Applied π Calculus

Paper presentation, by
Willard Thór Rafnsson

Department of Computer Science, Aalborg university, Denmark

October 4, 2007

- Paradigmic purity in calculi is a formally appealing feature.
- Problems rarely fit perfectly into a single paradigm.
- Solution: *adjust the calculus to our needs*.
  - Sπ calculus.
- Adjusting a calculus takes time: examine the impact of our modifications, etc.

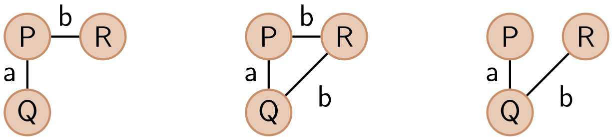
Wouldn't it be nice to have a calculus which is particularly easy to tailor to our needs?

The authors:

Martín Abadi, Cédric Fournet. *"Mobile Values, New Names, and Secure Communication"*. Proceedings of the 28th ACM Symposium on Principles of Programming Languages, January 2001.

The π calculus:

- A process calculus.
- Developed by Robin Milner (amongst others).
- Extends CCS with name-passing.



π calculus syntax:

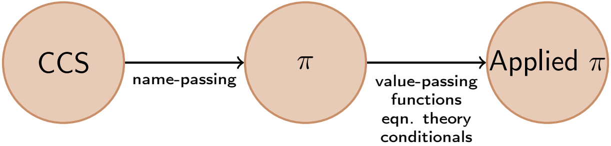
$$
\begin{array}{lcl}
P & ::= & x(y).P \\
 & | & \overline{x}\langle y\rangle.\ P \\
 & | & P \mid P \\
 & | & P + P \\
 & | & (\nu x)P \\
 & | & !P \\
 & | & 0
\end{array}
$$

Example (Coffee-delivery system)

$$
\begin{array}{lcl}
CS & \stackrel{\text{def}}{=} & (\nu n)\overline{c}\langle n\rangle.n.\overline{p}.CS \\
CM & \stackrel{\text{def}}{=} & c(s).\overline{s}.CM \\
DCS & \stackrel{\text{def}}{=} & (\nu c)(CS \mid \cdots \mid CS \mid !CM)
\end{array}
$$

The Applied π calculus extends the π calculus with:

- value-passing
- primitive functions
- equations among terms
- conditionals



Abstractions easily expressed in the Applied π calculus:

- `let` P `in` Q
- `pair(x,y)`, and thereby lists and other datastructures
- nonces, hashing functions, enc / dec functions, etc.

This also brings the π calculus a large step closer to more practical applicability.

Applied π calculus Syntax:

$$\begin{array}{lcl} a, b, c & \in & \mathbf{Nam}, \\ x, y, z & \in & \mathbf{Var}, \\ u, v, w & \in & \mathbf{Nam} \cup \mathbf{Var}, \\ f, g, h & \in & \Sigma \end{array}$$

$$\begin{array}{lcl} A & ::= & P \\ & \mid & A|A \\ & \mid & \nu a.A \\ & \mid & \nu x.A \\ & \mid & \{^M/_x\} \\ M & ::= & a \\ & \mid & x \\ & \mid & f(M_1, \ldots, M_k) \end{array} \qquad \begin{array}{lcl} P & ::= & u(x).P \\ & \mid & \overline{u}\langle x\rangle.P \\ & \mid & P|P \\ & \mid & \nu a.P \\ & \mid & !P \\ & \mid & 0 \\ & \mid & \texttt{if}\ M = N\ \texttt{then}\ P\ \texttt{else}\ P \end{array}$$

Example (Gambling away)

$$
\begin{array}{lcl}
ACC & \stackrel{\text{def}}{=} & o(r).\ \texttt{if}\ gt(-r,b)=true\\
& & \qquad\texttt{then}\ \overline{o}\langle 0\rangle.ACC\\
& & \qquad\texttt{else}\ \overline{o}\langle -r\rangle.ACC\{^{sum(b,r)}/_{b}\}\\
PLR & \stackrel{\text{def}}{=} & \overline{o}\langle -10\rangle.o(c).\ \texttt{if}\ c=10\\
& & \qquad\texttt{then}\ \overline{d}\langle c\rangle.d(c').\overline{o}\langle c'\rangle.PLR\\
DLR & \stackrel{\text{def}}{=} & d(v).\overline{g}\langle v\rangle.g(w).\overline{d}\langle w\rangle.DLR\\
CH & \stackrel{\text{def}}{=} & g(v).\overline{g}\langle x\rangle.CH\\
HSE & \stackrel{\text{def}}{=} & \nu g.(DLR \mid CH\{^{0}/_{x}\} \mid CH\{^{5}/_{x}\} \mid CH\{^{20}/_{x}\})\\
GBL & \stackrel{\text{def}}{=} & \nu d.\left(\left(\nu o.\left(PLR \mid ACC\{^{100}/_{b}\}\right)\right) \mid HSE\right)
\end{array}
$$

Note (proven later): if $x \notin fv(M)$, then

$$A\{M/_{x}\} \equiv \nu x.(\{M/_{x}\} \mid A) \equiv \texttt{let}\ x = M\ \texttt{in}\ A$$

Example (Something is amiss)

$$\begin{aligned} SQ &\overset{\text{def}}{=} a(x).\overline{a}\langle \mathrm{square}(x)\rangle.SQ \\ A &\overset{\text{def}}{=} \overline{a}\langle y\rangle.a(z).\overline{z} \end{aligned}$$

Individually, these processes are syntactically valid, but upon interaction, $SQ \mid A$, a strange thing happens.

The Applied π calculus relies on a Milner-like *sort system*.

- `Integer`
- `Key`
- `Data`
- $\texttt{channel}\langle\tau\rangle$, where τ is a sort.
- ...

Sort system – General idea:

$$\begin{array}{rcl} \mathbf{T} & = & \text{set of sorts} \\ \Gamma & : & \mathbf{Nam} \cup \mathbf{Var} \longrightarrow \mathbf{T} \end{array}$$

Define:

- a well-behaved process, *wbp*
- a sound system of type rules

You now have a type system for inferring whether a process is well-behaved:

$$=_\xi, \Sigma, \Gamma \vdash A : \mathit{wbp}$$

Example (Something is amiss, revisited)

$$\begin{aligned} SQ &\stackrel{\text{def}}{=} a(x).\overline{a}\langle \text{square}(x)\rangle.SQ \\ A &\stackrel{\text{def}}{=} \overline{a}\langle y\rangle.a(z).\overline{z} \end{aligned}$$

Let

$$\Gamma = \left\{ \begin{array}{ll} a : \texttt{channel}\langle\texttt{Integer}\rangle, & x : \texttt{Integer}, \\ z : \texttt{channel}\langle\rangle, & y : \texttt{Integer} \end{array} \right\}$$

However, since $\Gamma(z) = \texttt{channel}\langle\rangle$, $\Gamma(a) = \texttt{channel}\langle\texttt{Integer}\rangle$ and z is bound to an input from a, it follows that

$$=_\xi, \Sigma, \Gamma \not\vdash SQ \mid A : wbp$$

Your task is to:

- Register all functions you wish to use in signature Σ,
- Develop a suitable sort system w. type environment Γ
- Provide an equational theory $=_\xi$ (an equiv. rel. on terms).

Example (Pair)

$$\Sigma = \{\mathrm{pair}, \mathrm{fst}, \mathrm{snd}\}$$

$$\mathrm{fst}(\mathrm{pair}(x, y)) =_\xi x,$$
$$\mathrm{snd}(\mathrm{pair}(x, y)) =_\xi y.$$

pair(M, N) is abreviated by (M, N). Whenever pairs are used from now on, we assume these facilities to be present in $\Sigma, =_\xi$.

Example (Asymmetric Encryption)

Keys for encryption and decryption differ.

$$\Sigma = \{\text{enc}, \text{dec}, \text{ sk}, \text{ pk}\}$$

$$\text{dec}(\text{enc}(x, \text{pk}(y)), \text{sk}(y)) =_{\xi} x$$

A process which shares its public key and decrypts a received message by use of its secret key can now be written as follows:

$$\nu s.(\overline{a}\langle \text{pk}(s)\rangle \mid b(x).\overline{c}\langle \text{dec}(x, \text{sk}(s))\rangle)$$

Example (Public-key digital signatures)

$$\begin{array}{rcl} \Sigma & = & \{check, sign, sk, pk\} \\ \mathtt{ok} & \in & \mathbf{T} \end{array}$$

$$\mathrm{check}(x, \mathrm{sign}(x, \mathrm{sk}(y)), \mathrm{pk}(y)) =_{\xi} \mathtt{ok}$$

A filter which drops forged messages can now be written as follows:

$$\begin{array}{rl} & (\nu s.\{^{\mathrm{pk}(s)}/y\} \mid \overline{a}\langle(M, \mathrm{sign}(M, \mathrm{sk}(s)))\rangle) \\ \mid & a(x).\mathtt{if}\ \mathrm{check}(\mathrm{fst}(x), \mathrm{snd}(x), y) = \mathtt{ok}\ \mathtt{then}\ \overline{b}\langle \mathrm{fst}(x)\rangle \end{array}$$

The strength in the Applied π calculus lies in

- Value-passing
- The *signature* Σ,
- The *equational theory* $=_\xi$,
- The *active substitution* $\{^M/_x\}$.

Other interesting concepts yet discussed:

- Context $C[\_]$,
- Frame φ,
- Static- and Observational Equivalence, $\approx_s$ and $\approx$
- Bisimilarity $\approx_l$

Definition (Closed Extended Process)

A closed $\Longleftarrow$ all $x \in$ **Var** in A are:

- bound (by a restriction), or
- defined by an active substitution ($\{^M/_x\}$) in A.

Recall the public-key digital signature example:

$$\begin{array}{rcl} A & \stackrel{\text{def}}{=} & (\nu s.\{^{\mathsf{pk}(s)}/y\} \mid \overline{a}\langle(M, \mathsf{sign}(M, \mathsf{sk}(s)))\rangle) \\ & \mid & a(x).\mathtt{if}\ \ \mathsf{check}(\mathsf{fst}(x), \mathsf{snd}(x), y) = \mathtt{ok}\ \ \mathtt{then}\ \overline{b}\langle \mathsf{fst}(x)\rangle \end{array}$$

A is closed, since $x, s \in \mathrm{bv}(A)$, and y is defined by an active substitution.

Definition (Context)

$C[\_]$ is an A or P with a "hole". $C[\_]$ *closes* $A \Longleftarrow C[A]$ is closed.

Example (Context and Closure)

$$
\begin{aligned}
C[\_] &= \nu a.\nu b.[\text{– the hole –}] \\
A &= \overline{a}\langle b\rangle.b.0 \mid a(c).\overline{c}.0 \\
C[A] &= \nu a.\nu b.(\overline{a}\langle b\rangle.b.0 \mid a(c).\overline{c}.0) \\
\mathrm{fn}(C[A]) \cup \mathrm{fv}(C[A]) &= \emptyset
\end{aligned}
$$

Definition (Structural Equivalence)

$\equiv$ is the smallest equivalence relation on A's that is:

- Closed by α-conversion on a's and x's
- Closed by application of $C[\_]$,

such that:

PAR-0	A	$\equiv A \mid 0$
PAR-A	$A \mid (B \mid C)$	$\equiv (A \mid B) \mid C$
PAR-C	$A \mid B$	$\equiv B \mid A$
REPL	$!P$	$\equiv P \mid !P$
NEW-0	$\nu n.0$	$\equiv 0$
NEW-C	$\nu u.\nu v.A$	$\equiv \nu v.\nu u.A$
NEW-PAR	$A \mid \nu u.B$	$\equiv \nu u.(A \mid B)$, if $u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)$

$$
\begin{array}{lll}
\text{ALIAS} & \nu x.\{^M/_x\} & \equiv 0 \\
\text{SUBST} & \{^M/_x\} \mid A & \equiv \{^M/_x\} \mid A\{^M/_x\} \\
\text{REWRITE} & \{^M/_x\} & \equiv \{^N/_x\}, \text{ if } \Sigma \vdash M = N
\end{array}
$$

Example (Let)

For $x \notin fv(M)$,

$$
\begin{array}{rcll}
A\{^M/_x\} & \equiv & A\{^M/_x\} \mid 0 & \text{by PAR-0} \\
& \equiv & 0 \mid A\{^M/_x\} & \text{by PAR-C} \\
& \equiv & (\nu x.\{^M/_x\}) \mid A\{^M/_x\} & \text{by ALIAS} \\
& \equiv & \nu x.(\{^M/_x\} \mid A\{^M/_x\}) & \text{by NEW-PAR} \\
& \equiv & \nu x.(\{^M/_x\} \mid A) & \text{by SUBST} \\
& \equiv & \mathtt{let}\ x = M\ \mathtt{in}\ A &
\end{array}
$$

Definition (Internal Reduction)

$\rightarrow$ is the smallest relation on A's that is:

- Closed by $\equiv$,
- Closed by application of $C[\_]$,

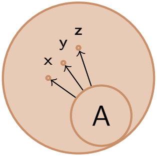
such that:

$$
\begin{array}{lrl}
\text{COMM} & \overline{a}\langle x\rangle.P \mid a(x).Q & \rightarrow P \mid Q \\
\text{THEN} & \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \rightarrow P \\
\text{ELSE} & \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \rightarrow Q, \text{ if } \Sigma \not\vdash M = N
\end{array}
$$

Definition (Frame)

φ is an A which P's have been replaced by 0s. dom(φ) is the set of names that φ exports.

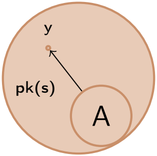
$$\begin{array}{lcl} \varphi(A) & = & \tilde{n}.\sigma \\ \mathrm{dom}(\varphi(A)) = \mathrm{dom}(A) & = & \{x \in \mathbf{Var} \mid x \in \mathrm{fn}(A) \wedge x \text{ subst. in } A\} \end{array}$$



Thus, $\varphi(A)$ denotes “what information A leaks to the world and where to”, while dom(A) denotes “where A leaks information to”.

Example

$$
\begin{array}{lcl}
A & \stackrel{\text{def}}{=} & (\nu s.\{^{\mathsf{pk}(s)}/y\} \mid \overline{a}\langle(M, \mathsf{sign}(M, \mathsf{sk}(s)))\rangle) \\
 & \mid & a(s).\mathtt{if}\ \ \mathsf{chk}(\mathsf{fst}(x), \mathsf{snd}(x), y) = \mathtt{ok}\ \ \mathtt{then}\ \ \overline{b}\langle \mathsf{fst}(x)\rangle \\
\varphi(A) & = & \nu s.\{^{\mathsf{pk}(s)}/y\} \\
\mathsf{dom}(A) & = & \{y\}
\end{array}
$$



Definition (Observational Equivalence)

$\approx$ is the largest binary symmetric relation $\mathcal{R}$ where A, B are closed, $\mathrm{dom}(A) = \mathrm{dom}(B)$ and s.t.

1. $A \Downarrow a \Longrightarrow B \Downarrow a$
2. $A \rightarrow^* A' \Longrightarrow \exists B' \, [B \rightarrow^* B' \wedge A' \mathcal{R} B']$
3. $\forall C[\_]; C$ closes $A, B \, [C[A] \mathcal{R} C[B]]$

Note: $A \Downarrow a$ if $A \rightarrow^* C[\overline{a}\langle M\rangle.P]$ for some context $C[\_]$ which nay binds a.

Definition (Term equality in a frame)

M, N equal in φ, $(M = N)\varphi$, $\Longleftarrow$

- $\varphi \equiv \nu \tilde{n}.\sigma$,
- $M\sigma = N\sigma$, and
- $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$

Definition (Static Equivalence)

$\approx_s$: let φ, ψ be closed.

- $$\varphi \approx_s \psi \Longleftarrow \quad \begin{array}{l} \mathrm{dom}(\varphi) = \mathrm{dom}(\psi) \wedge \\ \forall M, N\,[(M = N)\varphi \Leftrightarrow (M = N)\psi] \end{array}$$
- $A \approx_s B \Longleftarrow \varphi A \approx_s \varphi(B)$

Note: $\approx = \approx_s$ on frames, $\approx \subset \approx_s$ otherwise. That is, $\approx \Longrightarrow \approx_s$.

By expanding $\equiv$ and $\rightarrow$, we can obtain a *Labelled Operational Semantics*. For the one given in the article, the following holds:

Definition (Labeled Bisimilarity)

$\approx_l$ is the largest binary symmetric relation $\mathcal{R}$ satisfying:

$$A\mathcal{R}B \Longrightarrow$$
$$\left(\begin{array}{l} A \approx_s B \wedge \\ (A \rightarrow A' \Longrightarrow \exists B' \, [B \rightarrow^* B' \wedge A'\mathcal{R}B']) \wedge \\ \left(\begin{array}{l} (A \xrightarrow{\alpha} A' \wedge fv(\alpha) \subseteq dom(A) \wedge bn(\alpha) \cap fn(B) = \emptyset) \\ \Longrightarrow \exists B'[B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B' \wedge A'\mathcal{R}B'] \end{array} \right) \end{array} \right)$$

Theorem (Observational Equivalence = Labeled Bisimilarity)

$\approx = \approx_l$.

- Only concurrency-specific actions are modelled using traditional π calculus abstractions.
- A very general calculus; easy to extend with desired abstractions $(\Sigma, =_\xi, \mathbf{T}, \Gamma)$
- Frames capture exactly which information is leaked from a process to the environment
- A neat framework for proving Observational Equivalence is provided.