Finitary Choice Cannot Express Fairness: A Metric Space Technique

Abha Moitra
Prakash Panangaden

TR 86-788

October 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853
Finitary Choice Cannot Express Fairness: A Metric Space Technique

Abha Moitra
Prakash Panangaden
Computer Science Department, Cornell University

November 10, 1986

Abstract

In this paper we use metric space arguments to settle an open problem relating to expressing fairness with traditional programming language constructs. The question is as follows. Given a programming language with a primitive to express finitary choice is it possible to express fairness? We show that the answer is no. We prove this result in two different programming paradigms. These are dataflow programs and CCS. Our proofs are based on very general topological considerations rather than on detailed scenario analysis. Furthermore our methods generalize in a straightforward way to any liveness properties.
1 Introduction

In this paper we establish two negative results about expressing fair computations using programming language constructs. The questions we address both ask the same basic question: is it possible to write an expression which captures all the fair behaviors of a system. We show that this cannot be done in general in CCS or in a traditional dataflow language with a primitive for bounded non-determinism. In this paper we will not discuss what fairness is exactly but rather we will choose examples which are fair by any reasonable definition of fairness. It has been remarked many times in the literature that bounded non-determinism in not sufficient for expressing fairness [19] [17]. A recent book by Francez [13] devoted to studying fairness does not contain any result of the type we have mentioned even though the relevant metric space techniques are discussed. This sort of result has been established formally only for limited cases, for example Stark [21] has shown such a property for semaphore primitives. The result which we present in section 2 is known to deBakker and Zucker [8] but they did not make the observation which constitutes the title of the present paper.

The arguments we present use metric space techniques to establish our results. Surprisingly, the results we need on metric spaces have been available in the literature for several years now [4], [5], [9], [8], [10], [14], [20]. There have also been explicit uses of metric space techniques to discuss fairness in CSS by Costa [6]. A general characterization of liveness properties using metric space methods has been given by Degano and Montanari [11] and similar topological characterizations have been given by Arnold and Nivat [3] and by Alpern and Schneider [2]. Our contribution is to use these results to establish limits on the ability to express fairness.

The first problem which we consider is the following. Suppose $B$ is a behavior expression in CCS [18] which expresses a non-deterministic computation. For example, $B$ could be defined by $B = (a + b).B$, where $a$ and $b$ are atomic actions and the $+$ construct in CCS models non-deterministic choice. The possible behaviors of this expression are represented by all possible infinite sequences of $a$s and $b$s. These will include sequences which after some finite prefix become sequences of pure $a$s or pure $b$s. The latter kind of sequence we will call unfair sequences since they correspond to the idea that one possibility is being consistently chosen over another. Is it
possible to write another expression in CCS \( \partial B \) which has all and only the fair behaviors of the expression \( B \)? Intuitively the answer should be “no” since CCS only provides a finitary choice construct whereas one expects that a fair expression will require some kind of unbounded choice. In fact this is the case. Viewed metrically, the result follows from the fact that CCS expressions can only define “closed” sets of behaviors whereas the set of fair behaviors can converge to an unfair behavior and hence cannot form a closed set in the metric topology. The subtle part of the proof is to show that even the use of recursive definitions of behavior expressions does not permit one to express unbounded choice.

The second problem has a very similar flavor but is expressed in the language of dataflow networks [7]. Suppose that we have a dataflow language in which one has the usual constructs for defining expressions. Suppose one adds to this a construct called \texttt{amb} which when given two arguments selects one or the other and discards the unselected argument. In the context of dataflow networks an \texttt{amb} node would be fed two streams of input tokens and would select one element from each stream. Implicit in the semantics of such a network is the idea that each choice is being made independently so that no attempt is being made to enforce fairness. The question we ask is can a fair \texttt{merge} operator be built using deterministic constructs and the \texttt{amb}? Once again the answer is negative, provided that the set of data tokens is finite. Essentially the reason for the latter restriction is that one can simulate unbounded branching if there are infinitely many different data tokens which can all be distinguished by a primitive equality test. This question has been probably asked many times but the earliest reference we have seen to it in the literature is in the discussion following Keller’s paper at IFIP 1977 [16]. There Dijkstra remarks “I would be quite disturbed if merge could be programmed using choice”. Essentially the same question was posed to the second author by Samson Abramsky in 1984 [1].
2 Expressing Fairness in CCS

In this section we show that it is impossible to take a general CCS behavior expression, $B$, and construct another behavior expression $\partial B$ with the property that $\partial B$ expresses all the fair behaviors of $B$. As discussed in the introduction, this result follows from the fact that with finitary choice one can only express closed sets of possible behaviors while the set of fair behaviors will not in general form a closed set. To make these remarks precise we need to spell out the topology on the set of possible behaviors. We will specify the topology as the topology induced by an appropriate metric. This is not strictly necessary, the topology can be easily defined independantly of the metric but the metric space arguments are a little more intuitive than the purely topological arguments.

First we need to define the notion of behavior which we use to establish our result. In CCS a behavior expression is defined starting from a set of possible atomic actions, $\Sigma$. A behavior expression is defined inductively by the following clauses:

1. The distinguished symbol $NIL$ is a behavior expression.

2. If $B_1$ and $B_2$ are behavior expressions then so is $B_1 + B_2$.

3. If $a$ is in $\Sigma$ and $B$ is a behavior expression then $a.B$ is a behavior expression.

4. If $a_1 \ldots a_n$ are in $\Sigma$ then $\text{rec}(b; (a_1 + \ldots + a_n).B)$ where $B$ is a behavior expression containing free occurrences of $b$, is also a behavior expression.

The last clause is notation for the behavior expression which solves the recursive equation $b = (a_1 + \ldots + a_n).B$. In addition, there are operations on behavior expressions called relabeling, restriction and composition. These do not add to what can be expressed with the notation above but are intended as basic operations which are useful in formalizing the idea of communication between autonomous computing agents. Since we are concerned with the expressiveness of CCS we shall ignore them in the subsequent discussion. A behavior expression can be taken to denote an unordered tree of atomic actions. The nodes are the atomic actions while the branching
represents the possible choices, denoted syntactically by the $+$ operator. These trees are called synchronization trees by Milner and form the basis of the semantic accounts of CCS. For our purposes a behavior will be a possible sequence of atomic actions and a behavior expression will be taken to denote the set of possible behaviors or, in other words, the set of possible paths in the synchronization tree. This (cruder) notion of process behavior will serve to distinguish between sets of fair behaviors and sets of behaviors which are expressible in CCS. Such models of CCS behavior are called trace models and are well known in the literature. We shall use the notation $\text{tr}[B]$ to represent the set of possible behaviors of a behavior expression $B$. With the synchronization tree model of CCS there come various notions of equivalence of behavior expressions. In a trace model the natural notion of equivalence is equality of the associated trace sets. With such a definition of equivalence it follows that (for example) $(a_1 + a_2).B = (a_1.B) + (a_2.B)$ and thus we do not lose any expressiveness by having the third clause above only permit a single atomic action to be appended onto an already defined behavior expression.

Given a set $\Sigma$ of elements, the set of all finite or infinite sequences of elements from $\Sigma$ will be written $\Sigma^\omega$. We shall use the notation $z[n]$ to represent the $n$th component of an element of $\Sigma^\omega$. We define a metric on $\Sigma^\omega$ as follows:

**Def 1** Let $z, y$ be in $\Sigma^\omega$. Let $n$ be the smallest integer such that $z[n] \neq y[n]$. Define $d(z, y)$ to be $2^{-n}$. If $z = y$ then $d(z, y)$ is defined to be 0.

It is well known that $d$ as defined above is a metric on $\Sigma^\omega$. Henceforth we shall view $\Sigma^\omega$ as a topological space equipped with the metric topology.

We shall prove by structural induction that any behavior expression defines a closed subset of $\Sigma^\omega$. This is expressed by the following theorem:

**Theorem 1** If $B$ is a behavior expression then $\text{tr}[B]$ is closed in $\Sigma^\omega$.

Proof: To establish such a theorem we need to specify the function $\text{tr}$ by induction on the structure of behavior expressions. We begin with the simple cases. The behavior expression $NIL$ defines the set which contains only the empty sequence. This set is certainly closed. In symbols $\text{tr}[NIL] = \{\varepsilon\}$. 

5
The next case is the choice construct. We have \( tr[B_1 + B_2] = tr[B_1] \cup tr[B_2] \). By the inductive hypothesis \( tr[B_1] \) and \( tr[B_2] \) are closed; since the union of a finite number of closed sets is also closed we have that \( tr[B_1 + B_2] \) is also closed.

The third case is \( B = a \cdot B' \). We first define \( tr \) in this case by \( tr[a \cdot B'] = \{a \cdot x | x \in tr[B']\} \). The notation \( a \cdot x \) stands for the sequence obtained by attaching the atomic action \( a \) to the front of the sequence \( x \). Here we assume that \( tr[B'] \) is a closed subset of \( \Sigma^w \). Let \( \{z_i\} \) be a Cauchy sequence in \( tr[B] \). By the definition of \( tr \), every member of the Cauchy sequence can be written as \( a \cdot y_i \) where \( y_i \notin B' \). Clearly, the \( y_i \) also forms a Cauchy sequence. Since \( tr[B'] \) is closed the Cauchy sequence \( \{y_i\} \) converges to some sequence \( y \in tr[B'] \). But then the Cauchy sequence \( \{z_i\} \) converges to \( a \cdot y \). By the definition of \( tr \) it follows that \( a \cdot y \) is in \( tr[B] \). Thus is this case also \( tr[B] \) is closed.

So far the arguments are routine. The subtlety arises in the case where we permit recursive definitions of behavior expressions. Here we need to define the meaning of \( tr \) through some sort of successive approximation scheme. The technique which we shall use to define the denotation of the recursive construct uses an inverse limit construction to express the successive approximation process. This will prove to be very useful since there are standard topological results about the closure properties of inverse limits.

Intuitively, the recursive behavior expressions correspond to the possibility of repeating some of the actions indefinitely. The successive approximations can be expressed as a sequence in the following way. We assume that the behavior expression is \( \text{rec}(b; (a_1 + \ldots + a_n) \cdot B) \).

\[
\begin{align*}
b_0 &= tr[B[b/NIL]] \\
b_{n+1} &= \{x \| a_i | i \in \{1 \ldots n\}, x \in b_n\}.
\end{align*}
\]

The notation \( x \| a \) signifies that the atomic action \( a \) is attached to the end of the sequence \( x \). If \( x \) is an infinite sequence then this operation will just produce \( x \). The above expressions just capture the idea that the recursion can be understood by iteratively unwinding the behavior expression. How do we form the "limit" of this sequence? Clearly we cannot just take the union of these sets. If we had a sequence of sequences we could define the
limit to be the least upper bound in the domain of sequences with the prefix ordering. Clearly $\Sigma^\omega$ can be viewed as a complete partial order with the prefix order. Unfortunately we have a sequence of sets of sequences.

We define a sequence of functions between the $b_i$s, $\phi_i : b_{i+1} \to b_i$ by the definition

$$x \parallel a_i \mapsto x.$$  

These functions are clearly continuous in the induced topologies of the subsets $b_i$. Now consider a typical sequence of the following form:

$$\alpha_i \in b_i, i \in N,$$

where each member of the sequence satisfies:

$$\phi_{i-1}(\alpha_i) = \alpha_{i-1}.$$

These sequences form chains in $\Sigma^\omega$ viewed as a complete partial order. Thus each such sequence has a limit $\alpha$ in $\Sigma^\omega$. Now we shall take the collection of limits of all such sequences to be the denotation of the recursive behavior expression. This construction is just an instance of the well known inverse limit construction [12]. This construction is illustrated in Fig 1 Note that each of the $b_i$s in this inverse limit construction is a Hausdorff space being a subspace of a metric space. It now follows by a standard theorem (see for example pg 429 of Dugundji [12]) that the limit space is closed in $\Sigma^\omega$.

It is of course possible to define the denotation of the recursive behavior expression without mentioning the elaborate machinery of inverse limits but with this way of expressing the construction we get for free the result that the resulting limit space is a closed subset of $\Sigma^\omega$.

With this theorem in hand we can easily exhibit the counterexample that was mentioned in the introduction. Consider a behavior expression $\text{rec}(b; (p + q).b)$. The set of traces for this expression is $\{p, q\}^\omega$. The set of fair behaviors for this is the subset of $\{p, q\}^\omega$ in which every sequence has the property that every suffix has occurrences of both $p$ and $q$. It is easy to see that this latter set is not closed in the metric topology; thus by our theorem no CCS expression can possibly express all and only the fair behaviors of $\text{rec}(b; (p + q).b)$.
Figure 1: The Inverse Limit Construction for $\text{rec}(b; (p + q).b)$. 
3 Expressing Fair Merge Using Choice

In this section we show that any network consisting of deterministic operators and the amb operator can produce only closed sets in an appropriate metric on the set of all streams. The theorem which we prove in this section requires that the alphabet on which the streams are defined be finite. We will exhibit an example where an infinite alphabet can be used to simulate unbounded choice.

Deterministic networks are described semantically by order-continuous functions on the domain of streams [15]. The domain of streams over an alphabet $\Sigma$ is an $\omega$-algebraic complete partial order. Henceforth we shall use the word "domain" as an abbreviation for $\omega$-algebraic complete partial order. We can metrize this domain by a metric $d_1$ so that the Scott continuous functions are also closed mappings in the metric topology. This is done by several authors including Comyn and Dauchet [5] and deBakker and Zucker [9]. It is a standard result of general topology that a continuous function between compact metric spaces is a closed map as well [12]. Unfortunately, it is not the case that a continuous function in the order sense gives rise to a metric continuous function, so we cannot just use the theorem.

The metric $d_1$ that we will use is one used by Comyn and Dauchet [5] and is defined as follows. Since the basis $B$ of $\Sigma^\omega$ is countable, a numbering on $\Sigma; \phi : N \to B$ can be defined. The basis $B$ of $\Sigma^\omega$ is $\Sigma^*$, the set of words over $\Sigma$. First we define symmetric difference between two elements $x$ and $y$ of $\Sigma^\omega$.

**Def 2** $\Delta y = \{ s \in \Sigma^* | s \leq x \wedge s \not\leq y \text{ or } s \leq y \wedge s \not\leq x \}$

where $\Delta y$ is the notation for the symmetric difference. Now the metric $d_1 : \Sigma^\omega \times \Sigma^\omega \to R$ can be defined.

**Def 3**

$$d_1(x, y) = \begin{cases} 0 & \text{if } x = y \\ \frac{1}{1 + \mu[n|\phi(n) \in \Delta y]} & \text{otherwise} \end{cases}$$

The metric notion of convergence and the order-theoretic notion of convergence must be related. The former is defined in terms of Cauchy sequences
while the latter is defined in terms of directed sets or chains. Now a Cauchy sequence is in general not directed but a directed set can be associated with a Cauchy sequence as shown in Comyn and Dauchet [5]. First define a regulator $\chi$ which indicates at what point in a sequence the elements are less than $1/n$ apart. Then on this regulator $\chi$ another regulator called the standard regulator $\psi$ can be defined which gives the minimum distance in a sequence so that the elements are separated by less than $1/n$.

**Def 4** Let $\beta = \{s_i | i \in N\}$ be a Cauchy sequence in $\Sigma^\omega$. Let $\chi$ satisfy $\forall p \geq \chi(n), \forall q \geq \chi(n), d_1(s_p, s_q) < 1/n$. Then we can define $\psi$ by the equation $\psi(n) = \mu p[\forall (i, j) \in N \ast N, i \geq p, j \geq p, d_1(s_i, s_j) < 1/n]$.

Now we define the direction of a Cauchy sequence $\beta$. This is a directed set, $\text{Dir}(\beta)$, (in the order-theoretic sense) which has a least upper bound (lub) equal to the metric limit of the original sequence.

**Def 5** $\text{Dir}(\beta) = \{a_i | i \in N\}$ where $a_i = \{\phi(j) | j \leq i \land \phi(j) \leq s_{\phi(j)}\}$.

Proposition 6 of [5] shows that $\lim \beta = \text{lub}(\text{Dir}(\beta))$. This ability to build directed sets from Cauchy sequences is the key to proving the main theorem. In the context of this paper we will take the operators in a dataflow network to be functions from sets of input values to sets of input values. We will also assume that if a particular value is in the set of possible values then so are all its prefixes. This will allow us to prove the theorem and is a harmless modification to the normal way of expressing the semantics. It amounts to using the Hoare powerdomain. Operationally speaking, this assumption means that we view all the possible partial results that we may see as a stream is being produced as possible results. This is a relatively crude view of the semantics in the sense that equality in the model is quite crude; not all the distinctions that we could have made are being made. The point of this paper is not to provide the definitive model of non-determinate computation but rather to show that even with this imprecise semantic theory we can see that fairness is not expressible.

**Theorem 2** Any order-continuous function on $\Sigma^\omega$ when extended pointwise to sets of streams will take prefix-closed and metric-closed sets to prefix-closed and metric-closed sets. Metric-closed means closed with respect to the topology induced by the metric $d_1$. 

10
**Proof:** Consider any arbitrary deterministic operator $f$ with input channels $U^1, U^2, \ldots, U^n$ and output channels $Z^1, Z^2, \ldots, Z^m$. Let $\beta = (Z^k)_{i \in N}$ be a Cauchy sequence from the set of outputs on the channel $Z^h$. From $\beta$ we can construct the directed sequence $\text{Dir}(\beta)$. Since the inputs are prefix-closed it follows that we can build a directed sequence of inputs $\{\alpha_i\}$ such that $f(\{\alpha_i\}) = \text{Dir}(\beta)$. Let lub $(\alpha_i) = \alpha$; now since the inputs are closed under the metric $d_1$ it follows that $\alpha$ is in the input set. Therefore $f(\alpha) = f(\text{lub}(\alpha_i)) = \text{lub} (f(\alpha_i)) = \text{lub}(\text{Dir}(\beta)) = \lim \beta$. Hence $\lim \beta$ is a possible output and hence the set of possible outputs is closed in the metric sense. This completes the proof of the theorem. We have not used any mechanism to handle recursive definition because we know from Kahn’s results that all such networks will denote stream–functions which are Scott–continuous.

Note that we have not claimed that in general the order–continuous functions define either metric continuous functions or closed maps in the metric sense. The best result one can hope for is that an *increasing* metric–continuous function is order–continuous.

The metric $d_1$ is not the most convenient for use in our actual examples but it is equivalent to another metric which has been often used before. We now define this other metric $d_2 : \Sigma^\omega X \Sigma^\omega \rightarrow R$ as follows

**Def 6**

$$d_2(x, y) = \begin{cases} 
0 & \text{if } x = y \\
\frac{1}{\mu_n(x[n] \neq y[n])} & \text{otherwise}
\end{cases}$$

The reason we can switch between these two metrics is that Comyn and Dauchet [5] have shown that they are equivalent whenever $\text{Card}(\Sigma)$ is finite. We will later also discuss what are the consequences when the cardinality of $\Sigma$ is not finite.

Now the non-deterministic operator $\text{amb}$ can be modelled as a deterministic operator $\text{d-amb}$ as follows. $\text{d-amb}$ has three input channels, $U, V$ and $W$ and one output channel $Z$. If the input token on $W$ is 0 (1) then $\text{d-amb}$ passes on the first token on $U$ ($V$) to the output channel $Z$ and discards the first token on $V$ ($U$).

With this observation and the theorem it immediately follows that any network composed of deterministic operators and the $\text{amb}$ operator over a finite alphabet generates prefix and metric closed sets whenever the inputs
are prefix and metric closed. Now if we consider a fair merge operator and we feed as the inputs $a^\infty$ and $b^\infty$ then the set of possible outputs will contain streams of the form $a^n b(ab)^\infty$. These form a Cauchy sequence converging metrically to $a^\infty$. This last stream is excluded as a possible output since it would correspond to starving the channel carrying the $b$s. It follows then that it is impossible to realize fair merge using amb operator along with the standard deterministic operators if the alphabet is finite.

We now consider what happens when the cardinality of $\Sigma$ is not finite. In that case the argument used above does not go through and modulo certain technical details we can construct a fair merge as follows. Construct an operator d-merge with three input channels, $U, V$ and $W$ and one output channel $Z$. The input stream on channel $W$ is from the alphabet $N$ - the set of all integers greater than 0. The d-merge operates by alternating the consumption of tokens from the two channels $U$ and $V$. The number of tokens consumed from the chosen channel is dictated by the token value on channel $W$. Of course, if there are not enough tokens on a channel to meet the number specified by the token on $W$ then a deadlock will result even though there may be unconsumed tokens on the other channel. The reason for not allowing a token with value 0 on $W$ is to avoid consuming an infinite number of tokens from $U$ (or $V$). So if there are an infinite number of tokens on $U$ and $V$ then no deadlock can occur.

On the other hand if the cardinality of the alphabet underlying channel $W$ is finite then d-merge does not realize all fair behaviours even if we allow the possibility of deadlock. It might seem that one could use a distinguished token to serve as a delimiter so that with the remaining tokens one could have the effect of infinitely many different token values. It is easy to see that this does not work because the token being used as a delimiter can be interspersed in such a way as to permit unfair behaviors.
4 Conclusion

In this paper we have shown that finitary choice constructs cannot express the unbounded nondeterminacy needed to express fairness properties. The techniques we use are metric space techniques which have been already used by many authors to study fairness properties. The results we have obtained settle questions whose answer have been known for a long time but for which formal proofs have been lacking. The work of deBakker's group [9] [8] contains essentially the result we have proved in section 2. They prove that the limit of sequences of finite sets is compact and observe that when $\Sigma$ is finite then compactness co-incides with closedness. The dataflow version which we have presented in section 3 appears never to have been worked out before. It appears that metric space techniques offer interesting alternatives to the more standard domain theoretic methods of Scott and Plotkin [19]. The key ingredient which seems to make the metric space methods work is that the metric characterization essentially carries negative information whereas the domain theoretic methods only convey positive information. Technically this means that the metric topology defines the (much finer) Lawson topology on a complete partial order instead of the Scott topology. It is fairly clear that our results can be generalized to discuss the expressibility of any liveness properties rather than just fairness.
References


