$$a?P_1 + P_2 \parallel a!Q_1 + Q_2 \rightarrow P_1 \parallel Q_1$$

- processes which synchronise via handshake
- Limitations:
  - no data passed during communication;
  - the communication capabilities of processes are static.

**Observation 1.** A process can only communicate on the channels which are present in its description.

Contrast with a server, it communicates with us because we send it our address. In CCS this is not possible.
**CCS with value passing**

\[ a？x P_1 + P_2 \parallel a!v Q_1 + Q_2 \rightarrow P_1[v/x] \parallel Q_1 \]

- \(a？x\) is a *binder*, it binds occurrences of \(x\) in its scope;
- \(a!v\) sends some value \(v\) from a predefined set of values;
- \(P_1[v/x]\) denotes the substitution of \(v\) for all occurrences of \(x\).

**Observations:**
- differently from \(\lambda\), this is all first-order, \(v\) comes from some fixed set of values \(V\);
- \(V\) is disjoint from the set of channel names \(A\).
Fixing disjoint sets of channel names $A$ and values $V$:

$$
P ::= 0 \mid a?xP \mid a!vP \\
| P \parallel P \mid P + P \mid \nu aP \mid X \mid \mu X.P \quad (a \in A, \; v \in V)
$$

Semantics (in SOS style):

$$(a?xP_1 + P_2) \parallel (a!vQ_1 + Q_2) \rightarrow P_1[v/x] \parallel Q_1$$

$\begin{align*}
P \rightarrow P' & \quad \Rightarrow \quad P \parallel Q \rightarrow P' \parallel Q \\
\nu aP \rightarrow \nu aP'
\end{align*}$$
CCS with value passing, limitations

\[ a?xP_1 + P_2 \parallel a!vQ_1 + Q_2 \rightarrow P_1 \parallel Q_1 \]

- processes which synchronise via handshake with value passing;
- Limitations:
  - the communication capabilities of processes are static.
Pi calculus, idea

- Modify value passing CCS so that the value passed are the channel names themselves;
- This results in a calculus where communication capabilities change dynamically during computation;

\[(a?xP_1 + P_2)\parallel (a!bQ_1 + Q_2) \rightarrow P_1[b/x]\parallel Q_1\]
Pi syntax

\[ M ::= x | a | 0 | M \ || \ M | M!MM | M?xM \]
\[ \ | \ \nu x M | rep(M) | if M = M then M else M \]

- the above, like \( \lambda \), is presented as an untyped syntax
- types are much simpler than in \( \lambda \), there are only two!
  - \( \mathcal{P} \) - for processes
  - \( \mathcal{N} \) - for names
Typing

- a typing context consists of
  - a set of names $\Delta$;
  - a finite map from variables to types $\Gamma$.

$$\Delta, \Gamma \vdash M : \sigma$$

- means $M$ types with $\sigma$, assuming that its free names are contained in $\Delta$ and its free variables in $\Gamma$;

- we will consider only typeable closed terms.
Typing ctd

\[
\begin{align*}
  a \in \Delta & \quad (\text{:NAME}) & \quad \Gamma(x) = \sigma & \quad (\text{:VAR}) & \quad \Delta, \Gamma \vdash 0 : \mathcal{P} \\
  \Delta, \Gamma \vdash a : N & \quad \Delta, \Gamma \vdash x : \sigma & \quad \Delta, \Gamma \vdash \sigma : \mathcal{N} & \quad \Delta, \Gamma \vdash x : \sigma \\
  \Delta, \Gamma \vdash k : N & \quad \Delta, \Gamma \vdash l : N & \quad \Delta, \Gamma \vdash M : \mathcal{P} & \quad (\text{:OUTPREF}) \\
  \Delta, \Gamma \vdash k!lM : \mathcal{P} & \\
  \Delta, \Gamma, x : N \vdash M : \mathcal{P} & \quad \Delta, \Gamma \vdash k : N & \quad (\text{:INPREF}) \\
  \Delta, \Gamma \vdash k ? x M : \mathcal{P} & \\
  \Delta, a, \Gamma \vdash M : \mathcal{P} & \quad \Delta, \Gamma \vdash M : \mathcal{P} & \quad \Delta, \Gamma \vdash M' : \mathcal{P} & \quad (\text{:PAR}) \\
  \Delta, \Gamma \vdash \nu a M : \mathcal{P} & \quad \Delta, \Gamma \vdash M \parallel M' : \mathcal{P} & \\
  \Delta, \Gamma \vdash \nu a M : \mathcal{P} & \quad \Delta, \Gamma \vdash M \parallel M' : \mathcal{P} & \\
  \Delta, \Gamma \vdash k : N & \quad \Delta, \Gamma \vdash l : N & \quad \Delta, \Gamma \vdash P : \mathcal{P} & \quad \Delta, \Gamma \vdash Q : \mathcal{P} & \quad (\text{:IFTHENELSE}) \\
  \Delta, \Gamma \vdash \text{if } k = l \text{ then } P \text{ else } Q : \mathcal{P} &
\end{align*}
\]
Contexts

contexts, informally:

\[ C = \mathbf{\Pi} (C') \mid \text{if } k = l \text{ then } P \text{ else } C \mid \text{if } k = l \text{ then } C \text{ else } P \]

note:

- terms are closed, hence \( a ? x C \) cannot bind name variables;
- \( \nu a C \) may bind free names;
Structural congruence

\[(P \parallel Q) \parallel R \equiv P \parallel (Q \parallel R)\]

\[P \parallel Q \equiv Q \parallel P\]

\[P \parallel 0 \equiv P\]

\[\nu a \nu b P \equiv \nu b \nu a P\]

\[\nu a 0 \equiv 0\]

\[\nu a (P \parallel Q) \equiv P \parallel \nu a Q\]

\[\nu a P \equiv \nu b P[b/a] (b \notin P)\]

\[r_P(P) \equiv P \parallel r_P(P)\]

\[r_P(P \parallel Q) \equiv r_P(P) \parallel r_P(Q)\]

\[r_P(0) \equiv 0\]

\[k?x P \equiv k?y P[y/x]\]
Reduction semantics

- in our transition system, it will be useful to keep $\Delta$ during the computation as part of the state;
- one can think of $\Delta$ as the names known to this computation.

\[
\begin{align*}
\langle \Delta \mid a!bP \parallel a?xQ \rangle &\rightarrow \langle \Delta \mid P\parallel Q[b/x] \rangle \\
\langle \Delta \mid P \rangle &\rightarrow \langle \Delta \mid P' \rangle \\
\langle \Delta \mid P\parallel Q \rangle &\rightarrow \langle \Delta \mid P'\parallel Q \rangle \\
\langle \Delta, a \mid P \rangle &\rightarrow \langle \Delta, a \mid P' \rangle \\
\langle \Delta \mid \nu aP \rangle &\rightarrow \langle \Delta \mid \nu aP' \rangle
\end{align*}
\]
Dynamic communication capabilities

a simple example:

- $P_1 = a!c$;
- $P_2 = b!d.d?xP_3$;
- $Q = a?x.b?y.y!x$.

then:

$$P_1 || P_2 || Q \rightarrow P_2 || b?y.y!c \rightarrow d?xP_3 || d!c \rightarrow P_3[c/x]$$

the channel name $c$ has transferred from $P_1$ to $P_2$ without any direct communication between the two processes.
Scope extrusion

- In CCS the scoping of $\nu$ is static;
- Because of the dynamic nature of Pi communication, the scope of $\nu$ can extend during computation.
- This phenomenon is known as scope extrusion;
- Eg

$$a?x.x!c \parallel (\nu b.a!b) \rightarrow \nu b.b!c$$

- Technically, to perform the reduction, one must first extend the scope using the $\nu b. (P \parallel Q) \equiv (\nu b.P) \parallel Q (b \notin Q)$ axiom of structural congruence;
Reduction barbed congruence

- Like in CCS, we need to add a basic notion of observable in order to deal with infinite computations properly;
- Like in CCS, the *barbs* are instantaneous capability to input or output on a channel;
- Barbed congruence is the notion of contextual equivalence for Pi;
- it is the smallest congruence which is
  - barb closed;
  - a bisimulation wrt reductions.
Labelled characterisations

- The Pi-calculus has traditionally two associated LTSs;
  - early - substitution occurs at communication;
  - late - substitution occurs after communication.

- In general, late bisimilarity is finer than early;

- We will concentrate on the early variant because it is closely related with contextual equivalence;

- In fact, in our presentation early bisimilarity is a congruence and agrees with reduction barbed congruence.
Labelled characterisation, free names

$$\Delta \subseteq \Delta', \ b \in \Delta'$$

$$\langle \Delta \mid a?xP \rangle \xrightarrow{a?b} \langle \Delta' \mid P[b/x] \rangle$$  \hspace{2cm} \text{(IN)}

$$\langle \Delta \mid a!bP \rangle \xrightarrow{a!b} \langle \Delta \mid P \rangle$$  \hspace{2cm} \text{(OUT)}

$$\langle \Delta \mid P \rangle \xrightarrow{a!b} \langle \Delta \mid P' \rangle \quad \langle \Delta \mid Q \rangle \xrightarrow{a?b} \langle \Delta \mid Q' \rangle$$

$$\langle \Delta \mid P \parallel Q \rangle \xrightarrow{\tau} \langle \Delta \mid P' \parallel Q' \rangle$$  \hspace{2cm} \text{(COMM)}

$$\langle \Delta \mid P \rangle \xrightarrow{\alpha} \langle \Delta' \mid P' \rangle$$

$$\langle \Delta \mid P \parallel Q \rangle \xrightarrow{\alpha} \langle \Delta' \mid P' \parallel Q \rangle$$  \hspace{2cm} \text{(PAR)}

$$\langle \Delta, a \mid P \rangle \xrightarrow{\alpha} \langle \Delta', a \mid P' \rangle$$  \hspace{2cm} \text{(RES)}$$\quad (a \notin \alpha)$$

$$\langle \Delta \mid \nu a P \rangle \xrightarrow{\alpha} \langle \Delta' \mid \nu a P' \rangle$$

\[\text{just like CCS.}\]
Labelled characterisation, bound names

\[ \langle \Delta, b \mid P \rangle \xrightarrow{a!b} \langle \Delta, b \mid P' \rangle \quad (a \neq b) \]

\[ \langle \Delta \mid \nu b P \rangle \xrightarrow{a!(b)} \langle \Delta, b \mid P' \rangle \]

\[ \langle \Delta \mid P \rangle \xrightarrow{a!(b)} \langle \Delta, b \mid P' \rangle \quad \langle \Delta \mid Q \rangle \xrightarrow{a?b} \langle \Delta, b \mid Q' \rangle \]

\[ \langle \Delta, b \mid P \parallel Q \rangle \xrightarrow{\tau} \langle \Delta \mid \nu b (P' \parallel Q') \rangle \]

- the Open/Close system is a hack to deal properly with bound names
- the Open rule gives the ability to observe “freshness”;
- the Close rule allows communication when a bound name is outputted;
Soundness

one can show that:
- bisimilarity is a congruence;
- barbs are clearly distinguished;
- \( \tau \)'s characterise reductions;
- hence \( \sim \subseteq \cong \), bisimilar processes are contextually equivalent.
Completeness

one has to show that labelled transitions can be characterised by context interactions:

- to observe a free output $a!b$ we need to use equality testing;
- to observe a bound output $a!(b)$ we need to use inequality testing;
- $\cong \subseteq \sim$, contextually equivalent processes are bisimilar.
Asynchronous variants

- communication in CCS and the Pi-calculus is *synchronous*;
  - $a!b.P$ is blocked until there is some process ready to receive;

- this is not realistic from the point of view of modelling communication protocols which are normally asynchronous, i.e.
  - a process sends $P$ a message through a communication medium;
  - $P$ goes on doing something else;
  - at some time later $Q$ receives the message.
Asynchronous CCS

- a hack which involves restricting CCS syntactically
- \(a!b. P\) is allowed only for \(P = 0\);
- idea:
  - no process can block on output;
  - \(a?.(b! \parallel P)\)
- barbs are now restricted to outputs;
- \(a?.a! \simeq \tau\);
- the labelled characterisation has to be altered;
- the same game can be played with Pi.
Higher-order variants

- processes can pass other processes as arguments during communication;

- for the higher-order Pi-calculus, there is a nice labelled characterisation of contextual equivalence called *normal bisimilarity*, due to Davide Sangiorgi.