Explicit Fusions

Lucian Wischik (Cambridge)

Bologna, October 2000

joint work with Philippa Gardner

paper in proceedings of MFCS2000

what is an “explicit fusion”?

```
void P (int &x, int &y)
{ x = x+y;
  y = x-y;
  x = x-y;
}
```

```
fun P x y =
( x := !x + !y;
  y := !x - !y;
  x := !x - !y
);
```

How does P behave?

How does P behave in a context where x=y ?

We write:

$$<x{=}y> \;\mid\; P$$

$$\equiv \quad <x{=}y> \;\mid\; P\{{}^{y}\!/\!_{x}\}$$

$$\equiv \quad <x{=}y> \;\mid\; P\{{}^{x}\!/\!_{y}\}$$

why explicit fusions?

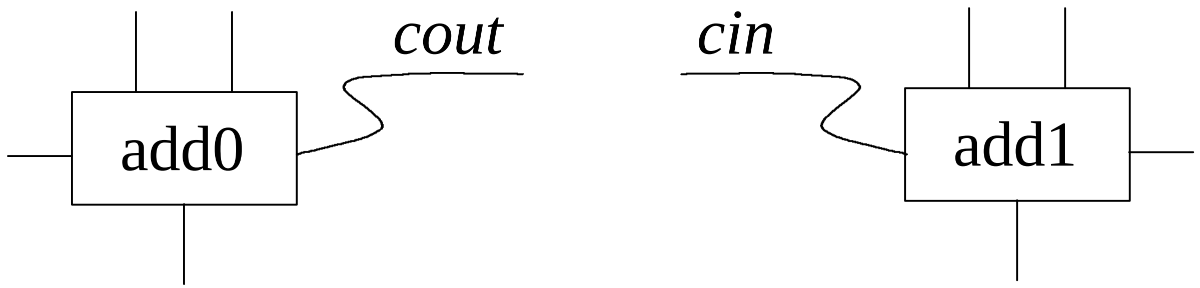
- they occur in real world
- help implementation
- momentum of related work
- simplify syntax

- can add explicit fusions to different calculi. For example:

pi-F calculus

- like pi but with explicit fusions
- LTS, bisimulation
- embeds pi, fusion calculi
- full abstraction for fusion calc.

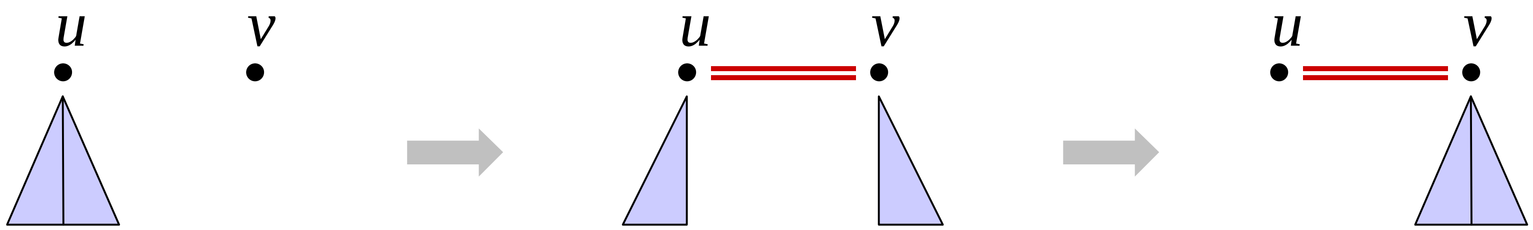
fusions in hardware



$$(\text{<}cout\text{>} \mid A_0) \mathbin{@} (\text{<}cin\text{>} \mid A_1) \quad \equiv \quad \text{<}cout{=}cin\text{>} \mid A_0 \mid A_1$$

e.g. SHRM [G00]

fusions in internet



Fusion allows for *asynchronous* migration of code:
migration can happen gradually, and is non-blocking.

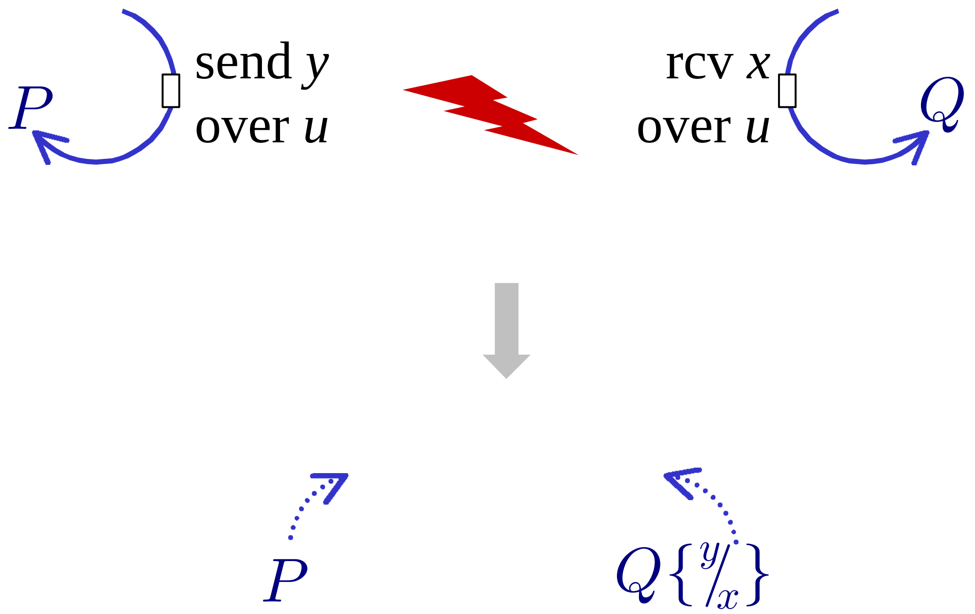
eg. ambients [FLS00], pi-calculus machine [in progress]

"Explicitly record and delay the effect of the fusion"
cf. *Explicit Substitutions* [ACCL 91]

pi-calculus

- communication between output & input

$\overline{u}.\langle y\rangle P \,|\, u.(x)Q \;\searrow\; P \,|\, Q\{^y/_x\}$



- dynamic creation of names/channels
- mobile scope of names/channels $(\nu y)P \mid Q \;\equiv\; (\nu y)(P \,|\, Q)$

related work 1

pi. [MPW89]

$\overline{u}.<y>P$ output

$u.(x)Q$ input

$(\nu z)R$ name-hiding

$$\overline{u}.<y>P \mid u.(x)Q \searrow P \mid Q\{{}^{y}\!/\!_{x}\}$$

explicit fusions. [GW00]
fusion calculus. [VP98]
chi calculus. [FU97]

$\overline{u}.<y>P, \quad u.<x>Q, \quad (\nu z)R$

$$\overline{u}.<y>P \mid u.<x>Q \searrow ???$$

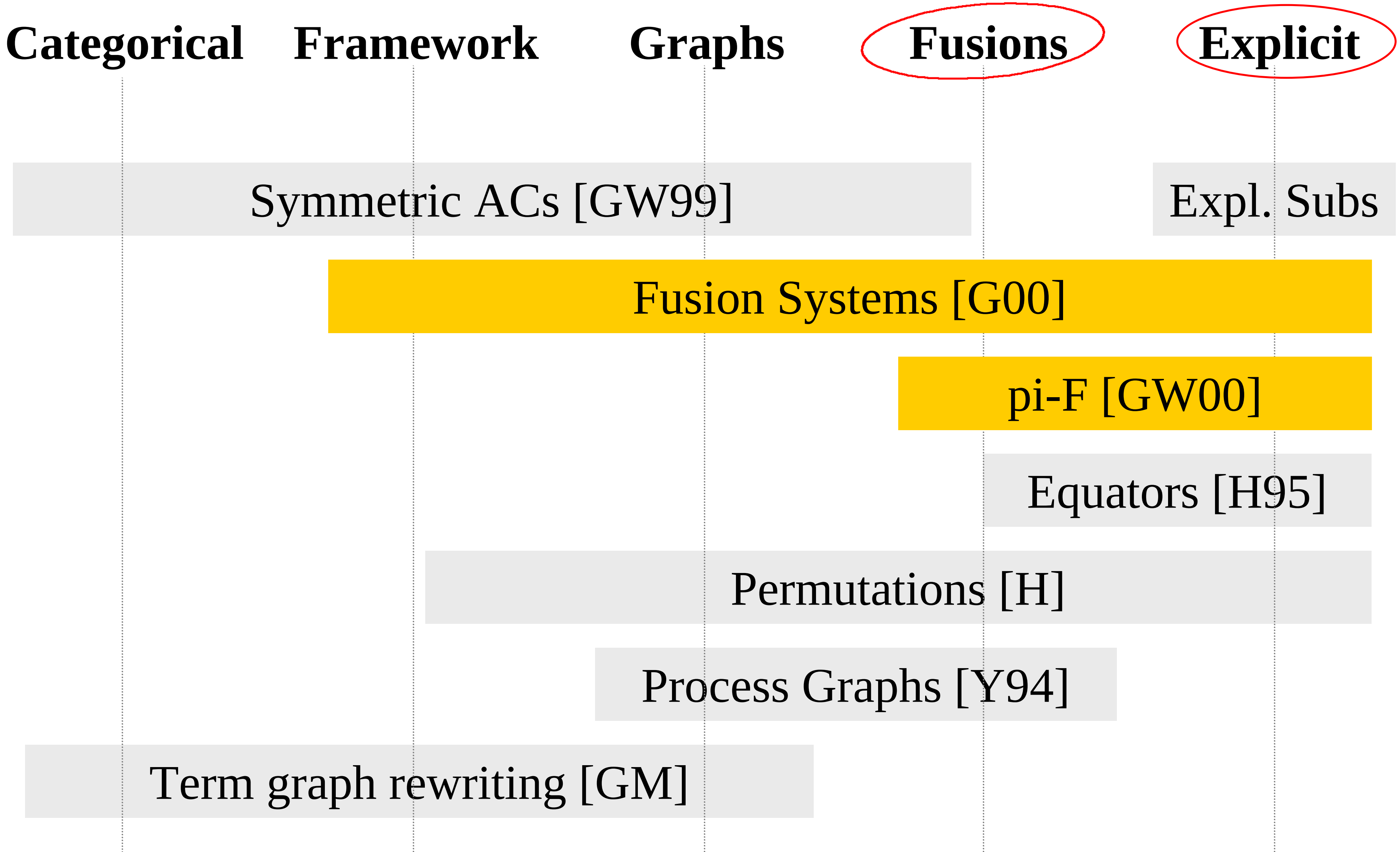
pi-I. [Sang96]

$\overline{u}.(y)P, \quad u.(x)Q, \quad (\nu z)R$

$$\overline{u}.(y)P \mid u.(x)Q \searrow (\nu z)(P\{{}^{z}\!/\!_{y}\} \mid Q\{{}^{z}\!/\!_{x}\})$$

z fresh

related work 2



pi-F calculus

- grammar
- reaction relation
- structural congruence (e.g. alpha conversion)

labels, bisimulation

- use CCS labels $\xrightarrow{\overline{u}}$ $\xrightarrow{u}$ $\xrightarrow{\tau}$
- “open” bisimulation
- result: is a congruence

embedding pi, fusion

- how to encode bound inp.
- result: reaction preserved. Full abstraction for fusion but not pi

pi-F calculus

$P \;::=$	$\mathbf{nil}$	empty process
	$x.P$	input
	$\overline{x}.P$	output
	$P \mid P$	parallel composition
	$(\nu x)P$	name-hiding
	$<\!x\!>$	datum
	$<\!x\!=\!y\!>$	explicit fusion

datums

- are like concretions
- not commutative
- polyadic, through parallel composition

$$<\!x\!> \mid <\!y\!> \mid P \;\cdots\; <\!xy\!>P$$

explicit fusion

- denotes an equivalence relation
- finite basis
- has substitutive effect

pi-F reaction

$$\overline{u}.P \,|\, u.Q \;\searrow\; P \,@\, Q$$

effect of @ is to fuse datums

interface

$$\overline{u}.(<y>|P) \;|\; u.(\nu x)(<x>|Q) \;\searrow\; (<y>|P) \;@\; (\nu x)(<x>|Q)$$

$$\equiv\; (\nu x)\big(<x{=}y>|P\,|\,Q\big)$$

$$\equiv\; P\{^{y}\!/_{x}\} \,|\, Q\{^{y}\!/_{x}\}$$

encode bound input $u.(x)Q$

encode lazy input $(\nu x)(u.<x>|\,Q)$

struct.cong. rules for fusions

$$(\nu x)(\overline{x}.\mathbf{nil})$$

$$\equiv \quad (\nu x)(\nu y)(<x=y>|\ \overline{x}.\mathbf{nil})$$ create fresh bound name y as alias for x

$$\equiv \quad (\nu x)(\nu y)(<x=y>|\ \overline{y}.\mathbf{nil})$$ substitute y for x

$$\equiv \quad (\nu y)(\overline{y}.\mathbf{nil})$$ remove the now-unused bound name x

$$<x=y>\ |\ <y=z> \quad \equiv \quad <x=y>\ |\ <x=z>$$ transitivity

$$<x=y> \quad \equiv \quad <y=x>$$ symmetry

⚠ $$<x=x> \quad \equiv \quad \mathbf{nil}$$ reflexivity

$$<x=y>|\ \overline{z}.P \quad \equiv \quad <x=y>|\ \overline{z}.(<x=y>|\ P)$$

$$<x=y>|\ \overline{x}.P \quad \equiv \quad <x=y>|\ \overline{y}.P$$ and other small-step substitutions

⚠ $$(\nu x)(<x=y>) \quad \equiv \quad \mathbf{nil}$$ restriction delimits scope of fusion

pi-F labels, bisimulation

$$\overline{u}.P \xrightarrow{\overline{u}} P \qquad u.P \xrightarrow{u} P \qquad \overline{u}.P \,|\, u.Q \xrightarrow{\tau} P @ Q$$

closed w.r.t. contexts and $\equiv$

Bisimulation is standard, augmented for fusion contexts:

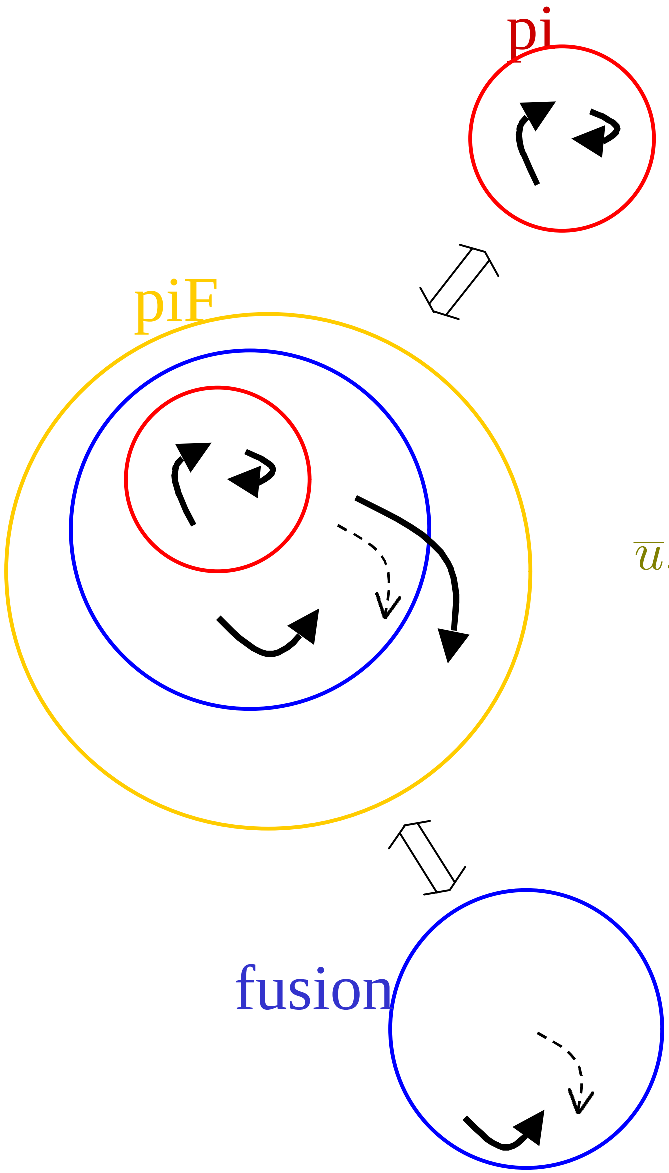
a relation S is an *open bisimulation* iff whenever PSQ then

- P, Q have same interface (same datums, fusions, restrictions)
  - for all x, y, if $<x=y> | P \xrightarrow{\alpha} P_1$ then $<x=y> | Q \xrightarrow{\alpha} Q_1$ and $P_1 \mathcal{S} Q_1$
  - and vice versa

To avoid the quantification, we add a *fusion transition*

$$\overline{u}.P \,|\, v.Q \xrightarrow{?u=v} P @ Q$$

embedding results 1



$$\overline{u}.{<}y{>}P \mid u.(x)Q \;\searrow\; P \mid Q\{^{y}\!/_{x}\}$$

$$\overline{u}.P \mid u.Q \;\searrow\; P @ Q$$

$$(\nu x)(\overline{u}.{<}x{>}P \mid u.{<}y{>}Q \mid R)$$

$$\searrow\; P\{^{y}\!/_{x}\} \mid Q\{^{y}\!/_{x}\} \mid R\{^{y}\!/_{x}\} \; *$$

embedding results 1b

(1) Structural congruence for pi and fusion embeddings

$P \equiv Q$ iff $P^* \equiv Q^*$

(2) Reaction relation for pi embedding

$P \searrow Q$ iff $P^* \searrow Q^*$

(3) Reaction relation for fusion embedding. Need constraint *

$P \searrow Q$ implies $P^* \searrow Q^*$

$P^* \searrow Q$ implies $\exists R, \vec{x}.\ (\nu\vec{x})P \searrow R$ and $R^* \equiv (\nu\vec{x})Q$

fusion calculus

$\overline{u}.\langle y \rangle P \qquad u.\langle x \rangle Q \qquad (\nu x)R$

$\overline{u}x.P$	$\xrightarrow{\overline{u}x}$	P	standard output
$(\nu x)\overline{u}x.P$	$\xrightarrow{(x)\overline{u}\,x}$	P	standard bound output
$\overline{u}x.P \mid uy.Q$	$\xrightarrow{!x=y}$	$P \mid Q$	interaction results in a fusion
$\overline{u}x.P \mid uy.Q \mid R$	$\xrightarrow{!x=y}$	$P \mid Q \mid R$	fusion has potentially global effect
$(\nu x)(\overline{u}x.P \mid uy.Q \mid R)$	$\xrightarrow{\mathbf{1}}$	$P\{y/x\} \mid Q\{y/x\} \mid R\{y/x\}$	delimit extent of fusion

embedding results 2

$$\overline{u}x.P \xrightarrow{\overline{u}\,x} P \quad \cdots\cdots\cdots \quad \overline{u}.(<x> \mid P^*) \xrightarrow{\overline{u}} <x> \mid P^*$$

$$(\nu x)\overline{u}x.P \xrightarrow{(x)\overline{u}\,x} P \quad \cdots\cdots\cdots \quad (\nu x)\overline{u}.(<x> \mid P^*) \xrightarrow{\overline{u}} (\nu x)(<x> \mid P^*)$$

$$\overline{u}x.P \mid uy.Q \xrightarrow{!x=y} P \mid Q \quad \cdots\cdots\cdots \quad \overline{u}.(<x> \mid P^*) \mid u.(<y> \mid Q^*) \xrightarrow{\tau} <x=y> \mid P^* \mid Q^*$$

embedding results 2b

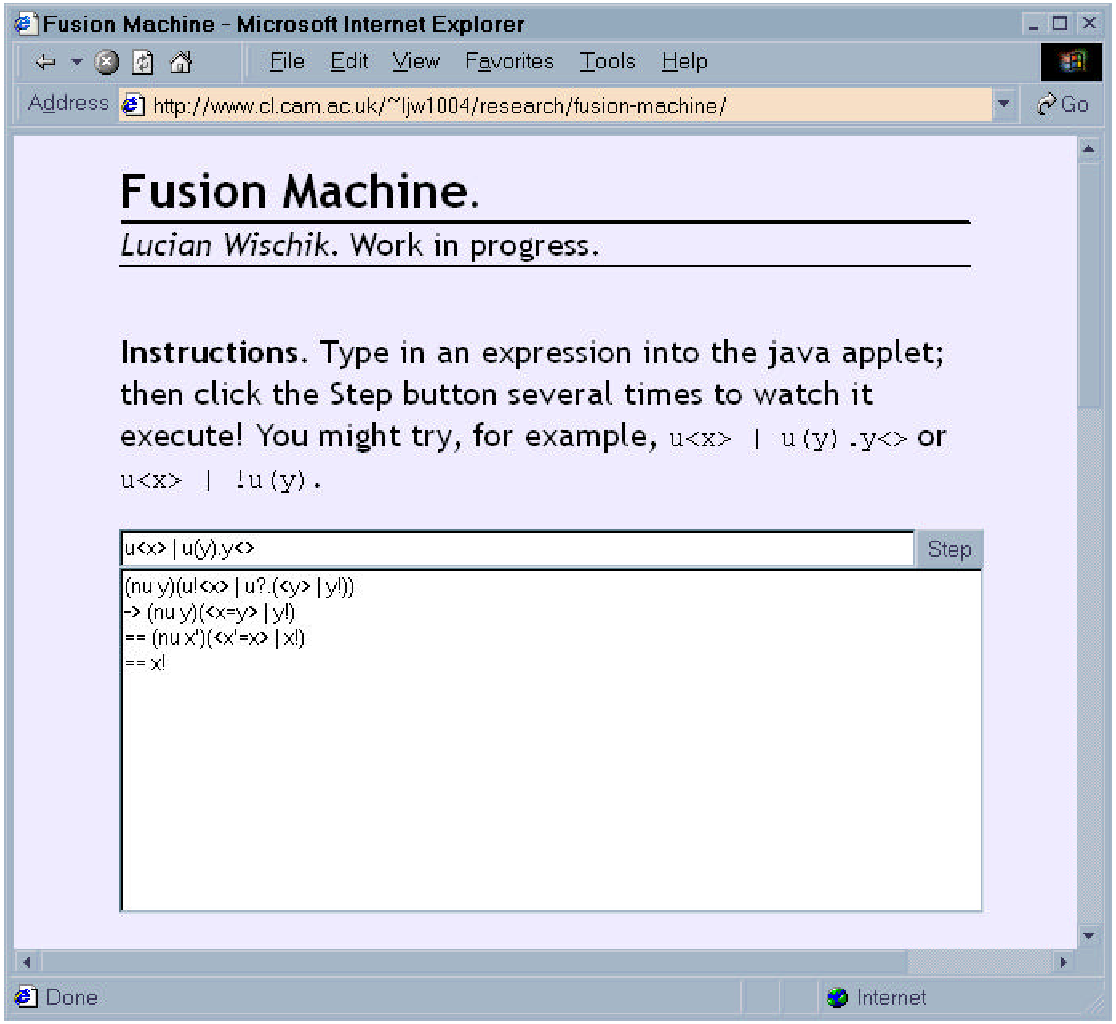
(1) fusion-calculus hyperequivalence coincides with pi-F bisimulation on its embedded image.

- proof: take hyperequivalence, close it under all interfaces, and the result is a pi-F bisimulation. And *vice versa*.

(2) pi-calculus open bisimulation is strictly weaker than pi-F bisimulation.

- “open” bisimulation [S96] : one that’s closed w.r.t. subsitututions — or fusions
- proof [VP98]: in following expression, no pi-context can make *x*=*y*; but pi-F can: $(\nu x\, y)(\overline{u}.{<}x\, y{>} \mid P\,)$

ongoing work



- (replication)
- implementation based on explicit fusions
- lambda calculus encoding, using datums to represent lambda variables
- use explicit fusions in other areas ("fusion systems")