1 External Events

dat transfers are followed by *ack* acknowledgments. *req* requests evoke the return of *acs* access permissions.

$C.U.dat(e)$
$U.C.dat(e)$

> C = data transfer channel.
> e = data value transferred.
>
> $C.U.dat$ is in the alphabets of channel end $out(C)$ and the user process.
> $U.C.dat$ is in the alphabets of channel end $in(C)$ and the user process.

$C.U.ack$
$U.C.ack$

> C = data transfer channel.
>
> $C.U.ack$ is in the alphabets of channel end $in(C)$ and the user process.
> $U.C.ack$ is in the alphabets of channel end $out(C)$ and the user process.

$C.U.req$
$U.C.req$

> C = access transfer channel.
>
> $C.U.req$ is in the alphabets of channel end $out(C)$ and the user process.
> $U.C.req$ is in the alphabets of channel end $in(C)$ and the user process.

$C.U.acs(F)$
$U.C.acs(F)$

> C = access transfer channel
> F = channel end, access to which is transferred.
>
> $C.U.acs$ *(F)* is in the alphabets of channel end $in(C)$ and the user process.
> $U.C.acs$ *(F)* is in the alphabets of channel end $out(C)$ and the user process.

2 Messages

2.1 Data Transfer Messages

ack is an acknowledgement of a *dat* message.

$Q.P^O.dat(e, C)$
$Q.P^I.dat(e, C)$

> P = port associated with channel end $out(C)$.
> Q = port associated with $in(C)$.
> e = data value transferred.
> C = data transfer channel.
>
> *dat* output is in the alphabets of port P, channel end $out(C)$, and the message delivery system. *dat* input is in the alphabets of port Q, channel end $in(C)$, and the message delivery system.

$P.Q^O.ack(C)$
$P.Q^I.ack(C)$

> P = port associated with channel end $out(C)$.
> Q = port associated with $in(C)$.
> C = data transfer channel.
>
> *ack* output is in the alphabets of port Q, channel end $in(C)$, and the message delivery system. *ack* input is in the alphabets of port P, channel end $out(C)$, and the message delivery system.

2.2 Access Transfer Messages

req requests access to some available channel end be transferred to a locally allocated port. "Acknowledgment" of the request goes through the control transfer messages introduced below.

$Q.P^O.req(R, C)$
$Q.P^I.req(R, C)$

> P = port associated with channel end $out(C)$.
> Q = port associated with $in(C)$.
> R = recipient port, to which access will be transferred.
> C = data transfer channel.
>
> *req* output is in the alphabets of ports P and R, channel end $out(C)$, and the message delivery system. *req* input is in the alphabets of port Q, channel end $in(C)$, and the message delivery system.

2.3 Control Transfer Messages

The *ini* message is sent from a port process to its successor on termination of the former. It carries information needed to initialise the new port.

$P.S^O.ini(E, Q, s, D)$
$P.S^I.ini(E, Q, s, D)$

P = port formerly associated with channel end E.
S = successor port to P.
E = a channel end.
Q = port associated with partner of E.
s = value defining a set of events pending on E at time of transfer.
D = access transfer channel.

ini output is in the alphabets of channel end E, port P, and the message delivery system. *ini* input is in the alphabets of channel ends E and $out(D)$, port S, and the message delivery system.

2.4 Port Repointing Messages

These messages are used during transfer of access to a channel end to inform the port associated with the *partner* channel end of the move. *quo* is an acknowledgement of a *chg* message.

$Q.P^O.chg(E, S, D)$
$Q.P^I.chg(E, S, D)$

P = port formerly associated with channel end E.
Q = port associated with partner of E.
S = successor port to P.
E = a channel end.
D = access transfer channel.

chg output is in the alphabets of channel end $in(D)$, port P, and the message delivery system. *chg* input is in the alphabets of port Q and the message delivery system.

$P.Q^O.quo(E, D)$
$P.Q^I.quo(E, D)$

P = port formerly associated with channel end E.
Q = port associated with partner of E.
E = a channel end.
D = access transfer channel.

quo output is in the alphabets of port Q, and the message delivery system. *quo* input is in the alphabets of port P and the message delivery system.

3 The Channel-End Process

Associated with each C-channel are an input channel-end process and an output channel-end process. These processes are largely formal devices, with no unique physical location. Their alphabets incorporate the events of the external interface of the C-channel. They are responsible for correct sequencing of the individual messages involved in communication on the C-channel.

3.1 Output channel-end

The alphabet of an output channel end process $out(C)$ contains

Control transfer events

$$\_.\_^{O}.ini(out(C), \_, \_, \_)$$
$$\_.\_^{I}.ini(out(C), \_, \_, \_)$$

Data transfer events

$$C.U.dat\ \_$$
$$\_.\_^{O}.dat(\_, C)$$
$$\_.\_^{I}.ack(C)$$
$$U.C.ack$$

Access transfer events

$$C.U.req$$
$$\_.\_^{O}.req(\_, C)$$
$$\_.\_^{I}.ini(\_, \_, \_, C)$$
$$U.C.acs\ \_$$

In this alphabet, the data and access transfer events together will be called "communication events". Communication events excluding the input events on channels $\_.\_^{I}$ will be called "non-input communication events". The traces of $out(C)$ satisfy

1. The trace restricted to control transfer events and non-input events (ie, excluding input communication events) is a prefix of a sequence built from the atomic subsequences

   $\langle c \rangle$, where c is any non-input communication event, and
   $\langle \_.\_^{O}.ini(out(C), \_, s, \_),\ \_.\_^{I}.ini(out(C), \_, \_, \_) \rangle$

2. The trace restricted to communication events is a prefix of a sequence built from the atomic subsequences

$\langle C.U.dat(e),\ \_.\_^{O}.dat(e,C),\ \_.\_^{I}.ack(C), U.C.ack\rangle$, and
$\langle C.U.req,\ \_.\_^{O}.req(R,C),\ R.\_^{I}.ini(F,\_,\_,C), U.C.acs(F)\rangle$

3. If one of the atomic subsequences in 2 is interrupted by an *ini* output, the s field in that output contains sufficient information to determine how to resume the subsequence at the point of interruption.

3.2 Input channel-end

The alphabet of an input channel end process $in(C)$ contains

Control transfer events

$\_.\_^{O}.ini(in(C),\_,\_,\_)$
$\_.\_^{I}.ini(in(C),\_,\_,\_)$

Data transfer events

$\_.\_^{I}.dat(\_,C)$
$U.C.dat\ \_$
$C.U.ack$
$\_.\_^{O}.ack(C)$

Access transfer events

$\_.\_^{I}.req(\_,C)$
$U.C.req$
$C.U.acs\ \_$
$\_.\_^{O}.chg(\_,\_,C)$

The traces of $in(C)$ satisfy

1. The trace restricted to control transfer events and non-input events (ie, excluding input communication events) is a prefix of a sequence built from the atomic subsequences

   $\langle c\rangle$, where c is any non-input communication event, and
   $\langle \_.\_^{O}.ini(in(C),\_,s,\_),\ \_.\_^{I}.ini(in(C),\_,\_,\_)\rangle$

2. The trace restricted to communication events is a prefix of a sequence built from the atomic subsequences

   $\langle \_.\_^{I}.dat(e,C), U.C.dat(e), C.U.ack,\ \_.\_^{O}.ack(C)\rangle$, and
   $\langle \_.\_^{I}.req(R,C), U.C.req, C.U.acs(F),\ \_.\_^{O}.chg(F,R,C)\rangle$

3. (As for output channel-end).

Requirement 1 implies that the only event that can occur between output of an ini message and its subsequent receipt is an input communication event. During this interval the channel end is "moving", and most types of event are disabled. Inputs from the message system cannot be disabled here, because (as it will turn out) this would violate the important eager readership assumption. Requirement 3 implies that resumption of a channel-end process after moving to a different processor doesn't depend on any miraculous non-local transfer of information. For example, suppose the move of an input channel-end, implied by insertion of an ini subsequence, occurred after reception of a $dat(e, C)$ message from the message delivery system, but before delivery of the datum e to the user process. The s field in the ini message would have to include the value of e.

Eventually it will be necessary to put some further restrictions on these traces, to avoid other forms of non-locality.

4 The Port Process

A port process may be "initialised" or "uninitialised". An initialised port is associated from the outset with some channel end $out(C)$ or $in(C)$. It has a partner initialised port, associated with opposite other channel end. An uninitialised port starts off in the "free pool" of ports. Port P is allocated from this pool when it successfully engages in a $\_.\_^{O}.req(P, \_)$ event.

The alphabet of a port process P contains

Main control thread events

$$\_.\_^O.req(P, \_)$$
$$P.\_^I.ini\ \_$$
$$\_.P^O.ini\ \_$$

Repointing events

$$P.\_^I.chg\ \_$$
$$\_.P^O.quo\ \_$$
$$\_.P^O.chg\ \_$$
$$P.\_^I.quo\ \_$$

Communication events

$$\_.P^O.dat\ \_$$
$$P.\_^I.ack\ \_$$
$$\_.P^O.req\ \_$$
$$P.\_^I.dat\ \_$$
$$\_.P^O.ack\ \_$$
$$P.\_^I.req\ \_$$

The traces of P satisfy

1. The trace restricted to main thread events and *chg* output events is a prefix of some sequence

   $$\langle \_.P^O.chg(\_, R, D),\ R.P^O.ini(\_, \_, \_, D)\rangle$$

   if P is an initialised port, or

   $$\langle \_.\_^O.req(P, \_),\ P.\_^I.ini\ \_,\ \_.P^O.chg(\_, R, D),\ R.P^O.ini(\_, \_, \_, D)\rangle$$

   if it is uninitialised. Moreover, $req(P, \_)$ output can only occur as the first event of the full trace, and *ini* output can only occur as the last.

2. In the trace restricted to *ini* input events and output events on the channels $\_.P^O$, an *ini* event can only appear in the first position of the sequence, and a *chg* output event can only appear in the last. (ie, no outputs are initiated before initialisation, or after initiation of a "move").

3. The trace restricted to repointing events is a prefix of a sequence built from any number of atomic subsequences of the form

   $$\langle P.Q^I.chg(E, \_, D),\ Q.P^O.quo(E, D)\rangle$$

terminated by *one* subsequence from

$$\langle \_.P^O.chg\ \_,\ P.\_^I.quo\ \_\rangle,$$
$$\langle \_.P^O.chg\ \_,\ P.\_^I.chg\ \_\rangle, \text{ or } \langle P.\_^I.chg\ \_,\ \_.P^O.chg\ \_\rangle$$

All incoming *chg* messages are acknowledged. The three possibilities for termination embody the only slightly subtle part of the repointing protocol.

4. In any event from

$$Q.P^O.dat\ \_$$
$$Q.P^O.ack\ \_$$
$$Q.P^O.req\ \_$$
$$Q.P^O.chg\ \_$$
$$\_.P^O.ini(\_, Q, \_, \_)$$

appearing in a trace, the Q field must have the same value as the Q field in the most recent preceding event of type

$$P.\_^I.chg(Q, \_)$$

If there is no preceding *chg* input, the Q field must match either the initial value for the partner port (if P is initialised), or the value in the trace's (compulsory)

$$P.\_^I.ini(\_, Q, \_, \_)$$

event (if it is uninitialised). These constraints impose that communications are always sent to the active partner port.

5. The trace must be constructed from

$$\_.\_^O.req(P, \_)$$
$$P.\_^I.ini(out(C), \_, \_, \_)$$
$$\_.P^O.ini(out(C), \_, \_, \_)$$
$$P.\_^I.chg\ \_$$
$$\_.P^O.chg(\_, out(C), \_)$$
$$\_.P^O.dat(\_, C)$$
$$P.\_^I.ack(C)$$
$$\_.P^O.req(\_, C)$$

or (exclusively) from

$$\_.\_^O.req(P, \_)$$
$$P.\_^I.ini(in(C), \_, \_, \_)$$
$$\_.P^O.ini(in(C), \_, \_, \_)$$
$$P.\_^I.chg\ \_$$

$$\begin{array}{l}
.P^{O}.chg(,in(C),_) \\
.P^{I}.dat(,C) \\
P._^{O}.ack(C) \\
.P^{I}.req(,C)
\end{array}$$

where the value of the C field is common to all events in a trace. (MISSING quo EVENTS HERE). If P is an initialised port with channel end $out(C)$ the first set is applicable; if it is initialised with $in(C)$ the second set applies. A port will only support one channel end throughout its life. (IT'S LIKELY THAT I WANT TO REMOVE SOME UNNECESSARY RESTRICTIONS ON INPUT EVENTS).

5 "Solving" the Trace Restrictions

Suppose C is some channel name, and $\boldsymbol{P}, \boldsymbol{Q}$ are non-empty sequences of port names, with $n = \#\boldsymbol{P}, m = \#\boldsymbol{Q}$. Introduce the sets of event sequences:

$$\begin{array}{rcl}
\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D}) & = & \{\langle Q_j.P_i^O.chg(out(C), P_{i+1}, D)\rangle\} \\
\mathcal{P}^{\in}_{\rangle|}(\mathcal{D}) & = & \{\langle Q_j.P_i^I.chg(out(C), P_{i+1}, D)\rangle\} \\
\mathcal{P}^{\ni}_{\rangle|}(\mathcal{D}) & = & \{\langle P_i.Q_j^O.quo(out(C), D)\rangle\} \\
\mathcal{P}^{\triangle}_{\rangle|}(\mathcal{D}) & = & \{\langle P_i.Q_j^I.quo(out(C), D), \\
 & & P_{i+1}.P_i^O.ini(out(C), Q_j, s, D), \\
 & & P_{i+1}.P_i^I.ini(out(C), Q_j, s, D)\rangle \mid s \in Contexts\}
\end{array}$$

for $0 \leq i < n-1, 0 \leq j < m$, and

$$\begin{array}{rcl}
\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D}) & = & \{\langle P_i.Q_j^O.chg(in(C), Q_{j+1}, D)\rangle\} \\
\mathcal{Q}^{\in}_{\rangle|}(\mathcal{D}) & = & \{\langle P_i.Q_j^I.chg(in(C), Q_{j+1}, D)\rangle\} \\
\mathcal{Q}^{\ni}_{\rangle|}(\mathcal{D}) & = & \{\langle Q_j.P_i^O.quo(in(C), D)\rangle\} \\
\mathcal{Q}^{\triangle}_{\rangle|}(\mathcal{D}) & = & \{\langle Q_j.P_i^I.quo(in(C), D), \\
 & & Q_{j+1}.Q_j^O.ini(in(C), P_i, s, D), \\
 & & Q_{j+1}.Q_j^I.ini(in(C), P_i, s, D)\rangle \mid s \in Contexts\}
\end{array}$$

for $0 \leq i < n, 0 \leq j < m-1$, and

$$\begin{array}{rcl}
\overline{\mathcal{P}}^{4}_{ij}(D, D') & = & \{\langle P_i.Q_j^I.chg(in(C), Q_{j+1}, D'), \\
 & & P_{i+1}.P_i^O.ini(out(C), Q_{j+1}, s, D), \\
 & & P_{i+1}.P_i^I.ini(out(C), Q_{j+1}, s, D)\rangle \mid s \in Contexts\} \\
\overline{\mathcal{Q}}^{4}_{ij}(D, D') & = & \{\langle Q_j.P_i^I.chg(out(C), P_{i+1}, D),
\end{array}$$

$$Q_{j+1}.Q_j^O.ini(in(C), P_{i+1}, s, D'),$$
$$Q_{j+1}.Q_j^I.ini(in(C), P_{i+1}, s, D')\rangle \mid s \in Contexts\}$$

for $0 \leq i < n-1, 0 \leq j < m-1$. Next, define recursively

$$\mathcal{A}_{\rangle|}(\mathcal{D}) =$$
$$\left(\bigcup_{D'} \mathcal{A}_{\rangle-\infty,|}(\mathcal{D}')\mathcal{P}^{\ni}_{\rangle-\infty,|}(\mathcal{D}')\mathcal{P}^{\triangle}_{\rangle-\infty,|}(\mathcal{D}')\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{P}^{\in}_{\rangle|}(\mathcal{D})\right) \cup$$
$$\left(\bigcup_{D'} \mathcal{B}_{\rangle,|-\infty}(\mathcal{D}')\mathcal{Q}^{\ni}_{\rangle,|-\infty}(\mathcal{D}')((\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{P}^{\in}_{\rangle|}(\mathcal{D})) + \mathcal{Q}^{\triangle}_{\rangle,|-\infty}(\mathcal{D}'))\right) \cup$$
$$\left(\bigcup_{D',D''} \mathcal{C}_{\rangle-\infty,|-\infty}(\mathcal{D}',\mathcal{D}'')((\overline{\mathcal{P}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}')\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{P}^{\in}_{\rangle|}(\mathcal{D})) + \overline{\mathcal{Q}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}''))\right)$$

for $0 \leq i < n-1, 0 \leq j < m$, and

$$\mathcal{B}_{\rangle|}(\mathcal{D}) =$$
$$\left(\bigcup_{D'} \mathcal{A}_{\rangle-\infty,|}(\mathcal{D}')\mathcal{P}^{\ni}_{\rangle-\infty,|}(\mathcal{D}')(\mathcal{P}^{\triangle}_{\rangle-\infty,|}(\mathcal{D}') + (\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{Q}^{\in}_{\rangle|}(\mathcal{D})))\right) \cup$$
$$\left(\bigcup_{D'} \mathcal{B}_{\rangle,|-\infty}(\mathcal{D}')\mathcal{Q}^{\ni}_{\rangle,|-\infty}(\mathcal{D}')\mathcal{Q}^{\triangle}_{\rangle,|-\infty}(\mathcal{D}')\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{Q}^{\in}_{\rangle|}(\mathcal{D})\right) \cup$$
$$\left(\bigcup_{D',D''} \mathcal{C}_{\rangle-\infty,|-\infty}(\mathcal{D}',\mathcal{D}'')(\overline{\mathcal{P}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}') + (\overline{\mathcal{Q}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}'')\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D})\mathcal{Q}^{\in}_{\rangle|}(\mathcal{D})))\right)$$

for $0 \leq i < n, 0 \leq j < m-1$, and

$$\mathcal{C}_{\rangle|}(\mathcal{D},\mathcal{D}') =$$
$$\left(\bigcup_{D''} \mathcal{A}_{\rangle-\infty,|}(\mathcal{D}'')\mathcal{P}^{\ni}_{\rangle-\infty,|}(\mathcal{D}'')((\mathcal{P}^{\triangle}_{\rangle-\infty,|}(\mathcal{D}'')\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D})) + \mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D}'))\right) \cup$$
$$\left(\bigcup_{D''} \mathcal{B}_{\rangle,|-\infty}(\mathcal{D}'')\mathcal{Q}^{\ni}_{\rangle,|-\infty}(\mathcal{D}'')(\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D}) + (\mathcal{Q}^{\triangle}_{\rangle,|-\infty}(\mathcal{D}'')\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D}'))\right) \cup$$
$$\left(\bigcup_{D'',D'''} \mathcal{C}_{\rangle-\infty,|-\infty}(\mathcal{D}'',\mathcal{D}''')((\overline{\mathcal{P}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}'')\mathcal{P}^{\infty}_{\rangle|}(\mathcal{D})) + (\overline{\mathcal{Q}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}''')\mathcal{Q}^{\infty}_{\rangle|}(\mathcal{D}')))\right)$$

for $0 \leq i < n-1, 0 \leq j < m-1$. Here, juxtaposition of sequence sets implies concatentation of elements, and '+' means interleaving of elements. The conventions: $\mathcal{A}_{-\infty,\prime} = \mathcal{B}_{\prime,-\infty} = \mathcal{C}_{-\infty,-\infty} = \mathcal{P}^{\ni}_{-\infty,\rangle} = \mathcal{P}^{\triangle}_{-\infty,\rangle} = \overline{\mathcal{P}}^{\triangle}_{-\infty,\rangle-\infty} = \mathcal{Q}^{\ni}_{\rangle,-\infty} =$

$\mathcal{Q}^{\triangle}_{\rangle,-\infty} = \overline{\mathcal{Q}}^{\triangle}_{\rangle-\infty,-\infty} = \{\langle\rangle\}$ for $i \geq 0$, and $\mathcal{A}_{-\infty,\rangle} = \mathcal{B}_{-\infty,\rangle} = \mathcal{C}_{-\infty,\rangle-\infty} = \mathcal{A}_{\rangle,-\infty} = \mathcal{B}_{\rangle,-\infty} = \mathcal{C}_{\rangle-\infty,-\infty} = \emptyset$ for $i > 0$, supply the base cases for the recurrence.

Finally we construct:

$$
\begin{aligned}
&\mathcal{M}(\mathcal{C}, \boldsymbol{P}, \boldsymbol{Q}) = \\
&\quad \left(\bigcup_{D} \mathcal{A}_{\rangle-\infty,|}(\mathcal{D}) \mathcal{P}^{\ni}_{\rangle-\infty,|}(\mathcal{D}) \mathcal{P}^{\triangle}_{\rangle-\infty,|}(\mathcal{D}) \right) \cup \\
&\quad \left(\bigcup_{D} \mathcal{B}_{\rangle,|-\infty}(\mathcal{D}) \mathcal{Q}^{\ni}_{\rangle,|-\infty}(\mathcal{D}) \mathcal{Q}^{\triangle}_{\rangle,|-\infty}(\mathcal{D}) \right) \cup \\
&\quad \left(\bigcup_{D,D'} \mathcal{C}_{\rangle-\infty,|-\infty}(\mathcal{D}, \mathcal{D}') (\overline{\mathcal{P}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}) + \overline{\mathcal{Q}}^{\triangle}_{\rangle-\infty,|-\infty}(\mathcal{D}')) \right)
\end{aligned}
$$

where $i = n-1, j = m-1$.