Some CSP Proof Techniques Applied to Virtual Channel Protocols

D.B. Carpenter

Department of Electronics and Computer Science,
Southampton University,
Southampton, S09 5NH

November 18, 1994

1 Models and Simulations: I. Simple and Extended Transition Systems

State transition models are very often used to give operational semantics to concurrent systems. Here transition models will be assume a less fundamental role—they simply encode particular CSP processes in a convenient way. The main advantage of a transition model representation is that it makes available the powerful technique of *simulation* (and *bisimulation*) for establishing relationships and equivalences between processes. A simulation is a binary relation between the state-spaces of two transition models, which has some commutative relationship with the transition behaviour of the models.

There is a great deal of freedom in the definition of state transition models, especially for encoding non-deterministic processes. From our viewpoint, all that really matters is that useful simulation results should exist between the types of model chosen. Three (???) different types of model will be considered here.

Simple Transition Models (TMs) are represented by a tuple (S, A, T) where S is a state space, A is an alphabet of events, and T is a transition function. T is a tertiary relation from $S \times A \times S$ with the property

$$((i, e, i') \in T \wedge (i, e, i'') \in T) \Rightarrow i' = i''.$$

Usually T will regarded as a function from S to the space of *partial* functions from A to S. So $T(i)$ denotes such a partial function—evidently $(e \mapsto i') \in T(i)$ means that the system can move from state i to state i' through occurence of event e.

Extended Transition Models (ETMs) are a generalisation of TMs which include unlabelled arcs. They are represented by a tuple (S, A, T, U). S, A and T have the same role as for a TM, whereas U is a binary relation on S. U will be treated as a function from S to Pow S. $U(i)$ is a set of states to which i can move *without* engaging in any external event.

Acceptance Transition Models (ATMs) are a second generalisation of TMs, which introduce non-determinism in a different way. Represented by the tuple (S, A, T, W), S, A and T again follow their definitions for a TM. W is a function from S to Pow Pow A with the property

$$\bigcup_{w \in W(i)} w = \mathrm{dom}\ T(i).$$

Informally, when an ATM is in state i, it has an associated hidden variable whose value is some element w of $W(i)$. If the environment offers to engage in some event e from w, the ATM will accept that event and undergo the transition defined by $i' = T(i)(e)$ (and choose a new hidden value from $W(i')$). But if the environment only offers events outside w, the ATM refuses these events, even if they are in dom $T(i)$.

The semantics of these models, and their properties, will be discussed more fully in the following subsections. Before that, I'll try to motivate working with such a variety of different types of model.

TMs are good for representing deterministic processes, or for representing prefix-closed sets of traces. Deterministic processes represented as TMs can be composed in parallel in a straightforward way to build new TMs; conversely a deterministic process will often factorise cleanly into two or more parallel deterministic processes. Equivalences between the trace-sets represented by two TMs can be established through a particularly simple type of simulation. For such reasons it seems wise to work with TMs whenever they suffice.

At some stage, however, an abstraction step is needed, where some internal events are concealed. This is where an ETM is needed. A TM yields an ETM in a very straightforward way upon concealment of some subset of its alphabet. Unfortunately, ETMs aren't particularly easy to manipulate. Also, it isn't very easy to show what specifications they satisfy. For example, their traces aren't encoded in a simple way.

On the other hand, ATMs are relatively easy to work with—we will give simple rules for establishing that they satisfy certain specifications. In later subsections we give a simulation result which can be used for establishing that a process represented by an ETM implements a process represented by an ATM.

Notice, by the way, that the transition function, T, for both an ETM and an STM satisfies the same restriction as that for for a TM—one way in which non-determinism *won't* be introduced here is by allowing more than one arc with the same event labelling to emerge from a state.

1.1 Simple Transition Models

A TM can be viewed as a device for encoding a prefix-closed set of traces: $\mathcal{T}$, say. We have

$$\mathcal{T} \subset A^* \wedge ((s < s' \wedge s' \in \mathcal{T}) \Rightarrow s \in \mathcal{T}).$$

Suppose $\mathcal{S}$ is a function from $\mathcal{T}$ to some state space S. Define

$$\mathcal{S}^{-1}(i) = \{\underline{s} \mid \mathcal{S}(\underline{s}) = i\}$$

and

$$T(i) = \{(\underline{e}, \underline{j}) \mid \exists s : \mathcal{S}^{-1}(i).\ s\hat{\ }\langle \underline{e} \rangle \in \mathcal{S}^{-1}(\underline{j})\}. \tag{1}$$

Now (S, A, T) is a simple transition model of $\mathcal{T}$ iff

$$\forall i.\ \forall (e, j) : T(i).\ \forall s : \mathcal{S}^{-1}(i).\ s\hat{\ }\langle e \rangle \in \mathcal{S}^{-1}(j) \tag{2}$$

In terms of the transition model, $\mathcal{S}^{-1}(i)$ is the set of traces leading the system to state i. $(e, j) \in T(i)$ means that the system can move from state i to state j by executing event e. The requirement 1 states that if the system can get to state i after some history, then move to state j by executing e next, then *any* history leading to state i also allows this. In particular, this implies that $T(i)$ is a partial function. $\mathcal{S}(\langle\rangle)$ is the initial state for the transition model which generates $\mathcal{T}$.

1.1.1 Specifications

Suppose $\mathcal{S}$ is a partial function from A^* to S satisfying

$$\begin{aligned} \mathcal{S}(\langle\rangle) &= i, \\ (s\hat{\ }\langle e\rangle \mapsto j) \in \mathcal{S} &\Leftrightarrow s \in \operatorname{dom} \mathcal{S} \wedge (e \mapsto j) \in T(\mathcal{S}(s)), \end{aligned}$$

then the (deterministic) process generated by (A, S, T) from initial state i satisfies

$$tr \in \operatorname{dom} \mathcal{S}.$$

1.1.2 Simulations

Suppose (S, A, T) and (S', A, T') are two TMs. Also, suppose there exists a relation R from S to S' such that

$$i\ R\ i' \Rightarrow (\forall (e, j) : T(i).\ (\exists j'.\ (e, j') \in T'(i') \wedge j\ R\ j')). \tag{3}$$

Then R is a *simulation*, and if $i\ R\ i'$ the trace set generated by (S, A, T) from initial state i is a subset of the trace set generated by (S', A, T') from initial state i. If R^{-1} is also a simulation, then R is a *bisimulation*, and the two trace sets are equal.

1.1.3 Trace predicates, and composition

A set of traces can be regarded as a predicate on traces. A very common form of specification takes the form of the conjunction of a set of predicates, each applied to the process's trace restricted to some sub-alphabet. Suppose we have a specification of the form

$$\bigwedge_n ((tr \rhd A_n) \in \mathcal{T}_n)$$

and (S_n, A_n, T_n) is a TM generating $\mathcal{T}_n$ from initial state i_n. Then the specification above is equivalent to

$$(tr \rhd (\bigcup_n A_n)) \in \mathcal{T}$$

where $\mathcal{T}$ is generated by the TM (S, A, T) with

$$\begin{aligned} A &= \bigcup_n A_n \\ S &= S_1 \times S_2 \times \ldots \\ T(i_1, i_2, \ldots) &= \{(\underline{e} \mapsto (\underline{j_1}, \underline{j_2}, \ldots)) \mid \bigwedge_n Accept(A_n, T_n, \underline{e}, i_n, \underline{j_n})\} \end{aligned}$$

from initial state $(i_1, i_2, \ldots)$. Here

$$Accept(A, T, e, i, j) \Leftrightarrow (\mathbf{if}\ (e \in A)\ \mathbf{then}\ (e \mapsto j) \in T(i)\ \mathbf{else}\ j = i)$$

It may be helpful to present an explicit procedure for constructing $T(i_1, i_2, \ldots)$. This procedure may be used in cases where not *too* many systems are being composed. Note that very often the transition function is only needed on some subset of the composite state space.

Construct the set $D = \text{dom}\ T_1(i_1) \cup \text{dom}\ T_2(i_2) \cup \ldots$

For each $e \in D$:

if there is any $k \in \{1, 2, \ldots\}$ such that $e \in A_k$ but $e \notin \text{dom}\ T_k(i_k)$
this e isn't in dom $T(i_1, i_2, \ldots)$.

Otherwise
for each k,
if $e \in A_k$ set j_k to $T_k(i_k)(e)$, or
if $e \notin A_k$ set j_k to i_k.
Add $(e \mapsto (j_1, j_2, \ldots))$ to $T(i_1, i_2, \ldots)$.

1.1.4 Association with deterministic CSP processes

In CSP, a deterministic process is completely determined by its traces, so it can also be determined by a TM which generates those traces from some initial state. In fact, the set of deterministic CSP process (one for each initial state in S) defined in this way by (S, A, T) is given by the mutual recursion

$$\mu X : A.\ (e : \text{dom}\ T(i) \rightarrow X(T(i)(e)))$$

With this interpretation, the composition operation in the previous section is just CSP parallel composition.

1.2 Extended Transition Models

An ETM is a tuple (S, A, T, U). T is defined in the same way as for a simple TM. U is a *nilpotent* binary relation over S. The term "nilpotent" is used here in a weak sense meaning

$$\forall i : S.\ \exists n : \text{Nat}.\ \tilde{U}^n\{i\} = \emptyset$$

where $\tilde{U}X = \bigcup_{i \in X} U(i)$. We associate a set of CSP processes with an ETM, one for each element of S (any of which can be chosen as initial state) by the recursion

$$\mu X : A.\ P(U(i), \text{dom}\ T(i), T(i))(X) \tag{4}$$

where

$$\begin{aligned}
P(u, v, t)(X) &= \begin{cases} R(v, t)(X), & u = \emptyset \\ Q(u)(X) \sqcap (Q(u)(X) \,\square\, R(v, t)(X)), & \text{otherwise} \end{cases} \\
Q(u)(X) &= \bigsqcap_{j \in u} X(j) \\
R(v, t)(X) &= (e : v \rightarrow X(t(e)))
\end{aligned}$$

This definition has been chosen specifically to ensure that concealment of events has a simple representation. The nilpotency of U ensures that the processes defined in 4 are divergence-free. Note that a simple TM, interpreted as a model for a deterministic process, is recovered by setting $U = \emptyset$.

1.2.1 Concealment

The semantics of the ETM is chosen to facilitate concealement. If $Proc$ is the set of processes associated with an ETM (S, A, T, U), and

$$Proc'(i) = Proc(i) \setminus C$$

for some $C \subset A$, then (S, A, T', U') is an ETM for the system $Proc'$, where

$$\begin{aligned} T'(i) &= (A - C) \lhd T(i) \\ U'(i) &= U(i) \cup \text{ran } (C \lhd T(i)). \end{aligned}$$

1.2.2 Simulations

In this section we introduce a simulation relation wherebye the deterministic process associated with some simple TM can implement the process associated with some ETM. Of course, this is only possible if the latter system of processes is also deterministic (although this may not be obvious in the ETM representation). Suppose (S, A, T, U) is an ETM and (S', A, T') is a simple TM. The relation G from S to S' is a simulation if

$$\begin{aligned} i \; G \; i' \quad \Rightarrow \quad & ((\text{dom } T(i) \subset \text{dom } T'(i')) \wedge \\ & (\forall e : \text{dom } T(i).\; T(i)(e) \; G \; T'(i')(e)) \wedge \\ & (\forall j : U(i).\; j \; G \; i') \wedge \\ & (U(i) = \emptyset \Rightarrow \text{dom } T(i) = \text{dom } T'(i'))). \end{aligned}$$

If G is a simulation, and $i \; G \; i'$, and $Proc$ is the system of processes associated with the ETM, and $Proc'$ is the set of process associated with the TM, then

$$Proc(i) \sqsupseteq Proc'(i') \tag{5}$$

This kind of simulation isn't a direct generalisation of the traces-model simulation in section ??. The inequality here refers to the full refusals model. In fact, since $Proc'$ is deterministic, this inequality implies also equality (because deterministic processes are maximal in the refusals model), so in the special case $U = \emptyset$, this simulation is as strong as the previous bisimulation.

2 Channel Multiplexer Example

This section describes an application of the results of section 1, up to an including the results in 1.2. The application is to a system of communication channels. Each channel alternates transmission of forward and backward messages; it repeatedly accepts a forward message through an event label p, delivers it through an event labelled q, accepts a backward message through an event labelled r and delivers it through an event labelled s. In CSP the individual channel process can be written as

$$CHAN = \mu X.\; \Box_x p.x \rightarrow q.x \rightarrow \Box_y r.y \rightarrow s.y \rightarrow X$$

Sometimes the return message may be simply an acknowledgement, leading to an implementation of synchronous unidirectional channels. Often many logical channels have to be multiplexed over a single bidirectional line. We assume the availability of a line in each direction, each with an associated output queuing mechanism whose details are abstracted by regarding queue plus communication line as a FIFO. Messages accepted by the FIFO are labelled with the associated channel and the direction of the FIFO (f or b

for forward and backward lines), superfixed with O or I according to whether the event is an output *to* the FIFO or and input *from* the FIFO.

$$\begin{aligned} \mathit{FIFO}(C,f,q) &= (\square_{c\in C} \square_x c.f^O.x \to \mathit{FIFO}(C,f,q\,\hat{}\,\langle c.f^I.x\rangle))\,\square \\ &\quad (\textbf{if } q \neq \langle\rangle \textbf{ then } q_0 \to \mathit{FIFO}(C,f,q') \textbf{ else STOP}) \end{aligned}$$

Here C is the set of channels recognised and q is the set of messages currently in the fifo—represented here, for brevity, as a sequence of pending input events (recollect that **STOP** is the unit of $\square$, so when q is empty, outputs remain possible). For each channel we introduce in output buffer and an input buffer

$$\mathit{OBUF} = \mu X.\ \square_x p.x \to f^O.x \to \square_y b^I.y \to s.y \to X$$

$$\mathit{IBUF} = \mu X.\ \square_x f^I.x \to q.x \to \square_y r.y \to b^O.y \to X$$

The aim is to show that the system of buffers and FIFOs implements the set of channels:

$$\left(\left(\mathop{\|}_{c\,\in\,\mathit{Chans}} c : \mathit{OBUF}\right) \| \mathit{FIFO}(\mathit{Chans},f,\langle\rangle) \,\| \left(\mathop{\|}_{c\,\in\,\mathit{Chans}} c : \mathit{IBUF}\right) \| \mathit{FIFO}(\mathit{Chans},b,\langle\rangle)\right) \setminus H = \mathop{\|}_{c\,\in\,\mathit{Chans}} c : \mathit{CHAN}$$

where H is the set of internal events $\{\_.f^O.\_,\ \_.f^I.\_,\ \_.b^O.\_,\ \_.b^I.\_\}$.

2.1 Transition systems for buffers and FIFOs

First we write down simple TMs for the (deterministic) components of the system [1]. The channel processes and the buffers can be modelled on the same state space

$$S^B = \alpha_0 \mid \alpha_1 \textbf{ of } \mathit{Values} \mid \alpha_2 \mid \alpha_3 \textbf{ of } \mathit{Values}$$

where Values is the set of values that can be transmitted on a channel. The transition functions for output and input buffers are

$$\begin{aligned} T^O_c(\alpha_0) &= \{(c.p.\underline{x} \mapsto \alpha_1(\underline{x}))\} \\ T^O_c(\alpha_1(x)) &= \{(c.f^O.x \mapsto \alpha_2)\} \\ T^O_c(\alpha_2) &= \{(c.b^I.\underline{x} \mapsto \alpha_3(\underline{x}))\} \\ T^O_c(\alpha_3(x)) &= \{(c.s.x \mapsto \alpha_0)\} \\ \\ T^I_c(\alpha_0) &= \{(c.f^I.\underline{x} \mapsto \alpha_1(\underline{x}))\} \\ T^I_c(\alpha_1(x)) &= \{(c.q.x \mapsto \alpha_2)\} \\ T^I_c(\alpha_2) &= \{(c.r.\underline{x} \mapsto \alpha_3(\underline{x}))\} \\ T^I_c(\alpha_3(x)) &= \{(c.b^O.x \mapsto \alpha_0)\}. \end{aligned}$$

A fifo has state space $S^F = CV^*$, where $CV = \mathit{Chans} \times \mathit{Values}$, and transition function

$$\begin{aligned} T^F_f(m) &= \{(\underline{c}.f^O.\underline{x} \mapsto m\,\hat{}\,\langle(\underline{c},\underline{x})\rangle), \\ &\quad\ (\underline{c}.f^I.\underline{x} \mapsto m' \mid m \neq \langle\rangle \wedge m_0 = (\underline{c},\underline{x}))\} \end{aligned}$$

[1] In principle one should prove that these systems model the CSP processes introduced above. This could be done by using CSP proof rules to determine the trace sets, then defining a mapping from traces to the state spaces, as explained in section 1.1. Such a degree of formality, when the correspondence is obvious anyway, seems inappropriate here.

According to section 1.1.3, the complete system can be represented on a state space which is the product of all the component state spaces. Where several of the components are closely related, and indexed by elements of some set such as *Chans*, it is convenient to group the associated factors together in a function on the index space. So the composite state space can be taken as

$$S^{MUX} = (Chans \to S^B) \times S^F \times (Chans \to S^B) \times S^F$$

where the four factors correspond to output buffers, forward fifo, input buffers and backward fifo respectively. According to the expression in 1.1.3, the transition function on this system is given by

$$
\begin{array}{l}
T^{MUX}(O,m,I,n) = \\
\quad \begin{array}{ll}
\{\ (\underline{c}.p.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, m, I, n)) & |\, O(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.f^O.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, m\widehat{\ }\langle(\underline{c},\underline{x})\rangle, I, n)) & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.f^I.\underline{x} \mapsto (O, m', I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, n)) & |\, m \neq \langle\rangle \wedge m_0 = (\underline{c},\underline{x}) \wedge I(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.q.\underline{x} \mapsto (O, m, I \oplus \{(\underline{c} \mapsto \alpha_2)\}, n)) & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.r.\underline{x} \mapsto (O, m, (I \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, n)) & |\, I(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.b^O.\underline{x} \mapsto (O, m, I \oplus \{(\underline{c} \mapsto \alpha_0)\}, n\widehat{\ }\langle(\underline{c},\underline{x})\rangle)) & |\, I(\underline{c}) = \alpha_3(\underline{x}), \\
\ \ (\underline{c}.b^I.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, m, I, n')) & |\, n \neq \langle\rangle \wedge n_0 = (\underline{c},\underline{x}) \wedge O(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.s.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_0)\}, m, I, n)) & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \}
\end{array}
\end{array}
$$

The initial state is $(\{(\_ \mapsto \alpha_0)\}, \langle\rangle, \{(\_ \mapsto \alpha_0)\}, \langle\rangle)$. Concealing internal events gives

$$
\begin{array}{l}
T^{MUX2}(O,m,I,n) = \\
\quad \begin{array}{ll}
\{\ (\underline{c}.p.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, m, I, n)) & |\, O(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.q.\underline{x} \mapsto (O, m, I \oplus \{(\underline{c} \mapsto \alpha_2)\}, n)) & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.r.\underline{x} \mapsto (O, m, (I \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, n)) & |\, I(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.s.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_0)\}, m, I, n)) & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \}
\end{array} \\
U^{MUX2}(O,m,I,n) = \\
\quad \begin{array}{ll}
\{\ (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, m\widehat{\ }\langle(\underline{c},\underline{x})\rangle, I, n) & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (O, m', I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, n) & |\, m \neq \langle\rangle \wedge m_0 = (\underline{c},\underline{x}) \wedge I(\underline{c}) = \alpha_0, \\
\ \ (O, m, I \oplus \{(\underline{c} \mapsto \alpha_0)\}, n\widehat{\ }\langle(\underline{c},\underline{x})\rangle) & |\, I(\underline{c}) = \alpha_3(\underline{x}), \\
\ \ (O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, m, I, n') & |\, n \neq \langle\rangle \wedge n_0 = (\underline{c},\underline{x}) \wedge O(\underline{c}) = \alpha_2\ \}
\end{array}
\end{array}
$$

Now introduce the model

$$
\begin{array}{l}
T^{CHANS}(C) = \\
\quad \begin{array}{ll}
\{\ (\underline{c}.p.\underline{x} \mapsto C \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}) & |\, C(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.q.\underline{x} \mapsto C \oplus \{(\underline{c} \mapsto \alpha_2)\}) & |\, C(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.r.\underline{x} \mapsto C \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}) & |\, C(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.s.\underline{x} \mapsto C \oplus \{(\underline{c} \mapsto \alpha_0)\}) & |\, C(\underline{c}) = \alpha_3(\underline{x})\ \}
\end{array}
\end{array}
$$

Definition 2.1 *The relationship $(O,m,I,n)\ G\ C$ holds iff for any c one of the following (mutually exclusive) clauses obtains*

$$
\begin{array}{l}
C(c) = \alpha_0 \wedge \\
\qquad O(c) = \alpha_0 \wedge m \rhd \{(c,\_)\} = \langle\rangle \wedge I(c) = \alpha_0 \wedge n \rhd \{(c,\_)\} = \langle\rangle,
\end{array} \tag{6}
$$

$$
\begin{array}{l}
C(c) = \alpha_1(x) \wedge \\
\qquad O(c) = \alpha_1(x) \wedge m \rhd \{(c,\_)\} = \langle\rangle \wedge I(c) = \alpha_0 \wedge n \rhd \{(c,\_)\} = \langle\rangle,
\end{array} \tag{7}
$$

$$C(c) = \alpha_1(x) \wedge$$

$$\begin{aligned}
&\quad O(c) = \alpha_2 \wedge m \rhd \{(c, _)\} = \langle (c, x) \rangle \wedge I(c) = \alpha_0 \wedge n \rhd \{(c, _)\} = \langle\rangle, && (8)\\
&C(c) = \alpha_1(x) \wedge\\
&\quad O(c) = \alpha_2 \wedge m \rhd \{(c, _)\} = \langle\rangle \wedge I(c) = \alpha_1(x) \wedge n \rhd \{(c, _)\} = \langle\rangle, && (9)\\
&C(c) = \alpha_2 \wedge\\
&\quad O(c) = \alpha_2 \wedge m \rhd \{(c, _)\} = \langle\rangle \wedge I(c) = \alpha_2 \wedge n \rhd \{(c, _)\} = \langle\rangle, && (10)\\
&C(c) = \alpha_3(x) \wedge\\
&\quad O(c) = \alpha_2 \wedge m \rhd \{(c, _)\} = \langle\rangle \wedge I(c) = \alpha_3(x) \wedge n \rhd \{(c, _)\} = \langle\rangle, && (11)\\
&C(c) = \alpha_3(x) \wedge\\
&\quad O(c) = \alpha_2 \wedge m \rhd \{(c, _)\} = \langle\rangle \wedge I(c) = \alpha_0 \wedge n \rhd \{(c, _)\} = \langle (c, x) \rangle, && (12)\\
&C(c) = \alpha_3(x) \wedge\\
&\quad O(c) = \alpha_3(x) \wedge m \rhd \{(c, _)\} = \langle\rangle \wedge I(c) = \alpha_0 \wedge n \rhd \{(c, _)\} = \langle\rangle && (13)
\end{aligned}$$

Suppose $(O, m, I, n)\ G\ C$, then the following lemmas hold

Lemma 2.2 . *dom* $T^{MUX2}(O, m, I, n) \subset$ *dom* $T^{CHANS}(C)$.

Proof: From the definition T^{MUX2}

$$\text{dom } T^{MUX2}(C) = \{\ \begin{aligned}[t] &\underline{c}.p.\_ \mid O(\underline{c}) = \alpha_0,\\ &\underline{c}.q.\underline{x} \mid I(\underline{c}) = \alpha_1(\underline{x}),\\ &\underline{c}.r.\_ \mid I(\underline{c}) = \alpha_2,\\ &\underline{c}.s.\underline{x} \mid O(\underline{c}) = \alpha_3(\underline{x})\ \}. \end{aligned}$$

From the definition of G,

$$\begin{aligned}
O(\underline{c}) = \alpha_0 \quad &\Rightarrow \quad C(\underline{c}) = \alpha_0\\
I(\underline{c}) = \alpha_1(\underline{x}) \quad &\Rightarrow \quad C(\underline{c}) = \alpha_1(\underline{x})\\
I(\underline{c}) = \alpha_2 \quad &\Rightarrow \quad C(\underline{c}) = \alpha_2\\
O(\underline{c}) = \alpha_3(\underline{x}) \quad &\Rightarrow \quad C(\underline{c}) = \alpha_3(\underline{x})
\end{aligned}$$

(each antecedent is only compatible with one of the eight clauses in definition 4.1—in order, clauses 1, 4, 5 and 8. The consequents are part of the respective clauses). From the definition T^{CHANS}

$$\text{dom } T^{CHANS}(C) = \{\ \begin{aligned}[t] &\underline{c}.p.\_ \mid C(\underline{c}) = \alpha_0,\\ &\underline{c}.q.\underline{x} \mid C(\underline{c}) = \alpha_1(\underline{x}),\\ &\underline{c}.r.\_ \mid C(\underline{c}) = \alpha_2,\\ &\underline{c}.s.\underline{x} \mid C(\underline{c}) = \alpha_3(\underline{x})\ \} \end{aligned}$$

and the result follows.

Lemma 2.3 $e \in$ *dom* $T^{MUX2}(O, m, I, n) \Rightarrow T^{MUX2}(O, m, I, n)(e)\ G\ T^{CHANS}(C)(e)$.

Proof: There are four possibilities:

Firstly, $e = c.p.x$ and $O(c) = \alpha_0$. Clause 1 in definition 4.1 obtains so $m \rhd \{(c, \_)\} = \langle\rangle$ and $I(c) = \alpha_0$ and $n \rhd \{(c, \_)\} = \langle\rangle$. The new state of the *MUX2* system, $(O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, m, I, n)$, is therefore related to the new state of the *CHANS* system, $C \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}$, by clause 2.

Secondly, $e = c.q.x$ and $I(c) = \alpha_1(x)$. Clause 4 in definition 4.1 obtains so $O(c) = \alpha_2$, $m \rhd \{(c, \_)\} = \langle\rangle$ and $n \rhd \{(c, \_)\} = \langle\rangle$. Hence the new state of the *MUX2* system,

$(O, m, I \oplus \{(\underline{c} \mapsto \alpha_2)\}, n)$, is related to the new state, $C \oplus \{(\underline{c} \mapsto \alpha_2)\}$, of the *CHANS* system by clause 5.

The third and fourth cases, $e = c.r.x$ and $e = c.s.x$ are handled in the same way.

Lemma 2.4 $(O^1, m^1, I^1, n^1) \in U^{MUX2}(O, m, I, n) \Rightarrow (O^1, m^1, I^1, n^1)\ G\ C$.

Proof: There are four ways for $(O^1, m^1, I^1, n^1) \in U^{MUX2}(O, m, I, n)$ to be satisfied.

The first possibility is that for some c, $O(c) = \alpha_1(x)$. This is only possible if clause 2 in 4.1 holds. Thus $C(c) = \alpha_1(x)$, $m \rhd \{(c, \_)\} = \langle\rangle$ and $I(c) = \alpha_0$ and $n \rhd \{(c, \_)\} = \langle\rangle$. Consequently $(O^1, m^1, I^1, n^1) = (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, m\hat{\ }\langle(\underline{c}, \underline{x})\rangle, I, n)$. is related to C by clause 3 in definition 4.1.

The second possibility is that $m \neq \langle\rangle \wedge m_0 = (c, x) \wedge I(c) = \alpha_0$. The only clause which allows $m \rhd \{(c, \_)\} \neq \langle\rangle$ is clause 3. So $C(c) = \alpha_1(x)$, $O(c) = \alpha_2$ and $m \rhd \{(c, \_)\} = \langle(c, x)\rangle$ (so $m' \rhd \{(c, \_)\} = \langle\rangle$, since $m_0 = (c, x)$) and $n \rhd \{(c, \_)\} = \langle\rangle$. Hence $(O^1, m^1, I^1, n^1) = (O, m', I \oplus \{(\underline{c} \mapsto \alpha_1(x))\}, n)$ satisfies clause 4, and this state is indeed related to C.

The other two cases in the definition of U^{MUX2} are treated similarly.

Lemma 2.5 $U^{MUX2}(O, m, I, n) = \emptyset \Rightarrow dom\ T^{MUX2}(O, m, I, n) = dom\ T^{CHANS}(C)$.

Proof: According to discussion in the proof of the previous lemma, we can assume that for no c do clauses 2, 3, 6 or 7 hold, otherwise $U^{MUX2}(O, m, I, n)$ would not be empty. So for each c one of clauses 1, 4, 5 or 8 obtains. The result then follows from the discussion in the proof of lemma 2.2.

These lemmas establish that G is a simulation. Hence the multiplexor system implements the channel system.

3 Models and Simulations: II. Acceptance-Transition Systems

In the models described in section 1, the only structures associated with a state were transition functions (labelled or unlabelled). In this section we add a set of "acceptances" for each state.

3.1 Acceptance-Transition Models

An ATM is a generalisation of a simple TM, characterised by a tuple (S, A, T, W). S, A and T are defined as for a TM. W associates set of "acceptances" with each state. An acceptance (in a state i) is some subset of dom $T(i)$. The acceptances $W(i)$ can be thought of as a set of "substates" of the state i. There is no external way of telling which substate is active—this is selected non-deterministically by the system whenever it enters a new state. *If* an acceptance $w \in W(i)$ is active, the system is prepared to engage in any of the transitions in $w \lhd T(i)$, but it will refuse any event outside w. From this discussion we see that W is in $S \rightarrow$ Pow Pow A. It is convenient to require that

$$\bigcup_{w \in W(i)} w = \text{dom } T(i).$$

ie, there are no redundant transitions in $T(i)$ which the system never offers to engage in.

ATMs are very closely related to the Acceptance Trees of [?], but we have no requirement that $W(i)$ be convex closed or closed under union. Those conditions are necessary if every process must have a unique representation (as in a semantic domain), but less important here, where an ATM is just one amongst many possible parametrisations of a process.

The system of CSP processes associated with an ATM is defined as the (least) solution of

$$Proc(i) = \underset{w \in W(i)}{\sqcap} (e : w \rightarrow Proc(T(i)(e))) = \underset{w \in W(i)}{\sqcap} R(w, T(i))(Proc)$$

Both ETMs and ATMs can express non-determinism. In fact, any divergence-free CSP process can be modelled both by an ATM and by an ETM—often, by many. The advantage of the ETM has already been illustrated—it emerges naturally when internal events are concealed. The advantage of an ATM is that one can show much more directly that it meets a particular CSP specification.

3.1.1 Specifications

Suppose $\mathcal{S}$ is the partial function from A^* to S which maps the possible traces of the model, generated from an inital state i, to the state of the model after engaging in that trace. The process associated with i satisfies

$$tr \in \text{dom } \mathcal{S} \wedge (\exists w : W(\mathcal{S}(tr)).\ w \cap ref = \emptyset)$$

3.1.2 Simulations

We give a simulation from ETMs to ATMs. This can be used to show that an ETM arising from event-concealment implements a system modelled by an ATM. This generalises the simulation given in **??** to the case where the processes involved aren't necessarily deterministic.

Definition 3.1 *Suppose (S, A, T, U) is an ETM and (S', A, T', W') is an ATM. The relation G from S to S' is a simulation if whenever $i\ G\ i'$ the following clauses hold*

$$dom\ T(i) \subset dom\ T'(i'), \tag{1}$$
$$\forall e : dom\ T(i).\ T(i)(e)\ G\ T'(i')(e), \tag{2}$$
$$\forall j : U(i).\ j\ G\ i', \tag{3}$$
$$U(i) = \emptyset \Rightarrow (\exists w' : W'(i').\ dom\ T(i) \supset w') \tag{4}$$

Theorem 3.2 *If G is a simulation in this sense, and $Proc$ is the system of processes associated with the ETM, and $Proc'$ is the set of process associated with the ATM, then $i\ G\ i'$ implies*

$$Proc(i) \sqsupseteq Proc'(i')$$

The proof is deferred to appendix A.

3.2 Extended Acceptance-Transition Models

Finally, suppose we are given a system modelled by an acceptance-transition model, and are required to conceal some events in its alphabet. The ATM can be generalised to an extendeded acceptance-transition model (EATM), in much the same way as the TM was generalised to the ETM, to take account of this.

An EATM is a tuple (S, A, T, U, V), where S, A, T and U have the same properties as for an ETM, and V associates a set of pairs with each state in S. Each pair consists of a set of unlabelled transitions and an acceptance set. A pair can be regarded as a sub-state of the visible state, in the same sense that an acceptance was a sub-state in the ETM. Hence $V \in \text{Pow}\ (\text{Pow}\ S \times \text{Pow}\ A)$. V is constrained by

$$\bigcup_{(u,w)\in V(i)} w = \text{dom}\ T(i), \quad \bigcup_{(u,w)\in V(i)} u = U(i).$$

The system of CSP processes associated with an EATM is defined as the (least) solution of

$$Proc(i) = \bigsqcap_{(u,w)\in W(i)} P(u, w, T(i))(Proc).$$

3.2.1 Concealment

If $Proc$ is the set of processes associated with an ATM (S, A, T, W), and

$$Proc'(i) = Proc(i) \setminus C$$

for some $C \subset A$, then (S, A, T', V') is an AETM for the system $Proc'$, where

$$\begin{aligned} T'(i) &= (A - C) \lhd T(i) \\ V'(i) &= \{(\text{ran}\ ((\underline{w} \cap C) \lhd T(i)), \underline{w} - C) \mid \underline{w} \in W(i)\} \end{aligned}$$

3.2.2 Simulations

We give a simulation from EATMs to ATMs. This can be used to show that an EATM arising from event-concealment implements a system modelled by an ATM.

Definition 3.3 *Suppose (S, A, T, U, V) is an EATM and (S', A, T', W') is an ATM. The relation G from S to S' is a simulation if whenever $i\ G\ i'$ the following clauses hold*

$$dom\ T(i) \subset dom\ T'(i'), \tag{1}$$
$$\forall e : dom\ T(i).\ T(i)(e)\ G\ T'(i')(e), \tag{2}$$
$$\forall j : U(i).\ j\ G\ i', \tag{3}$$
$$\forall (u, w) : V(i).\ u = \emptyset \Rightarrow (\exists w' : W'(i').\ w \supset w') \tag{4}$$

Theorem 3.4 *If G is a simulation in this sense, and $Proc$ is the system of processes associated with the EATM, and $Proc'$ is the set of process associated with the ATM, then $i\ G\ i'$ implies*

$$Proc(i) \sqsupseteq Proc'(i')$$

The proof is deferred to appendix A.

4 Models for message-routers.

A message delivery system is regarded as a process which accepts messages from a system of user processes, and delivers messages to the user system. A few properties are desirable *a priori*. For example, the system shouldn't deliver messages which it hasn't previously accepted—it shouldn't generate spurious messages. Similarly, I'll assume that it doesn't duplicate messages—for each message accepted, at most one is delivered. Needless to say, many useful message delivery systems don't guarantee either of these properties.

However they do seem to be reasonable requirements in the context of tightly-coupled MIMD systems. It isn't difficult for hardware-assisted routing systems to implement these specifications.

Ignoring, for the moment only, liveness properties of the message delivery system, the simplest structure which meets these requirements is the *bag* [1]. Assume the set of message types is *Values*. The alphabet of the bag is

$$\alpha Bag = \{\underline{v}^O \mid \underline{v} \in Values, \quad \underline{v}^I \mid \underline{v} \in Values\}.$$

The event v^O corresponds to output of a message holding value v from the user to the delivery system. v^I corresponds to it's input (delivery of the message). According to the discussion in the previous paragraph any trace tr of Bag should satisfy

$$\forall v : Values. \quad tr \downarrow \{v^O\} \geq tr \downarrow \{v^I\}$$

In fact, so far as traces are concerned, this is the strongest specification needed to yield a workable message delivery system. One can also give a state transition model of this trace set. The state space is

$$Bstat = Values \rightarrow Nat$$

A state B in this space associates a non-negative value with each message type—$B(v)$ is the number of messages of type v presently in transit. The transition function is

$$\begin{aligned} T^{Bstat}(B) \quad &= \quad \{(\underline{v}^O \mapsto B \oplus (\underline{v} \mapsto B(\underline{v}) + 1)), \\ &\qquad (\underline{v}^I \mapsto B \oplus (\underline{v} \mapsto B(\underline{v}) - 1)) \mid B(\underline{v}) \neq 0\}. \end{aligned}$$

The important property of this transition function is that an event v^I can only occur when there are some messages of type v in transit. The required trace set is generated from the initial state $B = 0$. To show that this system meets the specification above, one can use the result in section 1.1.1 with

$$\begin{aligned} \mathrm{dom}\, \mathcal{S} \quad &= \quad \{\underline{s} \mid \forall s.\ s \leq \underline{s} \Rightarrow (\forall v.B_s(v) \geq 0)\} \\ \mathcal{S}(s) \quad &= \quad B_s,\ \text{if } s \in \mathrm{dom}\, \mathcal{S} \end{aligned}$$

where $B_s(v) = s \downarrow \{v^O\} - s \downarrow \{v^I\}$.

This specification of the system's traces embodies the "safety" properties demanded in the opening paragraph of this section. It says nothing about "liveness". The ideal router would be a deterministic process with the specified traces. This system would never refuse any event offered by the enviroment so long as the event was engagable according to the trace specification. The only events it ever refused would be those inputs corresponding to messages which hadn't yet been sent. It would always accept any output, and at any time it would be prepared to deliver any message it currently held (at least, it would deliver any such message in bounded time, without conditions on whether the user was refusing to input some other set of messages also in transit). No real message delivery system has these properties because they depend on the availability of infinite buffering. Infinite buffering can be approximated, up to a point, by, say, having *large* buffers permanently allocated, which is costly in memory, or by dynamically allocating memory to buffers as required, which is costly in processor time. This rules out such schemes out for the purposes of this article.

What is the minimal set of liveness properties for a useful delivery system? Certainly, if the delivery system is empty (no message in transit), we would expect it to be prepared to accept any message. It might sometimes refuse to accept outputs if there *are* messages in transit, because some buffers may have filled up. Secondly, if there *are* messages in transit, we would expect it to be willing to deliver at least one of them [2]. However, we will accept that delivery of some messages in transit may be deferred pending delivery of some other messages, also in transit, due to some internal forwarding buffers being occupied.

These minimal requirements are embodied in the (failures model) specification

$$\begin{array}{lll} tr \in \mathit{traces}(Bag) & \wedge & (\textbf{if}\ (\forall v : \mathit{Values}.\ \ tr \downarrow \{v^O\} = tr \downarrow \{v^I\})\ \textbf{then} \\ & & \quad (\forall v : \mathit{Values}.\ \ v^O \notin \mathit{ref})\ \textbf{else} \\ & & \quad (\exists v : \mathit{Values}.\ \ tr \downarrow \{v^O\} > tr \downarrow \{v^I\}\ \wedge\ v^I \notin \mathit{ref})) \end{array}$$

This asserts that the traces of the system are those described above, and that if there are no messages in transit, no refusal state refuses any output event; otherwise, in any refusal state there is (at least) one input event corresponding to a message in transit which won't be refused.

An acceptance transition model defining a process which satisfies this specification is $(\alpha Bag, Bstat, T^{Bstat}, W^{Bstat})$ where

$$W^{Bstat}(B) = \{\{\underline{v}^I\} \mid B(\underline{v}) > 0,\ \{\underline{\underline{v}}^I \mid B(\underline{\underline{v}}) > 0,\ \underline{\underline{v}}^O\}\}$$

(ie, $W^{Bstat}(B)$ contains singleton sets corresponding to each input event for which $B(v) > 0$, and a set containing all input events satisfying the same condition together with all output events). This can be proven immediately using the result in section ??.

4.1 "Virtual Channels" Implemented on a Message-Router

Suppose

$$\mathit{Message} = \{\underline{c}.f.\underline{x} \mid \underline{c} \in \mathit{Chans} \wedge \underline{x} \in \mathit{Values},\ \underline{c}.b.\underline{x} \mid \underline{c} \in \mathit{Chans} \wedge \underline{x} \in \mathit{Values}\}.$$

Consider the following putative implementation of a virtual channel system:

$$\left(\left(\mathop{\|}_{c \in \mathit{Chans}} c : \mathit{OBUF} \right) \| \atop \left(\mathop{\|}_{c \in \mathit{Chans}} c : \mathit{IBUF} \right) \| \mathit{BAG} \right) \setminus H$$

The system of input and output buffers is modelled by the TM (A^{IO}, S^{IO}, T^{IO}), where

$$\begin{array}{lcl} A^{IO} & = & \{\underline{c}.p.\underline{x},\ \underline{c}.f.\underline{x}^O,\ \underline{c}.f.\underline{x}^I,\ \underline{c}.q.\underline{x},\ \underline{c}.r.\underline{x},\ \underline{c}.b.\underline{x}^O,\ \underline{c}.b.\underline{x}^I,\ \underline{c}.s.\underline{x}\} \\ S^{IO} & = & (\mathit{Chans} \rightarrow S^O) \times (\mathit{Chans} \rightarrow S^I) \end{array}$$

and

$$\begin{array}{ll} \multicolumn{2}{l}{T^{IO}(O, I) =} \\ \{\ (\underline{c}.p.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, I)) & \mid O(\underline{c}) = \alpha_0, \\ \ \ (\underline{c}.f.\underline{x}^O \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, I)) & \mid O(\underline{c}) = \alpha_1(\underline{x}), \\ \ \ (\underline{c}.f.\underline{x}^I \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\})) & \mid I(\underline{c}) = \alpha_0, \\ \ \ (\underline{c}.q.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_2)\})) & \mid I(\underline{c}) = \alpha_1(\underline{x}), \\ \ \ (\underline{c}.r.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\})) & \mid I(\underline{c}) = \alpha_2, \\ \ \ (\underline{c}.b.\underline{x}^O \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_0)\})) & \mid I(\underline{c}) = \alpha_3(\underline{x}), \\ \ \ (\underline{c}.b.\underline{x}^I \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, I)) & \mid O(\underline{c}) = \alpha_2, \\ \ \ (\underline{c}.s.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_0)\}, I)) & \mid O(\underline{c}) = \alpha_3(\underline{x})\ \} \end{array}$$

[2] This property may not be satisfied if the delivery system is prone to internal deadlock.

The initial state is $(\{(\_ \mapsto \alpha_0)\}, \{(\_ \mapsto \alpha_0)\})$. Hence

$$
\begin{array}{l}
\mathrm{dom}\ T^{IO}(O,I) = \\
\quad \{\ \begin{array}{ll}
\underline{c}.p.\underline{x} & |\, O(\underline{c}) = \alpha_0, \\
\underline{c}.f.\underline{x}^O & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\underline{c}.f.\underline{x}^I & |\, I(\underline{c}) = \alpha_0, \\
\underline{c}.q.\underline{x} & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\underline{c}.r.\underline{x} & |\, I(\underline{c}) = \alpha_2, \\
\underline{c}.b.\underline{x}^O & |\, I(\underline{c}) = \alpha_3(\underline{x}), \\
\underline{c}.b.\underline{x}^I & |\, O(\underline{c}) = \alpha_2, \\
\underline{c}.s.\underline{x} & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \}.
\end{array}
\end{array}
$$

Now we need to calculate $\mathrm{dom}\ T^{IO}(O,I)\ [A^{IO} \sqcap A^{BAG}]\ w$ for each w in the set

$$
\begin{array}{l}
W^{BAG}(B) = \\
\quad \{\ \{\underline{c}.f.\underline{x}^I\} \,|\, B(\underline{c}.f.\underline{x}) > 0,\ \{\underline{c}.b.\underline{x}^I\} \,|\, B(\underline{c}.b.\underline{x}) > 0,\ \mathrm{dom}\ T^{BAG}(B)\ \}.
\end{array}
$$

where

$$
\begin{array}{l}
\mathrm{dom}\ T^{BAG}(B) = \\
\quad \{\ \underline{c}.f.\underline{x}^I \,|\, B(\underline{c}.f.\underline{x}) > 0,\ \underline{c}.b.\underline{x}^I \,|\, B(\underline{c}.b.\underline{x}) > 0,\ \underline{c}.f.\underline{x}^O,\ \underline{c}.b.\underline{x}^O\ \}.
\end{array}
$$

The result is

$$
\begin{array}{l}
W^{ROUT}(O,I,B) = \\
\quad \{\ \begin{array}{ll}
w^{EXT}(O,I) \cup \{\underline{c}.f.\underline{x}^I\} & |\, B(\underline{c}.f.\underline{x}) > 0 \wedge I(\underline{c}) = \alpha_0, \\
w^{EXT}(O,I) & |\, B(\underline{c}.f.\underline{x}) > 0 \wedge I(\underline{c}) \neq \alpha_0, \\
w^{EXT}(O,I) \cup \{\underline{c}.b.\underline{x}^I\} & |\, B(\underline{c}.b.\underline{x}) > 0 \wedge O(\underline{c}) = \alpha_2, \\
w^{EXT}(O,I) & |\, B(\underline{c}.b.\underline{x}) > 0 \wedge O(\underline{c}) \neq \alpha_2, \\
\mathrm{dom}\ T^{ROUT}(O,I,B)\ \}, &
\end{array}
\end{array}
$$

where

$$
\begin{array}{l}
w^{EXT}(O,I) = \\
\quad \{\ \begin{array}{ll}
\underline{c}.p.\underline{x} & |\, O(\underline{c}) = \alpha_0, \\
\underline{c}.q.\underline{x} & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\underline{c}.r.\underline{x} & |\, I(\underline{c}) = \alpha_2, \\
\underline{c}.s.\underline{x} & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \}
\end{array}
\end{array}
$$

and

$$
\begin{array}{l}
\mathrm{dom}\ T^{ROUT}(O,I,B) = \\
\quad w^{EXT}(O,I) \cup \{\ \begin{array}{ll}
\underline{c}.f.\underline{x}^I & |\, B(\underline{c}.f.\underline{x}) > 0 \wedge I(\underline{c}) = \alpha_0, \\
\underline{c}.b.\underline{x}^I & |\, B(\underline{c}.b.\underline{x}) > 0 \wedge O(\underline{c}) = \alpha_2, \\
\underline{c}.f.\underline{x}^O & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\underline{c}.b.\underline{x}^O & |\, I(\underline{c}) = \alpha_3(\underline{x})\ \}.
\end{array}
\end{array}
$$

Completing the definition of T^{ROUT}:

$T^{ROUT}(O,I,B) =$

$$
\begin{array}{ll}
\{\ (\underline{c}.p.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, I, B)) & |\, O(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.f.\underline{x}^O \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, I, B \oplus \{(\underline{c}.f.\underline{x} \mapsto B(\underline{c}.f.\underline{x}) + 1)\})) & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.f.\underline{x}^I \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, B \oplus \{(\underline{c}.f.\underline{x} \mapsto B(\underline{c}.f.\underline{x}) - 1)\})) & |\, I(\underline{c}) = \alpha_0 \wedge B(\underline{c}.f.\underline{x}) \neq 0, \\
\ \ (\underline{c}.q.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_2)\}), B) & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.r.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}), B) & |\, I(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.b.\underline{x}^O \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_0)\}, B \oplus \{(\underline{c}.b.\underline{x} \mapsto B(\underline{c}.b.\underline{x}) + 1)\})) & |\, I(\underline{c}) = \alpha_3(\underline{x}), \\
\ \ (\underline{c}.b.\underline{x}^I \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, I, B \oplus \{(\underline{c}.b.\underline{x} \mapsto B(\underline{c}.b.\underline{x}) - 1)\})) & |\, O(\underline{c}) = \alpha_2 \wedge B(\underline{c}.b.\underline{x}) \neq 0, \\
\ \ (\underline{c}.s.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_0)\}, I, B)) & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \},
\end{array}
$$

yields the ATM $(A^{IO}, S^{IO} \times S^{BAG}, T^{ROUT}, W^{ROUT})$ describing the combined system.

Next, we conceal the internal events to yield an EATM.

$$
\begin{array}{l}
T^{ROUT2}(O, I, B) = \\
\quad \begin{array}{ll}
\{\ (\underline{c}.p.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, I, B)) & |\, O(\underline{c}) = \alpha_0, \\
\ \ (\underline{c}.q.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_2)\}), B) & |\, I(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (\underline{c}.r.\underline{x} \mapsto (O, I \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}), B) & |\, I(\underline{c}) = \alpha_2, \\
\ \ (\underline{c}.s.\underline{x} \mapsto (O \oplus \{(\underline{c} \mapsto \alpha_0)\}, I, B)) & |\, O(\underline{c}) = \alpha_3(\underline{x})\ \},
\end{array}
\end{array}
$$

$$
\begin{array}{l}
U^{ROUT2}(O, I, B) = \\
\quad \begin{array}{ll}
\{\ (O \oplus \{(\underline{c} \mapsto \alpha_2)\}, I, B \oplus \{(\underline{c}.f.\underline{x} \mapsto B(\underline{c}.f.\underline{x}) + 1)\}) & |\, O(\underline{c}) = \alpha_1(\underline{x}), \\
\ \ (O, I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, B \oplus \{(\underline{c}.f.\underline{x} \mapsto B(\underline{c}.f.\underline{x}) - 1)\}) & |\, I(\underline{c}) = \alpha_0 \wedge B(\underline{c}.f.\underline{x}) \neq 0, \\
\ \ (O, I \oplus \{(\underline{c} \mapsto \alpha_0)\}, B \oplus \{(\underline{c}.b.\underline{x} \mapsto B(\underline{c}.b.\underline{x}) + 1)\}) & |\, I(\underline{c}) = \alpha_3(\underline{x}), \\
\ \ (O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, I, B \oplus \{(\underline{c}.b.\underline{x} \mapsto B(\underline{c}.b.\underline{x}) - 1)\}) & |\, O(\underline{c}) = \alpha_2 \wedge B(\underline{c}.b.\underline{x}) \neq 0\ \},
\end{array}
\end{array}
$$

$$
\begin{array}{l}
V^{ROUT2}(O, I, B) = \\
\quad \begin{array}{ll}
\{\ (w^{EXT}(O, I), \{(O, I \oplus \{(\underline{c} \mapsto \alpha_1(\underline{x}))\}, B \oplus \{(\underline{c}.f.\underline{x} \mapsto B(\underline{c}.f.\underline{x}) - 1)\})\}) & |\, I(\underline{c}) = \alpha_0 \wedge B(\underline{c}.f.\underline{x}) \neq 0, \\
\ \ (w^{EXT}(O, I), \emptyset) & |\, I(\underline{c}) \neq \alpha_0 \wedge B(\underline{c}.f.\underline{x}) \neq 0, \\
\ \ (w^{EXT}(O, I), \{(O \oplus \{(\underline{c} \mapsto \alpha_3(\underline{x}))\}, I, B \oplus \{(\underline{c}.b.\underline{x} \mapsto B(\underline{c}.b.\underline{x}) - 1)\})\}) & |\, O(\underline{c}) = \alpha_2 \wedge B(\underline{c}.b.\underline{x}) \neq 0, \\
\ \ (w^{EXT}(O, I), \emptyset) & |\, O(\underline{c}) \neq \alpha_2 \wedge B(\underline{c}.b.\underline{x}) \neq 0, \\
\ \ (w^{EXT}(O, I), U^{ROUT2}(O, I, B))\ \}. &
\end{array}
\end{array}
$$

Finally, we have to show that the EATM $(A^{CHANS}, S^{ROUT}, T^{ROUT2}, U^{ROUT2}), V^{ROUT2})$ simulates the TM $(A^{CHANS}, S^{CHANS}, T^{CHANS})$. We define the relation G from S^{ROUT} to S^{CHANS} as follows:

Definition 4.1 *The relationship $(O, m, I, n)\ G\ C$ holds iff for any c one of the following (mutually exclusive) clauses obtains for some x*

$$
\begin{array}{l}
C(c) = \alpha_0 \wedge \\
\qquad O(c) = \alpha_0 \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_0 \wedge (\forall y.\ B(c.b.y) = 0),
\end{array} \tag{5}
$$

$$
\begin{array}{l}
C(c) = \alpha_1(x) \wedge \\
\qquad O(c) = \alpha_1(x) \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_0 \wedge (\forall y.\ B(c.b.y) = 0),
\end{array} \tag{6}
$$

$$
\begin{array}{l}
C(c) = \alpha_1(x) \wedge \\
\qquad O(c) = \alpha_2 \wedge (\forall y.\ B(c.f.y) = \delta_{xy}) \wedge I(c) = \alpha_0 \wedge (\forall y.\ B(c.b.y) = 0),
\end{array} \tag{7}
$$

$$
\begin{array}{l}
C(c) = \alpha_1(x) \wedge \\
\qquad O(c) = \alpha_2 \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_1(x) \wedge (\forall y.\ B(c.b.y) = 0),
\end{array} \tag{8}
$$

$$
\begin{array}{l}
C(c) = \alpha_2 \wedge \\
\qquad O(c) = \alpha_2 \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_2 \wedge (\forall y.\ B(c.b.y) = 0),
\end{array} \tag{9}
$$

$$C(c) = \alpha_3(x) \wedge$$
$$O(c) = \alpha_2 \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_3(x) \wedge (\forall y.\ B(c.b.y) = 0), \tag{10}$$
$$C(c) = \alpha_3(x) \wedge$$
$$O(c) = \alpha_2 \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_0 \wedge (\forall y.\ B(c.b.y) = \delta_{xy}), \tag{11}$$
$$C(c) = \alpha_3(x) \wedge$$
$$O(c) = \alpha_3(x) \wedge (\forall y.\ B(c.f.y) = 0) \wedge I(c) = \alpha_0 \wedge (\forall y.\ B(c.b.y) = 0) \tag{12}$$

The first three properties needed to establish that G is a simulation are proved in exactly the same way as in lemmas ?? to ??. The last lemma needed is, assuming $(O, m, I, n)\ G\ C$

Lemma 4.2 $((w, u) \in V^{MUX2}(O, m, I, n) \wedge u = \emptyset) \Rightarrow (w \supset dom\ T^{CHANS}(C))$.

Proof: A priori, there are five cases to consider:

1. $I(c) = \alpha_0 \wedge B(c.f.x) \neq 0$ and $u = \{(O, I \oplus \{(c \mapsto \alpha_1(x))\}, B \oplus \{(c.f.x \mapsto B(c.f.x) - 1)\}))\}$.
2. $I(c) \neq \alpha_0 \wedge B(c.f.x) \neq 0$ and $u = \emptyset$.
3. $O(c) = \alpha_2 \wedge B(c.b.x) \neq 0$ and $u = \{(O \oplus \{(c \mapsto \alpha_3(x))\}, I, \oplus\{(c.b.x \mapsto B(c.b.x) - 1)\})\}$.
4. $O(\underline{c}) \neq \alpha_2 \wedge B(\underline{c}.b.\underline{x}) \neq 0$ and $u = \emptyset$.
5. $u = U^{ROUT2}(O, I, B)$.

In case 1, $u = \emptyset$ is false, so the implication is valid.

The precondition of case 2 contradicts the assumption that (O, I, B) is in the range of G (none of the clauses in the definition is consistent with this condition). So this case doesn't need further consideration.

Case 3 is like case 1.

Case 4 is like case 2.

In case 5 the implication is established as for lemma ??.

5 Discussion

5.1 Working with deterministic models

The example given in the previous section was sufficiently simple that we could afford to simply combine the non-deterministic delivery system with the IO buffer system in one step, then immediately conceal internal events in the composite model and verify a simulation from the result to a model of the specification.

In general it may be more convenient to go through intermediate simulations ("refinements") between the composition step and the concealment ("abstraction") step. Moreover, in some of the steps it may be possible to ignore the non-determinism in one or more of the original components. Note

$$P = P \parallel P^{det}$$

where P^{det} is the deterministic process with the same traces as P. In terms of our transition models, it can be represented by the TM obtained from any generalised transition model standing for P by ignoring any U, V or W fields. Evidently, life is simpler if one can work with this system rather than the original non-deterministic one.

Suppose P is being composed with some system Q. According to the observation above, if we can show

$$P^{det} \parallel Q \sqsupseteq R$$

then it follows by monotonicity of $\parallel$ that

$$P \parallel Q \sqsupseteq P \parallel R$$

The value of this theorem is that R may be more tractable than Q, so that composing it with the full P is simpler than direct use of R. On the other hand, the establishing the precondition is simplified because P^{det} is a deterministic process. For example, in the virtual channel example of section ??, composing the deterministic part of the delivery system with the IO buffer system yields the TM (A^{IO}, T^{ROUT}). Then, by a straightforward TM to TM simulation one can show equivalence of this system to the more concise deterministic system

$$\underset{c \in Chans}{\parallel} c : IOBUF$$

where

$$IOBUF = \mu X.\ \Box_x p.x \to f^O.x \to \Box_y f^I.y \to q.y \,\Box_z r.z \to b^O.z \,\Box_w b^I.w \to s.w \to X$$

Now this system is combined with the full non-deterministic model for the router, and the rest of the calculation can proceed essentially as before. In this case the simplification obtained in this way is not enormous (which is why this logical step was skipped in the earlier presentation). In general, there may be significant advantages to be gained by organising the proof in this way [3].

5.2 Eagerness

Low-level message delivery systems for multi-processors often come with a guarantee that they are "deadlock-free, provided the user processes are eager readers". The intuitive idea of a process being an "eager reader" is that it never refuses to accept any message that has been sent to it (or, at least, it only defers this acceptance for a bounded time, while it may be engaging in some internal activity).

Intuitively, eager readership seems to be an important concept. In this section we will try to give a more precise characterisation of eagerness. In this analysis, the slightly mysterious promise about deadlock freedom quoted in the previous paragraph could be replaced by: "the message delivery system is *effectively deterministic* provided the system of user processes is an eager reader" [4].

So, suppose $pending(tr)$ is a function on traces which returns the set of message-input events the delivery system *may* engage in (according to its traces) after the system as a whole has executed a trace tr. In the bag model this is just

$$pending(tr) = \{\underline{m}^I \mid tr \downarrow \{\underline{m}^O\} > tr \downarrow \{\underline{m}^I\}\}$$

[3] In passing, note that the clarity of the proof in section ?? would also have benefitted from inserting a further intermediate simulation between composition with the non-deterministic router and concealment of events—eliminating many redundant states before rather than after concealment

[4] The replacement, here, of individual "user processes" by the system as a whole recognises the fact that eagerness must be a global property—there is no implication that an individual process should always be waiting to accept notional messages which haven't actually been sent to it (by one of the other user processes).

Now, if P is the message delivery system, and Q is the system of user processes, Q is an eager-reading system if it satisfies

$$\forall s : traces(P \parallel Q).\ \forall X : refusals(Q/s \rhd \alpha Q).\ pending(s) \cap X = \emptyset$$

This says that Q never refuses any pending messages (***remark on appearance of P —— Q here***). Now one can easily proof the following quite striking theorem

Theorem 5.1 *If P' is any process with the same traces as P which satisfies the liveness property in section ??, then*

$$(P' \parallel Q) \setminus \{\underline{m}^I\} = (P \parallel Q) \setminus \{\underline{m}^I\}$$

In particular one can take $P' = P^{det}$. So if Q is an eager reader we can replace the message delivery system by its deterministic counterpart, and after the abstraction step of hiding message inputs, we end up with the same process as if we had used the full non-deterministic definition of P.

This appears to be a powerful result. It seems obvious that the system of independent *IOBUF* processes is an eager-reading system, so we can just hide inputs in that model without ever worrying about acceptances of the message delivery system, and get the proof that the system meets it's specification that much more easily.

In general, unfortunately, proving that a system has the eager readership property may itself be non-trivial, and the simplification implied here may be slightly illusory. Still, this is a proof technique which deserves investigation.

Incidentally, we have only given the simplest possible definition of eagerness here, in the context of a bag-like message router. Needless to say, the idea can be generalised in several directions, and similar theorems proved.

5.3 More structured models of message-delivery systems

The "bag" model of a message delivery system is useful for illustrative purposes, but it doesn't capture much of the structure of a typical message delivery systems. There is no inherent idea of locality—messages are inserted into the system and removed from it without any association with a point (in "processor space") from which they are originating or to which they are being delivered. Also, many delivery systems provide guarantees on sequencing of messages which are passed "between the same points"—messages emerge from the delivery system in order which they inserted.

These shortcomings can be addressed by adding "ports" to the delivery system. Logical channels exist between any pair of ports, and message delivery across any particular channel is sequence-preserving. Each port may be associated with a particular processor (there may be one or many ports per processor)

The alphabet of the new system is

$$\alpha Msys = \{\underline{M}^O.\underline{v} \mid \underline{M} \in Ports^2 \wedge \underline{v} \in Messages, \quad \underline{M}^I.\underline{v} \mid \underline{M} \in Ports^2 \wedge \underline{v} \in Messages,$$

where *Ports* is the space of ports. The traces associated with the system now satisfy

$$\forall (P, Q) : Ports^2.\ tr \downarrow \{(P, Q)^O\} > tr \downarrow \{(P, Q)^I\}$$

In fact, so far as traces are concerned, this is the strongest specification needed to yield a workable message delivery system. One can also give a state transition model of this trace set. The state space is

$$Mstat = Ports^2 \rightarrow Values^*$$

A state B in this space associates a non-negative value with each message type—$B(v)$ is the number of messages of type v presently in transit. The transition function is

$$\begin{aligned} T^{Msys}(M) &= \{((\underline{P},\underline{Q})^O.\underline{v} \mapsto M \oplus ((\underline{P},\underline{Q}) \mapsto M(\underline{P},\underline{Q})\hat{}\langle \underline{v}\rangle)), \\ &\qquad ((\underline{P},\underline{Q})^I.\underline{v} \mapsto M \oplus ((\underline{P},\underline{Q}) \mapsto M(\underline{P},\underline{Q})'))M(\underline{P},\underline{Q})') \neq \langle\rangle \wedge \underline{v} = M(\underline{P},\underline{Q})_0\}. \end{aligned}$$

A Proof of simulations

All our simulation results can be regarded as special cases of the result in section 3.2.2. That result is proved here. First we need an auxilliary lemma.

Lemma A.1 *Suppose W and W' are both sets of subsets of A (ie, elements of Pow Pow A) where the set A indexes a set of processes $\{P_{\underline{e}} \mid \underline{e} \in A\}$. If the union of the elements of W contains the union of the elements of W'*

$$(\bigcup_{w' \in W'} w') \subset (\bigcup_{w \in W} w),$$

and any w' in W' contains some w from W, ie [5]

$$\forall w' : W'.\ \exists w : W.\ w' \supset w$$

then

$$(\underset{w' \in W'}{\sqcap} \underset{e' \in w'}{\Box} P_e) \sqsupseteq (\underset{w \in W}{\sqcap} \underset{e \in w}{\Box} P_e)$$

Proof: First we show

$$a \Box b \sqsupseteq a \sqcap (a \Box b \Box c) \tag{1}$$

This follows because

$$\begin{aligned} a &\sqsupseteq a \sqcap (a \Box c) \\ b &\sqsupseteq a \sqcap b \end{aligned}$$

both of which are due to $A \sqsupseteq A \sqcap B$. Then monotonicity of $\Box$ gives

$$a \Box b \sqsupseteq (a \sqcap (a \Box c)) \Box (a \sqcap b) = a \sqcap ((a \Box c) \Box b)$$

where the equality depends on distribution of $\sqcap$ over $\Box$. This yields 1. Now, introduce w_1 and $\mathcal{W}$ satisfying $w_1 \subset w' \subset \mathcal{W}$, and identify

$$\begin{aligned} a &= \underset{e \in w_1}{\Box} P_e \\ b &= \underset{e \in w' - w_1}{\Box} P_e \\ c &= \underset{e \in \mathcal{W} - w'}{\Box} P_e. \end{aligned}$$

1 becomes

$$\underset{e \in w'}{\Box} P_e \sqsupseteq (\underset{e \in w_1}{\Box} P_e) \sqcap (\underset{e \in \mathcal{W}}{\Box} P_e). \tag{2}$$

[5] *This condition is sometimes [?] written as $W' \subset\subset W$*

If $w_1 \in W$,

$$\Box_{e \in w_1} P_e \sqsupseteq (\sqcap_{w \in W} \Box_{e \in w} P_e)$$

because $A \sqsupseteq A \sqcap B$. If, also, $\mathcal{W} = \bigcup_{w \in W} w$,

$$\Box_{e \in \mathcal{W}} P_e = (\Box_{w \in W} \Box_{e \in w} P_e) \sqsupseteq (\sqcap_{w \in W} \Box_{e \in w} P_e)$$

because $A \Box B \sqsupseteq A \sqcap B$. Subsituting these two results in 2, using monotonicity and idempotence of $\sqcap$, gives

$$\Box_{e \in w'} P_e \sqsupseteq (\sqcap_{w \in W} \Box_{e \in w} P_e).$$

This inequality depended only on the assumptions that there existed $w_1 \in W$ such that $w_1 \subset w'$, and that $w' \subset \bigcup_{w \in W} w$. Monotoniciy and idempotence of $\sqcap$ now establish the lemma.

Now, suppose G is a simulation in the sense of definition 3.3 from the EATM (S, A, T, U, V) to the ATM (S', A, T', W').

Lemma A.2 *There exists a function $I^U : S \rightarrow Nat$ such that*

$$\forall j : U(i).\ I^U(j) < I^U(i)$$

Proof: I^U can be constructed as

$$I^U(i) = \min\{n \mid \tilde{U}^n\{i\} = \emptyset\}$$

Well-definedness of this function is guaranteed by the nilpotence of U. The required property (which will be used in a proof by complete induction) follows easily.

Definition A.3 *For any i' in ran G define*

$$X'(i') = \sqcap_{j\ G\ i'} Proc(j)$$

Lemma A.4 *If $i\ G\ i'$,*

$$Proc(i) \sqsupseteq X'(i').$$

Proof: From the law $A \sqsupseteq A \sqcap B$.

Corollary A.5 *If $i\ G\ i'$,*

$$R(w, T(i))(Proc) \sqsupseteq R(w, T'(i'))(X')$$

for any $w \subset dom\ T(i)$.

Proof: By clauses 1 and 2 in 3.3, $T(i)(e)\ G\ T'(i')(e)$ for and e in dom $T(i)$. So, by the lemma

$$Proc(T(i)(e)) \sqsupseteq X'(T'(i')(e))$$

The definition of R, and monotonicity of the choice operation, then imply the corollary.

We can now derive the principle lemma needed to prove theorem 3.4:

Lemma A.6 *If $i\ G\ i'$ then*

$$Proc(i) \sqsupseteq \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X')$$

Proof: We use complete induction. The induction hypothesis is that the lemma is satisfied for all states j in dom G with $I^U(j) < n$, and we must show that it holds for any state i with $I^U(i) = n$. First we show

$$P(u, w, T(i))(Proc) \sqsupseteq \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X') \tag{1}$$

for any $(u, w) \in V(i)$. There are two cases to consider: $u = \emptyset$ and $u \neq \emptyset$.

Case I: P reduces to R, and

$$\begin{aligned} R(w, T(i))(Proc) \sqsupseteq\ & R(w, T'(i'))(X') \sqsupseteq \\ & \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X') \end{aligned}$$

where the first inequality follows from lemma A.5 and the second inequality uses final clause in the definition of G, together with lemma A.1, choosing $W = \{w\}$.

Case II: P contains two terms. First

$$\begin{aligned} Q(u)(Proc) =\ & \bigsqcap_{j \in u} Proc(j) \sqsupseteq \\ & \bigsqcap_{j \in u} \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X') = \\ & \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X') \end{aligned}$$

The first inequality follows because, according to the third clause in the definition of G, for each term in the left-hand side $j\ G\ i'$. So, by the properties of I^U, the induction hypothesis is applicable to the individual terms—the inequality then follows from monotonicity of $\sqcap$. The subsequent equality is due to idempotence of $\sqcap$. The second term in P is

$$\begin{aligned} & Q(u)(Proc) \,\square\, R(w, T(i))(Proc) \sqsupseteq \\ & (\bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X')) \,\square\, R(w, T'(i'))(X') = \\ & \bigsqcap_{w' \in W'(i')} R(w' \cup w, T'(i'))(X') \sqsupseteq \\ & \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X') \end{aligned}$$

The first inequality depends on monotonicity of $\square$, using the result above and corollary A.5. The middle equality follows from the definition of R, using the fact that $\square$ distributes over $\sqcap$. The final inequality is an application of A.1 with $W = \{\underline{w'} \cup w \,|\, \underline{w'} \in W'(i')\}$. In P, these two terms are combined non-deterministically. Due to the monotonicity and idempotence of $\sqcap$, P satisfies the same inequality as its two terms. This establishes 1, and the induction step follows from the equation satisfied by $Proc(i)$, namely

$$Proc(i) = \bigsqcap_{(u,w) \in W(i)} P(u, w, T(i))(Proc),$$

using monotonicity and idempotence of $\sqcap$.

Now, from the definition A.3, lemma A.6, and the monotonicity and idempotence of $\sqcap$,

$$X'(i') \sqsupseteq \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(X')$$

On the other hand, by definition, the system $Proc'$ is the least solution of

$$Proc'(i') = \bigsqcap_{w' \in W'(i')} R(w', T'(i'))(Proc')$$

Since least fixpoints are also least pre-fixpoints[6], we can conclude that

$$X'(i') \sqsupseteq Proc'(i')$$

Theorem 3.4 follows from lemma A.4.

References

[1] C.A.R. Hoare and He Jifeng, "Design and Proof of a Mail Service", in "Developments in Concurrency and Communication" edited by C.A.R. Hoare, Addison Wesley (1990).

[6] A pre-fixpoint of F satisfies $X \sqsupset FX$