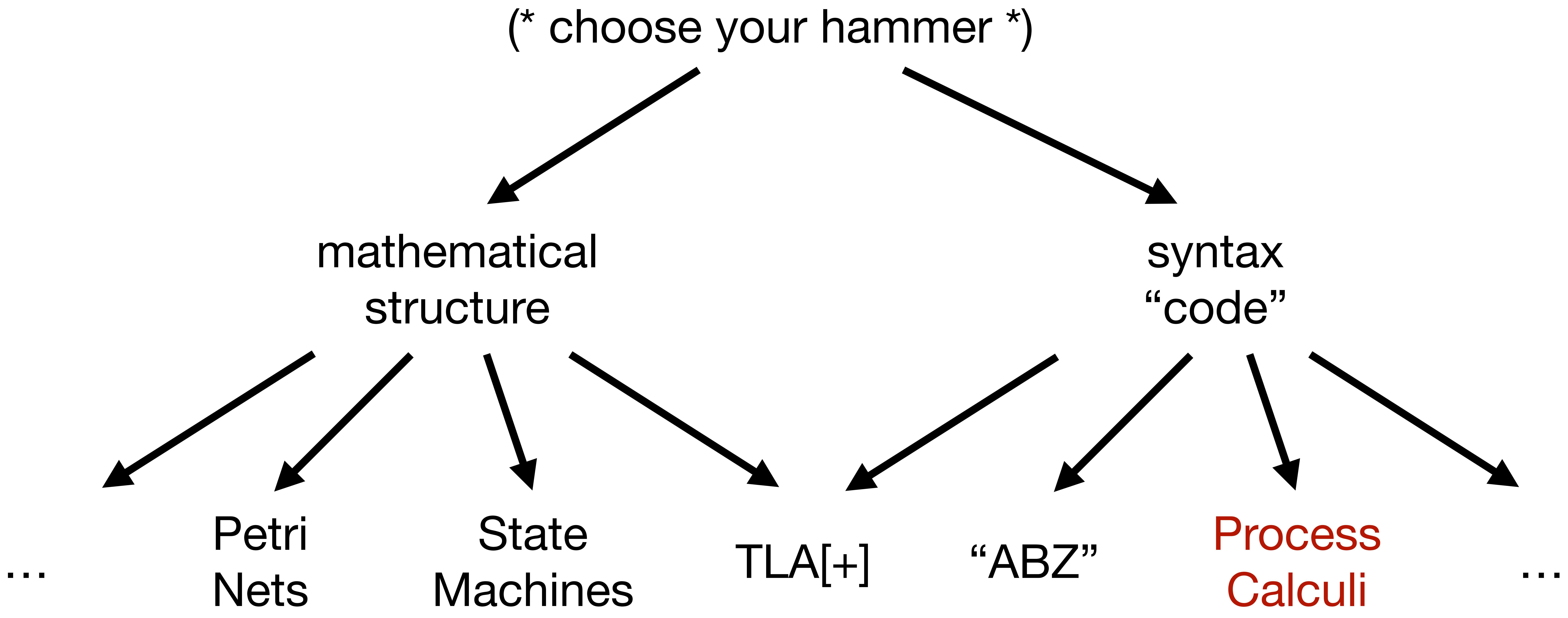
```
x_i := input;
for r := 1 to n do {  if r = i then broadcast x_i;
                      if alive(p_r) then x_i := input_from_broadcast };
output x_i;
```

Before round ti, “anything goes”.

In such a round, any process may receive the value proposed by any coordinator of the rounds until then. Or not. No guarantees …

In round ti, in which ti is coordinator,
no process can suspect it to have failed, so all with adopt ti’s proposal.

Formal Methods



Why Process Calculi ?

- precisely capture concurrent computation models
- rich algebraic theories (behavioural & logical)
- action-based proof technique

Why Process Calculi ?

- precisely capture concurrent computation models
- rich algebraic theories (behavioural & logical)
- action-based proof technique

choose an "expressive enough" PC of your liking
(*define a domain-specific one yourself*)

model the algorithm as a process term

model the specification
also as a process term? (*requires equational reasoning*)
as a logical formula? (*requires model-checking*)

Why Process Calculi ?

- precisely capture concurrent computation models
- rich algebraic theories (behavioural & logical)
- action-based proof technique

choose an "expressive enough" PC of your liking
(*define a domain-specific one yourself*)

model the algorithm as a process term

model the specification
also as a process term? (*requires equational reasoning*)
as a logical formula? (*requires model-checking*)

"executable" code
complete
formal
...

CONCUR 2003

Consensus in a Process Calculus

CONCUR 2003

Consensus in a Process Calculus

state-based reasoning is the key !

CONCUR 2003
Consensus in a Process Calculus

state-based reasoning is the key !

APC 2005
Much Ado About Nothing

Acta Informatica 2007
Consensus as a "State Machine"

TCS 2012 / PhD 2013
From Pseudo Code to Checked Proofs

CONCUR 2003
Consensus in a Process Calculus

APC 2005
Much Ado About Nothing

Acta Informatica 2007
Consensus as a "State Machine"

TCS 2012 / PhD 2013
From Pseudo Code to Checked Proofs

FORTE 2009 / Diploma Thesis 2012
Consensus in a Process Calculus, Again ...

state-based reasoning is the key !

CONCUR 2003
Consensus in a Process Calculus

APC 2005
Much Ado About Nothing

Acta Informatica 2007
Consensus as a "State Machine"

TCS 2012 / PhD 2013
From Pseudo Code to Checked Proofs

state-based reasoning is the key !

FORTE 2009 / Diploma Thesis 2012
Consensus in a Process Calculus, Again ...

EXPRESS 2014:
States in Process Calculi

CONCUR 2003
Consensus in a Process Calculus

state-based reasoning is the key !

APC 2005
Much Ado About Nothing

Acta Informatica 2007
Consensus as a "State Machine"

TCS 2012 / PhD 2013
From Pseudo Code to Checked Proofs

FORTE 2009 / Diploma Thesis 2012
Consensus in a Process Calculus, Again ...

EXPRESS 2014:
States in Process Calculi

TRENDS 2013

state-based reasoning is the key !

state-based reasoning is the key !

More precisely:

- Most of the correctness reasoning requires **invariants about the global state** of the system. (TLA-style …)

- *Global state* is composed of *local states* plus "*messages in travel*".

- *Local state* is not present in Process Calculi …

Explicit States in Distributed Process Calculi
—
Syntax

Memories

$$M : \mathcal{X} \to \mathbb{V} \cup \{\top, \perp\}$$

Variables **Values** “declared” “undefined”

Memories

$$M : \mathcal{X} \to \mathbb{V} \cup \{\top, \bot\}$$

Variables **Values** “declared” “undefined”

$\mathsf{w} \in \mathbb{V} \cup \{\top\}$

$$M\langle x \mapsto \mathsf{w}\rangle(y) \triangleq \begin{cases} \mathsf{w} & \text{if } x = y \\ M(y) & \text{if } x \neq y \end{cases}$$

Expressions & Evaluation

$$e ::= \mathsf{v} \mid x \mid (e, \ldots, e) \mid \mathrm{f}(e)$$

Expressions & Evaluation

$$e \;::=\; \mathsf{v} \;\big|\; x \;\big|\; (e, \ldots, e) \;\big|\; \mathrm{f}(e)$$

$$\mathrm{fetch}_M(e) \triangleq \begin{cases} e & \text{if } e \in \mathbb{V} \\ M(e) & \text{if } e \in \mathcal{X} \wedge M(e) \in \mathbb{V} \\ (\mathrm{fetch}_M(e_1), \ldots, \mathrm{fetch}_M(e_n)) & \text{if } e = (e_1, \ldots, e_n) \\ \mathrm{f}(\mathrm{fetch}_M(e')) & \text{if } e = \mathrm{f}(e') \\ \bot & \text{else} \end{cases}$$

Expressions & Evaluation

$$e \ ::= \ \mathsf{v} \mid \boxed{x} \mid (e, \ldots, e) \mid \mathrm{f}(e)$$

$$\mathrm{fetch}_M(e) \triangleq \begin{cases} e & \text{if } e \in \mathbb{V} \\ M(e) & \text{if } e \in \mathcal{X} \wedge M(e) \in \mathbb{V} \\ (\mathrm{fetch}_M(e_1), \ldots, \mathrm{fetch}_M(e_n)) & \text{if } e = (e_1, \ldots, e_n) \\ \mathrm{f}(\mathrm{fetch}_M(e')) & \text{if } e = \mathrm{f}(e') \\ \perp & \text{else} \end{cases}$$

$$\mathrm{eval}_M(e) \triangleq \mathrm{eval}(\mathrm{fetch}_M(e))$$

Message Reception

$a, b, c \in \mathscr{C}$ *(Channels)*

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$c(x)\,.\,P$

(1) input some value v from channel c

(2) use x to "remember" v afterwards

(3) continue with/as P

Message Reception

$a, b, c \in \mathscr{C}$ *(Channels)*

$c(x)\,.\,P$
(1) input some value v from channel c
(2) use x to "remember" v afterwards
(3) continue with/as P

Classically (e.g. [Milner]):

$$c(x)\,.\,P \quad \xrightarrow{c?v} \quad \{v/x\}P$$

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$c(x)\,.\,P$ $\left\{\begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to “remember” } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array}\right.$

<u>*Classically (e.g. [Milner]):*</u>

$c(x)\,.\,P \quad \xrightarrow{c?v} \quad \{v/x\}P$ for *any* value $v \in \mathscr{V}$

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$$c(x)\,.\,P \quad \left\{ \begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to “remember” } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array} \right.$$

<u>*Classically (e.g. [Milner])*</u>:

$$c(x)\,.\,P \xrightarrow{c?v} \{v/x\}P$$

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$c(x)\,.\,P$ $\left\{ \begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to "remember" } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array} \right.$

<u>*Classically (e.g. [Milner])*:</u>

$$c(x)\,.\,P \xrightarrow{c?v} \{v/x\}P$$

binding

Message Reception

$a, b, c \in \mathscr{C}$ *(Channels)*

$c(x)\,.\,P$ $\left\{ \begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to "remember" } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array} \right.$

Classically (e.g. [Milner]):

$$c(x)\,.\,P \xrightarrow{c?v} \{v/x\}P$$

binding

replaces all (free) occurrences of x in P with v

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$c(x) \,.\, P$ $\left\{ \begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to "remember" } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array} \right.$

Classically (e.g. [Milner]):

$$c(x) \,.\, P \xrightarrow{c?v} \{v/x\}P$$

binding

replaces all (free) occurrences of x in P with v

With local states (e.g. [Garavel]):

$$c(x) \,.\, P \xrightarrow{c?v} \textbf{?}$$

Message Reception

$a, b, c \in \mathscr{C}$ (Channels)

$$c(x) \,.\, P \quad \left\{ \begin{array}{ll} (1) & \text{input some value } v \text{ from channel } c \\ (2) & \text{use } x \text{ to “remember” } v \text{ afterwards} \\ (3) & \text{continue with/as } P \end{array} \right.$$

Classically (e.g. [Milner]):

$$c(x) \,.\, P \xrightarrow{c?v} \{v/x\}P$$

binding

replaces all (free) occurrences of x in P with v

With local states (e.g. [Garavel]):

$$c(x) \,.\, P \xrightarrow{c?v} \; ?$$

update the “associated” memory M by $M\langle x \mapsto v \rangle$

Message Reception

$a, b, c \in \mathscr{C}$ (*Channels*)

$c(x)\,.\,P$

(1) input some value v from channel c

(2) use x to "remember" v afterwards

(3) continue with/as P

Classically (e.g. [Milner]):

$$c(x)\,.\,P \xrightarrow{c?v} \{v/x\}P$$

binding

replaces all (free) occurrences of x in P with v

With local states (e.g. [Garavel]):

$$c(x)\,.\,P \xrightarrow{c?v} \textbf{?}$$

binding

update the "associated" memory M by $M\langle x \mapsto v\rangle$

Threads, Processes, Networks

$$
\begin{array}{llll}
\mu & ::= & \mathrm{var}\ x \;\Big|\; \langle x := e \rangle \;\Big|\; a(x) \;\Big|\; O & \text{actions} & \mathcal{A} \\
G & ::= & \mathbf{0} \;\Big|\; \mu.T \;\Big|\; G + G & \text{selections} & \mathcal{G} \\
T & ::= & G \;\Big|\; I^{x_1,\ldots,x_n} \;\Big|\; \mathsf{if}\ e\ \mathsf{then}\ T\ \mathsf{else}\ T \;\Big|\; T \mid T & \text{threads} & \mathcal{T} \\
P & ::= & [M \triangleleft T] & \text{processes} & \mathcal{P}
\end{array}
$$

Threads, Processes, Networks

$$
\begin{array}{rcll}
\mu & ::= & \mathrm{var}\ x \;\big|\; \langle x := e \rangle \;\big|\; a(x) \;\big|\; O & \text{actions} \qquad \mathcal{A} \\
G & ::= & \mathbf{0} \;\big|\; \mu.T \;\big|\; G + G & \text{selections} \qquad \mathcal{G} \\
T & ::= & G \;\big|\; I^{x_1,\ldots,x_n} \;\big|\; \textsf{if}\ e\ \textsf{then}\ T\ \textsf{else}\ T \;\big|\; T \mid T & \text{threads} \qquad \mathcal{T} \\
P & ::= & [M \triangleleft T] & \text{processes} \qquad \mathcal{P} \\
N & ::= & P \;\big|\; \text{Æ} \;\big|\; N \| N & \text{networks} \qquad \mathcal{N}
\end{array}
$$

Threads, Processes, Networks

$$
\begin{array}{lcll}
O & ::= & \emptyset \mid \{\!\!\{\overline{e}\langle e\rangle\}\!\!\} \mid O \uplus O & \text{outgoing bag} \\
\text{Æ} & ::= & \emptyset \mid \{\!\!\{\overline{\mathsf{c}}\langle \mathsf{v}\rangle\}\!\!\} \mid \text{Æ} \uplus \text{Æ} & \text{message aether} \quad \mathcal{Æ} \\
\\
\mu & ::= & \text{var } x \mid \langle x := e\rangle \mid a(x) \mid O & \text{actions} \quad \mathcal{A} \\
G & ::= & \mathbf{0} \mid \mu.T \mid G + G & \text{selections} \quad \mathcal{G} \\
T & ::= & G \mid I^{x_1,\ldots,x_n} \mid \mathsf{if}\ e\ \mathsf{then}\ T\ \mathsf{else}\ T \mid T \mid T & \text{threads} \quad \mathcal{T} \\
\\
P & ::= & [M \triangleleft T] & \text{processes} \quad \mathcal{P} \\
\\
N & ::= & P \mid \text{Æ} \mid N \| N & \text{networks} \quad \mathcal{N}
\end{array}
$$

Processes

$$[M \triangleleft T]$$

Processes

$$[M \triangleleft T]$$

All free variables of a thread *T* must be “bound” by *M*.

Binders …

$$[M \triangleleft T] =_\alpha [\ \{{}^{y}\!/_{x}\}M \ \triangleleft\ \{{}^{y}\!/_{x}\}T\ ]$$

Binders …

$$[M \triangleleft T] =_\alpha [\ \{{}^{y}\!/\!{}_{x}\}M \ \ \triangleleft\ \ \{{}^{y}\!/\!{}_{x}\}T\ ]$$

This is good. And bad.

Locations

$$\ell[M \triangleleft T]$$

Locations

$$\ell[M \triangleleft T]$$

Named Processes

Location-Aware Communication

Output $\overline{c}@l\langle e\rangle$ adds the name of the intended target;
Input $c@l(x)$ adds the name of the intended source;

(i) Location-aware send actions fit to the intended application domain.

(ii) Location-aware receive actions conveniently support suspicions.

Explicit States in Distributed Process Calculi — Semantics

Configurations

$$F \blacktriangleright_{\mathsf{trim}} N$$

Configurations

$$F \blacktriangleright_{\mathsf{trim}} N$$

Networks ***N***
running with *failed locations* in ***F***
with *trusted immortal* **trim**

Location-Aware Semantics

$$(\textsc{N-Fail})\ \frac{\mathsf{trim} \neq k \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \cup k \blacktriangleright_{\mathsf{trim}} N}$$

Location-Aware Semantics

$$(\textsc{Trim})\ \frac{\mathsf{trim} \in \mathcal{L}}{\emptyset \blacktriangleright N \longmapsto \emptyset \blacktriangleright_{\mathsf{trim}} N}$$

$$(\textsc{N-Fail})\ \frac{\mathsf{trim} \neq k \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \cup k \blacktriangleright_{\mathsf{trim}} N}$$

Location-Aware Semantics

$$(\textsc{TrIm})\ \frac{\mathsf{trim} \in \mathcal{L}}{\emptyset \blacktriangleright N \longmapsto \emptyset \blacktriangleright_{\mathsf{trim}} N}$$

$$(\textsc{N-Fail})\ \frac{\mathsf{trim} \neq k \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \cup k \blacktriangleright_{\mathsf{trim}} N}$$

$$(\textsc{N-Step})\ \frac{N \xrightarrow{\mathrm{step}\,@\ell} N' \qquad \mathrm{step} \in \{\mathrm{mem}, \mathrm{local}, \mathrm{snd}, \mathrm{rcv}\} \qquad \ell \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \blacktriangleright_{\mathsf{trim}} N'}$$

Location-Aware Semantics

$$(\textsc{Trim})\ \frac{\mathsf{trim} \in \mathcal{L}}{\emptyset \blacktriangleright N \longmapsto \emptyset \blacktriangleright_{\mathsf{trim}} N}$$

$$(\textsc{N-Fail})\ \frac{\mathsf{trim} \neq k \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \cup k \blacktriangleright_{\mathsf{trim}} N}$$

$$(\textsc{N-Step})\ \frac{N \xrightarrow{\mathrm{step}\ @\ell} N' \qquad \mathrm{step} \in \{\mathrm{mem}, \mathrm{local}, \mathrm{snd}, \mathrm{rcv}\} \qquad \ell \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \blacktriangleright_{\mathsf{trim}} N'}$$

$$(\textsc{N-Susp})\ \frac{N \xrightarrow{\mathrm{susp}(k)@\ell} N' \qquad k \neq \mathsf{trim} \qquad \ell \notin F}{F \blacktriangleright_{\mathsf{trim}} N \longmapsto F \blacktriangleright_{\mathsf{trim}} N'}$$

Located Steps (I)

$$(\textsc{Decl})\ \frac{\qquad}{\ell[\ M \ \triangleleft\ \mathrm{var}\ x.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\,@\ell} \ell[\ M\langle x \mapsto \top\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

Located Steps (I)

$$(\textsc{Decl})\ \frac{x \notin \mathrm{dom}(M) \cup \mathrm{fv}(\widehat{T})}{\ell[\ M \ \triangleleft\ \mathrm{var}\ x.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \top\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

Located Steps (I)

$$(\textsc{Decl})\ \frac{x \notin \mathrm{dom}(M) \cup \mathrm{fv}(\widehat{T})}{\ell[\ M \ \triangleleft\ \mathrm{var}\ x.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \top\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

$$(\textsc{Assign})\ \frac{\quad}{\ell[\ M \ \triangleleft\ \langle x := e\rangle.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \mathsf{v}\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

Located Steps (I)

$$(\textsc{Decl})\ \frac{x \notin \mathrm{dom}(M) \cup \mathrm{fv}(\widehat{T})}{\ell[\ M \ \triangleleft\ \mathrm{var}\ x.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \top\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

$$(\textsc{Assign})\ \frac{x \in \mathrm{dom}(M)}{\ell[\ M \ \triangleleft\ \langle x := e\rangle.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \mathsf{v}\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

Located Steps (I)

$$(\textsc{Decl})\ \frac{x \notin \mathrm{dom}(M) \cup \mathrm{fv}(\widehat{T})}{\ell[\ M \ \triangleleft\ \ \mathrm{var}\ x.T \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \top\rangle \ \triangleleft\ \ T \mid \widehat{T}\ \ ]}$$

$$(\textsc{Assign})\ \frac{x \in \mathrm{dom}(M) \qquad \mathrm{eval}_M(e) = \mathsf{v} \in \mathbb{V}}{\ell[\ M \ \triangleleft\ \ \langle x := e\rangle.T \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{mem}\ @\ell} \ell[\ M\langle x \mapsto \mathsf{v}\rangle \ \triangleleft\ \ T \mid \widehat{T}\ \ ]}$$

Located Steps (II)

$$
(\textsc{Snd})\ \frac{\overline{c}@l\langle e\rangle \in O \qquad\qquad O' = O\backslash\{\overline{c}@l\langle e\rangle\}}{\ell[\ M \ \triangleleft\ O.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{snd}\ @\ell} \ell[\ M \ \triangleleft\ O'.T \mid \widehat{T}\ ] \parallel \boxed{\{\!\![\ \mathsf{c}_{(\ell \to \mathsf{trg})}\mathsf{v}\ ]\!\!\}}}
$$

Located Steps (II)

$$
(\textsc{Snd})\ \frac{\overline{c}@l\langle e\rangle \in O \qquad O' = O\setminus\{\overline{c}@l\langle e\rangle\} \qquad \mathrm{eval}_M(c) = \mathsf{c} \in \mathbb{C}}{\ell[\ M \ \triangleleft\ O.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{snd}\ @\ell} \ell[\ M \ \triangleleft\ O'.T \mid \widehat{T}\ ] \parallel \{\!\!\{\ \mathsf{c}_{(\ell\to\mathsf{trg})}\mathsf{v}\ \}\!\!\}}
$$

Located Steps (II)

$$(\textsc{Snd})\ \frac{\begin{array}{cc} \overline{c}@l\langle e\rangle \in O & O' = O\backslash\{\overline{c}@l\langle e\rangle\} \\ \mathrm{eval}_M(c) = \mathsf{c} \in \mathbb{C} \qquad \mathrm{eval}_M(l) = \mathsf{trg} \in \mathbb{L} & \end{array}}{\ell[\ M \ \triangleleft\ O.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{snd}\ @\ell} \ell[\ M \ \triangleleft\ O'.T \mid \widehat{T}\ ] \ \| \ \boxed{\{\!\![\ \mathsf{c}_{(\ell\to\mathsf{trg})}\mathsf{v}\ ]\!\!\}}}$$

Located Steps (II)

$$(\textsc{Snd})\ \frac{\begin{array}{ccc} \overline{c}@l\langle e\rangle \in O & & O' = O\backslash\{\overline{c}@l\langle e\rangle\} \\ \mathrm{eval}_M(c) = \mathsf{c} \in \mathbb{C} & \mathrm{eval}_M(l) = \mathsf{trg} \in \mathbb{L} & \mathrm{eval}_M(e) = \mathsf{v} \in \mathbb{V} \end{array}}{\ell[\ M \ \triangleleft \ O.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{snd}\ @\ell} \ell[\ M \ \triangleleft \ O'.T \mid \widehat{T}\ ] \ \| \ \{\!\!\{\ \mathsf{c}_{(\ell \to \mathsf{trg})}\mathsf{v}\ \}\!\!\}}$$

Located Steps (II)

$$
(\textsc{Snd})\ \frac{\begin{array}{ccc} \overline{c}@l\langle e\rangle \in O & & O' = O\backslash\{\overline{c}@l\langle e\rangle\} \\ \mathrm{eval}_M(c) = \mathsf{c} \in \mathbb{C} & \mathrm{eval}_M(l) = \mathsf{trg} \in \mathbb{L} & \mathrm{eval}_M(e) = \mathsf{v} \in \mathbb{V} \end{array}}{\ell[\ M \ \triangleleft\ O.T \mid \widehat{T}\ ] \xrightarrow{\text{snd } @\ell} \ell[\ M \ \triangleleft\ O'.T \mid \widehat{T}\ ] \ \parallel\ \boxed{\{\!\!\{\ \mathsf{c}_{(\ell\to\mathsf{trg})}\mathsf{v}\ \}\!\!\}}}
$$

$$
(\textsc{Rcv})\ \frac{\mathrm{eval}_M(e) = \mathsf{c} \in \mathbb{C} \qquad \mathrm{eval}_M(l) = \mathsf{src} \in \mathbb{L} \qquad x \in \mathrm{dom}(M)}{\ell[\ M \ \triangleleft\ e@l(x).T \mid \widehat{T}\ ] \ \parallel\ \boxed{\{\!\!\{\ \mathsf{c}_{(\mathsf{src}\to\ell)}\mathsf{v}\ \}\!\!\}} \xrightarrow{\text{rcv } @\ell} \ell[\ M\langle x \mapsto \mathsf{v}\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}
$$

Located Steps (II)

$$(\textsc{Snd})\ \frac{\overline{c}@l\langle e\rangle \in O \qquad O' = O\backslash\{\overline{c}@l\langle e\rangle\} \qquad \mathrm{eval}_M(c) = \mathsf{c} \in \mathbb{C} \qquad \mathrm{eval}_M(l) = \mathsf{trg} \in \mathbb{L} \qquad \mathrm{eval}_M(e) = \mathsf{v} \in \mathbb{V}}{\ell[\ M \ \triangleleft\ O.T \mid \widehat{T}\ ] \xrightarrow{\mathrm{snd}\ @\ell} \ell[\ M \ \triangleleft\ O'.T \mid \widehat{T}\ ] \parallel \boxed{\llbracket\ \mathsf{c}_{(\ell\to\mathsf{trg})}\mathsf{v}\ \rrbracket}}$$

$$(\textsc{Rcv})\ \frac{\mathrm{eval}_M(e) = \mathsf{c} \in \mathbb{C} \qquad \mathrm{eval}_M(l) = \mathsf{src} \in \mathbb{L} \qquad x \in \mathrm{dom}(M)}{\ell[\ M \ \triangleleft\ e@l(x).T \mid \widehat{T}\ ] \parallel \boxed{\llbracket\ \mathsf{c}_{(\mathsf{src}\to\ell)}\mathsf{v}\ \rrbracket} \xrightarrow{\mathrm{rcv}\ @\ell} \ell[\ M\langle x \mapsto \mathsf{v}\rangle \ \triangleleft\ T \mid \widehat{T}\ ]}$$

$$(\textsc{Susp})\ \frac{\mathrm{eval}_M(l) = \mathsf{src} \in \mathbb{L}}{\ell[\ M \ \triangleleft\ e@l(x).T \mid \widehat{T}\ ] \xrightarrow{\mathrm{susp}(\mathsf{src})@\ell} \ell[\ M \ \triangleleft\ T \mid \widehat{T}\ ]}$$

Located Steps (III)

$$(\textsc{True})\ \frac{\mathrm{eval}_M(e) = \mathsf{t}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_1 \mid \widehat{T}\ \ ]}$$

$$(\textsc{False})\ \frac{\mathrm{eval}_M(e) = \mathsf{f}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_2 \mid \widehat{T}\ \ ]}$$

Located Steps (III)

$$(\textsc{True})\ \frac{\mathrm{eval}_M(e) = \mathsf{t}}{\ell[\ M \ \triangleleft\ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ T_1 \mid \widehat{T}\ ]}$$

$$(\textsc{False})\ \frac{\mathrm{eval}_M(e) = \mathsf{f}}{\ell[\ M \ \triangleleft\ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ T_2 \mid \widehat{T}\ ]}$$

$$(\textsc{Ident})\ \frac{}{\ell[\ M \ \triangleleft\ I^{x_1,\ldots,x_n} \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ G \mid \widehat{T}\ ]}$$

Located Steps (III)

$$(\textsc{True})\ \frac{\mathrm{eval}_M(e) = \mathsf{t}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_1 \mid \widehat{T}\ ]}$$

$$(\textsc{False})\ \frac{\mathrm{eval}_M(e) = \mathsf{f}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_2 \mid \widehat{T}\ ]}$$

$$(\textsc{Ident})\ \frac{I^{x_1,\ldots,x_n} \stackrel{\mathrm{def}}{=} G}{\ell[\ M \ \triangleleft\ \ I^{x_1,\ldots,x_n} \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ G \mid \widehat{T}\ ]}$$

Located Steps (III)

$$(\textsc{True})\ \frac{\mathrm{eval}_M(e) = \mathsf{t}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_1 \mid \widehat{T}\ ]}$$

$$(\textsc{False})\ \frac{\mathrm{eval}_M(e) = \mathsf{f}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_2 \mid \widehat{T}\ ]}$$

$$(\textsc{Ident})\ \frac{I^{x_1,\ldots,x_n} \stackrel{\mathrm{def}}{=} G \qquad \mathrm{fv}(G) \subseteq \{x_1, \ldots, x_n\}}{\ell[\ M \ \triangleleft\ \ I^{x_1,\ldots,x_n} \mid \widehat{T}\ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ G \mid \widehat{T}\ ]}$$

Located Steps (III)

$$(\textsc{True})\ \frac{\mathrm{eval}_M(e) = \mathsf{t}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_1 \mid \widehat{T}\ \ ]}$$

$$(\textsc{False})\ \frac{\mathrm{eval}_M(e) = \mathsf{f}}{\ell[\ M \ \triangleleft\ \ \mathsf{if}\ e\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2 \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ T_2 \mid \widehat{T}\ \ ]}$$

$$(\textsc{Ident})\ \frac{I^{x_1,\ldots,x_n} \stackrel{\mathrm{def}}{=} G \qquad \mathrm{fv}(G) \subseteq \{x_1, \ldots, x_n\} \qquad \{x_1, \ldots, x_n\} \subseteq \mathrm{dom}(M)}{\ell[\ M \ \triangleleft\ \ I^{x_1,\ldots,x_n} \mid \widehat{T}\ \ ] \xrightarrow{\mathrm{local}\ @\ell} \ell[\ M \ \triangleleft\ \ G \mid \widehat{T}\ \ ]}$$

Back to the Case Study …

Algorithm & ...

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

```
x_i := input;
for r := 1 to n do {  if r = i then  broadcast x_i;
                      if alive(p_r)  then x_i :=  input_from_broadcast };
output x_i;
```

$$\mathsf{L}_{\ell}^{\mathrm{chan,x,r,output}} \stackrel{\mathrm{def}}{=}$$

❷ $\langle r := r+1 \rangle.$

❸ if $r \le n$

then ❹ if $r = n$

then ❺ $\left(\biguplus_{\ell \neq j \in \{1,\ldots,n\}} \overline{\mathsf{chan}{\downarrow}r}@j\langle x \rangle \right).$ ❶ $\mathsf{L}_{\ell}^{\mathrm{chan,x,r,output}}$

else ❻ $\left(\mathsf{chan}{\downarrow}r@r(x) \right).$ ❶ $\mathsf{L}_{\ell}^{\mathrm{chan,x,r,output}}$

else ❼ $\langle \mathrm{output} := x \rangle.$ ❽ $\mathbf{0}$

… & Environment

$$\mathsf{Consensus}_{(\mathsf{input}_1,\ldots,\mathsf{input}_n)} \stackrel{\text{def}}{=}$$

$$\prod_{\ell\in\{1,\ldots,n\}} \ell\Big[$$

$$M_\perp \ \langle \mathrm{chan} \mapsto (\mathsf{c}_1,\ldots,\mathsf{c}_n)\rangle$$

$$\langle \mathrm{x} \mapsto \mathsf{input}_\ell\rangle$$

$$\langle \mathrm{r} \mapsto 0\rangle$$

$$\langle \mathrm{output} \mapsto \top\rangle$$

$$\triangleleft$$

$$❶ \quad \mathsf{L}_\ell^{\mathrm{chan,x,r,output}}$$

$$\Big]$$

$$\emptyset \blacktriangleright \mathsf{Consensus}_{(\mathsf{input}_1,\ldots,\mathsf{input}_n)}$$

Informal Correctness Argument

Table 4. *The Rotating Co-ordinator Algorithm for Participant i*

x_i `:= input;`
`for` r `:=` 1 `to` n `do {` `if` $r = i$ `then broadcast` x_i`;`
`if alive(`p_r`) then` x_i `:= input_from_broadcast };`
`output` x_i`;`

Before round ti, “anything goes”.

In such a round, any process may receive the value proposed by any coordinator of the rounds until then. Or not. No guarantees …

In round ti, in which ti is coordinator,
no process can suspect it to have failed, so all with adopt ti’s proposal.

Formal Correctness Argument

If $\quad \mathsf{Consensus}_{(\mathsf{input_1},\ldots,\mathsf{input_n})} \longmapsto^* F \blacktriangleright_{\mathsf{trim}} \quad \text{Æ} \;\parallel\; \prod_{\ell\in[1,n]} \ell[\, M_\ell \;\triangleleft\; pc \;\; T_\ell \,],$
then $\forall \ell \in [1, n]$.

$$\left(\begin{array}{ll} M_\ell(\mathrm{r}) < \mathsf{trim} & \to M_\ell(\mathrm{x}) \in \mathrm{Undecided} \\ M_\ell(\mathrm{r}) = \mathsf{trim} \wedge i \neq \mathsf{trim} & \to \Big(\big(pc \in \{❹, ❺, ❻\} \to M_\ell(\mathrm{x}) \in \mathrm{Undecided}\big) \\ & \quad \wedge \big(pc \in \{❼, ❶, ❷\} \to M_\ell(\mathrm{x}) = M_{\mathsf{trim}}(\mathrm{x})\big)\Big) \\ M_\ell(\mathrm{r}) > \mathsf{trim} & \to M_\ell(\mathrm{x}) = M_{\mathsf{trim}}(\mathrm{x}) \\ M_\ell(\mathrm{r}) > n \wedge pc = \{❽\} & \to M_\ell(\mathrm{output}) = M_\ell(\mathrm{x}) \end{array}\right)$$

$\mathrm{Undecided} \triangleq \{\mathsf{input}_1, \ldots, \mathsf{input}_n\}$

Conclusions

Open Problems

Does it work in sufficiently many **cases**?

What about **mechanization** support?

Do such calculi still qualify as <u>process</u> calculi?

What about the **meta theory** of such calculi?

Is it useful to extend them with **session types**?