# The pi-calculus
# Origin and recent developments

Joachim Parrow
Uppsala University

*Robin Milner*
*Photo from*
*Jan. 1986*

# Blackboard in Robin's office, April 1987

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

$$\overline{a}x.P \mid a(y).Q \xrightarrow{\tau} P \mid Q\{x/y\}$$

# Blackboard in Robin's office, April 1987

VALUE PASSING

$$\overline{a}x.P \stackrel{\overline{a}x}{\rightarrow} P \qquad a(y).Q \stackrel{a(y)}{\rightarrow} Q$$

$$\overline{a}x.P \mid a(y).Q \stackrel{\tau}{\rightarrow} P \mid Q\{x/y\}$$

# Blackboard in Robin's office, April 1987

$$\overline{a}x.P \stackrel{\overline{a}x}{\rightarrow} P \qquad a(y).Q \stackrel{a(y)}{\rightarrow} Q$$

$$\overline{a}x.P \mid a(y).Q \stackrel{\tau}{\rightarrow} P \mid Q\{x/y\}$$

$$(\overline{a}x.P)\backslash x$$

# Blackboard in Robin's office, April 1987

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

$$\overline{a}x.P \mid a(y).Q \xrightarrow{\tau} P \mid Q\{x/y\}$$

$$(\overline{a}x.P)\backslash x \xrightarrow{\overline{a}(x)} P$$

# Blackboard in Robin's office, April 1987

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

$$\overline{a}x.P \mid a(y).Q \xrightarrow{\tau} P \mid Q\{x/y\}$$

$$(\overline{a}x.P)\backslash x \xrightarrow{\overline{a}(x)} P$$

$$(\overline{a}x.P)\backslash x \mid a(y).Q$$

# Blackboard in Robin's office, April 1987


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

$$\overline{a}x.P \mid a(y).Q \xrightarrow{\tau} P \mid Q\{x/y\}$$

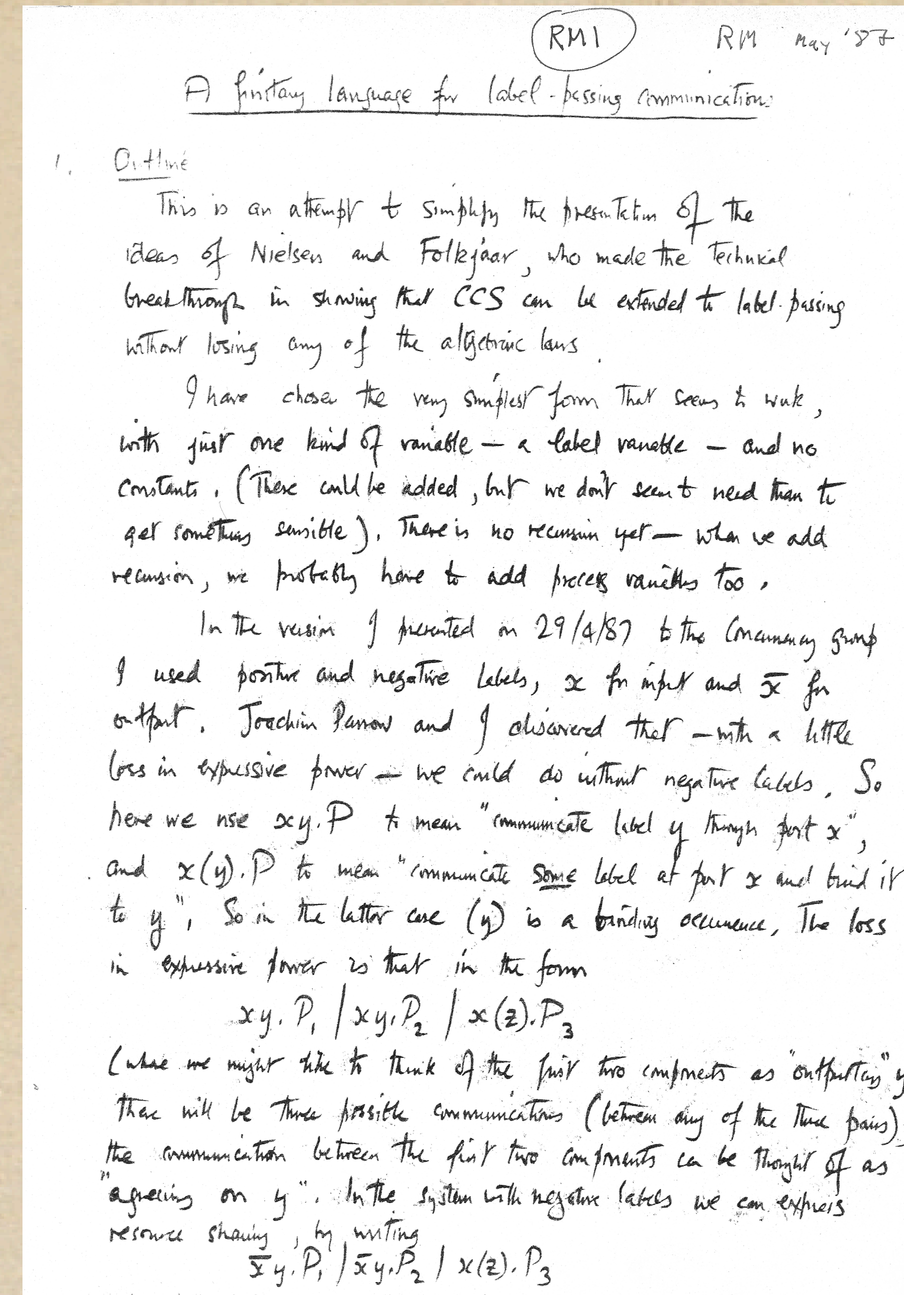
$$(\overline{a}x.P)\backslash x \xrightarrow{\overline{a}(x)} P$$

$$(\overline{a}x.P)\backslash x \mid a(y).Q \xrightarrow{\tau} (P \mid Q\{x/y\})\backslash x$$

The very first written note by Robin on what was to become the pi-calculus.

What do you think Robin did in the very first sentence?

1) Explained the main idea
x) Explained the motivation
2) Gave most of the credit to someone else


(RM1) RM May '87

A finitary language for label-passing communication

1. Outline

This is an attempt to simplify the presentation of the ideas of Nielsen and Folkjaar, who made the technical breakthrough in showing that CCS can be extended to label-passing without losing any of the algebraic laws.

I have chosen the very simplest form that seems to work, with just one kind of variable — a label variable — and no constants. (These could be added, but we don't seem to need them to get something sensible). There is no recursion yet — when we add recursion, we probably have to add process variables too.

In the version I presented on 29/4/87 to the Concurrency group I used positive and negative labels, $x$ for input and $\bar{x}$ for output. Joachim Parrow and I observed that — with a little loss in expressive power — we could do without negative labels. So here we use $xy.P$ to mean "communicate label $y$ through port $x$", and $x(y).P$ to mean "communicate some label at port $x$ and bind it to $y$". So in the latter case $(y)$ is a binding occurrence. The loss in expressive power is that in the form

$$xy.P_1 \mid xy.P_2 \mid x(z).P_3$$

(where we might like to think of the first two components as "outputting" $y$) there will be three possible communications (between any of the three pairs); the communication between the first two components can be thought of as "agreeing on $y$". In the system with negative labels we can express resource sharing, by writing

$$\bar{x}y.P_1 \mid \bar{x}y.P_2 \mid x(z).P_3$$

RMI RM May '87

# A finitary language for label-passing communications

## 1. Outline

This is an attempt to simplify the presentation of the ideas of Nielsen and Folkjaar, who made the technical breakthrough in showing that CCS can be extended to label-passing without losing any of the algebraic laws.

I have chosen the very simplest form that seems to work, with just one kind of variable — a label variable — and no constants. (These could be added, but we don't seem to need them to get something sensible). There is no recursion yet — when we add recursion, we probably have to add process variables too.

In the version I presented on 29/4/87 to the Concurrency group I used positive and negative labels, $x$ for input and $\bar{x}$ for output. Joachim Parrow and I observered that — with a little

RM1 RM May '87

# A fimitary language for label-passing communication

1. Outline

This is an attempt to simplify the presentation of the ideas of Nielsen and Folkjaar, who made the technical breakthrough in showing that CCS can be extended to label-passing without losing any of the algebraic laws.

"This is an attempt to simplify the presentation of the ideas of Nielsen and Folkjaar [sic], who made the technical breakthrough in showing that CCS can be extended to label-passing without losing any of the algebraic laws"

get something sensible), there is no recursion yet — when we add recursion, we probably have to add process variables too.

In the version I presented on 29/4/87 to the Concurrency group I used positive and negative labels, $x$ for input and $\bar{x}$ for output. Joachim Parrow and I observed that — with a little

# The first pi-calculus semantics (May '87)!

4.

5. Rules of action (Rules marked * have a symmetric form)

**FREE ACTION**

F-ACT: $xy.P \xrightarrow{xy} P$

**BOUND ACTION**

B-ACT: $x(y).P \xrightarrow{x(z)} P\{z/y\}$ $\quad z \notin FV(x(y).P)$

**SILENT ACTION**

τ-ACT: $\tau.P \xrightarrow{\tau} P$

**SUM**

SUM*: $$\frac{P_j \xrightarrow{a} P_j'}{\sum_i P_i \xrightarrow{a} P_j'}$$

**COMPOSITION**

F-COM*: $$\frac{P_1 \xrightarrow{xy} P_1'}{P_1 | P_2 \xrightarrow{xy} P_1' | P_2}$$

B-COM*: $$\frac{P_1 \xrightarrow{x(y)} P_1'}{P_1 | P_2 \xrightarrow{x(y)} P_1' | P_2} \quad (y \notin FV(P_2))$$

τ-COM*: $$\frac{P_1 \xrightarrow{\tau} P_1'}{P_1 | P_2 \xrightarrow{\tau} P_1' | P_2}$$

FB-COM*: $$\frac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{x(z)} P_2'}{P_1 | P_2 \xrightarrow{\tau} P_1' | P_2'\{y/z\}}$$

F-COM: $$\frac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{xy} P_2'}{P_1 | P_2 \xrightarrow{\tau} P_1' | P_2'}$$

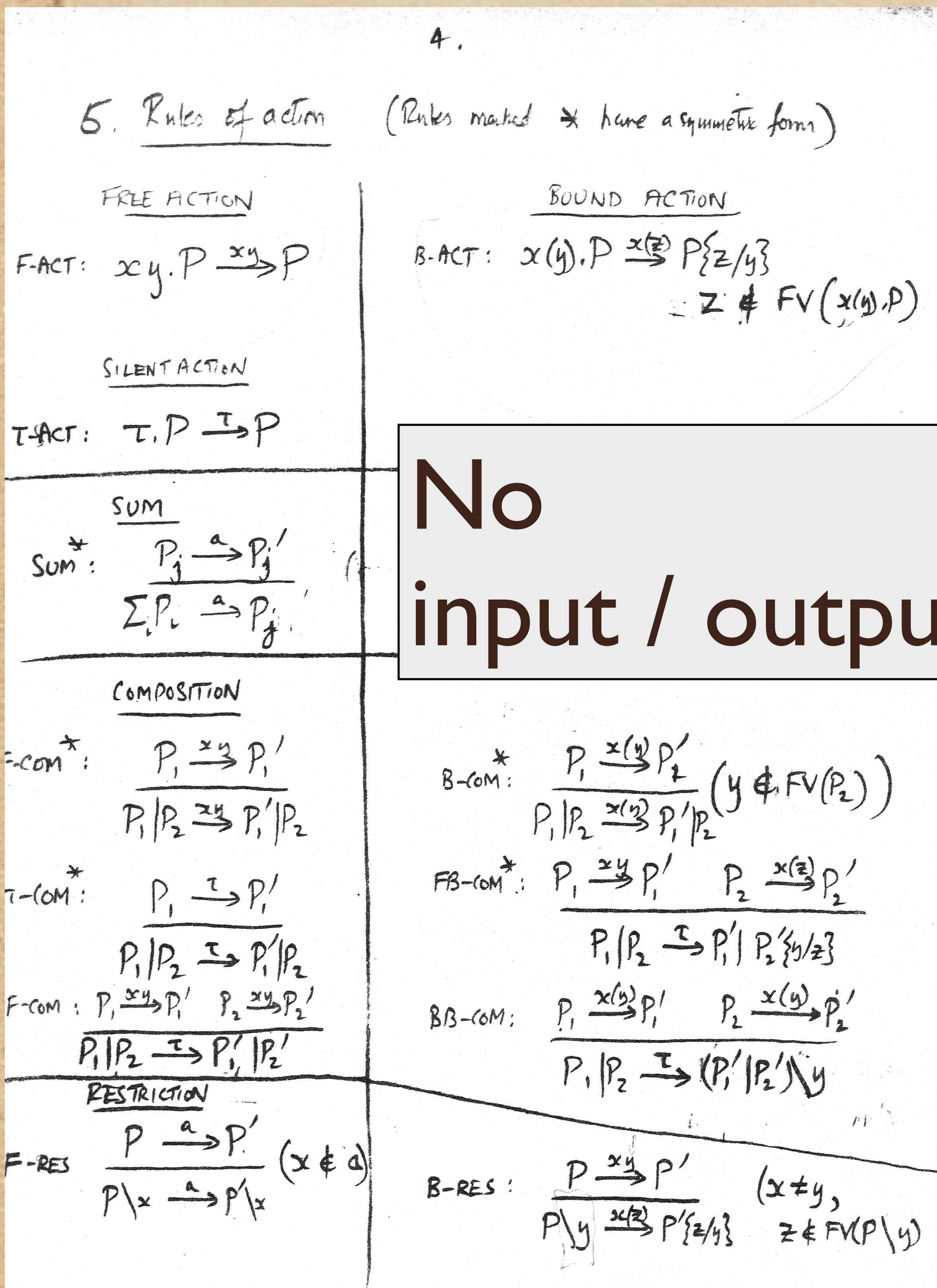
BB-COM: $$\frac{P_1 \xrightarrow{x(y)} P_1' \quad P_2 \xrightarrow{x(y)} P_2'}{P_1 | P_2 \xrightarrow{\tau} (P_1' | P_2')\backslash y}$$

**RESTRICTION**

F-RES: $$\frac{P \xrightarrow{a} P'}{P\backslash x \xrightarrow{a} P'\backslash x} \quad (x \notin a)$$

B-RES: $$\frac{P \xrightarrow{xy} P'}{P\backslash y \xrightarrow{x(z)} P'\{z/y\}} \quad (x \neq y,\ z \notin FV(P\backslash y))$$

# The first pi-calculus semantics (May '87)!



4.

5. Rules of action (Rules marked * have a symmetric form)

FREE ACTION

F-ACT: $xy.P \xrightarrow{xy} P$

SILENT ACTION

T-ACT: $\tau.P \xrightarrow{\tau} P$

SUM

SUM*: $\dfrac{P_j \xrightarrow{a} P_j'}{\Sigma_i P_i \xrightarrow{a} P_j'}$

COMPOSITION

F-COM*: $\dfrac{P_1 \xrightarrow{xy} P_1'}{P_1|P_2 \xrightarrow{xy} P_1'|P_2}$

T-COM*: $\dfrac{P_1 \xrightarrow{\tau} P_1'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2}$

F-COM: $\dfrac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{xy} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'}$

RESTRICTION

F-RES: $\dfrac{P \xrightarrow{a} P'}{P\backslash x \xrightarrow{a} P'\backslash x}$ $(x \notin a)$

BOUND ACTION

B-ACT: $x(y).P \xrightarrow{x(z)} P\{z/y\}$, $z \notin FV(x(y).P)$

B-COM*: $\dfrac{P_1 \xrightarrow{x(y)} P_1'}{P_1|P_2 \xrightarrow{x(y)} P_1'|P_2}$ $(y \notin FV(P_2))$

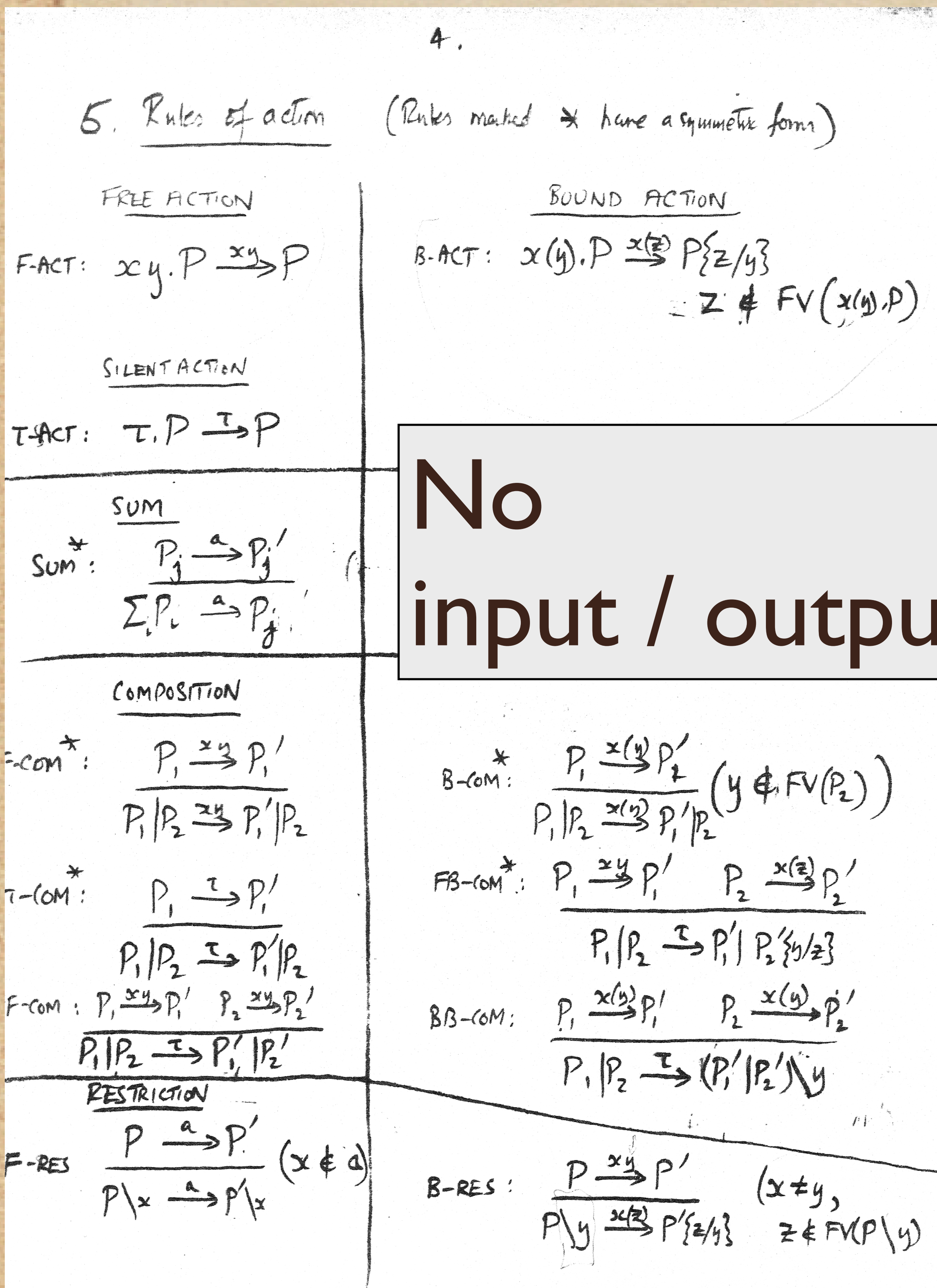
FB-COM*: $\dfrac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{x(z)} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'\{y/z\}}$

BB-COM: $\dfrac{P_1 \xrightarrow{x(y)} P_1' \quad P_2 \xrightarrow{x(y)} P_2'}{P_1|P_2 \xrightarrow{\tau} (P_1'|P_2')\backslash y}$

B-RES: $\dfrac{P \xrightarrow{xy} P'}{P\backslash y \xrightarrow{x(z)} P'\{z/y\}}$ $(x \neq y,\ z \notin FV(P\backslash y))$

No input / output

# The first pi-calculus semantics (May '87)!


4.

5. Rules of action (Rules marked * have a symmetric form)

FREE ACTION

F-ACT: $xy.P \xrightarrow{xy} P$

BOUND ACTION

B-ACT: $x(y).P \xrightarrow{x(z)} P\{z/y\}$ $\quad z \notin FV(x(y).P)$

SILENT ACTION

τ-ACT: $\tau.P \xrightarrow{\tau} P$

SUM

SUM*: $\dfrac{P_j \xrightarrow{a} P_j'}{\Sigma_i P_i \xrightarrow{a} P_j'}$

COMPOSITION

F-COM*: $\dfrac{P_1 \xrightarrow{xy} P_1'}{P_1|P_2 \xrightarrow{xy} P_1'|P_2}$

B-COM*: $\dfrac{P_1 \xrightarrow{x(y)} P_1'}{P_1|P_2 \xrightarrow{x(y)} P_1'|P_2}$ $(y \notin FV(P_2))$

τ-COM*: $\dfrac{P_1 \xrightarrow{\tau} P_1'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2}$

FB-COM*: $\dfrac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{x(z)} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'\{y/z\}}$

F-COM: $\dfrac{P_1 \xrightarrow{xy} P_1' \quad P_2 \xrightarrow{xy} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'}$

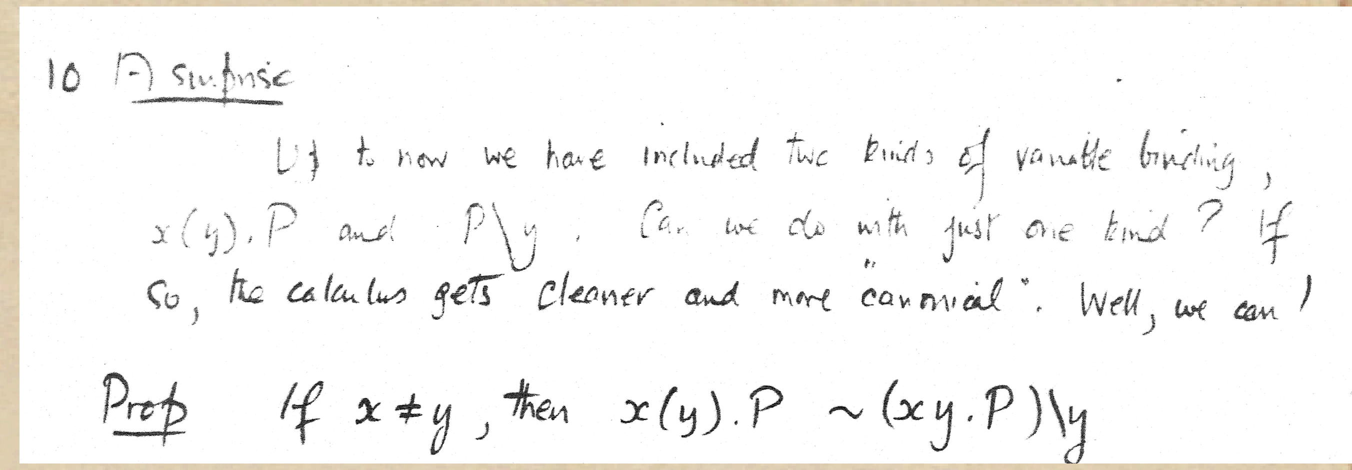
BB-COM: $\dfrac{P_1 \xrightarrow{x(y)} P_1' \quad P_2 \xrightarrow{x(y)} P_2'}{P_1|P_2 \xrightarrow{\tau} (P_1'|P_2')\backslash y}$

RESTRICTION

F-RES: $\dfrac{P \xrightarrow{a} P'}{P\backslash x \xrightarrow{a} P'\backslash x}$ $(x \notin a)$

B-RES: $\dfrac{P \xrightarrow{xy} P'}{P\backslash y \xrightarrow{x(z)} P'\{z/y\}}$ $(x \neq y,\ z \notin FV(P\backslash y))$

# No input / output


10 A Surprise

Up to now we have included two kinds of variable binding, $x(y).P$ and $P\backslash y$. Can we do with just one kind? If so, the calculus gets cleaner and more "canonical". Well, we can!

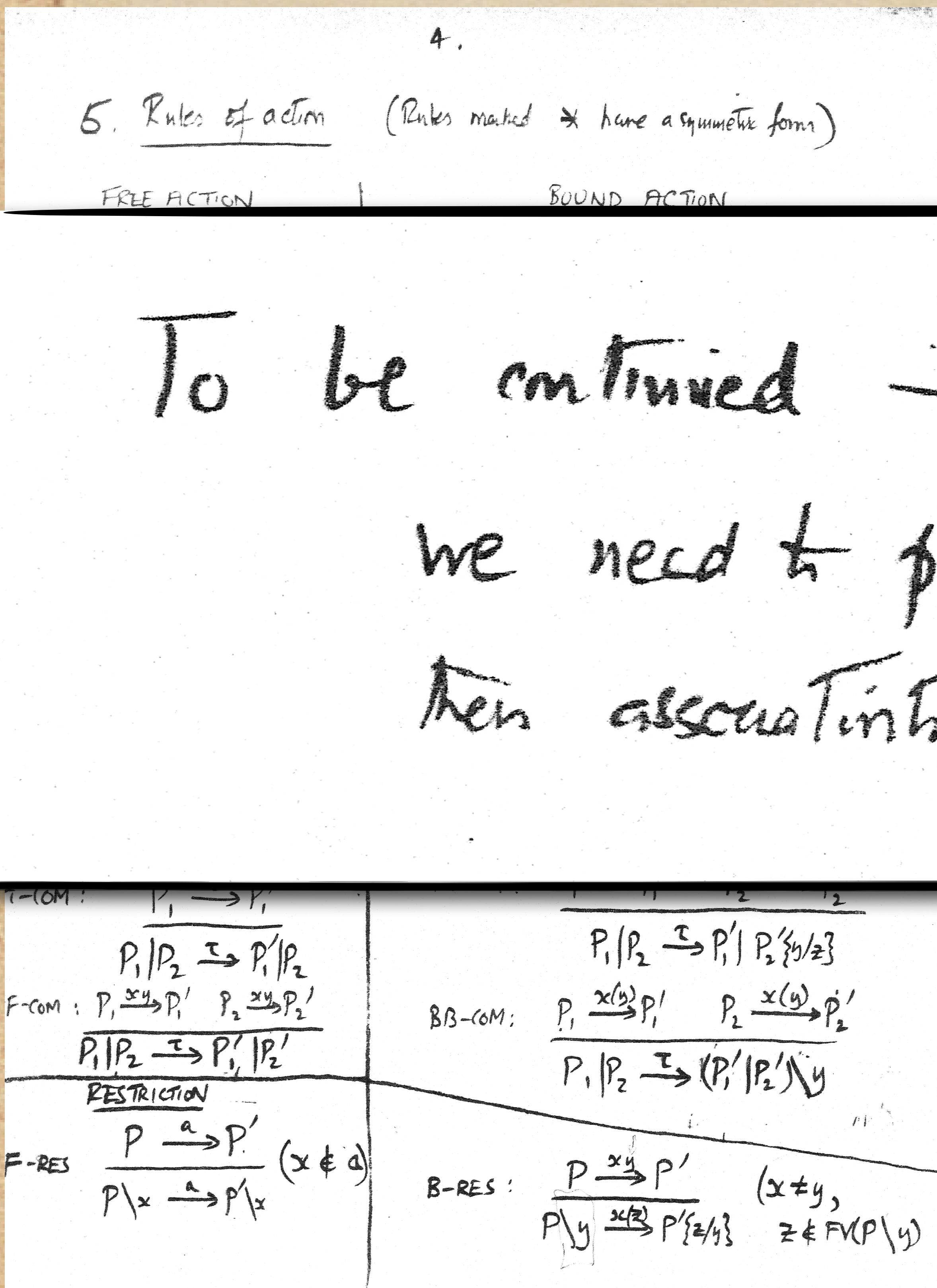
Prop If $x \neq y$, then $x(y).P \sim (xy.P)\backslash y$

A Surprise

Up to now we have included the two kinds of variable binding, $x(y).P$ and $P\backslash y$. Can we do with just one kind? If so, the calculus gets cleaner and more "canonical". Well, we can!

Prop If $x \neq y$, then $x(y).P \sim (xy.P)\backslash y$

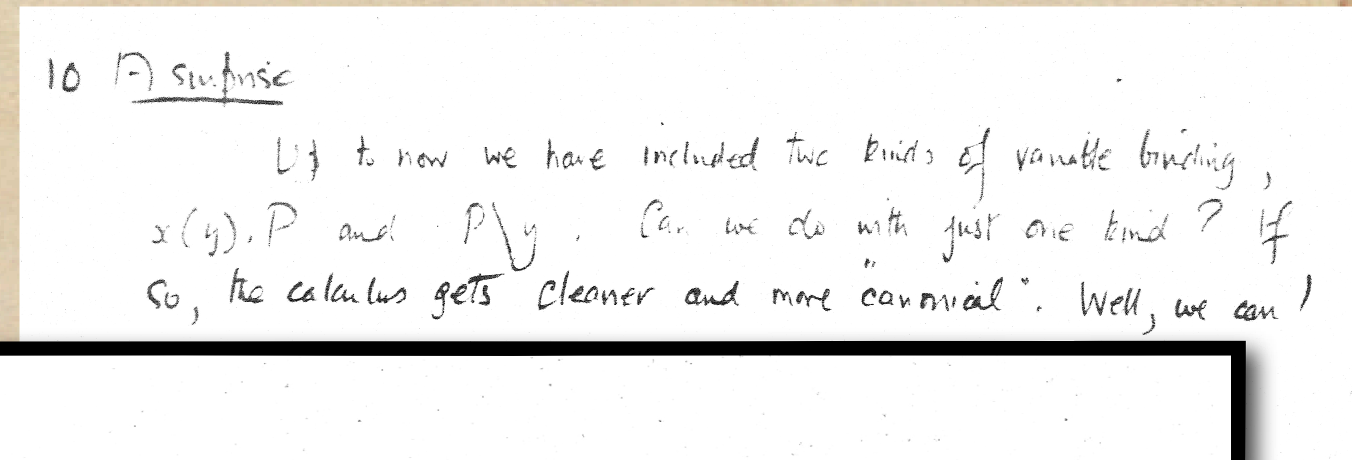
# The first pi-calculus semantics (May '87)!


4.

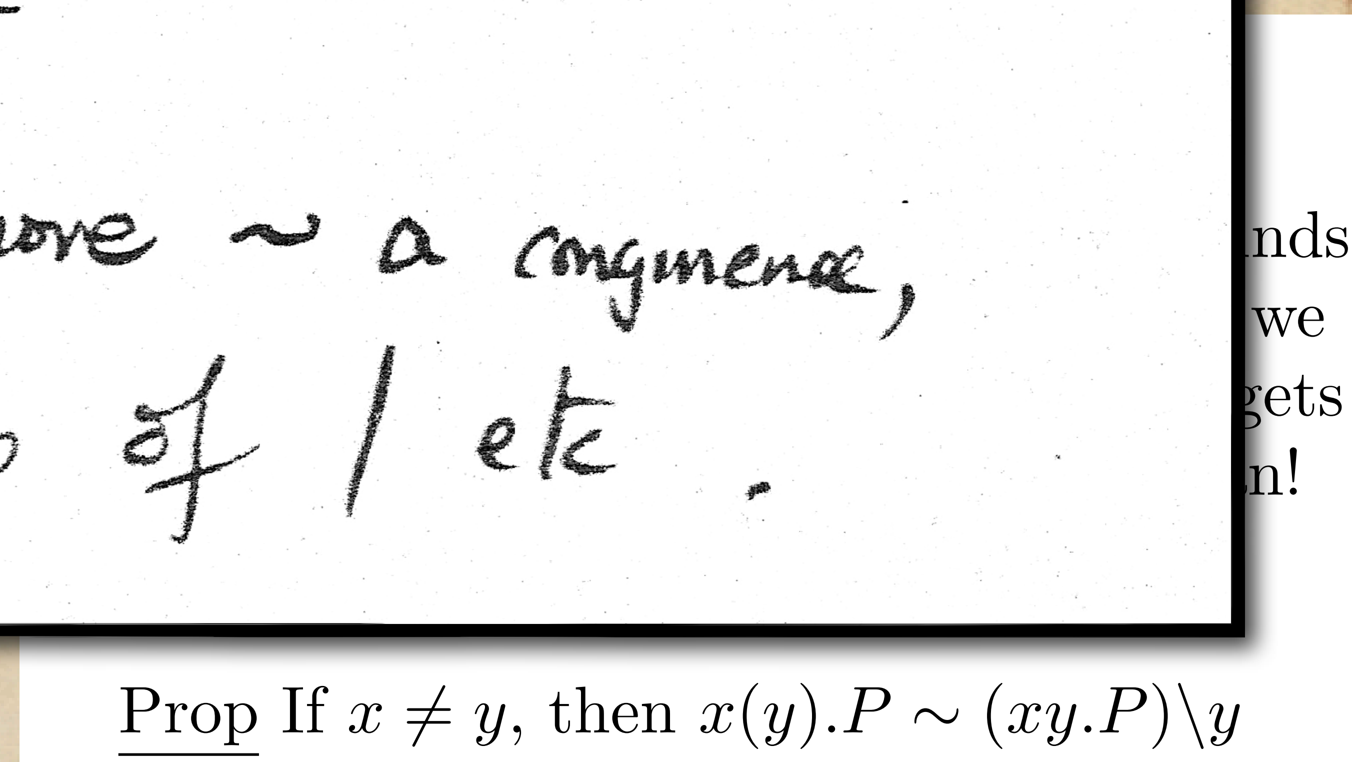
5. Rules of action (Rules marked * have a symmetric form)

FREE ACTION | BOUND ACTION




10 A Surprise

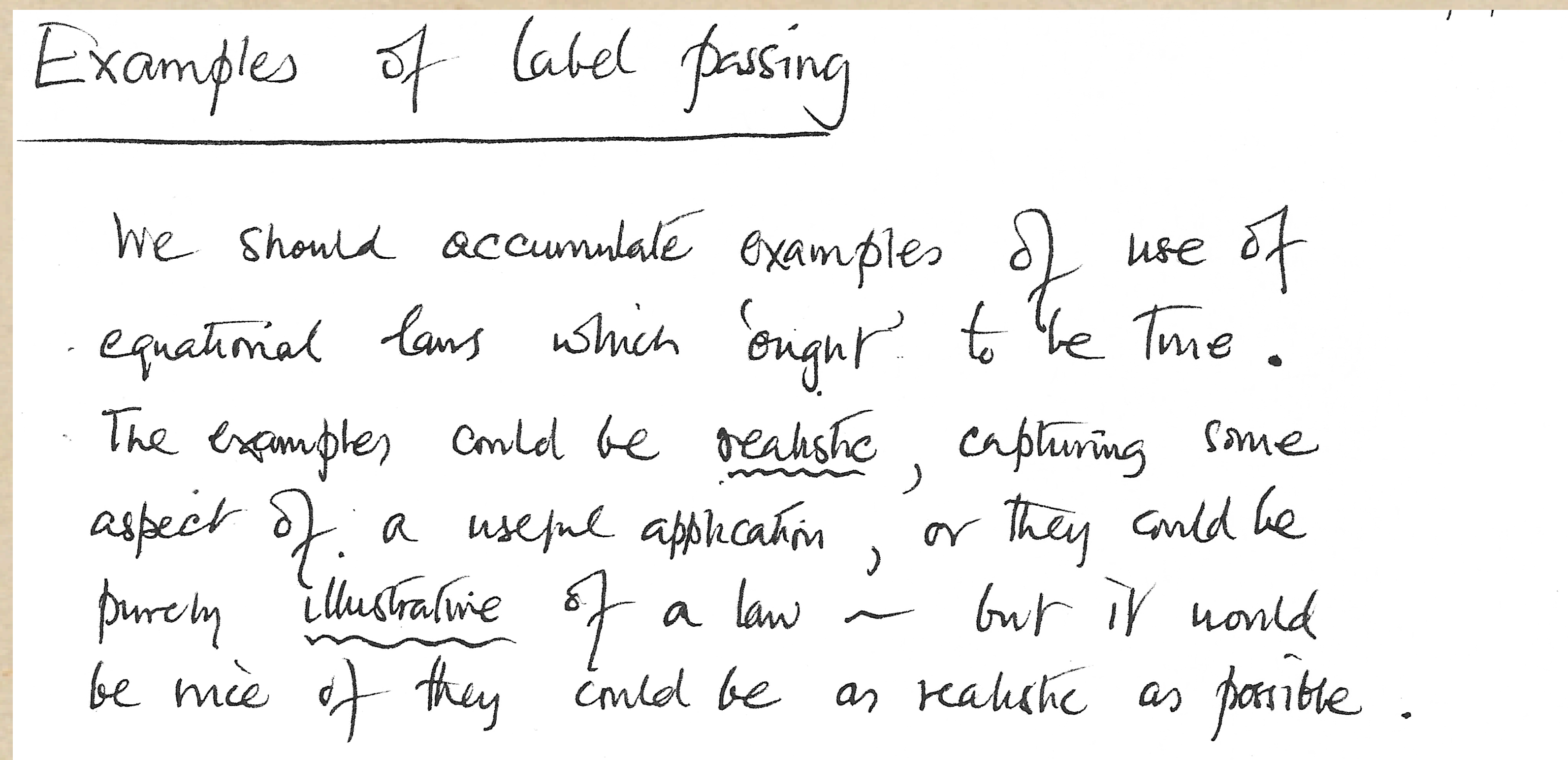
Up to now we have included two kinds of variable binding, $x(y).P$ and $P\backslash y$. Can we do with just one kind? If so, the calculus gets cleaner and more "canonical". Well, we can!


Prop If $x \neq y$, then $x(y).P \sim (xy.P)\backslash y$

To be continued —

we need to prove ~ a congruence, then associativity of | etc.

# Beginning of Robin's second note (June '87):


Examples of label passing

We should accumulate examples of use of equational laws which 'ought' to be true. The examples could be realistic, capturing some aspect of a useful application, or they could be purely illustrative of a law ~ but it would be nice if they could be as realistic as possible.

"We should accumulate examples of the use of equational laws which 'ought' to be true. The examples could be realistic, capturing some aspect of a useful application, or they could be purely illustrative of a law - but it would be nice if they could be as realistic as possible."

Turned out to have:

- Wrong basic constructors
- Wrong definition of bisimulation
- No sensible algebraic laws

Turned out to have:

- Wrong basic constructors
- Wrong definition of bisimulation
- No sensible algebraic laws

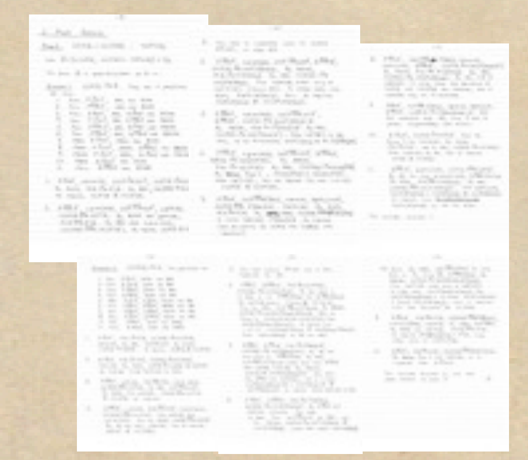
Revision

Turned out to have:

- Wrong basic constructors
- Wrong definition of bisimulation
- No sensible algebraic laws

Revision

Establish properties

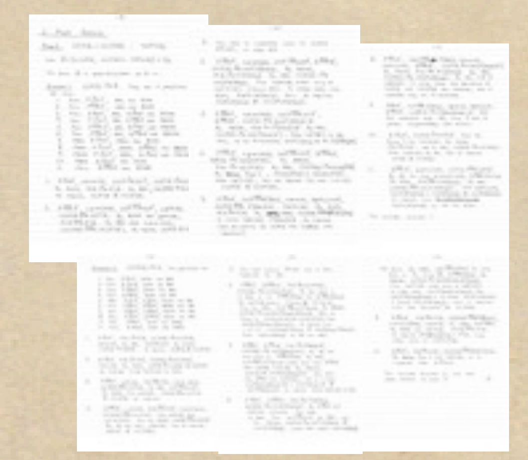


Turned out to have:

- Wrong basic constructors
- Wrong definition of bisimulation
- No sensible algebraic laws

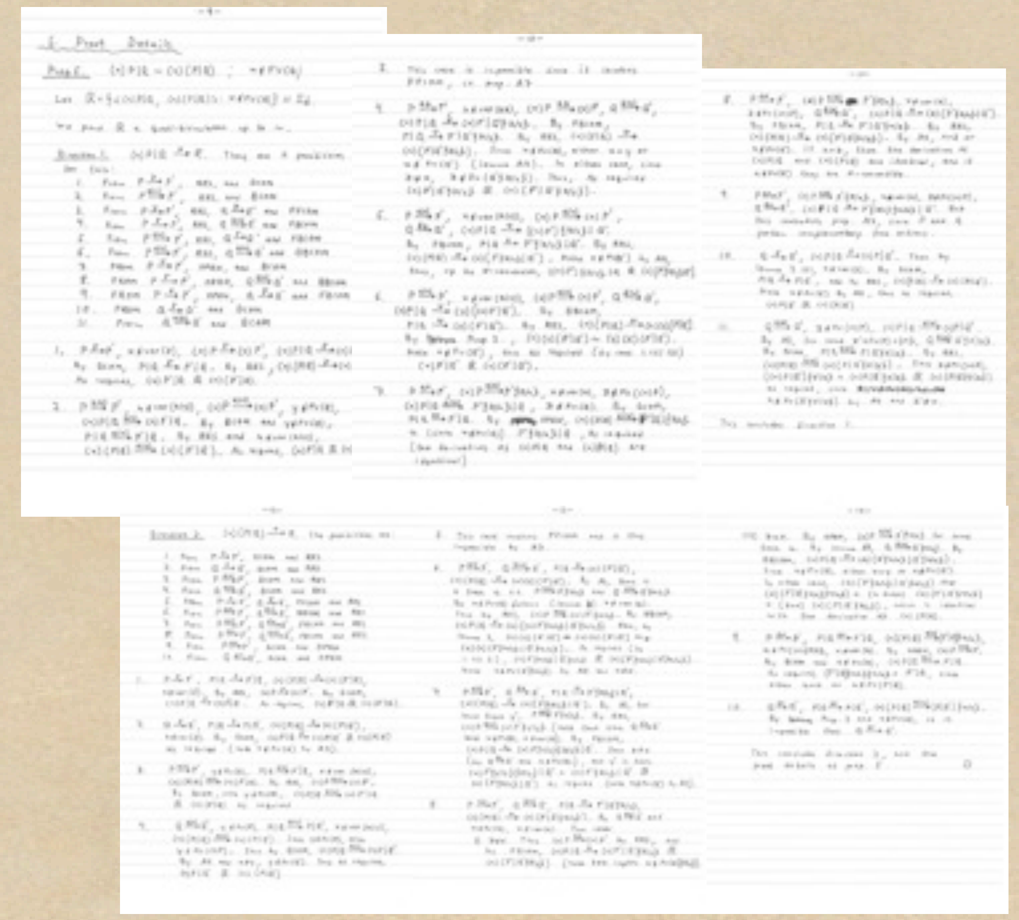
Revision

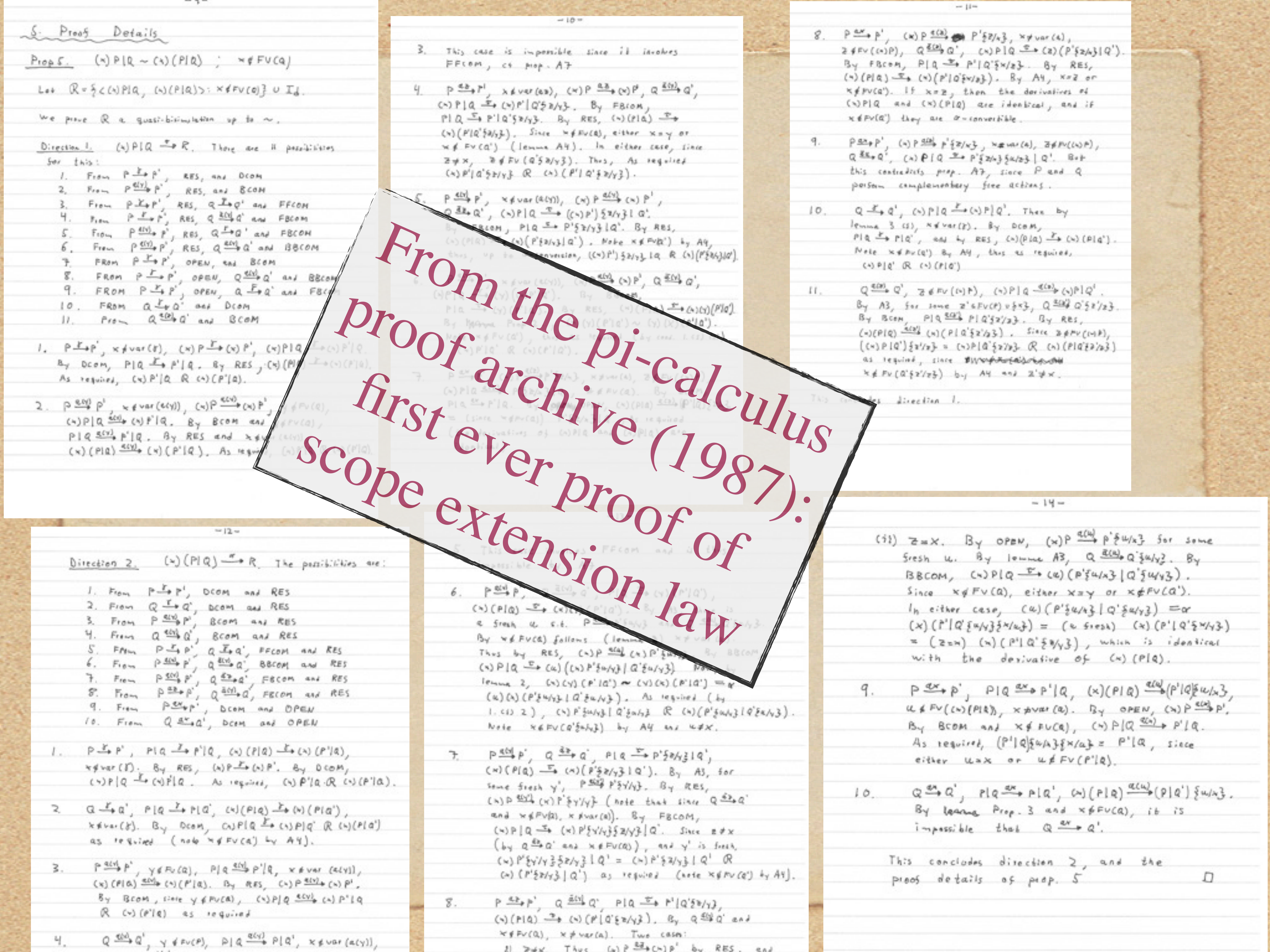
Establish properties



Time passes
Proof archive grows

# Time passes
# Proof archive grows





From the pi-calculus proof archive (1987): first ever proof of scope extension law
§. Proof Details
Direction 1.
Direction 2.
This concludes direction 2, and the proof details of prop. 5

# Two years later, this is called the ”pi-calculus”

Date: 12 Apr 89 15:13:18 BST
From: RM@ED.ECSVAX (Robin Milner)
Subject: How about this for a title and abstract?
To: jgp@ed.LFCS (N%"jgp@lfcs")
Message-Id: <"12-APR-1989 15:13:18">
Status: RO

Mobile processes (or the pi-calculus)
-----------------

Robin Milner, Joachim Parrow, David Walker

Process calculi such as TCSP, ACP, CCS have not, on the whole, allowed for shifting contiguity among agents (though they allow them to bifurcate and to die). The purpose of this talk is to present a very basic calculus in which shifting contiguity, modelled by the use of names to communicate

# Robin’s reply to my question ”why pi”?

I thought "process", or "pointer", or "parallel", but I also thought it a usable name -- if not too arrogant, and signifying that it aspires to primitivity like the lambda-calculus.  You could also think of it as a near successor to the lambda calculus.  Consider:

# Robin’s reply to my question ”why pi”?

I thought "process", or "pointer", or "parallel", but I also thought it a usable name -- if not too arrogant, and signifying that it aspires to primitivity like the lambda-calculus. You could also think of it as a near successor to the lambda calculus. Consider:

mu-calculus ... this significantly exists

# Robin's reply to my question "why pi"?

I thought "process", or "pointer", or "parallel", but I also thought it a usable name -- if not too arrogant, and signifying that it aspires to primitivity like the lambda-calculus. You could also think of it as a near successor to the lambda calculus. Consider:

mu-calculus ... this significantly exists

nu-calculus ... I thought we might have used this name, (nu standing for "name"), but mu and nu sound so alike.

# Robin's reply to my question "why pi"?

I thought "process", or "pointer", or "parallel", but I also thought it a usable name -- if not too arrogant, and signifying that it aspires to primitivity like the lambda-calculus. You could also think of it as a near successor to the lambda calculus. Consider:

mu-calculus ... this significantly exists

nu-calculus ... I thought we might have used this name, (nu standing for "name"), but mu and nu sound so alike.

omicron calculus ... who would want that?

# Robin's reply to my question "why pi"?

I thought "process", or "pointer", or "parallel", but I also thought it a usable name -- if not too arrogant, and signifying that it aspires to primitivity like the lambda-calculus. You could also think of it as a near successor to the lambda calculus. Consider:

mu-calculus ... this significantly exists

nu-calculus ... I thought we might have used this name,
(nu standing for "name"), but mu and nu
sound so alike.

omicron calculus ... who would want that?

which leads to

PI-CALCULUS

... I put it in parentheses to try it out ..

# So this is all settled?

Technically, yes, around 20 years ago

\+ Explains fundamental principles well

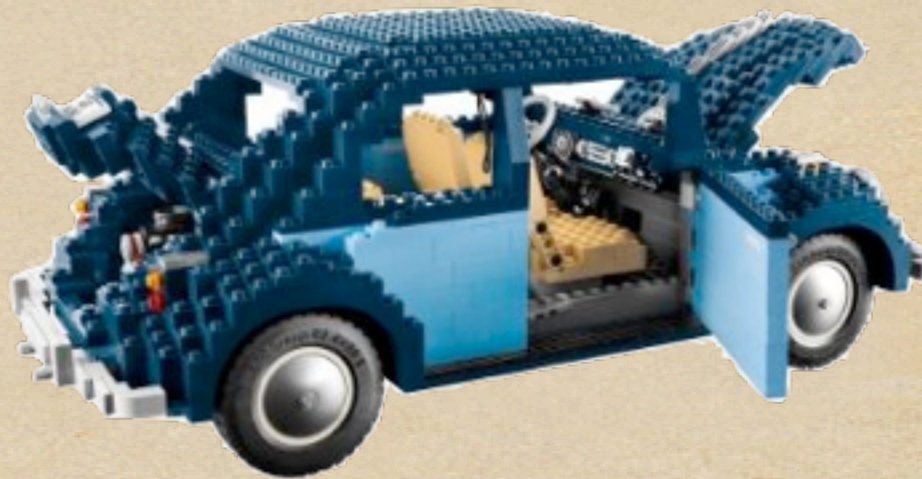
# So this is all settled?

Technically, yes, around 20 years ago

+ Explains fundamental principles well
- Really not usable in application projects

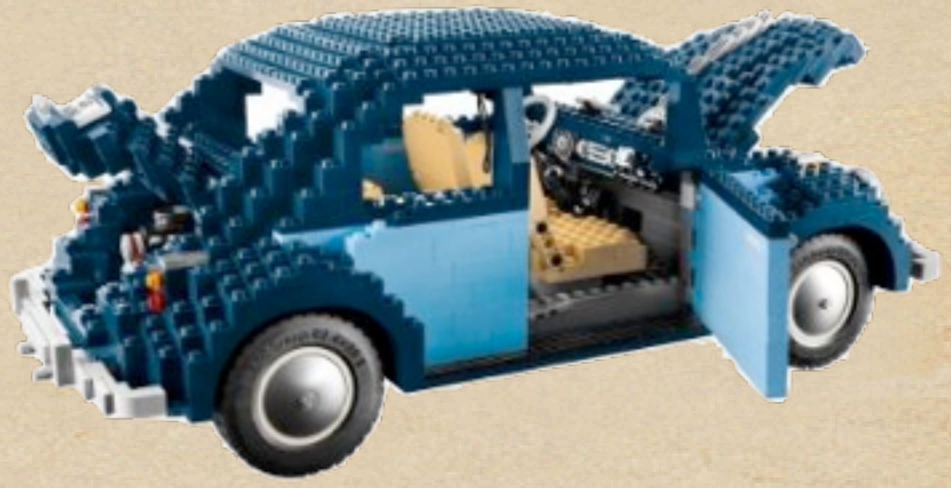
We need applied rather
than minimal models!

# Applied calculi



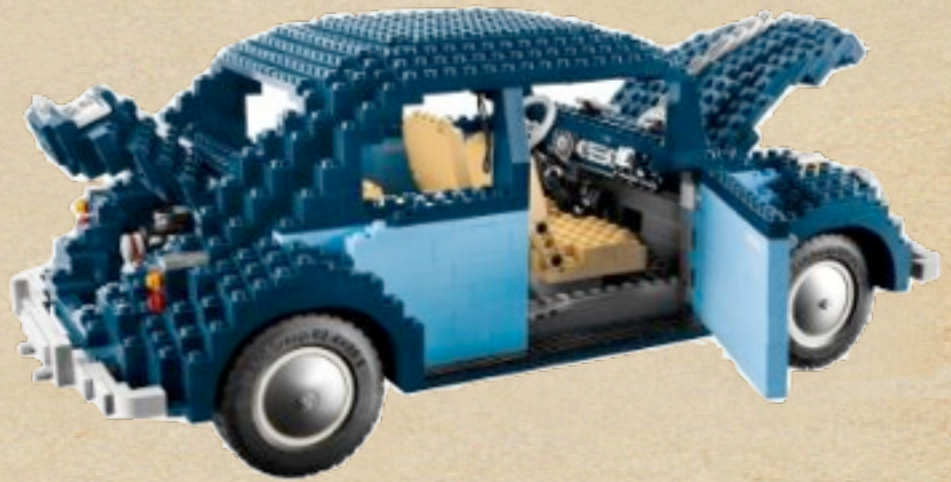
# Applied calculi

- Encodings: more constructs are derived from the few primitve ones.



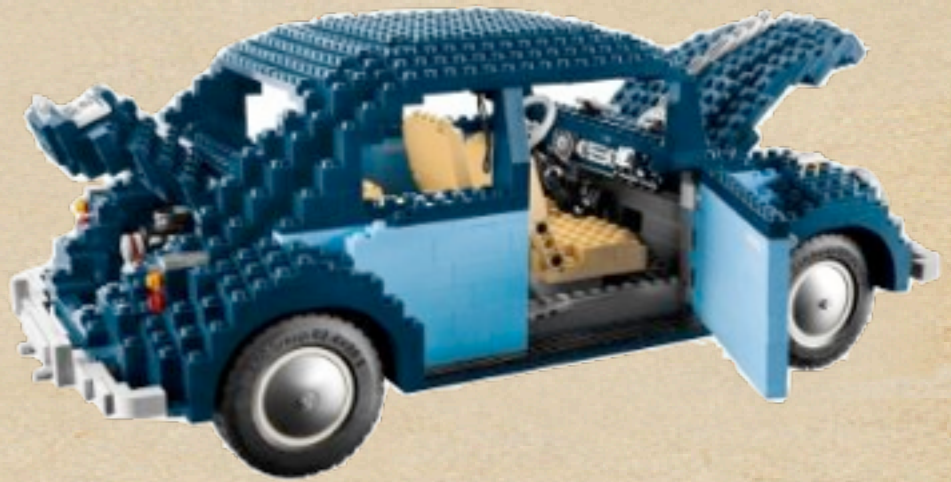
# Applied calculi

- Encodings: more constructs are derived from the few primitve ones.
- + Can inherit much theory



# Applied calculi

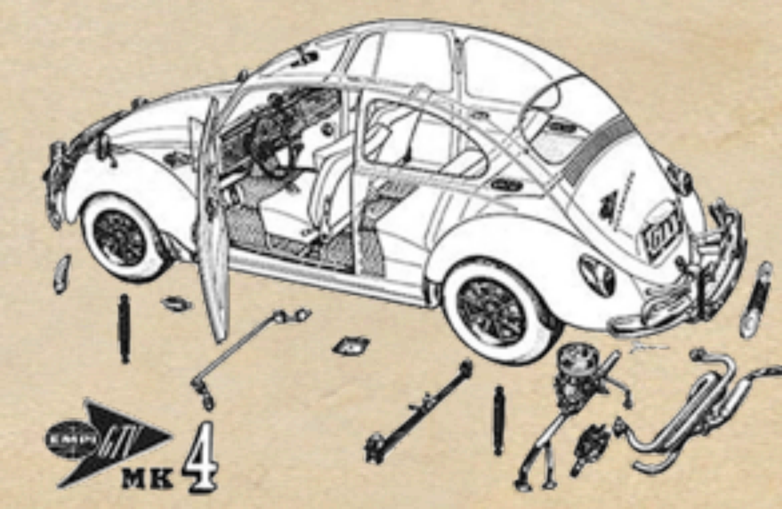
- Encodings: more constructs are derived from the few primitve ones.
- + Can inherit much theory
- - Encodings can be opaque



# Applied calculi

- **Encodings**: more constructs are derived from the few primitve ones.
- + Can inherit much theory
- - Encodings can be opaque

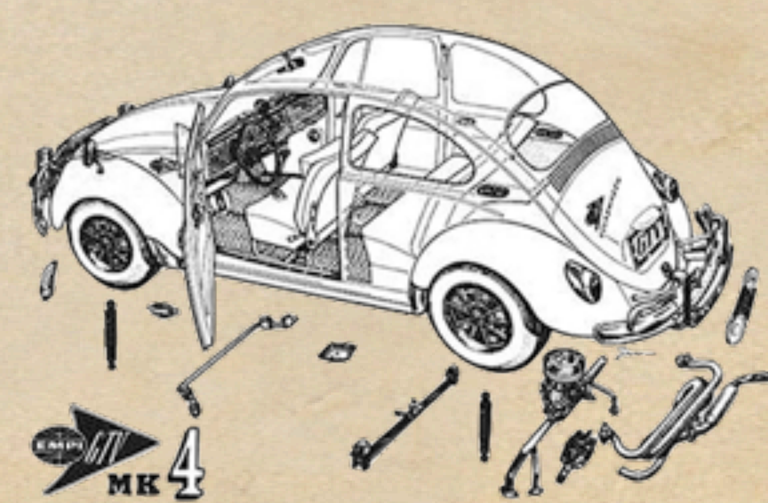
- **Enrichments**: new constructs are added.



# Applied calculi

- Encodings: more constructs are derived from the few primitve ones.
- + Can inherit much theory
- - Encodings can be opaque

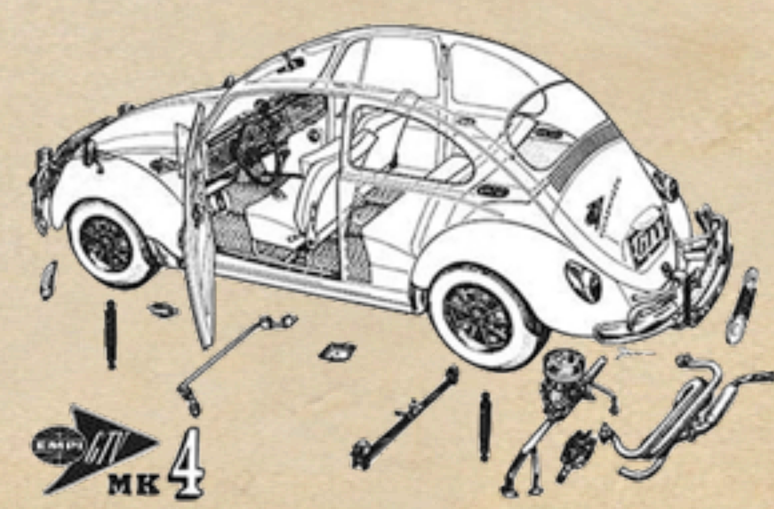
- Enrichments: new constructs are added.
- + More intuitive definitions



# Applied calculi

- **Encodings**: more constructs are derived from the few primitve ones.
- + Can inherit much theory
- - Encodings can be opaque

- **Enrichments**: new constructs are added.
- + More intuitive definitions
- - Theory needs to be redone



# A plethora of calculi


SPEECH
ZONE

# All-purpose calculus?

Just one problem:

# All-purpose calculus?

Just one problem:

In real life there is no such


AUTOMOBILWERK EISENACH
A FACTORY FOR CALCULI

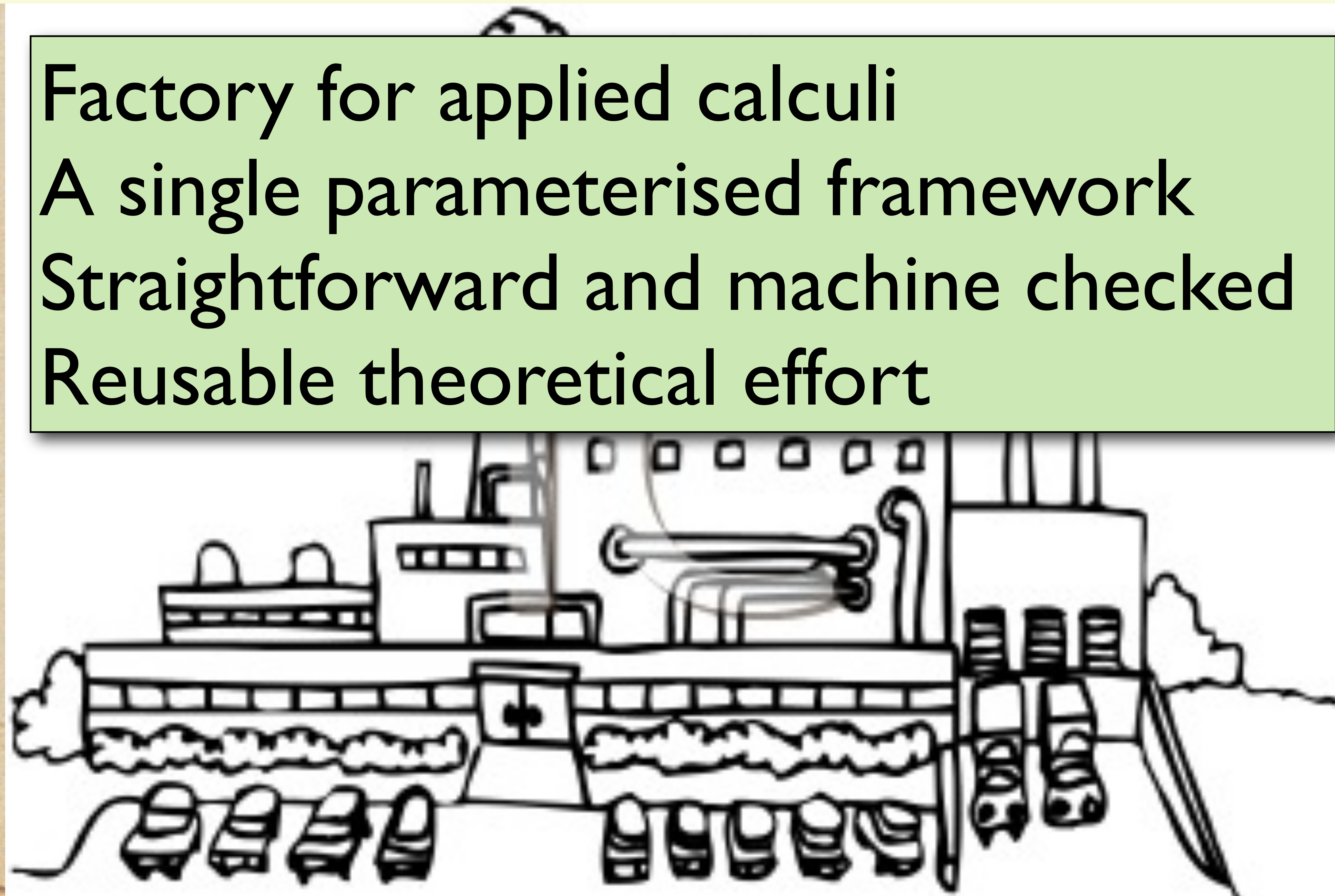
# Psi-calculi framework

(Bengtson, Johansson, Parrow, Victor 2008 - )

# Psi-calculi framework

(Bengtson, Johansson, Parrow, Victor 2008 - )

Factory for applied calculi
A single parameterised framework
Straightforward and machine checked
Reusable theoretical effort



# Psi-calculi

$$(\nu z)(\overline{a}z) \mid a(x).\,[x = b]P$$

Ordinary pi-calculus

# Psi-calculi

$$(\nu z)(\overline{a}z) \mid a(x).\,[x = b]P$$

Ordinary pi-calculus

arbitrary set of data

$$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$$

Data structures can be sent

# Psi-calculi

$(\nu z)(\overline{a}z) \mid a(x).\,[x = b]P$

Ordinary pi-calculus

arbitrary set of data

$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$

Data structures can be sent

$(\nu z)(\overline{a}M) \mid a(\lambda\tilde{x})N.\,[x = b]P$

Pattern matching

# Psi-calculi

$(\nu z)(\overline{a}z) \mid a(x).\,[x = b]P$

Ordinary pi-calculus

arbitrary set of data

$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$

Data structures can be sent

$(\nu z)(\overline{a}M) \mid a(\lambda\tilde{x})N.\,[x = b]P$

Pattern matching

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,[x = b]P$

Channels can be arbitrary structures

set of data

$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$

can be sent

$(\nu z)(\overline{a}M) \mid a(\lambda\tilde{x})N.\,[x = b]P$

Pattern matching

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,[x = b]P$

Channels can be arbitrary structures

set of data

$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$

can be sent

$(\nu z)(\overline{a}M) \mid a(\lambda\tilde{x})N.\,[x = b]P$

Pattern matching

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,[x = b]P$

Channels can be arbitrary structures

arbitrary logic

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,\text{if } \varphi \text{ then } P$

Tests can be arbitrary predicates

set of data

$(\nu z)(\overline{a}M) \mid a(x).\,[x = b]P$

can be sent

$(\nu z)(\overline{a}M) \mid a(\lambda\tilde{x})N.\,[x = b]P$

Pattern matching

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,[x = b]P$

Channels can be arbitrary structures

arbitrary logic

$(\nu z)(\overline{K}M) \mid L(\lambda\tilde{x})N.\,\text{if } \varphi \text{ then } P$

Tests can be arbitrary predicates

*new construct*

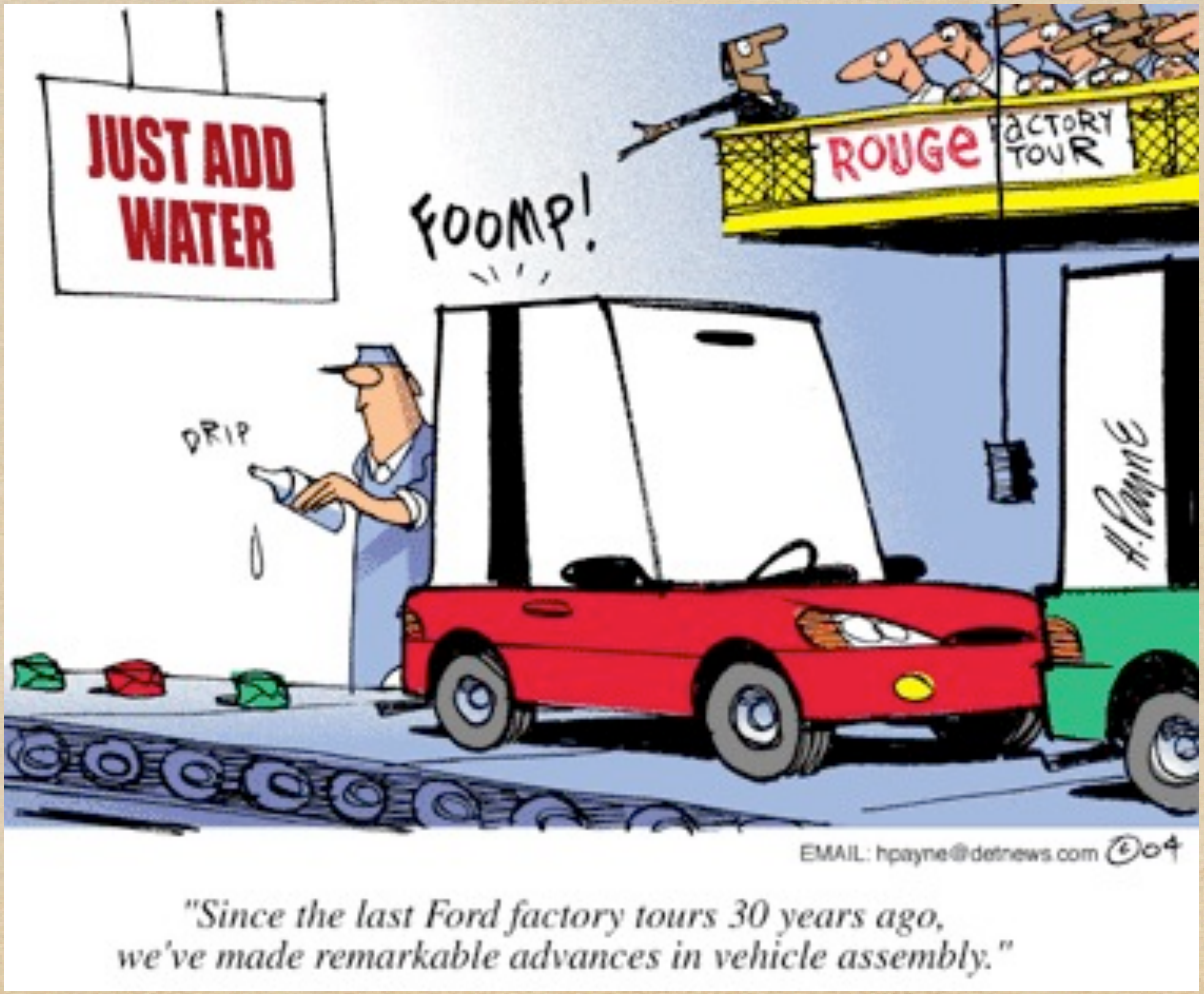
$(\nu z)(\overline{K}M).\,(\!|\Psi|\!) \mid L(\lambda\tilde{x})N.\,\text{if } \varphi \text{ then } P$

**assertions,** ie facts about data used to resove predicates

# Well, not completely arbitrary...

- Data sets and logics must be **nominal** (Pitts, Gabbay 2000) - there is a notion of name and what names are contained in what terms. *These names can be scoped*
- A few general requisites, eg composition of assertions is an abelian monoid

# Just add data and logic



# Just add data and logic

1. Define names, data terms, and a logic
   can be absolutely anything nominal.

2. Define a few operators, eg substitution, channel equivalence, ...
   must satisfy some requisites

# Just add data and logic

1. Define names, data terms, and a logic
   can be absolutely anything nominal.

2. Define a few operators, eg substitution, channel equivalence, ...
   must satisfy some requisites

→ **A Psi-calculus**

# Assertions:
information embedded in processes

# Assertions:
# information embedded in processes

- Global facts about data structures

$$\mathrm{first}(\mathrm{pair}(x, y)) = x$$
$$\mathrm{decrypt}(\mathrm{encrypt}(M, k), k) = M$$

# Assertions: information embedded in processes

- local knowledge

$$(\nu k)(\llparenthesis c = \mathrm{encrypt}(M, k) \rrparenthesis \mid P)$$

# Assertions:
## information embedded in processes

- parametrised

$$a(x)\,.\,(\llparenthesis c = \mathrm{encrypt}(M, x) \rrparenthesis \mid P)$$

# Assertions: information embedded in processes

- communicated

$$a(x)\,.\,(\llparenthesis x \rrparenthesis \mid P)$$

- **Can capture**
  - **Applied pi-calculus** (Abadi, Fournet 2001)
  - **Explicit fusion calculus** (Wischik, Gardner 2005)
  - **Concurrent constraint pi** (Buscemi, Montanari 2007)
  - **Polyadic synchronization** (Carbone, Maffeis 2003)
  - **Pattern matching** and **higher order values** (Various)
- **And moreover**
  - **Higher-order and non-monotonic concurrent constraints**
  - **Algebraic operators on communication channels**

- **Polyadic synchronization** (Carbone, Maffeis 2003)
- **Pattern matching** and **higher order values** (Various)

- **And moreover**
  - **Higher-order and non-monotonic concurrent constraints**
  - **Algebraic operators on communication channels**

# Results

- Standard Semantics
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

# Results

- Standard Semantics ← Definition of behaviour
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

# Results

- Standard Semantics ← Definition of behaviour
- Symbolic Semantics ← More "computable"
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

# Results

- Standard Semantics
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

Definition of behaviour

More ”computable”

If $P$ and $Q$ behave the same, then $P|R$ and $Q|R$ behave the same

# Results

- Standard Semantics
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

Efficient proof method

Definition of behaviour

More ”computable”

If $P$ and $Q$ behave the same, then $P|R$ and $Q|R$ behave the same

# Results

Efficient proof method

- Standard Semantics
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

Definition of behaviour

More "computable"

If $P$ and $Q$ behave the same, then $P|R$ and $Q|R$ behave the same

Intuitive equivalence

# Results

Efficient proof method

- Standard Semantics
- Symbolic Semantics
- Compositionality
- Strong and Weak Bisimulation
- Barbed Congruence
- Algebraic Laws

Definition of behaviour

More "computable"

If $P$ and $Q$ behave the same, then $P|R$ and $Q|R$ behave the same

Intuitive equivalence

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

# Results

- Standard Semantics
- Sy... ...tics
- C...
- Strong and Wea... ...tion
- Barbed Congruence
- Algebraic Laws

Machine checked once and for all

Efficient proof method

Intuitive equivalence

Definition of behaviour

More ”computable”

If $P$ and $Q$ behave the same, then $P|R$ and $Q|R$ behave the same

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

# Correctness: the holy grail

# Theory Development in a Theorem Prover

Advocated by Robin in the work on LCF

Benefit 1: **Certainty** (no false assertions)
Benefit 2: Good proof **structure** (clarity of arguments)

# Theory Development in a Theorem Prover

Advocated by Robin in the work on LCF

Benefit 1: **Certainty** (no false assertions)
Benefit 2: Good proof **structure** (clarity of arguments)

Benefit 3: **Flexibility** (easy to change details)
Benefit 4: **Generality** (easy to keep track of assumptions)

# Flexibility

- **Theory development is like programming**: It almost never starts from scratch. You continually add, improve, amend, adjust...

# Flexibility

- **Theory development is like programming**: It almost never starts from … nually … end, adjust...

Please change this one

- **Programming**: Every amendment needs a program recompilation.

- **Theory development**: Every amendment needs a re-check of all proofs. *A huge error source*.

- **Mechanised proofs** means we have a proof repository and can quickly assess ramifications of changes.

# Recent Developments I

**Higher-order**

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications

Already possible

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications

Already possible

A received process can be executed

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications

Already possible

A received process can be executed

Requires extension

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications

Already possible

A received process can be executed

Requires extension

Process definitions

# Recent Developments I

**Higher-order**

Processes can be transmitted in communications — Already possible

A received process can be executed — Requires extension

Process definitions $A(x) \Leftarrow \bar{b}x \,.\, c(x) \,.\, A(x)$

New syntax $\mathbf{run}\ M$

$M$ is any data term

New syntax $\mathbf{run}\ M$

$M$ is any data term

Process definitions $M \Leftarrow P$

**Determined by assertions!**

Can be global
local
dynamic
parameterised
communicated

New syntax $\mathbf{run}\ M$

*M* is any data term

Process definitions $M \Leftarrow P$

**Determined by assertions!**

Can be global
local
dynamic
parameterised
communicated

New semantic rule (assertions elided)

$$\frac{M \Leftarrow P \qquad P \xrightarrow{\alpha} P'}{\mathbf{run}\ M \xrightarrow{\alpha} P'}$$

$$\frac{M \Leftarrow P \qquad\qquad P \xrightarrow{\alpha} P'}{\mathbf{run}\ M \xrightarrow{\alpha} P'}$$

Now re-prove all the theory!

$$\frac{M \Leftarrow P \qquad P \xrightarrow{\alpha} P'}{\mathbf{run}\ M \xrightarrow{\alpha} P'}$$

Now re-prove all the theory!

With Isabelle: took a day and a night

# Recent Developments II

**Broadcast communication**

# Recent Developments II

**Broadcast communication**

One transmission : many listeners

Channels with dynamic connectivity

# Recent Developments II

**Broadcast communication**

One transmission : many listeners

Channels with dynamic connectivity

Six new semantic rules, two new kinds of action

$$\textsc{BrOut}\ \frac{\Psi \vdash M \mathrel{\dot{\prec}} K}{\Psi \rhd \overline{M}\,N\,.\,P \xrightarrow{!K\,N} P} \qquad \textsc{BrIn}\ \frac{\Psi \vdash K \mathrel{\dot{\succ}} M}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N\,.\,P \xrightarrow{?K\,N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y}:=\widetilde{L}]}$$

$$\textsc{BrMerge}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{?K\,N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{?K\,N} Q'}{\Psi \rhd P \mid Q \xrightarrow{?K\,N} P' \mid Q'}$$

$$\textsc{BrCom}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{!K\,(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{?K\,N} Q'}{\Psi \rhd P \mid Q \xrightarrow{!K\,(\nu\widetilde{a})N} P' \mid Q'}\ \widetilde{a}\#Q$$

$$\textsc{BrOpen}\ \frac{\Psi \rhd P \xrightarrow{!K\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{!K\,(\nu\widetilde{a}\cup\{b\})N} P'}\ \begin{matrix} b\#\widetilde{a},\Psi,K \\ b \in \mathrm{n}(N) \end{matrix}$$

$$\textsc{BrClose}\ \frac{\Psi \rhd P \xrightarrow{!K\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\tau} (\nu b)(\nu\widetilde{a})P'}\ \begin{matrix} b \in \mathsf{n}(K) \\ b\#\Psi \end{matrix}$$

Quite hard to get it right

# Benefit of Isabelle

What about combining broadcast and higher-order?

Re-prove all theory yet again, or just say ”these extensions don’t interact” (wild handwaving)

# Benefit of Isabelle

What about combining broadcast and higher-order?

Re-prove all theory yet again, or just say ”these extensions don’t interact” (wild handwaving)

With Isabelle, took half a day and a cup of tea

Robin was unique.
Look at this, from just before he got the Turing award. I was about to visit for a postdoc and was worried about the house I would live in.

Joachim,
We went to see your house (Mrs Cameron), whose husband is the (church) minister at Liberton. They are very nice, and Lucy and I think that the house would do you well. It's comfortable but not beautiful, I'd say.

.

.

Altogether we feel (not knowing the price) that you would do well to take it if it doesn't seem too expensive. I think it's pleasant and convenient. Let me know if I can check on anything. I'm prepared to take it in your name if that seems appropriate; no doubt you'll discuss that by phone with the Camerons.

Happy 1989 -- Robin.

```
 ---------------------             --------------------      -----
|                     |<...2m...> |                    |      :
| <.......3.5m.......>|           | <....3.5m........> |      :
|                    /             \                   |      :
|      sitting      /               \                  |      :
|          room       |   /       |    bedroom         |     4m
|                     |---    ----|                    |      :
|                     |  porch    |                    |      :
|                     |   /       |                    |      :
 -------------------------     -------------------------     ---
|                                                      |
|                                                      |
|                                                      |
|                                                      |
|                                                      |
|                                                      |
|                                                      |
|               front garden                           |
|                                                      |
|                                                      |
|                                                      |
|                                                      |
 --------------------|           |---------------------


                    STREET
```

Happy 1989 -- Robin.

```
                         PLAN

    ~~~  ~~   ~~~~         ~~   ~~~
|                                                      |
|             (garden of people upstairs; they         |     N  <---->  S
|               have stairs leading down to it)        |
|                                                      |
 ------------------------------------------------------
|        |     |                                       |
|        | shed|                                       |
|         -----                                        |
|-----                                                 |
|     |                                                |
|     |                                                |
|     ----------                                       |
|              |           Your garden                 |
|  kitchen     |             (mostly paved)            |
|              |                                       |
|             /                                        |
|            /                                         |
|              |                                       |
|   \          |                                       |
|    --------------------------------------------------      ---
|              |        |             |                |      :
| sitting      |        |             |                |      :
|      room    | bath   |   bedroom   |                |      :
|              |        |             |  bedroom       |      :
|              |        |             |                |     4m
|              |        |             |                |      :
|              |        |             |                |      :
|              |  /     |  /          |  /             |      :
|              |--    -----    -----------    ---------|     -----
|              |                                    |   |        :
|            \                                     | cupboard   2m
|             \                                    |   |        :
 ---------------------             --------------------      -----
|                     |<...2m...> |                    |      :
| <.......3.5m.......>|           | <....3.5m........> |      :
|                    /            \                    |      :
|      sitting      /              \                   |      :
|          room     |   /       |    bedroom           |     4m
|                     |---    ----|                    |      :
|                     |  porch    |                    |      :
|                     |   /       |                    |      :
 -------------------------     -------------------------     ---
|                                                      |
```

Manually typed in by Robin just to assure me that my house would be good enough for my family.

*Photo from Jan. 1986*