A METRIC CHARACTERIZATION OF FAIR COMPUTATIONS IN CCS *

Gerardo COSTA

Istituto di Matematica - Università di Genova

Via L.B. Alberti 4 - I 16132 Genova - Italy

ABSTRACT. We address the problem of characterizing fair (infinite) behaviours of concurrent systems as limits of finite approximations. The framework chosen is Milner's Calculus of Communicating Systems. The results can be summarized as follows. On the set FD of all finite derivations in the calculus we define three distances: da, dw, ds. Then the metric completion of (FD,da) yields the space of all derivations, while the completion of (FD,dw), resp. (FD,ds), yields the space of all finite derivations together with all - and only - the weakly, resp. strongly, fair computations (i.e. non-extendable derivations). The results concerning da and dw are a reformulation of previously known ones, while that concerning ds is - we believe - new.

INTRODUCTION

One of the most challanging problems in modelling fairness properties of concurrent systems is to reconcile infinite fair behaviours with our ideas and wishes about approximations and limits. We would like a framework in which we have some notions of finiteness, approximation and limit such that infinite behaviours are fully determined, via "countable limits", by the sets of their finite approximants. Now, fair infinite behaviours do not seem to fit into this pattern.

In [Pa1] Park provides a fixpoint relational semantics for data-flow networks with a fair merge operator. In doing so he has to consider limits (lub's and inf's) of transfinite chains. The denotational semantics for a while language with a fair parallel operator in [Pl] also appeals to transfinite chains. More recently, De Bakker and Zucker have proposed metric spaces of processes where they can interpret fair parallel operators using the standard notion of limit of (countable) sequences [BZ]. Here, however, the finiteness of the approximants is in question as fairness is achieved using the equivalent of random assignment. We see the same problem in the use of oracles in [Pa3].

Apparently, if there is a way out it is well concealed. A customary solution in such cases is to settle for less. Going in this direction, one realizes that the difficulties are not entirely due to the interaction of fairness on one side and our requirements, countable limits and finite approximations, on the other: our notion of behaviour plays its part. Some of the features that we see as most desirable become stumbling blocks. The most obvious one is abstraction: we are particularly keen to remove information which would allow us to capture fairness more easily. But there is another feature which, though desirable in general, becomes an impediment here: we want to consider global behaviours and handle them as a whole. For instance, in |Pa1| the behaviour of a network is its input-output relation; that of a program in |Pl| is the

* This research was carried out while the author was at the Dept. of Computer Science of the University of Edinburgh.

set of its (possible) results. It is such a global entity that one tries to obtain through the limiting process.

In a recent paper, Degano and Montanari consider a notion of behaviour closely connected to the operational notion of step-wise evolution of a system |DM|. Then they can characterize some liveness properties, including fairness, of the individual entities which constitute the global behaviour of systems in terms of convergence properties in appropriate metric spaces. We omit the details of their approach as here we proceed along the same lines, though in a slightly different setting: that provided by Milner's CCS. Indeed, the starting point for the present work has been to find an analogue of the results in |DM| which concern fairness within the framework proposed in |CS2|.

We consider the kind of concurrent systems that are represented by (closed) expressions in CCS without value-passing. These are systems in which components evolve asynchronously and can interact by exchanging signals on which they synchronize. We shall call them processes in the sequel. We take as behaviour of a process the set of its computations, i.e. non-extendable derivation sequences as defined by the rules of the calculus. As the bare derivation sequences (without mention of how each step is inferred) do not contain enough information, we add labels to the calculus. They allow us to see if a subprocess remains unduly idle in the course of a computation. The setting is rich enough to allow weak and strong fairness |Pa2,Pl| to be distinguished. On the set of all derivations we define three distances: da, dw,ds. Then, infinite computations - taken individually - are fully determined as limits w.r.t. da of (countable) sequences of finite derivations (which play the role of finite approximants). The same result is obtained for infinite computations which are weakly, resp. strongly, fair using dw, resp. ds.

The first two results are not new and are presented for completeness sake and as a stepping stone towards the third. The one concerning infinite computations, without regard to their fairness, is a straightforward adaptation of well known results (see e. g. |AN|). The result concerning weak fairness is a reformulation of one in |DM|. It is the result about strong fairness which is new, or, at least, an improvement on previous ones. Indeed, the characterization of strong fairness given by Degano and Montanari is weaker than ours.

It should be clear that we do not regard our results as a real breakthrough towards a solution of the initial problem. Forfeiting both abstraction and "globality" in the notion of behaviour is, perhaps, too much. On the other hand, the present work adds to our confidence in the framework for handling fairness developed jointly with Colin Stirling |CS1,CS2|. The approach in those papers is operational; the problem addressed is that of giving a finite set of rules for generating fair computations in CCS, without resorting to random assignment. In |CS2| two calculi are presented, one for weak and the other for strong fairness. A starting point is the notion of "fairness at i" (see Sect. 2) which yields, at least in the weak case, a more local characterization of fairness. Here, this same notion provides the basis for defining the distances dw and ds mentioned above.

In Section 1 we give a concise account of Milner's CCS and of the labelled calculus we use. In Section 2 we formalize the notion of fairness in the context of CCS and give a more local characterization of it. Most of the material in this two sections is borrow ed from |CS2|. Section 3 provides the basic tools used in defining our distances. The metric characterization of infinite (fair) computations is in Section 4. The final section is devoted to concluding remarks and the Appendix to an example. Proofs are omitted; they appear in the extended version of this paper |C|.

We assume the reader familiar with the elementary notions of the theory of metric spaces.

1. C C S

We give here a concise account of CCS without value-passing (hence communication is reduced to pure synchronization) and renaming. These features have been omitted for simplicity sake; they can be accomodated within the framework we develop here. For a detailed account of CCS see |M1|. Then we introduce the labelled CCS we shall actually use. Finally, we define the notions of derivation and computation. The example at the end of the paper provides an illustration of the labelled calculus and, by suppressing labels, of standard CCS.

Let AA be a set of atomic actions and $\overline{AA}$ be a set of co-actions disjoint from AA and in bijection with it. The bijection is $\bar{\ }$: $\bar{a}$ stands for the co-action of a and $\bar{\bar{a}} = a$. The calculus allows for synchronization of co-actions; this is represented by τ, a silent or internal action not in $AA \cup \overline{AA}$. Let $Act = AA \cup \overline{AA}$ and Move = $Act \cup \{\tau\}$ and a,b,c,... range over Act, m over Move and X over a suitable set of variables. The syntax of CCS expressions is:

E ::= X | NIL | mE | E+E | E|E | fixX.E | E\a .

NIL is the process which does nothing; + represents nondeterministic choice; | concurrency; fix recursion; \a restriction (prevention) of a and $\bar{a}$ actions. We assume that in fixX.E X is guarded in E: every free occurrence of X in E is within a subexpression of the form mG. The rules of the calculus are (where E–m→G means E becomes G by performing the move m and E[G/X] denotes substitution of G for X in E):

$$mE\text{–}m\!\to E \qquad \frac{E\text{–}m\!\to E'}{E+G\text{–}m\!\to E'} \qquad \frac{G\text{–}m\!\to G'}{E+G\text{–}m\!\to G'} \qquad \frac{E[\text{fix}X.E/X]\text{–}m\!\to E'}{\text{fix}X.E\text{–}m\!\to E'}$$

$$\frac{E\text{–}m\!\to E'}{E|G\text{–}m\!\to E'|G} \qquad \frac{G\text{–}m\!\to G'}{E|G\text{–}m\!\to E|G'} \qquad \frac{E\text{–}a\!\to E' \quad G\text{–}\bar{a}\!\to G'}{E|G\text{–}\tau\!\to E'|G'} \qquad \frac{E\text{–}m\!\to E' \quad m \notin \{a,\bar{a}\}}{E\backslash a\text{–}m\!\to E'\backslash a}$$

Three points are worth mentioning. Sometimes the + rules do not allow choice: (aE+bG)\b can only become E by performing 'a'. The rules for | do not compel synchronization: aE|$\bar{a}$G can perform a or $\bar{a}$ as well as τ – on the other hand, (aE|$\bar{a}$G)\a can only perform τ because of the restriction. Finally, the number of concurrent subprocesses may increase as moves are performed; for instance, if E = fixX.aX|bX, then E–a→E|bE.

In discussing fairness we shall need to know "who is doing what". One way of achieving this is to introduce labels into the calculus. The full definition of the labelled calculus appears in |CS2| (and a similar calculus is in |CS1|). The precise details are needed only in the proofs of the results in the following section (for them see |CS2|). We think it suffices here to point out the essential features of this calculus.

Labels are strings in $\{1,2\}^*$, with typical elements u,v,w,... and ε denoting the null string. They are assigned systematically following the structure of expressions. Due to recursion the labelling is in part dynamic: the rule for fix generates new labels. The syntax of expressions is unchanged. However, variables, NIL, symbols in Move and the operators +, |, \a, fixX are labelled. Each label occurs at most once in an expression; we call this property unicity of labels. An example is $(a_{11}NIL_{111} +_1 b_{12}NIL_{121}) |_\varepsilon (fixX)_2.a_{21}X_{211}$. In $(fixX)_u.E$ we assume that $(fixX)_u$ binds any (free) labelled X within E. Similarly, in $E(\backslash a)_u$ all labelled a and $\bar{a}$ are restricted. Moreover, a_u and $\bar{a}_v$ are assumed to be complementary, irrespective of u and v. The rules of the calculus are just the ones in standard CCS where we allow expressions to carry labels; moves remain unlabelled. Examples are:

$$m_u E \xrightarrow{m} E \ ; \qquad \frac{E \xrightarrow{m} E'}{E|_u G \xrightarrow{m} E'|_u G} \qquad \text{and} \qquad \frac{E \xrightarrow{m} E' \quad m \notin \{a,\bar{a}\}}{E(\backslash a)_u \xrightarrow{m} E'(\backslash a)_u} \ .$$

The only real change is in the rule for fix: standard substitution, [-/-] , is replaced by a substitution operation which also changes the labels in the substituted expression by prefixing them with the label(s) of the variable occurrence(s) it replaces. The net result is that, for instance, under the labelling fixX.aX is "equivalent" to the infinite expression $a_{u_1}a_{u_2}\ldots a_{u_i}\ldots$ where each u_i occurs only once.

If $E \xrightarrow{m} G$, then the labels in G are determined by those in E and unicity of labels is preserved. If a label occurs both in E and G it will be attached to the same symbol (and this will indicate that the move has not affected this symbol) or to a variable, say X, in E and to a fixX in G. Once a label is lost via a move, then it is never regained. Finally, if r(-) corresponds to the operation of removing labels: if $E \xrightarrow{m} G$, then $r(E) \xrightarrow{m} r(G)$ in standard CCS; if $E' \xrightarrow{m} G'$ in standard CCS, then for any E s.t. r(E)=E' there exists G s.t. r(G)=G' and $E \xrightarrow{m} G$.

From now on we assume that we are working within the labelled calculus. However, whenever possible, we leave labels implicit.

A derivation is any finite or infinite sequence of the form $E_0 \xrightarrow{m_1} E_1 \xrightarrow{m_2} E_2 \ldots$ A computation is just a non-extendable derivation: if it is finite its last term is unable to make a move; on the other hand any infinite derivation is a computation. It is useful to have a more precise definition and an alternative representation for derivations. If we know that the initial term is E_0, then the sequence above can be represented as a sequence of pairs: $(m_1,E_1)(m_2,E_2)\ldots$ This can be formalized as follows, where

[1 k] denotes the set $\{i,\ 1\leq i\leq k\}$, $k\geq 0$, and N^+ denotes the set of positive natural numbers. A finite derivation of lenght k from E_0, $k\geq 0$, is a sequence $Y=(m_1,E_1)\dots(m_k,E_k)$ s.t. (*) below holds, where $Dom(Y)=[1\ k]$. An infinite derivation from E_0 is an infinite sequence $Y=(m_1,E_1)\dots(m_k,E_k)\dots$ s.t. (*) below holds, where $Dom(Y)=N^+$.

(*) $\forall\ i \in Dom(Y)\ .\ E_{i-1}\ -m_i\rightarrow\ E_i$.

The following notation will also be convenient: $Dom0(Y) = Dom(Y) \cup \{0\}$.

In the sequel, we shall use both representations for derivations, according to which is more appropriate.

2. FAIRNESS IN CCS

Fairness imposes the constraint that each concurrent subprocess always eventually proceeds, unless it is deadlocked or has terminated. In CCS, however, subcomponents may not be able to proceed autonomously, hence weak and strong fairness are distinguishable [Pa2,Pl]. The weak fairness constraint states that if a subcomponent can almost always proceed then eventually it must do so, while the strong fairness contraint states that if a subcomponent can proceed infinitely often then it must proceed infinitely often. Consider $Y = E\text{-}a\rightarrow G\text{-}c\rightarrow E\text{-}a\rightarrow G\text{-}c\rightarrow \dots E\text{-}a\rightarrow G\text{-}c\rightarrow E \dots$, where $E=(F|\bar{b}NIL)\backslash b$, $F=$ fixX. a(cX+bNIL) and $G=((cF+bNIL)|\bar{b}NIL)\backslash b$. The subcomponent $\bar{b}NIL$ is blocked in E (as \b prevents it from moving autonomously and it cannot synchronize with F, whose only action is a) while it can synchronize with bNIL in G. Hence Y is not strongly fair: $\bar{b}NIL$ can move infinitely often and never does. On the other hand, all subcomponents which are almost always able to move do so; hence Y is weakly fair.

We stress the fact that we regard fairness as an issue concerning concurrent subcomponents only. For instance, if H = fixX. aX+bX, then the sequence $H\text{-}a\rightarrow H\text{-}a\rightarrow H\text{-}a\rightarrow \dots$ is fair. (Indeed, at each step in the above sequence there is a choice between a and b; if a is chosen - performed - then b is discarded. At each step ther is a "new" choice, with "new" a and b, independent from the previous - and future - ones.) So, from now on subprocess, subcomponent, component will mean concurrent subprocess, subcomponent, component.

Now we formalize the two fairness constraints and then give an alternative, more local, characterization for them. The reader can find more details in |CS2|.

We need to define the notion of (top-level) live subprocess of a process E: a (concurrent) subcomponent that can contribute to the performance of a move of E.

First, we let P(E) be the set of (top-level) subprocesses of E irrespective of liveness. This set is defined inductively, letting labels represent processes.

$P(X) = \emptyset$; $P(NIL_u) = P(m_u E) = \{u\}$; $P(fixX.E) = P(E\backslash a) = P(E)$;

$P(E|G) = P(E) \cup P(G)$; $P(E +_u G) =$ if $P(E) \cup P(G) = \emptyset$ then $\emptyset$ else $\{u\}$.

Note that the notion of subcomponent is dynamic: if $E = a_u a_v NIL|b_w NIL$ and E performs 'a' then the resulting subcomponents do not include u. A simpler, static, notion of subprocess is inadequate here because the number of subcomponents may grow under derivation.

We now define Act(E), the set of unlabelled actions in E (excluding τ) which can happen autonomously:

$Act(X) = Act(NIL) = Act(\tau E) = \emptyset$; $Act(aE) = \{a\}$; $Act(fixX.E) = Act(E)$;

$Act(E\backslash a) = Act(E) - \{a,\bar{a}\}$; $Act(E+G) = Act(E|G) = Act(E) \cup Act(G)$.

A simple consequence of this definition is: $a \in Act(E)$ iff $\exists G . E \text{-}a\text{->} G$.

Next, we define LP(E,A) to stand for the set of live subprocesses of E when the environment prevents the actions in A, a subset of Act. Clearly, LP(E,A) is a subset of P(E). If $\overline{Act(E)}$ denotes $\{\bar{a} : a \in Act(E)\}$; then:

$LP(X,A) = LP(NIL,A) = \emptyset$; $LP(fixX.E\ , A) = LP(E,A)$;

$LP(m_u E,A) =$ if $m \notin A$ then $\{u\}$ else $\emptyset$;

$LP(E +_u G\ , A) =$ if $LP(E,A) \cup LP(G,A) = \emptyset$ then $\emptyset$ else $\{u\}$;

$LP(E\backslash a\ , A) = LP(E\ , A \cup \{a,\bar{a}\})$;

$LP(E|G\ , A) = LP(E\ , A - \overline{Act(G)}) \cup LP(G\ , A - \overline{Act(E)})$.

The set of live subprocesses in E is defined as $LP(E,\emptyset)$ which we abbreviate to LP(E).

Looking at our running example (in Appendix) one sees that the labels in $P(E_i)$ and $LP(E_i)$ are the expected ones: the following lemma provides a formal justification for the definitions just given. (Recall that if a subcomponent contributes to a move, then afterwards it no longer exists.)

Lemma 2.1

1. if $u \in P(E) - LP(E)$ and $E\text{-}m\text{->}G$ then $u \in P(G)$;
2. if $u \in LP(E)$ then $\exists G, m .\ E\text{-}m\text{->}G$ and $u \notin P(G)$;
3. if $E\text{-}m\text{->}G$ then $\exists u \in LP(E) .\ u \notin P(G)$.

The set P(E) has a "persistency" property - not shared with LP(E) - which will be very useful in dealing with strong fairness.

Fact 2.2 If $E_0\text{-}m_1\text{->}E_1\text{-}m_2\text{->}\dots\text{-}m_k\text{->}E_k$ and $u \in P(E_0) \cap P(E_k)$ then $u \in P(E_i)$, $1 \leq i < k$.

Given these definitions, we can now define admissibility under the two fairness constraints. (Intuitively, E_0 below should be closed - and the reader is free to assume it is so - formally we do not need this condition.)

Definition 2.3. Let $Y = E_0 -m_1\rightarrow E_1 -m_2\rightarrow \ldots$ be a computation.

1. Y is weakly fair iff $\forall u . \forall i\ \exists n \geq i\ .\ u \notin LP(E_n)$;
2. Y is strongly fair iff $\forall u\ \exists i\ \forall n \geq i\ .\ u \notin LP(E_n)$.

Quantification over i and n is implicitly restricted to DomO(Y).

A computation is weakly fair iff no subcomponent becomes live and then remains live throughout. It is strongly fair iff no subcomponent is live infinitely often. Notice that a component cannot be live infinitely often and proceed infinitely often because of the labelling: as soon as a component contributes to a move then it "disappears". Clearly, if a computation is strongly fair it is also weakly fair; also, any finite computation is strongly fair (its final term will have no live subcomponents).

Our next step is to try and express fairness as a local property and not just as a property of complete computations. This attempt proves successful for weak fairness, through the notion of "w-fairness at i". Not surprisingly, it fails for strong fairness: if Y is "s-fair at i" this can only be known, in general, by inspecting the complete tail of Y after i. Let $Y = E_0 -m_1\rightarrow E_1 -m_2\rightarrow \ldots$ be a derivation (not necessarily a computation).

Definition 2.4

1. Y is w-fair at i iff $\exists k \geq i\ .\ \bigcap \{LP(E_j),\ i \leq j \leq k\} = \emptyset$;
2. Y is s-fair at i iff $\exists k \geq i\ \forall n \geq k\ .\ P(E_i) \cap LP(E_n) = \emptyset$.

Quantification over k and n is implicitly restricted to DomO(Y). Any k satisfying the r.h.s. in 1 (resp. 2) will be called a w-witness (resp. an s-witness) for i.

The following fact is straigthforward when x=w. In the case x=s, it relies on fact 2.2; it would be false if $LP(E_i)$ replaced $P(E_i)$ in Def. 2.4.2.

Fact 2.5 Let $x \in \{w,s\}$. If Y is x-fair at i, then Y is x-fair at j for all j<i, any x-witness for i being also an x-witness for j.

Borrowing the terminology from [M2] we can say that any u in $P(E_i)$ represents an expectation of E_i. This expectation is immediate when $u \in LP(E_i)$ - i.e. there is a move from E_i which involves u, fulfilling the expectation, see Lemma 2.1.2 - otherwise it is quiescent. Then, using Fact 2.5, we can read the definition of w-fairness at i as follows. Upon reaching E_k we know that each of the immediate expectations of E_j, $0 \leq j \leq i$, has disappeared completely, or it has become quiescent at least in one of the terms $E_{j+1}, \ldots, E_k$. A similar explanation - but taking into account the differences already pointed out - can be given of s-fairness at i. We thus have the intuitive justification for the next result.

Theorem 2.6 If Y is a computation then it is weakly (resp. strongly) fair iff Y is w-fair (resp. s-fair) at i for all i in DomO(Y).

3. THE BASIC TOOLS

Here we introduce some notation and lw, ls, LS: the basic tools we use in defining our distances for weak and strong fairness. We have found it convenient to concentrate on a fixed initial term. So we assume E_0 to be fixed throughout this and the next section. Later (at the end of Sect. 4), we shall outline how to remove this restriction.

In what follows, derivation, computation,... abreviate derivation from E_0, computation from E_0,... ; similarly, FD, D^ω, D,... below abbreviate $FD(E_0)$, $D^\omega(E_0)$, $D(E_0)$,...

Then: FD is the set of finite derivations; D^ω is the set of infinite derivations; $D^{\omega w}$ is the set of infinite weakly fair derivations; $D = D^\omega \cup FD$; $D^w = FD \cup D^{\omega w}$. $D^{\omega s}$ and D^s are defined in a similar way.

If Y is in D and i in Dom(Y), then: Y(i) is the i-th element of Y - i.e. (m_i, E_i) - and Y[i] is the initial segment of Y of lenght i - i.e. Y(1)...Y(i). By convention, $Y[0] = \varepsilon$ i.e. the null string.

Let $Y = E_0 - m_1 \rightarrow E_1 - m_2 \rightarrow \ldots - m_n \rightarrow E_n \ldots$ be in D.

Definition 3.1. For k in DomO(Y), $lw(Y,k) = \text{lub}\ \{i : 0 \leq i \leq k \text{ and } LP(E_i) \cap \ldots \cap LP(E_k) = \emptyset\}$.

The intuitive idea behind this definition is to take the one for w-fairness at i and read it backwards. The lub is taken in N -the set of natural numbers. We use lub instead of max just to handle the empty set: lub $\emptyset = 0$. For an example see Appendix. From this definition we obtain immediately fact 3.2. Lemma 3.3 provides yet another characterization of infinite weakly fair computations. Its proof is straightforward given theor. 2.6 and fact 3.2.1.

Fact 3.2. 1. $lw(Y,n) \geq i$, for some $n \geq i$, iff Y is w-fair at i;

2. if $m \leq n$ then $lw(Y,m) \leq lw(Y,n)$.

Lemma 3.3. If Y is in D^ω, then Y is weakly fair iff $\lim \{lw(Y,n), n \geq 0\} = +\infty$.

We now define an appropriate correlate of s-fairness at i. Clearly, the difference between s- and w-fairness at i will come forward. A definition which mimics that of lw will only provide us with a convenient notation: ls. The true correlate of s-fairness at i is LS, whose definition involves looking ahead in the derivation.

Definition 3.4. If k is in DomO(Y), then:

$ls(Y,k) = \text{lub}\ \{i : 0 \leq i \leq k \text{ and } P(E_i) \cap LP(E_k) = \emptyset\}$;

$LS(Y,k) = \text{lub}\ \{i : 0 \leq i \leq k \text{ and } \forall\ n \geq k, n \text{ in DomO}(Y)\ .\ P(E_i) \cap LP(E_n) = \emptyset\}$.

Clearly: $0 \leq LS(Y,k) \leq ls(Y,k) \leq lw(Y,k) \leq k$.

We now state the analogue of fact 3.2 for LS; its proof is immediate. Notice that neither part 1 nor part 2 hold of ls. This can be seen using the example given in the

Appendix (for part 1, consider the derivation Y[6] : ls(Y[6] , 5)=4 but Y[6] is not s-fair at 4). The link between ls and LS is established by lemma 3.6.

Fact 3.5. 1. $LS(Y,n) \geq i$, for some $n \geq i$, iff Y is s-fair at i;

2. if $m \leq n$ then $LS(Y,m) \leq LS(Y,n)$.

Lemma 3.6. $LS(Y,k) = \min \{ ls(Y,n) , n \geq k$ and n is in DomO(Y) $\}$.

As expected, LS provides an alternative characterization of strong fairness. But this is also true of ls: taking the limit we overcome the "weaknesses" of ls. Once more the proof is straightforward.

Lemma 3.7. If Y is in D^{ω}, then Y is strongly fair iff

$\lim \{ ls(Y,n) , n \geq 0 \} = +\infty$ iff $\lim \{ LS(Y,n) , n \geq 0 \} = +\infty$.

4. A METRIC CHARACTERIZATION OF D , D^{w} AND D^{s}

Here we introduce our distances: da,dw,ds. The first allows us to characterize all infinite computations as limits of sequences of finite derivations. The analogue for infinite weakly (resp. strongly) fair computations is achieved using dw (resp. ds). These characterization results are stated in theorem 4.6.

The distance da is essentially a well known distance on strings; also well known is the relative result (see e. g. [AN]). The other two distances are derived from lw and LS, hence from the notions of w- and s-fairness at i. However, an analogue of the result concerning dw can be found in [DM]: the characterization of "globally fair" computations as limits of "histories" w.r.t. d_2. What is new is the result concerning strong fairness; indeed the one by Degano and Montanari is ... weaker. (We shall come back to this point in the concluding section.)

Recall that here we assume that derivations have a fixed initial term, E_0. At the end of this section we outline how to remove this restriction.

Let Y and Z be two (finite or infinite) derivations s.t. $Y \neq Z$ and let P be their longest common prefix - as sequences - which is finite even if Y and Z are not. The lenght of P provides a natural basis for a distance between Y and Z without regard to fairness; hence the definition of δa and da below. The intuition behind lw and lemma 3.3 suggest that lw(P,lenght(P)) can play the same role w.r.t. a distance between Y and Z which takes weak fairness into account. This motivates the definition of δw - notice that if $k \leq$ lenght(P), then $lw(P,k) = lw(Y,k) = lw(Z,k)$. Then dw is obtained from δw in the same way as da is obtained from δa. Lemma 4.4 states that da and dw are indeed distances (actually, ultra-distances) and that dw is more "stringent" than da.

Definition 4.1. If Y and Z are in D and $Y \neq Z$ then:

$$\delta a(Y,Z) = \max \{i : Y[i] = Z[i]\} ; \quad \delta w(Y,Z) = lw(Y, \delta a(Y,Z)) .$$

Definition 4.2. If Y and Z are in D, $Y \neq Z$ and x=a,w, then:

$$dx(Y,Y) = 0 ; \quad dx(Y,Z) = [\delta x(Y,Z) + 1]^{-1}.$$

Our distance for strong fairness is defined directly from LS (but, probably, by refining the basic idea one could define some δs, hence derive ds in the same way as dw is). A more fundamental difference with dw is however the one inherited from the difference between LS and lw (and ls). While lw(Y,n) depends entirely upon the initial segment of Y, LS(Y,n) depends also on its tail. So if $\delta a(Y,Z) = n$ the distance dw uses lw to explore the common prefix of Y and Z while ds needs to know both LS(Y,n) and LS(Z,n). For examples concerning ds see Appendix.

Definition 4.3. If Y and Z are in D and $Y \neq Z$ and $n = \delta a(Y,Z)$ then:

$$ds(Y,Y) = 0 ; \quad ds(Y,Z) = [LS(Y,n) + 1]^{-1} + [LS(Z,n) + 1]^{-1}.$$

Lemma 4.4. da, dw and ds are distances on D. Moreover, for any Y and Z:

$$da(Y,Z) \leq dw(Y,Z) \leq \tfrac{1}{2} ds(Y,Z) .$$

The only non-trivial part in the proof of this lemma is to show that the triangular inequality holds for dw and ds.

We can now state the promised results. Theorem 4.5 says that the infinite computations are fully determined by the finite derivations and the distance da. Any computation in D can be obtained as limit of a sequence of finite derivations; conversely, any sequence of derivations which is a Cauchy sequence w.r.t. da has a limit in D. The analog is true of infinite weakly/strongly fair computations and dw/ds. In other words, when completing FD w.r.t. dw/ds, we add all and only the infinite computations which are weakly/strongly fair. Notice that if Y is infinite, then, rather trivially, Y is the limit w.r.t. da of $\{Y[n], n \geq 0\}$ - and of infinitely many other sequences. The "value" of dw/ds is in the following fact:

$\{Y[n], n \geq 0\}$ converges w.r.t. dw/ds <u>if and only if</u> Y is weakly/strongly fair (and then the limit is Y).

Notation. If D' is a subset of D and d is a distance on D, the restriction of d to D' (which is a distance on D') will also be denoted by d. If x = a,w,s, $\lim_x$ denotes a limit w.r.t. dx; limits are taken in D.

Theorem 4.5. The spaces (D,da), (D^w,dw), (D^s,ds) are (isomorphic to) the metric completion of (FD,da), (FD,dw), (FD,ds), respectively.

The proof of the result concerning da is well known - and, anyway, straightforward. The proof of the other two results relies on the previous one and on lemmas 4.6 and 4.7 below; given these it is straightforward. Part 1 of lemma 4.6 is immediate from lemma 4.4, while lemma 4.7 essentially follows from lemmas 3.3 and 3.7. The crucial proof is that of part 2 in lemma 4.6; not surprisingly, the case x=s is harder than the other.

Lemma 4.6. If x=w,s and S is a Cauchy sequence in (D,dx), then:

1. S is also a Cauchy sequence in (D,da);
2. $\lim_x S$ exists and coincides with $\lim_a S$.

Lemma 4.7. If x=w,s and S is a sequence in FD and $\lim_x S = Y$, then Y is in D^x. (The non-trivial case is when Y is infinite; then it is fair.)

Let us now outline how to extend the results above to all derivations. Recall that FD, D, ... have been used so far as explicit abbreviations for $FD(E_0)$, $D(E_0)$, ... Implicitly, also dx has been an abbreviation for $dx(E_0)$, x=a,w,s. Now we redefine FD, D,... and dx. If $\oplus$ denotes disjoint union and EXP denotes the set of (labelled) CCS expressions, then we let:

$FD = \oplus \{ FD(E) : E \text{ in } EXP \}$; $D = \oplus \{D(E) : E \text{ in } EXP\}$; and so on.

Therefore, an element of D can be represented as a pair: <E,Y> , with Y in D(E). Now, on D we can define, for x=a,w,s :

$dx(<E,Y> , <E,Z>) = dx(E)(<Y,Z>)$; $dx(<E,Y> , <G,Z>) = 2$, when $E \neq G$.

It is immediate to check that dx is a distance on D (notice that the diameter of D(E) w.r.t. dx(E) is at most 1). Moreover, $\{ <E_n,Y_n> , n \geq 0 \}$ is a Cauchy sequence in (D,dx) iff $\exists k \geq 0 \ \exists E$ s.t. $\forall n \geq k$. $E_n = E$ and $\{ <E,Y_n> , n \geq k \}$ is a Cauchy sequence in (D(E),dx(E)).

It is clear then that theorem 4.5 and the auxiliary lemmas hold also when FD , D, ..., da, dw and ds are those just defined.

CONCLUDING REMARKS

The final "bringing everything together" should not mislead the reader. What we have shown here is that fair computations, taken individually, can be characterized as limits of finite derivations. We do not know whether our setting allows a similar characterization ot the fairness of global behaviours keeping approximants finite.

The other question is whether we really need all the information we carry around: labels, intermediate steps in derivations (the E_i's) and so on. In |DM|, Degano and

Montanari characterize weakly fair computations as limits of finite "histories", which are more abstract than our derivations. On the other hand, the information contained in these histories seems insufficient to obtain the analogous result for strong fairness ("local fairness" in |DM|). The distance proposed in |DM|, d_3, seems the most stringent that can be defined in that framework, nevertheless it does not give the wanted results. If we translate d_3 in our framework we obtain a distance on finite derivations only; let us call it d_3'. If $(\overline{FD},\bar{d}_3')$ denotes the completion of (FD,d_3'), then $\overline{FD}$ is in bijection with D^w (but $\bar{d}_3'$ is not equivalent to dw). In other words, the completion contains also the infinite computations which are weakly fair but fail to be strongly fair. The infinite strongly fair computations can only be characterized through "canonical" Cauchy sequences (of finite derivations). We do not see an immediate analogue of our distance ds within the framework in |DM|: histories lack the information we use in defining ds. Our feeling is that (part of) this additional information is actually needed. If one could make this conjecture precise and prove it, this result would be - in our opinion - much more interesting than those presented here.

On the positive side, we can say that, as pointed out by M. Hennessy |H|, the present work is not confined to CCS. Our characterization is based on the sets LP(-) and P(-) and more precisely on their properties, rather than their explicit definition. Therefore, everything could be "lifted up" to the more abstract setting of (general labelled) transition systems, axiomatizing the notion of subprocess and "live" subprocess.

ACKNOWLEDGEMENTS.

I would like to thank Pierpaolo Degano and Ugo Montanari for discussions concerning their paper and Edmund Robinson for suggesting a better presentation of some of the results. Robin Milner and Gordon Plotkin should be thanked for their encouragement. Finally, I have relied heavily on the work on fairness done together with Colin Stirling; he has also made useful comments on an earlier version of this paper.

REFERENCES [(*)]

AN A. ARNOLD, M. NIVAT. Metric interpretations of infinite trees and semantics of non-deterministic recursive procedures. T.C.S. 11 (1980) 181-205.

BZ J.W. DE BAKKER, J.I. ZUCKER. Processes and a fair semantics for the ADA rendez-vous. ICALP'83, LNCS 154 (1983) 52-66.

C G. COSTA. A metric characterization of fair computations in CCS. Technical Rep. CSR-169-84, Dept. Comput. Sci. Univ. Edinburgh (1984).

CS1 G. COSTA, C. STIRLING. A fair calculus of communicating systems. To appear in Acta Informatica; shortened version: FCT'83, LNCS 158 (1983) 94-105.

CS2 G. COSTA, C. STIRLING. Weak and strong fairness in CCS. Technical Rep. CSR-167-84, Dept. Comput. Sci. Univ. Edinburgh (1984); shortened version: MFCS'84, LNCS 176 (1984) 245-254.

DM P. DEGANO, U. MONTANARI. Liveness properties as convergence in metric spaces. Proc. 16th ACM STOC (1984).

H M. HENNESSY. Private communication.

M1 R. MILNER. A calculus of communicating systems. LNCS 92 (1980).

M2 R. MILNER. A finite delay operator in synchronous CCS. Technical Rep. CSR-116-82 Dept. Comput. Sci. Univ. Edinburgh (1982).

Pa1 D. PARK. On the semantics of fair parallelism. LNCS 86 (1980) 504-526.

Pa2 D. PARK. A predicate transformer for weak fair iteration. Proc. 6th IBM Symp. on Mathematical Foundat. of Comput. Sci. , Hakone, Japan (1981).

Pa3 D. PARK. The "fairness" problem and nondeterministic computing networks. Foundat. of Comput. Sci. IV, De Bakker - Van Leuven edit. Amsterdam (1982).

Pl G. PLOTKIN. A powerdomain for countable nondeterminism. ICALP'82, LNCS 140 (1982) 418-482.

(*) LNCS n stands for Lecture Notes in Computer Science Vol.n, Springer.

APPENDIX : AN EXAMPLE

We use the following abbreviations: $\underline{n}$ denotes the string composed of n 1's; when u is a label, E^{u} denotes the term obtained by prefixing all labels in E with u.

Now let:

$F = (\mathrm{fix}X)_{\varepsilon} . a_{\underline{2}}b_{\underline{3}}X_{\underline{4}} +_{1} \bar{c}_{12}X_{121}$; $H = (\mathrm{fix}X)_{\varepsilon} . c_{1}X_{11}$; $E_{0} = (F^{\underline{2}} |_{1} H^{12})(\backslash c)_{\varepsilon}$

According to the definitions above we have, say, $\underline{3}21 = 11121$ and

$F^{\underline{2}} = (\mathrm{fix}X)_{\underline{2}} . a_{\underline{4}}b_{\underline{5}}X_{\underline{6}} +_{\underline{3}} \bar{c}_{\underline{3}2}X_{\underline{3}21}$ and $H^{12} = (\mathrm{fix}X)_{12} . c_{121}X_{1211}$.

Then, a derivation from E_{0} in the labelled calculus is

$Y = E_{0} \xrightarrow{a} E_{1} \xrightarrow{b} E_{2} \xrightarrow{a} E_{3} \xrightarrow{b} E_{4} \xrightarrow{a} E_{5} \xrightarrow{b} E_{6} \xrightarrow{\tau} E_{7} \xrightarrow{a} E_{8}$.

The expressions for E_{i} (suppressing some of the labels) are given in the table below, where $u = \underline{15}21$ and $v = 121$. The table also shows the sets $LP(E_{i})$ and $P(E_{i})$ and the values of $lw(Y,i)$, $ls(Y,i)$ and $LS(Y,i)$, for $0 \leq i \leq 8$.

i	E_i	$P(E_i)$	$LP(E_i)$	lw(Y,i)	ls(Y,i)	LS(Y,i)
0	$(\ F^{\underline{2}} \mid H^{12})\backslash c$	$\{\underline{3}, v\}$	$\{\underline{3}, v\}$	0	0	0
1	$(b_{\underline{5}} F^{\underline{6}} \mid H^{12})\backslash c$	$\{\underline{5}, v\}$	$\{\underline{5}\}$	0	0	0
2	$(\ F^{\underline{6}} \mid H^{12})\backslash c$	$\{\underline{7}, v\}$	$\{\underline{7}, v\}$	1	0	0
3	$(b_{\underline{9}} F^{\underline{10}} \mid H^{12})\backslash c$	$\{\underline{9}, v\}$	$\{\underline{9}\}$	2	2	0
4	$(\ F^{\underline{10}} \mid H^{12})\backslash c$	$\{\underline{11}, v\}$	$\{\underline{11}, v\}$	3	0	0
5	$(b_{\underline{13}} F^{\underline{14}} \mid H^{12})\backslash c$	$\{\underline{13}, v\}$	$\{\underline{13}\}$	4	4	0
6	$(\ F^{\underline{14}} \mid H^{12})\backslash c$	$\{\underline{15}, v\}$	$\{\underline{15}, v\}$	5	0	0
7	$(\ F^{u} \mid H^{v1})\backslash c$	$\{u1, v\underline{2}\}$	$\{u1, v\underline{2}\}$	6	6	6
8	$(b_{u\underline{3}} F^{u\underline{4}} \mid H^{v1})\backslash c$	$\{u\underline{3}, v\underline{2}\}$	$\{u\underline{3}\}$	7	7	7

Moreover, recalling that $Y[i] = E_{0} \xrightarrow{a} \ldots \xrightarrow{.} E_{i}$, we have, for instance:

$LS(Y[5],i) = 0$, $0 \leq i \leq 4$; $LS(Y[5],5) = 4$; $LS(Y[6],i) = 0$, $0 \leq i \leq 6$.

$$ds(Y_{3},Y_{4}) = [LS(Y_{3},3) + 1]^{-1} + [LS(Y_{4},3) + 1]^{-1}$$

$$= [ls(Y_{3},3) + 1]^{-1} + [ls(Y_{4},4) + 1]^{-1} = 1/3 + 1 ;$$

$ds(Y_{3},Y_{i}) = 1/3 + 1$, $i=5,6,7,8$;

$$ds(Y_{7},Y_{8}) = [ls(Y_{7},7) + 1]^{-1} + [ls(Y_{8},7) + 1]^{-1} = 1/(6+1) + 1/(6+1).$$