**Programming Research Group**

# A STEPWISE DEVELOPMENT OF THE ALTERNATING BIT PROTOCOL


Axel Wabenhorst



PRG-TR-12-97



Oxford University Computing Laboratory

Wolfson Building, Parks Road, Oxford OX1 3QD

# A Stepwise Development of the Alternating Bit Protocol

Axel Wabenhorst*

March 1997

**Abstract**

In this paper, the theory of fair action systems is used in the specification and development of the Alternating Bit Protocol. A version of the Movers Theorem for fair action systems is proved, which permits the interleaving of actions of one action system within another. This Movers Theorem is applied repeatedly in the development of the Alternating Bit Protocol.

## 1 Introduction

Our aim is to illustrate the usefulness of the theory of fair action systems presented in [Wab96a] through the development of the Alternating Bit Protocol. Particularly by addressing *fairness* in our approach to the verification of programs, will be able to develop, entirely within the formalism, designs whose infinitary behaviour previously required *ad hoc* reasoning.

The Movers Theorem provides support for introducing concurrency into a development by identifying conditions under which the actions of an iteration executed atomically as a single statement within an action system can be interleaved with their neighbouring actions. We use this result extensively in Section 5 in a stepwise implementation of the Alternating Bit Protocol from its simple specification.

A specific version of the Movers Theorem was first studied by Lipton [Lip75]. Lamport [Lam90] generalises Lipton's result to distributed systems where processes communicate via channels. Back's movers theorem [Bac89] is a similar result in the general setting of action systems.

Our Movers Theorem is a generalisation of Back's movers theorem, which was proved for boundedly nondeterministic action systems, while we extend the result to fair action systems exhibiting unbounded nondeterminism. In addition, we make a further generalisation to allow for the simultaneous execution of two instances of the loop to be interleaved with the rest of the action system.

It is unfortunate that the statement and proof of the Movers Theorem are not simpler. Perhaps this is not surprising, however, because the actions of an action system can interact in complicated ways. In any case, our result extends that of Back [Bac89], which

*e-mail Axel.Wabenhorst@comlab.ox.ac.uk

is already complex. The extra complication in the proof results not from fairness, but from unbounded nondeterminism.

The Alternating Bit Protocol (ABP) is an important case study because it has been used as a benchmark against which to test several new theories. As a result, it has been dealt with extensively in the literature [Mil80, PS91, Sne95, CM88, Col95]. We discuss it here for the same reason, and to gain an understanding of the results of previous sections.

In common with many other algorithms and protocols, the ABP has either not been verified rigorously, or has been proven correct using *ad hoc* methods which cannot be transported easily to other problems. We aim to remedy this difficulty by using fair action systems, which are general enough to be used in a wide variety of situations.

The ABP is a protocol for verifying data transmission through unreliable media. The principle is as follows: the sender obtains each datum from a (reliable) input channel and sends it via an unreliable medium to the receiver. We will presume that garbled data can be detected, so that we can assume that each datum sent is either received intact or is lost. The receiver outputs the datum received onto a (reliable) output channel and sends an acknowledgement back to the sender through another unreliable medium, in such a way that the sender knows which datum received is being acknowledged. Once the sender has received an acknowledgement for the datum which was last sent, the next datum can be sent.

The development of this protocol needs a formalism which can express fairness constraints on the media. If a medium loses each datum, no communication takes place, which no protocol can rectify. If the medium transmits each datum correctly, then no protocol is needed. So while the medium is permitted to lose data, we make the fairness assumption that from any point onward, the medium will eventually transmit the datum. We should not make any stronger assumption about the behaviour of the medium, since in real life we may have no other knowledge of or control over the pattern of data loss by the medium. The concept of fairness has not been available in action systems before, so our treatment of the ABP represents a new development.

The background for this paper is [Wab96a], and have appeared previously in [Wab96b].

### 1.1 Outline

We begin in Section 2 by recalling the definition of predicate transformers, fair action systems and action sequences.

From our definition of fairness and its elementary properties, we state two theorems [Wab96a] useful in the development of terminating programs from simple specifications in Section 3, namely monotonicity and adding skips.

In Section 4, the Movers Theorem is proved in order to provide support for introducing concurrency into a development by identifying conditions under which the actions of an iteration executed atomically as a single statement within an action system can be interleaved with their neighbouring actions. We use this result extensively in Section 5 in a stepwise development of the Alternating Bit Protocol from its simple specification.

Finally, in Section 6, we summarise our results.

## 2 Background

### 2.1 Predicate Transformers

Let $\Sigma_{\mathcal{A}}$ denote the set of states of a set of variables. These will be the variables of the program $\mathcal{A}$, which will be defined later. A *predicate* on the set of program variables is a function $f : \Sigma_{\mathcal{A}} \to \mathbb{B}$, the set of all of which will be denoted by $Pred\mathcal{A}$. If there is no ambiguity, we write $Pred$ instead of $Pred\mathcal{A}$.

A *weakest precondition transformer* [Dij76] is a predicate transformer $S : Pred \to Pred$ satisfying, for all $P, Q : Pred$, the axioms

- If $P \Rightarrow Q$, then $S(P) \Rightarrow S(Q)$ (monotonicity);
- If $I \neq \emptyset$, then $\forall \alpha \in I \; S(P_\alpha) \Rightarrow S(\forall \alpha \in I \; P_\alpha)$ (conjunctivity).

For each weakest precondition $S$, there is a *statement*, also called $S$. The statements will be the commands of our program. Informally, we can regard the weakest precondition $S(P)$ for $P : Pred$ to be the weakest predicate on the variable states from which execution of $S$ guarantees termination in a state satisfying $P$. In particular, $S(true)$ is the weakest predicate which guarantees that $S$ terminates. Notice that we allow statements which do not satisfy *strictness*, $S(false) \Leftrightarrow false$.

Examples of statements and their associated weakest precondition transformers satisfying monotonicity and conjunctivity are given in [DS90, Boo82].

**Definition 1.** A statement $S$ *is refined by* a statement or action $S'$ (notation $S \leq S'$) if for all $P : Pred \; S(P) \Rightarrow S'(P)$. $S$ and $S'$ are called the *abstract* and *concrete* statements respectively. $\square$

Note that refinement is transitive, which allows the stepwise refinement of programs.

### 2.2 Action Systems and Action Sequences

The syntactic definition of a fair action system [Wab96a] consists of variable declarations and an initialisation followed by an iteration statement. Let $I$ be an index set. For $\alpha \in I$, let $P_\alpha : Pred$ such that $\exists \alpha P_\alpha \Leftrightarrow true$. Let

$$\begin{array}{llll}
\mathcal{A} \;\hat{=} & \textbf{var} & & x_1, \ldots, x_m \\
& \textbf{init} & & x_1, \ldots, x_m :\in X_1, \ldots, X_m \\
& \textbf{do} & []_{i=1}^{n} & g_i \;\; \rightarrow \;\; S_i \\
& & \textbf{uf} & P_\alpha : uf_\alpha \\
& \textbf{od}, & &
\end{array}$$

where $uf_\alpha \subseteq seq_\omega \mathcal{A}$ for all $\alpha \in I$.

The semantics of a fair action system will be defined later. Informally, a nondeterministic choice is made between the actions whose guards are true, subject to the constraint that no infinite sequence of actions in $uf_\alpha$ occurs after an initialisation satisfying $P_\alpha$. The loop terminates if no guard is true. Concurrent execution of two actions is possible only if different variables are modified in each, but not if two actions modify the same variables.

For the purposes of calculating weakest preconditions, we regard the initialisation as a statement in sequential composition with the subsequent iteration and assume that the guards of the actions are calculated instantaneously.

An *action sequence* is a sequence (either finite or infinite) of actions in an action system. For an action system $\mathcal{A}$, $seq_k\mathcal{A}$ consists of the action sequences of length $k : \mathbb{N}$, with semantics given by the sequential composition between actions. $seq_0\mathcal{A} \;\widehat{=}\; \{skip\}$. The set of all finite action sequences is denoted by $seq\mathcal{A}$, and the set of infinite action sequences by $seq_\omega\mathcal{A}$. The set $\overline{seq}\mathcal{A}$ consists of all action sequences, both finite and infinite. For an action sequence $s$, $s|_k$ is the prefix of $s$ of length $k : \mathbb{N}$.

Let $u[A_i/A_j]$ be the action sequence resulting from the replacement of $A_j$ throughout $u$ by $A_i$. Similarly, $u[/A_j]$ is the action sequence resulting from the removal of all occurrences of $A_j$ in $u$. If $u$ consists only of $A_j$'s, then instead of $u[/A_j]$ being the empty sequence, we define $u[/A_j] \;\widehat{=}\; skip$ so that $u[/A_j]$ is still a predicate transformer.

For sets $S$ and $T$ of finite action sequences, $S;T$ has the usual pointwise definition.

Temporal logic enables the succinct characterisation of sets of action sequences. We define the temporal operators $\Box$ (always) and $\Diamond$ (eventually) in the spirit of Lamport's Temporal Logic of Actions (TLA) [Lam91]. For a set of actions $\mathcal{V}$,

- $\Box\mathcal{V}$ is the set of action sequences containing only actions in $\mathcal{V}$;
- $\Diamond\mathcal{V}$ is the set of action sequences which eventually contains an action in $\mathcal{V}$;
- $\Box\Diamond\mathcal{V}$ is the set of action sequences containing actions in $\mathcal{V}$ infinitely often; and
- $\Diamond\Box\mathcal{V}$ is the set of action sequences containing only actions in $\mathcal{V}$ from a certain point onwards.

We define the semantics of a fair action system to be

$$\begin{array}{rcl}\mathcal{A}(P) & \widehat{=} & \forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i) \\ & & \wedge \\ & & \exists \alpha \in I\ P_\alpha \wedge \exists \kappa\ \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}\ ,\end{array}$$

where

$$\begin{array}{rcl}\mathcal{H}^*_{0,\mathcal{A},\alpha} & \widehat{=} & \neg \bigvee_{i=1}^{n} g_i \\ \mathcal{H}^*_{\kappa+1,\mathcal{A},\alpha} & \widehat{=} & \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m},\alpha}) \\ \mathcal{H}^*_{\delta,\mathcal{A},\alpha} & \widehat{=} & \exists \kappa < \delta\ \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}\end{array}$$

and for $t : seq\mathcal{A}$,

$$\begin{array}{rcll}\mathcal{A}^t & \widehat{=} & \mathbf{do} & [\!]_{i=1}^{n}\ g_i \ \rightarrow\ S_i \\ & & & \mathbf{uf}\quad P_\alpha : \{u : seq_\omega\mathcal{A} \mid t; u : uf_\alpha\} \\ & & \mathbf{od}. & \end{array}$$

The first conjunct in the definition is a safety constraint, while the second conjunct is a liveness constraint.

Since fair action systems satisfy conjunctivity and monotonicity, fair action systems can be used as statements within other programs. A fair action system with unfair action sequences $uf_\alpha$ is refined by a fair action system with the same actions and unfair action sequences containing $uf_\alpha$.

**Lemma 2.** *Let $\kappa < \mu$. Then $\mathcal{H}^*_{\kappa,\mathcal{A}} \Rightarrow \mathcal{H}^*_{\mu,\mathcal{A}}$.* □

**Lemma 3.** *Let $s, t : seq\mathcal{A}$ with $s \sqsubseteq t$. Then $s(\mathcal{H}^*_{\kappa,\mathcal{A}^s}) \Rightarrow t(\mathcal{H}^*_{\kappa,\mathcal{A}^t})$.* □

### 2.3 Data Refinement

The following theorem gives a method for obtaining a concrete action system given an abstract action system and a data refinement relation $\varphi$:

**Theorem 4.** *Let $S_i \leq_\varphi T_i$ for $1 \leq i \leq n$. Let*

$$\begin{array}{lllll} \mathcal{A} \;\widehat{=} & \mathbf{do} & []_{i=1}^n & g_i & \rightarrow \; S_i \\ & & \mathbf{uf} & P_\alpha : uf_\alpha & \\ & \mathbf{od} & & & \end{array}$$

*and*

$$\begin{array}{lllll} \mathcal{B} \;\widehat{=} & \mathbf{do} & []_{i=1}^n & \overline{\varphi}(g_i) & \rightarrow \; T_i \\ & & \mathbf{uf} & \varphi(P_\alpha) : uf'_\alpha & \\ & & & \neg\exists \alpha \in I \;\varphi(P_\alpha) : \emptyset & \\ & \mathbf{od}, & & & \end{array}$$

*where $\overline{\varphi}(g_i) \;\widehat{=}\; \neg\varphi(\neg g_i)$ and*

$$uf'_\alpha \;\widehat{=}\; \{u[\overline{\varphi}(g_i) \rightarrow T_i / g_i \rightarrow S_i] \mid u : uf_\alpha \wedge 1 \leq i \leq n\}.$$

*Then $\mathcal{A} \leq_\varphi \mathcal{B}$.* □

Thus, data refinement $\leq_\varphi$ distributes through the components of a fair action system.

Since fair action systems satisfy the axioms of monotonicity and conjunctivity, data refinement of fair action systems is complete provided the abstract action system is boundedly nondeterministic (see Section 2.1).

## 3 Program Development

In this section, we give some simple rules by which programs may be developed.

### 3.1 Monotonicity

Monotonicity allows us to refine actions one-by-one with the result that the action system as a whole is refined also.

**Theorem 5 Monotonicity for Fair Action Systems.** *Let*

$$\mathcal{A} \mathrel{\widehat{=}} \begin{array}{ll} \textbf{do} & \begin{array}{lll} []_{i=1}^{n} & g_i & \rightarrow \ S_i \\ \textbf{uf} & P_\alpha : uf_\alpha & \end{array} \\ \textbf{od} & \end{array}$$

*and*

$$\mathcal{B} \mathrel{\widehat{=}} \begin{array}{ll} \textbf{do} & \begin{array}{lll} []_{i=1}^{n}[]_{j=1}^{m_i} & h_{i_j} & \rightarrow \ T_{i_j} \\ \textbf{uf} & P_\alpha : uf'_\alpha & \end{array} \\ \textbf{od}, & \end{array}$$

*where*

- $\bigvee_{i=1}^{n} g_i \Leftrightarrow \bigvee_{i=1}^{n} \bigvee_{j=1}^{m_i} h_{i_j}$,
- *for all $i$ such that $1 \leq i \leq n$, $g_i \rightarrow S_i \leq []_{j=1}^{m_i} h_{i_j} \rightarrow T_{i_j}$ and*
- *for each $\alpha \in I$, $uf'_\alpha = \{u[h_{i_j} \rightarrow T_{i_j}/g_i \rightarrow S_i] \mid u : uf_\alpha \ \wedge \ 1 \leq j \leq m_i\}$.*

*Then $\mathcal{A} \leq \mathcal{B}$.* $\square$

Note that in the definition of $uf'_\alpha$, different $h_{i_j} \rightarrow T_{i_j}$ with varying $j$ may replace $g_i \rightarrow S_i$ with fixed $i$ in the *same* unfair sequence.

Also, note that

- $uf_\alpha = \emptyset \ \Rightarrow \ uf'_\alpha = \emptyset$, which gives us monotonicity for action systems executing according to the standard definition of iteration;
- $uf_\alpha = seq_\omega \mathcal{A} \ \Rightarrow \ uf'_\alpha = seq_\omega \mathcal{B}$, which gives us monotonicity for action systems executing according to the weakest errorfree precondition; and

As an application of our Monotonicity Theorem, we can strengthen the guards of actions in a fair action system, provided their disjunction is not changed.

## 3.2 Adding Skips

Occasionally, we would like to *add* an action to an action system, rather than merely to refine existing actions. The first step in this process, that of adding *skip*s, is justified by the following result:

**Theorem 6 Adding Skips.** *Let*

$$\mathcal{A} \mathrel{\widehat{=}} \begin{array}{ll} \textbf{do} & \begin{array}{lll} []_{i=1}^{n} & g_i & \rightarrow \ S_i \\ \textbf{uf} & P_\alpha : uf_\alpha & \end{array} \\ \textbf{od} & \end{array}$$

*and*

$$\mathcal{B} \mathrel{\widehat{=}} \begin{array}{ll} \textbf{do} & \begin{array}{llll} []_{i=1}^{n} & g_i & \rightarrow \ S_i & \\ [] & \bigvee_{i=1}^{n} g_i & \rightarrow \ skip & \qquad (A) \\ \textbf{uf} & P_\alpha : uf'_\alpha & & \end{array} \\ \textbf{od}, & \end{array}$$

*where for each $\alpha \in I$, $uf'_\alpha = \{u : seq_\omega \mathcal{B} \mid u[/A] : uf_\alpha\} \cup \Diamond\Box A$. Then $\mathcal{A} \leq \mathcal{B}$ and $\mathcal{B} \leq \mathcal{A}$.* $\square$

## 4 The Movers Theorem

Let

$$
\begin{array}{lll}
\mathcal{A} \mathrel{\hat{=}} & init; & \\
& \textbf{do} & \quad g \rightarrow S \\
& & [\!]_{j=1}^{m} \; h_j \rightarrow T_j \\
& & \textbf{uf} \quad P_\alpha : uf_\alpha \\
& \textbf{od}, &
\end{array}
$$

and suppose that the action $g \rightarrow S$ is refined by the initialisation $g \rightarrow S_0$ followed by the action system

$$
\begin{array}{lll}
\mathcal{C} \mathrel{\hat{=}} & \textbf{do} & [\!]_{i=1}^{n} \; g_i \rightarrow S_i \\
& & \textbf{uf} \quad true : uf' \\
& \textbf{od}. &
\end{array}
$$

We wish to establish conditions under which the resulting action system can be "flattened" to obtain

$$
\begin{array}{lll}
\mathcal{B} \mathrel{\hat{=}} & init; & \\
& \textbf{do} & \quad g \rightarrow S_0 \\
& & [\!]_{i=1}^{n} \; g_i \rightarrow S_i \\
& & [\!]_{j=1}^{m} \; h_j \rightarrow T_j \\
& & \textbf{uf} \quad P_\alpha : uf''_\alpha \\
& \textbf{od} &
\end{array}
$$

for a suitable choice of $uf''_\alpha$. To prove this, we show that an action sequence in $\mathcal{A}$ can be refined to obtain an action sequence in $\mathcal{B}$ in "normal form" with actions in $\mathcal{C}$ consecutive. One can show that such an action sequence in normal form can be refined to an action sequence in $\mathcal{B}$ such that the actions of $\mathcal{C}$ are not necessarily consecutive, depending on the commutativity properties of the actions concerned. An arbitrary action sequence in $\mathcal{B}$ can be refined by some action sequence in $\mathcal{A}$ by this process.

First, we introduce some definitions (see [Bac89]) to help formalise these concepts.

**Definition 7.**

- An action $A$ *does not disable* actions $g_i \rightarrow S_i$, $1 \leq i \leq n$, if $\bigvee_{i=1}^{n} g_i \Rightarrow A(\bigvee_{i=1}^{n} g_i)$.
- An action $A$ *disables* actions $g_i \rightarrow S_i$, $1 \leq i \leq n$, if $A(\neg \bigvee_{i=1}^{n} g_i) \Leftrightarrow true$.
- An action $A$ *does not enable* actions $g_i \rightarrow S_i$, $1 \leq i \leq n$, if $\neg \bigvee_{i=1}^{n} g_i \Rightarrow A(\neg \bigvee_{i=1}^{n} g_i)$.
- The actions $g_i \rightarrow S_i$, $1 \leq i \leq n$, *exclude* an action $h \rightarrow T$ if $\bigvee_{i=1}^{n} g_i \Rightarrow \neg h$.
- An action $A$ *commutes with* an action $B$ if $A; B \leq B; A$.

□

Note that if $A$ commutes with $B$, then $B$ does not necessarily commute with $A$.

Now we define functions $\varphi$ and $\psi$ which map action sequences of $\mathcal{B}$ to those in $\mathcal{A}$ and $\mathcal{C}$ respectively. Given an action sequence $s : seq_\omega \mathcal{B}$, $\varphi(s)$ is the action sequence of $\mathcal{A}$ obtained by removing any $g_i \rightarrow S_i$ and changing $g \rightarrow S_0$ to $g \rightarrow S$, and which we will refine to $s$. The function $\psi$ gives us an action sequence of $\mathcal{C}$ by removing all actions except the $g_i \rightarrow S_i$ after the last $g \rightarrow S_0$. This function is not defined for $s : seq_\omega \mathcal{B}$ if $s$ has infinitely many $g \rightarrow S_0$.

$$\begin{array}{rcl}
\varphi : \overline{seq}\mathcal{B} & \rightarrow & \overline{seq}\mathcal{A}, \\
\varphi(skip) & \widehat{=} & skip \\
\varphi(g \rightarrow S_0; s) & \widehat{=} & g \rightarrow S; \varphi(s) \\
\varphi(g_i \rightarrow S_i; s) & \widehat{=} & \varphi(s) \\
\varphi(h_j \rightarrow T_j; s) & \widehat{=} & h_j \rightarrow T_j; \varphi(s)
\end{array}$$

with extension to $seq_\omega \mathcal{B}$ by continuity, and

$$\begin{array}{rcl}
\psi : \overline{seq}\mathcal{B} & \rightarrow & \overline{seq}\mathcal{C}, \\
\psi(skip) & \widehat{=} & skip \\
\psi(g \rightarrow S_0; s) & \widehat{=} & \psi(s) \\
\psi(h_j \rightarrow T_j; s) & \widehat{=} & \psi(s) \\
\psi(g_i \rightarrow S_i; s) & \widehat{=} & \left\{ \begin{array}{ll}
g_i \rightarrow S_i; s & \text{if } s \text{ does not contain } g \rightarrow S_0 \\
\varphi(s'') & \text{if } s = s'; g \rightarrow S_0; s'' \text{ and} \\
 & \quad s'' \text{ does not contain } g \rightarrow S_0 \\
\text{undefined} & \text{otherwise,}
\end{array} \right.
\end{array}$$

with extension to $seq_\omega \mathcal{B}$ by continuity.

We now define the "normal form" of an action sequence in $\mathcal{B}$. Informally, $nf$ tests if $s : seq\mathcal{B}$ is in "normal form", in which the initialising action $g \rightarrow S_0$ is followed immediately by the corresponding actions $g_i \rightarrow S_i$.

$$\begin{array}{rcl}
nf : seq\mathcal{B} & \rightarrow & \mathbb{B}, \\
nf(skip) & \widehat{=} & true \\
nf(s; g \rightarrow S_0) & \widehat{=} & nf(s) \\
nf(s; R) & \widehat{=} & nf(s) \\
nf(s; L) & \widehat{=} & \left\{ \begin{array}{ll}
nf(s) & \text{if } s = skip \text{ or } s = t; L \\
false & \text{otherwise}
\end{array} \right. \\
nf(s; g_i \rightarrow S_i) & \widehat{=} & \left\{ \begin{array}{ll}
false & \text{if} s = t; R \\
nf(s) & \text{otherwise.}
\end{array} \right.
\end{array}$$

The first step in the process of refining an action sequence in $\mathcal{A}$ is to obtain an action sequence in $\mathcal{B}$ in this normal form. Next, we change the order of the actions so that the $g_i \rightarrow S_i$ are not necessarily consecutive. We will now define the relation $\Xi$, expressing whether or not this last step is a refinement. Informally, we can read $(u', u) \in \Xi$ as "$u$ shuffles $u'$".

Let $\mathcal{L}$ and $\mathcal{R}$ be subsets of $\{h_j \rightarrow T_j \mid 1 \leq j \leq m\}$, called *left movers* and *right movers* respectively. Informally, left movers commute with every other kind of action and the $g_i \rightarrow S_i$ commute with right movers. In addition, if one of the $g_i \rightarrow S_i$ is enabled, then any $g_i \rightarrow S_i$ commutes with $g \rightarrow S_0$. This reflects the situation where $g \rightarrow S_0$ adds a datum to a channel while $g_i \rightarrow S_i$ removes a datum from the channel. In this case, if the channel is non-empty ($\bigvee_{i=1}^{n} g_i$ holds), then adding a datum to the channel followed by removing a datum from the channel is refined by removing then adding.

More formally, for $u', u : \overline{seq\mathcal{B}}$, $(u', u) \in \Xi_0$ if and only if $u' = u$ or there exists $s : seq\mathcal{B}$ and $v : \overline{seq\mathcal{B}}$ such that one of the following holds:

$$
\begin{array}{rcl}
\exists L \in \mathcal{L} & & u' = s; L; g \rightarrow S_0; v \\
& & \wedge \\
& & u = s; g \rightarrow S_0; L; v \\
& OR & \\
\exists L \in \mathcal{L}\ \exists i : 1 \leq i \leq n & & u' = s; L; g_i \rightarrow S_i; v \\
& & \wedge \\
& & u = s; g_i \rightarrow S_i; L; v \\
& OR & \\
\exists L \in \mathcal{L}\ \exists R \in \mathcal{R} & & u' = s; L; R; v \\
& & \wedge \\
& & u = s; R; L; v \\
& OR & \\
\exists R \in \mathcal{R}\ \exists i : 1 \leq i \leq n & & u' = s; g_i \rightarrow S_i; R; v \\
& & \wedge \\
& & u = s; R; g_i \rightarrow S_i; v \\
& OR & \\
s(\bigvee_{i=1}^{n} g_i) \Leftrightarrow s(true) & & u' = s; g_i \rightarrow S_i; g \rightarrow S_0; v \\
\wedge \exists i : 1 \leq i \leq n & & \wedge \\
& & u = s; g \rightarrow S_0; g_i \rightarrow S_i; v.
\end{array}
$$

Let $\Xi$ denote the transitive closure of the relation $\Xi_0$.

We are now ready to state the Movers Theorem; $!(\neg \bigvee_{i=1}^{n} g_i)$ is an assertion which skips if $(\neg \bigvee_{i=1}^{n} g_i)$ holds and aborts otherwise.

**Theorem 8 Movers Theorem.** *Let action systems $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be as above, with $uf''_\alpha$ yet to be defined. Let*

$$!(\neg \bigvee_{i=1}^{n} g_i); g \rightarrow S \leq g \rightarrow S_0; \mathcal{C}, \tag{1}$$

*and*

1. (a) *For all $i, j$ such that $1 \leq j \leq m$ and $1 \leq i \leq n$, $h_j \rightarrow T_j$ does not disable $g_i \rightarrow S_i$.*

(b) *For all $i, j$ such that $1 \leq j \leq m$ and $1 \leq i \leq n$, $h_j \rightarrow T_j$ does not enable $g_i \rightarrow S_i$.*

(c) *For all $i$ such that $1 \leq i \leq n$, the initialisation init disables $g_i \rightarrow S_i$.*

2. *EITHER*

   (a) *The actions $g_i \rightarrow S_i$, $1 \leq i \leq n$, exclude $g \rightarrow S_0$.*
   *OR*

   (b) i. *For all $i$ such that $1 \leq i \leq n$, $g_i \rightarrow S_i$ commutes with $g \rightarrow S_0$, provided $\bigvee_{i=1}^{n} g_i$ holds initially.*

   ii. *For all $i$ such that $1 \leq i \leq n$, $g \rightarrow S_0$ does not disable $g_i \rightarrow S_i$.*

   iii. *For all $s : seq\mathcal{C}$, either $g \rightarrow S_0; s(true) \Leftrightarrow g \rightarrow S_0; s(\bigvee_{i=1}^{n} g_i)$ or $g \rightarrow S_0; s(true) \Leftrightarrow g \rightarrow S_0; s(\neg \bigvee_{i=1}^{n} g_i)$.*

3. *The set $\{h_j \rightarrow T_j \mid 1 \leq j \leq m\}$ can be expressed as the disjoint union of sets $\mathcal{L}$ (called the* left movers*) and $\mathcal{R}$ (called the* right movers*), such that:*

   (a) *For all $L \in \mathcal{L}$, $L$ commutes with $g \rightarrow S_0$.*

   (b) *For all $L \in \mathcal{L}$ and $i$ such that $1 \leq i \leq n$, $L$ commutes with $g_i \rightarrow S_i$.*

   (c) *For all $L \in \mathcal{L}$ and $R \in \mathcal{R}$, $L$ commutes with $R$.*

   (d) *For all $R \in \mathcal{R}$ and $i$ such that $1 \leq i \leq n$, $g_i \rightarrow S_i$ commutes with $R$.*

   (e) *$\bigvee_{i=1}^{n} g_i \Rightarrow \mathcal{D}(true)$, where*

$$\mathcal{D} \mathrel{\widehat{=}} \textbf{do} \quad [\!]_{R \in \mathcal{R}}\ R \quad \textbf{uf} \quad P_\alpha : \{u : seq_\omega \mathcal{D} \mid \exists s : seq\mathcal{B}\ s; u : uf''_\alpha\} \quad \textbf{od}.$$

   (f) *No left mover occurs in $uf_\alpha$.*

   (g) *All left movers and all $g_i \rightarrow S_i$, $1 \leq i \leq n$, are boundedly nondeterministic.*

*Define*

$$uf''_\alpha \mathrel{\widehat{=}} \varphi^{-1}(uf_\alpha) \cup \psi^{-1}(uf') \cup U,$$

*where*

$$U \mathrel{\widehat{=}} \begin{cases} \emptyset & \text{if 2a holds} \\ \Box\Diamond g \rightarrow S_0 \ \cap \ \Diamond\Box(\{h_j \rightarrow T_j \mid 1 \leq j \leq m\} \cup \{g \rightarrow S_0\}) & \text{otherwise} \end{cases}$$

*Then $\mathcal{A} \leq \mathcal{B}$.* □

The statement of the theorem for the special case of $uf_\alpha = \emptyset$ and 2a is almost the same as Back's. Notice that 2a implies 2(b)i and 2(b)ii. The extra complication in 2 arises from our need to consider a different set of circumstances which would otherwise have required a separate theorem. In particular, we will see in the development of the Alternating Bit Protocol that $g \rightarrow S_0$ corresponds to the addition of a datum to a channel, while $g_1 \rightarrow S_1$

corresponds to the removal of a datum from the same channel. In this case, 2a does not hold, whereas 2b does. The extra unfair sequences in $U$, namely those containing infinitely many $g \to S_0$ but only finitely many $g_1 \to S_1$, correspond to the indefinite addition of data to the channel without removing any data.

We require that all left movers and all $g_i \to S_i$ be boundedly nondeterministic. While we have not given up hope that this is not a necessary requirement, we have not been able to derive a proof without this condition.

Before embarking on a proof, we need a series of lemmas.

The following lemma shows that the shuffle of two finite action sequences of the concrete action system constitutes a refinement.

**Lemma 9.** *Let $u', u : seq\mathcal{B}$ with $(u', u) \in \Xi$. Then $u' \leq u$.*

*Proof.* Since the five conditions 3a, 3b, 3c, 3d, 2(b)i correspond to the five lines in the definition of $\Xi$, the proof follows easily. □

The following lemma shows that any finite action sequence of the concrete action system can be shuffled to another finite action sequence in normal form. By Lemma 9, this is a refinement.

**Lemma 10.** *Let $s : seq\mathcal{B}$. Then there exists $s' : seq\mathcal{B}$ with $(s', s) \in \Xi$ and $nf(s')$.*

*Proof.* We will prove the lemma by induction on the length of $s$.

If the length of $s$ is 0, i.e. $s = skip$, then the result holds.

Suppose that the lemma holds for all of $seq_k\mathcal{B}$, and let $s : seq_{k+1}\mathcal{B}$, $s = t; A$, where $A$ is an action. There are several cases to consider.

<u>$A = R$ or $A = g \to S_0$</u>: By the induction hypothesis, there exists $t' : seq\mathcal{B}$ such that $(t', t) \in \Xi$ and $nf(t')$. Let $s' = t'; A$. By definition of $nf$, $nf(s')$. Also $(s', s) \in \Xi$.

<u>$A = g_i \to S_i$</u>: If $t \neq t_1; R$, then we follow the same reasoning as in the above case. If $t = t_1; R$, then by the induction hypothesis, there exists $t' : seq\mathcal{B}$ such that $(t', t_1; A) \in \Xi$ and $nf(t')$. Take $s' = t'; R$, then $nf(s')$. Since $(s', t_1; A; R) \in \Xi$ and $(t_1; A; R, s) = (t_1; A; R, t_1; R; A) \in \Xi$, it follows from the transitivity of $\Xi$ that $(s', s) \in \Xi$.

<u>$A = L$</u>: If $t = t_1; L'$, then we follow the same reasoning as in the first case. If $t = t_1; B$ for $B \neq L'$, then $(t_1; A; B, s) = (t_1; A; B, t_1; B; A) \in \Xi$. Now apply one of the above cases with $t_1; A; B$ for $s$ to obtain $s' : seq\mathcal{B}$ with $nf(s')$ and $(s', t_1; A; B) \in \Xi$. Then $(s'; s) \in \Xi$ follows from the transitivity of $\Xi$. □

If 2(b)i holds, then we can assume without loss of generality that $\neg \bigvee_{i=1}^{n} g_i$ holds before every $g \to S_0$, provided some $g_i \to S_i$ occurs after it.

The following lemma shows that a finite action sequence in the concrete action system almost refines the corresponding (via $\varphi$) action sequence in the abstract action sequence. This is not quite a refinement because the execution of $\mathcal{C}$ may not yet have terminated. Much of the complication of the proof of the Movers Theorem arises from showing the termination of $\mathcal{C}$ in the presence of unbounded nondeterminism.

**Lemma 11.** *For all $s : seq\mathcal{B}$, $\neg \bigvee_{i=1}^{n} g_i \wedge \varphi(s)(P) \Rightarrow s(P \vee \bigvee_{i=1}^{n} g_i)$.*

*Proof.* Let $s : seq\mathcal{B}$. From Lemma 10, we can assume without loss of generality that $nf(s)$. We proceed by induction on the construction of $s$.

The result holds trivially for $s = skip$ and $s = h_j \rightarrow T_j; t$. For $s = h_j \rightarrow T_j; t$,

$$
\begin{array}{ll}
& \neg \bigvee_{i=1}^{n} g_i \wedge \varphi(h_j \rightarrow T_j; t)(P) \\
\Leftrightarrow & \hfill \text{definition of } \varphi \\
& \neg \bigvee_{i=1}^{n} g_i \wedge h_j \rightarrow T_j; \varphi(t)(P) \\
\Rightarrow & \hfill \text{1b} \\
& h_j \rightarrow T_j(\neg \bigvee_{i=1}^{n} g_i \wedge \varphi(t)(P)) \\
\Rightarrow & \hfill \text{induction hypothesis and monotonicity} \\
& h_j \rightarrow T_j; t(P \vee \bigvee_{i=1}^{n} g_i).
\end{array}
$$

Otherwise, $s = g \rightarrow S_0; t_1; t_2$, where $t_1$, possibly empty, contains only $g_i \rightarrow S_i$ and $t_2$, possibly empty, does not begin with $g_i \rightarrow S_i$.

$$
\begin{array}{ll}
& \neg \bigvee_{i=1}^{n} g_i \wedge \varphi(g \rightarrow S_0; t_1; t_2)(P) \\
\Leftrightarrow & \hfill \text{definition of } \varphi \\
& \neg \bigvee_{i=1}^{n} g_i \wedge g \rightarrow S; \varphi(t_2)(P) \\
\Rightarrow & \hfill \text{from 1, } \neg \bigvee_{i=1}^{n} g_i \wedge g \rightarrow S(P) \Rightarrow g \rightarrow S_0(\forall s : seq\mathcal{C}\ s(P \vee \bigvee_{i=1}^{n} g_i)) \\
& g \rightarrow S_0; t_1(\varphi(t_2)(P) \vee \bigvee_{i=1}^{n} g_i) \\
\Leftrightarrow & \\
& g \rightarrow S_0; t_1((\neg \bigvee_{i=1}^{n} g_i \wedge \varphi(t_2)(P)) \vee \bigvee_{i=1}^{n} g_i) \\
\Rightarrow & \hfill \text{induction hypothesis and monotonicity} \\
& g \rightarrow S_0; t_1(t_2(P \vee \bigvee_{i=1}^{n} g_i) \vee \bigvee_{i=1}^{n} g_i).
\end{array}
$$

The result holds if $t_2 = skip$. Otherwise, $t_2 = h_j \rightarrow T_j; t_3$ or $t_2 = g \rightarrow S_0; t_3$.

In the latter case, we prove the result by induction on the number of occurrences of $g_i \rightarrow S_i$ in $t_3$. In the base case where $t_3$ contains no $g_i \rightarrow S_i$, the result holds after repeated applications of 1a and 2(b)ii. In the step case, we can conclude from $nf(s)$ that $t_3 = g_i \rightarrow S_i; t_4$ for some $t_4$. We have shown that

$$g \rightarrow S_0; t_1(t_2(P \vee \bigvee_{i=1}^{n} g_i) \vee \bigvee_{i=1}^{n} g_i) \tag{2}$$

and it can be shown similarly that

$$g \rightarrow S_0; t_1; g_i \rightarrow S_i(g \rightarrow S_0; t_4(P \vee \bigvee_{i=1}^{n} g_i) \vee \bigvee_{i=1}^{n} g_i).$$

Therefore,

$$g \to S_0; t_1; g_i \to S_i(g \to S_0; t_4(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ 2(b)ii and monotonicity

$$g \to S_0; t_1; g_i \to S_i; g \to S_0(t_4(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i)$$

$\Leftrightarrow$

$$g \to S_0; t_1((\neg \bigvee_{i=1}^n g_i \vee \bigvee_{i=1}^n g_i) \wedge g_i \to S_i; g \to S_0(t_4(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i))$$

$\Rightarrow$ $\neg \bigvee_{i=1}^n g_i \Rightarrow g_i \to S_i(Q)$

$$g \to S_0; t_1(\neg \bigvee_{i=1}^n g_i \vee (\bigvee_{i=1}^n g_i \wedge g_i \to S_i; g \to S_0(t_4(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i)))$$

$\Rightarrow$ 2(b)i and monotonicity

$$g \to S_0; t_1(\neg \bigvee_{i=1}^n g_i \vee g \to S_0; g_i \to S_i(t_4(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i))$$

$\Rightarrow$ induction hypothesis: $t_4$ contains one less $g_i \to S_i$ than $t_3$

$$g \to S_0; t_1(\neg \bigvee_{i=1}^n g_i \vee g \to S_0; g_i \to S_i(t_4(P \vee \bigvee_{i=1}^n g_i)))$$

$\Leftrightarrow$ definition of $t_2$

$$g \to S_0; t_1(\neg \bigvee_{i=1}^n g_i \vee t_2(P \vee \bigvee_{i=1}^n g_i)).$$

Taking the conjunction of this with equation 2, and applying the conjunctivity axiom, we obtain the desired result.

In the case where $t_2 = h_j \to T_j; t_3$, we apply

$$g \to S_0; t_1(t_2(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ 1a

$$g \to S_0; t_1(h_j \to T_j; t_3(P \vee \bigvee_{i=1}^n g_i) \vee h_j \to T_j(\bigvee_{i=1}^n g_i))$$

$\Rightarrow$ monotonicity

$$g \to S_0; t_1; h_j \to T_j(t_3(P \vee \bigvee_{i=1}^n g_i) \vee \bigvee_{i=1}^n g_i)$$

repeatedly until $t_3 = \mathit{skip}$ or $t_3 = g \to S_0; t_4$. Therefore, the result holds. $\square$

The following lemma shows that any fair infinite sequence of the concrete action system arises from a fair action sequence in the abstract action system or in $\mathcal{C}$. (This is an obvious criterion for $uf''_\alpha$ to satisfy.)

**Lemma 12.** *Let $s : seq_\omega \mathcal{B} \setminus uf''_\alpha$. Then either $\varphi(s) : seq_\omega \mathcal{A} \setminus uf_\alpha$ or $\psi(s) : seq_\omega \mathcal{C} \setminus uf'$.*

*Proof.* Certainly either $\varphi(s) : seq_\omega \mathcal{A}$ or $\psi(s) : seq_\omega \mathcal{C}$, otherwise we would have $s : seq \mathcal{B}$, a contradiction. Suppose that $\varphi(s) : seq_\omega \mathcal{A}$. If $\varphi(s) : uf_\alpha$, then $s : uf''_\alpha$ by definition of $uf''_\alpha$, a contradiction. On the other hand, suppose $\psi(s) : seq_\omega \mathcal{C}$ and $\psi(s) : uf'$. Then by definition of $uf''_\alpha$, $s : uf''_\alpha$, a contradiction. $\square$

The following lemma is the counterpart of Lemmas 9 and 10 for fair infinite action sequences rather than finite ones. It will allow us to assume that without loss of generality, all fair infinite action sequences in the concrete action system have all (finite) prefixes in normal form.

**Lemma 13.** *Let $s : seq_\omega \mathcal{B} \setminus uf''_\alpha$. Then there exists $v : seq_\omega \mathcal{B} \setminus uf''_\alpha$ such that for all $m < \omega$, $nf(v|_m)$ and*

$$\exists m < \omega \ v|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{v|m}}) \Rightarrow \exists m < \omega \ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|m}}).$$

*Proof.* By Lemma 10, we can assume that for all $m < \omega$, there exists $v_m : seq_m \mathcal{B}$ such that $(v_m, s|_m) \in \Xi$ and $nf(v_m)$.

Now we will show that

$$\forall m < \omega \ \exists m' > m \ \forall m'' > m' \ v_{m'}|_m = v_{m''}|_m.$$

Let $m < \omega$. Since left movers commute with every other kind of statement but no other kind of statement commutes with a left mover, choose $m'$ so that $v_{m'}$ contains at least $m$ left movers if possible, this gives the required result. If $s$ contains less than $m$ left movers, choose $m'$ so that $v_{m'}$ contains all left movers and as many occurrences of $g_i \rightarrow S_i$ as necessary to ensure that no application of 3d to any further $g_i \rightarrow S_i$ which changes $v_{m'}|_m$ is possible.

Define $v : seq_\omega \mathcal{B}$ by $v|_m \ \widehat{=} \ v_{m'}|_m$, where $m'$ is given above; this is well-defined by the above. For all $m < \omega$, $nf(v|_m)$ since $nf(v_{m'}) \Rightarrow nf(v_{m'}|_m)$.

We now show that $v : seq_\omega \mathcal{B} \setminus uf''_\alpha$. First suppose that $s : \varphi^{-1}(seq_\omega \mathcal{A} \setminus uf_\alpha)$. For a contradiction, suppose that $v : \varphi^{-1}(uf_\alpha)$. By 3f, $v : uf''_\alpha$ does not contain any left mover. Hence $s$ does not contain any left mover and $\varphi(v) = \varphi(s)$ by the definition of $\varphi$ and $\Xi$. This is a contradiction to $\varphi(s) : seq_\omega \mathcal{A} \setminus uf_\alpha$. Therefore, $v : \varphi^{-1}(seq_\omega \mathcal{A} \setminus (uf_\alpha)$.

If $s \notin \varphi^{-1}(seq_\omega \mathcal{A} \setminus uf_\alpha)$, then by Lemma 12, $s : \psi^{-1}(seq_\omega \mathcal{C} \setminus uf')$. Since $\psi(s) = \psi(v)$, $v : \psi^{-1}(seq_\omega \mathcal{C} \setminus uf')$ also.

Finally,

$$\exists m < \omega \ v|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{v|m}})$$

$\Rightarrow$ definition of $v$

$$\exists m < \omega \ \exists m' < \omega \ v_{m'}|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{v_{m'}|m}})$$

$\Rightarrow$ Lemma 3

$$\exists m' < \omega \ v_{m'}(\mathcal{H}^*_{\kappa,\mathcal{B}^{v|_{m'}}})$$

$\Rightarrow$ Lemma 9

$$\exists m' < \omega \ s|_{m'}(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_{m'}}}).$$

□

We now prove the theorem.

*Proof.* We will show by induction on $\kappa$ that for all $P : Pred$, there exists $\kappa'$ such that $\neg \bigvee_{i=1}^n g_i \wedge \mathcal{H}^*_{\kappa,\mathcal{A}}(P) \Rightarrow \mathcal{H}^*_{\kappa',\mathcal{B}}(P)$. It will follow that

$\mathcal{A}(P)$

$\Leftrightarrow$ definition of $\mathcal{A}$

$init(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{j=1}^{m} h_j \vee g)\ \wedge\ \exists\kappa \mathcal{H}^*_{\kappa,\mathcal{A}}(P))$

$\Leftrightarrow$ 1c and conjunctivity

$init(\neg \bigvee_{i=1}^{n} g_i \wedge \forall s : seq\mathcal{A}\ s(P \vee \bigvee_{j=1}^{m} h_j \vee g)\ \wedge\ \neg \bigvee_{i=1}^{n} g_i \wedge \exists\kappa \mathcal{H}^*_{\kappa,\mathcal{A}}(P))$

$\Rightarrow$

$init(\neg \bigvee_{i=1}^{n} g_i \wedge \forall s : seq\mathcal{B}\ \varphi(s)(P \vee \bigvee_{j=1}^{m} h_j \vee g)\ \wedge\ \neg \bigvee_{i=1}^{n} g_i \wedge \exists\kappa \mathcal{H}^*_{\kappa,\mathcal{A}}(P))$

$\Leftrightarrow$

$init(\forall s : seq\mathcal{B}\ \neg \bigvee_{i=1}^{n} g_i \wedge \varphi(s)(P \vee \bigvee_{j=1}^{m} h_j \vee g)\ \wedge\ \exists\kappa\ \neg \bigvee_{i=1}^{n} g_i \wedge \mathcal{H}^*_{\kappa,\mathcal{A}}(P))$

$\Rightarrow$ Lemma 11

$init(\forall s : seq\mathcal{B}\ s(P \vee \bigvee_{j=1}^{m} h_j \vee g \vee \bigvee_{i=1}^{n} g_i)\ \wedge\ \exists\kappa\ \neg \bigvee_{i=1}^{n} g_i \wedge \mathcal{H}^*_{\kappa,\mathcal{A}}(P))$

$\Rightarrow$ result to be shown

$init(\forall s : seq\mathcal{B}\ s(P \vee \bigvee_{j=1}^{m} h_j \vee g \vee \bigvee_{i=1}^{n} g_i)\ \wedge\ \exists\kappa' \mathcal{H}^*_{\kappa',\mathcal{B}}(P))$

$\Leftrightarrow$ definition of $\mathcal{B}$

$\mathcal{B}(P)$.

Fix $\alpha \in I$. For the base step,

$\neg \bigvee_{i=1}^{n} g_i \wedge \mathcal{H}^*_{0,\mathcal{A}}$

$\Leftrightarrow$ definition of $\mathcal{H}^*_{0,\mathcal{A}}$

$\neg \bigvee_{i=1}^{n} g_i \wedge \neg \bigvee_{j=1}^{m} h_j \wedge \neg g$

$\Leftrightarrow$ definition of $\mathcal{H}^*_{0,\mathcal{B}}$

$\mathcal{H}^*_{0,\mathcal{B}}$,

while for the limit step

$\neg \bigvee_{i=1}^{n} g_i \wedge \mathcal{H}^*_{\delta,\mathcal{A}}$

$\Leftrightarrow$ definition of $\mathcal{H}^*_{\delta,\mathcal{A}}$

$\neg \bigvee_{i=1}^{n} g_i \wedge \exists\kappa < \delta\ \mathcal{H}^*_{\kappa,\mathcal{A}}$

$\Leftrightarrow$

$\exists\kappa < \delta\ \neg \bigvee_{i=1}^{n} g_i \wedge \mathcal{H}^*_{\kappa,\mathcal{A}}$

$\Rightarrow$ induction hypothesis, with $\delta' \ \widehat{=}\$ any limit ordinal at least as big as $\bigcup_{\kappa<\delta} \kappa'$

$\exists\kappa' < \delta'\ \mathcal{H}^*_{\kappa',\mathcal{B}}$

$\Leftrightarrow$ definition of $\mathcal{H}^*_{\delta,\mathcal{B}}$

$\mathcal{H}^*_{\delta,\mathcal{B}}$,

so only the successor step remains to be shown. We need to show that for $s : seq_\omega\mathcal{B} \setminus uf''_\alpha$, $\exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_m}})$. By Lemma 13, we can assume without loss of generality that $nf(s|_m)$ for all $m < \omega$.

By Lemma 12, we need only consider the two cases $\varphi(s) : seq_\omega\mathcal{A} \setminus uf_\alpha$ and $\psi(s) : seq_\omega\mathcal{C} \setminus uf'$. We will deal with the former case first.

$$\neg \bigvee_{i=1}^n g_i \wedge \mathcal{H}^*_{\kappa+1,\mathcal{A}}$$

$\Leftrightarrow$ definition of $\mathcal{H}^*_{\kappa+1,\mathcal{A}}$

$$\neg \bigvee_{i=1}^n g_i \wedge \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m' < \omega\ s|_{m'}(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|_{m'}}})$$

$\Rightarrow$

$$\forall s : \varphi^{-1}(seq_\omega\mathcal{A} \setminus uf_\alpha)\ \exists m' < \omega\ \neg \bigvee_{i=1}^n g_i \wedge \varphi(s)|_{m'}(\mathcal{H}^*_{\kappa,\mathcal{A}^{\varphi(s)|_{m'}}})$$

$\Leftrightarrow$ $\varphi(s) : seq_\omega\mathcal{A}$

$$\forall s : \varphi^{-1}(seq_\omega\mathcal{A} \setminus uf_\alpha)\ \exists m < \omega\ \neg \bigvee_{i=1}^n g_i \wedge \varphi(s|_m)(\mathcal{H}^*_{\kappa,\mathcal{A}^{\varphi(s|_m)}})$$

$\Rightarrow$ Lemma 11

$$\forall s : \varphi^{-1}(seq_\omega\mathcal{A} \setminus uf_\alpha)\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{\varphi(s|_m)}} \vee \bigvee_{i=1}^n g_i)$$

$\Leftrightarrow$

$$\forall s : \varphi^{-1}(seq_\omega\mathcal{A} \setminus uf_\alpha)\ \exists m < \omega\ s|_m((\neg \bigvee_{i=1}^n g_i \wedge \mathcal{H}^*_{\kappa,\mathcal{A}^{\varphi(s|_m)}}) \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ induction hypothesis and monotonicity

$$\forall s : \varphi^{-1}(seq_\omega\mathcal{A} \setminus uf_\alpha)\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa',\mathcal{B}^{s|_m}} \vee \bigvee_{i=1}^n g_i).$$

There are three cases to consider,

- $g \rightarrow S_0$ occurs infinitely often in $s$,
- infinitely many left movers occur in $s$ and
- infinitely many right movers occur in $s$.

These are exhaustive.

<u>$g \rightarrow S_0$ occurs infinitely often in $s$</u>: Suppose first that 2a holds. By Lemma 3, we can assume without loss of generality that $s|_{m+1} = s|_m; g \rightarrow S_0$. Hence,

$$\exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_m}} \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ Lemma 3

$$\exists m < \omega\ s|_m(g \rightarrow S_0(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_{m+1}}}) \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ 2a and monotonicity

$$\exists m < \omega\ s|_m(g \rightarrow S_0(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_{m+1}}}) \vee \neg g)$$

$\Rightarrow$ monotonicity

$$\exists m < \omega \ s|_m(g \rightarrow S_0(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m+1}}}))$$

$\Leftrightarrow$

$$\exists m < \omega \ s|_{m+1}(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m+1}}}).$$

Now suppose that 2a does not hold, and that 2b holds. Since $nf(s|_m)$ for all $m < \omega$ and by Lemma 3, we can choose $m < \omega$ such that

$$s = s|_{m-1}; g_i \rightarrow S_i; v; g \rightarrow S_0; u,$$

where $v : seq\mathcal{B}$ consists only of right movers. By the definition of $uf''_\alpha$, $g_i \rightarrow S_i$ occurs infinitely often in $s$. Therefore, by 2(b)i, we can assume without loss of generality that

$$s|_{m-1}; g_i \rightarrow S_i(true) \Leftrightarrow s|_{m-1}; g_i \rightarrow S_i(\neg \bigvee_{i=1}^{n} g_i)$$

and so

$$\exists m < \omega \ s|_{m-1}; g_i \rightarrow S_i; v(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m-1}; g_i \rightarrow S_i; v}} \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ $\qquad s|_{m-1}; g_i \rightarrow S_i(true) \Leftrightarrow s|_{m-1}; g_i \rightarrow S_i(\neg \bigvee_{i=1}^n g_i)$

$$\exists m < \omega \ s|_{m-1}; g_i \rightarrow S_i(\neg \bigvee_{i=1}^n g_i \wedge v(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m-1}; g_i \rightarrow S_i; v}} \vee \bigvee_{i=1}^n g_i))$$

$\Rightarrow$ $\qquad$ 1b

$$\exists m < \omega \ s|_{m-1}; g_i \rightarrow S_i; v(\neg \bigvee_{i=1}^n g_i \wedge (\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m-1}; g_i \rightarrow S_i; v}} \vee \bigvee_{i=1}^n g_i))$$

$\Rightarrow$

$$\exists m < \omega \ s|_{m-1}; g_i \rightarrow S_i; v(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|_{m-1}; g_i \rightarrow S_i; v}}).$$

<u>Infinitely many left movers occur in $s$</u>: Since $nf(s|_m)$ for all $m < \omega$, $s$ contains only left movers. Therefore,

$$\neg \bigvee_{i=1}^n g_i \wedge \exists m < \omega \ s|_m(\mathcal{H}^*_{\kappa', \mathcal{B}^{s|_m}} \vee \bigvee_{i=1}^n g_i)$$

$\Leftrightarrow$

$$\exists m < \omega \ \neg \bigvee_{i=1}^n g_i \wedge s|_m(\mathcal{H}^*_{\kappa', \mathcal{B}^{s|_m}} \vee \bigvee_{i=1}^n g_i)$$

$\Rightarrow$ $\qquad$ 1b

$$\exists m < \omega \ s|_m(\neg \bigvee_{i=1}^n g_i \wedge (\mathcal{H}^*_{\kappa', \mathcal{B}^{s|_m}} \vee \bigvee_{i=1}^n g_i))$$

$\Rightarrow$

$$\exists m < \omega \ s|_m(\mathcal{H}^*_{\kappa', \mathcal{B}^{s|_m}}).$$

<u>Infinitely many right movers occur in $s$</u>: Having considered the above two cases, we can assume that finitely many left movers and $g \rightarrow S_0$ occur in $s$. Since $nf(s|_{m'})$ for all $m' < \omega$, $s$ contains only finitely many $g_i \rightarrow S_i$ also.

Let $m < \omega$ and let $u : seq_\omega\mathcal{B}$ such that $s|_m; u \notin uf''_\alpha$ and $nf(u|_{m'})$ for all $m' < \omega$. We will show that without loss of generality, either $u$ consists of right movers only, or, if 2b holds, contains $g \to S_0$ and right movers, but no $g_i \to S_i$. If $u$ begins with a left mover, $u = L; u'$, then for $P : Pred\mathcal{A}$,

$$s|_m(\exists m' < \omega \ u|_{m'}(P))$$

$\Leftrightarrow$

$$s|_m(\exists m' < \omega \ L; u'|_{m'}(P))$$

$\Leftrightarrow$ 3g (bounded nondeterminism of $L$)

$$s|_m; L(\exists m' < \omega \ u'|_{m'}(P)),$$

so without loss of generality, $u$ does not begin with $L$. Similarly, the bounded nondeterminism of $g_i \to S_i$ means that without loss of generality, $u$ does not begin with $g_i \to S_i$. Thus, $u$ begins with $g \to S_0$ or a right mover.

If $s|_m(\neg \bigvee_{i=1}^n g_i)$, then we can apply similar reasoning to the case of infinitely many left movers above. Otherwise, $s|_m(\bigvee_{i=1}^n g_i)$ holds. If $s|_m$ contains any $g \to S_0$ and 2a holds, then 1a ensures that for some $m' < \omega$, $s|_m; u|'_m(P)$ for any $P : Pred$, in particular the termination condition. If 2b holds, then 2(b)i ensures that without loss of generality, $u$ contains no $g_i \to S_i$.

We apply the argument just made $\nu.\omega$ times, where $\nu$ is given by $\bigvee_{i=1}^n g_i \Rightarrow \mathcal{H}^*_{\nu,\mathcal{D}}$ (see 3e). As a result, the $\nu.\omega$ action sequences either contain at least $\nu$ action sequences in a row containing only right movers – ensuring termination by 3e – or 2(b)iii holds and the $\nu.\omega$ action sequences contain infinitely many $g \to S_0$ but only finitely many $g_i \to S_i$. This latter possibility is excluded by the definition of $uf''_\alpha$.

<u>$\psi(s) : seq_\omega\mathcal{C} \setminus uf'$:</u> We can assume that $\varphi(s)$ is finite.

Let $m < \omega$ and let $u : seq_\omega\mathcal{B}$ such that $s|_m; u \notin uf''_\alpha$ and $nf(u|_{m'})$ for all $m' < \omega$. We can assume without loss of generality that $u : \psi^{-1}(seq_\omega\mathcal{C})$, otherwise we can apply one of the previous three cases. We can also assume that $u$ does not begin with a left mover, otherwise we apply the same argument as in the case of infinitely many right movers. We now show that $u$ does not begin with $g \to S_0$. For a contradiction, suppose not. If $\bigvee_{i=1}^n g_i$ holds at the beginning of $u$, then 2a ensures termination, or if 2b holds, then 2(b)i leads to a contradiction, without loss of generality. If $\neg \bigvee_{i=1}^n g_i$ holds at the beginning of $u$, then we must have $s|_m(\neg \bigvee_{i=1}^n g_i)$, ensuring termination. Therefore, $u$ must begin with $g_i \to S_i$ and may contain $g \to S_0$. If $u$ does contain $g \to S_0$, then $u$ begins with $g_i \to S_i$ and $\neg \bigvee_{i=1}^n g_i$ holds before $g \to S_0$.

We apply the argument just made $\mu.\omega$ times, where $\mu$ is given by

$$\neg \bigvee_{i=1}^n g_i \wedge g \to S(P) \Rightarrow g \to S_0(\mathcal{H}^*_{\mu,\mathcal{C}})$$

(see 1). As a result, the $\mu.\omega$ action sequences either contain at least $\mu$ action sequences in a row containing only $g_i \to S_i$ – ensuring termination by 1 – or the $\mu.\omega$ action sequences contain infinitely many $g \rightarrow S_0$. The action sequence in the latter case refines $\varphi$ applied to the action sequence, this belongs to $seq_\omega \mathcal{A} \setminus uf_\alpha$. $\square$

We now give a small example of the application of the Movers Theorem.

*Example 1.* Let

$$
\begin{array}{llrcll}
\mathcal{A} \ \hat{=} & \textbf{var} & \multicolumn{4}{l}{b, c : \mathbb{B}\,;\, m, n : \mathbb{N}} \\
& \textbf{init} & \multicolumn{4}{l}{c = true} \\
& \textbf{do} & b & \rightarrow & n :\in \mathbb{N}\,;\, b := false & \\
& & [\!] \ \neg b \wedge c \wedge n > 0 & \rightarrow & m, n := 0, n-1 & (X) \\
& \textbf{od}, & & & &
\end{array}
$$

$$
\begin{array}{llrcl}
\mathcal{C} \ \hat{=} & \textbf{var} & \multicolumn{3}{l}{c : \mathbb{B}\,;\, m : \mathbb{N}} \\
& \textbf{do} & \neg c \wedge m > 0 & \rightarrow & m := m - 1 \\
& & [\!] \ \neg c \wedge m = 0 & \rightarrow & c := true \\
& \textbf{od} & & &
\end{array}
$$

and

$$
\begin{array}{llrcll}
\mathcal{B} \ \hat{=} & \textbf{var} & \multicolumn{4}{l}{b, c : \mathbb{B}\,;\, m, n : \mathbb{N}} \\
& \textbf{init} & \multicolumn{4}{l}{c = true} \\
& \textbf{do} & b & \rightarrow & n :\in \mathbb{N}\,;\, b := false & (A) \\
& & [\!] \ \neg b \wedge c \wedge n > 0 & \rightarrow & n, c := n-1, false; m :\in \mathbb{N} & (B) \\
& & [\!] \quad \neg c \wedge m > 0 & \rightarrow & m := m - 1 & (C) \\
& & [\!] \quad \neg c \wedge m = 0 & \rightarrow & c := true & (D) \\
& \textbf{od}. & & & &
\end{array}
$$

The abstract action system $\mathcal{A}$ assigns an arbitrary natural number to $n$, then decrements $n$ until $n = 0$. The variable $m$ is assigned 0. Subsequently in the concrete action system $\mathcal{B}$, every time $n$ is decremented, $m$ is assigned an arbitrary natural number which is then decremented until $m = 0$.

$\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ satisfy the hypothesis of the Movers Theorem with

$$(X) \leq (B); \mathcal{C},$$

so that $\mathcal{A} \leq \mathcal{B}$. In $\mathcal{B}$, statement $(B)$ is $g \rightarrow S_0$, $(C)$ and $(D)$ are the $g_i \rightarrow S_i$, and $(A)$ is a right mover.

Notice that while

$$
\begin{array}{lcl}
\mathcal{H}^*_{0,\mathcal{A}} & \Leftrightarrow & \neg b \wedge (\neg c \vee n = 0) \\
\mathcal{H}^*_{1,\mathcal{A}} & \Leftrightarrow & \neg b \\
\mathcal{H}^*_{2,\mathcal{A}} & \Leftrightarrow & true
\end{array}
$$

and

$$
\begin{array}{lcl}
\mathcal{H}^*_{0,\mathcal{C}} & \Leftrightarrow & \neg c \\
\mathcal{H}^*_{1,\mathcal{C}} & \Leftrightarrow & true,
\end{array}
$$

$$\begin{aligned}
\mathcal{H}^*_{0,\mathcal{B}} &\Leftrightarrow \neg b \wedge c \wedge n = 0 \\
\mathcal{H}^*_{k+1,\mathcal{B}} &\Leftrightarrow \neg b \wedge 0 \le n \le k \quad \text{for} \quad k : \mathbb{N} \\
\mathcal{H}^*_{\omega,\mathcal{B}} &\Leftrightarrow \neg b \\
\mathcal{H}^*_{\omega+1,\mathcal{B}} &\Leftrightarrow \mathit{true}.
\end{aligned}$$

The increase in effort needed to ensure termination of $\mathcal{B}$ compared with $\mathcal{A}$ and $\mathcal{C}$ results from the unbounded nondeterminism of $(B)$, which is itself invoked an unbounded number of times by $(A)$. □

# 5 The Alternating Bit Protocol

This section gives a development of the ABP in the action systems formalism, beginning with the specification that the data from an input channel is transferred instantaneously to an output channel. The first intermediate stage will be an action system where data are transferred from sender to receiver one at a time through a reliable medium. Then acknowledgements and the duplication of sent data are introduced in order to prepare for the loss of data in unreliable media, which is the last refinement step.

## 5.1 Specification

We will begin with the specification of the ABP. First, we define the program variables. Let *in* and *out* be finite the channels from which the sender S obtains data and to which the receiver R outputs data respectively. Then

$$out, in := in, \emptyset$$

is the specification of the ABP. Strictly, this is not an action system. In any case, we can refine this specification to an action system

$$\begin{array}{ll}
\textbf{var} & in, out : seqData \\
\textbf{init} & out := \emptyset \\
\textbf{do} & in \neq \emptyset \;\; \rightarrow \;\; in, out := tl(in), out \frown hd(in) \\
\textbf{od} &
\end{array}$$

which moves data from *in* to *out* one at a time. This refinement step is made via one of the techniques in [Mor94], for example Exercise 5.6. Here *hd* and *tl* are the head and tail respectively of the channels, and $\frown$ denotes catenation.

We will refine this program in subsequent sections. Our aim is to obtain a program which is a true reflection of the situation we are modelling. Our program should be distributed into processes: one for the sender, one for the receiver, and two for the media, with communication between them by (reliable) channels. Figure 1 shows the data flow between processes which we wish to model.

Here, the arrows represent data flow between processes. $S$ is the sender, $R$ the receiver, and $M_S$ and $M_R$ are the unreliable media. The labels on the arrows between processes are channels, all other variables of the program being local to one of the four processes.

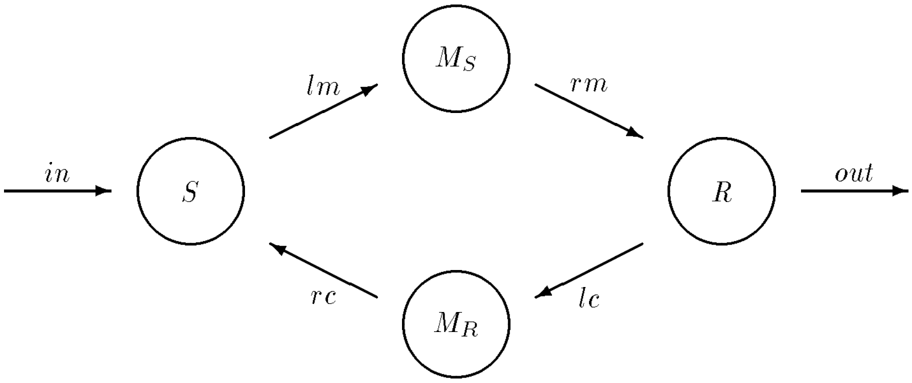


Figure 1: Data flow between processes

The development of the ABP consists of little more than a series of judicious applications of the Movers Theorem. Unless otherwise stated, all of the $h_j \rightarrow T_j$ are right movers. It is possible to reverse the order of some of the development steps. The order we have chosen seems to be the most succinct, however.

### 5.2 Sender and Receiver

In order to separate sending and receiving processes, the first development step is to add a buffer between sender and receiver, namely $rm$. First, in Figure 2, we add $rm$ as a new variable, noting that $rm = \emptyset$ is an invariant of the action system.

$$
\begin{array}{ll}
\textbf{var} & in, rm, out : seqData \\
\textbf{init} & rm, out := \emptyset, \emptyset \\
\textbf{do} & in \neq \emptyset \wedge rm = \emptyset \quad \rightarrow \quad in, out := tl(in), out \frown hd(in) \\
\textbf{od} &
\end{array}
$$

Figure 2: Introduction of $rm$

Next, we use the Movers Theorem to incorporate $rm$ as a buffer between the sender and the receiver (Figure 3). With the definitions

$$
\begin{array}{rcl}
g \rightarrow S & \widehat{=} & in \neq \emptyset \wedge rm = \emptyset \rightarrow in, out := tl(in), out \frown hd(in) \\
g \rightarrow S_0 & \widehat{=} & in \neq \emptyset \wedge rm = \emptyset \rightarrow in, rm := tl(in), rm \frown hd(in) \\
g_1 \rightarrow S_1 & \widehat{=} & rm \neq \emptyset \rightarrow rm, out := \emptyset, out \frown hd(rm),
\end{array}
$$

the hypotheses of the Movers Theorem are easy to verify.

$$
\begin{array}{lll}
\textbf{var} & \multicolumn{2}{l}{in, rm, out : seqData} \\
\textbf{init} & \multicolumn{2}{l}{rm, out := \emptyset, \emptyset} \\
\textbf{do} & in \neq \emptyset \wedge rm = \emptyset \rightarrow & in, rm := tl(in), rm \frown hd(in) \\
 & [\!]\quad rm \neq \emptyset \rightarrow & rm, out := \emptyset, out \frown hd(rm) \\
\textbf{od} & &
\end{array}
$$

Figure 3: $rm$ as a buffer between sender and receiver

### 5.3 Duplication of Data

The sending of each datum only once will not be sufficient when unreliable media are introduced, because that one datum may be lost. We now introduce the duplication of data. The receiver needs to be able to distinguish, however, between the same datum which the sender has sent repeatedly and the same datum occurring consecutively in the channel $in$. We introduce the boolean variable $tag$ in order to tag the data, so that consecutive data from $in$ have different tags. The introduction of $tag$, and the duplication of data is a simple data refinement, with datum $x$ in $rm$ being replaced by the data $(x, tag)^{\mathbb{N}\setminus\{0\}}$. Here, the superscript $\mathbb{N} \setminus \{0\}$ denotes the duplication of $(x, tag)$ $n$ times, where $n \in \mathbb{N} \setminus \{0\}$ is chosen nondeterministically. The result is shown in Figure 4, with $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$. Notice that all copies of the datum are transferred from the

$$
\begin{array}{lll}
\textbf{var} & \multicolumn{2}{l}{in, rm, out : seqData; tag : \mathbb{B}} \\
\textbf{init} & \multicolumn{2}{l}{rm, out, tag := \emptyset, \emptyset, 0} \\
\textbf{do} & in \neq \emptyset \wedge rm = \emptyset \rightarrow & in, rm, tag := tl(in), rm \frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
 & [\!]\quad rm \neq \emptyset \rightarrow & rm, out := \emptyset, out \frown \pi_1(hd(rm)) \\
\textbf{od} & &
\end{array}
$$

Figure 4: Multiple copies of data sent at the same time

sender to the receiver at the same time. In Section 5.5, we will ensure that the transfer of data occurs one at a time.

### 5.4 Acknowledgements

We now place another buffer, $rc$, between the sender and the receiver. Just as $rm$ was introduced as a buffer for the data being sent, so too $rc$ is a buffer for the acknowledgements.

But first, we introduce the variable $s\_tag$ (see Figure 5). This variable stores the last acknowledgement received by the sender. When it is compared with $tag$, the tag of the last datum sent, the sender can determine whether or not the receiver has received the last datum sent, and thus whether or not the sender should send the next datum. This condition is expressed by the guard of the first statement of Figure 5, which is obtained from Figure 4 via the invariant $rm = \emptyset \Leftrightarrow tag = s\_tag$. As a result, the sender no longer requires a knowledge of the state of $rm$.

$$
\begin{array}{lrcl}
\textbf{var} & \multicolumn{3}{l}{in, rm, out : seqData; tag, s\_tag : \mathbb{B}} \\
\textbf{init} & \multicolumn{3}{l}{rm, out, tag, s\_tag := \emptyset, \emptyset, 0, 0} \\
\textbf{do} & in \neq \emptyset \wedge tag = s\_tag & \rightarrow & in, rm, tag := tl(in), rm \frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
 & [\,]\quad rm \neq \emptyset & \rightarrow & rm, out, s\_tag := \emptyset, out \frown \pi_1(hd(rm)), \pi_2(hd(rm)) \\
\textbf{od} & & &
\end{array}
$$

Figure 5: Addition of $s\_tag$

We can now add the buffer $rc$ for acknowledgements between receiver and sender using the Movers Theorem. We define

$$
\begin{array}{rcl}
g \rightarrow S & \widehat{=} & rm \neq \emptyset \rightarrow rm, out, s\_tag := \emptyset, out \frown \pi_1(hd(rm)), \pi_2(hd(rm)) \\
g \rightarrow S_0 & \widehat{=} & rm \neq \emptyset \rightarrow rm, out, rc := \emptyset, out \frown \pi_1(hd(rm)), \pi_2(hd(rm)), rc \frown \pi_2(rm) \\
g_1 \rightarrow S_1 & \widehat{=} & rc \neq \emptyset \rightarrow rc, s\_tag := \emptyset, hd(rc).
\end{array}
$$

The hypothesis of the Movers Theorem can be verified, with 1, 2a and 3d following from the invariant

$$rc \neq \emptyset \Rightarrow rm = \emptyset \wedge hd(rc) = tag \neq s\_tag,$$

giving Figure 6. Note that now, the sender (executing the first two statements) and receiver

$$
\begin{array}{lrcl}
\textbf{var} & \multicolumn{3}{l}{in, rm, rc, out : seqData; tag, s\_tag : \mathbb{B}} \\
\textbf{init} & \multicolumn{3}{l}{rm, rc, out, tag, s\_tag := \emptyset, \emptyset, \emptyset, 0, 0} \\
\textbf{do} & rc \neq \emptyset & \rightarrow & rc, s\_tag := \emptyset, hd(rc) \\
 & [\,]\ in \neq \emptyset \wedge tag = s\_tag & \rightarrow & in, rm, tag := tl(in), rm \frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
 & [\,]\quad rm \neq \emptyset & \rightarrow & rm, out, rc := \emptyset, out \frown \pi_1(hd(rm)), rc \frown \pi_2(rm) \\
\textbf{od} & & &
\end{array}
$$

Figure 6: Introduction of the acknowledgements buffer $rc$

(executing the third statement) share only the variables $rm$ and $rc$. This condition will not be violated until the media are introduced explicitly.

### 5.5 Restriction of Channel Operations

In Figure 6, multiple copies of data are added to and removed from the channels $rm$ and $rc$ at the same time. In this section we will restrict the channel operations so that data are added to or removed from a channel one-by-one. We will impose this condition first on data being removed from $rc$, then on data being transferred from $rm$ to $rc$, and finally on data being added to $rm$.

To ensure that data are removed from $rc$ one at time, we use the Movers Theorem with

$$\begin{array}{rcl} g \rightarrow S & \widehat{=} & rc \neq \emptyset \wedge s\_tag \neq hd(rc) \rightarrow rc, s\_tag := \emptyset, hd(rc) \\ g \rightarrow S_0 & \widehat{=} & rc \neq \emptyset \wedge s\_tag \neq hd(rc) \rightarrow rc, s\_tag := tl(rc), hd(rc) \\ g_1 \rightarrow S_1 & \widehat{=} & rc \neq \emptyset \wedge s\_tag = hd(rc) \rightarrow rc, s\_tag := tl(rc), hd(rc) \end{array}$$

and obtain Figure 7. The guard of the first statement in Figure 6 can be changed to

$$\begin{array}{lrcl} \textbf{var} & \multicolumn{3}{l}{in, rm, rc, out : seqData; tag, s\_tag : \mathbb{B}} \\ \textbf{init} & \multicolumn{3}{l}{rm, rc, out, tag, s\_tag := \emptyset, \emptyset, \emptyset, 0, 0} \\ \textbf{do} & rc \neq \emptyset \wedge s\_tag \neq hd(rc) & \rightarrow & rc, s\_tag := tl(rc), hd(rc) \\ & [\!]\ rc \neq \emptyset \wedge s\_tag = hd(rc) & \rightarrow & rc, s\_tag := tl(rc), hd(rc) \\ & [\!]\ in \neq \emptyset \wedge tag = s\_tag & \rightarrow & in, rm, tag := tl(in), rm \frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\ & [\!]\ rm \neq \emptyset & \rightarrow & rm, out, rc := \emptyset, out \frown \pi_1(hd(rm)), rc \frown \pi_2(rm) \\ \textbf{od} & & & \end{array}$$

Figure 7: Removal of data from $rc$ one at a time

$rc \neq \emptyset \wedge s\_tag \neq hd(rc)$ (so that 2a holds) because

$$rc \neq \emptyset \Rightarrow s\_tag \neq hd(rc)$$

is an invariant of the action system. The condition 1 holds because

$$\forall n : \mathbb{N}\ \ rc \neq \emptyset \Rightarrow hd(rc) = hd(tl^n(rc))$$

is an invariant, while the invariant

$$rc = \emptyset \wedge rm \neq \emptyset \Rightarrow s\_tag \neq \pi_2(hd(rm))$$

helps to establish 1b and 3d. The other conditions are easy to verify.

To prepare for the second step, that of transferring data from $rm$ to $rc$ one-by-one, we need to introduce another variable, $r\_tag$. This variable stores the tag of the datum that the receiver last received. By comparing it with the tag currently being received, the receiver is able to determine whether or not the current datum has been received previously, and therefore whether or not to output the datum to the channel $out$.

We obtain Figure 8 by three small steps: first we combine the first two statements of Figure 7 to obtain the first statement of Figure 8 by the Monotonicity Theorem. Next, we add $r\_tag$ to the body of the last statement of Figure 7 Finally, we add the conjunct $\pi_2(hd(rm)) \neq r\_tag$ to the last statement of the resulting action system because $rm \neq \emptyset \Rightarrow \pi_2(hd(rm)) \neq r\_tag$ is an invariant of that action system. This invariant is proved by

$$
\begin{array}{lrcl}
\textbf{var} & \multicolumn{3}{l}{in, rm, rc, out : seqData; tag, r\_tag, s\_tag : \mathbb{B}} \\
\textbf{init} & \multicolumn{3}{l}{rm, rc, out, tag, r\_tag, s\_tag := \emptyset, \emptyset, \emptyset, 0, 0, 0} \\
\textbf{do} & rc \neq \emptyset & \rightarrow & rc, s\_tag := tl(rc), hd(rc) \\
 & [\,]\quad in \neq \emptyset \wedge tag = s\_tag & \rightarrow & in, rm, tag := \\
 & & & \qquad tl(in), rm ^\frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
 & [\,]\ rm \neq \emptyset \wedge \pi_2(hd(rm)) \neq r\_tag & \rightarrow & rm, out, rc, r\_tag := \\
 & & & \emptyset, out ^\frown \pi_1(hd(rm)), rc ^\frown \pi_2(rm), \pi_2(hd(rm)) \\
\textbf{od} & & &
\end{array}
$$

Figure 8: Addition of $r\_tag$

induction on the size of $rm$ and $rc$ because the preservation of the invariant in the first statement requires information about the contents of $rc$, which in turn requires information about $rm$.

For the second step in this section, transferring data from $rm$ to $rc$ one-by-one, we use the Movers Theorem again with

$$
\begin{array}{rcl}
g \rightarrow S & \widehat{=} & rm \neq \emptyset \wedge r\_tag \neq \pi_2(hd(rm)) \rightarrow rm, out, rc, r\_tag := \\
 & & \qquad \emptyset, out ^\frown \pi_1(hd(rm)), rc ^\frown \pi_2(rm), \pi_2(hd(rm)) \\
g \rightarrow S_0 & \widehat{=} & rm \neq \emptyset \wedge r\_tag \neq \pi_2(hd(rm)) \rightarrow out, r\_tag := out ^\frown \pi_1(hd(rm)), \pi_2(hd(rm)) \\
g_1 \rightarrow S_1 & \widehat{=} & rm \neq \emptyset \wedge r\_tag = \pi_2(hd(rm)) \rightarrow rm, rc := tl(rm), rc ^\frown \pi_2(hd(rm))
\end{array}
$$

to obtain Figure 9. The conditions 1b and 3d follow from the invariant

$$
\begin{array}{lrcl}
\textbf{var} & \multicolumn{3}{l}{in, rm, rc, out : seqData; tag, r\_tag, s\_tag : \mathbb{B}} \\
\textbf{init} & \multicolumn{3}{l}{rm, rc, out, tag, r\_tag, s\_tag := \emptyset, \emptyset, \emptyset, 0, 0, 0} \\
\textbf{do} & rc \neq \emptyset & \rightarrow & rc, s\_tag := tl(rc), hd(rc) \\
 & [\,]\quad in \neq \emptyset \wedge tag = s\_tag & \rightarrow & in, rm, tag := \\
 & & & \qquad tl(in), rm ^\frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
 & [\,]\ rm \neq \emptyset \wedge \pi_2(hd(rm)) \neq r\_tag & \rightarrow & out, r\_tag := out ^\frown \pi_1(hd(rm)), \pi_2(hd(rm)) \\
 & [\,]\ rm \neq \emptyset \wedge \pi_2(hd(rm)) = r\_tag & \rightarrow & rm, rc := tl(rm), rc ^\frown \pi_2(hd(rm)) \\
\textbf{od} & & &
\end{array}
$$

Figure 9: Data transferred from $rm$ to $rc$ one-by-one

$$rm = \emptyset \wedge tag = s\_tag \Rightarrow tag = r\_tag,$$

while 1 is proved from the invariant

$$\forall n : \mathbb{N}\ \ rc \neq \emptyset \Rightarrow hd(rc) = hd(tl^n(rc)).$$

Both of these invariants require induction on the sizes of $rm$ and $rc$.

Finally in this section, we ensure that data are added to $rm$ one-by-one. Before we apply the Movers Theorem again, we introduce the variable $datum$ which stores both the datum last sent and its tag $tag$. This makes $tag$ redundant, so we use data refinement with the invariant $\pi_2(datum) = tag$. We need to store the datum last sent for the sender to send to $rm$ the next time. Having introduced $datum$, we apply the Movers Theorem with

$$
\begin{array}{rcl}
g \to S & \hat{=} & in \neq \emptyset \wedge tag = s\_tag \to in, rm, tag := tl(in), rm \frown (hd(in), \neg tag)^{\mathbb{N}\setminus\{0\}}, \neg tag \\
g \to S_0 & \hat{=} & in \neq \emptyset \wedge \pi_2(datum) = s\_tag \to in, datum := tl(in), (hd(in), \neg \pi_2(datum)) \\
g_1 \to S_1 & \hat{=} & \pi_2(datum) \neq s\_tag \to rm := rm \frown datum
\end{array}
$$

and $\mathcal{C}$ having the unfair action sequence $\{g_1 \to S_1^\omega\}$ to obtain Figure 10. We need the

$$
\begin{array}{llrcll}
\textbf{var} & \multicolumn{5}{l}{in, rm, rc, out : seqData; r\_tag, s\_tag : \mathbb{B}; datum : Data \times \mathbb{B}} \\
\textbf{init} & \multicolumn{5}{l}{rm, rc, out, r\_tag, s\_tag, datum := \emptyset, \emptyset, \emptyset, 0, 0, (:\in Data, 0)} \\
\textbf{do} & & rc \neq \emptyset & \to & rc, s\_tag := tl(rc), hd(rc) & \\
 & [] & in \neq \emptyset \wedge \pi_2(datum) = s\_tag & \to & in, datum := tl(in), (hd(in), \neg \pi_2(datum)) & \\
 & [] & \pi_2(datum) \neq s\_tag & \to & rm := rm \frown datum & (3) \\
 & [] & rm \neq \emptyset \wedge \pi_2(hd(rm)) \neq r\_tag & \to & out, r\_tag := out \frown \pi_1(hd(rm)), \pi_2(hd(rm)) & \\
 & [] & rm \neq \emptyset \wedge \pi_2(hd(rm)) = r\_tag & \to & rm, rc := tl(rm), rc \frown \pi_2(hd(rm)) & \\
 & \multicolumn{5}{l}{\textbf{uf } true : \Box\Diamond(3)} \\
\textbf{od} & & & & &
\end{array}
$$

Figure 10: Data added to $rm$ one-by-one

invariant

$$rc \neq \emptyset \wedge tag = s\_tag \Rightarrow tag = hd(rc),$$

which again is proved by induction on the sizes of $rm$ and $rc$.

Notice that for the first time, we have an action system with a fairness constraint. This reflects the necessity of preventing the sender from sending copies of data via action (3) to the exclusion of other actions, thereby preventing progress.

### 5.6 Unreliable Media

The last part of our development is to introduce the unreliable media. First we will introduce the unreliable medium for the data being sent ($M_S$), then for the acknowledgements ($M_R$).

To simulate the medium $M_S$ implicitly, we add the statement

$$\pi_2(datum) \neq s\_tag \to skip \quad (A)$$

to the action system in Figure 10 using the Adding Skips Theorem. This action simulates the failed attempt to transmit the datum $datum$, while action (3) simulates a successful transmission. In applying the Adding Skips Theorem, the unfair sequences become

$$\Box\Diamond(3)\ \cup\ \Diamond\Box(A).$$

In order to obtain an action system in a suitable form for the application of the Movers Theorem, we use the Monotonicity Theorem to replace the actions (3) and (A) by the single action

$$\pi_2(datum) \neq s\_tag \rightarrow rm := rm ^\frown datum \ [\!]\ skip \quad (3'),$$

and the unfair sequences by $\Box\Diamond(3')$.

We now apply the variant of the Movers Theorem where 2b holds, the only time we use this variant in our development of the ABP, with

$$\begin{array}{rcl} g \rightarrow S & \hat{=} & \pi_2(datum) \neq s\_tag \rightarrow rm := rm ^\frown datum \ [\!]\ skip \\ g \rightarrow S_0 & \hat{=} & \pi_2(datum) \neq s\_tag \rightarrow lm := lm ^\frown datum \\ g_1 \rightarrow S_1 & \hat{=} & lm \neq \emptyset \rightarrow lm, rm := tl(lm), rm ^\frown hd(rm) \ [\!]\ lm := tl(lm). \end{array}$$

This yields Figure 11. We have introduced the channel $lm$ between the sender and $M_S$,

$$\begin{array}{llrcll}
\textbf{var} & \multicolumn{5}{l}{in, lm, rm, rc, out : seqData; r_tag, s_tag : \mathbb{B}; datum : Data \times \mathbb{B}} \\
\textbf{init} & \multicolumn{5}{l}{lm, rm, rc, out, r_tag, s_tag, datum := \emptyset, \emptyset, \emptyset, \emptyset, 0, 0, (:\in Data, 0)} \\
\textbf{do} & & rc \neq \emptyset & \rightarrow & rc, s_tag := tl(rc), hd(rc) & \\
& [\!] & in \neq \emptyset \wedge \pi_2(datum) = s_tag & \rightarrow & in, datum := tl(in), (hd(in), \neg\pi_2(datum)) & \\
& [\!] & \pi_2(datum) \neq s_tag & \rightarrow & lm := lm ^\frown datum & (3) \\
& [\!] & lm \neq \emptyset & \rightarrow & lm, rm := tl(lm), rm ^\frown hd(rm) & \\
& & & & [\!]\ lm := tl(lm) & (4) \\
& [\!] & rm \neq \emptyset \wedge \pi_2(hd(rm)) \neq r_tag & \rightarrow & out, r_tag := out ^\frown \pi_1(hd(rm)), \pi_2(hd(rm)) & (5) \\
& [\!] & rm \neq \emptyset \wedge \pi_2(hd(rm)) = r_tag & \rightarrow & rm, rc := tl(rm), rc ^\frown \pi_2(hd(rm)) & (6) \\
& \multicolumn{5}{l}{\textbf{uf}\ \ true : \Box\Diamond\{(3), (4)\}} \\
\textbf{od} & & & & &
\end{array}$$

Figure 11: Introduction of the channel $lm$

thus modelling the transmission and loss of data explicitly via action (4). Our final step, before introducing $M_R$, is to split the action (4) into the two actions

$$\begin{array}{lcll} lm \neq \emptyset & \rightarrow & lm, rm := tl(lm), rm ^\frown hd(rm) & (4A) \\ lm \neq \emptyset & \rightarrow & lm := tl(lm) & (4B) \end{array}$$

using the Monotonicity Theorem. The unfair sequences become

$$\Box\Diamond\{(3), (4A), (4B)\},$$

but this can be reduced to $\Box\Diamond(3)$ since infinitely many $(4A)$s or $(4B)$s but only finitely many (3)s results in termination.

The introduction of the unreliable medium $M_R$ for acknowledgements between the receiver and sender involves introducing the channel $lc$, just as $lm$ was introduced for $M_S$. This step is similar and yields the final implementation of the ABP, as is shown in Figure 12. The statements executed by the sender are (1), (2) and (3), those by $M_S$ (4)

$$
\begin{array}{llcll}
\textbf{var} & \multicolumn{4}{l}{in, lm, rm, lc, rc, out : seqData; s\_tag, r\_tag : \mathbb{B}; datum : Data \times \mathbb{B}} \\
\textbf{init} & \multicolumn{4}{l}{lm, rm, lc, rc, out, s\_tag, r\_tag, datum := \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, 0, 0, (:\in Data, 0)} \\
\textbf{do} & rc \neq \emptyset & \rightarrow & rc, s\_tag := tl(rc), hd(rc) & (1) \\
[] & in \neq \emptyset \wedge \pi_2(datum) = s\_tag & \rightarrow & in, datum := tl(in), (hd(in), \neg\pi_2(datum)) & (2) \\
[] & \pi_2(datum) \neq s\_tag & \rightarrow & lm := lm \frown datum & (3) \\
[] & lm \neq \emptyset & \rightarrow & lm, rm := tl(lm), rm \frown hd(lm) & (4) \\
[] & lm \neq \emptyset & \rightarrow & lm := tl(lm) & (5) \\
[] & rm \neq \emptyset \wedge \pi_2(hd(rm)) \neq r\_tag & \rightarrow & out, r\_tag := out \frown \pi_1(hd(rm)), \pi_2(hd(rm)) & (6) \\
[] & rm \neq \emptyset \wedge \pi_2(hd(rm)) = r\_tag & \rightarrow & rm, lc := tl(rm), lc \frown \pi_2(hd(rm)) & (7) \\
[] & lc \neq \emptyset & \rightarrow & lc, rc := tl(lc), rc \frown hd(lc) & (8) \\
[] & lc \neq \emptyset & \rightarrow & lc := tl(lc) & (9) \\
\textbf{uf} & \multicolumn{4}{l}{true : \Box\Diamond(3)} \\
\textbf{od}. & & & &
\end{array}
$$

Figure 12: Final result

and (5), those by the receiver (6) and (7), and those executed by $M_R$ are (8) and (9). The only variables shared between the four processes are the channels $lm$, $rm$, $lc$ and $rc$, each of which is shared between exactly two processes.

It is possible to make the channels $lm$, $rm$, $lc$ and $rc$ bounded in size (of at least one), which would involve strengthening the guards of (3), (4), (7) and (8) via Theorem 5. In the case of (3), for example, the conjunct $size(lm) < max\_size(lm)$ would be added to the guard. The disjunction of the guards of the action system is not changed because

$$\pi_2(datum) \neq s\_tag \wedge size(lm) = max\_size(lm) \Rightarrow lm \neq \emptyset,$$

the guard of (3) and (4). Similar arguments apply for the other actions, (4), (7) and (8).

## 6 Summary

In this paper, we have illustrated the theory of fair action systems through the development of the Alternating Bit Protocol.

In order to facilitate program development, we have generalised Back's Movers Theorem to fair action systems exhibiting unbounded nondeterminism. This theorem is relatively simple to use in practice as they have conditions which are easy to verify, and they allow for modularity in the development in the sense that they change only small parts of the syntax of an action system.

## References


[Bac89] R.J.R. Back, *A Method of Refining Atomicity in Parallel Algorithms*, LNCS 366, Springer-Verlag, 1989

[Boo82] H.J. Boom, *A Weaker Precondition for Loops*, ACM Transactions on Programming Languages and Systems, Vol.4, 1982

[CM88] K.M. Chandy, J. Misra, *Parallel Program Design: A Foundation*, Addison-Wesley, 1988

[Col95] I.M. Collier, *Program Development in Action Systems*, D.Phil. Thesis, Oxford University Computing Laboratory, 1995

[Dij76] E.W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, 1976

[DS90] E.W. Dijkstra, C.S. Scholten, *Predicate Calculus and Program Semantics*, Springer-Verlag, 1990

[Lam90] L. Lamport, *A Theorem on Atomicity in Distributed Algorithms*, Distributed Computing, Vol. 4, 1990

[Lam91] L. Lamport, *The Temporal Logic of Actions*, DEC SRC Report 79, 1991

[Lip75] R.J. Lipton, *Reduction: A Method of Proving Properties of Parallel Programs*, Communications of the ACM, Vol. 18, 1975

[Mil80] R. Milner, *A Calculus of Communicating Systems*, LNCS 92, Springer Verlag, 1980

[Mor94] C.C. Morgan, *Programming from Specifications*, 2nd Edition, Prentice-Hall, 1994

[PS91] K. Paliwoda, J.W. Sanders, *An Incremental Specification of the Sliding-Window Protocol*, Distributed Computing, Vol. 5, 1991

[Sne95] J.L.A. van de Snepscheut, *The Sliding Window Protocol Revisited*, Formal Aspects of Computing, Vol.7, 1995

[Wab96a] A.K. Wabenhorst, *Developing Fairness in Terminating and Reactive Programs*, Technical Report TR-1-96, Programming Research Group, Oxford University Computing Laboratory, 1996

[Wab96b] A.K Wabenhorst, *On Fairness in Terminating and Reactive Systems*, D.Phil. Thesis, Oxford University Computing Laboratory, 1996