Axiom 9.8: The value-passing channel.

Axiom 9.9: Now, define the "next" function.

The objects of the labels.

formal parameters for which a can be introduced. We call a the subject, and a respectively a

on the other hand, is able to receive a list of values a along channel a; the list a determines

is interpreted to send a list of values a = a", which channel a. An input-label a(x),

integers or booleans, or parameters for which values can be introduced. An output-label a(y)
in value-passing CCS (VP), labels in Y are considered to contain values of some type such as

2.1.2 Value-Passing CCS

http://www.bjelida-communication.de/publ/class/la/2001/reciprocity

Axiom 3.9: CCS must data / Value-Passing CCS
\{z/\Sigma, \hat{n}/n, x/x\} D (\Sigma, n, x) a \quad \text{and} \quad D (\Sigma, n, x) a \quad \text{a transition, yielding a transition,}

\{z/\Sigma, \hat{n}/n, x/x\} D | \! | D (\Sigma, n, x) a | D (\Sigma, n, x) a \quad \text{a transition, yielding a transition,}

a transition, we can say that it is based on the two transitions, a transition, yielding a transition, the first process transmits the values \(\Sigma\), \(n\), and \(x\) to the process \(\{z/\Sigma, \hat{n}/n, x/x\} D (\Sigma, n, x) a\) and \(D (\Sigma, n, x) a\) as a concrete example, consider the process \(\{z/\Sigma, \hat{n}/n, x/x\} D (\Sigma, n, x) a\) and \(D (\Sigma, n, x) a\) in a communication.
\[ \sum_{j \in p} \mathcal{C}_{\text{out}, n} \mathcal{C} \]

Summation. Thus the defining equation for \( C \) becomes

Thus this way the use of a bound variable \( x \) is replaced by any input value, because it binds the variable \( x \), we translate it to

Now consider the prefix \( \text{tin}(x) \). To reflect the fact that it can accept

\[ (\wedge \in \mathcal{C}) \quad \mathcal{C}_{\text{out}, \mathcal{C}} \mathcal{C} \quad \mathcal{C}_{\text{out}, \mathcal{C}} \]

Consider first the parameterized constant \( \mathcal{C} \). It will become a family

\[ \text{out}(x) \quad \mathcal{C}_{\text{out}, \mathcal{C}} \mathcal{C} \]

\[ \text{tin}(x) \quad \mathcal{C}_{\text{out}, \mathcal{C}} \mathcal{C} \]

Take our very first example, the buffer cell:

B \equiv \text{DEPARTMENT}
In this case a family of defining equations:

Thus the boolean expressions internal to an agent expression have be-

\[
\begin{align*}
\text{Usetool}(j_0) & \quad \text{VWselect}(j_0) \\
\text{Usetool}(j_0) & \quad \text{VWcomplete}(j_0) \\
\text{Finish}(j_0) & \quad \text{Start}(j_0) \\
\text{Jobber} & \quad \text{In}(j_0, \text{Start}(j_0))
\end{align*}
\]

Translation is as follows:

Since Start takes a parameter, like C, we expect the second equation to

\[
\begin{align*}
\text{Start}(j_0) & \quad \text{VWcomplete}(j_0) \\
\text{Start}(j_0) & \quad \text{VWselect}(j_0) \\
\text{Jobber} & \quad \text{VWcomplete}(j_0)
\end{align*}
\]

Next, let us take the first two equations in Section 1.3 defining the
The behaviour of the components can be described operationally as follows:

\[ (s)x^{-1} p \quad (x)s^{-1} p \]

\[ (x)s^{-1} p \quad (s)x^{-1} p \]

\[ 0 \leq ? \quad (1 + ?)p^{-1} \quad (1 + ?)p \]

\[ 0 \leq ? \quad (1 + ?)p^{-1} \quad (1 + ?)p \]

In practice, one often does not write down systems in terms of rules, but provides transition-rules describing their behaviour together with the state of an event-based system.
structure of the list of values it manipulates.

notwithstanding the definition of the consumer is a bit clumsier, because its behaviour is determined by

as well. Giving a

and

that the two producers can easily be modelled in terms of (infinite families of) recursive
In the system, if there are copies of \( P \) in the environment, it is as many as there are copies of \( P \) in the system. Delayed delivery to the environment determines the behavior of the system: of each number \( i \in \mathbb{N} \), at most \( n \) copies will be delivered to the environment. Assume the number of producers is \( P(0) \), the number of consumers is \( C(0) \), and the process is \( P \). As an example, consider \( (P_1(0) \cdots P_r(0)) \) where the number of producers is replaced by an abstract parameter \( r \) times, \( \sum r \). A component for \( A \) is a component of a component, the actual number is then usually replaced by an abstract parameter \( r \) times, \( \sum r \). A third source of \( \aleph_0 \) for \( \sqrt{\aleph_0} \) is the Cantor-Bernstein theorem. Another example—another example—from pure CCS—of structurally infinite. Yet, the consumer has the potential to store as many values as it likes; it is not instance. (3) For instance, (2) yet, the consumer has the potential to store as many values as it likes; it is not instance. (2) Yet, the consumer has the potential to store as many values as it likes; it is not instance. (1) Process \( P \), for example, employs values of an infinite type, and so does \( P \). We therefore call them data-infinite. The two systems are infinite-state, that is, they yield transition graphs with an infinite number of nodes. Infinity can have various sources. (1) Process \( P \), for example, employs values of an infinite type, and so does \( P \). We therefore call them data-infinite. The two systems are infinite-state, that is, they yield transition graphs with an infinite number of nodes. Infinity can have various sources. (1)
applying VP to CCS

Reducing VP to pure CCS is usually not easy, because the values

applying VP to CCS

Reducing VP to pure CCS is usually not easy, because the values

applying VP to CCS

Reducing VP to pure CCS is usually not easy, because the values

applying VP to CCS

Reducing VP to pure CCS is usually not easy, because the values
(6) If then E, a Conditional
(5) $E \rightarrow F$, a Relabeling (f a Relabeling function)
(4) $E : T \subseteq F$, a Restriction
(3) $E \leftarrow F$, a Composition
(2) $E \leftarrow T$, a Summation (f an indexing set)
(1) $a(x) \leftarrow E \leftarrow T$, Prefixes (a $\in A$)

and also the following expressions, where $E$, $F$, $T$ are already in $+$

Then $+^+$ is the smallest set containing every agent variable $X \in A$.
Every parameterized constant $A(e_1, \ldots, e_n)$, for $e_1, \ldots, e_n$ with any $n$,

Proceeding to the general translation, let us first give the set $+^+$ of agent expressions of the full calculus. We assume that to each agent

Constant $A \in A$ is assigned an arity, a non-negative integer representing

Boolean expressions $E$ and Boolean expressions $F$, built from value variables $x, y$, and symbols $A$ and $E$ representing the number of parameters which it takes. Also we assume value expressions

Value-Passing Calculus

2.8
Furthermore, each Constant $A$ with arity $n$ has a defining equation:

\[ A(x_1, \ldots, x_n) \overset{\text{def}}{=} E \]

where the right-hand side $E$ may contain no agent variables, and no free value variables except $x_1, \ldots, x_n$ (which must be distinct).

The one-armed conditional expression in (6) is enough, because the two-armed conditional if $b$ then $E$ else $E'$ can be defined as

\[(\text{if } b \text{ then } E) + (\text{if } \neg b \text{ then } E')\]

Also, for simplicity we have assumed that every action name $a$ takes a value parameter (those which do not will require no translation).

In translating $E^+$ to $E$, we confine our attention to agent expressions which contain no free value variables $x, y, \ldots$. (If $E \in E^+$ contains $x$ free, then it can be considered as a family of expressions $E\{v/x\}$, one for each value constant $v$.) Our translation of $E^+$ into $E$ rests upon the idea that to each label $\ell$ in the full calculus corresponds a set $\{\ell_v : v \in V\}$ of labels in the basic calculus. Thus we think of a single port labelled $\ell$ as a set of ports labelled $\ell_v$, one for each value $v \in V$. 
The Value-Passing Calculus
As for Rule C6 concerning communication, we consider \( a(\alpha) \) and \( \#(\alpha) \) to complement one another. In an early semantics, the rule can thus be used without modification.

Rules C2b and C2c that restriction is performed over sets of channels:

\[
\frac{N \not\models \Phi \quad \rho \vdash N \not\models \Phi}{N \not\models \Phi \quad \rho \vdash N \not\models \Phi} \quad \frac{N \not\models \Phi \quad \rho \vdash N \not\models \Phi}{N \not\models \Phi \quad \rho \vdash N \not\models \Phi} \quad \frac{N \not\models \Phi \quad \rho \vdash N \not\models \Phi}{N \not\models \Phi \quad \rho \vdash N \not\models \Phi}
\]

Remark: As there is a clear distinction between input and output in VP, we can assume in

\[
\frac{[x/\alpha]}{\rho \vdash N \not\models \Phi} \quad \frac{[x/\alpha]}{\rho \vdash N \not\models \Phi} \quad \frac{[x/\alpha]}{\rho \vdash N \not\models \Phi}
\]

as follows:

A labelled transition system in early style for VP can be obtained from that for pure CCS in Table 2.2 by slightly modifying rules C1 and C2.