Advanced Process Calculi

Lecture 2: psi-calculi

Copenhagen, August 2013

Joachim Parrow

The pi-calculus

$$\begin{array}{l}\overline{a}u.P\\ a(x).P\\ [x=y]P\\ [x\neq y]P\\ P\,|\,Q\\ P+Q\\ (\nu z)P\\ !P\end{array}$$

a,...,z just names (atomic things with an identity)

names can move, be scoped and tested for identity

Well understood semantics

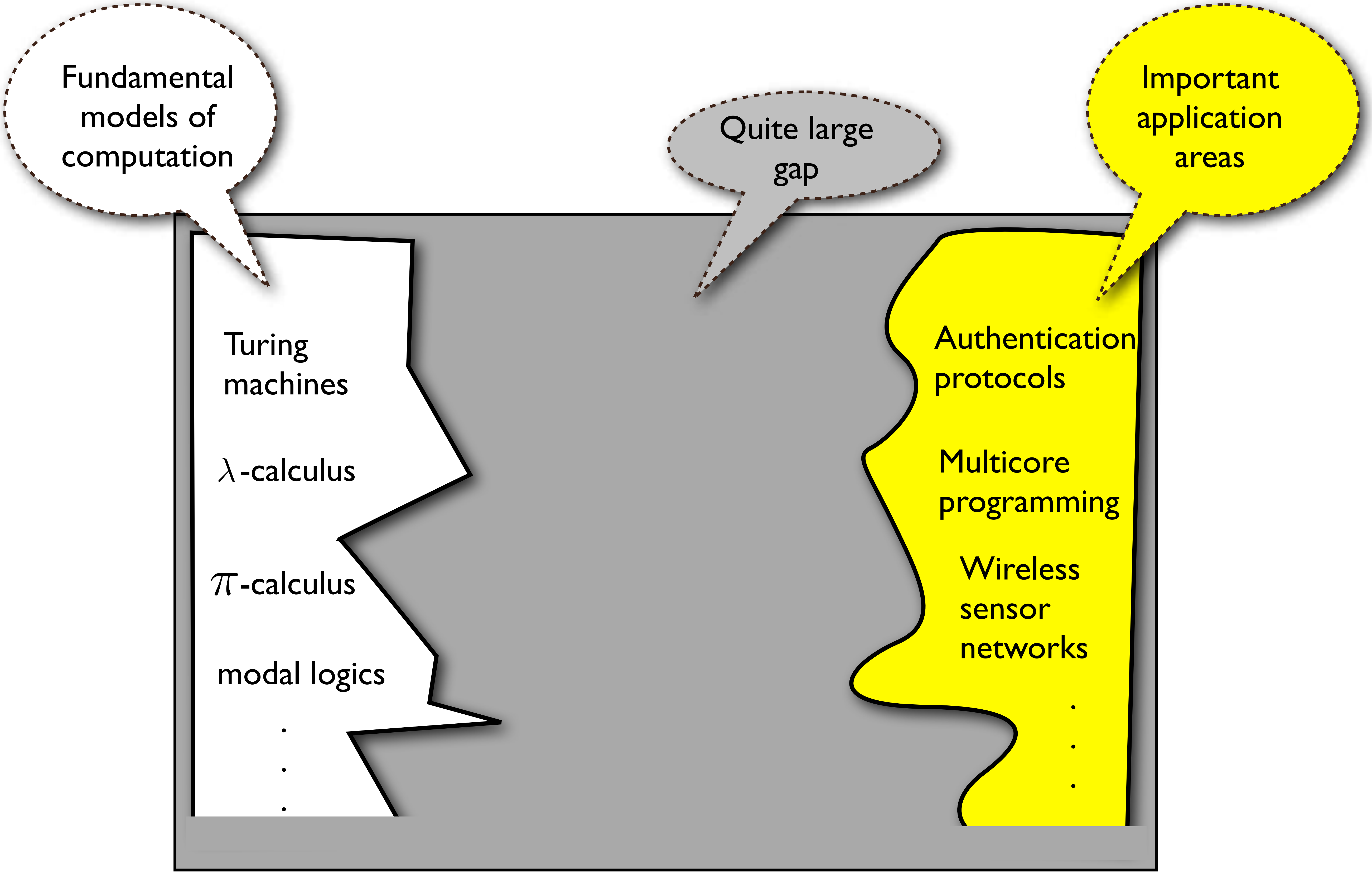
Highly expressive

So the quest is over?

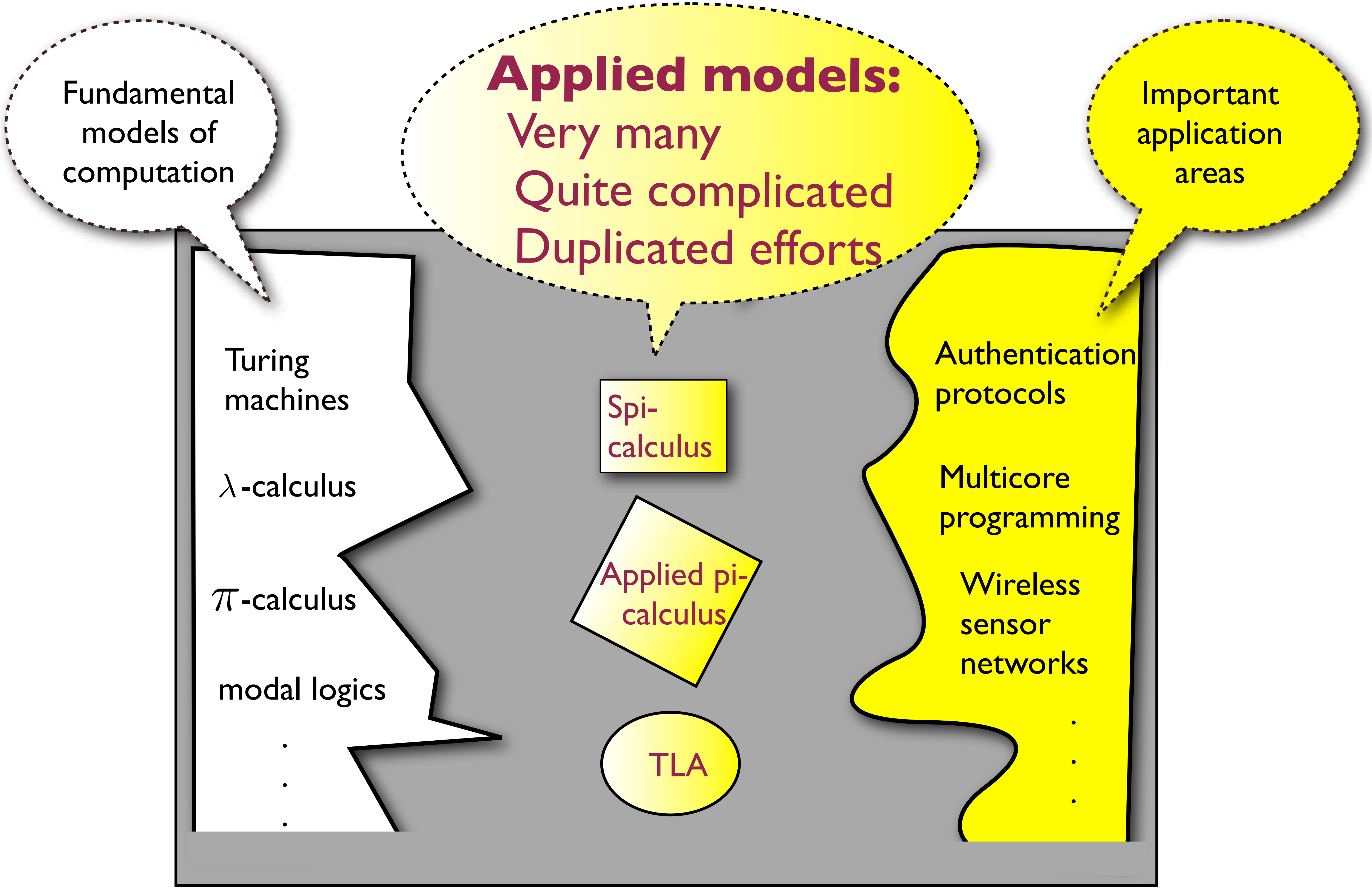
Not directly usable in application projects.

In the same way as turing machines or pure lambda-calculus

Problem


Fundamental models of computation
Quite large gap
Important application areas
Turing machines
λ-calculus
π-calculus
modal logics
. . .
Authentication protocols
Multicore programming
Wireless sensor networks
. . .

Situation


Fundamental models of computation
Applied models:
Very many
Quite complicated
Duplicated efforts
Important application areas
Turing machines
λ-calculus
π-calculus
modal logics
. . .
Spi-calculus
Applied pi-calculus
TLA
Authentication protocols
Multicore programming
Wireless sensor networks
. . .

Applied models

- Data structures, eg integers, lists, etc

$$a(x).b(y).\overline{c}\langle x+y\rangle \cdots$$

- Structures of names (tuples, aliases) etc

- Implicit logical theories

$$\forall A, B.\ \texttt{first}(\texttt{pair}(A, B)) = A$$
$$\forall k, M.\ \texttt{decode}(\texttt{encode}(k, M), k) = M$$
$$A \subseteq A \cup B$$
$$\forall y. \exists x.\ x > y^2$$
$$y \notin F \Rightarrow \lambda x.F = \lambda y.F[y := x]$$

Example: extend pi with integers

$$a(x).b(y).\overline{c}\langle x+y\rangle \cdots$$

Example: extend pi with integers

$$a(x).b(y).\overline{c}\langle x + y\rangle \cdots$$

$$\xrightarrow{a8} b(y).\overline{c}\langle 8 + y\rangle \cdots$$

Example: extend pi with integers

$$a(x).b(y).\overline{c}\langle x + y\rangle \cdots$$

$$\xrightarrow{a8} b(y).\overline{c}\langle 8 + y\rangle \cdots$$

$$\xrightarrow{b5} \overline{c}\langle 8 + 5\rangle \cdots$$

Example: extend pi with integers

$$
\begin{aligned}
&a(x).b(y).\overline{c}\langle x+y\rangle \cdots \\
&\xrightarrow{a8} b(y).\overline{c}\langle 8+y\rangle \cdots \\
&\xrightarrow{b5} \overline{c}\langle 8+5\rangle \cdots \\
&= \overline{c}\langle 13\rangle \cdots
\end{aligned}
$$

Looks simple, but

$\overline{a}5\,.\,\mathbf{0} \mid a(b)\,.\,\overline{b}3 \cdots$

Looks simple, but

$$\overline{a}5 \,.\, \mathbf{0} \mid a(b) \,.\, \overline{b}3 \cdots$$

$$\xrightarrow{\tau} \mathbf{0} \mid \overline{5}3 \cdots \quad ????$$

Looks simple, but

$$\overline{a}5\,.\,\mathbf{0} \mid a(b)\,.\,\overline{b}3\cdots$$

$$\xrightarrow{\tau} \mathbf{0} \mid \overline{5}3\cdots \quad ????$$

We probably need a type system to prevent this!

Looks simple, but

$$\overline{a}5\,.\,\mathbf{0} \mid a(b)\,.\,\overline{b}3 \cdots$$

$$\xrightarrow{\tau} \mathbf{0} \mid \overline{5}3 \cdots \quad ????$$

We probably need a type system to prevent this!

Also, integers should not be scoped?

$$(\nu 5)\overline{a}5\,.\,\mathbf{0} \mid a(x)\,.\,\overline{b}\langle x+8\rangle \;\; ????$$

Looks simple, but

$$\overline{a}5\,.\,\mathbf{0} \mid a(b)\,.\,\overline{b}3\cdots$$

$$\xrightarrow{\tau} \mathbf{0} \mid \overline{5}3\cdots \quad ????$$

We probably need a type system to prevent this!

Also, integers should not be scoped?

$$(\nu 5)\overline{a}5\,.\,\mathbf{0} \mid a(x)\,.\,\overline{b}\langle x+8\rangle \quad ????$$

Does it matter exactly what the evaluation of expressions is?

Is an extension OK?

- Encode integers in pure pi (possible in roughly the same way as Church numerals in the lambda calculus).
- Or, redo **all** the theory. All proofs of compositionality etc need to be rechecked.

Example: name tuples

$a(x_1, \ldots, x_n) \,.\, P$ Input a tuple

$\overline{a}\langle x_1, \ldots, x_n \rangle \,.\, P$ Output a tuple

$$\overline{a}\langle b, c \rangle \,.\, \mathbf{0} \mid a(x, y) \,.\, x(u) \,.\, \overline{y}u \,.\, \mathbf{0}$$

Example: name tuples

$a(x_1, \ldots, x_n) \,.\, P$ Input a tuple

$\overline{a}\langle x_1, \ldots, x_n \rangle \,.\, P$ Output a tuple

$$\overline{a}\langle b, c\rangle \,.\, \mathbf{0} \mid a(x, y) \,.\, x(u) \,.\, \overline{y}u \,.\, \mathbf{0}$$

$$\xrightarrow{\tau} \mathbf{0} \mid b(u) \,.\, \overline{c}u \,.\, \mathbf{0}$$

Looks simple, but

$$\overline{a}\langle b, c\rangle \,.\, \mathbf{0} \mid a(x) \,.\, \mathbf{0} \quad \xrightarrow{\tau} ?????$$

Looks simple, but

$$\overline{a}\langle b, c\rangle \,.\, \mathbf{0} \mid a(x) \,.\, \mathbf{0} \quad \xrightarrow{\tau} ?????$$

Need type system again ? Or just decide this has no transition?

Looks simple, but

$$\overline{a}\langle b, c\rangle \,.\, \mathbf{0} \mid a(x) \,.\, \mathbf{0} \quad \xrightarrow{\tau} ?????$$

Need type system again ? Or just decide this has no transition?

Does it matter if substitution is simultaneous or sequential?

Looks simple, but

$$\overline{a}\langle b, c\rangle \,.\, \mathbf{0} \mid a(x) \,.\, \mathbf{0} \quad \xrightarrow{\tau} ?????$$

Need type system again ? Or just decide this has no transition?

Does it matter if substitution is simultaneous or sequential?

Can tuples occur as channels? $\overline{\langle a, b\rangle} c \,.\, \mathbf{0}$????

Aliases

$\{^M/_x\}$ An agent that declares that x is an alias for M (some data term)

enc = encrypt
dec = decrypt
dec(enc(m,k),k) = m

Aliases

$\{^{M}\!/_{x}\}$ An agent that declares that x is an alias for M (some data term)

$$Q = (\nu x, k)(\{^{\mathsf{enc}(m,k)}\!/_{x}\} \mid \overline{a}x\,.\,P')$$

enc = encrypt
dec = decrypt
dec(enc(m,k),k) = m

Aliases

$\{^{M}\!/\!_{x}\}$ An agent that declares that x is an alias for M (some data term)

$$Q = (\nu x, k)(\{^{\mathsf{enc}(m,k)}\!/\!_{x}\} \mid \overline{a}x \,.\, P')$$

$$\xrightarrow{\overline{a}\ (\nu x)x} (\nu k)(\{^{\mathsf{enc}(m,k)}\!/\!_{x}\} \mid P')$$

enc = encrypt
dec = decrypt
dec(enc(m,k),k) = m

Aliases

$\{^{M}/_{x}\}$ An agent that declares that x is an alias for M (some data term)

$$Q = (\nu x, k)(\{^{\mathsf{enc}(m,k)}/_{x}\} \mid \overline{a}x \, . \, P')$$

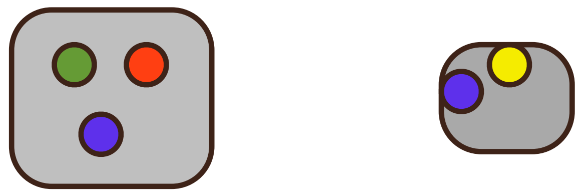
$$\xrightarrow{\overline{a}\ (\nu x)x} (\nu k)(\{^{\mathsf{enc}(m,k)}/_{x}\} \mid P')$$

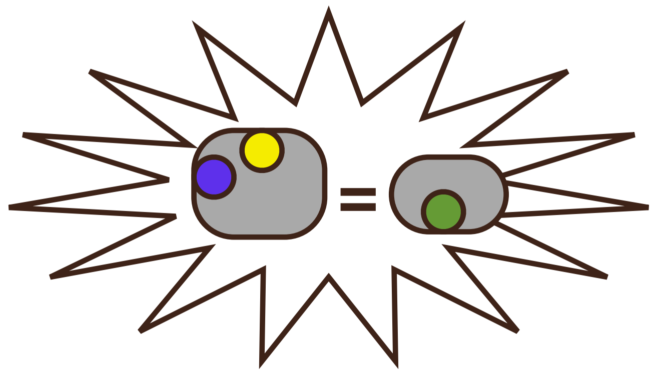
enc = encrypt
dec = decrypt
dec(enc(m,k),k) = m

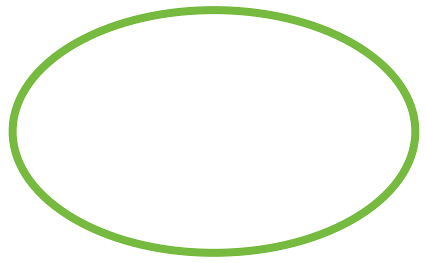
The environment receives x - a kind of ”ciphertext”, but not until it also receives k can it infer dec(x,k)=m

Elements:

Names Things that can be scoped

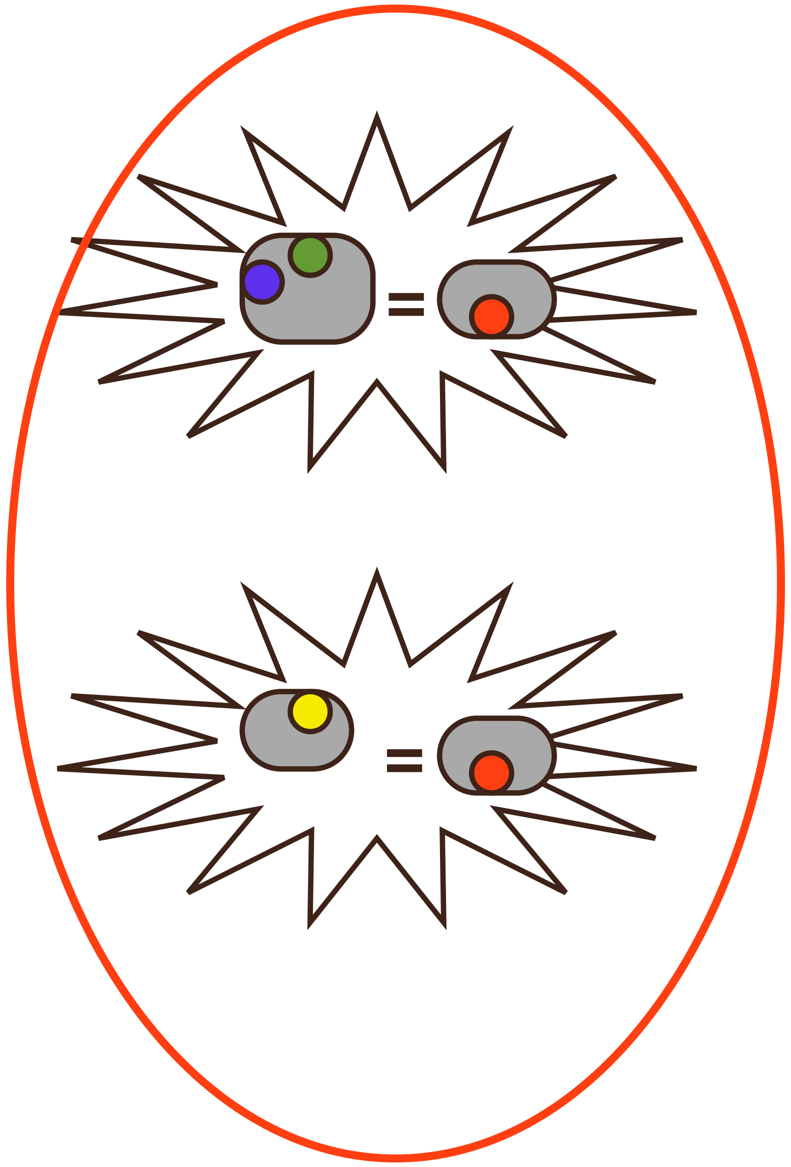
Data  Can contain names. Processes can create and compute on data.

Assertions  To express facts about data, eg aliases.

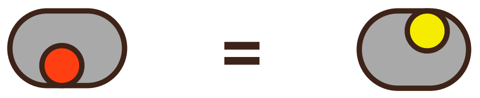
Scopes 

Example

A so called frame: assertion(s) enclosed by a scope

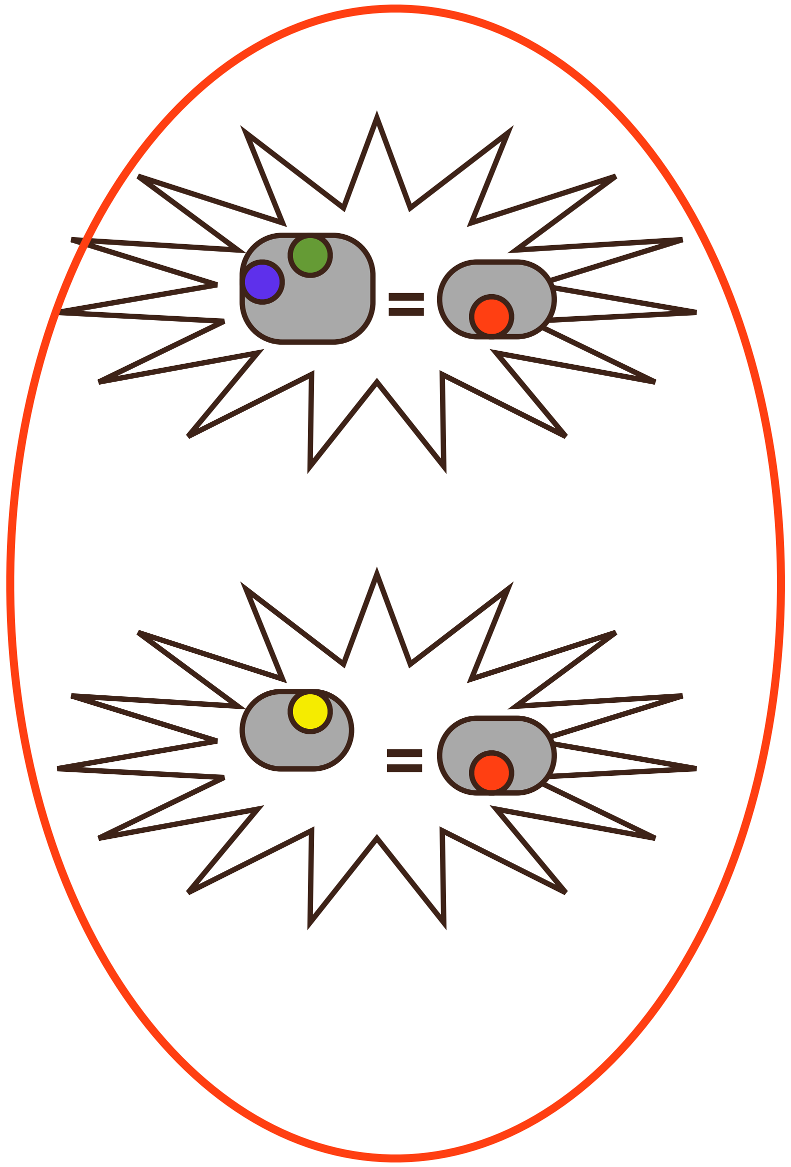


Otside the scope we cannot infer

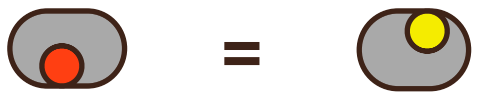


Example

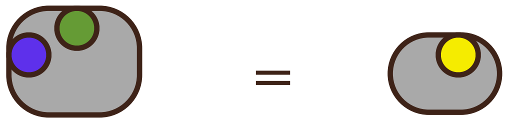
A so called frame: assertion(s) enclosed by a scope



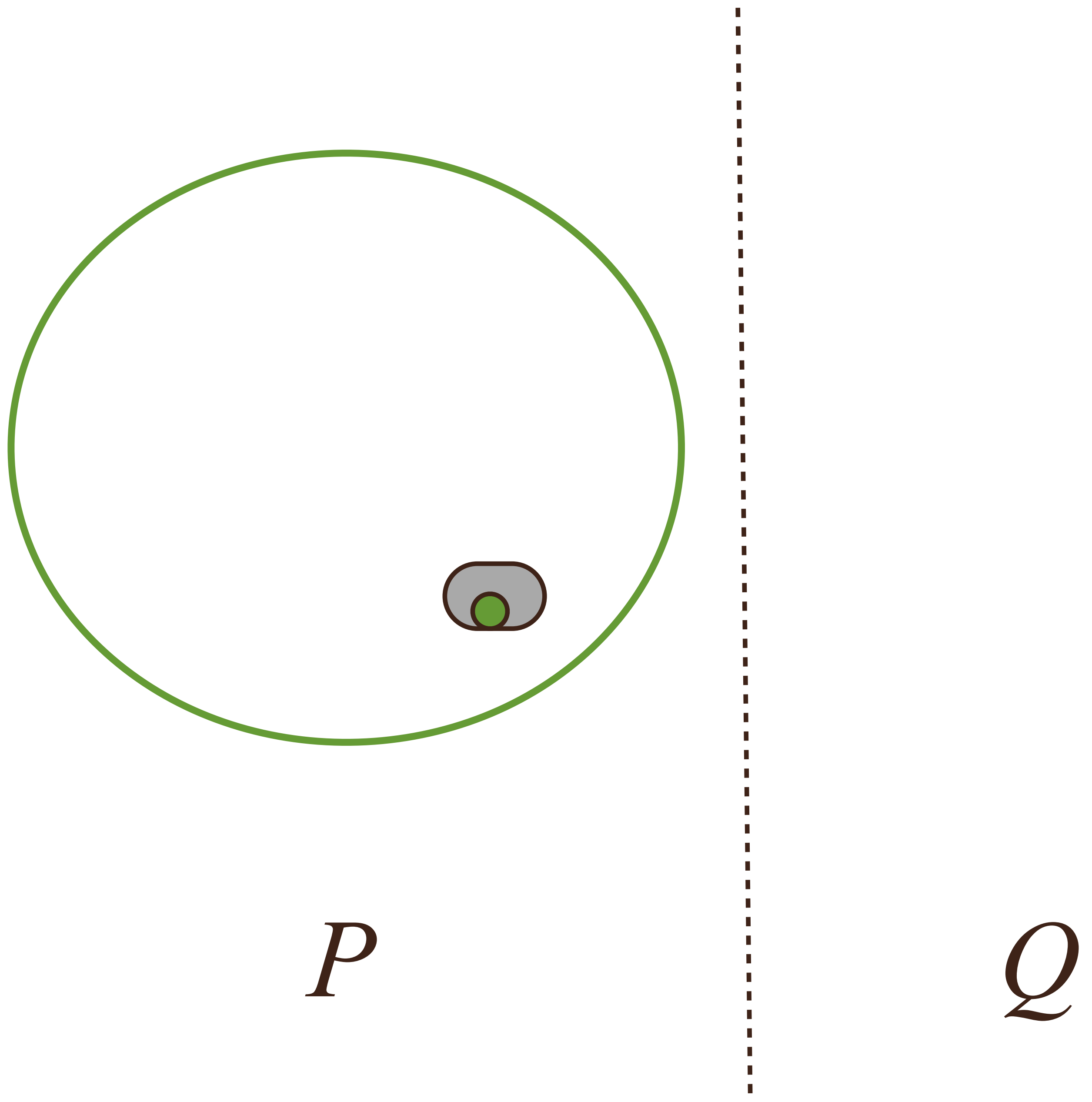
Otside the scope we cannot infer



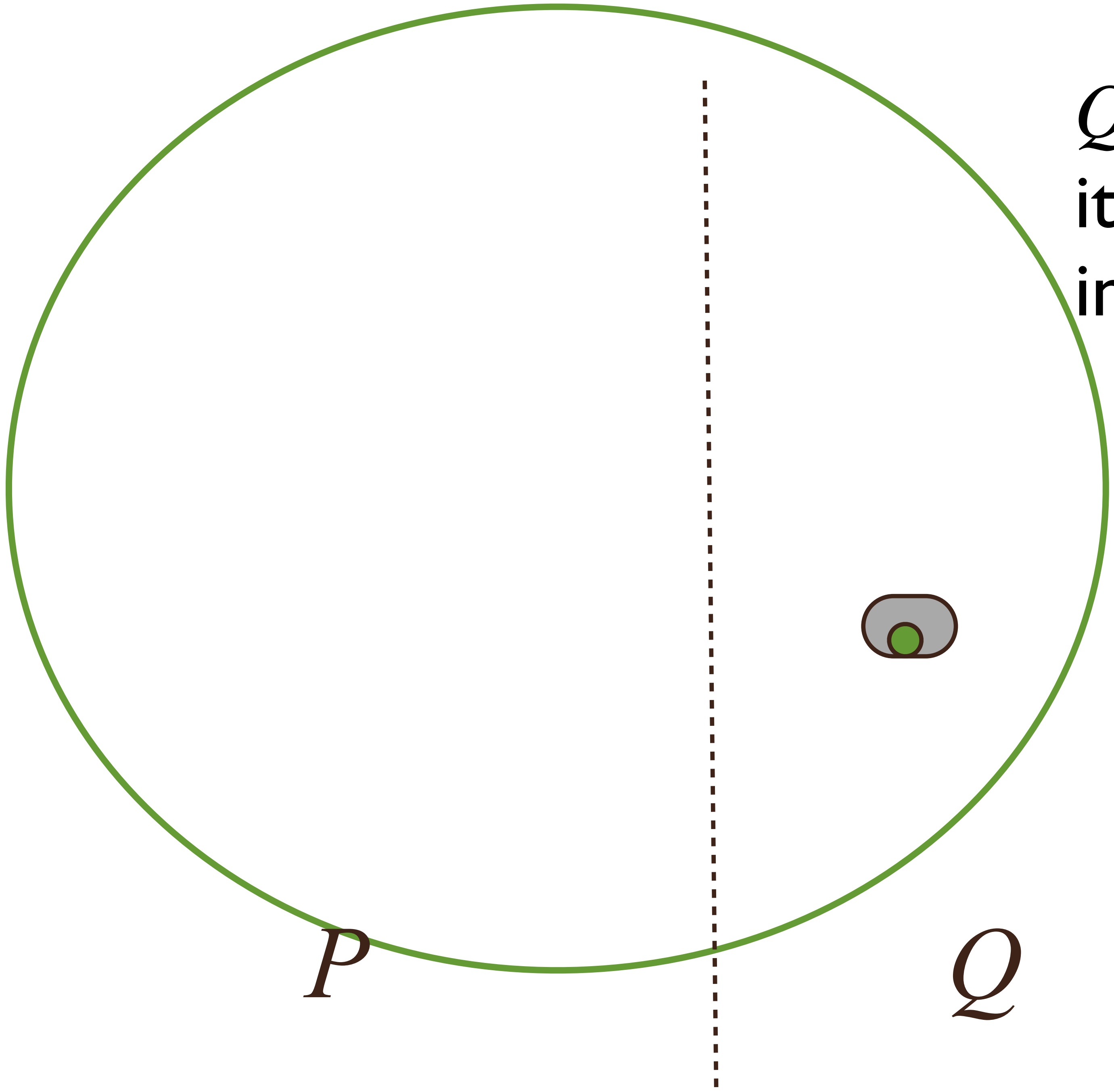
But we can infer



Sending a private item

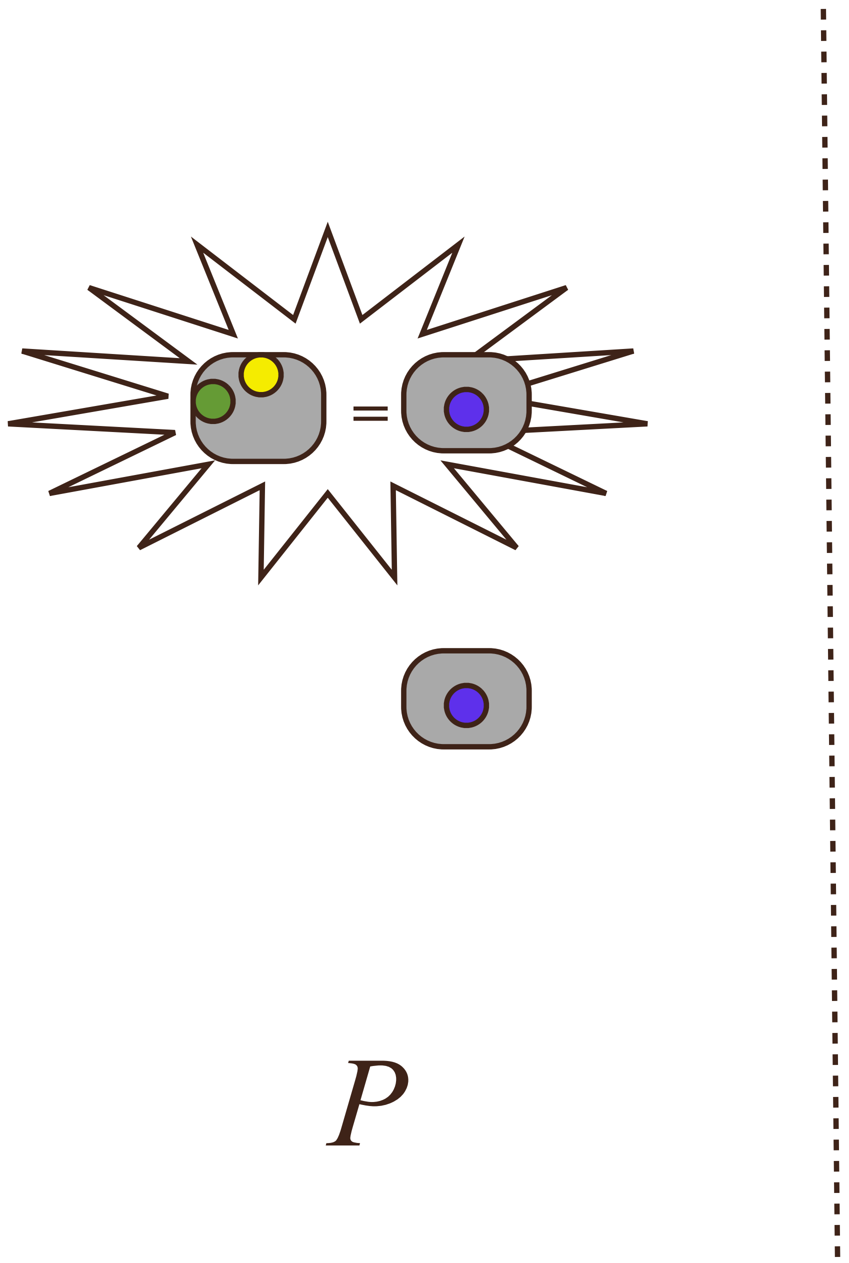


Sending a private item



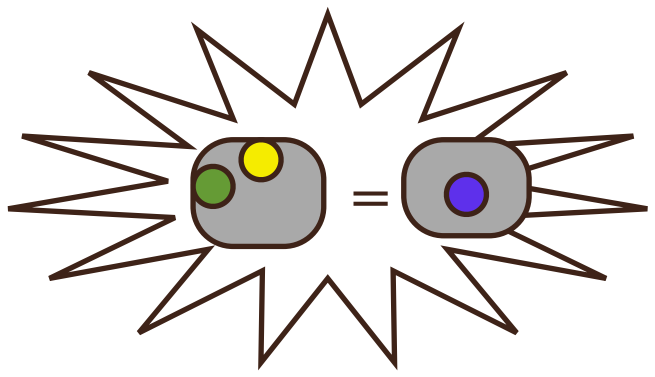
Q can now use the item and is included in its scope

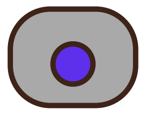
Sending an alias



Sending an alias

Q has received an item and can infer that ● is part of it

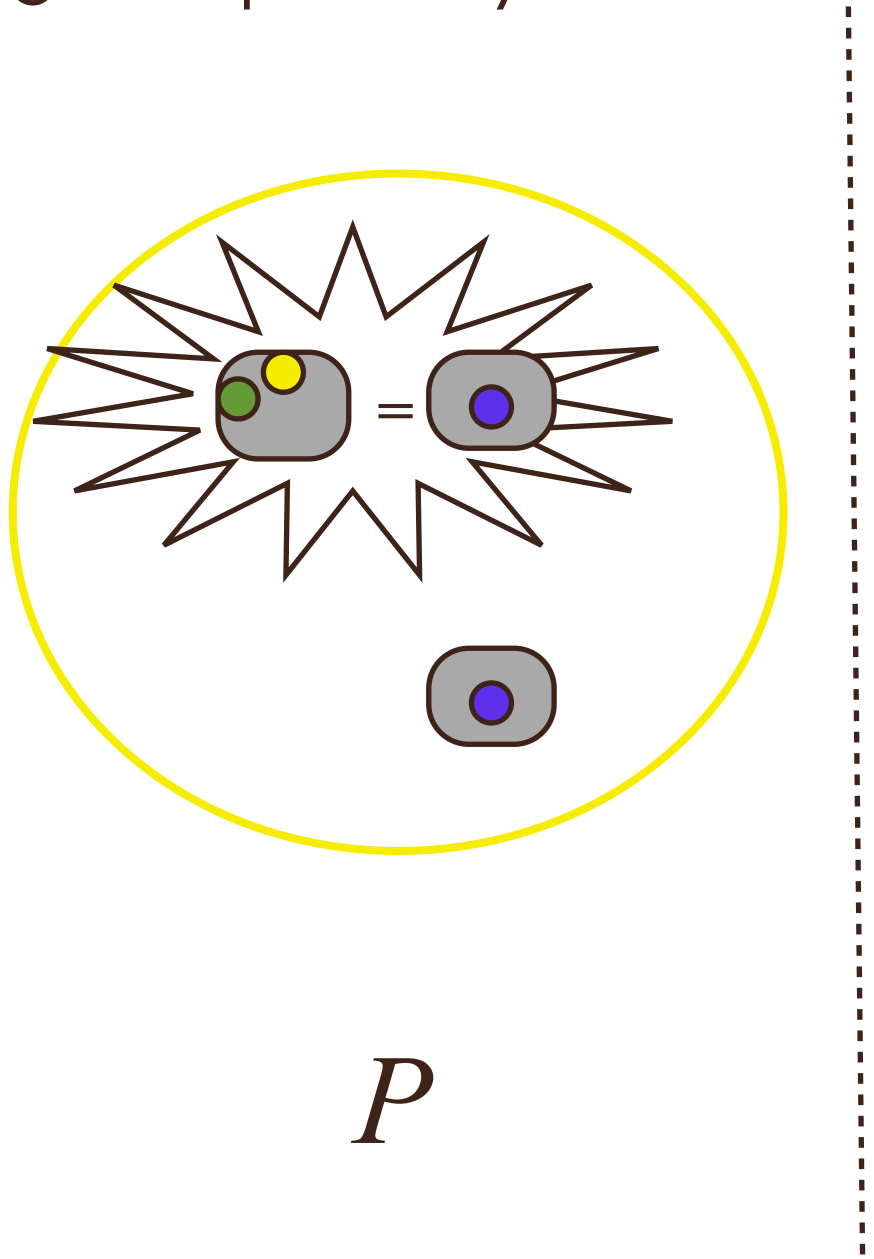




P Q

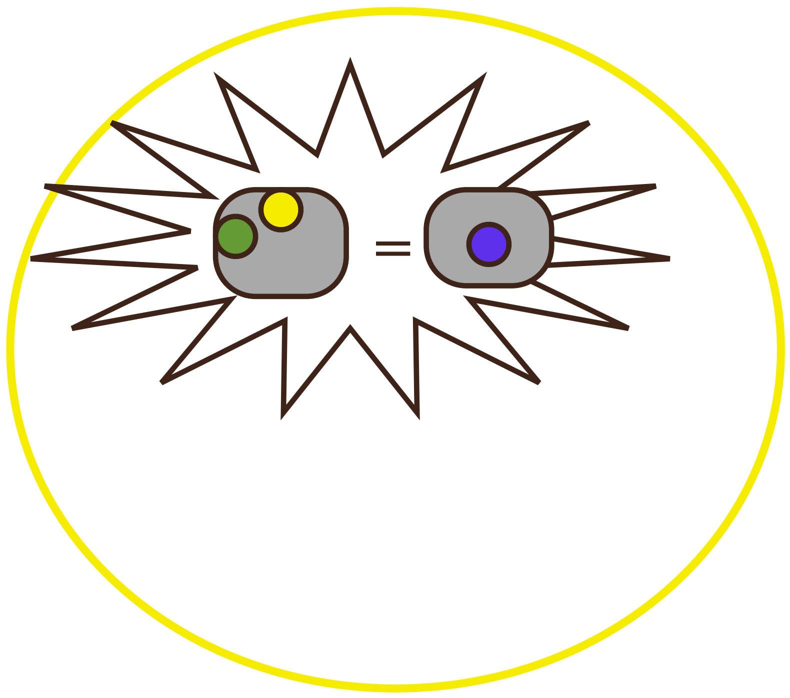
Sending a ciphertext

● is a ”private key”



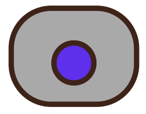
Sending a ciphertext

● is a "private key"



P

Q has received an item and cannot infer that ● is part of it

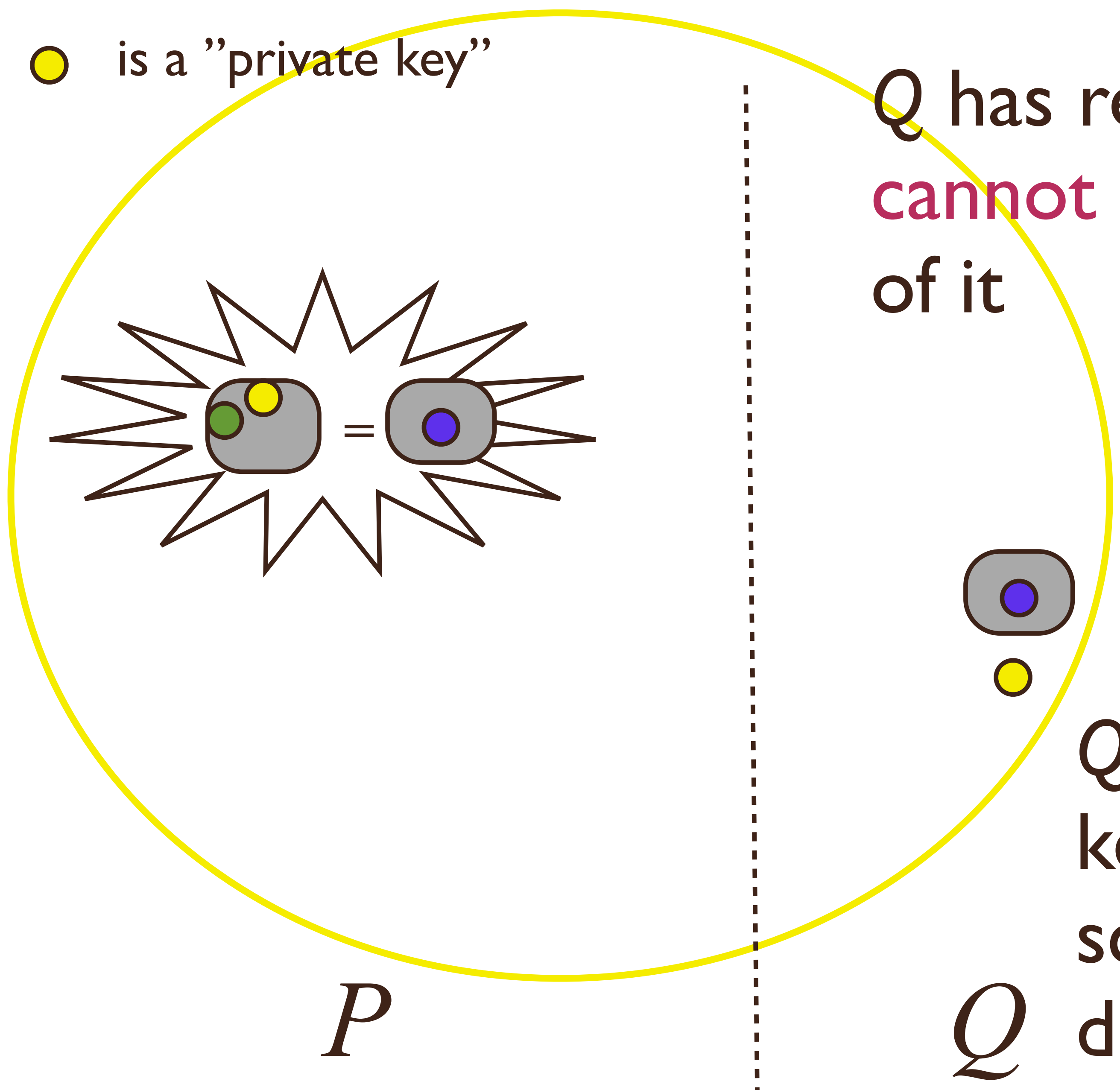


Q

Sending a ciphertext

● is a "private key"

Q has received an item and cannot infer that ● is part of it



Q has now received the key and is included in its scope, and can therefore decipher.

What is the limit?

Lists and tables of channel names?

Data type constructors?

Higher-order data

Sending functions on integers?

Sending functions on channel names?

Predicate logic in tests?

and so on

and so on

and so on

What is the limit?

Lists and tables of channel names?

Data type constructors?

Higher-order data

Sending functions on integers?

Sending functions on channel names?

Predicate logic in tests?

and so on

and so on

and so on

First axiom of applied calculi:

For any such extension there is an application that profits from it!

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

- Theory requires a lot of stamina to redo
- Subtly related to similar efforts
- But cannot formally re-use earlier efforts
- And needs to be redone over and over

=> Errors creep in

Example: applied pi

Abadi, Fournet 2001

Extend pi-calculus with algebraic datatypes and aliases

$$\{^{M}/_{x}\}$$

Output of single names. Instead of $\overline{a}M\,.\,P$ write

$$(\nu x)\overline{a}x\,.\,(P \mid \{^{M}/_{x}\})$$

Heavily used for desription and analysis
of security protocols (~700 citations)

Applied pi

Example that it is **not compositional**

$A = (\nu a)(\{{}^{a}/_{x}\} \mid x.b.\mathbf{0}) \qquad B = (\nu a)(\{{}^{a}/_{x}\} \mid \mathbf{0})$

A and B have the same transitions (none)

and frame (namely $(\nu a)\{{}^{a}/_{x}\}$)

Applied pi

Example that it is **not compositional**

$$A = (\nu a)(\{^a/_x\} \mid x.b.\mathbf{0}) \qquad B = (\nu a)(\{^a/_x\} \mid \mathbf{0})$$

A and B have the same transitions (none)

and frame (namely $(\nu a)\{^a/_x\}$)

But compose with $R = \overline{x}.\mathbf{0}$ and apply scope extension:

$$A|R \equiv (\nu a)(\{^a/_x\} \mid x.b.\mathbf{0} \mid \overline{x}.0)$$

This has an interaction, but $B|R$ has not!

Another example: CC-pi

Buscemi, Montanari 2007

- A formalism for concurrent constraints
- Agents can communicate and have a constraint store
- **ask** and **tell** primitives
- Constraints form a semiring (ie has conjunction, disjunction and satisfies the ring axioms except inverse)
- ~ 140 citations

Example: CC-pi

Example that it is not compositional

$$P = (\nu b, x)(x = b \mid \overline{a}x \,.\, b \,.\, c) \qquad Q = (\nu b, x)(x = b \mid \overline{a}x)$$

A constraint that x is b

P and Q have the same store, and the same transitions (opening the scope of x and then nothing more), but composed with $R = a(y).\overline{y}$ there are further transitions for P but not Q

A plethora of calculi

SPEECH
ZONE

All-purpose calculus?

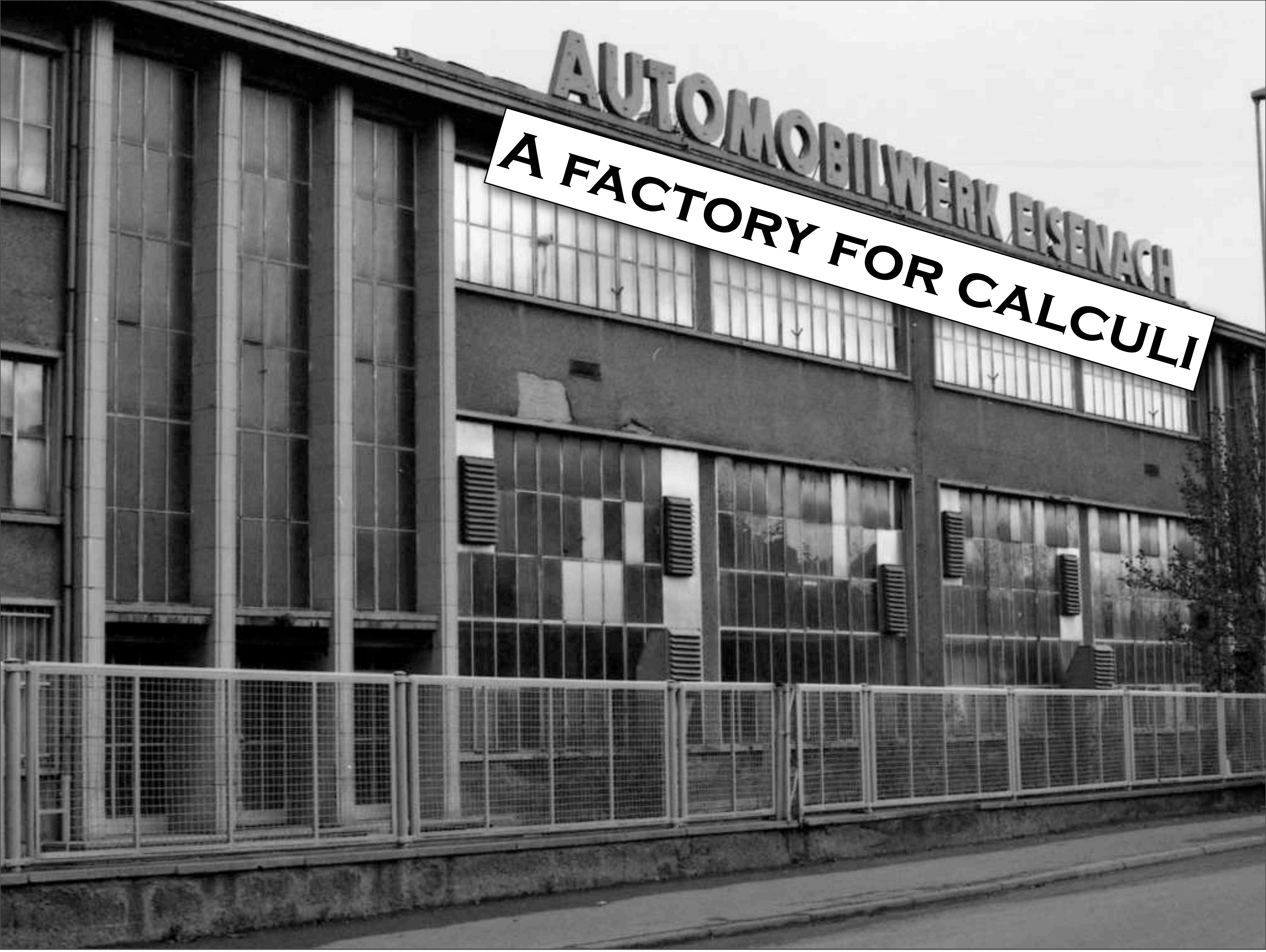
Just one problem:

All-purpose calculus?

Just one problem:

In real life there is no such

A FACTORY FOR CALCULI



Psi-calculi framework

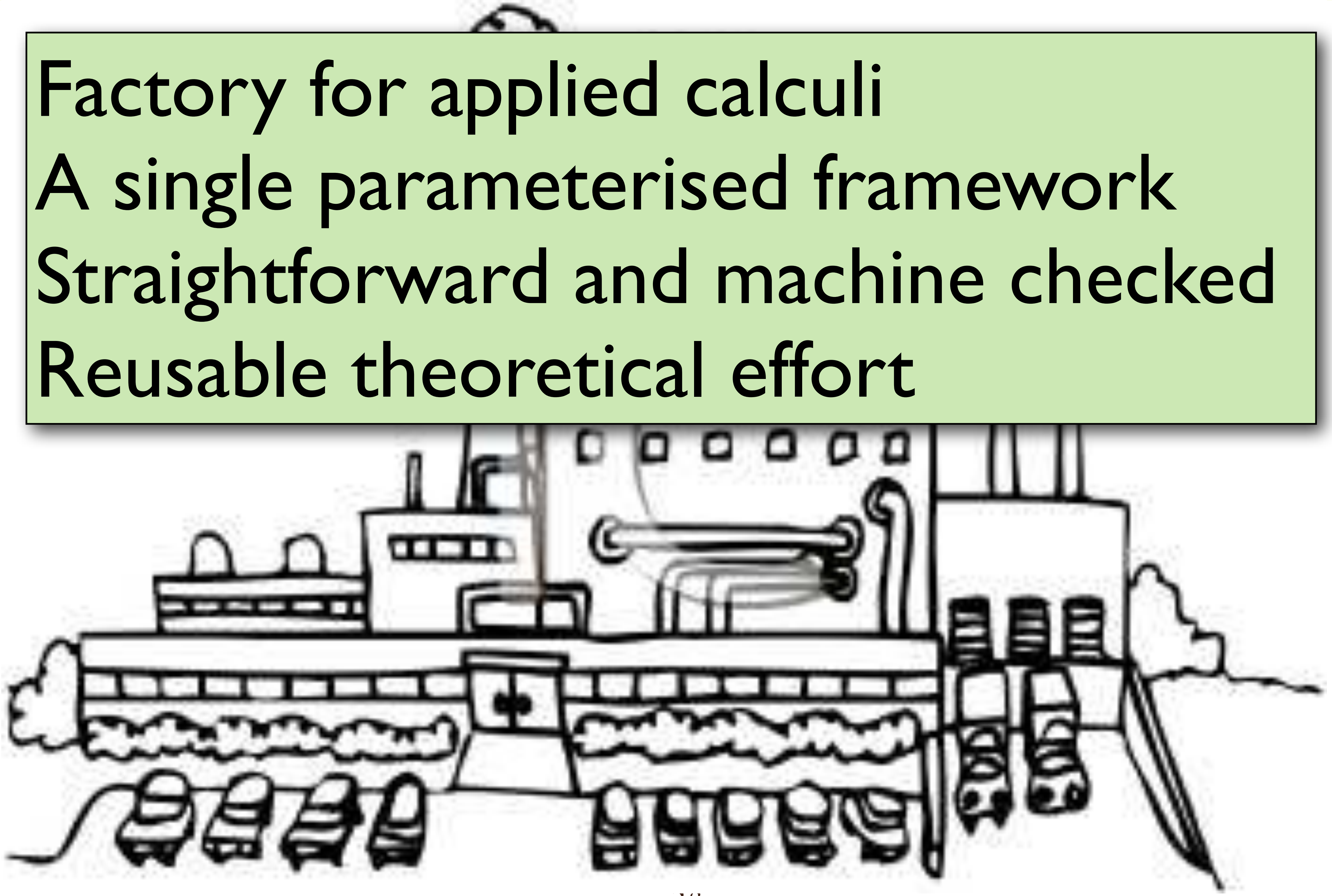
Originally from 2008 by Bengtson, Johansson, Parrow, Victor

Lately contributions also by Borgström, Gutkovas, Raabjerg, Weber, Åman-Pohjola

Psi-calculi framework

Originally from 2008 by Bengtson, Johansson, Parrow, Victor

Lately contributions also by Borgström, Gutkovas, Raabjerg, Weber, Åman-Pohjola



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

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

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

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

assertions, ie facts about data used to resove predicates

Psi-calculi

Aim for maximum generality!

Data structures that can be (almost) arbitrary sets, including inductively defined data types, higher-order types including binders, sets defined by comprehension etc.

These may contain names which can be scoped and subjected to substitution

And any of them may be used to designate communication channels.

Psi-Calculi

General notion of assertions (aka constraints), conditions (aka tests) and entailment between them.

These can contain names and be anything, including eg inference systems in higher-order logics.

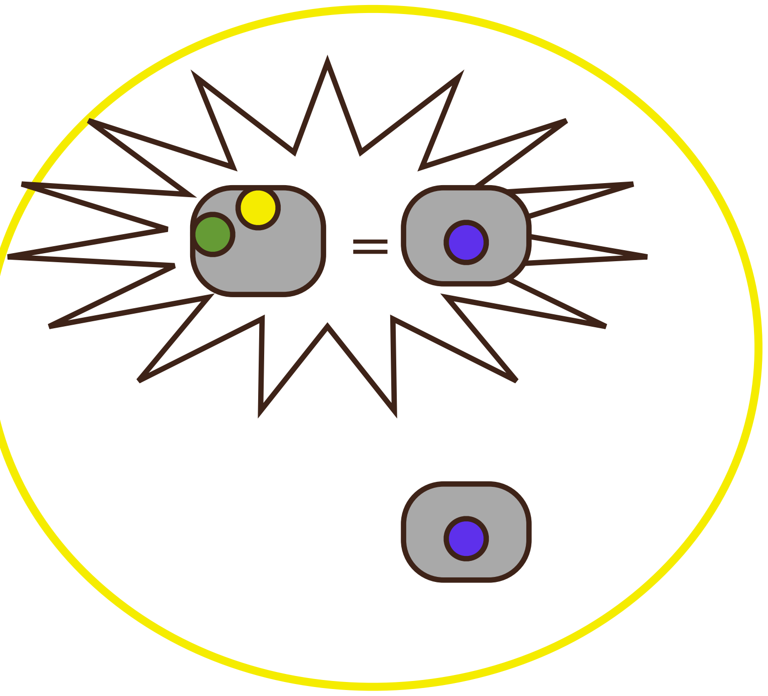
MAIN IDEA:

In order to define a psi-calculus, the user needs to define what are these data, assertions, conditions etc.

May differ between applications => different psi-calculi

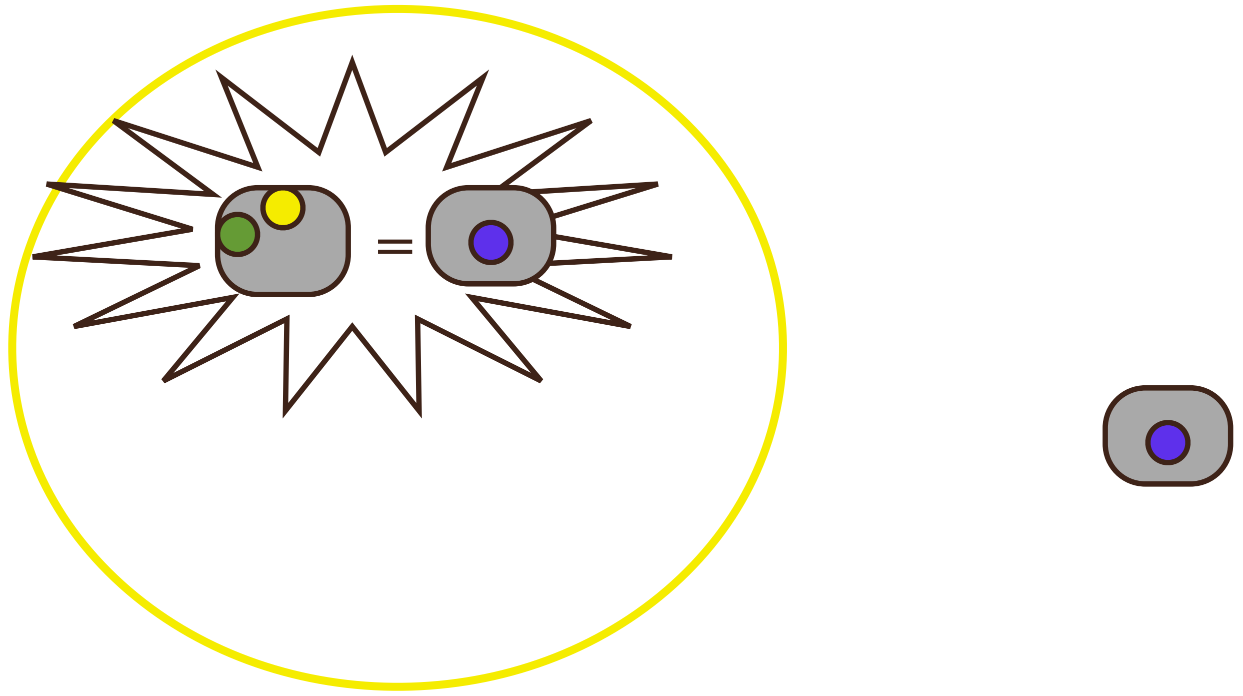
Can we really accommodate **any** set and **any** logic?

Well, almost. We do need to know eg:
In a given data value, which names are contained?
What is the effect of changing a name (alpha-conversion)
What is the effect of substituting a name?



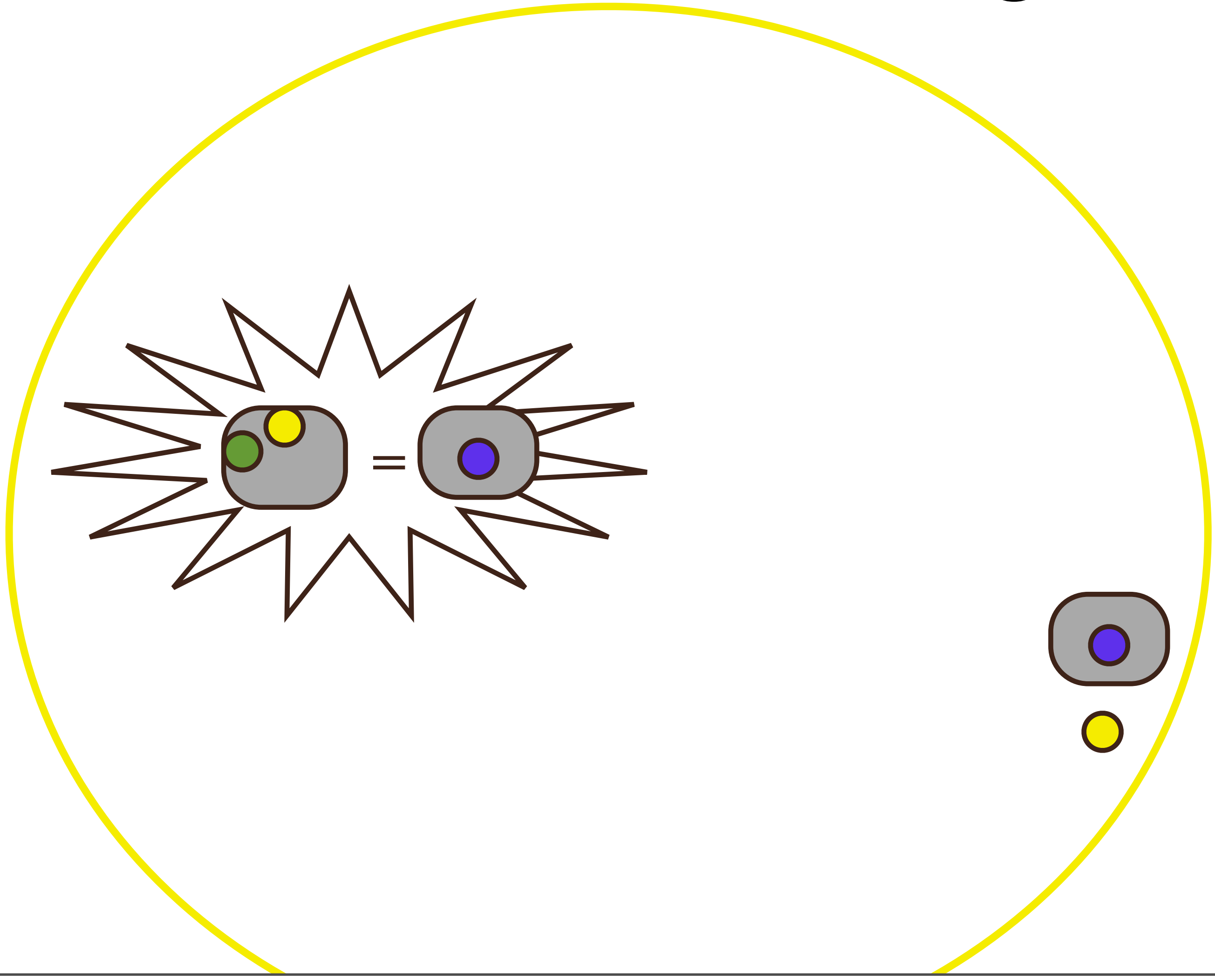
Can we really accommodate **any** set and **any** logic?

Well, almost. We do need to know eg:
In a given data value, which names are contained?
What is the effect of changing a name (alpha-conversion)
What is the effect of substituting a name?



Can we really accommodate **any** set and **any** logic?

Well, almost. We do need to know eg:
In a given data value, which names are contained?
What is the effect of changing a name (alpha-conversion)
What is the effect of substituting a name?



Nominal sets (Pitts 2000)

A nominal set is an ordinary set equipped with a group of permutation actions, intutively to permute names in members of the set.

We shall mainly use singular permuations written as *(a b)A*, meaning swapping the names *a* and *b* in *A*.

Example 1: Let names just be integers. Consider *P*(N) with the intuitive permutation function on its members, eg

$$(1\ 3)\{1,2,4\} = \{3,2,4\}$$

Nominal sets (Pitts 2000)

A nominal set is an ordinary set equipped with a group of permutation actions, intutively to permute names in members of the set.

We shall mainly use singular permuations written as *(a b)A*, meaning swapping the names *a* and *b* in *A*.

Example 1: Let names just be integers. Consider *P*(N) with the intuitive permutation function on its members, eg

(1 3){1,2,4} = {3,2,4}

(1 3){1,3,4} = {3,1,4} = {1,3,4}

Example 2:

Let names be variables of the lambda-calculus.

Consider the set of lambda terms quotiented by alpha-conversion, ie members are alpha-equivalence classes.

Swapping is textual replacement on any representative (we need to show that the choice of representative does not matter!) Use underline to denote alpha-equivalence class:

$$(x\ y)\underline{\lambda z.xx} = \underline{\lambda z.yy}$$

$$(x\ y)\underline{\lambda z.xy} = \underline{\lambda z.yx}$$

$$(x\ y)\underline{\lambda x.zx} = \underline{\lambda y.zy} = \underline{\lambda x.zx}$$

Here swapping has no effect!

Nominal set main idea

The carrier set and swapping action can be anything that satisfies:

$\boxed{(a\ a)P = P}$ identity swapping never has any effect

$\boxed{(a\ b)((a\ b)P) = P}$ swapping twice never has any effect

$\boxed{(a\ a')((b\ b')P) = ((a\ a')b\ (a\ a')b')((a\ a')P)}$

distributing swappings

In particular this is satisfied by textual replacement

Support

Given a nominal set, define a **support** of a member X to be a set of names S such that for all a,b not in S, $(a\ b)X = X$

Intuition: names outside a support cannot have effect in swappings

Example 1 continued (Let names just be integers.) Give examples of supports of {1,2,3}

Axiom: any member X of a nominal set must have a **finite** support.

Thm: any member X then has a **smallest** finite support, written $\mathrm{supp}(X)$.

Intuition: $\mathrm{supp}(X)$ contains exactly the names that can be affected by swappings in X

Example 1: Let names just be integers.

What is the support of the following?

{1}
{3,7,9}
{}
N - {3,7,9}
N - {1}
N
the set of
even numbers

Example 1: Let names just be integers.

What is the support of the following?

{1} {1}

{3,7,9}

{}

N - {3,7,9}

N - {1}

N

the set of
even numbers

Example 1: Let names just be integers.

What is the support of the following?

{1}

{3,7,9}

{}

N - {3,7,9}

N - {1}

N

the set of
even numbers

{1}

{3,7,9}

Example 1: Let names just be integers.

What is the support of the following?

{1}

{3,7,9}

{}

N - {3,7,9}

N - {1}

N

the set of even numbers

{1}

{3,7,9}

{}

Example 1: Let names just be integers.

What is the support of the following?

{1}	{1}
{3,7,9}	{3,7,9}
{}	{}
N - {3,7,9}	{3,7,9}
N - {1}	
N	
the set of even numbers	

Example 1: Let names just be integers.

What is the support of the following?

{1}	{1}
{3,7,9}	{3,7,9}
{}	{}
N - {3,7,9}	{3,7,9}
N - {1}	{1}
N	
the set of even numbers	

Example 1: Let names just be integers.

What is the support of the following?

{1}	{1}
{3,7,9}	{3,7,9}
{}	{}
N - {3,7,9}	{3,7,9}
N - {1}	{1}
N	{}
the set of even numbers	

Example 1: Let names just be integers.

What is the support of the following?

{1}	{1}
{3,7,9}	{3,7,9}
{}	{}
N - {3,7,9}	{3,7,9}
N - {1}	{1}
N	{}
the set of even numbers	has no finite and no minimal support. The set of even numbers is a support, as is the set of odd numbers. So this does not exist in a nominal set.

Example 2:
Let names be variables of the lambda-calculus.

Consider the set of lambda terms quotiented by alpha-conversion, ie members are alpha-equivalence classes.

$$\underline{\lambda x.xy}$$

Example 2:
Let names be variables of the lambda-calculus.

Consider the set of lambda terms quotiented by alpha-conversion, ie members are alpha-equivalence classes.

$\underline{\lambda x.xy}$

support = $\{y\}$.
x is not in the support since swapping x is an alpha-conversion and does not affect the alpha-equivalence class.

Example 2:
Let names be variables of the lambda-calculus.

Consider the set of lambda terms quotiented by alpha-conversion, ie members are alpha-equivalence classes.

$\underline{\lambda x.xy}$

support = $\{y\}$.
x is not in the support since swapping x is an alpha-conversion and does not affect the alpha-equivalence class.

In general, the support of an alpha-equivalence class is the set of free variables.

Nominal terminology

$\boxed{\mathrm{n}(X)}$ Short for $\mathrm{supp}(X)$

Nominal terminology

$\boxed{\mathrm{n}(X)}$ Short for $\mathrm{supp}(X)$

$\boxed{a \# X}$ ”a fresh in X”
Short for a not in $\mathrm{n}(X)$

Nominal terminology

$\boxed{\mathrm{n}(X)}$ Short for $\mathrm{supp}(X)$

$\boxed{a \# X}$ ”a fresh in X”
Short for a not in $\mathrm{n}(X)$

$x \# \lambda y.zy$

Nominal terminology

$\boxed{\mathrm{n}(X)}$ Short for $\mathrm{supp}(X)$

$\boxed{a \,\#\, X}$ "a fresh in X"
Short for a not in $\mathrm{n}(X)$

$$x \,\#\, \lambda y.zy$$

$$x \in \mathrm{n}(\lambda x.zx)$$

Nominal terminology

$\boxed{\mathrm{n}(X)}$ Short for $\mathrm{supp}(X)$

$\boxed{a \# X}$ "a fresh in X"
Short for a not in $\mathrm{n}(X)$

$$x \# \lambda y.zy$$

$$x \in \mathrm{n}(\lambda x.zx)$$

$$x \# \underline{\lambda x.zx}$$

Nominal terminology

A function f is **equivariant** if

$$f((a\ b)X) = (a\ b)f(X)$$

Similarly for higher arity functions and relations

Nominal terminology

A function f is **equivariant** if

$$f((a\ b)X) = (a\ b)f(X)$$

Similarly for higher arity functions and relations

For constant functions f this means $f() = (a\ b)f()$
ie $f()$ has empty support

Nominal terminology

A function f is **equivariant** if

$$f((a\ b)X) = (a\ b)f(X)$$

Similarly for higher arity functions and relations

For constant functions f this means $f() = (a\ b)f()$
ie $f()$ has empty support

Intuition: Something is equivariant if it does not treat any name special.

Example: which of these are equivariant?

$$f(X) = X \cup \{1\}$$

$$f(X) = X$$

$$f(X) = \mathrm{N} - X$$

$$f(X) = \{\}$$

$$f(X) = \{3, 7\}$$

Example: which of these are equivariant?

$$f(X) = X \cup \{1\}$$ X

$$f(X) = X$$

$$f(X) = \mathrm{N} - X$$

$$f(X) = \{\}$$

$$f(X) = \{3, 7\}$$

Example: which of these are equivariant?

$f(X) = X \cup \{1\}$ ✗

$f(X) = X$ ✓

$f(X) = \mathrm{N} - X$

$f(X) = \{\}$

$f(X) = \{3, 7\}$

Example: which of these are equivariant?

$f(X) = X \cup \{1\}$ ✗

$f(X) = X$ $\surd$

$f(X) = \mathrm{N} - X$ $\surd$

$f(X) = \{\}$

$f(X) = \{3, 7\}$

Example: which of these are equivariant?

$f(X) = X \cup \{1\}$ ✗

$f(X) = X$ ✓

$f(X) = \mathrm{N} - X$ ✓

$f(X) = \{\}$ ✓

$f(X) = \{3, 7\}$

Example: which of these are equivariant?

$f(X) = X \cup \{1\}$ ✗

$f(X) = X$ ✓

$f(X) = \mathrm{N} - X$ ✓

$f(X) = \{\}$ ✓

$f(X) = \{3, 7\}$ ✗

We need three nominal sets:

Things that can be transmitted in communications

T (Data) Terms ($M, N, \ldots$)

A Assertions ($\Psi, \Psi', \ldots$) To state properties of data

C Conditions ($\varphi, \varphi', \ldots$)

To formulate tests on data

These are not necessarily disjoint

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

$\overline{M}\, N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \,[\,]\, \cdots \,[\,]\, \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\vert \Psi \vert\!)$	Assertion

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

Channel

$\overline{M}\ N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
case $\varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\mid \Psi \mid\!)$	Assertion

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

Channel Object

$\overline{M}\,N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\mid \Psi \mid\!)$	Assertion

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

Channel

Object

Pattern

$\overline{M}\,N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\mid \Psi \mid\!)$	Assertion

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

Channel Object Pattern Test (aka guard)

$\overline{M}\,N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \,[]\, \cdots \,[]\, \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\vert \Psi \vert)$	Assertion

Agents ($P, Q, \ldots$)

T	(Data) Terms	M, N
A	Assertions	Ψ, Ψ'
C	Conditions	φ, φ'

Channel Object Pattern Test (aka guard)

$\overline{M}\, N.P$	Output
$\underline{M}(\lambda\widetilde{x})N.P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \,[\!]\, \cdots \,[\!]\, \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\vert \Psi \vert\!)$	Assertion

To mean tell Ψ

Assertion effect

An assertion that says x =3

$$(\!|x = 3|\!) \mid \textbf{case } x < 10 : \alpha.P_1 \, [] \, c = f(d) : P_2$$

Assertion effect

An assertion that says x =3

Condition made true

$$\{\!|x = 3|\!\} \mid \mathbf{case}\ x < 10 : \alpha.P_1 \ [] \ c = f(d) : P_2$$

Assertion effect

An assertion that says x =3

Condition made true

$$(\!| x = 3 |\!) \mid \mathbf{case}\ x < 10 : \alpha.P_1 \;[]\; c = f(d) : P_2$$

$$\xrightarrow{\alpha} (\!| x = 3 |\!) \mid P_1$$

Assertion effect

An assertion that says x =3

Condition made true

$$(\!|x = 3|\!) \mid \mathbf{case}\ x < 10 : \alpha.P_1 \ [\,]\ c = f(d) : P_2$$

$$\xrightarrow{\alpha} (\!|x = 3|\!) \mid P_1$$

$$(\!|a \dot{\leftrightarrow} f(b)|\!) \mid \underline{a}N.P \mid \overline{f(b)}N.Q$$

An assertion that says a and $f(b)$ denote the same communication channel

Assertion effect

An assertion that says x =3

Condition made true

$$(\!|x = 3|\!) \mid \textbf{case}\ x < 10 : \alpha.P_1 \,[]\, c = f(d) : P_2$$

$$\xrightarrow{\alpha} (\!|x = 3|\!) \mid P_1$$

Channels made equal

$$(\!|a \dot{\leftrightarrow} f(b)|\!) \mid \underline{a}N.P \mid \overline{f(b)}N.Q \rightarrow (\!|a \dot{\leftrightarrow} f(b)|\!) \mid P \mid Q$$

An assertion that says a and $f(b)$ denote the same communication channel

Assertions: information embedded in processes

Assertions: information embedded in processes

- Global facts about data structures

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

Assertions: information embedded in processes

- local knowledge

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

Assertions: information embedded in processes

- parametrised

$$a(x)\,.\,((\!| c = \text{encrypt}(M, x) |\!)\mid P)$$

Assertions: information embedded in processes

- communicated

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

Transitions

Actions α,...

For scope extrusions

Subject

$\overline{M}\,(\nu\tilde{a})N$	*Output*
$\underline{M}\;N$	*Input*
τ	*Silent*

Object

$$\boxed{\Psi \rhd P \xrightarrow{\alpha} P'}$$

In an environment that asserts Ψ the agent P can perform the action α and become P'

Rules

To define the rules we shall need a few other properties of our parameters.

Example: **case**

$$\Psi \;\rhd\; \mathbf{case}\,\widetilde{\varphi} : \widetilde{P} \;\xrightarrow{\alpha}\; P'$$

Rules

To define the rules we shall need a few other properties of our parameters.

Example: **case**

$$\Psi \;\rhd\; \mathbf{case}\,\widetilde{\varphi} : \widetilde{P} \;\xrightarrow{\alpha}\; P'$$

Should hold when one of the branches $\varphi_i : P_i$ is such that $P_i \xrightarrow{\alpha} P'$ and $\Psi \vdash \varphi_i$

Rules

To define the rules we shall need a few other properties of our parameters.

Example: **case**

$$\Psi \;\triangleright\; \mathbf{case}\,\widetilde{\varphi} : \widetilde{P} \;\xrightarrow{\alpha}\; P'$$

Should hold when one of the branches $\varphi_i : P_i$ is such that $P_i \xrightarrow{\alpha} P'$ and $\Psi \vdash \varphi_i$

Ergo, we need a notion of **entailment**

$$\boxed{\vdash \subseteq \mathbf{A} \times \mathbf{C}}$$

Rules

To define the rules we shall need a few other properties of our parameters.

Example: **case**

$$\Psi \ \rhd \ \mathbf{case}\,\widetilde{\varphi} : \widetilde{P} \ \xrightarrow{\alpha} \ P'$$

Should hold when one of the branches $\varphi_i : P_i$ is such that $P_i \xrightarrow{\alpha} P'$ and $\Psi \vdash \varphi_i$

Ergo, we need a notion of **entailment**

$$\boxed{\vdash \subseteq \mathbf{A} \times \mathbf{C}}$$

Entailment needs to be equivariant

$$\frac{\Psi \;\rhd\; P_i \;\xrightarrow{\alpha}\; P' \qquad \Psi \vdash \varphi_i}{\Psi \;\rhd\; \mathbf{case}\,\widetilde{\varphi} : \widetilde{P} \;\xrightarrow{\alpha}\; P'}$$

Consider **output** $\overline{M}\,N\,.\,P$

It should be able to send N along any channel which is represented by the data term M

Consider **output** $\overline{M}\, N\, .\, P$

It should be able to send N along any channel which is represented by the data term M

In general, *different* terms may represent the *same* channel (cf aliases)!

Consider **output** $\overline{M}\,N\,.\,P$

It should be able to send N along any channel which is represented by the data term M

In general, *different* terms may represent the *same* channel (cf aliases)!

$$M \dot{\leftrightarrow} M'$$

Means that M and M' represent the same channel

Dynamic: the environment may contain things like aliases

So, include channel equivalence among the conditions (which can be entailed by assertions)

$$\boxed{\dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \rightarrow \mathbf{C}}$$

Dynamic: the environment may contain things like aliases

So, include channel equivalence among the conditions (which can be entailed by assertions)

$$\boxed{\dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \to \mathbf{C}}$$

Needs to be equivariant and an equivalence relation on a subset of $\mathbf{T}$

$$\frac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \;\rhd\; \overline{M}\; N.P \;\xrightarrow{\overline{K}\; N}\; P}$$

Consider **input**

$$\underline{M}(\lambda\widetilde{y})N.P$$

Consider **input**

$$\underline{M}(\lambda \widetilde{y})N.P$$

It can receive anything that matches the pattern

Consider **input**

$$\underline{M}(\lambda\widetilde{y})N.P$$

It can receive anything that matches the pattern

Pattern matching is usually defined by substitution

Consider **input**

$$\underline{M}(\lambda\widetilde{y})N.P$$

It can receive anything that matches the pattern

Pattern matching is usually defined by substitution

So, we need a notion of substitution!

Substitution

$$\boxed{X[\widetilde{x} := \widetilde{T}]}$$

X where all *x*:es are replaced by *T*:s everywhere

Substitution

$\boxed{X[\widetilde{x} := \widetilde{T}]}$

X where all x:es are replaced by T:s everywhere

Defined how exactly?

Substitution

$\boxed{X[\widetilde{x} := \widetilde{T}]}$

X where all *x*:es are replaced by *T*:s everywhere

Defined how exactly?

We don't care!!

Substitution

$\boxed{X[\widetilde{x} := \widetilde{T}]}$ X where all x:es are replaced by T:s everywhere

Defined how exactly?

We don't care!!

Only needs to satisfy some criteria:

$$\begin{array}{ll}
\text{Equivariance:} & p \cdot (X[\tilde{x} := \tilde{T}]) = (p \cdot X)\big[(p \cdot \tilde{x}) := (p \cdot \tilde{T})\big] \\
\text{Freshness:} & \text{if } \ \tilde{x} \subseteq \mathrm{n}(X) \text{ and } a\#X[\tilde{x} := \tilde{T}] \text{ then } a\#\tilde{T} \\
\text{Alpha-equivalence:} & \text{if } p \subseteq \tilde{x} \times (p \cdot \tilde{x}) \text{ and } (p \cdot \tilde{x})\#X \text{ then} \\
 & \qquad X[\tilde{x} := \tilde{T}] = (p \cdot X)[(p \cdot \tilde{x}) := \tilde{T}]
\end{array}$$

$$\frac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}$$

$$\underline{M}\underbrace{(\lambda xy)f(x,g(y))}_{\text{A pattern}}.P \quad \mid \quad \overline{M}\underbrace{f(h(z),g(u))}_{\text{Data}}.Q$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \;\rhd\; \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\; N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$f(h(z), g(u))$$

$$\underline{M}\underbrace{(\lambda xy)f(x, g(y))}_{\text{A pattern}}.P \quad \mid \quad \overline{M}\underbrace{f(h(z), g(u))}_{\text{Data}}.Q$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \ \rhd \ \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$f(h(z), g(u))$$

$$\underline{M}\underbrace{(\lambda xy)f(x, g(y))}_{\text{A pattern}}.P \quad | \quad \overline{M}\underbrace{f(h(z), g(u))}_{\text{Data}}.Q$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \;\rhd\; \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\; N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\, P}}$$

$$f(\qquad , g(u))$$

$$\underline{M}\underbrace{(\lambda xy)f(x, g(y))}_{\text{A pattern}}.P \qquad | \qquad \overline{M}\underbrace{f(h(z), g(u))}_{\text{Data}}.Q$$

$$[x := h(z), \quad := \quad]$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \;\rhd\; \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\;N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$f(\qquad , g(\;))$$

$$\underline{M}\underbrace{(\lambda xy)f(x,g(y))}_{\text{A pattern}}.P \qquad | \qquad \overline{M}\,\underbrace{f(h(z),g(u))}_{\text{Data}}.Q$$

$$[x := h(z), y := u]$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \ \rhd \ \underline{M}(\lambda\widetilde{y})N.P \ \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} \ P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$\underline{M}\underbrace{(\lambda xy)f(x,g(y))}_{\text{A pattern}}.P \qquad | \qquad \overline{M}\underbrace{f(h(z),g(u))}_{\text{Data}}.Q$$

$$\rightarrow \qquad P \ \ [x := h(z), y := u] \ \ | \, Q$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \ \rhd \ \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$\underline{M}\underbrace{(\lambda xy)f(x,g(y))}_{\text{A pattern}}.P \qquad \mid \qquad \overline{M}\,\underbrace{f(h(z),g(u))}_{\text{Data}}.Q$$

$$\rightarrow \qquad P\ \ [x := h(z), y := u]\ \ \mid Q$$

$$\boxed{\dfrac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \;\rhd\; \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\ N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]\,P}}$$

$$\underline{M}\underbrace{(\lambda xy)f(x,g(y))}_{\text{A pattern}}.P \qquad | \qquad \overline{M}\underbrace{f(h(z),g(u))}_{\text{Data}}.Q$$

$$\rightarrow \qquad P\;\;[x := h(z), y := u] \;\;| \, Q$$

”Ordinary” input $a(x).P$ is then $\underline{a}(\lambda x)x.P$

Restriction: just as in the pi-calculus, where an action can extrude any number of names

$$\frac{\Psi \;\rhd\; P \;\xrightarrow{\alpha}\; P'}{\Psi \;\rhd\; (\nu b)P \;\xrightarrow{\alpha}\; (\nu b)P'}\; b\#\alpha, \Psi$$

$$\frac{\Psi \;\rhd\; P \;\xrightarrow{\overline{M}\;(\nu\widetilde{a})N}\; P'}{\Psi \;\rhd\; (\nu b)P \;\xrightarrow{\overline{M}\;(\nu\widetilde{a}\cup\{b\})N}\; P'}\; \begin{array}{c} b\#\widetilde{a}, \Psi, M \\ b \in \mathrm{n}(N) \end{array}$$

Replication: just as in pi

$$\frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'}$$

Parallel composition: an assertion in P may affect a transition from Q in $P|Q$

$$(\!|x = 3|\!) \mid \mathbf{if}\ x < 5 \cdots$$

Parallel composition: an assertion in P may affect a transition from Q in $P|Q$

$$(\!|x=3|\!) \mid \mathbf{if}\ x<5 \cdots$$

But beware that restrictions may localise assertions

$$((\nu x)(\!|x=3|\!)) \mid \mathbf{if}\ x<5 \cdots$$

Several parallel assertions may have a combined effect:

$$(\!| x = 1 |\!) \mid (\!| y = 7 |\!) \mid \mathbf{if}\ x + y = 8 \cdots$$

Several parallel assertions
may have a combined effect:

$$(\!|x = 1|\!) \mid (\!|y = 7|\!) \mid \textbf{if } x + y = 8 \cdots$$

Need an operator to compose assertions

$$\boxed{\otimes \;: \mathbf{A} \times \mathbf{A} \to \mathbf{A}}$$

$$\boxed{\mathbf{1} \;: \mathbf{A}}$$

Intuition: $(\!|\Psi|\!) \mid (\!|\Psi'|\!)$ has the effect of $\Psi \otimes \Psi'$

$\mathbf{1}$ is the empty (default) assertion

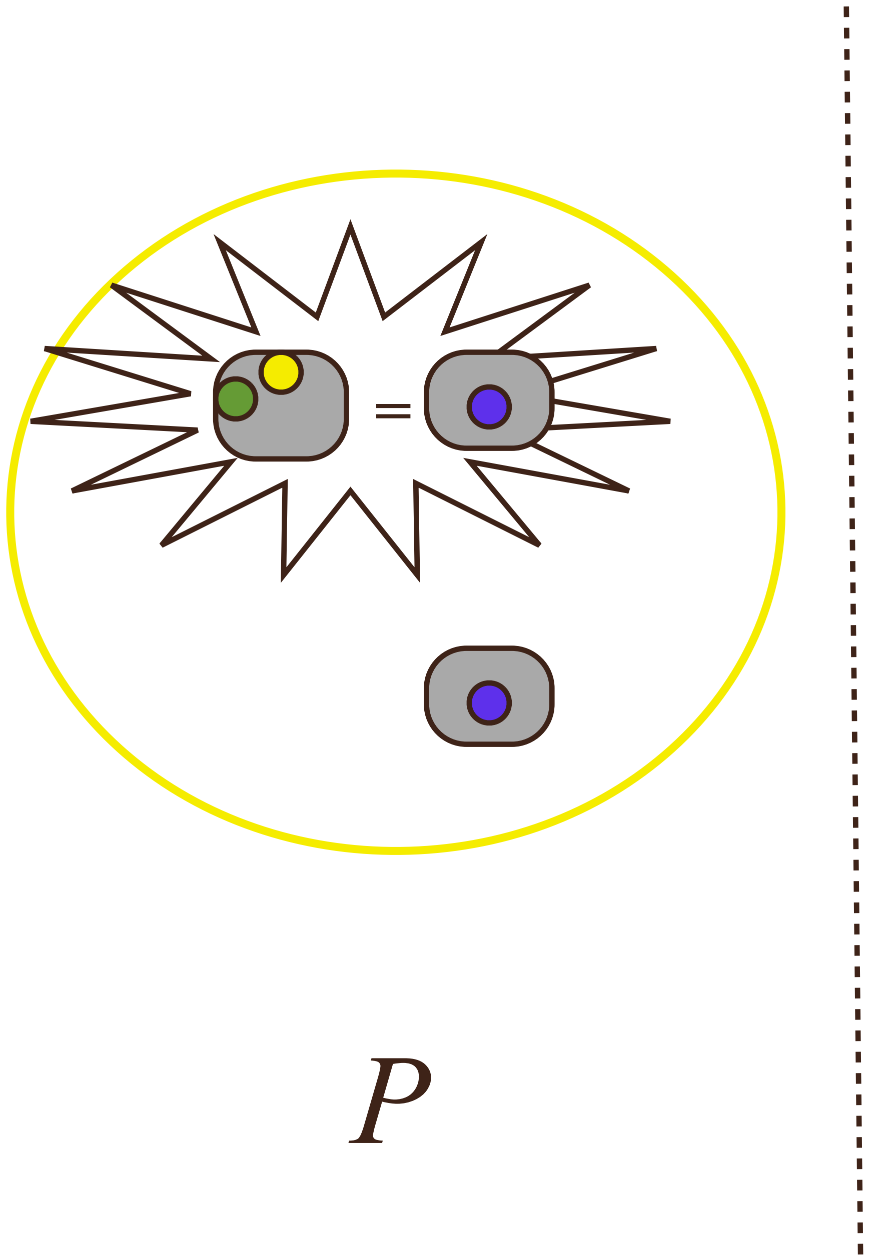
$\Psi \simeq \Psi'$ means $\forall \varphi.\ \Psi \vdash \varphi \Leftrightarrow \Psi' \vdash \varphi$

Required properties:

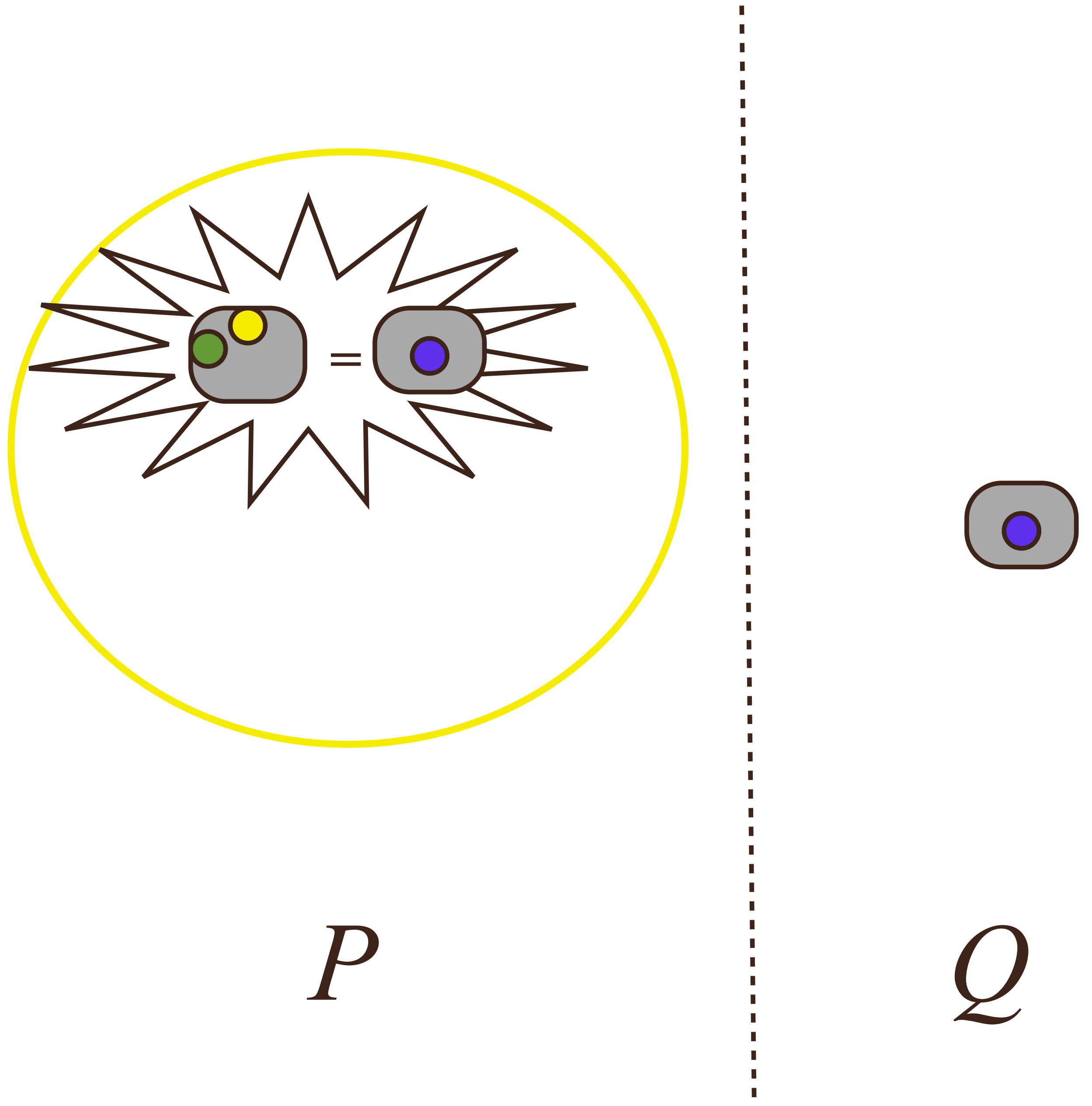
Compositionality:	$\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$
Identity:	$\Psi \otimes \mathbf{1} \simeq \Psi$
Associativity:	$(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$
Commutativity:	$\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Normally it is some kind of conjunction

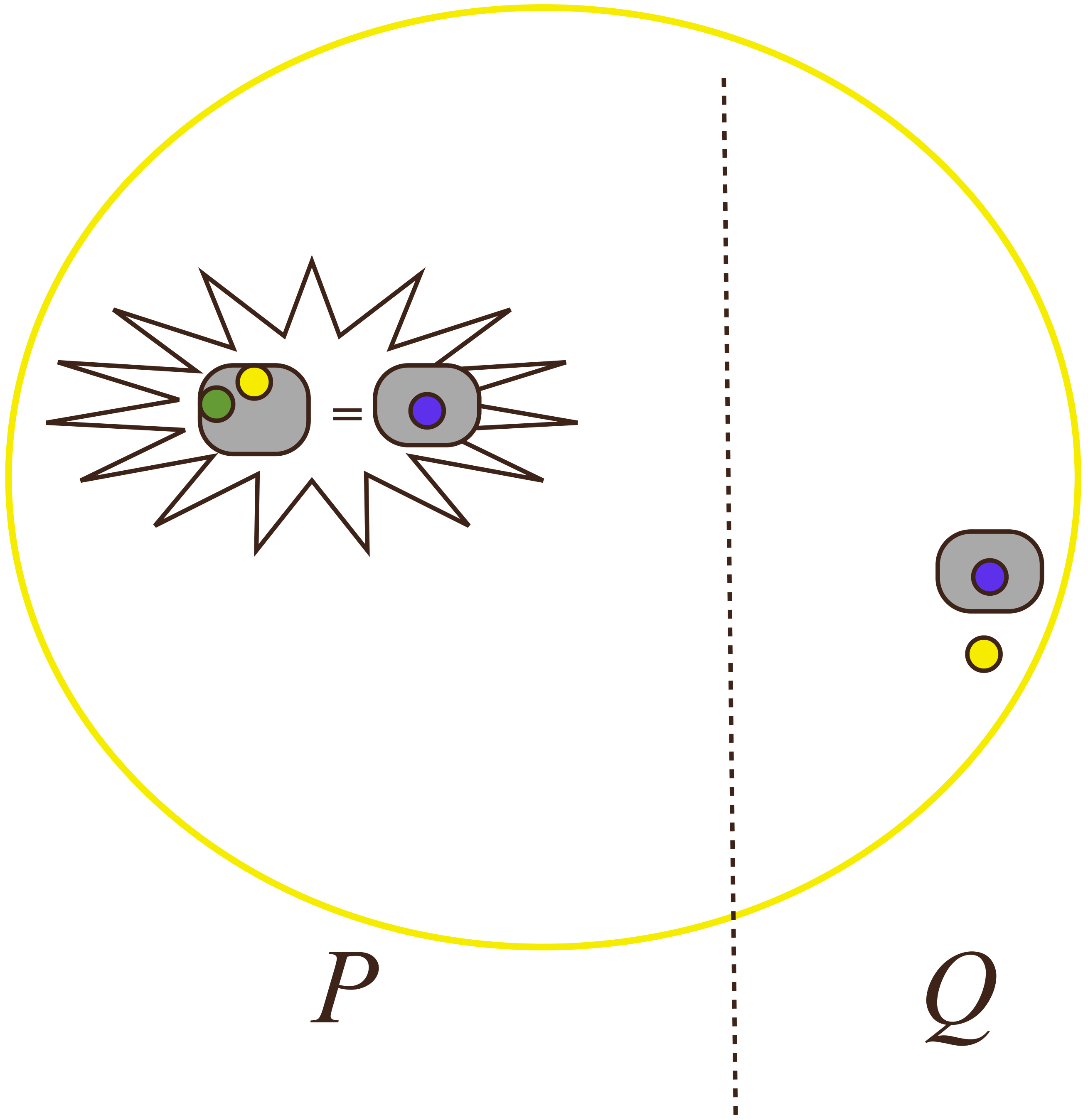
Recalling the ciphertext example



Recalling the ciphertext example

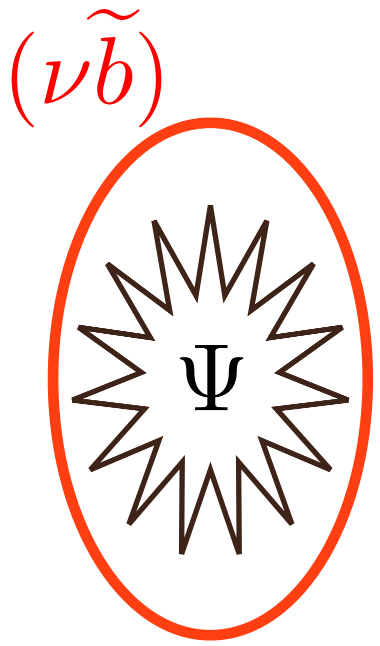


Recalling the ciphertext example



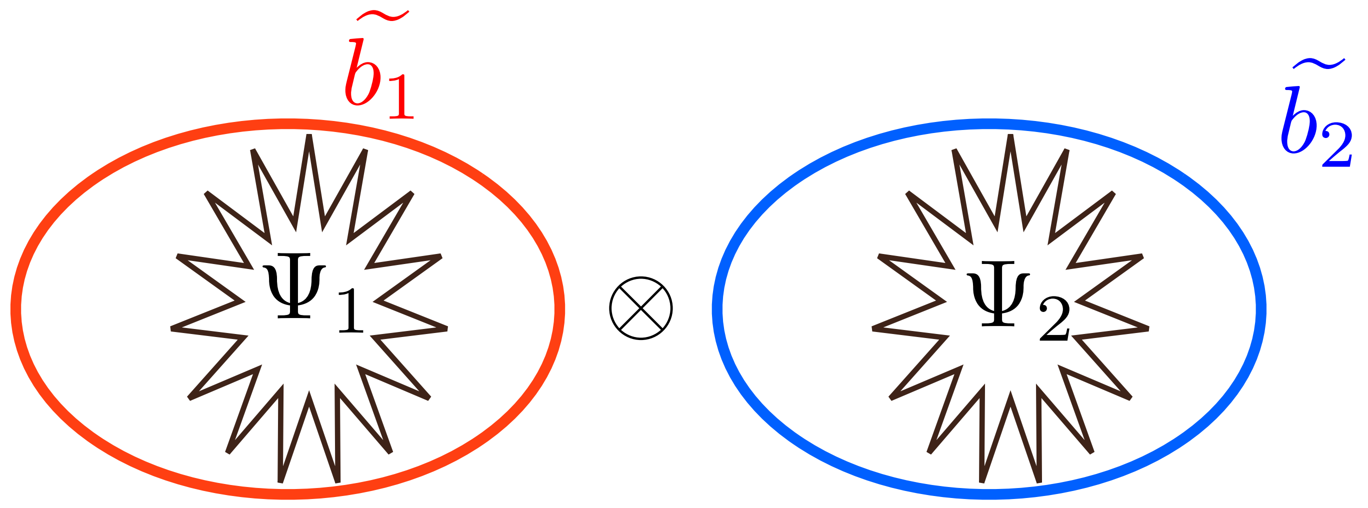
Frame

A **frame** F is of kind $(\nu\widetilde{b})\Psi$



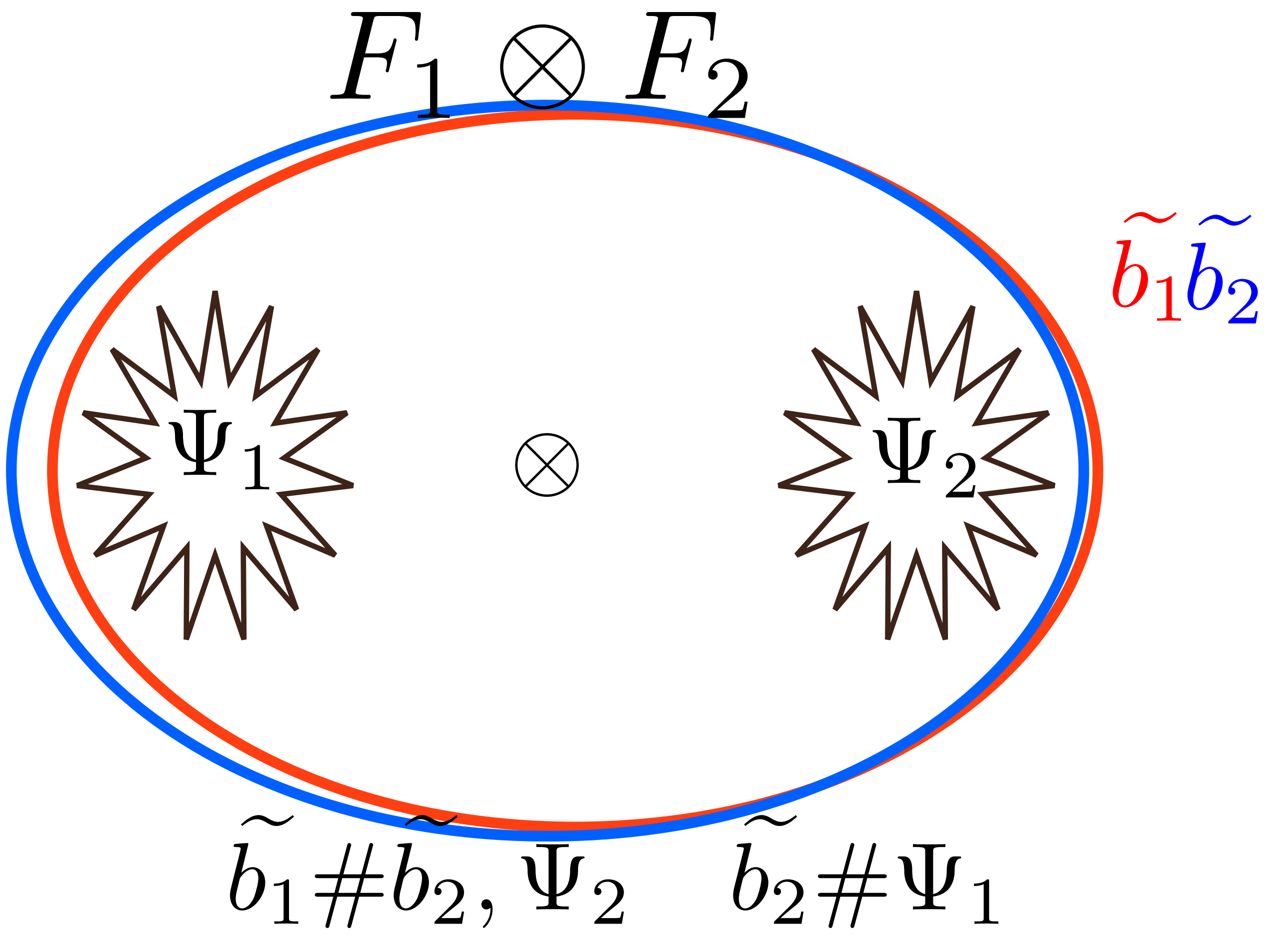
Frame composition

$$F_1 \otimes F_2$$



$$\widetilde{b_1} \# \widetilde{b_2}, \Psi_2 \quad \widetilde{b_2} \# \Psi_1$$

Frame composition


$F_1 \otimes F_2$
$\widetilde{b_1}\widetilde{b_2}$
Ψ_1
$\otimes$
Ψ_2
$\widetilde{b_1}\#\widetilde{b_2}, \Psi_2$
$\widetilde{b_2}\#\Psi_1$

Frame of an agent

The frame of an agent is the composition of its top level assertions, pulling binders outmost.

Intuition: this is what the agent contributes to its environment

$$\mathcal{F}(\mathbf{0}) = \mathcal{F}(\underline{M}(\lambda\widetilde{x})N.P) = \mathcal{F}(\overline{M}\ N.P) =$$
$$\mathcal{F}(\mathbf{case}\,\widetilde{\varphi} : \widetilde{P}) = \mathcal{F}(!P) = \mathbf{1}$$
$$\mathcal{F}((\!|\Psi|\!)) = \Psi$$
$$\mathcal{F}(P \mid Q) = \mathcal{F}(P) \otimes \mathcal{F}(Q)$$
$$\mathcal{F}((\nu b)P) = (\nu b)\mathcal{F}(P)$$

Example

$$P = (\nu a)((\!|\Psi_1|\!) \mid (\nu b)(\!|\Psi_2|\!) \mid \overline{M}N.(\!|\Psi_3|\!))$$

Example

$$P = (\nu a)((\!|\Psi_1|\!) \mid (\nu b)(\!|\Psi_2|\!) \mid \overline{M}N.(\!|\Psi_3|\!))$$

$$\mathcal{F}(P) = (\nu a\, b)(\Psi_1 \otimes \Psi_2 \otimes \mathbf{1}) \simeq (\nu ab)\; (\Psi_1 \otimes \Psi_2)$$

Example

$$P = (\nu a)((\!|\Psi_1|\!) \mid (\nu b)(\!|\Psi_2|\!) \mid \overline{M}N.(\!|\Psi_3|\!))$$

$$\mathcal{F}(P) = (\nu a\, b)(\Psi_1 \otimes \Psi_2 \otimes \mathbf{1}) \simeq (\nu ab)\ (\Psi_1 \otimes \Psi_2)$$

Assuming $b\#\Psi_1$

Parallel

Here Q is the environment of P and therefore Q contributes its frame for P's action

$$\boxed{\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \ \rhd\ P \ \xrightarrow{\alpha}\ P'}{\Psi \ \rhd\ P|Q \ \xrightarrow{\alpha}\ P'|Q}\ \mathrm{bn}(\alpha)\#Q}$$

$$\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q \text{ where } \widetilde{b_Q}\#\Psi, P, \alpha$$

Example

Objects are omitted.

$\Psi \vdash a \dot{\leftrightarrow} b$

Example

$$\Psi \vdash a \mathrel{\dot{\leftrightarrow}} b$$

$$\Psi \rhd \overline{a}.P \xrightarrow{\overline{b}} P$$

Objects are omitted.

$$\textsc{Out}\ \frac{\Psi \vdash M \mathrel{\dot{\leftrightarrow}} K}{\Psi \rhd \overline{M}\, N.P \xrightarrow{\overline{K}\, N} P}$$

Example

Objects are omitted.

$$\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \ \rhd\ P \ \xrightarrow{\alpha}\ P'}{\Psi \ \rhd\ P|Q \ \xrightarrow{\alpha}\ P'|Q}\ \mathrm{bn}(\alpha)\#Q$$

$$\Psi \vdash a \ \dot{\leftrightarrow}\ b.$$

$$\Psi \rhd \overline{a}.P \xrightarrow{\overline{b}} P$$

$$\mathbf{1} \rhd\ \overline{a}.P \mid (\!|\Psi|\!) \xrightarrow{\overline{b}} P \mid (\!|\Psi|\!)$$

Example

Objects are omitted.

$$\textsc{Scope}\ \frac{\Psi \ \rhd\ P \ \xrightarrow{\alpha}\ P'}{\Psi \ \rhd\ (\nu b)P \ \xrightarrow{\alpha}\ (\nu b)P'}\ b\#\alpha, \Psi$$

$$\Psi \vdash a \dot{\leftrightarrow} b.$$

$$\Psi \rhd \overline{a}.P \xrightarrow{\overline{b}} P$$

$$\mathbf{1} \rhd \ \overline{a}.P \mid (\!|\Psi|\!) \xrightarrow{\overline{b}} P \mid (\!|\Psi|\!)$$

$$\mathbf{1} \rhd \ (\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \xrightarrow{\overline{b}} (\nu a)(P \mid (\!|\Psi|\!))$$

Example

Objects are omitted.

$$\Psi \vdash a \dot{\leftrightarrow} b.$$

$$\Psi \rhd \overline{a}.P \xrightarrow{\overline{b}} P$$

$$\mathbf{1} \rhd \overline{a}.P \mid (\!|\Psi|\!) \xrightarrow{\overline{b}} P \mid (\!|\Psi|\!)$$

$$\mathbf{1} \rhd (\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \xrightarrow{\overline{b}} (\nu a)(P \mid (\!|\Psi|\!))$$

Which one has an internal action?

$$(\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \mid a.Q \qquad (\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \mid b.Q$$

Example

Objects are omitted.

$$\Psi \vdash a \dot{\leftrightarrow} b.$$

$$\Psi \rhd \overline{a}.P \xrightarrow{\overline{b}} P$$

$$\mathbf{1} \rhd \ \overline{a}.P \mid (\!|\Psi|\!) \xrightarrow{\overline{b}} P \mid (\!|\Psi|\!)$$

$$\mathbf{1} \rhd \ (\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \xrightarrow{\overline{b}} (\nu a)(P \mid (\!|\Psi|\!))$$

Which one has an internal action?

$(\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \mid a.Q$ $\quad$ $(\nu a)(\overline{a}.P \mid (\!|\Psi|\!)) \mid b.Q$

Interaction

Here P and Q are in the environment of each other

$$\textsc{Com}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\,N} Q' \qquad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \stackrel{\cdot}{\leftrightarrow} K}{\Psi \rhd P \mid Q \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q')}\ \widetilde{a} \# Q$$

$\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q$ and $\mathcal{F}(P) = (\nu\widetilde{b_P})\Psi_P$
where $\widetilde{b_P} \# \widetilde{b_Q}, \Psi, P, Q, M$
and correspondingly for $\widetilde{b_Q}$

Interaction Example

$$a \# \Psi_b, \quad b \# \Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$$

Eg, Ψ_x says "x is unit for a given group"

Interaction Example

$$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$$

Eg, Ψ_x says "x is unit for a given group"

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

Interaction Example

$$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$$

Eg, Ψ_x says "x is unit for a given group"

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0} \mid (\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

Just reorder the terms

Interaction Example

$$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \leftrightarrow b$$

Eg, Ψ_x says "x is unit for a given group"

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0} \mid (\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

Just reorder the terms

Interaction Example

$$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$$

Eg, Ψ_x says "x is unit for a given group"

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a} \,.\, \mathbf{0} \mid b \,.\, \mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a} \,.\, \mathbf{0} \mid (\!|\Psi_b|\!) \mid b \,.\, \mathbf{0}) \xrightarrow{\tau}$$

Just reorder the terms

Interaction Example

$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$

Eg, Ψ_x says "x is unit for a given group"

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

Just reorder the terms

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0} \mid (\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0}) \mid (\nu b)((\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

This should hold by scope extension!

Interaction Example

$a \# \Psi_b, \quad b \# \Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$

Eg, Ψ_x says "x is unit for a given group"

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a}\,.\,\mathbf{0} \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

Just reorder the terms

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0} \mid (\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a)((\!|\Psi_a|\!) \mid \overline{a}\,.\,\mathbf{0}) \mid (\nu b)((\!|\Psi_b|\!) \mid b\,.\,\mathbf{0}) \xrightarrow{\tau}$$

This should hold by scope extension!

How infer this?

Interaction Example

$a\#\Psi_b, \quad b\#\Psi_a, \quad \Psi_a \otimes \Psi_b \vdash a \dot{\leftrightarrow} b$

Eg, Ψ_x says "x is unit for a given group"

By the Com rule: the subjects of an output and input are made channel equivalent by the environment

$$(\nu a, b)((\!|\Psi_a|\!) \mid (\!|\Psi_b|\!) \mid \overline{a} \,.\, \mathbf{0} \mid b \,.\, \mathbf{0}) \xrightarrow{\tau}$$

$$(\nu a, b)((\!|\Psi_a|\!) \mid \overline{a} \,.\, \mathbf{0} \mid (\!|\Psi_b|\!) \mid b \,.\, \mathbf{0}) \xrightarrow{\tau}$$

Just reorder the terms

$$(\nu a)((\!|\Psi_a|\!) \mid \overline{a} \,.\, \mathbf{0}) \mid (\nu b)((\!|\Psi_b|\!) \mid b \,.\, \mathbf{0}) \xrightarrow{\tau}$$

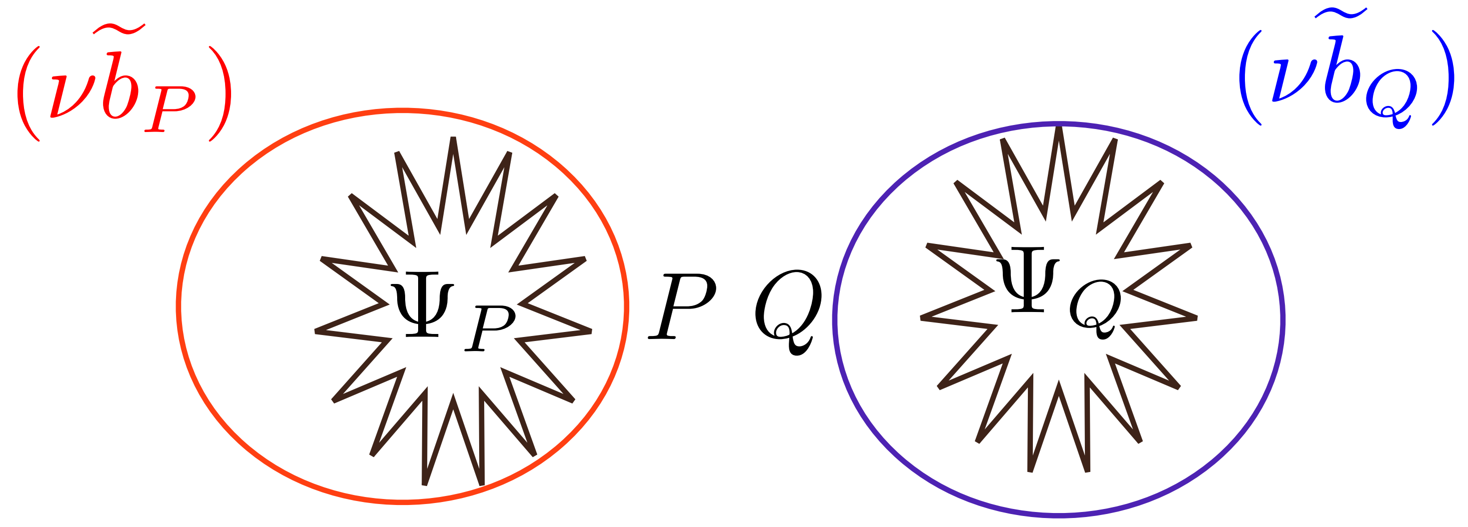
This should hold by scope extension!

$$\Psi_b \rhd (\nu a)((\!|\Psi_a|\!) \mid \overline{a} \,.\, \mathbf{0}) \xrightarrow{\overline{b}}$$

$$\Psi_a \rhd (\nu b)((\!|\Psi_b|\!) \mid b \,.\, \mathbf{0}) \xrightarrow{a}$$

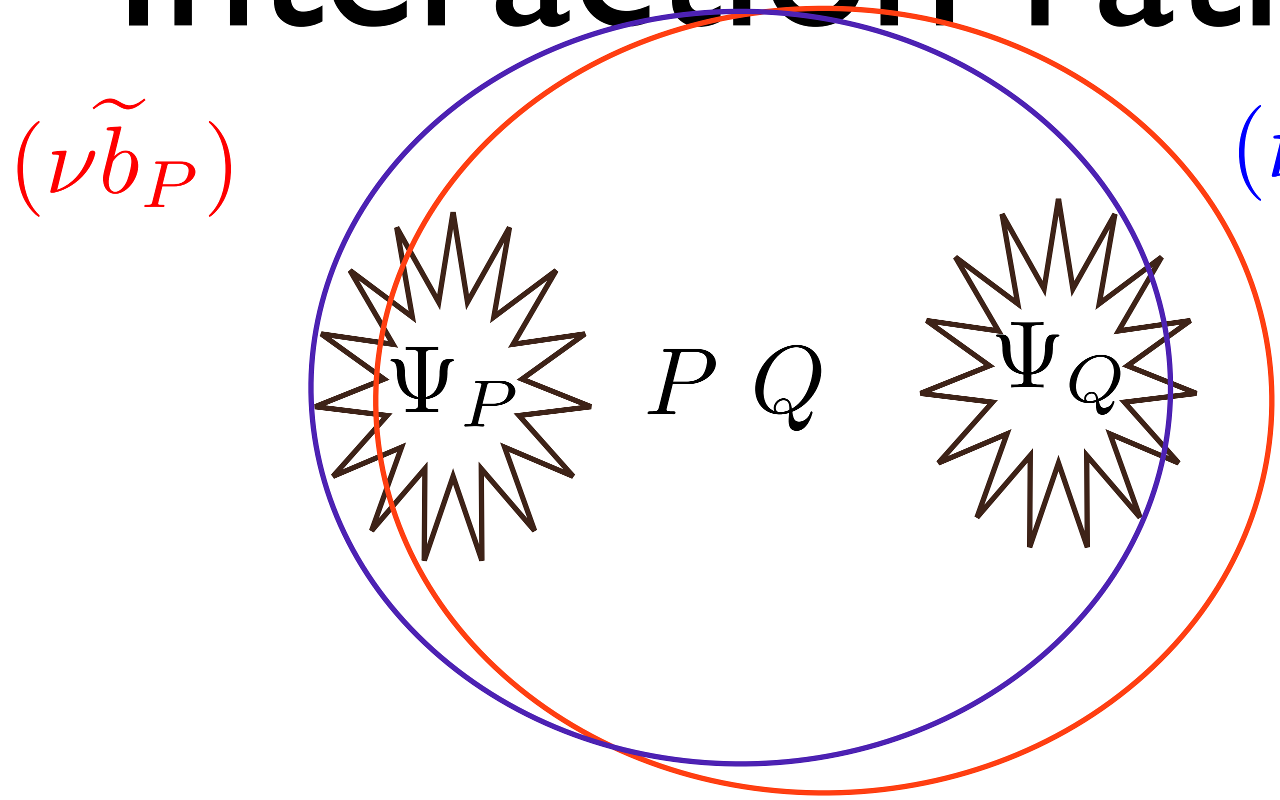
$$\textsc{Com}\ \frac{\Psi_Q \otimes \Psi \ \rhd\ P \xrightarrow{\overline{M}\,(\nu\tilde{a})N} P' \qquad \Psi_P \otimes \Psi \ \rhd\ Q \xrightarrow{\underline{K}\,N} Q' \qquad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \dot{\leftrightarrow} K}{\Psi \ \rhd\ P \mid Q \xrightarrow{\tau} (\nu\tilde{a})(P' \mid Q')}\ \tilde{a}\#Q$$

Interaction rationale



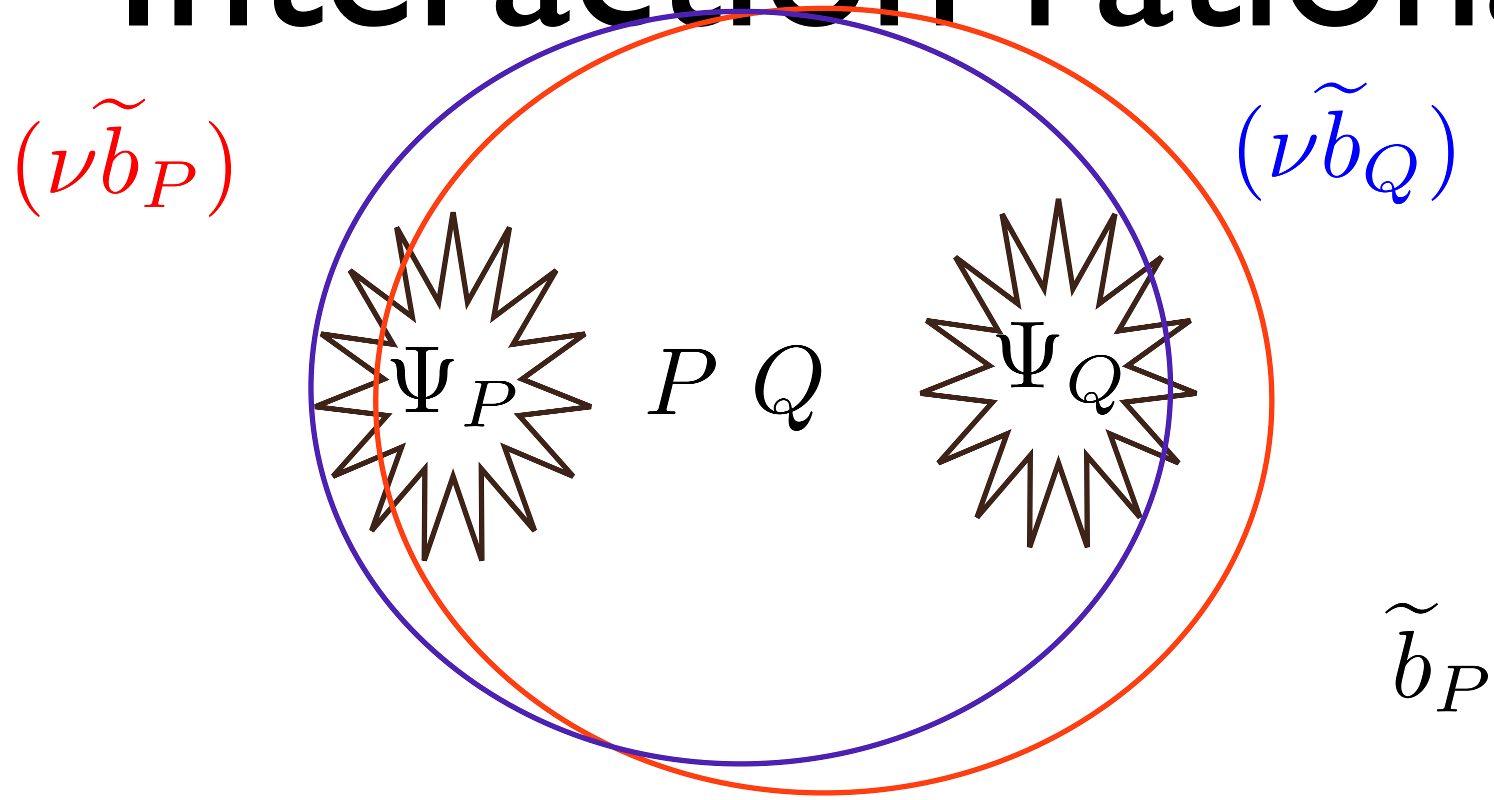
$\widetilde{b}_P \# \widetilde{b}_Q, \ldots$

Interaction rationale



$\widetilde{b}_P \# \widetilde{b}_Q, \ldots$

Interaction rationale



$\widetilde{b}_P\#\widetilde{b}_Q,\ldots$

$P \xrightarrow{\overline{M}N} P' \qquad Q \xrightarrow{\underline{K}N} Q' \qquad M \dot{\leftrightarrow} K$

$P|Q \xrightarrow{\tau} P'|Q'$

All the rules

$$\textsc{In}\ \frac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N.P \xrightarrow{\underline{K}\, N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y} := \widetilde{L}]} \qquad \textsc{Out}\ \frac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \rhd \overline{M}\, N.P \xrightarrow{\overline{K}\, N} P} \qquad \textsc{Case}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\alpha} P'}$$

$$\textsc{Com}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\, N} Q' \qquad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \dot{\leftrightarrow} K}{\Psi \rhd P \mid Q \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q')}\ \widetilde{a} \# Q$$

$$\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd P|Q \xrightarrow{\alpha} P'|Q}\ \mathrm{bn}(\alpha)\#Q \qquad \textsc{Scope}\ \frac{\Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd (\nu b)P \xrightarrow{\alpha} (\nu b)P'}\ b\#\alpha, \Psi$$

$$\textsc{Open}\ \frac{\Psi \rhd P \xrightarrow{\overline{M}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\overline{M}\,(\nu\widetilde{a}\cup\{b\})N} P'}\ \begin{array}{l} b\#\widetilde{a}, \Psi, M \\ b \in \mathrm{n}(N) \end{array} \qquad \textsc{Rep}\ \frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'}$$

+ freshness conditions in Par and Com

+ symmetric variants

Summary: For a psi-calculus we need:

Three nominal sets $\mathbf{T}, \mathbf{A}, \mathbf{C}$

Four morphisms:

$\dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \rightarrow \mathbf{C}$

$\otimes : \mathbf{A} \times \mathbf{A} \rightarrow \mathbf{A}$

$\mathbf{1} : \mathbf{A}$

$\vdash \subseteq \mathbf{A} \times \mathbf{C}$

All equivariant

Requisites

Channel Symmetry: $\Psi \vdash M \dot{\leftrightarrow} N \implies \Psi \vdash N \dot{\leftrightarrow} M$

Channel Transitivity: $\Psi \vdash M \dot{\leftrightarrow} N \;\wedge\; \Psi \vdash N \dot{\leftrightarrow} L \implies \Psi \vdash M \dot{\leftrightarrow} L$

Composition: $\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$

Identity: $\Psi \otimes \mathbf{1} \simeq \Psi$

Associativity: $(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$

Commutativity: $\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Requisites

Partial equivalence relation

Channel Symmetry:	$\Psi \vdash M \dot{\leftrightarrow} N \implies \Psi \vdash N \dot{\leftrightarrow} M$
Channel Transitivity:	$\Psi \vdash M \dot{\leftrightarrow} N \ \wedge\ \Psi \vdash N \dot{\leftrightarrow} L \implies \Psi \vdash M \dot{\leftrightarrow} L$
Composition:	$\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$
Identity:	$\Psi \otimes \mathbf{1} \simeq \Psi$
Associativity:	$(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$
Commutativity:	$\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Requisites

Composition preserves equivalence

Channel Symmetry:	$\Psi \vdash M \dot{\leftrightarrow} N \implies \Psi \vdash N \dot{\leftrightarrow} M$
Channel Transitivity:	$\Psi \vdash M \dot{\leftrightarrow} N \wedge \Psi \vdash N \dot{\leftrightarrow} L \implies \Psi \vdash M \dot{\leftrightarrow} L$
Composition:	$\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$
Identity:	$\Psi \otimes \mathbf{1} \simeq \Psi$
Associativity:	$(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$
Commutativity:	$\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Requisites

Channel Symmetry: $\Psi \vdash M \stackrel{\cdot}{\leftrightarrow} N \implies \Psi \vdash N \stackrel{\cdot}{\leftrightarrow} M$

Channel Transitivity: $\Psi \vdash M \stackrel{\cdot}{\leftrightarrow} N \ \wedge \ \Psi \vdash N \stackrel{\cdot}{\leftrightarrow} L \implies \Psi \vdash M \stackrel{\cdot}{\leftrightarrow} L$

Composition: $\Psi \simeq \Psi' \implies \Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$

Identity: $\Psi \otimes \mathbf{1} \simeq \Psi$

Associativity: $(\Psi \otimes \Psi') \otimes \Psi'' \simeq \Psi \otimes (\Psi' \otimes \Psi'')$

Commutativity: $\Psi \otimes \Psi' \simeq \Psi' \otimes \Psi$

Abelian monoid

Note that we do **not** require eg

monotonicity $\Psi \vdash \varphi \;\Rightarrow \Psi \otimes \Psi' \vdash \varphi$

idempotence $\Psi \otimes \Psi \simeq \Psi$

This means we can accomodate nonmonotonic logics such as constraints with retracts!

Substitution

$\mathbf{X} \times \text{name List} \times \mathbf{T}\ \text{List} \to \mathbf{X}$ For $\mathbf{X}$ each of $\mathbf{A}$,$\mathbf{C}$,$\mathbf{T}$

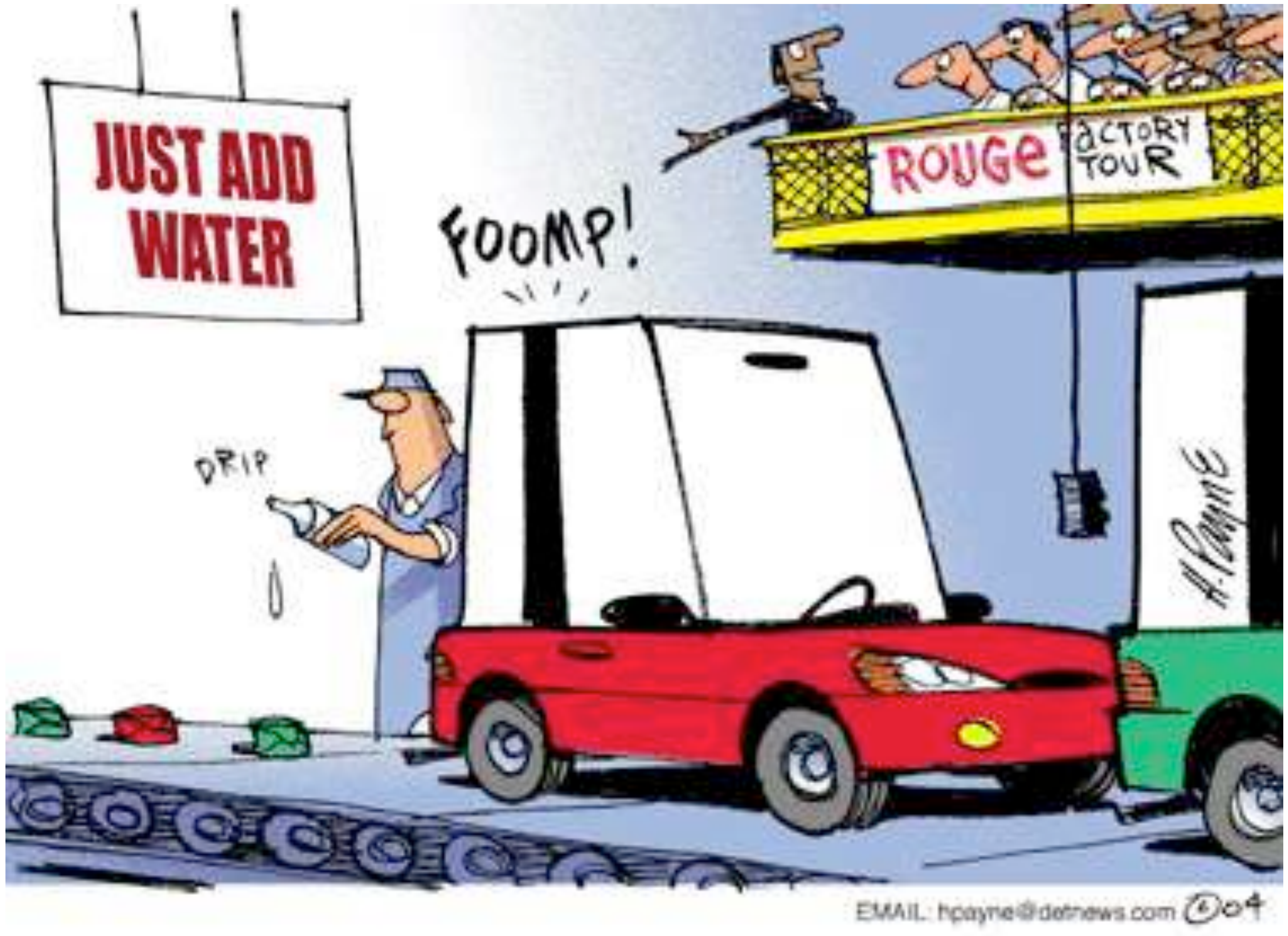
Equivariance: $p \cdot (X[\tilde{x} := \tilde{T}]) = (p \cdot X)\big[(p \cdot \tilde{x}) := (p \cdot \tilde{T})\big]$

Freshness: if $\tilde{x} \subseteq \mathrm{n}(X)$ and $a \# X[\tilde{x} := \tilde{T}]$ then $a \# \tilde{T}$

Alpha-equivalence: if $p \subseteq \tilde{x} \times (p \cdot \tilde{x})$ and $(p \cdot \tilde{x}) \# X$ then

$$X[\tilde{x} := \tilde{T}] = (p \cdot X)[(p \cdot \tilde{x}) := \tilde{T}]$$

Just add data and logic



"Since the last Ford factory tours 30 years ago, we've made remarkable advances in vehicle assembly."