Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.J.M.M. Rutten, J.I. Zucker

A semantic approach to fairness

bq D13 , bq F12

A Semantic Approach to Fairness

Jan Rutten *

Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Jeffery Zucker **

Department of Computer Science
State University of New York at Buffalo, USA

In the *semantic* framework of metric process theory, we undertake a general investigation of *fairness* of processes from two points of view: (1) *intrinsic fairness* of processes, and (2) *fair operations* on processes. Regarding (1), we shall define a "fairification" operation on processes called *Fair* such that for every (generally unfair) process *p* the process *Fair(p)* is fair, and contains precisely those paths of *p* that are fair. Its definition uses systematic alternation of random choices. The second part of this paper treats the notion of fair operations on processes: suppose given an operator on processes (like merge, or infinite iteration), we want to define a fair version of it. For the operation of infinite iteration we define a fair version, again by a "fair scheduling" technique.

1980 Mathematics Subject Classification: 68B10, 68C01, 68C05.
1986 Computing Review Categories: D.1.3, F.1.2.
Key Words & Phrases: fairness, semantic domains of metric processes, fair infinite iteration, alternation of random choices.

1. INTRODUCTION

The most basic context in which the notion of *fairness* can be defined is that of a repetitive choice among alternatives. In [F] the reader can find an elaborate introduction to the notion(s) of fairness, with an extensive overview of the research in this area. Here "fairness" means that, in having to choose repeatedly among alternatives, no alternative will be postponed forever. Usually a nondeterministic programming language is taken as the context for such a study, especially the language of *guarded commands* ([D]).

In this paper we propose a different approach, which could be called a *semantic* one, as opposed to the *language* (or *syntax*) *directed* approach mentioned above. Our point of departure is a *semantic domain* for nondeterministic languages in general, without limiting ourselves to the choice of a particular language. Such a semantic domain will in general be a solution of some reflexive *domain equation*

$$FP \cong P,$$

where F is a functor on some category of mathematical domains, and "$\cong$" means "is isomorphic to". Various techniques have been developed for solving this type of equation. We follow a *metric*

approach, introduced by De Bakker and Zucker in [BZ1], and reformulated and extended in a category-theoretic setting in [AR]. The category $\mathcal{C}$ under consideration consists of complete metric spaces, and the functors on $\mathcal{C}$ are so-called *contracting functors*. These spaces are composed from basic metric spaces (sets provided with the trivial 0-1 metric) by the operations of union, Cartesian product, forming function spaces, and forming the set of all (closed) subsets of a given space. Examples would be complete metric spaces satisfying one of the following equations:

$$P \cong A \cup (B \times P), \quad \text{or}$$

$$P \cong A \cup (B \rightarrow (C \times P)),$$

where A, B and C are arbitrary sets and $\cong$ stands for "is isometric to". (Since elements of $\mathcal{C}$ are pairs $<P,d_P>$, consisting of a set P and a metric d_P on P, domain equations over $\mathcal{C}$ should also specify a condition on these metrics. In this introduction, however, we omit such details.)

Another example of a domain is a metric space P satisfying the domain equation:

$$P \cong \{p_0\} \cup \mathcal{P}_{cl}(B \times P).$$

(Here $\mathcal{P}_{cl}(\cdots)$ denotes the set of all closed subsets of $(\cdots)$.) Since this is the domain we shall use in this paper as a starting point for our study of fairness, we discuss it in some detail. The (possibly infinite) set $B = \{b_1, b_2, \ldots\}$ is called the *alphabet* of P. The elements of P are called *processes*. A process $p \in P$ is either p_0, the so-called *nil* process, or a (closed) set of the form

$$p = \{<b_i, p_i> \mid <b_i, p_i> \in B \times P, i \in I\}$$

for some set I of indices. (Here the set I represents the *choice* among alternatives.) Then p can be regarded as a process that for each $i \in I$ can take a step b_i, and then continues with the process p_i (called the *resumption* of b_i). This is itself either p_0, indicating that the process p has terminated after performing step b_i, or again a (closed) set of possible next steps and corresponding resumptions.

Roughly, one can think of these processes as *tree-like* entities. However, there are some differences. Trees with a left branch labeled a and a right branch labeled b, and with a left branch labeled b and a right branch labeled a, are identified, and both are represented by $\{<a,p_0>, <b,p_0>\}$. A tree with only one branch labeled a is identified with a tree with two branches both labeled a. Furthermore, we do not consider arbitrary subsets of $B \times P$, but only closed ones. For an extensive comparison of trees and processes we refer to [BK].

In our approach the elements of B, which are called *basic steps*, are atomic actions, whose possible interpretations have been abstracted from. One such interpretation would be to associate a basic step b_i with each component of a *guarded command*, indicating that the i-th component of that command is selected. Another interpretation would be to regard b_i as an arbitrary action of the i-th component of a system of (possibly infinitely many) active components, indicating that "progress" is being made by that component. A context in which this interpretation makes sense is that of *object-oriented programming* (see e.g. [ABKR]). The basic steps could also be thought of as being different possible actions (e.g. read, write, assignment, etc.) which a single component can perform.

In this framework of metric process theory, we undertake a general investigation of *fairness* of processes from two points of view: (1) *intrinsic fairness* of processes, and (2) *fair operations* on processes.

Regarding (1), a process p is called (*intrinsically*) *fair* if all its paths are fair. A *path* for p is a sequence of pairs: $<a_1,p_1>, <a_2,p_2>, \ldots$, such that $<a_1,p_1> \in p$ and $<a_{i+1},p_{i+1}> \in p_i$ for all $i \geqslant 1$. The difference between fair and unfair paths can easily be illustrated with a simple example: consider a process $p \in P$ satisfying

$$p = \{<0,p>, <1,p>\}.$$

This process must choose infinitely often (in fact at every step) whether to perform the basic step "0" or the basic step "1". The following path in p

$$<0,p>, <0,p>, <0,p>, \ldots$$

is unfair (with respect to basic step "1"), because step "1" can be taken infinitely often, but never is. An example of a fair path is

$$<0,p>, <1,p>, <0,p>, <1,p>, \ldots$$

There are actually two notions (at least) of fairness current in the literature. The notion we are considering in this paper is often called "strong" fairness (e.g. in [OA]), as opposed to "weak" fairness. In our context a path π would be called weakly fair if every basic step that is from some moment on *continually* enabled in π occurs infinitely often in π. (For the definition of *enabled* see 2.3.) This notion is also called *justice* ([LPS]). A path is *strongly fair* if every basic step that is enabled *infinitely often* (but not necessarily continually) in π occurs infinitely often in π.) The difference between these two notions can again be illustrated with a simple example: consider a process $p \in P$ satisfying

$$p = \{<0, \{<0,p>\}>, <1, \{<1,p>\}>\}.$$

This process can choose infinitely often whether to perform twice the basic step "0", or twice the basic step "1". Then the path in p

$$<0, \{<0,p>\}>, <0,p>, <0, \{<0,p>\}>, <0,p>, \ldots$$

is weakly fair but *not* strongly fair. We do not consider weak fairness further in this paper.

We shall define in section 3 (for a finite alphabet B) a "fairification" operation

$$\textbf{\textit{Fair}}: P \to P^{Ind}$$

(where P^{Ind} is a suitably extended version of P), such that the process $\textbf{\textit{Fair}}(p)$ is fair, and contains precisely those paths of p that are fair, or, more precisely, representatives of such paths. The relation between $\textbf{\textit{Fair}}(p)$ and p will be clarified by the definition of a mapping from the paths of $\textbf{\textit{Fair}}(p)$ to those paths of p which they represent. Roughly, $\textbf{\textit{Fair}}(p)$ is defined by associating *indices* with the subprocesses (or "*nodes*") of p so as to provide a "bookkeeping" of the way in which alternative subprocesses are chosen in forming paths. These indices indicate priorities for each of the basic steps b_j. During the construction of $\textbf{\textit{Fair}}(p)$, new sets of indices will from time to time be chosen by certain random choices. (This idea of implementing fair scheduling by means of systematic alternation of random choices is well known (see e.g. [AO], [BZ2,3], [P]).) In section 4 this theory is extended to an infinite alphabet, with an "expanding" system of indices (i.e. increasing in length), so that an index at a node records all the (finitely many) basic steps already encountered on the path to that node.

We turn now to (2), the notion of *fair operations* on processes. Suppose given an operation Θ on processes, which is, say, binary: $\Theta: P \times P \to P$. We want to define a fair version $\Theta_f: P \times P \to P$ of Θ, such that for all $p_1, p_2 \in P$: first, if p_1 and p_2 are (intrinsically) fair, then so is $\Theta_f(p_1, p_2)$; and second, $\Theta_f(p_1, p_2)$ is *fair with respect to the operation* Θ. This second condition must be explicated for each operation Θ. A good example is the *merge* operation $\|: P \times P \to P$. In [BZ2,3] a fair version $\|_f$ is defined. In this case the second condition is the requirement that all paths in the resulting process $p_1 \|_f p_2$ must be fair with regard to *alternate scheduling* from p_1 and p_2. A trivial and wrong solution to the problem would be to define

$$p_1 \|_f p_2 = \textbf{\textit{Fair}}(p_1 \| p_2).$$

Obviously, the first condition would be satisfied, but not so the second. The reason for this is, roughly, that in the resulting process $p_1 \|_f p_2$, (intrinsically) unfair paths of $p_1 \| p_2$ that *are* fair with respect to the alternate scheduling from p_1 and p_2 should still be present. The operation $\textbf{\textit{Fair}}$, however, would remove them from $p_1 \| p_2$. So this solution would be too coarse. A satisfactory solution was given in [BZ2,3], where the fair merge was defined on the basis of alternate sequences of random choices.

In this paper (section 5) we shall consider another example of an operation on processes, namely *infinite iteration* $(\cdots)^\omega: P \to P$, defined by

$$p^\omega = \lim_{n\to\infty} p^n,$$

where $p^0 = p_0$ and $p^{n+1} = p^n \circ p$. (Here "$\circ$" stands for sequential composition of processes.) We define the *fair* infinite iteration p^{ω_f} of a process $p \in P$ and, after explicating the notion of *fairness with respect to infinite iteration*, prove that the conditions above are indeed satisfied.

An area that remains to be investigated is that of fairness for *non-uniform* processes [BZ1], where our uninterpreted basic actions are replaced by basic state transformations, since here even the *definition* of fairness of paths in such processes is problematic.

ACKNOWLEDGEMENTS: It was Jaco de Bakker who first noticed that fair scheduling, implemented by systematic alternation of random choices (as in [P]), could be used to model fair merge in the semantic framework of process domains, as in [BZ2,3]. The second author had useful discussions with Shenquan Xie on fairification and fair infinite iteration.

2. MATHEMATICAL PRELIMINARIES

DEFINITION 2.1 (Domains)
We shall use mathematical domains P of processes p, which are such that:
(1) P is a complete metric space,
(2) P satisfies the following reflexive equation:

$$P \cong \{p_0\} \cup \mathcal{P}_{cl}(A \times P),$$

where $\cong$ stands for "is isometric to", p_0 is a null process, $\mathcal{P}_{cl}(\cdots)$ denotes the set of all closed subsets of $(\cdots)$ and A, with typical elements a, is such that it contains as a subset a (possibly infinite) alphabet

$$B = \{b_1, b_2, \ldots\}$$

of *basic steps*.

We shall not dwell too long upon the mathematical details of the construction of a domain P which satisfies the above definition. Let us just briefly mention two different approaches. First, one can take the metric *completion* of the union of metric spaces $P_0 \subset P_1 \subset \cdots$ defined inductively by

$$P_0 = \{p_0\},$$

$$P_{n+1} \cong \{p_0\} \cup \mathcal{P}_{cl}(A \times P_n).$$

(The metric on P_0 is trivial, the metric on P_{n+1} can be defined using the metric on P_n.) For this method, full mathematical details and extensive motivation are supplied in [BZ1]. Secondly, one can interpret the reflexive equation for P as defining a functor F on a category of complete metric spaces, thus:

$$FP = \{p_0\} \cup \mathcal{P}_{cl}(A \times P).$$

(The definition of F should also specify a metric for FP.) In [AR] it is shown how to define F as a so-called *contraction*, which has a (unique) fixed point; so

$$FP \cong P.$$

Thus this method also presents us with a solution.

REMARK: We should be more precise about the metrics involved. We should have written the equation above like

$$FP = \{p_0\} \cup \mathcal{P}_{cl}(A \times id_{\frac{1}{2}}(P)),$$

where, for any positive real number c, id_c maps a metric space (M,d) onto (M,d') with $d'(x,y) = c \cdot d(x,y)$. For the details see [AR].

We now introduce a number of concepts related to processes.

DEFINITION 2.2 (Paths)
A *path* for a process $p \in P$ is a (finite or infinite) sequence

$$\pi = (<a_1,p_1>,<a_2,p_2>, \ldots)$$

such that

$$<a_1,p_1> \in p \wedge \forall i \geq 1 [<a_{i+1},p_{i+1}> \in p_i].$$

We say that π *passes through* p_i, and p_i will be called a *node* of p or a *subprocess* of p (for $i \geq 1$). The set of all paths for p will be called *Paths*(p).

The following definition explains which processes we want to consider *fair*.

DEFINITION 2.3 (Fairness)
(a) Let $b_i \in B$. Consider a path

$$\pi \equiv (<a_1,p_1>,<a_2,p_2>, \ldots).$$

We say that b_i is *enabled* in π (or i is enabled in π) whenever

$$\exists k \in \mathbb{N} \, \exists q \in P [<b_i,q> \in p_k].$$

If $<b_i,q> \in p_k$ we also say that b_i is enabled at step k. We say that b_i *occurs* in π, whenever

$$\exists k \in \mathbb{N} [a_k = b_i].$$

(b) We call a path π *fair* whenever for all $b_i \in B$, if b_i is enabled infinitely often in π, then it occurs infinitely often in π.
(c) A process $p \in P$ is called *fair* if all its paths are fair.

EXAMPLE: Let $p \in P$ be such that $p = \{<a,p>,<b,p>\}$. Then b is continually enabled in

$$\pi = (<a,p>, <a,p>, \ldots),$$

but never occurs in it. Thus, the path π is unfair.

Please note that only *basic steps* $b_i \in B$ are taken into account in the definition of fairness.

3. FAIRIFICATION OF PROCESSES WITH FINITE ALPHABET

Let P be defined by

$$P \cong \{p_0\} \cup \mathcal{P}_{cl}(B \times P),$$

with B a *finite* alphabet:

$$B = \{b_1, \ldots, b_m\}.$$

Given a process $p \in P$, we want to form a new process *Fair*(p), which is, in some sense, a fair version

of p. For this purpose we want to define a function

$$Fair: P \to P^{Ind}$$

such that there is an obvious correspondence between the paths of $Fair(p)$ and the *fair* paths of p. Here P^{Ind} is given by:

$$P^{Ind} = \{p_0\} \cup \mathcal{P}_{cl}(A \times P^{Ind}),$$

where $A = B \cup Index$, and *Index* is a set of indices (to be defined below). A node p' of a process $p \in P^{Ind}$ with

$$p' = \{<\nu, p_\nu> \mid \nu \in I\},$$

for some subset I of *Index*, is called a *sum node* and is denoted by

$$p' = \sum_{\nu \in I'} p_\nu.$$

After having defined the function *Fair*, we shall clarify the relation between p and $Fair(p)$ by defining a mapping

$$\Phi: Paths(Fair(p)) \to Paths(p),$$

that will satisfy the following two properties. First, for every path $\pi \in Paths(Fair(p))$ we have that $\Phi(\pi)$ is fair. Secondly, any fair path in p will be in the range of Φ. The function *Fair* will be defined in such a way that it transforms a process p into a fair process $Fair(p)$ by labeling each node of p with an *index* and, moreover, interspersing some new nodes consisting of *sums* of indices (to be defined below). Indices are the main building blocks in the definition of the function *Fair*. They are defined as follows.

DEFINITION 3.1 (Indices)
The set *Index* of indices, with typical elements ν, is given by

$$Index = \{<n_1^{s_1}, \ldots, n_m^{s_m}> \mid$$

$$\forall i \in \{1, \ldots, m\} \; [n_i \geqslant 0 \land 0 < s_i \leqslant \infty \land (n_i = 0 \Leftrightarrow s_i = \infty)]\},$$

where m is the number of elements in B, and $n_i^{s_i}$ denotes the Cartesian pair $<n_i, s_i>$.

Let p be a process and ν an index. The process p^ν, which is defined below, can be viewed, informally speaking, as a process that behaves like p as far as is allowed by the index ν. Consider the i-th element of ν, say $n_i^{s_i}$. It is related to b_i, the i-th element of our alphabet B. The interpretation of $n_i^{s_i}$ (relative to p) is that in paths starting in p, a step b_i is permitted n_i times with *priority* s_i.

For the priorities s_i we have the convention that a *low* number indicates a *high* priority. It is possible that two or more s_i's have the same value, the corresponding b_i's having the same priority. The symbol ∞ indicates the lowest priority possible. Because it is always associated with an n that is 0, it can also be interpreted as indicating no priority at all.

REMARK
The interpretation of the i-th component $n_i^{s_i}$ is in a sense orthogonal to the approach taken in e.g. [AO]. There a single number z_i is used to indicate the priority of the i-th component of some system of active components. This number z_i indicates, roughly, the number of times a computation can "allow itself" *not to choose* this component as the next one to make progress. In our approach the number n_i indicates the number of times we are allowed *to choose* b_i (the i-th component) as the next step, before another component gets the highest priority.

Now suppose we have a process p containing a step $<b_i, q>$:

7

$$p = \{\ldots, <b_i, q>, \ldots\};$$

and assume furthermore that we have $\nu \in \textbf{Index}$ with

$$\nu = <\ldots, n_i^{s_i}, \ldots>$$

where $n_i > 0$ and $s_i = \min\{s_1, \ldots, s_m\}$. Then, according to our interpretation of p'', it is permitted to choose $<b_i, q>$ as the first step of a path starting from p. With the resumption q of this step will be associated a new index $\nu^-[i]$, in which n_i is decreased by one. If $n_i > 1$ nothing happens to the priority s_i of b_i. If $n_i = 1$ (and so decreased to 0) it is, for the time being, the last time that b_i is allowed, and s_i is changed to ∞ (the lowest priority possible). As we will see, at some later stage it will be taken care of that n_i and s_i are reset again, so that $n_i > 0$ and $s_i < \infty$. All this is formalized in the following definition.

DEFINITION 3.2
Let $\nu \in \textbf{Index}$ be such that

$$\nu = <n_1^{s_1}, \ldots, n_i^{s_i}, \ldots, n_m^{s_m}>,$$

and let $i \in \{1, \ldots, m\}$. We define

$$\nu^-[i] = \begin{cases} <n_1^{s_1}, \ldots, (n_i - 1)^{s_i}, \ldots, n_m^{s_m}> & \text{if } n_i > 1 \\ <n_1^{s_1}, \ldots, 0^\infty, \ldots, n_m^{s_m}> & \text{if } n_i = 1 \\ \text{undefined} & \text{if } n_i = 0. \end{cases}$$

There is one other operation on indices we shall need.

DEFINITION 3.3
Let $\nu \in \textbf{Index}$ be such that

$$\nu = <n_1^{s_1}, \ldots, n_m^{s_m}>,$$

then

$$N(\nu) = \{<\tilde{n}_1^{\tilde{s}_1}, \ldots, \tilde{n}_m^{\tilde{s}_m}> \;|$$

$$\forall j \in \{1, \ldots, m\}\; [(n_j = 0 \wedge s_j = \infty) \Rightarrow (\tilde{n}_j > 0 \wedge \tilde{s}_j = s + 1) \wedge$$

$$(n_j > 0 \wedge s_j < \infty) \Rightarrow (\tilde{n}_j = n_j \wedge \tilde{s}_j = s_j)]\}$$

where $s = \max(\{s_1, \ldots, s_m\} \setminus \{\infty\})$.

The elements $\tilde{\nu}$ in $N(\nu)$ are obtained from ν by changing, for all i with $n_i = 0$ and $s_i = \infty$, the value of n_i to an arbitrary positive number and the value of s_i to $s + 1$. In words, this means that b_i is again allowed to be chosen ($\tilde{n}_i$ times) but with a priority lower than all other priorities present in ν that are not ∞. This definition will also be used in the definition of $\textbf{Fair}$, where it will be further elucidated. We now give this definition, upon which an explanation will follow.

DEFINITION 3.4 (Fairification)
We define a function

$$\textbf{Fair}: P \rightarrow P^{Ind}.$$

Let $p \in P$. Then

$$\textbf{Fair}(p) = \sum_{\nu \in I_0} \textit{fair}(p, \nu),$$

8

where

$$I_0 = \{ <n_1^1, \ldots, n_m^1> \mid n_i > 0, \ i = 1, \ldots, m \}$$

and

$$fair : P \times Index \rightarrow P^{Ind}$$

is defined as follows. (We often write p^ν for $fair(p, \nu)$.) For all $\nu \in Index$ we define

$$fair(p_0, \nu) = p_0.$$

For $p \neq p_0$ we distinguish two cases.

Case 1:

If $\exists i \in \{1, \ldots, m\} \, [\, n_i > 0 \wedge s_i < \infty \wedge enabled(i) \,]$,

then $p^\nu = \{ <b_j, q^{\nu^{\,[j]}}> \mid <b_j, q> \in p \wedge s_j = \min\{s_1, \ldots, s_m\} \}$.

Case 2:

If $\forall i \in \{1, \ldots, m\} \, [\, enabled(i) \Rightarrow (n_i = 0 \wedge s_i = \infty) \,]$,

then $p^\nu = \sum_{\tilde{\nu} \cdot N(\nu)} p^{\tilde{\nu}}$.

REMARKS

(1) The definition of $fair : P \times Index \rightarrow P^{Ind}$ is self-referential and therefore needs some justification. We observe that $fair$ could be defined as the fixed point of a mapping

$$\Phi : (P \times Index \rightarrow P^{Ind}) \rightarrow (P \times Index \rightarrow P^{Ind}),$$

which can be defined according to the definition scheme of $fair$ above. It is straightforward to see that such a definition yields a contracting function, which thus has a unique fixed point (cf. Banach's fixed point theorem ([BZ1], [AR])).

(2) Because case 2 never occurs twice in succession, $fair(p, \nu)$ never contains two sum nodes successively.

(3) Every node in $Fair(p)$ is either a sum node, or of the form $\{ <b_{i_j}, p_j> \mid j \in I \}$, for some set of indices I.

(4) We give some informal intuition for this definition. The indices $\nu \in Index$ in the definition above can be interpreted as strategies for the construction of a process $Fair(p)$ such that every path in this process will be fair with respect to every b_i in B. An element ν in I_0 can be regarded as permission, for each i, to choose b_i n_i times. All i are supplied at the beginning with the same priority, that is 1.

We will treat p^ν for the case that $p \neq p_0$. As long as case 1 applies there is no need to change our strategy or, in other words, to choose a new ν. Each b_i that is enabled at p, and for which $n_i > 0$ and $s_i = \min\{s_1, \ldots, s_m\}$, may be chosen as the next step in the new process we are constructing. The index ν is changed according to the definition of $\nu^-[i]$, so n_i is decreased by 1 and the priority s_i remains constant, unless n_i was 1. Then it is set to ∞, indicating no priority at all. Because every application of case 1 causes the decrease of an n_i, it is obvious that after a finite number of such applications case 2 must hold. For didactic purposes we shall now make a conceptual distinction between two possible situations that may arise in this case. Formally however, as may be inferred from the definition of case 2, this is not necessary.

First, it may be the case that all n_i's have been decreased to 0 (and all s_i's have been set to ∞). Then we can consider the strategy suggested by the ν we started with to be a great success: every b_i has been chosen the number of times we had in mind for it (n_i). The fact that originally all n_i's were strictly positive implies that so far we have made sure that all b_i's have been treated

fairly. It is clear what to do next: we can just restart by choosing a new index ν, with all n_i strictly positive and all s_i set to 1. According to the definition of $N(\nu)$, this is exactly what happens in this case.

The second situation is more typical. It concerns the case that for all i that are *enabled* at p, $n_i = 0$ and $s_i = \infty$. But we have not finished the strategy suggested by the original ν, because there exists at least one j *not* enabled at p, with $n_j > 0$ and $s_j < \infty$. Although we have not finished our first strategy, we are forced to change it because it does not tell us what to do about the i's that are enabled at p. A new strategy $\bar{\nu}$ is defined such that for all j with $n_j > 0$ and $s_j < \infty$ these values remain unchanged, thus preserving that part of the first strategy (ν) that has not yet been dealt with. For all other i (enabled or not enabled) the value of n_i is set to an arbitrary strictly positive number, and the value of s_i to $\max\{s_1, \ldots, s_m\} + 1$. So the new priority introduced here is lower than all the already existing priorities. When at a later stage one of the j's, for which n_j and s_j remain unchanged here, is enabled, it will take precedence over those i's for which a new priority is introduced. Thus a fair treatment of such j's is ensured for the future.

Now for the rest of this section let $p \in P$ be fixed. We define a mapping

$$\Phi: \textbf{Paths}(\textbf{Fair}(p)) \rightarrow \textbf{Paths}(p),$$

relating to each path π in $\textbf{Fair}(p)$ a fair path in p. For its formal definition we shall make use of the following lemma.

LEMMA 3.5
For all $\bar{p} \in P$ with $\bar{p} \neq p_0$, $\nu \in \textbf{Index}$ and $<a, q> \in \textbf{fair}(\bar{p}, \nu)$, there exist $p' \in P$ and $\nu' \in \textbf{Index}$ such that

$$q = \textbf{fair}(p', \nu') \wedge$$

$$a \in \textbf{Index} \Rightarrow p' = \bar{p} \wedge$$

$$a \in B \Rightarrow <a, p'> \in \bar{p}.$$

The proof is straightforward from the definition of $\bar{p}^\nu$ ($= \textbf{fair}(\bar{p}, \nu)$).

DEFINITION 3.6 (The mapping Φ)
Let

$$\pi = <a_0, q_0>, <a_1, q_1>, \ldots$$

be a path in $\textbf{Fair}(p)$. By the above lemma and the definition of $\textbf{Fair}(p)$ we can rewrite it as

$$\pi = <a_0, p^\nu>, <a_1, p_1^{\nu_1}>, \ldots,$$

for certain $\nu, \nu_1, \ldots \in \textbf{Index}$ and $p_1, p_2, \ldots \in P$. Now if we delete all pairs $<a_i, p_i^{\nu_i}>$ with $a_i \in \textbf{Index}$, and all superscripts ν_i, we get a sequence

$$\Phi(\pi) = <a_{i_1}, p_{i_1}>, <a_{i_2}, p_{i_2}>, \ldots,$$

which is a path in p. We call $\Phi(\pi)$ the path in p *corresponding* to the path π in $\textbf{Fair}(p)$. This defines a mapping

$$\Phi: \textbf{Paths}(\textbf{Fair}(p)) \rightarrow \textbf{Paths}(p).$$

Next, we have an important theorem.

THEOREM 3.7
$\textbf{Fair}(p)$ is fair. That is, for all $\pi \in \textbf{Paths}(\textbf{Fair}(p))$, π is fair.

PROOF

Let $\pi \in Paths(Fair(p))$ be such, that

$$\pi = <a_1,q_1>,<a_2,q_2>, \ldots$$
$$= <a_1,p_1^{v_1}>,<a_2,p_2^{v_2}>, \ldots ..$$

Suppose b_i is enabled infinitely often in π. We must show that b_i occurs infinitely often within π. It is sufficient to show that for any j, if b_i is enabled at the node $p_j^{v_j}$ of π, then b_i occurs further on in the path π, that is, for some $j' \geqslant j$: $b_i = a_{j'}$.

We consider the sequence $v_j, v_{j+1}, \ldots$ and observe that for every $k \in \mathbb{N}$, v_{k+1} is obtained from v_k by an application of case 1 or 2 in the definition of $fair(p,v)$ (definition 3.4). Now let

$$v_j = <n_1^{s_1}, \ldots, n_m^{s_m}>.$$

We consider all possible cases.

(1) $n_i = 0$:

Then $s_i = \infty$. For every application of case 1 (above) one of the n_k's must decrease. Therefore eventually case 2 must apply, which makes all n_k's positive and brings us to the next case.

(2) $n_i > 0$: This implies $s_i < \infty$. As long as s_i is not the highest priority, the following may happen. Any application of case 1 results in *either* the decrease of an n_k, not to 0, *or* the decrease of an n_k to 0 and the removal of a higher priority than s_i. After a finite number of applications of case 1, the latter must happen. Any application of case 2 introduces only priorities that are lower than s_i, and must be followed by an application of case 1. Furthermore, during any of these applications, n_i and s_i remain constant. It follows then that eventually s_i will be the highest priority. Because b_i is enabled infinitely often in π, it must be enabled at some step beyond this, at which point case 1 will be applied to it and b_i will occur at the next step.

Now that we have proved that we did not promise too much, that is to say that $Fair(p)$ indeed contains only fair paths, let us also make sure that for all fair paths in p there is a corresponding path in $Fair(p)$.

THEOREM 3.8
Any fair path in p is in the range of the mapping Φ.

PROOF

Given a fair path $\pi' \in Paths(p)$, we must construct a path $\pi \in Paths(Fair(p))$ such that

$$\Phi(\pi) = \pi'.$$

First, we partition the set $\{1, \ldots, m\}$ into two parts F and I, where F is the set of all i such that b_i is enabled finitely often (perhaps never) in π', and I is the set of all i such that b_i is enabled infinitely often in π'. Thus:

$$\{1, \ldots, m\} = I \cup F.$$

Note that for all $i \in F$, b_i occurs only finitely often in π', and for all $i \in I$, b_i occurs infinitely often in π', since π' is fair. Let $l_1 \in \mathbb{N}$ be so big that

(1) no b_i with $i \in F$ is enabled in the part of π' at or after step l_1;

(2) every b_i with $i \in I$ occurs at least once by then.

Now for $i = 1, \ldots, m$, let n_i' be the number of times that b_i occurs before (or at) step l_1 and then define

$$n_i = \begin{cases} n_i' + 1 & \text{if } i \in F \\ n_i' & \text{if } i \in I. \end{cases}$$

We define our first index ν_1 by

$$\nu_1 = <n_1^1, \ldots, n_m^1>.$$

Now we can construct the first part of the path π corresponding with the part of π' before step l_1, by starting with p^ν, and repeatedly applying case 1 for the appropriate b_i, thus decreasing the n_i's until (at step l_1) our index is such that for all $i \in \{1, \ldots, m\}$:

$$i \in F \Rightarrow n_i = 1 \wedge s_i = 1,$$

$$i \in I \Rightarrow n_i = 0 \wedge s_i = \infty.$$

Now case 2 must be applied to get a sum node, since no $i \in F$ is enabled at step l_1. To determine the following index ν_2 we again choose a number $l_2 \in \mathbb{N}$, with $l_2 > l_1$, such that every b_i with $i \in I$ occurs at least once between steps l_1 and l_2 (including l_1, excluding l_2). Then choose an index ν_2 such that, for $i \in I$, n_i denotes the number of occurrences of b_i between l_1 and l_2. We proceed as before, constructing the part of π' between l_1 and l_2. Continuing in this way, we construct a path π in *Fair(p)* such that $\Phi(\pi) = \pi'$.

REMARK: This function Φ is *not* bijective. In general there are more than one (in fact, infinitely many) paths in *Fair(p)* that are mapped by Φ to the same path in p.

4. FAIRIFICATION OF PROCESSES WITH INFINITE ALPHABET

We now want to extend our technique of fairification to a set of processes, which we shall (again) call P, defined by

$$P \cong \{p_0\} \cup \mathcal{P}_{cl}(B \times P),$$

with B an *infinite* alphabet:

$$B = \{b_1, b_2, \ldots\}.$$

We shall again define a function

$$\textit{Fair}: P \rightarrow P^{Ind},$$

where P^{Ind} is given by

$$P^{Ind} = \{p_0\} \cup \mathcal{P}_{cl}(A \times P^{Ind}),$$

$$A = B \cup \textit{Index},$$

with *Index* to be defined below. We shall repeat the approach of the previous section with some small but essential changes. The definitions, lemmas and theorems that need not be changed will be mentioned, but not repeated in full.

An important change is the new definition of indices. They no longer have a fixed length.

DEFINITION 4.1 (Indices)
The set *Index* of indices, with typical elements ν, is given by

$$\textit{Index} = \bigcup_{m \in \mathbb{N}} \textit{Index}^{[m]},$$

with

$$\textit{Index}^{[m]} = \{<n_1^{s_1}, \ldots, n_m^{s_m}> \mid \forall i \in \{1, \ldots, m\} \ [n_i \geq 0 \wedge 0 < s_i \leq \infty \wedge (n_i = 0 \Leftrightarrow s_i = \infty)]\}.$$

An index of length k is related to the first k elements of our alphabet B. The interpretation of n_i and priority s_i is as before. When we define, for a given process p, a fair version **Fair**(p), we shall, during the construction, increase the length of the indices used, thus considering fairness with respect to a growing number of basic steps b_i. Once the length of an index is bigger than or equal to some $i \in \mathbb{N}$, it is ensured that b_i is treated fairly thereafter. The definition of the first operation on indices, $v^-[\cdots]$, remains unchanged, but for the fact that the original definition (3.2) should hold for indices of arbitrary length. The most important adaptation of this section lies in the following new definition of $N(v)$.

DEFINITION 4.2
Let $v \in \mathbf{Index}$ be such that $v = <n_1^{s_1}, \ldots, n_m^{s_m}>$ and let $p \in P$. We define

$$N(v,p) = \{ <\tilde{n}_1^{\tilde{s}_1}, \ldots, \tilde{n}_{m'}^{\tilde{s}_{m'}}> \mid$$

$$m' > m \wedge$$

$$\{k \mid 1 \leqslant k \leqslant m' \wedge \tilde{n}_k > 0\} \cap \{k \mid k \text{ enabled at } p\} \neq \varnothing \wedge$$

$$\forall j \ [((1 \leqslant j \leqslant m) \wedge n_j = 0 \wedge s_j = \infty) \Rightarrow (\tilde{n}_j > 0 \wedge \tilde{s}_j = s + 1)) \wedge$$

$$((1 \leqslant j \leqslant m) \wedge n_j > 0 \wedge s_j < \infty) \Rightarrow (\tilde{n}_j = n_j \wedge \tilde{s}_j = s_j)) \wedge$$

$$m < j \leqslant m' \Rightarrow (\tilde{n}_j > 0 \wedge \tilde{s}_j = s + 1) \vee (\tilde{n}_j = 0 \wedge \tilde{s}_j = \infty))]\}$$

where $s = \max\{s_1, \ldots, s_m\}$.

Let us see how this definition is used in the definition of the function **Fair** below, and then try to comment on its intuitive interpretation. Although we do not change the definition of **Fair** (definition 3.4), we repeat its most interesting part and discuss it in the context of the altered definition of $N(v)$.

If $p \in P$ with $p \neq p_0$, then p^v $(=fair(p,v))$ is given by:
Case 1:

If $\exists i \in \{1, \ldots, length(v)\}$ $[n_i > 0 \wedge s_i < \infty \wedge enabled(i)]$,

then $p^v = \{ <b_j, q^{v^{[j]}}> \mid <b_j,q> \in p \wedge s_j = \min\{s_1, \ldots, s_{length(v)}\}\}$.

Case 2:

If $\forall i \in \{1, \ldots, length(v)\}$ $[enabled(i) \Rightarrow (n_i = 0 \wedge s_i = \infty)]$,

then $p^v = \sum_{\tilde{v} \cdot N(v,p)} p^{\tilde{v}}$.

The interpretation of case 1 is the same as before. When the condition of case 2 holds, we are obliged to change our strategy, that is to choose a new index, because our current strategy does not say anything about the i's that are enabled at p. This can have two reasons. For such an i we either have $n_i = 0$ and $s_i = \infty$ or $i > length(v)$. In order to be able to continue our construction, we therefore allow several new strategies $\tilde{v} \in N(v,p)$, which all must satisfy the following constraints. First, the part of the old strategy v that has not been dealt with yet has to be preserved: for $1 \leqslant i \leqslant length(v)$ with $n_i > 0$ and $s_i < \infty$ we have $\tilde{n}_i = n_i$ and $\tilde{s}_i = s_i$. Then, for $1 \leqslant i \leqslant length(v)$ with $n_i = 0$ and $s_i = \infty$, the values of n_i and s_i are reset: $\tilde{n}_i$ arbitrary positive, $\tilde{s}_i = 1 + s$. As in the finite case, the new priority is lower than the existing ones. Because we want each $b_k \in B$ eventually to be treated fairly, for each k there should be a moment in our construction where an index v is introduced with $length(v) > k$. Therefore we require the length of the new index $\tilde{v}$ to be strictly greater than the length of v. For the

newly introduced j's ($length(v) < j \leq m$) we require

$$(\tilde{n}_j > 0 \wedge \tilde{s}_j = s + 1) \vee (\tilde{n}_j = 0 \wedge \tilde{s}_j = \infty).$$

Although here $\tilde{n}_j = 0$ is allowed, we know that the next time that case 2 is applied $\tilde{n}_j$ will be set to a strictly positive value. The newcomers, so to speak, are granted one (and only one) moment of respite. The motivation for this generosity lies in the rather selfish wish to prove theorem 4.4. It appears that it would be too restrictive to demand for all such j that $\tilde{n}_j > 0$. Finally, the condition that

$$\{k \mid 1 \leq k \leq m \wedge \tilde{n}_k > 0\} \cap \{k \mid k \text{ enabled at } p\} \neq \emptyset$$

entails that case 2 can never occur twice in succession.

Now for the rest of this subsection let $p \in P$ be fixed. We define a mapping

$$\Phi: Paths(Fair(p)) \rightarrow Paths(p),$$

relating to each path π in $Fair(p)$ a fair path in p, in exactly the same way as in definition 3.6. We finally repeat theorems 3.7 and 3.8 of the previous section, which together show that the definition of $Fair(p)$ (using the new definition of $N(v,p)$) is satisfactory. The former proofs of these theorems have to be altered, as can be seen below.

THEOREM 4.3
$Fair(p)$ is fair. That is, for all $\pi \in Paths(Fair(p))$, π is fair.

PROOF
Let $p \in P$ and let $\pi \in Paths(Fair(p))$ be such that

$$\pi \equiv <a_1, q_1>, <a_2, q_2>, \ldots$$
$$\equiv <a_1, p_1^{v_1}>, <a_2, p_2^{v_2}>, \ldots..$$

Suppose b_i is enabled infinitely often in π. We must show that b_i occurs infinitely often within π. From the construction of $Fair(p)$ it follows that in the sequence $(v_j)_j$ each index v_{j+1} is obtained from v_j by an application of case 1 or 2. Since case 1 can be applied only finitely many times in succession, it follows that case 2 must have been applied infinitely many times, each application increasing the length of the index. Therefore there is an $N \in \mathbb{N}$ such that for all $j > N$:

$$length(v_j) > i.$$

Now we are back in the old situation of the previous section! The proof can be completed as before, but for the new observation that with the increase of the length of an index, only priorities lower than the existing ones are introduced.

THEOREM 4.4
Any fair path in p is in the range of the mapping Φ.

PROOF
Given a fair path $\pi' \in Paths(p)$,

$$\pi' = <b_{i_1}, p_1>, <b_{i_2}, p_2>, \ldots,$$

we must construct a path $\pi \in Paths(Fair(p))$ such that

$$\Phi(\pi) = \pi'.$$

First, we partition $\mathbb{N}$ into two parts F and I, where F is the set of all i such that b_i is enabled finitely often (perhaps never) in π', and I is the set of all i such that b_i is enabled infinitely often in π'. Thus:

$$\mathbb{N} = I \cup F.$$

Note that (as in 3.4) for all $i \in F$ b_i occurs only finitely often in π' and for all $i \in I$ b_i occurs infinitely often in π', since π' is fair. Secondly, we introduce the following functions that will be very useful in our proof.

(a) For all $L \in \mathbb{N}$ we define a *(position)* function $Pos_L : B \to \mathbb{N}$ by

$$Pos_L(b_k) = \begin{cases} \text{smallest } L' \geqslant L \text{ such that} \\ \forall j \geqslant L' \, [b_k \notin p_j] & \text{if } k \in F \\ \text{smallest } L' \geqslant L \text{ such that} \\ \exists j \, [L < j < L' \wedge b_{i_j} = b_k] & \text{if } k \in I. \end{cases}$$

For $k \in F$ this function gives the smallest position greater than L after which b_k is never enabled again. For $k \in I$ the smallest position greater than L is chosen such that b_k has occurred (at least) once since L.

(b) For all $L, L' \in \mathbb{N}$, with $L \leqslant L'$, we define a *(number)* function $Num_{L,L'} : B \to \mathbb{N}$ by

$$Num_{L,L'}(b_k) = \begin{cases} 1 + \text{ (number of occurrences in } \pi' \\ \quad \text{of } b_k \text{ between } L \text{ and } L') & \text{if } k \in F \\ \text{(number of occurrences in } \pi' \\ \quad \text{of } b_k \text{ between } L \text{ and } L') & \text{if } k \in I. \end{cases}$$

(In this definition *between L and L'* means *including L* and *excluding L'*.)

We shall define, at each of an infinite sequence of stages k, an index ν_k and, corresponding to that index, the k-th part of the path π corresponding to π'. After we have constructed, at stage $k - 1$, the $(k - 1)$-th approximation of path π corresponding to the initial segment

$$<b_{i_1}, p_1>, \ldots, <b_{i_l}, p_l>$$

of path π', then at stage k we shall take into account the basic steps $b_{i_{l+1}}$ and all the b_j's we have encountered in the preceding stages. We shall make sure that the length of the index ν_k will be, as prescribed by definition 4.2, strictly bigger than the length of ν_{k-1}. Note that in the previous section, where our alphabet was finite, from the beginning we could focus on all b_j's at the same time.

Stage 1

For the definition of our first index ν_1 we focus on basic step b_{i_1}. We define

$$L_1 = Pos_1(b_{i_1}),$$

$$R_1 = \{i_1, \ldots, i_{L_1 - 1}\},$$

$$M_1 = \max R_1.$$

Our first index ν_1, with $\nu_1 = <n_1^{s_1}, \ldots, n_{M_1}^{s_{M_1}}>$, is defined so that

$$\forall 1 \leqslant j \leqslant M_1 \, [j \in R_1 \Rightarrow (n_j = Num_{1,L_1}(b_j) \wedge s_j = 1) \wedge$$

$$j \notin R_1 \Rightarrow (n_j = 0 \wedge s_j = \infty)].$$

The length of ν_1 is M_1, because according to the definition of indices no holes are allowed in ν_1, that is: every index is related to an initial part of the enumeration of our infinite alphabet $\{b_1, b_2, \ldots\}$. For those basic steps b_j that do not occur in the path π' before place L_1, default values $n_j = 0$ and $s_j = \infty$ are chosen in ν_1. (Here we use the fact that for newly introduced j's, n_j can get the value 0 once. See the corresponding remark in the explanation following definition 4.2.) With ν_1 we can construct the first part of π corresponding to the part of π' before L_1, starting with p^{r_1}, and repeatedly

applying case 1 for the appropriate b_i, thus decreasing the n_i's until (at step L_1) our index is such that for all $1 \leqslant i \leqslant M_1$:

$$(i \in F \cap R_1) \Rightarrow (n_i = 1 \wedge s_i = 1),$$

$$(i \in I \cap R_1 \vee i \notin R_1) \Rightarrow (n_i = 0 \wedge s_i = \infty).$$

Now case 2 must be applied, since no $j \in F \cap R_1$ is enabled at step L_1. This brings us to stage 2.

Stage 2
We define our next index ν_2, taking into account all steps encountered at stage 1, that is all b_i's with $1 \leqslant i \leqslant M_1$, and the next step in the path π', that is $b_{i_{L_1}}$. We define

$$L_2 = \max(\{Pos_{L_1}(b_{i_{L_1}})\} \cup \{Pos_{L_1}(b_k) \mid 1 \leqslant k \leqslant M_1\}),$$

$$R_2 = \{1, \ldots, M_1\} \cup \{i_{L_1}, \ldots, i_{L_2-1}\},$$

$$M_2 = 1 + \max R_2.$$

We define our second index ν_2, with $\nu_2 = <\tilde{n}_1^{\tilde{s}_1}, \ldots, \tilde{n}_{M_2}^{\tilde{s}_{M_2}}>$, such that

$$\forall 1 \leqslant j \leqslant M_2 \ [((1 \leqslant j \leqslant M_1 \wedge n_j = 0) \vee (j > M_1 \wedge j \in R_2) \Rightarrow$$

$$\tilde{n}_j = Num_{L_1, L_2}(b_j) \wedge \tilde{s}_j = 1 + \max\{s_k \mid 1 \leqslant k \leqslant M_1\}) \wedge$$

$$(1 \leqslant j \leqslant M_1 \wedge n_j = 1) \Rightarrow (\tilde{n}_j = n_j \wedge \tilde{s}_j = s_j) \wedge$$

$$(j \notin R_2) \Rightarrow (\tilde{n}_j = 0 \wedge \tilde{s}_j = \infty)].$$

Note that M_2, the length of ν_2, is strictly bigger than M_1, the length of ν_1. We proceed as before, constructing the part of π corresponding to the part of π' between L_1 and L_2. Continuing in this way, we construct a path π in *fair*(p) such that $\Phi(\pi) = \pi'$.

5. INFINITE ITERATION

Let P be the mathematical domain of section 3, that is, a complete metric space satisfying

$$P \cong \{p_0\} \cup \mathcal{P}_{cl}(B \times P)$$

where B is a finite alphabet

$$B = \{b_1, \ldots, b_m\}.$$

The operation of sequential composition on P is defined in

DEFINITION 5.1 (Sequential composition)
Let $\circ: P \times P \to P$ be given by

$$p \circ q = \begin{cases} q & \text{if } p = p_0 \\ \{<b, p' \circ q> \mid <b, p'> \in p\} & \text{if } p \neq p_0 \end{cases}$$

for all p and q in P.

REMARKS
(1) Because this definition is self-referential, it needs some justification. We observe that $\circ$ can be

defined as the unique fixed point of a contraction Φ of type $\Phi: (P \times P \to P) \to (P \times P \to P)$. (Cf. definition 3.4.)

(2) It is not very difficult to show that:

$$\forall p,q,q' \in P \; [p \neq p_0 \Rightarrow d_P(p \circ q, p \circ q') \leq \tfrac{1}{2} d_P(q,q')].$$

We shall use this property below.

In this section we want to study the operation of *infinite iteration* of a process $p \in P$. It is defined as follows:

DEFINITION 5.2 (Infinite iteration)
Let $(\cdots)^{\omega}: P \to P$ be given by

$$p^{\omega} = \lim_{n \to \infty} p^n$$

for $p \in P$, where $p^0 = p_0$ and $p^{n+1} = p^n \circ p$.
(This limit exists, as can be easily proved using the property of remark (2) above).

Let us now explain how fairness issues come into play by taking the infinite iteration of $p \in P$. Generally, taking the infinite iteration of a process $p \in P$ introduces new infinite paths in p^{ω} that were not yet present in p. When we take, for example, $p = \{<a,p_0>,<b,p_0>\}$, then p does not contain any infinite paths, whereas p^{ω}, which satisfies

$$p^{\omega} = \{<a,p^{\omega}>,<b,p^{\omega}>\},$$

contains many. Some of these are unfair, such as

$$\pi = <a,p^{\omega}>,<a,p^{\omega}>,<a,p^{\omega}>,\ldots,$$

which is unfair with respect to b_1. Such unfair paths π we call *globally* unfair. We do *not* call every unfair path in p^{ω} globally unfair, only those that are introduced, so to speak, by taking the infinite iteration of p. Another example may illustrate this point. (Formal definitions follow below.) Consider a process $p \in P$ satisfying

$$p = \{<a,p>,<b,p_0>\}.$$

Then p^{ω} will contain the unfair paths

$$<a,p>,<a,p>,\ldots,$$

$$<b,p>,<a,p>,<a,p>,\ldots,$$

$$<b,p>,<b,p>,<a,p>,<a,p>,\ldots, \text{ etc.}$$

The unfairness of these paths is, as it were, reducible to the unfairness of the path

$$<a,p>,<a,p>,\ldots,$$

which was already present in p. Therefore they will not be called globally unfair paths.

There is a second notion of unfairness, which plays a role here. It is called *node* (or *local*) unfairness. Again we explain it here by giving an example, the formal definition following below. Let $p \in P$ contain the node $p' = \{<a,p_1>,<b,p_2>\}$. Let $\pi \in \textbf{Paths}(p^{\omega})$ and suppose π passes through p' infinitely many times. If it is the case that in π the next step that is taken after passing through p' is always a, and never b, we call π *node* unfair (with respect to the node p'). The reason for this terminology is obvious: although b is infinitely often enabled in π *at node p'*, it is never chosen in π as the next step *after p'*.

The notions of global and node unfairness are in a sense independent. Let $p \in P$ be given by

$$p = \{<b,p'>\}, \quad \text{where}$$

$$p' = \{<a,p>,<b,p_0>\}.$$

Consider $\pi \in \mathbf{Paths}(p^\omega)$, given by

$$\pi = <b,p'>,<a,p>,<b,p'>,<a,p>,\ldots.$$

This path is not globally unfair, but is node unfair with respect to the node p'. Thus node unfairness does not imply global unfairness. The same holds in the opposite direction. Let $p \in P$ be defined by

$$p = \{<a^n,\{<a,p_0>,<b,p_0>\}> \mid n \in \mathbb{N}\} \cup \{a^\omega\},$$

using a^n and a^ω as shorthand with an obvious interpretation. (The fact that $a^\omega \in p$ is not important for the point we want to make with this example, but is implied by the (topological) closedness of p.) Now it is not difficult to find a path

$$\pi = <a,p_1>,<a,p_2>,<a,p_3>,\ldots$$

in $\mathbf{Paths}(p^\omega)$ (with $p_1,p_2,p_3,\ldots$ nodes of p) that is globally unfair (with respect to b), but fair with respect to every node of p, although it passes through p infinitely many times.

Let us now proceed with formally defining these notions of global and node unfairness. Actually, we shall define what we consider to be globally fair and node fair. For this we need the following notion.

DEFINITION 5.3 (Iteration paths)
Let $p \in P$, $\pi \in \mathbf{Paths}(p^\omega)$. We call π an *(infinite) iteration path*, whenever π is the concatenation of an infinite sequence of finite paths $\pi_1,\pi_2,\ldots \in \mathbf{Paths}(p)$:

$$\pi = \pi_1 \circ \pi_2 \circ \pi_3 \circ \ldots.$$

For a basic step b occurring in π_k we say that b occurs in the *k-th instantiation* of p.

REMARK
We have not defined the concatenation of finite paths. It is just what one would expect: if $\pi_1 = <a_1,p_1>,\ldots,<a_m,p_0>$, and $\pi_2 = <b_1,q_1>,\ldots,<b_m,p_0>$ are finite paths in $\mathbf{Paths}(p)$, then:

$$\pi_1 \circ \pi_2 = <a_1,p_1>,\ldots,<a_n,p>,<b_1,q_1>,\ldots,<b_m,p_0>.$$

(Note that finite paths always end in $<a,p_0>$, for some $a \in B$.)

DEFINITION 5.4 (Global fairness)
Let $p \in P$, $\pi \in \mathbf{Paths}(p^\omega)$. We call π *globally fair* whenever
(1) π is fair (in the sense of definition 2.3); or
(2) π is *not* an iteration path.
We call p^ω globally fair whenever all paths in p^ω are globally fair.

REMARK: It follows that a path in p^ω is globally unfair if and only if it is an iteration path and unfair.

DEFINITION 5.5 (Node fairness)
Let $p \in P$, $\pi \in \mathbf{Paths}(p^\omega)$. We call π *node fair with respect to p'*, for a subnode p' of p, whenever it is the case that: if π passes through p' infinitely often, then for all $b \in B$ that are enabled in p': b occurs infinitely often in π, *immediately after p'*. We call π node fair if it is node fair with respect to every subnode p' of p. Finally we call p^ω node fair if all paths in $\mathbf{Paths}(p^\omega)$ are node fair.

REMARK

In this definition the phrase "π passes through p' infinitely often" is not altogether clear: it may be the case that a subnode p' occurs in p on more than one place; p might even contain infinitely many instances of p'. Below we shall overcome this ambiguity by being more precise in identifying subnodes of p.

The aim of this section is to define two fair versions of the infinite iteration operator:

$$(\cdots)^{\omega_{fair}}: p \to p$$

such that the result $p^{\omega_{fair}}$ will be globally fair and node fair respectively. For this purpose we first give an alternative definition of infinite iteration, which will be used as a starting point for defining $(\cdots)^{\omega_{fair}}$.

PROPOSITION 5.6 (Alternative definition of infinite iteration)
Let $p \in P$. We define $App_p: P \to P$ by

$$App_p(p_0) = p \circ App_p(p)$$

$$App_p(q) = \{<a, App_p(q')> \mid <a, q'> \in q\}, \text{ if } q \neq p_0.$$

(Read "append" for App.) Then we have:

$$p^{\omega} = App_p(p)$$

REMARKS
(1) Formally, App_p can be defined as the unique fixed point of the function $\Phi_p: (P \to P) \to (P \to P)$, given by

$$\Phi_p(F)(p_0) = p \circ F(p),$$

$$\Phi_p(F)(q) = \{<a, F(q')> \mid <a, q'> \in q\}, \text{ if } q \neq p_0.$$

(It is straightforward to show that Φ_p is contracting.)
(2) The function App_p applied to an argument $q \in P$ replaces all occurrences of p_0 in q by p, in which, recursively, all occurrences of p_0 are again replaced by p.
(3) From proposition 5.6 it follows that $App_p(p_0) = App_p(p)$.

PROOF OF THE PROPOSITION
We define, for fixed $p \in P$, a function $\phi_p: P \to P$ by

$$\phi_p(q) = q \circ p^{\omega}.$$

We have

$$\phi_p(p_0) = p_0 \circ p^{\omega} = p^{\omega} = p \circ p^{\omega}$$

$$= p \circ (p \circ p^{\omega}) = p \circ \phi_p(p)$$

and, for $q \in P$, $q \neq p_0$:

$$\phi_p(q) = q \circ p^{\omega} \quad \text{(definition of } \circ \text{)}$$

$$= \{<a, q' \circ p^{\omega}> \mid <a, q'> \in q\}$$

$$= \{<a, \phi_p(q')> \mid <a, q'> \in q\}.$$

From this it follows that ϕ_p is also a fixed point of Φ_p. Because Φ_p is contracting, it has a unique fixed point, thus $\phi_p = App_p$. Thus

$$p^\omega = p \circ p^\omega = \phi_p(p) = App_p(p).$$

(1) Global fairness

In this subsection we set out to define a fair version

$$(\cdots)^{\omega_{fair}} : P \to P^{FInd}$$

of the operation of infinite iteration such, that for p in P the result $p^{\omega_{fair}}$ will be globally fair. The range P^{FInd} of this mapping $(\cdots)^{\omega_{fair}}$ is given by

$$P^{FInd} = \{p_0\} \cup \mathscr{P}_{cl}(A \times P^{FInd}),$$

with

$$A = B \cup FIndex,$$

where *FIndex* is a set of indices to be defined below. A naive first attempt would be to define

$$p^{\omega_{fair}} = Fair(p^\omega),$$

with the function *Fair* as in definition 3.4. This would be wrong, according to our definition of global fairness. The function *Fair* transforms its argument into a process, in which *all* unfair paths have disappeared. However, not every unfair path in p^o is globally unfair, only those that are iteration paths. Thus the function *Fair* removes too many paths from p^ω. (For an illustration see the informal explanation above.) Therefore we have to come up with another solution. We shall use the definition of p^ω as $App_p(p)$ as a starting point for the definition of $p^{\omega_{fair}}$, but changing it by again using indices (as we did in the definition of *Fair*) to label the nodes of p. After having defined $p^{\omega_{fair}}$, we shall clarify the relation between $p^{\omega_{fair}}$ and p^ω by defining a mapping

$$\Phi: Paths(p^{\omega_{fair}}) \to Paths(p^\omega).$$

Although the idea of defining $p^{\omega_{fair}}$ as $Fair(p^\omega)$ does not work (as was mentioned above), the definition of $(\cdots)^{\omega_{fair}}$ will be surprisingly similar to that of the function *Fair*. The reason is the following: in constructing $p^{\omega_{fair}}$ for a given $p \in P$, we do two things at the same time. On the one hand we construct (a special version of) the infinite iteration of p, and on the other hand we select certain paths, namely those that are globally fair. The first task is performed along the lines of the definition of App_p, the second task is realised following the definition of *Fair*. So in some sense the definition of $p^{\omega_{fair}}$ will be a combination of the definitions of App_p and *Fair* (see proposition 5.6 and definition 3.4).

DEFINITION 5.7 (Flag indices). The set of *flag indices*, with typical element μ, is defined by:

$$FIndex = \{ <<n_1, s_1, f_1>, \ldots, <n_m, s_m, f_m>> \mid n_i \geq 0,\ 0 \leq s_i \leq \infty,\ f_i \in \{U, D\} \}$$

where m is the number of basic steps in our finite alphabet B, and $\{U, D\}$ is the set of flags, containing two elements: U (for "up") and D (for "down").

The interpretation of n_i and s_i is as in definition 3.1 (see the informal explanation that follows there), but for the difference that only the *first* occurrence of b_i in each instantiation of p in $p^{\omega_{fair}}$ will cause n_i to be decreased by 1. Whether or not b_i has been chosen in a given instantiation of p, is indicated by the flag f_i. If it is up, b_i has not yet been chosen, and if it is down, b_i has been chosen at least once in the current instantiation of p.

We need the following operations on indices.

DEFINITION 5.8
Let $\mu \in FIndex$, with $\mu = <<n_1, s_1 f_1>, \ldots, <n_m, s_m, f_m>>$, and let $i \in \{1, \ldots, m\}$. We define

$$\mu^-[i] = \begin{cases} \mu & \text{if } f_i = D \\ <<n_1,s_1,f_1>, \ldots, <n_i-1,s_i,D>, \ldots, <n_m,s_m,f_m>> & \text{if } f_i = U \wedge n_i > 1 \\ <<n_1,s_1,f_1>, \ldots, <0,\infty,D>, \ldots, <n_m,s_m,f_m>> & \text{if } f_i = U \wedge n_i = 1 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

For $\mu \in \textbf{FIndex}$ with $f_i = U$ the interpretation of $\mu^-[i]$ is as in definition 3.2, with the difference that U is changed to D. This indicates that in the current instantiation of p the basic step b_i has been chosen (at least once). If $f_i = D$, then $\mu^-[i] = \mu$, as indicated above. This will be explained below, after the definition of $p^{\omega_{fair}}$.

DEFINITION 5.9

Let $\mu \in \textbf{FIndex}$, with $<<\mu = n_1,s_1,f_1>, \ldots, <n_m,s_m,f_m>>$. We define

$$N(\mu) = \{ <<\tilde{n}_1,\tilde{s}_1,\tilde{f}_1>, \ldots, <\tilde{n}_m,\tilde{s}_m,\tilde{f}_m>> \mid$$

$$\forall i \in \{1, \ldots, m\} \, [(n_i = 0 \wedge s_i = \infty \Rightarrow \tilde{n}_i > 0 \wedge \tilde{s}_i = 1 + \max\{s_j \mid 1 \leqslant j \leqslant m\})$$

$$\wedge (n_i > 0 \wedge s_i < \infty \Rightarrow \tilde{n}_i = n_i \wedge \tilde{s}_i = s_i)$$

$$\wedge \tilde{f}_i = f_i]\}.$$

The interpretation of $N(\mu)$ is as in definition 3.3, because the flags do not matter here.

DEFINITION 5.10

Let $\mu \in \textbf{FIndex}$ with, $\mu = <<n_1,s_1,f_1>, \ldots, <n_m,s_m,f_m>>$. Then

$$\mu^U = <<n_1,s_1,U>, \ldots, <n_m,s_m,U>>.$$

This operation sets all flags to "up" and is used upon entrance to a new instantiation of p. Now we are ready to define $(\cdots)^{\omega_{fair}}$.

DEFINITION 5.11 (Fair infinite iteration)

We define $(\cdots)^{\omega_{fair}} : P \rightarrow P^{FInd}$. Let $p \in P$. Then

$$p^{\omega_{fair}} = \sum_{\mu, \, I_0} App_p(p,\mu),$$

where

$$I_0 = \{ <<n_1,1,U>, \ldots, <n_m,1,U>> \mid n_i > 0\}$$

and for given $p \in P$

$$App_p : P \times \textbf{FIndex} \rightarrow P^{FInd}$$

is defined as follows. (We write q^μ for $App_p(q,\mu)$.) Let $\mu \in \textbf{FIndex}$. We define

$$App_p(p_0,\mu) = App_p(p,\mu^U).$$

For $q \in P$, $q \neq p_0$, we distinguish two cases.

Case 1:

If $\exists i \in \{1, \ldots, m\} \, [enabled(i) \wedge (f_i = D \vee (s_i < \infty \wedge n_i > 0))]$,

then $q^\mu = \{ <b_i, \bar{q}^{\mu^-[i]}> \mid <b_i, \bar{q}> \in q \wedge$

$$(f_i = D \vee (s_i < \infty \wedge n_i > 0 \wedge s_i = \min\{s_1, \ldots, s_m\}))\}.$$

Case 2:

If $\forall i \in \{1,...,m\}$ $[\text{enabled}(i) \Rightarrow (f_i = U \wedge s_i = \infty \wedge n_i = 0)]$

then $q^\mu = \sum_{\mu' \cdot N(\mu)} q^{\mu'}$.

REMARKS

(1) The remarks (1), (2), and (3) following definition 3.4 apply also to the above definition.

(2) We give some informal explanation of this definition by referring to remark (4) after definition 3.4 and making explicit what is different here. First, when we reach p_0 in the definition (3.4) of *fair*, we are done: $fair(p_0, \nu) = p_0$. Here we continue by appending p to p_0, together with the index μ changed into μ^U: $App_p(p_0, \mu) = App_p(p, \mu^U)$. The reason why we append p to p_0 is obvious: we are building the infinite iteration of p. (See proposition 5.6.) The index μ is changed to μ^U, that is all flags f_i of μ are set to U to indicate the entrance of a new instantiation of p. The second important difference between this definition and definition 3.4 is the role played by the flags. Let $q \in P$ with $<b_i, \overline{q}> \in q$ for some $\overline{q} \in P$, $b_i \in B$. If $f_i = D$ (down), then b_i has already been chosen (at least once) in the current instantiation of p. Therefore it may be chosen unrestrictedly, even infinitely many times, within *this instantiation* of p (no matter what the values of n_i and s_i are). In this case we have: $\mu^-[i] = \mu$, formally expressing that b_i may pass "for free" without changing the values of n_i and s_i. The reason for letting b_i pass for free is that it provides us with the presence within $p^{\omega_{iur}}$ of those infinite paths (possibly unfair) that are *not* iteration paths (and, hence, not globally unfair). If on the other hand $f_i = U$ and $n_i > 0$ and $s_i = \min\{s_1, \ldots, s_m\} < \infty$, then b_i may be chosen (as in case 2 of definition 3.4), but now μ is changed into $\mu^-[i]$ by changing the values of n_i and s_i (as in definition 3.4) and by changing the flag f_i to D.

Now for the rest of this subsection let $p \in P$ be fixed. We define a mapping

$$\Phi: Paths(p^{\omega_{iur}}) \to Paths(p^\omega),$$

relating to each iteration path in $p^{\omega_{iur}}$ a corresponding fair iteration path in p^ω. We start by re-stating lemma 3.5.

LEMMA 5.12

Let $\overline{p} \in P$, with $\overline{p} = p_0$, $\mu \in FIndex$, and $<a, q> \in App_p(\overline{p}, \mu)$ for $a \in B$ and $q \in P$. Then there exist $p' \in P$ and $\mu' \in FIndex$ such that

$$q = App_p(p', \mu') \wedge$$

$$a \in FIndex \Rightarrow p' = \overline{p} \wedge$$

$$a \in B \Rightarrow <a, p'> \in \overline{p}.$$

The proof is straightforward from the definition of $\overline{p}^\mu$ ($= App_p(\overline{p}, \mu)$).

DEFINITION 5.13 (The mapping Φ)

Let

$$\pi = <a_0, q_0>, <a_1, q_1>, \ldots$$

be a path in $p^{\omega_{iur}}$. We can rewrite it as:

$$\pi = <a_0, p^\mu>, <a_1, p_1^{\mu_1}>, <a_2, p_2^{\mu_2}>, \ldots$$

for certain $\mu, \mu_1, \mu_2, \ldots \in FIndex$ and $p_1, p_2, \ldots \in P$. If we omit in π all pairs $<a_i, p_i^{\mu_i}>$ with $a_i \in FIndex$, and further all superscripts μ_i, we get a sequence

$$\Phi(\pi) = <a_{i_1}, p_{i_1}>, <a_{i_2}, p_{i_2}>, \ldots$$

which is a path in p^ω. We call $\Phi(\pi)$ the path in p *corresponding to* the path π in $p^{\omega_{fair}}$. This defines a mapping

$$\Phi: Paths(p^{\omega_{fair}}) \to Paths(p^\omega).$$

THEOREM 5.14

$p^{\omega_{fair}}$ *is globally fair. That is, for all* $\pi \in Paths(p^{\omega_{fair}})$, *if* π *is an iteration path, then* π *is fair.*

PROOF

Let $\pi \in Paths(p^{\omega_{fair}})$ and suppose π is an iteration path. We reduce the proof of this theorem to that of theorem 3.7 by making the following observation. Since π is an infinite iteration paths it enters infinitely often into a new instantiation of p. Upon each entrance, all flags are raised (set to "up"). As was observed above, if $f_i = U$ (for $i \in \{1, \ldots, m\}$), then b_i is treated in case 1 of definition 5.11 above in exactly the same way as in case 1 of definition 3.4. Because this situation arises infinitely often, the argument given in the proof of theorem 3.7 also applies here. (Note that case 2 in both definitions 3.4 and 5.11 is the same.)

REMARK: Formally we have to extend definition 5.4 of global fairness to processes in P^{FInd}. This can be done straightforwardly.

THEOREM 5.15: *Any globally fair path in p^ω is in the range of the mapping Φ.*

PROOF

Let $\pi' \in Paths(p^\omega)$ such that π' is globally fair. We must construct a path $\pi \in Paths(p^{\omega_{fair}})$ such that

$$\Phi(\pi) = \pi'.$$

We distinguish between two cases: first, that π' is *not* an iteration path (and possibly unfair); second, that π' *is* an iteration path and fair.

(1) Suppose π' is not an iteration path. Without loss of generality we may assume that π' lies entirely within p (that is, the first instantiation of p in p^ω). We define a flag index μ by

$$\mu = <<1,1,U>, \ldots, <1,1,U>>$$

and take $<\mu,p^\mu>$ as the first element of the path π that we are constructing. Now we can continue the construction of π by repeatedly applying case 1 (of definition 5.11) for the appropriate b_i's (namely, those that occur in π'). Each time we encounter a b_i for the first time, the corresponding triple $<1,1,U>$ in the index is changed into $<0,\infty,D>$. >From this moment on b_i may be chosen unrestrictedly within this instantiation of p (in which the path π' lies), without changing the index. The path π thus constructed is an element of $Paths(p^{\omega_{fair}})$. Furthermore: $\Phi(\pi) = \pi'$. (Note that it is of no importance whether π' is fair or not.)

(2) Suppose π' is a fair infinite iteration path. As in the proof of theorem 5.14, we reduce this proof to that of the corresponding theorem in section 3 (theorem 3.8) by observing that the latter only needs a slight modification. When we count the number of times that a certain b_i occurs before a given step l_j in the path π', we have to count only the first occurrences of b_i in different instantiations of p. With this change in mind the proof of 3.8 can easily be transformed into a proof of this theorem.

(2) Node fairness

Let us now forget about global fairness and focus on the second notion: node fairness. We again set out to define a fair version

$$(\ \cdots \ )^{\omega_{fair}} \colon P \rightarrow P^{NInd}$$

of the operation of infinite iteration but now such, that for all $p \in P$ the result $p^{\omega_{fair}}$ will be *node* fair. The domain P^{NInd} is like P^{Ind} and P^{FInd}, but with

$$A \ = \ B \cup \textbf{\textit{NIndex}},$$

with **NIndex** a set of indices to be defined below.

In constructing this second version of infinite iteration we proceed globally as in the previous subsection, now using *node* indices in order to ensure the node fairness of $p^{\omega_{fair}}$, instead of flag indices, which were used above. We shall characterize (and even identify) a subnode of a given process $p \in P$ by the subpath in p that leads to it.

DEFINITION 5.16 (Nodes)
Let $p \in P$. We define the set of nodes of p by

$$\textbf{\textit{Nodes}}(p) \ = \ \{\pi \mid \exists \pi' \in \textbf{\textit{Paths}}(p) \ [\pi \text{ is a finite initial part of } \pi']\}.$$

For $\pi \in \textbf{\textit{Nodes}}(p)$, with $\pi = <a_1,p_1>, \ldots, <a_n,p_n>$, we define

$$\textbf{\textit{end}}(\pi) \ = \ p_n.$$

(When no confusion is possible we sometimes identify π and $\textbf{\textit{end}}(\pi)$.) We set $\textbf{\textit{end}}(\epsilon)=p$, where ϵ is the empty path.

The set of *node indices* for a given $p \in P$ is defined as follows. Each node index for p has two components: the first is a finite mapping, associating with each of a (finite) set of nodes of p an index as defined in 3.1; and the second is a node of p. Such a node index schedules the fairness of paths with respect to this second component. At each moment in the construction of $p^{0_{fair}}$, we consider only a finite number of nodes (the domain of the first component), namely those that we have encountered thus far.

DEFINITION 5.17 (Node indices)
Let $p \in P$. We define the set of node indices for p as follows:

$$\textbf{\textit{NIndex}}_p \ = \ (\textbf{\textit{Nodes}}(p) \rightarrow^{fin} \textbf{\textit{Index}}) \ \times \ \textbf{\textit{Nodes}}(p),$$

where $\rightarrow^{fin}$ denotes the set of partial functions on **Nodes**(p) with a finite domain, and **Index** is defined as in definition 3.1. A typical element of **NIndex** is denoted $\rho=(\rho_1,\rho_2)$. For $\rho_1 \in \textbf{\textit{Nodes}}(p) \rightarrow^{fin} \textbf{\textit{Index}}$ we use the variant notation for functions: for $\pi,\pi' \in \textbf{\textit{Nodes}}(p)$ and $\nu \in \textbf{\textit{Index}}$,

$$\rho_1\{\nu/\pi\}(\bar{\pi}) \ = \ \begin{cases} \nu & \text{if } \pi = \bar{\pi} \\ \rho_1(\bar{\pi}) & \text{if } \pi \neq \bar{\pi}. \end{cases}$$

(We shall use this notation whether or not $\pi \in \textbf{\textit{domain}}(\rho)$.)

We again need the operations $\nu^-[i]$ and $N(\nu)$ on indices $\nu \in \textbf{\textit{Index}}$ (see definitions 3.2 and 3.3). They are used in the following

DEFINITION 5.18 (Fair infinite iteration)
We define $(\ \cdots \ )^{\omega_{fair}} \colon P \rightarrow P^{NInd}$. Let $p \in P$. Then

$$p^{\omega_{fair}} = \sum_{\rho_1, \{\epsilon\}\to I_0} App_p(p,(\rho_1,\epsilon))$$

where ϵ is the empty subpath of p (identifying p as a subnode of itself),

$$I_0 = \{<n_1^1, \ldots, n_m^1> \mid n_i > 0\}$$

and for given $p \in P$

$$App_p: P \times NIndex \to P^{NInd}$$

is defined as follows. (We write q^ρ for $App_p(q,\rho)$.) Let $\rho \in NIndex$, $\rho = (\rho_1,\rho_2)$ and let $q \in P$. If $q \neq end(\rho_2)$, then $App_p(q,\rho)$ is undefined. Now suppose that $q = end(\rho_2)$. Then we define

$$App_p(p_0,\rho) = App_p(p_0,(\rho_1,\epsilon)).$$

For $q \neq p_0$ we distinguish two cases.

Case (a): $\rho_2 \in domain(\rho_1)$. Let $\rho_1(\rho_2) = v = <n_1^{s_1}, \ldots, n_m^{s_m}>$. Then

$(a1)$ If $\exists i \in \{1, \ldots, m\}$ [enabled$(i) \wedge n_i > 0 \wedge s_i < \infty$], then

$$q^\rho = \{<b_i,\bar{q}^{-(\rho_1\{v \ [i]/\rho_2\} , \rho_2 \circ <b_i,\bar{q}>)}> \mid <b_i,\bar{q}> \in q \wedge s_i = \min\{s_j \mid 1 \leqslant j \leqslant m\}\}.$$

$(a2)$ If $\forall i \in \{1, \ldots, m\}$ [enabled$(i) \Rightarrow n_i = 0 \wedge s_i = \infty$], then

$$q^\rho = \sum_{v', N(v)} q^{(\rho_1\{v'/\rho_2\},\rho_2)}.$$

Case (b): $\rho_2 \notin domain(\rho_1)$. Then

$$q^\rho = \sum_{v', I_0} q^{(\rho_1\{v'/\rho_2\},\rho_2)},$$

where I_0 is as above.

REMARKS

(1) The remarks (1), (2), and (3) following definition 3.4 apply also to the above definition.

(2) We have that $App_p(q,\rho)$ is undefined whenever $q \neq end(\rho_2)$. This implies (since $\rho_2 \in Nodes(p)$) that App_p is defined on nodes of p only, which seems quite natural.

(3) We give some informal explanation of the definition above. If we arrive at p_0, with index p, we continue with $App_p(p,\rho')$. Here $\rho' = (\rho_1,\epsilon)$, that is, the second component of ρ' now indicates that the node we are treating next is $(end(\epsilon)=) p$ itself. The interpretation of cases $(a1)$ and $(a2)$ above is entirely similar to that of the cases 1 and 2 in definition 3.4: if $\rho_2 \in domain(\rho_1)$, then $v = \rho_1(\rho_2)$ is treated exactly as before. A small difference is that, in $(a1)$, the second component ρ_2 is extended with $<b_i,\bar{q}>$ to denote that the next node of p that is treated is $\bar{q}$ $(= end(\rho_2 \circ <b_i,\bar{q}>))$. If $\rho_2 \notin domain(\rho_1)$, an extension of the domain of ρ_1 takes place. Here I_0 is the set of initial indices (as in definition 3.4).

Now for the rest of this subsection let $p \in P$ be fixed. As in definitions 3.6 and 5.13 we can define a mapping

$$\Phi: Paths(p^{\omega_{fair}}) \to Paths(p^\omega).$$

The following two theorems can be proved by easy generalizations of the corresponding proofs (3.7 and 3.8) in section 3.

THEOREM 5.19 $p^{\omega_{fair}}$ *is node fair.*

THEOREM 5.20 *Any node fair path in* p^ω *is in the range of the mapping* Φ.

Combining global and node fairness
We could now combine the two definitions (5.11 and 5.18) of fair infinite iteration and construct a function

$$(\ \cdots \ )^{\omega_{fair}} : P \to P$$

such that $p^{\omega_{fair}}$ would be both globally and node fair. We do not do this and confine ourselves to the observation that it would be a straightforward and dull exercise. Similarly for the generalization to an infinite alphabet.

6. REFERENCES

[ABKR]   P. AMERICA, J.W. DE BAKKER, J.N. KOK, J.J.M.M. RUTTEN, *A denotational semantics for a parallel object-oriented language,* report CS-R8626, Centre for Mathematics and Computer Science, Amsterdam, Netherlands, August 1986. (To appear in: Information and Computation.)

[AO]   K.R. APT, E.-R. OLDEROG, *Proof rules dealing with fairness,* in: Science of Computer Programming 3, 1983, pp. 65-100.

[AR]   P. AMERICA, J.J.M.M. RUTTEN, *Solving reflexive domain equations in a category of complete metric spaces,* Report CS-R8709, Centre for Mathematics and Computer Science, Amsterdam, Netherlands, February 1987. (To appear in: Proceedings of the Third Workshop on Mathematical Foundations of Programming Language Semantics, Springer-Verlag.)

[BK]   J.A. BERGSTRA, J.W. KLOP, *A convergence theorem in process algebra,* Report CS-R8733, Centre for Mathematics and Computer Science, Amsterdam, Netherlands, July 1987.

[BZ1]   J.W. DE BAKKER, J.I. ZUCKER, *Processes and the denotational semantics of concurrency,* Information and Control 54 (1982) 70-120.

[BZ2]   J.W. DE BAKKER, J.I. ZUCKER, *Processes and a fair semantics for the ADA rendez-vous,* in: Proceedings 10th ICALP (J. Diaz, ed.) LNCS 154, Springer-Verlag (1983) 52-66.

[BZ3]   J.W. DE BAKKER, J.I. ZUCKER, *Compactness in semantics for merge and fair merge,* in: Proceedings Workshop Logics of Programs, (E. Clarke & D. Kozen, eds.) Pittsburgh, LNCS 164 Springer-Verlag (1983) 18-33.

[D]   E.W. DIJKSTRA, *A discipline of programming,* Prentice-Hall, 1976.

[F]   N. FRANCEZ, *Fairness,* Springer-Verlag, 1986.

[LPS]   D. LEHMANN, A. PNUELI, J. STAVI, *Impartiality, justness and fairness: the ethics of concurrent termination,* in: Proc. 8th ICALP (Acre, Israel, July 1981), LNCS 115 (eds. O. Kariv, S. Even), Springer-Verlag, 1981.

[OA]   E.-R. OLDEROG, K.R. APT, *Transformations realizing fairness assumptions for parallel programs,* in: Proc. STACS 1984, LNCS 166, Springer-Verlag, 1984.

[P]   G.D. PLOTKIN, *A powerdomain for countable nondeterminism,* in: Automata, Languages and Programming, Proc. 9th ICALP, Aarhus, July 1982 (eds. M. Nielsen, E.M. Schmidt), LNCS 140, Springer-Verlag, pp. 418-428.