Programming Research Group

DEVELOPING FAIRNESS IN TERMINATING AND REACTIVE PROGRAMS

Axel Wabenhorst

PRG-TR-1-96

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Developing Fairness in Terminating and Reactive Programs

Axel Wabenhorst[*]

January 1996

Abstract

This paper presents a theory of fairness within the framework of Dijkstra's weakest preconditions and Back's action systems. It is general enough to deal with all kinds of fairness, not just strong and weak fairness. A definition of fairness and refinement to deal with non-terminating programs is also obtained. For terminating and non-terminating programs with and without fairness, theorems about program development such as monotonicity are derived and illustrated by means of examples, principally the readers-writers problem.

1 Introduction

1.1 Comparisons

By addressing fairness we are able to develop, entirely within the formalism, designs whose infinitary behaviour previously required *ad hoc* reasoning (for example, realistic forms of the readers-writers problem). Fairness is a generic term and so is difficult to define formally; thus any such attempt must incorporate an appeal to intuition which is necessarily more tenuous than, say, for the semantics of iteration. Despite the absence of an obvious single simple definition of fairness, our definition is consistent with the intuition we have about many concrete examples. Intuitively, fairness guarantees that in the repeated nondeterministic choice between statements in a program, none of them is neglected indefinitely. There are many different kinds of fairness, depending on the decision one makes about what constitutes the "neglect" of a statement. For example, weak fairness guarantees that a statement whose guard is true from a certain time onwards is executed infinitely often, and

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

strong fairness guarantees that a statement whose guard is true infinitely often is executed infinitely often. Our definition extends those formalised in [Fra86], the results in which are anyway restricted to the verification of sequential programs, largely termination (see also [MP92]). Our general definition of fairness enables us to achieve implementations for a wide variety of fairness constraints which give different priorities to different processes.

Action systems [Bac89], are a useful formalism because of their expressive power and developmental clarity. In particular, they allow the derivation of syntactic refinement laws. Informally, an action system consists of variable declarations and an initialisation followed by an iteration statement. As we will see in Section 4, fairness constraints can be expressed succinctly using them. In Section 5, we will derive analogous results for program development to those in [Bac89] incorporating fairness. For example, we develop a theorem about stepwise refinement of programs through monotonicity. One disadvantage of action systems is their rigid structure: for example, expressing the sequential composition of two actions requires the introduction of a boolean clocking variable. Another disadvantage is their lack of support for modularisation.

Other possible formalisms include UNITY [CM88], CSP [Hoa85] and Z [Spi89]. Z is a descriptive specification language whose strength lies in its modularisation. The schema calculus, which describes both state and operation, allows very succinct, high-level specifications. This has its negative side though, because in order to deduce which sequential compositions of operations are allowed, it is necessary to calculate preconditions (which are left implicit). Data refinement is well-handled through modularisation, and preserves the type of input and output. But this preservation of input and output means that the original specification is in a sense low-level. Another deficiency of Z is its lack of combinators for parallel composition. Some modification of Z is necessary for action refinement, but the results we will derive for action systems can be translated to apply to Z.

UNITY specifications are given as a set of safety and liveness properties, which are then downcoded to programs that look similar to action systems. UNITY has a built-in fairness assumption, that each statement is executed infinitely often. The main disadvantage of UNITY is that it does not cater for stepwise refinement: the proof of correctness of each high-level program must be made directly from the specification in the logic, and there is no modularisation. The resulting low level of UNITY programs leads to complicated proofs.

CSP is a process algebra with laws involving equality of programs rather than refinement. Communication is handled by synchronous message passing, while refinement is by semantics (failures and divergences) rather than syntax. CSP is useful in proving that two programs are refinement equivalent, but to show strict refinement is difficult: a modification of CSP is required [Xu92]. This is a major drawback because almost all of the refinements we will deal with are strict. We deal with strict refinements because the method must be able to construct a concrete deterministic implementation, eventually, from a more high-level nondeterministic one, and refinement equality (in particular the laws of CSP) preserve nondeterminism.

Throughout the paper, we use the predicate transformer semantics of the language of guarded commmands as developed by Dijkstra [Dij76] and extended in [Nel89, Mor87, Bac88, Mor89, Boo82, DS90]. These semantics and the resulting definition of refinement, used in the context of action systems, are concise and calculational. We will also define refinement for non-terminating (or reactive) action systems using predicate transformer semantics, taking into account the intermediate states of the program. Subsequently, we will show that there are analogous definitions and theorems to those for terminating action systems, both unfair and fair.

One alternative approach to our use of predicate transformer semantics would be to specify an action system by its alphabet, finite behaviours and fair infinite behaviours or, equivalently, its alphabet, safety conditions and liveness conditions [Col95]. This requires extra reasoning to show the correspondence between an action system and its specification, which may be difficult to do rigorously. In any case, the resulting trace-inclusion definition of refinement is much more restrictive about when an action system refines another than in our approach.

Related work, with similar aims, is due to [BX95] and [Mor90]. In [BX95], forward and backward simulation are used to show refinement between fair action systems. The conditions under which refinement take place are easy to check, and the proof rules are similar to our combination of monotonicity, adding skips and data refinement in Sections 5 and 6. However, the applicability of the results in [BX95] are limited to strong and weak fairness. In addition, it is assumed that the actions of the refining action system either refine *skip* or correspond to actions in the action system to be refined.

Morris [Mor90] gives a treatment of fairness using fixed-point theorems. The calculation of weakest preconditions sometimes involves two nested recursions, but the proof rules for termination are derived succinctly using fixed points. In addition, the results hold in a more general framework than action systems, as the syntactic shape of action systems is not exploited. However, the type of fairness catered for is restricted to strong or weak fair choice between statements.

Probabilistic semantics of sequential programs can be thought of as providing a strong form of fairness on the nondeterministic choice beteen two program statements. A framework for developing probabilistic programs (rather than systems) can be found, for example, in [MMSS95a].

The results in this paper will form part of my D.Phil. thesis, which I expect to complete in the second half of 1996.

1.2 Outline

We begin in Section 2 by recalling the definition of predicate transformers, action systems and their elementary properties. Predicate transformers are then used to recall the definition of refinement and data refinement. Section 3 introduces action sequences and some of their properties. In particular, we use them to give a characterisation of the semantics of iteration which will provide the inspiration for our semantics of fairness. In Section 4, we motivate and illustrate our definition of fairness by simple examples. From this definition, we prove some theorems useful in the development of terminating programs from simple specifications in Section 5. In Section 6, we define refinement of non-terminating action systems in terms of refinement of terminating action systems. This approach allows us to state definitions of fairness and data refinement as well as theorems for program development analogous to those for terminating action systems. The whole theory developed in this paper is used in Section 7 in the extensive development of various solutions to the readers-writers problem.

2 Background

We use the following notation: Ord is the class of ordinals; X^κ denotes the type of all functions from $\kappa : Ord$ (seen as the set of ordinals less than κ) to X. Other notation will be defined as needed.

2.1 Predicate Transformers

Let $\Sigma_\mathcal{A}$ denote the set of states of a set of variables. These will be the variables of our 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* [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.

We define statements in terms of weakest preconditions rather than vice-versa as most authors have done, because

- monotonicity and conjunctivity are the only conditions which our statements need to satisfy for our results to hold; and

- our approach is more general in the sense that we allow statements other than the standard.

We use the notation $S(P)$ rather than $wp.S.P$ which some other authors use [Hes92, BM95] because our notation is more succinct and sequential composition corresponds to functional composition. Moreover, we will not need to differentiate the weakest precondition from other predicate transformers in the rest of this paper. The *weakest errorfree precondition* and *weakest liberal precondition* will be introduced in (Section 3.4), where it will be shown that the former can be expressed in terms of the weakest precondition of action sequences, while the latter will only be referred to in comparison with the work of other authors. Since we will not use the weakest liberal precondition in the development of our results, our "healthiness conditions" are different to those of [Hes92], where the weakest liberal precondition is used as often as the weakest precondition.

Often, our two axioms are combined into the equivalent single axiom $\forall \alpha \in I\ S(P_\alpha) \Leftrightarrow S(\forall \alpha \in I\ P_\alpha)$, but we prefer to keep them separate in order to be explicit in our reasoning.

Examples of statements and their associated weakest precondition transformers are shown in Figure 1. The assertion statements $!b$ skip if b holds, otherwise abort. They can be used, for example, to delay consideration of exceptional circumstances (such as unexpected errors) during program development by simply aborting in those circumstances [Mor88].

The weakest precondition transformers in Figure 1 can be shown to satisfy monotonicity and conjunctivity, using structural induction where necessary (see, for example, [Hes92]).

Conjunctivity, called *positive conjunctivity* in [Hes92], is sometimes weakened to what is called *finite conjunctivity* in [Hes92]: for any $P, Q : Pred$

- *skip*: $skip(P) \;\widehat{=}\; P$
- guard: $(?b)(P) \;\widehat{=}\; \neg b \vee P$
- *miracle*: $miracle \;\widehat{=}\; ?false$
- assertion: $(!b)(P) \;\widehat{=}\; b \wedge P$
- *abort*: $abort \;\widehat{=}\; !false$
- assignment:

  $$(x_1, \ldots, x_n :\in X_1, \ldots, X_n)(P) \;\widehat{=}\; \bigwedge_{i=1}^{n} \forall y_i \in X_i \; P[y_1, \ldots, y_n / x_1, \ldots, x_n],$$

  where $P[y_1, \ldots, y_n / x_1, \ldots, x_n]$ is the predicate P with y_i replacing all occurrences of x_i. If X is a singleton, we write $x := X$ instead of $x :\in X$.
- sequential composition: $(S_1; S_2)(P) \;\widehat{=}\; S_1 \circ S_2(P)$
- choice: $(S_1 \,[]\, S_2)(P) \;\widehat{=}\; S_1(P) \wedge S_2(P)$
- actions: $g \rightarrow S \;\widehat{=}\; ?g; S$. Here, g is called the *guard* of the action.
- alternation: $(\textbf{if} \;\; []_{i=1}^{n} \, g_i \rightarrow S_i \; \textbf{fi})(P) \;\widehat{=}\; \bigvee_{i=1}^{n} g_i \wedge ([]_{i=1}^{n} \, g_i \rightarrow S_i)(P)$
- iteration: $(\textbf{do} \;\; []_{i=1}^{n} \, g_i \rightarrow S_i \; \textbf{od})(P) \;\widehat{=}\; \exists \kappa : Ord \; H_\kappa(P)$ [Boo82], where

  $$\begin{aligned} H_0(P) &\;\widehat{=}\; P \wedge \neg \bigvee_{i=1}^{n} g_i, \\ H_{\kappa+1}(P) &\;\widehat{=}\; H_0(P) \vee (\textbf{if} \;\; []_{i=1}^{n} \, g_i \rightarrow S_i \; \textbf{fi})\,(H_\kappa(P)) \text{ and} \\ H_\delta(P) &\;\widehat{=}\; \exists \kappa < \delta \; H_\kappa(P), \text{ where } \delta \text{ is a limit ordinal.} \end{aligned}$$

Figure 1: Some statements and their weakest preconditions

$S(P) \wedge S(Q) \Rightarrow S(P \wedge Q)$. In most cases, we will need positive conjunctivity rather than finite conjunctivity.

The weakest precondition also satisfies

$$S(P) \vee S(Q) \Rightarrow S(P \vee Q) \;\; (or - continuity),$$

a property which follows from monotonicity. However, we do *not* have the converse $S(P \vee Q) \Rightarrow S(P) \vee S(Q)$ because of nondeterminism: taking

$$S \;=\; \mathbf{if}\; true \rightarrow n := 1 \;[]\; true \rightarrow n := 2 \;\mathbf{fi}$$

we have $S(n = 1 \vee n = 2) \Leftrightarrow true$, but $S(n = 1) \Leftrightarrow false \Leftrightarrow S(n = 2)$.

Some authors have insisted that condition $S(false) \Leftrightarrow false$ (strictness) should be satisfied by all statements. This does not hold for statements involving guards $?b$, although it does hold for alternation statements. We allow non-strict statements because they will not restrict our results.

It can be shown that if all statements S_i in the iteration

$$\mathbf{do}\;\; []_{i=1}^{n}\, g_i \rightarrow S_i \;\mathbf{od}$$

satisfy upper continuity [Hes92], which is equivalent to bounded nondeterminism, then $(\mathbf{do}\;\; []_{i=1}^{n}\, g_i \rightarrow S_i \;\mathbf{od})(P) \Leftrightarrow H_\omega(P)$. This was Dijkstra's initial definition of iteration [Dij76]. For a statement S_i in Figure 1 to satisfy upper continuity, it suffices to show that any assignments in S_i satisfy upper continuity.

It is straightforward to show by induction that the termination condition of the iteration, $(\mathbf{do}\;\; []_{i=1}^{n}\, g_i \rightarrow S_i \;\mathbf{od})(true)$, is equivalent to

$$(\mathbf{do}\;\; []_{i=1}^{n}\, g_i \rightarrow S_i \;\mathbf{od})(\neg \bigvee_{i=1}^{n} g_i).$$

The semantics of iteration also have a fixed-point characterisation. We define fixed points as follows: For an action system $\mathcal{A}$ and a predicate transformer $f : Pred\mathcal{A} \rightarrow Pred\mathcal{A}$, we say that $P : Pred\mathcal{A}$ is a *least fixed point* [*greatest fixed point*] of f (notation: $P \Leftrightarrow (\mu X : f(X))$ [$P \Leftrightarrow (\eta X : f(X))$]) if

- $P \Leftrightarrow f(P)$; and
- if $Q \Leftrightarrow f(Q)$, then $P \Rightarrow Q$ [$Q \Rightarrow P$].

The fixed-point characterisation of iteration [DS90] is

$$(\mathbf{do}\;\; []_{i=1}^{n}\, g_i \rightarrow S_i \;\mathbf{od})(P) \Leftrightarrow (\mu X : (P \vee \bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n}\, g_i \rightarrow S_i)(X)).$$

2.2 Refinement

We now define refinement and data refinement and discuss the lattice property endowed on the set of predicate transformers.

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. □

Note that

- refinement is transitive.
- for an action $g \rightarrow S$ to be refined by an action $g' \rightarrow S'$, it is sufficient (but not necessary) that $g \Leftarrow g'$ and $S \leq S'$.

The set of predicate transformers forms a complete boolean lattice [BvW89] with partial order given by $f \leq f'$ defined by $\forall P : Pred\ f(P) \Rightarrow f'(P)$, which coincides with the refinement relation when the predicate transformer is a weakest precondition. The meet $\wedge$ corresponds to choice $[]$ [BvW89]. The bottom element of the lattice is *abort*, while the top element is *miracle*.

The following definition can be found in [GM91]:

Definition 2 Data Refinement. Let $\vec{a}$ be a set of abstract variables in a statement S which we would like to replace by a set of concrete variables $\vec{c}$ to form a statement T. Let φ be a predicate transformer from predicates on $\vec{a}$ to predicates on $\vec{c}$. We say that S is *data-refined* by T (notation $S \leq_\varphi T$) if $\varphi \circ S \leq T \circ \varphi$.

Note that if φ is the identity function, this definition reduces to that of refinement.

Throughout this paper, we will assume that φ satisfies the following properties:

- $(\forall \vec{a}\ P \Rightarrow Q) \Rightarrow (\forall \vec{c}\ \varphi(P) \Rightarrow \varphi(Q))$ (monotonicity)
- $\varphi(\exists \alpha \in I\ P_\alpha) \Leftrightarrow \exists \alpha \in I\ \varphi(P_\alpha)$ (or-distributivity).

With these two conditions, the following holds:

Theorem 3. *Data refinement $\leq_\varphi$ distributes through the components of a statement.*

Proof. See [GM91]. □

2.3 Action Systems

Informally, an action system [Bac89] consists of variable declarations and an initialisation followed by an iteration statement. A nondeterministic choice is made between the actions whose guards are true, and 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. As we will see in Section 7, it is sometimes possible to achieve greater concurrency via a data refinement which duplicates variables, each copy being used locally.

We now give a formal definition of an action system.

Definition 4 Action System. An *action system* has the form

$$\begin{array}{lll} \textbf{var} & & x_1, \ldots, x_m \\ \textbf{init} & & x_1, \ldots, x_m :\in X_1, \ldots, X_m \\ \textbf{do} & []_{i=1}^{n} & A_i \;\; \textbf{od}, \end{array}$$

where $x_1, \ldots, x_m$ are typed variables, $m : \mathbb{N} \setminus \{0\}$ and the A_i are actions. $\square$

In the definition, the statement $x_1, \ldots, x_m :\in X_1, \ldots, X_m$ is called the *initialisation* of the action system. It will be omitted if $X_1, \ldots, X_n$ are the same as the types of the variables $x_1, \ldots, x_n$. Sometimes, we will omit the variable declaration if this does not lead to ambiguity. For the purposes of calculating weakest preconditions, we regard the initialisation as a statement in sequential composition with the following iteration and assume that the guards of the actions are calculated instantaneously.

3 Action Sequences

Informally, an action sequence is a sequence (either finite or infinite) of actions in an action system. In later sections, we will see that action sequences are useful for expressing fairness properties: while all finite action sequences are fair, certain infinite ones may be unfair.

3.1 Introduction

Let

$$\mathcal{A} \;\hat{=}\; \textbf{do} \;\; []_{i=1}^{n} \;\; A_i \;\; \textbf{od}$$

be an action system. For $\kappa \leq \omega$, $seq_\kappa \mathcal{A} \;\hat{=}\; \{A_i \mid 1 \leq i \leq n\}^\kappa$. For $s : seq_\kappa \mathcal{A}$, if $\kappa' < \kappa$, then $s|_{\kappa'}$ is defined by $s|_{\kappa'}(\alpha) = s(\alpha)$ for $\alpha < \kappa'$. (Recall that we can characterise an ordinal by the set of ordinals less than it.) We say that $s' : seq_{\kappa'}\mathcal{A}$ is a *prefix* of $s : seq_\kappa\mathcal{A}$ (notation: $s' \sqsubseteq s$) if $s' = s|_{\kappa'}$. We also define $seq\mathcal{A} \ \widehat{=}\ \bigcup_{\kappa<\omega} seq_\kappa$ and $\overline{seq}\mathcal{A} \ \widehat{=}\ seq\mathcal{A} \cup seq_\omega\mathcal{A}$, the latter are the *action sequences* of $\mathcal{A}$. Note that $seq_\kappa\mathcal{A}$ $(\kappa < \omega)$, $seq\mathcal{A}$ and $seq_\omega\mathcal{A}$ have cardinalities n^k, $\aleph_0$ and $2^{\aleph_0}$ respectively.

We define the predicate transformer semantics of finite action sequences as follows. Let $P : Pred\mathcal{A}$. Then

- if $s : seq_0\mathcal{A}$, $s(P) \ \widehat{=}\ P$,
- if $\kappa < \omega$ and $s : seq_{\kappa+1}\mathcal{A}$, $s(P) \ \widehat{=}\ s|_\kappa; s(\kappa)(P)$.

We will sometimes abuse notation, so that when action sequences s and t satisfy $s(P) \Leftrightarrow t(P)$ for all $P : Pred\mathcal{A}$, we will say that $s = t$. Thus, we will identify $seq_0\mathcal{A}$ with $\{skip\}$. (Note that $seq_0\mathcal{A}$ is a singleton.) Also, we will identify $seq_\kappa\mathcal{A}$ with the sequential composition of actions of length κ, $\kappa < \omega$. We will also use ; to denote catenation within infinite action sequences. It should be emphasised that in this context, ; is not sequential composition.

If $s, u : seq$ such that $s \sqsubseteq u$, then there exists $t : seq$ such that $s; t = u$. We write $u - s \ \widehat{=}\ t$.

With our definition of action sequences, we permit impossible juxtapositions of actions. The result is a type more general than that which contains just those sequences of actions of $\mathcal{A}$ which actually occur, which we exploit in the results to follow.

For example,

$$\mathcal{B} \ \widehat{=}\ \quad \textbf{do} \quad g \rightarrow g := false \quad (A) \ \ \textbf{od},$$

has "real" action sequences $skip$ and A, but seq also contains $A; A$, $A; A; A$ and so on. In general, these "unreal" action sequences are weaker than the "real" ones (see Theorem 7 in Subsection 3.2), with the result that the "unreal" action sequences are irrelevant when taking conjunctions of all action sequences such as in Lemma 8. This property can be useful when making calculations, as it permits us to treat all action sequences uniformly, rather than forcing us to consider two classes.

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.

We can now specify program invariants using the definitions of this subsection: we call $P : Pred\mathcal{A}$ an *invariant* of an action system $\mathcal{A}$ if

$$(init(P) \Leftrightarrow true) \ \wedge \ (\forall s : seq\mathcal{A}(P \Rightarrow s(P)) \Leftrightarrow true),$$

where *init* is the initialisation of $\mathcal{A}$.

3.2 Basic Properties

The action sequences of

$$\mathcal{A} \ \widehat{=} \quad \mathbf{do} \quad []_{i=1}^{n} \ \ g_i \rightarrow S_i \ \ \mathbf{od}$$

have some simple properties which we will now derive.

Proposition 5 Monotonicity and Conjunctivity.

- *If $s : seq\mathcal{A}$ and $P, Q : Pred\mathcal{A}$ such that $P \Rightarrow Q$, then $s(P) \Rightarrow s(Q)$;*
- *If $s : seq\mathcal{A}$, $I \neq \emptyset$ and $P_\alpha : Pred\mathcal{A}$ for all $\alpha \in I$, then*

$$\forall \alpha \in I \ s(P_\alpha) \Rightarrow s(\forall \alpha \in I \ P_\alpha).$$

Proof. By induction on the length of s, using the monotonicity and conjunctivity of the individual actions and of sequential composition. □

In particular,

Corollary 6. *If $s, t_1, t_2, u : seq\mathcal{A}$ and $t_1 \leq t_2$, then $s; t_1; u \leq s; t_2; u$.* □

This shows that finite action sequences but not infinite action sequences can be used as statements within programs.

If execution of an action sequence results in the termination of the action system, then any extension of this action sequence also results in termination of the action system. Similarly, if an action sequence does not abort, then no prefix of it aborts either.

Theorem 7. *Let $s, u : seq\mathcal{A}$ such that $s \sqsubseteq u$. Then*

- $s(\neg \bigvee_{i=1}^{n} g_i) \Rightarrow u(\neg \bigvee_{i=1}^{n} g_i)$;
- *if $s \neq u$ and $P : Pred\mathcal{A}$, then $s(\neg \bigvee_{i=1}^{n} g_i) \Rightarrow u(P)$;*
- $s(true) \Leftarrow u(true)$.

Proof. There exists an action sequence t such that $s; t = u$. If $t = skip$ then the theorem holds trivially, so assume that for some j, $t = g_j \rightarrow S_j; t'$. Then

$$
\begin{array}{ll}
 & s(\neg \bigvee_{i=1}^{n} g_i) \\
\Rightarrow & \text{monotonicity of action sequences applied to } \neg \bigvee_{i=1}^{n} g_i \ \Rightarrow\ \neg g_j \vee S_j; t'(P) \\
 & s(\neg g_j \vee S_j; t'(P)) \\
\Leftrightarrow & \text{definition of actions and sequential composition} \\
 & s; g_j \rightarrow S_j; t'(P) \\
\Leftrightarrow & s; g_j \rightarrow S_j; t' = u \\
 & u(P)
\end{array}
$$

and

$$
\begin{array}{ll}
 & u(true) \\
\Leftrightarrow & \\
 & s; t(true) \\
\Leftrightarrow & \text{definition of sequential composition} \\
 & s(t(true)) \\
\Rightarrow & \text{applying monotonicity of action sequences to } t(true) \Rightarrow true \\
 & s(true).
\end{array}
$$

□

3.3 A Characterisation of Iteration

The following characterisation of the weakest precondition of an action system, stated in [BM95], can be thought of as decomposing the weakest precondition of iteration. The first conjunct is a condition on the finite action sequences which states that if $\mathcal{A}$ terminates at all, it terminates in state P, while the second conjunct is a restriction on the infinite action sequences which states that $\mathcal{A}$ terminates eventually.

It is useful to isolate the conditions on the finite and infinite action sequences when we consider fairness, because we shall see that fairness is a condition on the infinite action sequences rather than the finite ones.

Lemma 8. *Let*

$$\mathcal{A} \ \hat{=}\ \ \mathbf{do}\ \ []_{i=1}^{n}\ \ g_i \rightarrow S_i\ \ \mathbf{od}.$$

Then for all $P : Pred\mathcal{A}$*,*

$$H_\kappa(P) \Leftrightarrow (\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i)) \wedge H_\kappa(\neg \bigvee_{i=1}^{n} g_i).$$

Proof. Base step:

$H_0(P)$

$\Leftrightarrow$ definition of $H_0(P)$

$P \wedge \neg \bigvee_{i=1}^n g_i$

$\Leftrightarrow$

$(P \vee \bigvee_{i=1}^n g_i) \wedge \neg \bigvee_{i=1}^n g_i$

$\Leftrightarrow$ $\neg \bigvee_{i=1}^n g_i \Rightarrow g_i \rightarrow S_i(Q)$ for any $Q : Pred\mathcal{A}$

$skip(P \vee \bigvee_{i=1}^n g_i) \wedge \forall s : seq\mathcal{A} \setminus \{skip\}\ s(P \vee \bigvee_{i=1}^n g_i) \wedge skip(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$ definition of seq_0

$(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i)) \wedge H_0(\neg \bigvee_{i=1}^n g_i)$.

Successor ordinals:

$H_{\kappa+1}(P)$

$\Leftrightarrow$ definition of $H_{\kappa+1}(P)$

$H_0(P) \vee (\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})(H_\kappa(P))$

$\Leftrightarrow$ induction hypothesis and monotonicity of weakest preconditions

$H_0(P) \vee (\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})((\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i)) \wedge H_\kappa(\neg \bigvee_{i=1}^n g_i))$

$\Leftrightarrow$ for $\Rightarrow$ monotonicity, for $\Leftarrow$ conjunctivity

$H_0(P) \vee ((\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i))$
$\wedge(\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})(H_\kappa(\neg \bigvee_{i=1}^n g_i)))$

$\Leftrightarrow$

$(H_0(P) \vee (\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})((\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i))$
$\wedge\, (H_0(P) \vee (\textbf{if}\ \ []_{i=1}^n\ g_i \rightarrow S_i\ \textbf{fi})(H_\kappa(\neg \bigvee_{i=1}^n g_i)))$

$\Leftrightarrow$

$((P \wedge \neg \bigvee_{i=1}^n g_i) \vee ((\bigvee_{i=1}^n g_i) \wedge \bigwedge_{i=1}^n (g_i \rightarrow S_i(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i))))$
$\wedge H_{\kappa+1}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$(P \vee \bigvee_{i=1}^n g_i) \wedge \bigwedge_{i=1}^n (g_i \rightarrow S_i(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^n g_i))) \wedge H_{\kappa+1}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$ for $\Rightarrow$ monotonicity, for $\Leftarrow$ conjunctivity

$(P \vee \bigvee_{i=1}^n g_i) \wedge \forall s : seq\mathcal{A} \ \bigwedge_{i=1}^n (g_i \rightarrow S_i(s(P \vee \bigvee_{i=1}^n g_i))) \wedge H_{\kappa+1}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$skip(P \vee \bigvee_{i=1}^n g_i) \wedge \forall s : seq\mathcal{A} \setminus \{skip\} \ s(P \vee \bigvee_{i=1}^n g_i) \wedge H_{\kappa+1}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^n g_i) \wedge H_{\kappa+1}(\neg \bigvee_{i=1}^n g_i)$.

Limit ordinals:

$H_\delta(P)$

$\Leftrightarrow$ definition of $H_\delta(P)$

$\exists \kappa < \delta \ H_\kappa(P)$

$\Leftrightarrow$ induction hypothesis and monotonicity of weakest preconditions

$\exists \kappa < \delta \ (\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^n g_i)) \wedge H_\kappa(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$(\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^n g_i)) \wedge (\exists \kappa < \delta \ H_\kappa(\neg \bigvee_{i=1}^n g_i))$

$\Leftrightarrow$ definition of $H_\delta(\neg \bigvee_{i=1}^n g_i)$

$(\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^n g_i)) \wedge H_\delta(\neg \bigvee_{i=1}^n g_i)$.

$\square$

Theorem 9.

$$\mathcal{A}(P) \Leftrightarrow (\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^n g_i)) \wedge \mathcal{A}(\neg \bigvee_{i=1}^n g_i)$$

Proof. Take the disjunction over κ of Lemma 8. $\square$

Lemma 10.

$$H_{\kappa+\omega}(\neg \bigvee_{i=1}^n g_i) \Leftrightarrow \forall s : seq_\omega\mathcal{A} \ \exists m < \omega \ s|_m(H_\kappa(\neg \bigvee_{i=1}^n g_i))$$

Proof. We show by induction that for all $m < \omega$,

$$H_{\kappa+m}(\neg \bigvee_{i=1}^n g_i) \Leftrightarrow \forall s : seq_m\mathcal{A} \ s(H_\kappa(\neg \bigvee_{i=1}^n g_i)).$$

The case $m = 0$ is trivial since $seq_0\mathcal{A} = \{skip\}$. For the induction step,

$H_{\kappa+m+1}(\neg \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$ definition of $H_{\kappa+m+1}$

$H_0(\neg \bigvee_{i=1}^{n} g_i) \vee (\bigvee_{i=1}^{n} g_i \wedge (\textbf{if } \|_{i=1}^{n} g_i \rightarrow S_i \textbf{ fi})(H_{\kappa+m}(\neg \bigvee_{i=1}^{n} g_i)))$

$\Leftrightarrow$ induction hypothesis and monotonicity

$(\neg \bigvee_{i=1}^{n} g_i) \vee ((\bigvee_{i=1}^{n} g_i) \wedge \bigwedge_{i=1}^{n}(g_i \rightarrow S_i(\forall s : seq_m\mathcal{A}\ s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i)))))$

$\Leftrightarrow$ for $\Rightarrow$ monotonicity, for $\Leftarrow$ conjunctivity

$(\neg \bigvee_{i=1}^{n} g_i) \vee ((\bigvee_{i=1}^{n} g_i) \wedge \forall s : seq_m\mathcal{A}\ \bigwedge_{i=1}^{n} g_i \rightarrow S_i; s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i)))$

$\Leftrightarrow$

$(\neg \bigvee_{i=1}^{n} g_i) \vee ((\bigvee_{i=1}^{n} g_i) \wedge \forall s : seq_{m+1}\mathcal{A}\ s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i)))$

$\Leftrightarrow$

$\forall s : seq_{m+1}\mathcal{A}\ s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$.

For the moment, give $\{g_i \rightarrow S_i \mid 1 \leq i \leq n\}$ the discrete topology and $seq_\omega\mathcal{A}$ the induced product topology. (For a treatment of the topological concepts in this paragraph, see [Men75]). The sets $\{s|_m; seq_\omega\mathcal{A} \mid s : seq_\omega\mathcal{A}\}$ form an open cover of the (by Tychonov's Theorem) compact set $seq_\omega\mathcal{A}$, so they have a finite subcover indexed by s_i and m_i. Take $k \mathrel{\widehat{=}} \max_i m_i$. We claim that for all $s : seq_\omega\mathcal{A}$, $s|_k(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$ holds. It suffices to show that for $s : seq\mathcal{A}$, $s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i)) \Rightarrow s; g_i \rightarrow S_i(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$. This follows from

$s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$

$\Rightarrow$ $H_\kappa(P) \Rightarrow H_{\kappa+1}(P)$ and monotonicity of action sequences

$s(H_{\kappa+1}(\neg \bigvee_{i=1}^{n} g_i))$

$\Leftrightarrow$ definition of $H_{\kappa+1}$

$s(H_0(\neg \bigvee_{i=1}^{n} g_i) \vee ((\bigvee_{i=1}^{n} g_i) \wedge \bigwedge_{i=1}^{n} g_i \rightarrow S_i(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))))$

$\Rightarrow$

$s((\neg \bigvee_{i=1}^{n} g_i) \vee (g_i \rightarrow S_i(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))))$

$\Rightarrow$

$s; g_i \rightarrow S_i(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$.

Therefore,

$H_{\kappa+\omega}(\neg \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$

$\exists m < \omega \; H_{\kappa+m}(\neg \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$ above

$\exists m < \omega \; \forall s : seq_m\mathcal{A} \; s(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$

$\Leftrightarrow$

$\exists m < \omega \; \forall s : seq_\omega\mathcal{A} \; s|_m(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$

$\Leftrightarrow$ $\Rightarrow$ is trivial, for $\Leftarrow$ see above

$\forall s : seq_\omega\mathcal{A} \; \exists m < \omega \; s|_m(H_\kappa(\neg \bigvee_{i=1}^{n} g_i))$.

$\square$

We now come to the main theorem of this section. We will use it later as our inspiration for the definition of fair action systems.

Theorem 11. *Let $\mathcal{A}$ be an action system. Then*

$$\mathcal{A}(P) \;\Leftrightarrow\; \forall s : seq\mathcal{A} \; s(P \vee \bigvee_{i=1}^{n} g_i) \;\wedge\; \mathcal{A}(\neg \bigvee_{i=1}^{n} g_i)$$

with

$$\mathcal{A}(\neg \bigvee_{i=1}^{n} g_i) \;\Leftrightarrow\; \exists \kappa \; \mathcal{H}_\kappa^*,$$

where

$$\begin{array}{rcl} \mathcal{H}_0^* & \widehat{=} & \neg \bigvee_{i=1}^{n} g_i \\ \mathcal{H}_{\kappa+1}^* & \widehat{=} & \forall s : seq_\omega\mathcal{A} \; \exists m < \omega \; s|_m(\mathcal{H}_\kappa^*) \\ \mathcal{H}_\delta^* & \widehat{=} & \exists \kappa < \delta \; \mathcal{H}_\kappa^*. \end{array}$$

Proof. The first part of the theorem follows from Theorem 9. We now show by induction on κ that

$$\mathcal{H}_\kappa^* \Leftrightarrow H_{\omega.\kappa}(\neg \bigvee_{i=1}^{n} g_i).$$

The case $\kappa = 0$ is trivial, as both sides evaluate to $\neg \bigvee_{i=1}^{n} g_i$. For the successor step,

$\mathcal{H}_{\kappa+1}^*$

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

$\forall s : seq_\omega \mathcal{A}\ \exists m < \omega\ s|_m(\mathcal{H}^*_\kappa)$

$\Leftrightarrow$ induction hypothesis and monotonicity

$\forall s : seq_\omega \mathcal{A}\ \exists m < \omega\ s|_m(H_{\omega.\kappa}(\neg \bigvee_{i=1}^n g_i))$

$\Leftrightarrow$ Lemma 10

$H_{\omega.\kappa+\omega}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$H_{\omega.(\kappa+1)}(\neg \bigvee_{i=1}^n g_i)$,

while for the limit step

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

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

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

$\Leftrightarrow$ induction hypothesis

$\exists \kappa < \delta\ H_{\omega.\kappa}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$\exists \kappa < \omega.\delta\ H_\kappa(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$ definition of $H_{\omega.\delta}(\neg \bigvee_{i=1}^n g_i)$

$H_{\omega.\delta}(\neg \bigvee_{i=1}^n g_i)$.

Therefore,

$\mathcal{A}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$ definition of $\mathcal{A}(\neg \bigvee_{i=1}^n g_i)$

$\exists \kappa\ H_\kappa(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$

$\exists \kappa\ H_{\omega.\kappa}(\neg \bigvee_{i=1}^n g_i)$

$\Leftrightarrow$ above

$\exists \kappa\ \mathcal{H}^*_\kappa$.

$\square$

3.4 Weakest Errorfree Preconditions

The *weakest liberal precondition* was introduced in [Dij76], and defined only recently to be

$$wlp.\mathcal{A}.P \;\widehat{=}\; (\eta X : (P \vee \bigvee_{i=1}^{n} g_i) \wedge wlp.\,[]_{i=1}^{n}\, g_i \rightarrow S_i(X))$$

for an action system $\mathcal{A}$ [Hes92, MMSS95b, MM95]. Informally, $wlp.S.P$ is the weakest predicate on the variable states such that subsequent execution of S either does not terminate or else guarantees termination in a state satisfying P; and is thus weaker than the weakest precondition.

One of the healthiness conditions

$$\mathcal{A}(P) \;\Leftrightarrow\; wlp.\mathcal{A}.P \wedge \mathcal{A}(true)$$

of [Dij76] and [Hes92] strongly resembles Theorem 9. In fact the two would be the same, bearing in mind that $\mathcal{A}(true) \Leftrightarrow \mathcal{A}(\neg \bigvee_{i=1}^{n} g_i)$, if we had $wlp.\mathcal{A}.P$ equivalent to $\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i)$. However this is not the case. Nevertheless, $\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i)$ is something similar: the weakest errorfree precondition (see also [Hes95]). Informally, it is the weakest predicate on states at which execution of $\mathcal{A}$ guarantees either termination in a state satisfying P or infinite execution of the loop of $\mathcal{A}$.

Definition 12 Weakest Errorfree Precondition. The *weakest errorfree precondition of an action system $\mathcal{A}$ with respect to $P : Pred\mathcal{A}$ is*

$$wep.\mathcal{A}.P \;\widehat{=}\; \forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i).$$

We will see now that the weakest errorfree precondition also has a fixed-point characterisation.

Proposition 13. *Let $\mathcal{A}$ be an action system. Then*

$$wep.\mathcal{A}.P \;\Leftrightarrow\; (\eta X : (P \vee \bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n}\, g_i \rightarrow S_i)(X)) \tag{1}$$

Proof. For the first condition in the definition of greatest fixed point,

$$(P \vee \textstyle\bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n}\, g_i \rightarrow S_i)(\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i))$$

$\Leftrightarrow$ monotonicity ($\Rightarrow$) and conjunctivity ($\Leftarrow$)

$$(P \vee \textstyle\bigvee_{i=1}^{n} g_i) \wedge \forall s : seq\mathcal{A}\ ([]_{i=1}^{n}\, g_i \rightarrow S_i)(s(P \vee \bigvee_{i=1}^{n} g_i))$$

$\Leftrightarrow$ definitions of alternation and sequential composition

$(P \vee \bigvee_{i=1}^{n} g_i) \wedge \forall s : seq\mathcal{A} \ \ \bigwedge_{i=1}^{n} g_i \rightarrow S_i; s(P \vee \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$

$skip(P \vee \bigvee_{i=1}^{n} g_i) \wedge \forall s : seq\mathcal{A} \setminus \{skip\} \ s(P \vee \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$

$\forall s : seq\mathcal{A} \ s(P \vee \bigvee_{i=1}^{n} g_i)$.

For the second condition, suppose that

$$Q \ \Leftrightarrow \ (P \vee \bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n} \ g_i \rightarrow S_i)(Q).$$

We now show that $Q \Rightarrow \forall s : seq_k\mathcal{A} \ s(P \vee \bigvee_{i=1}^{n} g_i)$ by induction on $k : \mathbb{N}$.
<u>Base step:</u>

Q

$\Leftrightarrow$ hypothesis

$(P \vee \bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n} \ g_i \rightarrow S_i)(Q)$

$\Rightarrow$

$(P \vee \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$

$\forall s : seq_0 \ s(P \vee \bigvee_{i=1}^{n} g_i)$.

<u>Induction step:</u>

Q

$\Leftrightarrow$ hypothesis

$(P \vee \bigvee_{i=1}^{n} g_i) \wedge ([]_{i=1}^{n} \ g_i \rightarrow S_i)(Q)$

$\Rightarrow$

$\bigwedge_{i=1}^{n}(g_i \rightarrow S_i)(Q)$

$\Rightarrow$ induction hypothesis

$\bigwedge_{i=1}^{n}(g_i \rightarrow S_i)(\forall s : seq_{k-1}\mathcal{A} \ s(P \vee \bigvee_{i=1}^{n} g_i))$

$\Rightarrow$ monotonicity

$\forall s : seq_{k-1}\mathcal{A} \ \ \bigwedge_{i=1}^{n} g_i \rightarrow S_i; s(P \vee \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$

$\forall s : seq_k\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i)$.

$\square$

Comparing this characterisation of the weakest errorfree precondition with the definition of the weakest liberal precondition, we see that the only difference is that while the weakest liberal precondition allows non-termination of all kinds, including aborts of statements within the action system, the only non-termination allowed in the weakest errorfree precondition is the non-termination of the iteration loop. Thus, the weakest errorfree precondition is stronger than the weakest liberal precondition, but weaker than the weakest precondition (Theorem 9).

Formula 1 can also be expressed in the notation of [Mor90]:

$$wep.\mathcal{A}.P \Leftrightarrow A_S.(P \vee \bigvee_{i=1}^{n} g_i, P),$$

where $S \mathrel{\widehat{=}} []_{i=1}^{n}\ g_i \rightarrow S_i$.

4 Fairness

In this section, we define fair terminating action systems using our results on action sequences from Section 3. For this we use weakest precondition semantics, which permit us to use existing definitions of refinement and data refinement. It generalises several forms of fairness and so permits them to be treated in a uniform manner.

4.1 Introduction

Before motivating our definition of fairness, we consider an example which encapsulates much of our intuition about it.

Example 1. If the action system

$$\begin{array}{llllll} \mathcal{A} \mathrel{\widehat{=}} & \textbf{var} & b : \mathbb{B} & & & \\ & \textbf{do} & b & \rightarrow & b := false & (A0) \\ & [] & b & \rightarrow & skip & (A1) \\ & \textbf{od}, & & & & \end{array}$$

executes in accordance with the standard definition of iteration, it is not guaranteed to terminate because action $(A0)$ may never execute. If, on the other hand, we force it to execute fairly in the sense that action $(A0)$ is not always neglected, the action system is guaranteed to terminate. Our definition should allow us to specify that $(A0)$ should not be neglected indefinitely, yielding $\mathcal{A}(true) \Leftrightarrow true$.

Now consider the action system

$$\begin{array}{rllllll} \mathcal{B} \;\widehat{=} & \textbf{var} & & n : \mathbb{N} & & & \\ & \textbf{do} & & n > 0 & \rightarrow & n := n - 1 & (B0) \\ & & [] & n > 0 & \rightarrow & skip & (B1) \\ & \textbf{od}, & & & & & \end{array}$$

which is similar to $\mathcal{A}$. The major difference is that while only one execution of statement $(A0)$ is necessary to guarantee termination in $\mathcal{A}$, multiple executions of $(B0)$ are necessary to ensure termination in $\mathcal{B}$, the necessary number of executions of $(B0)$ being dependent on the initial value of n. In fact, no fixed finite number of executions of $(B0)$ can guarantee termination for all initial values of n. Termination in $\mathcal{B}$ is guaranteed only if $\mathcal{B}$ exhibits weak fairness, which in this case is the same as strong fairness. Our definition of fair action systems should allow us to specify that $\mathcal{B}$ executes in accordance with weak fairness or, for that matter, any other kind of fairness.

The action system

$$\begin{array}{rllllll} \mathcal{C} \;\widehat{=} & \textbf{var} & & b : \mathbb{B}\,;\, n : \mathbb{N} & & & \\ & \textbf{init} & & b = true & & & \\ & \textbf{do} & & b & \rightarrow & n :\in \mathbb{N}\,;\, b := false & (C0) \\ & & [] & \neg b \wedge n > 0 & \rightarrow & n := n - 1 & (C1) \\ & & [] & \neg b \wedge n > 0 & \rightarrow & skip & (C2) \\ & \textbf{od} & & & & & \end{array}$$

is similar to $\mathcal{B}$, except that the value of n is fixed after entering the loop rather than before. The variable b is a clocking variable which ensures that n is initialised to an arbitrary natural number before the other two statements, which decrease n and skip, are executed. Unlike $\mathcal{B}$, $\mathcal{C}$ exhibits unbounded nondeterminism: statement $(C0)$ has an unbounded number of possible outcomes. As a result, no (finite) number of executions of $(C1)$ can guarantee termination of the action system. We would like to be able to specify that $\mathcal{C}$ executes in accordance with weak fairness, and that $\mathcal{C}$ terminates in this case. $\square$

As another example, in the readers-writers problem co-ordinated access to a single file is required by many processes. These processes may either read or write with the constraint that if a process writes to the file, no other process may access it. In order to implement solutions, one needs some fairness assumptions to ensure that all processes have equal access. Our general definition of fairness should enable us to achieve implementations for a variety of fairness constraints which give different priorities to different processes: in Section 7, we will see that this is the case.

4.2 Motivation

We now motivate some of the details of our definition by considering various simple fair systems.

We shall constrain an action system to behave "fairly" by specifying those (infinite) unfair sequences which do not occur. We choose to specify the unfair sequences rather than their complement the fair ones, because we find it more convenient to specify the former.

Thus, when considering the termination of a fair action system, we insist that fair infinite action sequences must terminate; in other words, a finite prefix of each such action sequence must terminate. We make no such demand on unfair infinite action sequences, because they do not occur. We then calculate the weakest precondition incorporating this consideration.

Example 2. To ensure a fair execution of the action system $\mathcal{A}$ in Example 1, we take $A1^\omega$ to be the only unfair action sequence.

For the action system $\mathcal{B}$ in Example 1, we take the unfair action sequences to be $\{s; B1^\omega \mid s : seq\mathcal{B}\}$. These are exactly the infinite sequences which do not execute statement $(B0)$ from a certain point onwards, despite the guard of $(B0)$ being true.

Similarly, we take the unfair action sequences in $\mathcal{C}$ to be

$$\{C0; s; C2^\omega \mid s : seq\mathcal{C} \text{ contains no } (C0)\}.$$

□

The only restriction which we place on the unfair sequences is that they be infinite. Different kinds of fairness are captured by different kinds of unfair sequences. In addition to weak and strong fairness, we can, for example, model the tossing of a fair coin by specifying that the fair sequences are exactly those for which heads and tails occur with probability half. Unfortunately, the different sets of unfair sequences have no obvious common properties. It might be thought, for example, that the set of unfair sequences should be non-empty or non-full. However, we will see that if we specify the set of unfair sequences to be empty, we will obtain Boom's definition of iteration. Also, if we specify *all* infinite sequences to be unfair, we obtain the weakest errorfree precondition of iteration (see Subsection 4.3). This demonstrates the wide applicability of our generic definition and, as a result, we place no further restrictions on the unfair sequences.

The following example demonstrates that we need the flexibility to be able to specify different unfair action sequences for different initial conditions.

Example 3. We claim that intuitively, a fair execution of the action system

$$\begin{array}{llrcll} \mathcal{A} \;\widehat{=} & \textbf{var} & m : \mathbb{Z} & & & \\ & \textbf{do} & m > 0 & \rightarrow & m := 0 & (A0) \\ & & [\!]\;\; m \neq 0 & \rightarrow & \mathit{skip} & (A1) \\ & \textbf{od} & & & & \end{array}$$

should guarantee termination exactly under the initial condition $m \geq 0$. For this to be true, will show that we need to specify different unfair action sequences for different initial conditions.

Under the initial condition $m > 0$, the guards of both $(A0)$ and $(A1)$ are true and will continue to be true until action $(A0)$ is executed. If action $(A0)$ is always neglected, however, $\mathcal{A}$ fails to terminate. Thus, to ensure a fair execution of $\mathcal{A}$, which will guarantee termination, we need to specify $A1^{\omega}$ as an unfair sequence under the initial condition $m > 0$.

Under the initial condition $m = 0$, both guards are $false$ and so $\mathcal{A}$ terminates immediately. Thus, we do not need to specify any unfair action sequences although it does no harm to do so.

Under the initial condition $m < 0$, only the guard of $(A1)$ is true, so that the action sequence $A1^{\omega}$ is the only possible execution. We are not neglecting $(A0)$ because its guard is never enabled, so there are no unfair action sequences for initial condition $m < 0$.

We conclude that if $\mathcal{A}$ is to execute fairly in the sense we have described, guaranteeing termination exactly when $m \geq 0$ initially, we need to specify different unfair sequences for different initial conditions. Namely, we need to regard $A1^{\omega}$ as an unfair sequence when $m > 0$ initially but not when $m < 0$. □

To allow different specifications of unfair action sequences for different initial conditions, we use the notation $P_{\alpha} : uf_{\alpha}$ corresponding to the unfair action sequences uf_{α} for initial condition P_{α}. Here, the indices α are taken from any index set I, possibly infinite. Note that once we have specified the unfair sequences for certain initial conditions, we do not have further control over the execution of the action system inside the loop — the demon choosing the execution sequences has full control subject to the constraint of the unfair sequences specified upon initialisation.

4.3 Definition

We now define fair action systems $\mathcal{A}$ which do not engage in any of the "unfair" action sequences uf_α under initial condition P_α, but otherwise behaves like ordinary action systems. To formalise this description of the behaviour of $\mathcal{A}$, we define $\mathcal{A}$ as a predicate transformer.

Definition 14. Let I be an index set. For $\alpha \in I$, let $P_\alpha : Pred$ such that $\exists \alpha P_\alpha \Leftrightarrow true$. Let

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

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

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

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 $t : seq\mathcal{A}$,

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

□

In Proposition 15, we will show that $\mathcal{A}(P)$ is well-defined in the sense that it is independent of the representation of the $P_\alpha : uf_\alpha$. When there is no ambiguity, we will write $\mathcal{H}^*_{\kappa,\mathcal{A}}$ instead of $\mathcal{H}^*_{\kappa,\mathcal{A},\alpha}$.

In the definition, we think of uf_α as being the unfair infinite sequences of the action system under initial condition P_α. We insist on $\exists \alpha P_\alpha$ being $true$ rather than $\bigvee_{i=1}^{n} g_i$ since the latter would add the disjunct $\neg \bigvee_{i=1}^{n} g_i$ to the second conjunct in the definition of $\mathcal{A}(P)$. Otherwise, $\mathcal{A}(P)$ is unchanged because any unfair sequences specified for initial conditions outside the disjunction of the guards do not affect the calculation.

It can be seen that the definition is a generalisation of the standard definition of iteration: when $uf_\alpha = \emptyset$ for all initial conditions, corresponding to the unfair execution of $\mathcal{A}$, we obtain the characterisation of the standard definition given in Theorem 11. The first conjunct (giving the weakest precondition under which the termination of $\mathcal{A}$ is guaranteed to establish $P : Pred$) is the same as in Theorem 11. The second conjunct, the termination condition for $\mathcal{A}$, is different in the general case. In our definition of fair action systems we discount all unfair action systems in our calculation, because they are prevented from occurring. In the definition of $\mathcal{H}^*_{\kappa+1,\mathcal{A},\alpha}$, we delete the prefix $s|_m$ from any unfair sequences when calculating $\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m},\alpha}$ because we have already executed $s|_m$ before entering $\mathcal{A}^{s|m}$.

Also, it can be seen that when $uf_\alpha = seq_\omega \mathcal{A}$ for all initial conditions, $\mathcal{H}^*_{1,\mathcal{A},\alpha} \Leftrightarrow true$ and so the second conjunct in the definition is satisfied trivially. Thus, we obtain the weakest errorfree precondition of the action system (Definition 12).

If $s : seq_\omega \mathcal{A}$ is guaranteed to terminate, i.e. $\exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}}) \Leftrightarrow true$, then it makes no difference to the calculation of $\mathcal{A}(P)$ whether or not s is included in uf_α. This reflects the fact that s is an "unreal" sequence, and cannot occur anyway.

Our definition of fair action systems in terms of predicate transformer semantics allows us to use the definitions of refinement and data refinement in Subsection 2.2.

We now exemplify the calculations involved in the definition.

Example 4. We consider the action systems mentioned in Examples 1 and 2 taking fairness into account. Let

$$
\begin{array}{lllllll}
\mathcal{A} \;\widehat{=} & \textbf{var} & & b : \mathbb{B} & & & \\
& \textbf{do} & & b & \rightarrow & b := false & (A0) \\
& & [] & b & \rightarrow & skip & (A1) \\
& & \textbf{uf} & \multicolumn{3}{l}{true : \{A1^\omega\}} & \\
& \textbf{od}. & & & & &
\end{array}
$$

Let $s : seq_\omega \mathcal{A} \setminus \{A1^\omega\}$. Then for some $k : \mathbb{N}$ and some $t : seq_\omega \mathcal{A}$, $s = A1^k; A0; t$. Since $A1^k; A0(\neg b) \Leftrightarrow true$, $\mathcal{H}^*_{1,\mathcal{A}} \Leftrightarrow true$. Thus the second conjunct in the definition evaluates to $true$, and the action system is guaranteed to terminate.

For the first conjunct, notice that for $k : \mathbb{N}$ and $P : Pred\mathcal{A}$,

$$A1^k; A0(P \vee b) \Leftrightarrow (b \Rightarrow P[false/b])$$

and $skip(P \vee b) \Leftrightarrow (\neg b \Rightarrow P)$, whereas for other $s : seq\mathcal{A}$, $s(P \vee b) \Leftrightarrow true$. Therefore, the first conjunct (and hence $\mathcal{A}(P)$) evaluates to

$$(\neg b \Rightarrow P) \wedge (b \Rightarrow P[false/b]).$$

Now consider the action system

$$\begin{array}{llllll} \mathcal{B} \mathrel{\widehat{=}} & \textbf{var} & & n : \mathbb{N} & & \\ & \textbf{do} & & n > 0 & \rightarrow & n := n - 1 \qquad (B0) \\ & & [] & n > 0 & \rightarrow & skip \qquad (B1) \\ & & \textbf{uf} & \multicolumn{3}{l}{true : \{s; B1^{\omega} \mid s : seq\mathcal{B}\}} \\ & \textbf{od}. & & & & \end{array}$$

Let $s : seq_{\omega}\mathcal{B} \setminus \{s; B1^{\omega} \mid s : seq\mathcal{B}\}$, note that s contains infinitely many occurrences of $(B0)$. Take any $t \sqsubseteq s$, and let $k : \mathbb{N}$ be the number of occurrences of $(B0)$ in t. Then $t(n \leq 0) \Leftrightarrow n \leq k$. Since we can choose t to contain arbitrarily many k, $\exists m < \omega \; s|_m(n \leq 0) \Leftrightarrow true$. Thus $\mathcal{H}^*_{1,\mathcal{B}}$, and hence the second conjunct in the definition, evaluate to $true$. So the action system is guaranteed to terminate.

For the first conjunct, notice that for $s : seq\mathcal{B}$, if s contains $k : \mathbb{N}$ occurrences of $(B0)$ and ends in $(B0)$, then for $P : Pred\mathcal{B}$,

$$s(P \vee n > 0) \Leftrightarrow (n = k \Rightarrow P[n - k/n]) \Leftrightarrow (n = k \Rightarrow P[0/n]).$$

If s ends in $(B1)$, then $s(P \vee n > 0) \Leftrightarrow true$, while

$$skip(P \vee n > 0) \Leftrightarrow (n \leq 0 \Rightarrow P).$$

Therefore, the first conjunct (and hence $\mathcal{B}(P)$) evaluates to

$$(n \leq 0 \Rightarrow P) \wedge (n > 0 \Rightarrow P[0/n]).$$

Finally, consider the action system

$$\begin{array}{llllll} \mathcal{C} \mathrel{\widehat{=}} & \textbf{var} & & b : \mathbb{B}\,; n : \mathbb{N} & & \\ & \textbf{init} & & b = true & & \\ & \textbf{do} & & b & \rightarrow & n :\in \mathbb{N}\,; b := false \qquad (C0) \\ & & [] & \neg b \wedge n > 0 & \rightarrow & n := n - 1 \qquad (C1) \\ & & [] & \neg b \wedge n > 0 & \rightarrow & skip \qquad (C2) \\ & & \textbf{uf} & \multicolumn{3}{l}{true : \{C0; s; C2^{\omega} \mid s : seq\{C1, C2\}\}} \\ & \textbf{od}. & & & & \end{array}$$

We evaluate the second conjunct first. $\mathcal{H}^*_{0,\mathcal{C}} \Leftrightarrow \neg b \wedge n = 0$. For $\mathcal{H}^*_{1,\mathcal{C}}$, we calculate that

$$\begin{array}{rcl} \exists m < \omega\ (C0; s; C1^\omega)|_m(\mathcal{H}^*_{0,\mathcal{C}}) & \Leftrightarrow & \neg b \\ \exists m < \omega\ (s; C1^\omega)|_m(\mathcal{H}^*_{0,\mathcal{C}}) & \Leftrightarrow & true \\ \exists m < \omega\ (s; C2^\omega)|_m(\mathcal{H}^*_{0,\mathcal{C}}) & \Leftrightarrow & 0 \leq n \leq p \\ \exists m < \omega\ u|_m(\mathcal{H}^*_{0,\mathcal{C}}) & \Leftrightarrow & true, \end{array}$$

where $s : seq\mathcal{C}$ contains $(C1)$ p times but does not contain $(C0)$, and u is any other member of $s : seq_\omega\mathcal{C}$ which is not unfair. (These u are action sequences with $(C0)$ occurring at places other than the start of the action sequence.) Thus, we evaluate $\mathcal{H}^*_{1,\mathcal{C}} \Leftrightarrow \neg b \wedge n = 0$. The calculation of $\mathcal{H}^*_{2,\mathcal{C}}$ gives exactly the same results as for $\mathcal{H}^*_{1,\mathcal{C}}$, except that

$$\exists m < \omega\ (C0; s; C1^\omega)|_m(\mathcal{H}^*_{1,\mathcal{C}(C0;s;C1^\omega)|m}) \Leftrightarrow true$$

for $s : seq\mathcal{C}$ containing no $(C0)$, so that $\mathcal{H}^*_{2,\mathcal{C}} \Leftrightarrow b \vee n = 0$. For $\kappa > 2$, $\mathcal{H}^*_{\kappa,\mathcal{C}} \Leftrightarrow b \vee n = 0$ also, so the second conjunct evaluates to $b \vee n = 0$. Since the initialisation is $b := true$, and since $(b := true)(b \vee n = 0) \Leftrightarrow true$, we see that the action system is guaranteed to terminate.

For the first conjunct,

$$\begin{array}{rcl} skip(P \vee b \vee n > 0) & \Leftrightarrow & P \vee b \vee n > 0 \\ C0(P \vee b \vee n > 0) & \Leftrightarrow & (b \Rightarrow P[false, n :\in \mathbb{N}/b, n]) \\ s(P \vee b \vee n > 0) & \Leftrightarrow & (\neg b \wedge n = k \Rightarrow P[0/n]) \\ t(P \vee b \vee n > 0) & \Leftrightarrow & (b \Rightarrow P[false, 0/b, n]) \end{array}$$

where s consists of $(C2)$s and k $(C1)$s, and ends in $(C1)$, and t begins with $(C0)$, contains no other $(C0)$, and ends with $(C1)$. All other members of $seq\mathcal{C}$ evaluate to $true$. Hence, taking the conjunction, we obtain

$$(b \Rightarrow P[false, 0/b, n]) \wedge (\neg b \Rightarrow P[0/n]).$$

Since the initialisation is $b := true$ and

$$(b := true)((b \Rightarrow P[false, 0/b, n]) \wedge (\neg b \Rightarrow P[0/n])) \Leftrightarrow P[false, 0/b, n],$$

we see that the first conjunct, and hence $\mathcal{C}(P)$, evaluates to $P[false, 0/b, n]$. □

Proposition 15. *Definition 14 is independent of the representation* $P_\alpha : uf_\alpha$.

Proof. Let I and J be index sets and $P_\alpha, Q_\beta : Pred$ for all $\alpha \in I$ and $\beta \in J$. Suppose that

$$\exists \alpha \in I \ P_\alpha \ \Leftrightarrow \exists \beta \in J \ Q_\beta \ \Leftrightarrow \ true.$$

Suppose that whenever $P_\alpha \wedge Q_\beta \neq false$ for some $\alpha \in I$ and some $\beta \in J$, we have $uf_\alpha = uf_\beta$. We show that

$$\exists \alpha \in I \ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha} \ \Leftrightarrow \ \exists \beta \in J \ Q_\beta \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\beta}.$$

For $\alpha \in I$, define $S_\alpha \ \widehat{=} \ \{\beta \in J | P_\alpha \wedge Q_\beta \neq false\}$. Then

$$
\begin{array}{ll}
 & \exists \alpha \in I \ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha} \\
\Leftrightarrow & \hfill \exists \beta \in J \ Q_\beta \ \Leftrightarrow \ true \\
 & (\exists \alpha \in I \ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}) \wedge \exists \beta \in J \ Q_\beta \\
\Leftrightarrow & \\
 & \exists \alpha \in I \ (P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha} \wedge (\exists \beta \in S_\alpha \ Q_\beta \ \vee \ \exists \beta \in J \setminus S_\alpha \ Q_\beta)) \\
\Leftrightarrow & \hfill \text{by construction of } S_\alpha,\ \exists \alpha \in I \ (P_\alpha \ \wedge \ \exists \beta \in J \setminus S_\alpha \ Q_\beta) \ \Leftrightarrow \ false \\
 & \exists \alpha \in I \ (P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha} \ \wedge \ \exists \beta \in S_\alpha \ Q_\beta) \\
\Leftrightarrow & \hfill \text{by construction of } S_\alpha,\ uf_\alpha = uf_\beta \text{ for } \beta \in S_\alpha \\
 & \exists \alpha \in I \ (P_\alpha \wedge (\exists \beta \in S_\alpha \ Q_\beta \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\beta})) \\
\Rightarrow & \\
 & \exists \alpha \in I \ \exists \beta \in S_\alpha \ Q_\beta \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\beta} \\
\Rightarrow & \\
 & \exists \beta \in J \ Q_\beta \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\beta}.
\end{array}
$$

The result holds by symmetry. □

4.4 Properties

In this subsection we establish some basic properties of Definition 14.

The following result shows that fair action systems can be used as statements within other programs.

Proposition 16. *Fair action systems as defined in Definition 14 satisfy conjunctivity and monotonicity.*

Proof. Let $\mathcal{A}$ be a fair action system with guards g_i, $1 \leq i \leq n$.
Monotonicity: Suppose P,Q : $Pred$ with $P \Rightarrow Q$. Then

$$\mathcal{A}(P)$$

$\Leftrightarrow$

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

$\Rightarrow$ monotonicity of action sequences

$$\forall s : seq\mathcal{A}\ s(Q \vee \bigvee_{i=1}^{n} g_i) \ \wedge\ \exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}$$

$\Leftrightarrow$

$$\mathcal{A}(Q).$$

Conjunctivity: Let $J \neq \emptyset$, let $P_\gamma : Pred$ for $\gamma \in J$. Then

$$\forall \gamma \in J\ \mathcal{A}(P_\gamma)$$

$\Leftrightarrow$

$$\forall \gamma \in J\ (\forall s : seq\mathcal{A}\ s(P_\gamma \vee \bigvee_{i=1}^{n} g_i) \wedge \exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha})$$

$\Leftrightarrow$

$$(\forall s : seq\mathcal{A}\ \forall \gamma \in J\ s(P_\gamma \vee \bigvee_{i=1}^{n} g_i)) \wedge \exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}$$

$\Rightarrow$ conjunctivity of finite action sequences (Proposition 5)

$$\forall s : seq\mathcal{A}\ s((\forall \gamma \in J\ P_\gamma) \vee \bigvee_{i=1}^{n} g_i) \wedge \exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A},\alpha}$$

$\Leftrightarrow$

$$\mathcal{A}(\forall \gamma \in J\ P_\gamma).$$

□

Lemma 17. *Let $\kappa < \mu$. Then $\mathcal{H}^*_{\kappa,\mathcal{A}} \Rightarrow \mathcal{H}^*_{\mu,\mathcal{A}}$.*

Proof. We will prove the result by induction on κ. The case where $\kappa = 0$ and $\mu + 1$ is a successor ordinal follows from

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

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

$$\neg \bigvee_{i=1}^{n} g_i$$

$\Rightarrow$ Theorem 7

$$\forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\mu,\mathcal{A}^{s|m}})$$

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

$$\mathcal{H}^*_{\mu+1,\mathcal{A}}.$$

If $\mu = \delta$ is a limit ordinal, then

$\mathcal{H}^*_{\kappa,\mathcal{A}}$

$\Rightarrow$ $\kappa < \delta$

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

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

$\mathcal{H}^*_{\delta,\mathcal{A}}$.

If $\kappa + 1$ and $\mu + 1$ are successor ordinals, then

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

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

$\forall s : seq_\omega \mathcal{A} \setminus uf_\alpha \ \exists m < \omega \ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}})$

$\Rightarrow$ $\kappa < \mu$ and induction hypothesis

$\forall s : seq_\omega \mathcal{A} \setminus uf_\alpha \ \exists m < \omega \ s|_m(\mathcal{H}^*_{\mu,\mathcal{A}^{s|m}})$

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

$\mathcal{H}^*_{\mu+1,\mathcal{A}}$.

Finally, if $\kappa = \delta$ is a limit ordinal, then

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

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

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

$\Rightarrow$ induction hypothesis and $\kappa < \delta < \mu$ implies $\kappa < \mu$

$\mathcal{H}^*_{\mu,\mathcal{A}}$.

□

Lemma 18. *Let $s, t : seq\mathcal{A}$ with $s \sqsubseteq t$. Then*

$$s(\mathcal{H}^*_{\kappa,\mathcal{A}^s}) \Rightarrow t(\mathcal{H}^*_{\kappa,\mathcal{A}^t}).$$

Proof. The case $s = t$ is trivial. Otherwise, we use induction on κ. For the base step,

$s(\mathcal{H}^*_{0,\mathcal{A}^s})$

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

$$s(\neg \bigvee_{i=1}^{n} g_i)$$

$\Rightarrow$ Theorem 7

$$t(\neg \bigvee_{i=1}^{n} g_i)$$

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

$$t(\mathcal{H}^*_{0,\mathcal{A}^t}),$$

for the successor step

$$s(\mathcal{H}^*_{\kappa+1,\mathcal{A}^s})$$

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

$$s(\forall u : seq_\omega \mathcal{A} \setminus \{v : seq_\omega \mathcal{A} \mid s; v : uf_\alpha\} \ \exists m < \omega \ u|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s;u|m}}))$$

$\Rightarrow$

$$s(\forall u : seq_\omega \mathcal{A} \setminus \{v : seq_\omega \mathcal{A} \mid t; v : uf_\alpha\} \ \exists m < \omega \ (t - s; u)|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s;(t-s;u)|m}}))$$

$\Rightarrow$ induction hypothesis

$$s(\forall u : seq_\omega \mathcal{A} \setminus \{v : seq_\omega \mathcal{A} \mid t; v : uf_\alpha\} \ \exists m < \omega \ t - s; u|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{t;u|m}}))$$

$\Leftrightarrow$ conjunctivity

$$t(\forall u : seq_\omega \mathcal{A} \setminus \{v : seq_\omega \mathcal{A} \mid t; v : uf_\alpha\} \ \exists m < \omega \ u|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{t;u|m}}))$$

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

$$t(\mathcal{H}^*_{\kappa+1,\mathcal{A}^t})$$

and for the limit step

$$s(\mathcal{H}^*_{\delta,\mathcal{A}^s})$$

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

$$s(\exists \kappa < \delta \ \mathcal{H}^*_{\kappa,\mathcal{A}^s})$$

$\Rightarrow$ induction hypothesis and monotonicity

$$s(\exists \kappa < \delta \ t - s(\mathcal{H}^*_{\kappa,\mathcal{A}^t}))$$

$\Rightarrow$

$$t(\exists \kappa < \delta \ \mathcal{H}^*_{\kappa,\mathcal{A}^t})$$

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

$$t(\mathcal{H}^*_{\delta,\mathcal{A}^t}).$$

$\square$

Theorem 19. *If all actions in $\mathcal{A}$ are boundedly nondeterministic, then*

$$\exists\kappa\mathcal{H}^*_{\kappa,\mathcal{A}} \Leftrightarrow \mathcal{H}^*_{1,\mathcal{A}}.$$

Proof. Recall that bounded nondeterminism is equivalent to upper continuity.

We will show by induction on κ that $\mathcal{H}^*_{\kappa,\mathcal{A}} \Rightarrow \mathcal{H}^*_{1,\mathcal{A}}$.

The case $\kappa = 0$ follows from Lemma 17. For the successor step,

$$\begin{aligned}
& \mathcal{H}^*_{\kappa+1,\mathcal{A}} \\
\Leftrightarrow & \qquad \text{definition of } \mathcal{H}^*_{\kappa+1,\mathcal{A}} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|_m}}) \\
\Rightarrow & \qquad \text{induction hypothesis} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{1,\mathcal{A}^{s|_m}}) \\
\Leftrightarrow & \qquad \text{definition of } \mathcal{H}^*_{1,\mathcal{A}^{s|_m}} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\forall t : seq_\omega\mathcal{A} \setminus \{u : seq_\omega\mathcal{A} \mid s|_m ; u : uf_\alpha\} \\
& \qquad \exists m' < \omega\ t|_{m'}(\mathcal{H}^*_{0,\mathcal{A}^{s|_m;t|_{m'}}})) \\
\Rightarrow & \qquad \text{monotonicity} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ \forall t : seq_\omega\mathcal{A} \setminus \{u : seq_\omega\mathcal{A} \mid s|_m ; u : uf_\alpha\} \\
& \qquad s|_m(\exists m' < \omega\ t|_{m'}(\mathcal{H}^*_{0,\mathcal{A}^{s|_m;t|_{m'}}})) \\
\Leftrightarrow & \qquad \text{upper continuity, the chain property following from Lemma 18} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ \forall t : seq_\omega\mathcal{A} \setminus \{u : seq_\omega\mathcal{A} \mid s|_m ; u : uf_\alpha\} \\
& \qquad \exists m' < \omega\ s|_m ; t|_{m'}(\mathcal{H}^*_{0,\mathcal{A}^{s|_m;t|_{m'}}}) \\
\Leftrightarrow & \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{0,\mathcal{A}^{s|_m}}) \\
\Leftrightarrow & \qquad \text{definition of } \mathcal{H}^*_{1,\mathcal{A}} \\
& \mathcal{H}^*_{1,\mathcal{A}}.
\end{aligned}$$

The limit step is

$$\begin{aligned}
& \mathcal{H}^*_{\delta,\mathcal{A}} \\
\Leftrightarrow & \qquad \text{definition of } \mathcal{H}^*_{\delta,\mathcal{A}} \\
& \exists\kappa < \delta\ \mathcal{H}^*_{\kappa,\mathcal{A}} \\
\Rightarrow & \qquad \text{induction hypothesis} \\
& \mathcal{H}^*_{1,\mathcal{A}}.
\end{aligned}$$

$\square$

4.5 Data Refinement

The following theorem gives us a method for obtaining a concrete action system given an abstract action system and a data refinement relation φ:

Theorem 20. *Let $S_i \leq_\varphi T_i$ for $1 \leq i \leq n$. Let*

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

and

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

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

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

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

Proof. First, we show that $\varphi(\mathcal{H}^*_{\kappa,\mathcal{A}}) \Rightarrow \mathcal{H}^*_{\kappa,\mathcal{B}}$ by induction on κ. For the base step,

$\varphi(\mathcal{H}^*_{0,\mathcal{A}})$

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

$\varphi(\neg \bigvee_{i=1}^{n} g_i)$

$\Rightarrow$ monotonicity of φ

$\neg \bigvee_{i=1}^{n} \overline{\varphi}(g_i)$

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

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

while for the successor step

$\varphi(\mathcal{H}^*_{\kappa+1,\mathcal{A}})$

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

$$\varphi(\forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}}))$$

$\Rightarrow$ monotonicity of φ

$$\forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \varphi(\exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}}))$$

$\Rightarrow$ or-distributivity of φ

$$\forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ \varphi \circ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}})$$

$\Rightarrow$ definition of uf'_α, $g_i \to S_i \leq_\varphi \overline{\varphi}(g_i) \to T_i$ and Theorem 3

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m < \omega\ s|_m \circ \varphi(\mathcal{H}^*_{\kappa,\mathcal{A}^{\varphi^{-1}(s|m)}})$$

$\Rightarrow$ induction hypothesis

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|m}})$$

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

$$\mathcal{H}^*_{\kappa+1,\mathcal{B}}$$

and for the limit step

$$\varphi(\mathcal{H}^*_{\delta,\mathcal{A}})$$

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

$$\varphi(\exists\kappa < \delta\ \mathcal{H}^*_{\kappa,\mathcal{A}})$$

$\Leftrightarrow$ or-distributivity of φ

$$\exists\kappa < \delta\ \varphi(\mathcal{H}^*_{\kappa,\mathcal{A}})$$

$\Rightarrow$ induction hypothesis

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

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

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

Therefore,

$$\varphi \circ \mathcal{A}(P)$$

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

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

$\Rightarrow$ monotonicity of φ

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

$\Rightarrow$ $g_i \to S_i \leq_\varphi \overline{\varphi}(g_i) \to T_i$ and Theorem 3

$$\forall s : seq\mathcal{B}\ s \circ \varphi(P \vee \bigvee_{i=1}^{n} g_i) \wedge \varphi(\exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A}})$$

$\Rightarrow$ or-distributivity of φ

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

$\Rightarrow$ from monotonicity and or-distributivity of φ, $\varphi \leq \overline{\varphi}$

$$\forall s : seq\mathcal{B}\ s(\varphi(P) \vee \bigvee_{i=1}^{n} \overline{\varphi}(g_i)) \wedge \exists \alpha \in I\ \varphi(P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{A}})$$

$\Rightarrow$ monotonicity and or-distributivity of φ

$$\forall s : seq\mathcal{B}\ s(\varphi(P) \vee \bigvee_{i=1}^{n} \overline{\varphi}(g_i)) \wedge \exists \alpha \in I\ \varphi(P_\alpha) \wedge \exists \kappa\ \varphi(\mathcal{H}^*_{\kappa,\mathcal{A}})$$

$\Rightarrow$

$$\begin{aligned}\forall s : seq\mathcal{B}\ s(\varphi(P) \vee \bigvee_{i=1}^{n} \overline{\varphi}(g_i)) \wedge ((\exists \alpha \in I\ \varphi(P_\alpha) \wedge \exists \kappa\ \varphi(\mathcal{H}^*_{\kappa,\mathcal{A}}))\\ \vee((\neg \exists \alpha \in I\ \varphi(P_\alpha)) \wedge\ \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{B}}))\end{aligned}$$

$\Rightarrow$ above, and definition of $\mathcal{B}$

$$\mathcal{B} \circ \varphi(P).$$

□

As a corollary, we obtain Theorem 3 for fair action systems:

Corollary 21. *Data refinement $\leq_\varphi$ distributes through the components of a fair action system.* □

5 Program Development

In this section, we give some simple rules by which programs may be developed. In the case-study of the readers-writers problem in Section 7, we will see that applying them in practice is fairly simple compared to the relatively complicated nature of some of the proofs.

5.1 Strengthening Fairness Requirements

Strengthening the fairness requirements of an action system provides a refinement:

Theorem 22. *Let I be an index set and for $\alpha \in I$, let $P_\alpha : Pred\mathcal{A}$ such that $\exists \alpha\ P_\alpha \Leftrightarrow true$. Let*

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

and

$$\mathcal{B} \;\widehat{=}\; \begin{array}{ll} \mathbf{do} & []_{i=1}^{n} \;\; g_i \;\; \rightarrow \;\; S_i \\ & \mathbf{uf} \qquad P_\alpha : uf'_\alpha \\ \mathbf{od}, & \end{array}$$

where for all $\alpha \in I$, $uf_\alpha \subseteq uf'_\alpha$. Then $\mathcal{A} \leq \mathcal{B}$.

Proof. First we show by induction on κ that $\mathcal{H}^*_{\kappa,\mathcal{A}} \Rightarrow \mathcal{H}^*_{\kappa,\mathcal{B}}$. The case $\kappa = 0$ is straightforward, as both sides evaluate to $\neg \bigvee_{i=1}^{n} g_i$. For the successor step,

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

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

$\forall s : seq_\omega \mathcal{A} \setminus uf_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}})$

$\Leftrightarrow$ $\overline{seq}\mathcal{A} = \overline{seq}\mathcal{B}$

$\forall s : seq_\omega \mathcal{B} \setminus uf_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}})$

$\Rightarrow$ $uf_\alpha \subseteq uf'_\alpha$

$\forall s : seq_\omega \mathcal{B} \setminus uf'_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|m}})$

$\Rightarrow$ induction hypothesis

$\forall s : seq_\omega \mathcal{B} \setminus uf'_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|m}})$

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

$\mathcal{H}^*_{\kappa+1,\mathcal{B}}$.

The limit case is straightforward also:

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

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

$\exists \kappa < \delta \; \mathcal{H}^*_{\kappa,\mathcal{A}}$

$\Rightarrow$ induction hypothesis

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

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

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

Therefore,

$\mathcal{A}(P)$

$\Leftrightarrow$ definition of $\mathcal{A}(P)$

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

$\Leftrightarrow$ $seq\mathcal{A} = seq\mathcal{B}$

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

$\Rightarrow$ above

$(\forall s : seq\mathcal{B}\ s(P \vee \bigvee_{i=1}^{n} g_i)) \wedge (\exists \alpha \in I\ P_\alpha \wedge \exists \kappa \mathcal{H}^*_{\kappa,\mathcal{B}})$

$\Leftrightarrow$ definition of $\mathcal{B}(P)$

$\mathcal{B}(P)$

□

Thus, fair action systems with fixed initial condition, fixed body and varying fairness criteria form a complete lattice under the partial order $\leq$. For any subset of points in this lattice, the least upper bound (greatest lower bound) equals the union (intersection) of corresponding uf_α's. The minimum element of the lattice is the weakest precondition of the action system given by the standard definition; the maximum element is the weakest errorfree precondition of the action system.

The following corollary is a generalisation of Theorem 22:

Corollary 23. *Let I and J be index sets and for $\alpha \in I$ and $\beta \in J$, let $P_\alpha, Q_\beta : Pred$ such that $\exists \alpha\ P_\alpha \ \Leftrightarrow\ true \ \Leftrightarrow\ \exists \beta\ Q_\beta$. Let*

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

and

$$\begin{array}{lll} \mathcal{B} \ \widehat{=} & \mathbf{do} & []_{i=1}^{n}\ g_i \ \rightarrow\ S_i \\ & & \mathbf{uf} \quad Q_\beta : uf'_\beta \\ & \mathbf{od}, & \end{array}$$

where for all $\alpha \in I$ and $\beta \in J$, $P_\alpha \cap Q_\beta \neq \emptyset \ \Rightarrow\ uf_\alpha \subseteq uf'_\beta$. Then $\mathcal{A} \leq \mathcal{B}$.

Proof. For $P_\alpha \cap Q_\beta \neq \emptyset$, replace P_α in $\mathcal{A}$ and Q_β in $\mathcal{B}$ by $P_\alpha \cap Q_\beta$, which is valid by Proposition 15. Then apply Theorem 22. □

5.2 Monotonicity

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

Theorem 24 Monotonicity for Fair Action Systems. *Let*

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

and

$$\begin{array}{rcl} \mathcal{B} & \widehat{=} & \mathbf{do} \quad []_{i=1}^{n} []_{j=1}^{m_i} \; h_{i_j} \;\rightarrow\; T_{i_j} \\ & & \qquad \mathbf{uf} \qquad\quad P_\alpha : uf'_\alpha \\ & & \mathbf{od}, \end{array}$$

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 $\mathcal{A} \leq \mathcal{B}$.

Note that in the definition of uf'_α, 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.

Proof. Define

$$\begin{array}{rcl} \psi : \overline{seq}\mathcal{B} & \rightarrow & \overline{seq}\mathcal{A} \\ by \;\; \psi(s) & = & s[g_i \rightarrow S_i/h_{i_j} \rightarrow T_{i_j}]. \end{array}$$

Note that $uf'_\alpha = \psi^{-1}(uf_\alpha)$.

One can show by induction on the length of s that $\forall s : seq\mathcal{B}\;\psi(s) \leq s$. One requires only $g_i \rightarrow S_i \leq []_{j=1}^{m_i} h_{i_j} \rightarrow T_{i_j}$ and Corollary 6.

First Conjunct:

$\forall s : seq\mathcal{A}\; s(P \vee \bigvee_{i=1}^{n} g_i)$

$\Leftrightarrow$ $\qquad\qquad$ $\psi(seq\mathcal{B}) = seq\mathcal{A}$

$\forall s : seq\mathcal{B}\; \psi(s)(P \vee \bigvee_{i=1}^{n} g_i)$

$\Rightarrow$ $\qquad\qquad$ $\psi(s) \leq s$ (see above)

$\forall s : seq\mathcal{B}\; s(P \vee \bigvee_{i=1}^{n} g_i)$

$\Rightarrow$ $\qquad\qquad$ $\bigvee_{i=1}^{n} g_i \;\Rightarrow\; \bigvee_{i=1}^{n} \bigvee_{j=1}^{m_i} h_{i_j}$ and monotonicity of action sequences

$\forall s : seq\mathcal{B}\; s(P \vee \bigvee_{i=1}^{n} \bigvee_{j=1}^{m_i} h_{i_j})$.

Second Conjunct: Fix $\alpha \in I$. We show by induction on κ that $\mathcal{H}^*_{\kappa,\mathcal{A}} \Rightarrow \mathcal{H}^*_{\kappa,\mathcal{B}}$. The case $\kappa = 0$ holds since $\bigvee_{i=1}^n g_i \;\Leftarrow\; \bigvee_{i=1}^n \bigvee_{j=1}^{m_i} h_{i_j}$.

We now consider the successor case. Let $s : seq_\omega\mathcal{B} \setminus uf'_\alpha$. Then $\psi(s) : seq_\omega\mathcal{A}$. Also $\psi(s) \notin uf_\alpha$ since $uf'_\alpha = \psi^{-1}(uf_\alpha)$. So $\psi(s) : seq_\omega\mathcal{A} \setminus uf_\alpha$. Therefore,

$$
\begin{array}{ll}
& \mathcal{H}^*_{\kappa+1,\mathcal{A}} \\
\Leftrightarrow & \hfill \text{definition of } \mathcal{H}^*_{\kappa+1,\mathcal{A}} \\
& \forall s : seq_\omega\mathcal{A} \setminus uf_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|_m}}) \\
\Rightarrow & \hfill \text{remark just made} \\
& \forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha \; \exists m < \omega \; \psi(s)|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{\psi(s)|_m}}) \\
\Rightarrow & \hfill \psi(s)|_m \leq s|_m \\
& \forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{\psi(s)|_m}}) \\
\Rightarrow & \hfill \text{induction hypothesis} \\
& \forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_m}}) \\
\Leftrightarrow & \hfill \text{definition of } \mathcal{H}^*_{\kappa+1,\mathcal{B}} \\
& \mathcal{H}^*_{\kappa+1,\mathcal{B}}.
\end{array}
$$

The proof for the limit step is standard, see for example the proof of Theorem 22. □

Note that

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

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:

Corollary 25 Strengthen Guards. *If, for all i such that $1 \leq i \leq n$,*

$$g_i \Leftarrow h_i \quad \textit{and} \quad \bigvee_{i=1}^n g_i \Rightarrow \bigvee_{i=1}^n h_i,$$

then

$$\begin{array}{lll} \mathbf{do} & [\!]_{i=1}^{n} & g_i \rightarrow S_i \\ & \mathbf{uf} & P_\alpha : uf_\alpha \\ \mathbf{od} & & \end{array} \quad \leq \quad \begin{array}{lll} \mathbf{do} & [\!]_{i=1}^{n} & h_i \rightarrow S_i \\ & \mathbf{uf} & P_\alpha : uf_\alpha \\ \mathbf{od}. & & \end{array}$$

□

This corollary will account for many of the practical applications of Theorem 24 in the case study of the readers-writers problem.

5.3 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 *skips*, is justified by the following result:

Theorem 26. *Let*

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

and

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

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

Proof. Define

$$\begin{array}{rcl} \psi : \overline{seq}\mathcal{B} & \rightarrow & \overline{seq}\mathcal{A} \\ by \;\; \psi(s) & = & s[/\bigvee_{i=1}^{n} g_i \rightarrow skip]. \end{array}$$

One can show by induction on the length of s that $\forall s : seq\mathcal{B}\ \psi(s) \leq s$. One requires only $skip \leq \neg \bigvee_{i=1}^{n} g_i \rightarrow skip$ and Corollary 6.
First Conjunct:

$$\forall s : seq\mathcal{A}\ s(P \vee \textstyle\bigvee_{i=1}^{n} g_i)$$

$\Leftrightarrow$ $\qquad\qquad\qquad\qquad$ $\psi(seq\mathcal{B}) = seq\mathcal{A}$

$$\forall s : seq\mathcal{B}\ \psi(s)(P \vee \textstyle\bigvee_{i=1}^{n} g_i)$$

$\Rightarrow$ $\qquad\qquad$ $\psi(s) \leq s$ (see above)

$$\forall s : seq\mathcal{B}\ s(P \vee \textstyle\bigvee_{i=1}^{n} g_i)$$

$\Rightarrow$ $\qquad\qquad$ $seq\mathcal{A} \subseteq seq\mathcal{B}$

$$\forall s : seq\mathcal{A}\ s(P \vee \textstyle\bigvee_{i=1}^{n} g_i).$$

Therefore,

$$\forall s : seq\mathcal{A}\ s(P \vee \bigvee_{i=1}^{n} g_i) \Leftrightarrow \forall s : seq\mathcal{B}\ s(P \vee \bigvee_{i=1}^{n} g_i).$$

Second Conjunct: Fix $\alpha \in I$. Define $\phi = \psi|_{seq_\omega\mathcal{B}\cap\psi^{-1}(seq\mathcal{A})}$ and $\phi_\alpha = \psi|_{\psi^{-1}(uf_\alpha)}$. Note that $uf'_\alpha = \phi_\alpha^{-1}(uf_\alpha) \cup \phi^{-1}(seq\mathcal{A})$.

We show by induction on κ that $\mathcal{H}^*_{\kappa,\mathcal{A}} \Leftrightarrow \mathcal{H}^*_{\kappa,\mathcal{B}}$. The case $\kappa = 0$ holds since both sides evaluate to $\neg \bigvee_{i=1}^{n} g_i$.

We now consider the successor case. Let $s : seq_\omega\mathcal{B} \setminus uf'_\alpha$. Then $\psi(s) : seq_\omega\mathcal{A}$ since if $\psi(s) : seq\mathcal{A}$, then $s : \phi^{-1}(seq\mathcal{A})$, which contradicts $s \notin uf'_\alpha$. Also $\psi(s) \notin uf_\alpha$, since otherwise $s : \phi_\alpha^{-1}(uf_\alpha)$, which also contradicts $s \notin uf'_\alpha$. Therefore, $\psi(s) : seq_\omega\mathcal{A} \setminus uf_\alpha$ and

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

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

$$\forall s : seq_\omega\mathcal{A} \setminus uf_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{s|_m}})$$

$\Rightarrow$ $\qquad\qquad$ remark just made

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m < \omega\ \psi(s)|_m(\mathcal{H}^*_{\kappa,\mathcal{A}^{\psi(s)|_m}})$$

$\Rightarrow$ $\qquad\qquad$ $\psi(s)|_m = \psi(s|_{m'}) \leq s|_{m'}$

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m' < \omega\ s|_{m'}(\mathcal{H}^*_{\kappa,\mathcal{A}^{\psi(s|_{m'})}})$$

$\Leftrightarrow$ $\qquad\qquad$ induction hypothesis

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m' < \omega\ s|_{m'}(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_{m'}}})$$

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

$$\mathcal{H}^*_{\kappa+1,\mathcal{B}}$$

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

$$\forall s : seq_\omega\mathcal{B} \setminus uf'_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_m}})$$

$\Rightarrow$ $\qquad\qquad$ $seq_\omega\mathcal{A} \subseteq seq_\omega\mathcal{B}$

$$\forall s : seq_\omega\mathcal{A} \setminus uf'_\alpha\ \exists m < \omega\ s|_m(\mathcal{H}^*_{\kappa,\mathcal{B}^{s|_m}})$$

$\Leftrightarrow$ $\qquad$ $seq_\omega \mathcal{A} \cap uf'_\alpha = uf_\alpha$

$$\forall s : seq_\omega \mathcal{A} \setminus uf_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa, \mathcal{B}^{s|m}})$$

$\Leftrightarrow$ $\qquad$ induction hypothesis

$$\forall s : seq_\omega \mathcal{A} \setminus uf_\alpha \; \exists m < \omega \; s|_m(\mathcal{H}^*_{\kappa, \mathcal{A}^{s|m}})$$

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

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

This shows that $\mathcal{H}^*_{\kappa+1,\mathcal{A}} \Leftrightarrow \mathcal{H}^*_{\kappa+1,\mathcal{B}}$.

The proof for the limit step is standard, see for example the proof of Theorem 22. $\square$

In Section 7, this theorem is usefully combined with the following:

Theorem 27. *If variable v is new, E an expression, $\vec{x}$ a collection of $n : \mathbb{N}$ variables and $\vec{X}$ a collection of n expressions, then*

- $skip \leq v := E$
- $\vec{x} := \vec{X} \leq \vec{x}, v := \vec{X}, E$.

Proof. For $P : Pred$,

$$skip(P)$$

$\Leftrightarrow$

$$P$$

$\Leftrightarrow$ $\qquad$ since v is new, P is independent of v

$$(v := E)(P).$$

Similarly, $(\vec{x} := \vec{X})(P) \Leftrightarrow (\vec{x}, v := \vec{X}, E)(P)$. $\square$

Use of Theorem 27 in combination with Theorem 26 allows us to introduce auxiliary variables to an action system. As we will see in Section 7, these auxiliary variables can then be integrated into the program as a whole by strengthening guards with conjuncts involving them, using Corollary 25.

6 Non-Terminating Action Systems

In this section, we define refinement for non-terminating action systems. We propose a definition which takes into account intermediate states of an action system, or equivalently models a reactive system by a sequence of finite approximations. We show that there are analogous definitions and theorems to those for terminating action systems.

6.1 Introduction

Definition 1 of refinement of action systems was made with terminating action systems in mind, yet it is natural to consider also non-terminating action systems. The term "non-terminating action system" is misleading, because it implies that action systems may be divided into two distinct classes. However, in general, an action system may be guaranteed to terminate for some intial conditions, guaranteed not to terminate for other initial conditions, and may or may not terminate for a third class of initial conditions.

With some programs, such as operating systems, communications protocols and electronic banking databases, non-termination does not reflect an error (as has been convenient to assume until now), but represents normal behaviour. These programs typically involve interaction with the environment. We find that Definition 1 is inadequate as a definition of refinement of such non-terminating action systems, because

$$\forall P : Pred\mathcal{A}\ (\mathcal{A}(P) \Leftrightarrow false)$$

for any action system $\mathcal{A}$ which never terminates. As a result, $\mathcal{A}$ is refined by any other action system. So a new definition of refinement is required which will

- supply us with a sensible notion of refinement of non-terminating systems
- relate to Definition 1 (refinement of terminating systems)
- satisfy desirable properties such as monotonicity and transitivity.

Since there may be no final state in a non-terminating action system, the new definition needs to take into account the *intermediate* states of the action system. Informally, we propose that a concrete action system refines an abstract one if those initial conditions which guarantee that the abstract action system will eventually reach a state satisfying $P : Pred$ will also guarantee that a state satisfying P will be reached in the concrete action system.

This condition on the states of the two action systems is weaker than the condition on traces imposed by trace-refinement. The trace-refinement definition requires the concrete action system to have almost the same actions as the abstract action system, with stuttering allowed. We find this too restrictive a definition, although it is a condition which is simple to verify in practice.

According to our informal definition, the concrete action system may contain intermediate states which are not present in the abstract one. For example, a transaction in an electronic banking database which transfers money from one account to another will be refined by one which at different times withdraws the money from one account and deposits the same amount in the other. Also, a prime number generator which generates prime numbers two at a time will be refined by one which generates them one at a time, but not vice-versa.

This informal definition of refinement of non-terminating action systems places no constraint on the *order* in which the intermediate states are reached. This may or may not be undesirable. For a prime-number generator, the order of generation may not be important so that a generator which generates primes in increasing order may refine (and be refined by) a generator which generates primes in some other order. On the other hand, an electronic banking database action system which transfers money between accounts would be expected to have the same order of variable transitions in the concrete as in the abstract: this reflects our desire that the balance of the account should always be up-to-date.

The constraint of preserving the order of states of variables can be incorporated into the action system by making the variables *sequences* which hold all of the different states of the variables. For example, the prime number generator

$$\begin{array}{lll} \mathcal{A} \mathrel{\widehat{=}} & \mathbf{var} & p : \mathbb{N} \\ & \mathbf{init} & p = 2 \\ & \mathbf{do} & \quad true \;\; \rightarrow \;\; p := \text{some prime not already generated } \mathbf{od} \end{array}$$

and the action systems which refine it under the new definition may have the primes generated in entirely different orders. On the other hand,

$$\begin{array}{lll} \mathcal{B} \mathrel{\widehat{=}} & \mathbf{var} & p : seq\,\mathbb{N} \\ & \mathbf{init} & p = 2 \\ & \mathbf{do} & \quad true \;\; \rightarrow \;\; p := p \frown \text{some prime not already generated } \mathbf{od} \end{array}$$

and all the action systems which refine it under the new definition have the primes generated in the same order.

6.2 Definition

We now give a formal definition of refinement of non-terminating action systems based on the informal discussion in Subsection 6.1. We could proceed by defining a new predicate transformer on a non-terminating action system

$$\mathcal{A} \mathrel{\widehat{=}} \mathbf{do} \;\; []_{i=1}^{n} \; g_i \;\; \rightarrow \;\; S_i \;\; \mathbf{od}$$

to capture the weakest precondition $wrp.\mathcal{A}.P$ which guarantees that $\mathcal{A}$ eventually reaches (and maybe passes through) a state satisfying $P : Pred\mathcal{A}$. Thus we would have $wrp.\mathcal{A}.P \;\widehat{=}\; \exists\kappa : Ord\ G_\kappa(P)$ (cf. the definition of iteration in Subsection 2.1), where

$$\begin{array}{rcl} G_0(P) & \widehat{=} & P, \\ G_{\kappa+1}(P) & \widehat{=} & G_0(P) \vee (\textbf{if } []_{i=1}^{n}\ g_i \rightarrow S_i \textbf{ fi})(G_\kappa(P)) \textit{ and} \\ G_\delta(P) & \widehat{=} & \exists\kappa < \delta\ G_\kappa(P). \end{array}$$

However, that would require us to develop a set of properties of this new predicate transformer. Instead, for $P : Pred\mathcal{A}$, we consider the *terminating* action system

$$\mathcal{A}_P \;\widehat{=}\; \textbf{do} \quad []_{i=1}^{n} \quad \neg P \wedge g_i \quad \rightarrow \quad S_i \quad \textbf{od},$$

and notice that $\mathcal{A}_P(P) \Leftrightarrow wrp.\mathcal{A}.P$.

Other properties of $\mathcal{A}_P$ include

- if $Q \Rightarrow \neg\bigvee_{i=1}^{n} g_i$, then $\mathcal{A}_Q(P) \Leftrightarrow \mathcal{A}(P)$; in particular, $\mathcal{A}_{false}(false) \Leftrightarrow \mathcal{A}(false)$; and
- $\mathcal{A}_{true}(true) \Leftrightarrow true$.

Since we have already developed results for the weakest precondition, we save a great deal of work by using $\mathcal{A}_P$ instead of wrp. Thus we have the following definition (we will deal with data refinement in Subsection 6.4):

Definition 28 Refinement for Non-Terminating Action Systems. A non-terminating action system $\mathcal{A}$ *is refined by* a non-terminating action system $\mathcal{B}$ (notation $\mathcal{A} \leq \mathcal{B}$) if for all $P : Pred\mathcal{A}$,

$$\mathcal{A}(P) \Rightarrow \mathcal{B}(P) \ \wedge\ \mathcal{A}_P(P) \Rightarrow \mathcal{B}_P(P).$$

□

Note that this definition is valid only for action systems, and not for arbitrary statements, for which we always use Definition 1. In particular, a non-terminating action system may not be a statement within another action system since we insist that all statements are executed atomically.

The first conjunct is a condition on the *final* state of the program, while the second conjunct is a condition on the *intermediate* state of the program. Definition 28 imposes a partial order on action systems and is stronger than Definition 1 since the first conjunct in the new definition is exactly

Definition 1. This first conjunct necessary in the case of action systems which terminate, otherwise we could have a refinement of a terminating system by another system terminating in a different state, but whose intermediate states contain those of the first system. For example, we would have

$$\textbf{do } n < 10 \rightarrow n := n + 1 \textbf{ od} \;\leq\; \textbf{do } n < 100 \rightarrow n := n + 1 \textbf{ od}.$$

Definition 28 could be weakened in order to give similar results to Definition 1 when applied to terminating action systems, while still giving meaningful results in the case of non-terminating action systems. Namely, we could modify Definition 28 by considering only predicates which mention certain variables we wish to regard as global. Alternatively, we could make a copy of all the original variables in the initialisation of the concrete action system, use them in all calculations, and finally copy them back to the original set of variables at the end of the calculation. The copied variables are not taken into account when considering refinement since they only exist in the concrete action system.

6.3 Fairness

We need to consider fairness even for non-terminating action systems. In the following example, we exhibit non-terminating action systems where fairness is required to show refinement, despite not being required in closely related terminating action systems.

Example 5 One-Place Buffer. We will obtain a one-place buffer by introducing an extra channel, *mid*, into a specification where data are transferred from an input channel *in* to an output channel *out*. Consider the action systems

$$\begin{array}{llllll}
\mathcal{A} \;\widehat{=} & \textbf{var} & in : seq_\omega Data, out : seqData & & & \\
& \textbf{init} & out = \emptyset & & & \\
& \textbf{do} & in \neq \emptyset & \rightarrow & in, out := tl(in), out \frown hd(in) & (A) \\
& \textbf{od} & & & &
\end{array}$$

and

$$\begin{array}{llllll}
\mathcal{B} \;\widehat{=} & \textbf{var} & in : seq_\omega Data, mid, out : seqData & & & \\
& \textbf{init} & mid, out = \emptyset, \emptyset & & & \\
& \textbf{do} & in \neq \emptyset & \rightarrow & in, mid := tl(in), mid \frown hd(in) & (B1) \\
& & [\,]\; mid \neq \emptyset & \rightarrow & mid, out := tl(mid), out \frown hd(in) & (B2) \\
& \textbf{od}, & & & &
\end{array}$$

where $\emptyset$ is the empty list and $\frown$ denotes catenation.

In determining whether or not $\mathcal{A} \leq \mathcal{B}$, the first conjunct in Definition 28 is unimportant because the system never terminates. (This is because the channel in is infinite.) Calculation of the second conjunct yields, for

$$P \ \widehat{=}\ in = a \wedge out = b,$$

$\mathcal{A}_P(P) \Leftrightarrow in = b \frown a$ and

$$\mathcal{B}_P(P) \Leftrightarrow b = \emptyset \wedge (in = a \vee (mid = \emptyset \wedge tl(in) = a)).$$

We do not have $\mathcal{A}_P(P) \Rightarrow \mathcal{B}_P(P)$, so $\mathcal{A}$ is not refined by $\mathcal{B}$. The problem lies in the lack of fairness in $\mathcal{B}$: $(B1)$ may execute indefinitely, never guaranteeing that the channel out receives any data.

However,

$$\begin{array}{lll} \mathcal{B}' \ \widehat{=} & \mathbf{var} & in : seq_\omega Data, mid, out : seqData \\ & \mathbf{init} & mid, out = \emptyset, \emptyset \\ & \mathbf{do} & \quad in \neq \emptyset \ \rightarrow \ in, mid := tl(in), mid \frown hd(in) \qquad (B1) \\ & & [\!]\ mid \neq \emptyset \ \rightarrow \ mid, out := tl(mid), out \frown hd(in) \quad (B2) \\ & \mathbf{uf} & true : \{s; B1^\omega \mid s : seq\mathcal{B}\} \\ & \mathbf{od}, & \end{array}$$

refines $\mathcal{A}$.

Notice that if the channel in is finite, i.e. has type $seqData$ like the other channels, we obtain a refinement of terminating action systems which does not require fairness. □

With this motivation, we now define refinement for fair non-terminating action systems.

Definition 29. Let

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

We define

$$\begin{array}{lll} \mathcal{A}_P \ \widehat{=} & \mathbf{do} & [\!]_{i=1}^{n} \ \neg P \wedge g_i \ \rightarrow \ S_i \\ & & \mathbf{uf} \quad \neg P \wedge P_\alpha : uf'_\alpha \\ & & \qquad P : \emptyset \\ & \mathbf{od}, & \end{array}$$

where $uf'_\alpha \ \widehat{=}\ \{u[\neg P \wedge g_i \rightarrow S_i / g_i \rightarrow S_i] \mid u : uf_\alpha \wedge 1 \leq i \leq n\}$. □

With this definition of $\mathcal{A}_P$, we can use Definition 28 for refinement. Thus, by applying the relevant theorems to each conjunct in Definition 28, we conclude that our theorems for program development, Theorems 22, 24, 26 and their corollaries hold for non-terminating action systems as well.

6.4 Data Refinement

Definition 30 Data Refinement for Non-Terminating Systems. For non-terminating action systems $\mathcal{A}$ and $\mathcal{B}$, we say that $\mathcal{A}$ is *data-refined* by $\mathcal{B}$ (notation: $\mathcal{A} \leq_\varphi \mathcal{B}$) if for all $P : Pred\mathcal{A}$,

$$\varphi \circ \mathcal{A}(P) \Rightarrow \mathcal{B} \circ \varphi(P) \quad \wedge \quad \varphi \circ \mathcal{A}_P(P) \Rightarrow \mathcal{B}_{\varphi(P)} \circ \varphi(P).$$

□

Theorem 20 gives us the method of calculating the concrete action system given φ and the abstract action system.

7 Readers-Writers Problem

7.1 A Simple Specification

The readers-writers problem [BL93] provides a typical example of the movement in computer science towards implementing greater concurrency. Its task is to co-ordinate access to a file by many processes, where we distinguish between those processes wishing to make changes to the file, called *writers*, and those wishing merely to access the file without making changes, called *readers*. Access to the file is subject to the constraint that if a process is writing, no other process may have access to the file. On the other hand, if there are no writers, arbitrarily many processes may read the file.

We will specify the readers-writers problem as a non-terminating action system and implement solutions using the theorems about program development in the previous sections. We will see that to achieve refinement, we will need some fairness assumptions.

The action system

$$\begin{array}{lrcl}
\textbf{var} & \multicolumn{3}{l}{readers : \mathbb{N} \cup \{-1\}} \\
\textbf{init} & \multicolumn{3}{l}{readers = 0} \\
\textbf{do} & true & \rightarrow & readers := readers + 1 \\
 & [\!]\ readers \geq 0 & \rightarrow & readers := readers - 1 \\
\textbf{od} & & &
\end{array}$$

captures the informal specification of the readers-writers problem mentioned above. If non-negative, the variable *readers* states the number of readers, but if its value is -1, then there is one writer. The first action in the action system corresponds to a new reader accessing the file or a writer ceasing to access it, while the second action corresponds to a writer beginning to write or a reader ceasing to read.

7.2 Data Refinement: Individual Processes

We would like to be more explicit now about exactly which processes are reading or writing. First, we modify the specification using monotonicity (Theorem 24) to distinguish actions involving readers from those involving writers:

$$
\begin{array}{lrcll}
\textbf{var} & \multicolumn{4}{l}{readers : \mathbb{N} \cup \{-1\}} \\
\textbf{init} & \multicolumn{4}{l}{readers = 0} \\
\textbf{do} & readers \geq 0 & \rightarrow & readers := readers + 1 & (A') \\
 & [\,]\ readers = -1 & \rightarrow & readers := readers + 1 & (D') \\
 & [\,]\ readers > 0 & \rightarrow & readers := readers - 1 & (B') \\
 & [\,]\ readers = 0 & \rightarrow & readers := readers - 1 & (C') \\
\textbf{od} & & & &
\end{array}
$$

Now we introduce the finite set I containing the names of all processes wishing to access the file. In order to replace *readers* and *writers* in a data refinement, we also introduce rlh which contains the processes currently reading, or "read lock holders". Similarly, wlh contains the processes (at most one) currently writing, or "write lock holders". Define φ such that

$$
\begin{array}{rcl}
for N \geq 0, \quad \varphi(readers = N) & = & wlh = \emptyset \wedge |rlh| = N \\
\varphi(readers = -1) & = & |wlh| = 1 \wedge rlh = \emptyset,
\end{array}
$$

and otherwise distributes over the operations $\wedge$, $\vee$, $\neg$, $\exists$ and $\forall$. φ is or-distributive and monotonic, so we can apply Theorem 3 followed by Theorem 24 to obtain an action system which is a data refinement of the previous one:

$$
\begin{array}{llrcll}
\textbf{var} & & \multicolumn{4}{l}{rlh, wlh : \mathbb{P}(I)} \\
\textbf{init} & & \multicolumn{4}{l}{rlh, wlh = \emptyset, \emptyset} \\
\textbf{do} & [\,]_{i \in I} & wlh = \emptyset & \rightarrow & rlh := rlh \cup \{i\} & (A) \\
 & [\,]_{i \in I} & i \in rlh & \rightarrow & rlh := rlh \setminus \{i\} & (B) \\
 & [\,]_{i \in I} & rlh \cup wlh = \emptyset & \rightarrow & wlh := wlh \cup \{i\} & (C) \\
 & [\,]_{i \in I} & i \in wlh & \rightarrow & wlh := wlh \setminus \{i\} & (D) \\
\textbf{od} & & & & &
\end{array}
$$

7.3 Adding Queues

In the above action system, processes begin and end reading or writing with the choice of subsequent processes being made nondeterministically. Our implementation, on the other hand, should distinguish between a process *desiring* to access the file but being prevented because of the nature of the processes already accessing it, and a process which does not wish to access the file. Moreover, our implementation should explicitly provide some decision method by which waiting processes may access the file next. To model reality more closely, therefore, we reduce nondeterminism and introduce the concept of a *queue*, which will hold those processes *waiting* to read or write. These queues will be denoted by rq and wq for reading and writing respectively. We now show that the action system

$$
\begin{array}{llrcll}
\textbf{var} & & \multicolumn{4}{l}{rlh, rq, wlh, wq : \mathbb{P}(I)} \\
\textbf{init} & & \multicolumn{4}{l}{rlh, rq, wlh, wq = \emptyset, \emptyset, \emptyset, \emptyset} \\
\textbf{do} & []_{i \in I} & i \notin rlh \cup wlh & \rightarrow & rq := rq \cup \{i\} & (1) \\
 & []_{i \in I} & i \notin rlh \cup wlh & \rightarrow & wq := wq \cup \{i\} & (2) \\
 & []_{i \in I} & wlh = \emptyset \wedge i \in rq & \rightarrow & rlh, rq := rlh \cup \{i\}, rq \setminus \{i\} & (3) \\
 & []_{i \in I} & wlh \cup rlh = \emptyset \wedge i \in wq & \rightarrow & wlh, wq := wlh \cup \{i\}, wq \setminus \{i\} & (4) \\
 & []_{i \in I} & i \in rlh & \rightarrow & rlh := rlh \setminus \{i\} & (5) \\
 & []_{i \in I} & i \in wlh & \rightarrow & wlh := wlh \setminus \{i\} & (6) \\
 & \textbf{uf} & \multicolumn{4}{l}{true : \{s; (1 or 2)^{\omega} \mid s : seq\}} \\
\textbf{od} & & & & &
\end{array}
$$

refines the previous one. Since the disjunction of guards is $true$ in both this action system and the previous one, we can use Theorem 26 to add the action $true \rightarrow skip$. By Theorem 27, $true \rightarrow skip \leq (1) \; [] \; (2)$. Applying monotonicity (Theorem 24) to this, to $(A) \leq (3)$ and to $(C) \leq (4)$, and noticing that (B) and (D) are the same as (5) and (6) respectively, we obtain the above action system.

The specification of the unfair sequences prevents a process from adding itself repeatedly to the queue to read or write, thereby neglecting other actions. We can now strengthen the guards of (1) and (2) to $i \notin rlh \cup wlh \cup rq$ and $i \notin rlh \cup wlh \cup wq$ respectively by Corollary 25. This prevents a process from adding itself to the queue to read or write until it has finished reading or writing. As a result, all the unfair sequences $\{s; (1 or 2)^{\omega} \mid s : seq\}$ terminate, so we can replace the unfair sequences by $\emptyset$ in the action system. We have used fairness in order to obtain a concrete action system from an abstract one, where neither contain fairness restrictions. This is reminiscent of the role played by miracles in program development in [Mor88].

This implements a readers' priority solution: once one reader begins reading, other readers may join in without ever allowing a writer access to the file. We can use use Corollary 25 to implement other solutions merely by strengthening the guards of some actions. For example, replacing (3) by

$$[]_{i \in I}\ wlh \cup wq = \emptyset \wedge i \in rq \rightarrow rlh, rq := rlh \cup \{i\}, rq \setminus \{i\}$$

gives a writers' priority solution: no process waiting to read may do so if any process is waiting to write. Similarly, replacing (3) by

$$[]_{i \in I}\ wlh \cup rlh = \emptyset \wedge i \in rq \rightarrow rlh, rq := rlh \cup \{i\}, rq \setminus \{i\}$$

ensures that only one process accesses the file at any time. Of course, these implementations are not likely to be used in practice, but they demonstrate the power of Corollary 25.

Not only have we modelled reality more closely, we have introduced greater concurrency as we may regard actions (1), (2), (5) and (6) as being executed by the processes, while (3) and (4) are executed by some central co-ordinating process.

7.4 A More Just Solution

The previous implementations have either favoured waiting readers by giving them unconditional priority over waiting writers, or vice-versa. We now give a more just (we use the term informally) solution to the readers-writers problem by giving a waiting writer the chance to access the file even though there may be waiting readers who joined the queue recently. Similarly, waiting readers are given the chance to access the file even though there may be waiting writers.

We introduce new variables rq' and rqq to replace rq in a data refinement with $rq = rq' \cup rqq$. rq' will contain those processes waiting to read that will be allowed to read as soon as possible, whereas rqq will hold processes that will have to wait at least until all those in rq' have begun reading. We introduce the boolean variable $unld$ which determines if processes joining the queue to read are added to rq' or rqq.

More formally, we first introduce $unld$ as an auxiliary variable using Theorems 26 and 27, and then strengthen the guards by Corollary 25. With this justification, we add the following actions to the action system:

$$\begin{array}{rcll} wlh = \emptyset \wedge \neg unld & \rightarrow & unld := true & (3A) \\ unld & \rightarrow & unld := false & (3C) \end{array}$$

As a next step (which we could have done before introducing $unld$), we data-refine the action system using φ defined by $\varphi(rq) = rq' \cup rqq$ and making all operations distributive. Since

(1)

$=$

$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq \rightarrow rq := rq \cup \{i\}$

$=$

$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq \rightarrow rq := rq \cup \{i\}$
$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq \rightarrow rq := rq \cup \{i\}$

$\leq_\varphi$ data refinement

$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq' \cup rqq \rightarrow rq' := rq' \cup \{i\}$
$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq' \cup rqq \rightarrow rqq := rqq \cup \{i\}$

$\leq$ strengthening guards

$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq' \cup rqq \wedge \neg unld \rightarrow rq' := rq' \cup \{i\} \qquad (1A)$
$[\!]_{i \in I}\ i \notin rlh \cup wlh \cup rq' \cup rqq \wedge unld \rightarrow rqq := rqq \cup \{i\} \qquad (1B)$

and

(3)

$=$

$[\!]_{i \in I}\ wlh = \emptyset \wedge i \in rq \rightarrow rlh, rq := rlh \cup \{i\}, rq \setminus \{i\}$

$\leq_\varphi$ data refinement

$[\!]_{i \in I}\ wlh = \emptyset \wedge i \in rq' \cup rqq \rightarrow rlh, rq' \cup rqq := rlh \cup \{i\}, rq' \cup rqq \setminus \{i\}$

$\leq$ strengthening guard

$[\!]_{i \in I}\ unld \wedge wlh = \emptyset \wedge i \in rq' \rightarrow rlh, rq' := rlh \cup \{i\}, rq' \setminus \{i\} \qquad (3B)$

and

$(3C)$

$=$

$unld \rightarrow unld := false$

$\leq$

$unld \rightarrow rq, unld := rq, false$

$\leq_\varphi$ data refinement

$unld \rightarrow rq', rqq, unld := rqq, rq', false$

$\leq$ strengthening guard

$rq' = \emptyset \wedge unld \rightarrow rq', rqq, unld := rqq, \emptyset, false \qquad (3C')$

we can apply Theorem 24 to obtain the new implementation in Figure 2 by replacing (1) by $(1A) \; [] \; (1B)$ and (3) by $(3A) \; [] \; (3B) \; [] \; (3C')$. Note that

$$
\begin{array}{lllll}
\textbf{var} & \multicolumn{4}{l}{rlh, rq', rqq, wlh, wq : \mathbb{P}(I); unld : \mathbb{B}} \\
\textbf{init} & \multicolumn{4}{l}{rlh, rq', rqq, wlh, wq = \emptyset, \emptyset, \emptyset, \emptyset, \emptyset; unld = false} \\
\textbf{do} & []_{i \in I} & i \notin rlh \cup wlh \cup rq' \cup rqq & & \\
 & & \qquad \wedge \neg unld & \rightarrow \quad rq' := rq' \cup \{i\} & (1A) \\
 & []_{i \in I} & i \notin rlh \cup wlh \cup rq' \cup rqq & & \\
 & & \qquad \wedge unld & \rightarrow \quad rqq := rqq \cup \{i\} & (1B) \\
 & []_{i \in I} & i \notin rlh \cup wlh & \rightarrow \quad wq := wq \cup \{i\} & (2) \\
 & [] & wlh = \emptyset \wedge \neg unld & \rightarrow \quad unld := true & (3A) \\
 & []_{i \in I} & i \in rq' \wedge wlh = \emptyset \wedge unld & \rightarrow & \\
 & & & rlh, rq' := rlh \cup \{i\}, rq' \setminus \{i\} & (3B) \\
 & [] & rq' = \emptyset \wedge unld & \rightarrow & \\
 & & & rq', rqq, unld := rqq, \emptyset, false & (3C') \\
 & []_{i \in I} & wlh \cup rlh = \emptyset \wedge i \in wq & \rightarrow & \\
 & & & wlh, wq := wlh \cup \{i\}, wq \setminus \{i\} & (4) \\
 & []_{i \in I} & i \in rlh & \rightarrow \quad rlh := rlh \setminus \{i\} & (5) \\
 & []_{i \in I} & i \in wlh & \rightarrow \quad wlh := wlh \setminus \{i\} & (6) \\
 & \textbf{uf} & \multicolumn{3}{l}{\{s; (3A, 3C')^{\omega} \mid s : seq\}} \\
\textbf{od}. & & & &
\end{array}
$$

Figure 2: Giving writers a chance

the disjunction of the guards is still *true* – as is necessary for the correct application of Theorem 24. The unfair sequences as specified prevent the alternate execution of $(3A)$ and $(3C')$ indefinitely, thereby neglecting all other actions. Thus, we prevent *unld* flipping between *true* and *false* when $rq' = \emptyset$ and $wlh = \emptyset$.

The action system in Figure 2 prevents waiting readers from reading if *unld* is *true* (see the guard of $(3B)$), which gives waiting writers a chance to write as long as $(3A)$ is not executed. The centralised co-ordinating process is responsible for executing $(3A)$, $(3B)$, $(3C')$ and (4).

7.5 A Very Just Solution

The action system of the previous section is not an entirely just implementation, since it merely ensures that every so often there is a nondeterministic choice between waiting readers and waiting writers. To reduce nondeterminism and ensure a just choice with readers and writers accessing the file alternately, we introduce a new variable $flag$ which has the value r or w, thus determining if the readers' or the writers' turn is next. Formally,

$$
\begin{array}{ll}
 & (3A) \\
= & \\
 & wlh = \emptyset \wedge \neg unld \rightarrow unld := true \\
\leq & \hfill \text{Theorem 27} \\
 & wlh = \emptyset \wedge \neg unld \rightarrow flag, unld := w, true \\
\leq & \hfill \text{strengthening guard} \\
 & flag = r \wedge wlh = \emptyset \wedge \neg unld \rightarrow flag, unld := w, true \qquad (3A')
\end{array}
$$

and

$$
\begin{array}{ll}
 & (4) \\
= & \\
 & []_{i \in I}\ wlh \cup rlh = \emptyset \wedge i \in wq \rightarrow wlh, wq := wlh \cup \{i\}, wq \setminus \{i\} \\
\leq & \hfill \text{Theorem 27} \\
 & []_{i \in I}\ wlh \cup rlh = \emptyset \wedge i \in wq \rightarrow wlh, wq, flag := wlh \cup \{i\}, wq \setminus \{i\}, r \\
\leq & \hfill \text{strengthening guard} \\
 & []_{i \in I}\ \neg unld \wedge flag = w \wedge wlh \cup rlh = \emptyset\ \wedge i \in wq \\
 & \qquad \rightarrow wlh, wq, flag := wlh \cup \{i\}, wq \setminus \{i\}, r \qquad (4A).
\end{array}
$$

Also, we add the action

$$\neg unld \wedge flag = w \wedge wlh \cup rlh \cup wq = \emptyset \rightarrow flag := r \qquad (4B)$$

by Theorem 26, Theorem 27 and strengthening the guard. After these steps, we obtain the "just" implementation in Figure 3. Notice that the disjunction of the guards is still $true$, so we have applied Theorem 24 correctly. The unfair sequences prevent $flag$ and $unld$ from changing their values when $wlh \cup rlh \cup rq' \cup wq = \emptyset$ while other statements are neglected indefinitely. Notice that $flag$ can only be changed from w to r if there are no waiting writers $(4B)$ or a writer has begun writing $(4A)$. Changing $flag$ from r to w also means setting $unld$ to $true$ $(3A')$, which allows any waiting readers to read.

Finally, since $unld \Rightarrow wlh = \emptyset$ is an invariant of the action system, we can replace $(3B)$ by

$$[]_{i \in I}\ i \in rq' \wedge unld \rightarrow rlh, rq' := rlh \cup \{i\}, rq' \setminus \{i\} \qquad (3B')$$

$$
\begin{array}{lllcll}
\textbf{var} & & rlh, rq', rqq, wlh, wq : \mathbb{P}(I); unld : \mathbb{B}; flag : \{w, r\} & & & \\
\textbf{init} & & rlh, rq', rqq, wlh, wq = \emptyset, \emptyset, \emptyset, \emptyset, \emptyset; unld = false & & & \\
\textbf{do} & \square_{i \in I} & i \notin rlh \cup wlh \cup rq' \cup rqq & & & \\
 & & \quad \wedge \neg unld & \rightarrow & rq' := rq' \cup \{i\} & (1A) \\
 & \square_{i \in I} & i \notin rlh \cup wlh \cup rq' \cup rqq & & & \\
 & & \quad \wedge unld & \rightarrow & rqq := rqq \cup \{i\} & (1B) \\
 & \square_{i \in I} & i \notin rlh \cup wlh & \rightarrow & wq := wq \cup \{i\} & (2) \\
 & \square & flag = r \wedge wlh = \emptyset \wedge \neg unld & \rightarrow & flag, unld := w, true & (3A') \\
 & \square_{i \in I} & i \in rq' \wedge wlh = \emptyset \wedge unld & \rightarrow & & \\
 & & & & rlh, rq' := rlh \cup \{i\}, rq' \setminus \{i\} & (3B) \\
 & \square & rq' = \emptyset \wedge unld & \rightarrow & & \\
 & & & & rq', rqq, unld := rqq, \emptyset, false & (3C') \\
 & \square_{i \in I} & \neg unld \wedge flag = w \wedge wlh \cup rlh = \emptyset \wedge i \in wq & & & \\
 & & \quad \rightarrow wlh, wq, flag := wlh \cup \{i\}, wq \setminus \{i\}, r & & & (4A) \\
 & \square & \neg unld \wedge flag = w & & & \\
 & & \quad \wedge wlh \cup rlh \cup wq = \emptyset & \rightarrow & flag := r & (4B) \\
 & \square_{i \in I} & i \in rlh & \rightarrow & rlh := rlh \setminus \{i\} & (5) \\
 & \square_{i \in I} & i \in wlh & \rightarrow & wlh := wlh \setminus \{i\} & (6) \\
 & \textbf{uf} & \{s; (3A', 3C', 4B)^{\omega} \mid s : seq\} & & & \\
\textbf{od}. & & & & &
\end{array}
$$

Figure 3: The final solution

We have not shown that this is a just implementation in the sense that any process which wants to access the file eventually will, merely that this implementation meets the specification.

7.6 Further Refinements

As a further possible refinement, one could replace the sets rq', rqq and wq by sequences, which would serve to introduce the notion of "first come first served" amongst the readers and writers. Alternatively, one could attach priorities to processes. Either of these steps is a simple data refinement.

8 Conclusion

In this paper, we have developed a theory of the fair execution of action systems exhibiting bounded nondeterminism using weakest precondition semantics. Our approach is valid for all kinds of fairness, not just weak and strong fairness. In order to facilitate program development, we have derived theorems about monotonicity and adding skips, the former analogous to an existing result for unfair action systems.

We have related our definition of refinement and data refinement of non-terminating action systems to the existing definitions for terminating action systems. We have shown that with our definition, the results we have derived for the program development of fair terminating action systems also hold for fair non-terminating action systems. We have illustrated these results by our development of solutions to the readers-writers problem.

References

[Bac88] R.J.R. Back, *A Calculus of Refinements for Program Derivations*, Acta Informatica, Vol.25, 1988

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

[Bac90] R.J.R. Back, *Refinement Calculus, Part II: Parallel and Reactive Programs*, LNCS 430, Springer-Verlag, 1990

[BvW89] R.J.R. Back, J. von Wright, *A Lattice-theoretical Basis for a Specification Language*, LNCS 375, Springer-Verlag, 1989

[BL93] A.J. Bernstein, P.M. Lewis, *Concurrency in Programming and Database Systems*, Jones and Bartlett, 1993

[BM95] M.J. Butler, C.C. Morgan, *Action Systems, Unbounded Nondeterminism, and Infinite Traces*, Formal Aspects of Computing, Vol. 7, 1995

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

[BX95] R.J.R. Back, Q.W. Xu, *Fairness in Action Systems*, unpublished, 1995

[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

[Fra86] N. Francez, *Fairness*, Springer-Verlag, 1986

[GM91] P. Gardiner, C.C. Morgan, *Data Refinement of Predicate Transformers*, Theoretical Computer Science, Vol. 87, 1991

[Hes92] W.H. Hesselink, *Programs, Recursion and Unbounded Choice*, Cambridge University Press, 1992

[Hes95] W.H. Hesselink, *Safety and Progress of Recursive Procedures*, Formal Aspects of Computing, Vol. 7, 1995

[Hoa85] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985

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

[Men75] B. Mendelson, *Introduction to Topology*, Allyn and Bacon, 1975

[MM95] C.C. Morgan, A. McIver, *Unifying wp and wlp*, Technical Report TR-23-95, Programming Research Group, Oxford University Computing Laboratory, 1995

[MMSS95a] C.C. Morgan, A. McIver, K. Seidel, J.W. Sanders, *Probabilistic Predicate Transformers*, Technical Report TR-4-95, Programming Research Group, Oxford University Computing Laboratory, 1995

[MMSS95b] C.C. Morgan, A. McIver, K. Seidel, J.W. Sanders, *Proof Rules for Probabilistics Loops*, unpublished, 1995

[Mor88] C.C. Morgan, *Data Refinement by Miracles*, Information Processing Letters, Vol. 26, 1988

[Mor89] C.C. Morgan, *Programming from Specifications*, Prentice-Hall, 1989

[Mor87] J.M. Morris, *Stepwise Refinement and the Programming Calculus*, Science of Computer Programming, Vol. 9, 1987

[Mor90] J.M. Morris, *Temporal Predicate Transformers and Fair Termination*, Acta Informatica, Vol. 27, 1990

[MP92] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag, 1992

[Nel89] G. Nelson, *A Generalization of Dijkstra's Calculus*, ACM Transactions on Programming Languages and Systems, Vol. 11, 1989

[Spi89] J.M. Spivey, *The Z Notation*, Prentice-Hall, 1989

[Tan88] A.S. Tanenbaum, *Computer Networks*, 2nd ed, Prentice-Hall, 1988

[Xu92] Q.W. Xu, D.Phil. Thesis, Oxford University Computing Laboratory, 1992