Reflexive CHAM and Join Calculus

- Attempt to overcome "distribution" issues in π-Calculus, yet keeping formal investigation results.
- Bridge gap between theory and practice, and form the basis for a practical programming language.
- Imposes locality and adds reflexion to generic CHAM of Boudol and Berry.
- Combines restriction, reception, and replication (of π-Calculus) into a single (joint) receptor definition.

Reflexive Chemical Abstract Machine and Join Calculus

$x\langle y \rangle$ atom

$M \mid M$ molecule

e.g.

ready⟨laser⟩

ready⟨laser⟩ | job⟨1⟩

D or $J \rhd P$ reaction

e.g.

ready⟨printer⟩ | job⟨file⟩ ▷ printer⟨file⟩

$D \vdash M$ higher-order solutions

Printer Example

$\emptyset \vdash \underline{ready}\langle laser\rangle, job\langle 1\rangle, job\langle 2\rangle$

$\rightleftharpoons \emptyset \vdash ready\langle laser\rangle \mid job\langle 1\rangle, job\langle 2\rangle$

Let $D = \underline{ready}\langle printer\rangle \mid \underline{job}\langle file\rangle \triangleright$ $\underline{printer}\langle \underline{file}\rangle$

$D \vdash ready\langle laser\rangle \mid job\langle 1\rangle, job\langle 2\rangle$

$\rightarrow D \vdash laser\langle 1\rangle, job\langle 2\rangle$

Reflexion

def D in P molecule

$\emptyset \vdash$ def D in $\underline{ready}\langle laser\rangle \mid \underline{job}\langle 1\rangle \mid \underline{job}\langle 2\rangle$

$\rightarrow D \vdash ready\langle laser\rangle \mid job\langle 1\rangle \mid job\langle 2\rangle$

NAMES, PROCESSES, DEFINITIONS

P	$\stackrel{def}{=}$	$x\langle\vec{v}\rangle$	asynchronous polyadic message
		$\text{def } D \text{ in } P$	definition of new names
		$P \mid P$	parallel composition
J	$\stackrel{def}{=}$	$x\langle\vec{v}\rangle$	join patterns
		$J \mid J$	
D	$\stackrel{def}{=}$	$J \triangleright P$	definitions matching join patterns to processes.
		$D \wedge D$	

Variable Types

Received Variables (join patterns)

$$rv(x\langle\tilde{v}\rangle) \stackrel{def}{=} \{u \in \tilde{v}\}$$

$$rv(J \mid J') \stackrel{def}{=} rv(J) \cup rv(J')$$

Defined Variables (join patterns and definitions)

$$dv(x\langle\tilde{v}\rangle) \stackrel{def}{=} \{x\}$$

$$dv(J \mid J') \stackrel{def}{=} dv(J) \cup dv(J')$$

$$dv(J \triangleright P) \stackrel{def}{=} dv(J)$$

$$dv(D \wedge D') \stackrel{def}{=} dv(D) \cup dv(D')$$

FREE VARIABLES (processes and definitions)

$$fv(J \triangleright P) \stackrel{def}{=} dv(J) \cup (fv(P) - rv(J))$$

$$fv(D \wedge D') \stackrel{def}{=} fv(D) \cup fv(D')$$

$$fv(x\langle \tilde{v} \rangle) \stackrel{def}{=} \{x\} \cup \{u \in \tilde{v}\}$$

$$fv(\text{def } D \text{ in } P) \stackrel{def}{=} (fv(P) \cup fv(D)) - dv(D)$$

$$fv(P \mid P') \stackrel{def}{=} fv(P) \cup fv(P')$$

Operational Semantics
Structural Equivalence

(str-join) $\vdash P|Q \rightleftharpoons \vdash P, Q$

"$\rightarrow$" heating
"$\leftarrow$" cooling

(str-and) $D \wedge E \vdash \rightleftharpoons D, E \vdash$

(str-def) $\vdash \text{def } D \text{ in } P \rightleftharpoons D_{\sigma_{dv}} \vdash P_{\sigma_{dv}}$

(red) $J \triangleright P \vdash J_{\sigma_{rv}} \Rightarrow J \triangleright P \vdash P_{\sigma_{rv}}$

σ_{dv} instantiates port variables $dv(D)$ to fresh names.

σ_{rv} substitutes transmitted names for distinct received variables $rv(J)$.

Examples

(1) $\mathrm{def}\ x\langle u\rangle \triangleright y\langle u\rangle\ \mathrm{in}\ P$

messages on local name x in P are forwarded to the outside as messages on y.

(2) $\mathrm{def}\ y\langle u\rangle \triangleright x\langle u\rangle\ \mathrm{in}\ \mathrm{def}\ x\langle u\rangle \triangleright y\langle u\rangle\ \mathrm{in}\ P$

messages on local name x are forwarded in two steps on the external name x.

(internal x needs renaming)

(3) $\mathrm{def}\ x_1\langle u\rangle \mid x_2\langle v\rangle \triangleright x\langle u,v\rangle\ \mathrm{in}\ P$

performs multiplexing of messages on x whose parts are supplied on x_1 and x_2.

Examples (Continued)

(4) $\mathsf{def}\ x\langle v\rangle \mid y\langle k\rangle \triangleright K\langle v\rangle\ \mathsf{in}\ P$

models π-calculus-like channels, as values are sent on x and requests for values are sent on y.

(5) $\mathsf{def}\ s\langle\rangle \triangleright P \wedge s\langle\rangle \triangleright Q\ \mathsf{in}\ s\langle\rangle$

expresses internal non-determinism $P + Q$

(6) $\mathsf{def}\ once\langle\rangle \mid y\langle v\rangle \triangleright x\langle v\rangle\ \mathsf{in}$
$y\langle 1\rangle \mid y\langle 2\rangle \mid y\langle 3\rangle \mid once\langle\rangle$

uses the message on once as a lock for non-determinism $x\langle 1\rangle + x\langle 2\rangle + x\langle 3\rangle$

Recursion Example

def loop<> ▷ P | loop<> in loop<>

replicates the process P (!P), starting a new copy each time the definition is used.

Reference Cell in the Join Calculus

$$\text{def } mkcell\langle v_0, k_0\rangle \triangleright \left(\begin{array}{l} \text{def } get\langle k\rangle \mid s\langle v\rangle \triangleright k\langle v\rangle \mid s\langle v\rangle \\ \wedge\ set\langle u, k\rangle \mid s\langle v\rangle \triangleright K\langle\rangle \mid s\langle u\rangle \\ \text{in } s\langle v_0\rangle \mid k_0\langle get, set\rangle \end{array} \right)$$

in P