CONCURRENCY AND AUTOMATA ON INFINITE SEQUENCES

David Park
Computer Science Dept.,
University of Warwick,
Coventry CV4 7AL,
England.

Abstract: The paper is concerned with ways in which fair concurrency can be modelled using notations for omega-regular languages - languages containing infinite sequences, whose recognizers are modified forms of Büchi or Muller-McNaughton automata. There are characterization of these languages in terms of recursion equation sets which involve both minimal and maximal fixpoint operators. The class of ω-regular languages is closed under a fair concurrency operator. A general method for proving/deciding equivalences between such languages is obtained, derived from Milner's notion of "simulation".

1. Introduction

If a construct $(C_1 \underline{\text{par}}\ C_2)$ is understood to call for concurrent interpretation of commands, 'fairness' is the constraint on the language involved which guarantees that

$$x:=\underline{\text{true}};\ y:=1;\ ((\underline{\text{while}}\ x\ \underline{\text{do}}\ y:=y+1)\ \underline{\text{par}}\ x:=\underline{\text{false}})$$

always terminates, though without guaranteeing any bound on the value of y when it does so. The constraint appears to present a number of fundamental problems - one reason being its association with'unbounded nondeterminism', as exhibited by the value of y in the example, which comes out with a defined, but unbounded value. Apparently this variety of nondeterminism can be given an accurate semantics only by resorting to features - such as the use of monotone, non-continuous functions - which do not otherwise occur in computation theory. Thus the rich variety of models due to Scott, Plotkin and others is not apparently available, so that satisfactory answers seem to call for a return to more primitive notions.

In a previous paper [11] a semantics was obtained for a primitive programming language with fair concurrency. This involved the use of what are here called ω-regular expressions. This paper will be concerned primarily with the theory surrounding such notations. We hope to discuss practical examples in a later paper, but briefly point out here the sort of reasoning we would expect to be useful in practice.

One would expect that a construct such as

$$\underline{\text{while}}\ E\ \underline{\text{do}}\ C$$

should be associated with an ω-regular expression of the form

$$A^{\dagger}(\equiv A^{*} + A^{\omega})$$

where A is determined by E and C. And concurrency (the fair merging of command

sequences) will turn out to be expressible by an operator

$$A||B$$

on ω-regular languages. A sequence of commands such as the initial example should then be associated with something like

$$A(B^{\dagger}||c)$$

where c corresponds to the 'atomic' command x:=false. The crucial equivalence in establishing termination is then

$$A(B^{\dagger}||c) = AB^{*}(B||c)B^{\dagger}$$

which captures the fact that only finitely many Bs occur before c is obeyed. Reference to the interpretations involved should then allow replacement of $(B||c)B^{+}$ by $(B||c)$, reducing the expression $A(B^{\dagger}||c)$ to $AB^{*}(B||c)$, consisting only of finite sequences, and therefore with guaranteed termination.

2. Basic Notions

Given any set Σ, $\Sigma^{\dagger}$ denotes the extended sequences over Σ

$$\Sigma^{\dagger} = \Sigma^{*} \cup \Sigma^{\omega}$$

Σ^{*} being the finite sequences, Σ^{ω} the infinite sequences over Σ. Subsets of $\Sigma^{\dagger}$ are extended languages:

For $\sigma \in \Sigma$, $x \in \Sigma^{*}$, $y \in \Sigma^{+}$, $X, Y \subseteq \Sigma^{+}$

λ denotes the null sequence

xy denotes the concatenation of x with y

XY denotes $\{xy \mid x \in \Sigma^{*} \cap X, y \in Y\} \cup (X \cap \Sigma^{\omega})$

The notion of concatenation here is different from that of [11] in that

$$X\phi = X \cap \Sigma^{\omega}$$

is not necessarily empty. The intuitive justification for this definition should be clear - the set of computation paths through a serial composition of programs should be the concatenation of computation sets for its components.

Apart from $X\phi = \phi$, other expected elementary properties do hold for this definition. Thus:

$$X(YZ) = (XY)Z$$
$$X(Y \cup Z) = XY \cup XZ$$
$$(X \cup Y)Z = XZ \cup YZ$$
$$\phi X = \phi$$
$$\{\lambda\}X = X\{\lambda\} = X$$

There is a variety of iteration operators:

star-closure $A^* = \{\lambda\} \cup A \cup A^2 \cup \ldots\ldots$

omega-closure $A^\omega = \Sigma^\dagger$ if $\lambda \in A$

$= \{w_o w_1 w_2 \ldots\ldots \mid w_i \in A\} \cup (A^* \cap \Sigma^\omega)$ otherwise

dagger-closure $A^\dagger = A^* \cup A^\omega$

[Note: it is convenient to take $\lambda^\omega = \Sigma^\dagger$, for consistency with the fixpoint versions below. Notice the case for A^ω when A contains infinite sequences.]

Iteration operators satisfy fixpoint equations

$$A^* = \{\lambda\} \cup AA^* = \{\lambda\} \cup A^*A$$
$$A^\omega = AA^\omega$$
$$A^\dagger = \{\lambda\} \cup AA^\dagger = \{\lambda\} \cup A^\dagger A$$

also $(\{\lambda\} \cup A)^* = A^*$

but $(\{\lambda\} \cup A)^\dagger = (\{\lambda\} \cup A)^\omega = \Sigma^\dagger$

cf. Salomaa [13].

3. Omega-regular expressions:

One proceeds by straightforward analogy, to develop a formalism with alphabet formed from $\{\phi, +, *, \omega, (,)\}$, using syntax variables $\sigma \in \Sigma$, and $\varepsilon \in L$

$$\varepsilon :: = \sigma \mid \phi \mid (\varepsilon + \varepsilon) \mid \varepsilon\varepsilon \mid \varepsilon^* \mid \varepsilon^\omega$$

One generates the language L of omega-regular expressions over Σ. The semantics of L should be clear from the previous section. The standard regular expressions over Σ are obtained by omitting the final clause of the grammar

$\lambda, A^\dagger$ are introduced by definition

$\lambda \equiv \phi^*$ $\qquad A^\dagger \equiv A^\omega + A^*$

We will use expressions from L ourselves in preference to standard set theoretic expressions.

Some interesting expressions over $\{0,1\}$ are

$(0^*1)^\omega$ = {strings with infinitely many 1s}

$(1^*0)^\omega$ = {strings with infinitely many 0s}

$(0^*11^*0)^\omega = (00^*1 + 11^*0)^\omega$

= {strings with infinitely many 1s and 0s}

The extended languages denotable by omega-regular expressions are omega-regular languages. We are interested in establishing identities between such extended languages. An equational system in the style of Salomaa [13] might therefore be possible. Remarkably, all except one of the axioms and rules of the system F_1 [13] are still valid. But the exception, "Arden's rule", is important, since its validity depends on the unique fixpoint property, that for $\lambda \notin X$:

$$X = AX + B \quad \text{iff} \quad X = A^*B$$

For omega-regular expressions this fails. $A^\dagger B$ is another solution. And there may be intermediate solutions:

$$(0^*1)^\omega \quad (1^*0)^\omega \quad (0^*11^*0)^\omega$$

all satisfy $X = 0X + \phi$, but none of them is either $0^*\phi = \phi$ or $0^\dagger\phi = 0^\omega$.

There is a normal form theorem:

Theorem 3.1: Every ω-regular A can be written in the form

$$A = \text{fin}(A) + \text{inf}(A)$$

with A regular (i.e. mentioning no B^ω), and inf (A) of the form of a finite sum

$$\text{inf}(A) = \Sigma_i B_i C_i^\omega$$

with B_i, C_i regular, $\lambda \notin C_i$.

The proof is by induction on the form of A; fin (A) is the result of replacing any subexpression B^ω by ϕ (or by Σ^*, if $\lambda \in B$); also

$$\begin{aligned}
&\text{inf}(\sigma) = \text{inf}(\phi) = \phi \\
&\text{inf}(A+B) = \text{inf}(A) + \text{inf}(B) \\
&\text{inf}(AB) = \text{fin}(A)\,\text{inf}(B) + \text{inf}(A) \\
&\text{inf}(A^*) = \text{fin}(A)^*\,\text{inf}(A) \\
&\text{inf}(A^\omega) = \Sigma^\omega \text{ if } \lambda \in A \\
&\qquad\qquad \text{fin}(A)^*\,\text{inf}(A) + (\text{fin}(A))^\omega \text{ otherwise.}
\end{aligned}$$

4. Fixpoints of linear recursion equations:

For a summary of the relevant fixpoint theory, see [11].

Since the class of extended languages forms a complete lattice, monotone functions on it have maximal as well as minimal fixpoints. Use regular expressions over $\Sigma \cup \{X\}$ to denote such functions, and forms

$$\mu X.\ F(X) \qquad \nu X.\ F(X)$$

to denote minimal and maximal fixpoints respectively, of such functions. Now we have

Theorem 4.1:
1) $A^*B = \mu X.\ (AX + B)$
2) $A^\omega = \nu X.\ (AX)$
3) $A^\dagger B = \nu X.\ (AX + B)$

Proof: The standard proof for (1) is still valid. As regards (2), if $\lambda \in A$ then $\Sigma^\dagger = A\Sigma^\dagger$, so $A^\omega = \Sigma^\dagger = \nu X.\ AX$ ($\Sigma^\dagger$ is the maximal language). If $\lambda \notin A$, then A^ω satisfies $X = AX$, from its definition; so $A^\omega \subseteq \nu X.\ (AX)$. Conversely, suppose $w \in \nu X.\ AX$; then $w \in A\nu X.AX$. Then either $w \in \text{inf}(A) \subseteq A^\omega$; or w has the form $w_1 w'$ for some $w_1 \in A$, $w' \in \nu X.AX$. And this step can be repeated to obtain either an

infinite sequence $w = w_1w_2w_3 \ldots\ldots$, with $w_i \in fin(A)$ or a finite sequence $w = w_1w_2 \ldots w_n$ with $w_n \in inf(A)$.

The proof of (3) is similar.

Caution: It is surprisingly easy to construct fallacious proofs about omega-regular languages. The following is a counterexample to one tempting assumption:

Take $A = (2(0+1)^* + 0)$, $s = 21010^210^31 \ldots$

Then s has a prefix in each A^k, $k > 0$, but $s \notin A^\omega$. Technically this is a failure of cocontinuity of the function

$$f(X) = AX$$

More strongly, this is an example where

$$A^\omega \neq \bigcap_{i=0}^{\infty} A^i\Sigma^\omega$$

holds - contra intuition.

The fixpoint identities can be used to eliminate iteration operators in favour of various fixpoints. For example

$$\begin{aligned} 0^*1(0^*11)^\omega &= \mu X.(0X + 1(0^*11)^\omega) \\ &= \mu X.(0X + 1\mu Y.(0^*11Y)) \\ &= \mu X.(0X + 1\mu Y.\mu Z.(0Z + 11Y)) \\ &= \mu X.(0X + 1\mu Y.\mu Z.(0Z + 1\,\mu W.1Y)) \end{aligned}$$

The last right hand side can be written in a less opaque notation, using recursion equations - writing the conventional $\Leftarrow$ where a minimal fixpoint is intended, and $\Rightarrow$ in case of a maximal fixpoint. Thus

$$\begin{aligned} X &\Leftarrow 0X + 1Y \\ Y &\Rightarrow Z \\ Z &\Leftarrow 0Z + 1W \\ W &\Leftarrow 1Y \end{aligned}$$

The sequence in which this set of equations is written is significant, since it conveys functional dependencies between the equations, which affect the behaviour of fixpoint operators. In the above set, the solution obtained for Z is got by minimizing over possible solutions after eliminating W, and has Y (and in principle X) as parameters.

Strictly, we are regarding subsets of $\Sigma \dagger \{X_1, X_2, \ldots . X_n\}$ as defining n-ary linear functions on extended languages. Manipulations like those above are justified since the operators involved respect substitutions for free parameters in linear (i.e. rightmost) contexts. (Not true for all contexts -- consider $X\emptyset$). If such a language of linear forms is ω-regular, it can be denoted by an expression

$$F(X_1 \ldots X_n) = A + \sum_i A_i X_i$$

for A, A_i ω-regular over Σ.

The function is atomic linear if, in addition each A, A_i is a finite sum of elements selected from $\Sigma \cup \{\lambda\}$.

A linear system (of recursion equations) (with free parameters $\{Y_1 \ldots Y_m\}$) is a sequence of equations of the form

$$\begin{aligned} X_1 &\Leftrightarrow F_1(X_1 \ldots X_n, Y_1 \ldots Y_m) \\ X_2 &\Leftrightarrow F_2(X_1 \ldots X_n, Y_1 \ldots Y_m) \\ &\cdots\cdots \\ X_n &\Leftrightarrow F_n(X_1 \ldots X_n, Y_1 \ldots Y_m) \end{aligned}$$

with each F_i linear over $\Sigma \cup \{X_1 \ldots X_n\} \cup \{Y_1 \ldots Y_m\}$ and each occurrence of $\Leftrightarrow$ standing for either $\Leftarrow$ or $\Rightarrow$. The intended solution to such a system can be defined by induction on n, to obtain solutions for $X_1 \ldots X_n$ which are linear over $\Sigma \cup \{Y_1 \ldots Y_n\}$. Given n+1 equations for $X_1 \ldots X_{n+1}$, solve the last n, to obtain solutions for $X_1 \ldots X_n$ linear over $\Sigma \cup \{X_1\} \cup \{Y_1 \ldots Y_n\}$. Substitute for these into the equation for X_1 and obtain a solution (maximal if $\Rightarrow$, minimal if $\Leftarrow$) linear over $\Sigma \cup \{Y_1 \ldots Y_n\}$. Substitute this solution into those already obtained for $X_2 \ldots X_n$ to obtain the remaining components.

The formal manipulations corresponding to these steps are applications of Theorem 4.1 (1) and (3). At the (n-k+1)th step this means solving an equation

$$X_k \Leftrightarrow B_k + \sum_i^k B_{ik} X_i$$

with B_k a linear form over $\{Y_1 \ldots Y_m\}$ and B_{ik} ω-regular to obtain the solution

$$X_k = B_{kk}^{*} (B_k + \sum_i^{k-1} B_{ik} X_i)$$

[or the same form, with $\dagger$ in place of *, in the case of $\Rightarrow$].
This solution can then be substituted back into the remaining equations for $X_1 \ldots X_{k-1}$, and into the linear forms already obtained for $X_k, \ldots X_n$. These steps conform to the denotation rules given above for the system; and they preserve linearity of the forms involved - from associativity and distributivity rules.

One might expect a hierarchy structure on ω-regular languages dependent on the sequence of $\Rightarrow$, and $\Leftarrow$ operators needed to define them. It is well-known, for example, that precisely the (standard) regular sets are definable by atomic linear systems involving only the minimum fixpointing $\Leftarrow$. The proof of this is closely related to that of Kleene's theorem. But in fact for ω-regular sets only one alternation of fixpoints above is needed. Systems in which no $\Leftarrow$ precedes a $\Rightarrow$ equation can define arbitrary ω-regular sets. This follows from the normal form result 3.1

$$A = D + \sum_i B_i C_i^{\omega}$$

for D, B_i, C_i all regular, $\lambda \notin C_i$. The equations needed to define A can be taken as

$$X \Rightarrow Y + \sum_{i=1}^{n} Z_i$$

$$W_1 \Rightarrow V_1$$

$$W_2 \Rightarrow V_2$$

$$\cdots\cdots$$

$$W_n \Rightarrow V_n$$

followed by equations using $\Leftarrow$ to define

$$Y = D$$

$$Z_i = B_i W_i \qquad i = 1,2 \ldots n$$

$$V_i = C_i W_i \qquad i = 1,2 \ldots n$$

which are all standard regular in $\Sigma \cup \{W_i\}$.
(Precise equations may be obtained from the transition rules for nondeterministic recognisers for these sets, in an obvious way). These results are summarised in the following

Theorem 4.2: The following are identical

1) the class of ω-regular languages
2) the class of languages denotable by linear systems of equations
3) the class of languages denotable by atomic linear systems of the form

$$X_1 \Rightarrow F_1(X_1 \ldots X_n)$$

$$\cdots\cdots$$

$$X_k \Rightarrow F_k(X_1 \ldots X_n)$$

$$X_{k+1} \Leftarrow F_{k+1}(X_1 \ldots X_n)$$

$$\cdots\cdots$$

$$X_n \Leftarrow F_n(X_1 \ldots X_n)$$

Call this last form of system an atomic max-min system.

5. Omega Automata

These are formed from standard finite automata by the addition of structure for 'recognising' infinite sequences. The automata specified here differ from those usually defined in that they also recognise finite sequences (in the standard way). The resulting modifications do not essentially alter the existing theory.

The reader will be familiar with standard (nondeterministic) finite automata (with ε-moves, here). Our notation for such automata over Σ is

$$\mathbb{M} = \langle S, s_o, M, F \rangle$$

with S the state set of $\mathbb{M}$, with start state $s_o \in S$, and with accept states $F \subseteq S$ transition function $M : S \times (\Sigma \cup \{\lambda\}) \to P(S)$. $\overline{M} : S \times \Sigma^* \to P(S)$ is the usual extension

of M to finite sequences. The finite tapes accepted by $\mathcal{M}$ are then fin ($\mathcal{M}$) = {w | $\overline{M}$ (s_o,w) ∩ F ≠ ϕ}. We will need to talk about paths $\pi \in S^t$ which lead from a state s ∈ S,and correspond (via iterations of M) to a sequence $w \in \Sigma^t$; we refer to this relation as π ∈ paths (s,w). If $\pi \in S^\omega$, In(π) is the set of states which occur infinitely often in π.

An omega-automaton is a finite automaton to which some additional structure is added which allows a set inf($\mathcal{M}$) to be recognised consisting of infinite sequences. The two principal varieties are

B-automaton: an additional set G ⊆ S of green states is specified

inf ($\mathcal{M}$) = {w | there exists $\pi \in$ paths (s_o, w), with In(π) ∩ G ≠ ϕ} ⊆ Σ^ω

(introduced by Buchi [3]).

M-automaton: a set $\mathbb{D} \subseteq P(S)$ of accepting sets is specified

inf ($\mathcal{M}$) = {w| there exists $\pi \in$ paths (s_o,w), with In(π) ∈ $\mathbb{D}$} ⊆ Σ^ω

(first mentioned in Muller [9] - in the context of concurrent circuit theory; see particularly McNaughton [8] and Choueka [4]).

The combined T($\mathcal{M}$) ⊆ Σ^t is now

$$T(\mathcal{M}) = \text{fin}(\mathcal{M}) \cup \text{inf}(\mathcal{M})$$

[Note: the condition that inf($\mathcal{M}$) ⊆ Σ^ω consist of only infinite sequences is intended to imply that no sequence of ε-moves cycles through an accepting set.]

Every B-automaton can be converted to an equivalent M-automaton, taking

$\mathbb{D}$ = {T | G ⊆ T ⊆ S}

Theorem 5.1: The following classes are identical

1)the class of **ω-regular languages**

2)the class of **sets of the form** T($\mathcal{M}$) $\mathcal{M}$ **a non-deterministic B-automaton**

3)the class of sets of the form T($\mathcal{M}$) $\mathcal{M}$ a non-deterministic M-automaton

4)the class of sets of the form T($\mathcal{M}$) $\mathcal{M}$ a deterministic M-automaton

One of the steps here is notoriously hard. This is the inequality (1) ⊆ (4), which can be established by the ingenious and elaborate construction in McNaughton [8] , which we cannot pursue here. Minor modifications are required to all proofs to deal with finite strings, which may be accepted by the automata specified here. These rest on observations that the results can be decomposed into separate discussion of the finite and the infinite sequences involved. For just finite sequences, the results are embodied in the familiar development of Kleene's theorem, while the infinite results are precisely those available in the literature on Büchi and Muller-McNaughton automata. The other observations called for are that each of the classes is closed under union (for class (4) a deterministic recogniser is needed - which follows from a product construction analogous to that used in Rabin-Scott [12]).

We should discuss the result that B-automata recognise precisely the ω-regular languages; this follows directly from the observation of 4.2(3), that ω-regular languages are recognisable by atomic linear systems of the form

$$X_1 \Rightarrow \sum_{j=1}^{n} A_{1j}X_j + A_1$$
$$\cdots\cdots\cdots\cdots$$
$$X_k \Rightarrow \cdots\cdots\cdots$$
$$X_{k+1} \Leftarrow \cdots\cdot$$
$$\cdots\cdots\cdots$$
$$X_n \Leftarrow \sum_{j=1}^{n} A_{nj}X_j + A_n$$

with $A_i, A_{ij} \in$ {finite sums of elements from $\Sigma \cup \{\lambda\}$}

The correspondence is very close.

Take $\mathfrak{M} = < S, s_o, M, F, G >$ with

$$S = \{X_1 \ldots X_n, \lambda\}$$
$$s_o = X_1$$
$$F = \{\lambda\}$$
$$G = \{X_1 \ldots X_k\}$$

and $M(X_i,\sigma) = \{X_j \mid \sigma = A_{ij}\} \cup \{\lambda \mid \sigma = A_i\}$

We now want to show that $T(\mathfrak{M})$ is the same language as is obtained by solving the equation set.

First, solve the last n-k equations, substituting back at each stage to obtain a sequence of systems

$$X_1 \Rightarrow \sum_{j}^{m} B^m_{1j}X_j + B^m_1$$
$$\cdots\cdots\cdots\cdots$$
$$X_k \Rightarrow \cdots\cdots\cdots$$
$$X_{k+1} \Leftarrow \cdot\cdot$$
$$\cdots\cdots\cdots$$
$$X_m \Leftarrow \sum_{j}^{m} B^m_{mj}X_j + B^m_m$$

Here
$$B^{m-1}_{ij} = B^m_{ij} + B^m_{im}B^{m*}_{mm}B^m_{mj}$$
$$B^{m-1}_{i} = B^m_{i} + B^m_{im}B^{m*}_{mm}B^m_{m} \qquad i, j, k < m \leq n$$

This is the familiar sequence of identities used in the construction of regular expressions from finite automata. Use the notations:

$C(s,s',m) = \{w \mid$ some $\pi \in$ paths(s,w) reaches s', passing through no X_q, $q \leq m\} \subseteq \Sigma^*$

$D(s,m) = \{w \mid$ some $\pi \in$ paths(s,w) passes through no X_q, $q \leq m$, but $X_p \in$ In(π) for some $m < p \leq k\} \subseteq \Sigma^\omega$

then $B^m_{ij} = C(X_i, X_j, m)$

$B^m_i = C(X_i, \lambda, m)$ $\qquad k \leqslant m, \quad i,j \leqslant m$

Above k, we have

$$B^{m-1}_{ij} = B^m_{ij} + B^m_{im} B^m_{mm} \dagger B^m_{mj}$$
$$B^{m-1}_{i} = B^m_{i} + B^m_{im} B^m_{mm} \dagger B^m_{m} \qquad i, j < m \leqslant k$$

And now, by induction on k-m:

$B^m_{ij} = C(X_i, X_j, m) \cup D(X_i, m)$

$B^m_i = C(X_i, \lambda, m) \cup D(X_i, m)$ $\qquad i,j \leqslant m < k$

So finally, the solution for X_1 is

$$X_1 = B^\circ_1 = \{w \mid \text{some } \pi \in \text{paths}(X_1, w) \text{ has } X_p \in \text{In}(\pi), \text{some } p \leqslant k\}$$
$$= T(\mathcal{M})$$

[Note: if $\lambda \in B^m_{mm}$, some $m \leqslant k$, then $\mathcal{M}$ has an accepting ε-loop, and must be amended, so that all sequences, finite and infinite, are accepted from X_m. In this case we need $M(X_m, \sigma) = X_m$, $\sigma \in \Sigma$; and X_m must be added to F. We omit proof that the resulting $\mathcal{M}$ accepts the solution to the equation sequence, in this case.]

It is worth noting that this direct relationship of max-min sets with B-automata generalises to 'min-max-min sets' of atomic linear equations with alternations looking like

$X_1 \Leftarrow$
$\cdots\cdots\cdots$
$X_j \Rightarrow$
$\cdots\cdots\cdots$
$X_n \Leftarrow$

These correspond to a variety of automata used in McNaughton [8]. The varieties defined by the initial $\Leftarrow$ equations correspond to a set of 'red' states R. The acceptance criteria are as for B-automata, except for requiring that R be passed through only finitely often - i.e. $\text{In}(\pi) \cap R = \phi$.

From (1) = (4) it follows that the ω-regular sets, as defined here, are closed under complementation as well as under union, since an obvious complementation construction applies to deterministic M-automata. So ω-regular sets are closed under intersection also.

6. Concurrency Operators:

The deterministic merge function

$$\text{dmerge} : \Sigma^\dagger \times \Sigma^\dagger \times \{0,1\}^\omega \to \Sigma^\dagger$$

is the unique function satisfying

$$\text{dmerge}(\lambda,w,d) = \text{dmerge}(w,\lambda,d) = w$$
$$\text{dmerge}(\sigma w,w',\sigma d) = \sigma\ \text{dmerge}(w,w',d)$$
$$\text{dmerge}(w,\sigma w',\sigma d) = \sigma\ \text{dmerge}(w,w',d)$$

[d is usually called an oracle for the merge].

$$w'' \quad \underline{\text{merges}} \quad w,w' \text{ if}$$
$$w'' = \text{dmerge}(w,w',d) \text{ for some } d \in \{0,1\}^\omega$$

and w" is a fair merge if d can be chosen with infinitely many of both 0s and 1s, i.e. if $d \in (0^*11^*0)^\omega$

Given extended languages $A,B \subseteq \Sigma^\dagger$, their fair merge is

$$A||B = \{w'' \mid w'' \text{ fair merges some } w \in A,\ w' \in B\}$$

Theorem 6.1: For each M-automaton $\mathbb{M} = \langle S,s_0,M,F,\mathbb{D}\rangle$ over Σ, and each $\mathbb{M}' = \langle S',s_0',M',F',\mathbb{D}'\rangle$ over Σ', there is an M-automaton $(\mathbb{M}||\mathbb{M}')$ over $\Sigma \cup \Sigma'$ with

$$T(\mathbb{M}||\mathbb{M}') = T(\mathbb{M})||T(\mathbb{M}')$$

Proof: $\mathbb{M},\mathbb{M}'$ must first be transformed so that there are

(1) No self-loops; $s \notin M(s,\sigma)$, any $s \in S$, $\sigma \in \Sigma \cup \{\lambda\}$
[If there is such an s, adjoin a new s' to S, with $M(s',\sigma) = M(s,\sigma)$; then substitute s' for s in $M(s,\sigma)$; add s' to F if $s \in F$; and $D \cup \{s'\}$ to $\mathbb{D}$ whenever $s \in D \in \mathbb{D}$]

(2) No singleton accepting sets $\{s\} \in \mathbb{D}$
[Given (1), no such set can be $\text{In}(\pi)$; so it may be removed from $\mathbb{D}$]

Now, assume $\mathbb{M}, \mathbb{M}'$ have properties (1) - (2) above. Define $\mathbb{M}||\mathbb{M}' = \langle S'',s_0'',M'',F'',\mathbb{D}''\rangle$ as follows:

$$S'' = S \times S'$$
$$s_0'' = \langle s_0,s_0'\rangle$$
$$M''(\langle s,s'\rangle,\sigma) = \{\langle \hat{s},s'\rangle \mid \hat{s} \in M(s,\sigma),\ \sigma \in \Sigma \cup \{\lambda\}\}$$
$$\cup\ \{\langle s,\hat{s}'\rangle \mid \hat{s}' \in M(s',\sigma),\ \sigma \in \Sigma' \cup \{\lambda\}\}$$
$$F'' = F \times F'$$
$$\mathbb{D}'' = \{D \times \{s'\} \mid D \in \mathbb{D},\ s' \in F'\}$$
$$\cup\ \{\{s\} \times D' \mid D' \in \mathbb{D}',\ s \in F$$
$$\cup\ \{X \mid X \subseteq S \times S',\ \pi_1(X) \in \mathbb{D},\ \pi_2(X) \in \mathbb{D}'\}$$

where π_1, π_2 are the projection functions on pairs. By induction on the length of w", we have, for finite w"

$$\pi'' \in \text{paths}(\langle s,s'\rangle,w'') \Leftrightarrow \pi'' \text{ merges } \pi,\ \pi';\ w'' \text{ merges } w,\ w';$$
$$\text{with } \pi \in \text{paths}(s,w), \pi' \in \text{paths}(s',w')$$

The equivalence follows trivially from this, the only complications arising in showing

$$w'' \in \inf(\mathcal{M}||\mathcal{M}') \Rightarrow w'' \in T(\mathcal{M})||T(\mathcal{M}')$$

The reasoning is as follows: under the hypothesis

$$\text{In}(\pi'') \in \mathcal{D}'', \text{ some } \pi'' \in \text{paths}(s_0'',w'').$$

If $\pi_1(\text{In}(\pi''))$ = {s} is a singleton, then $\{s\} \notin \mathcal{D}$ from (2), so $s \in F$; and after some finite initial segment of π'', each state must have first component s; so π'' involves no $\mathcal{M}$ moves after this point, from (1). So π'' merges π,π' with π finite and $\text{In}(\pi') \in \mathcal{D}'$. This means the corresponding w,w' are accepted, and $w'' \in T(\mathcal{M})||T(\mathcal{M}')$.

Similarly, if $\pi_2(\text{In}(\pi''))$ is a singleton. Finally, if $\pi_1(\text{In}(\pi'')) \in \mathcal{D}$, $\pi_2(\text{In}(\pi'')) \in \mathcal{D}'$, then neither set is a singleton, from (2), and this is only possible if π'' fairmerges appropriate π,π', since π'' makes an infinite number of transitions of both sorts.

Corollary: If A,B are ω-regular. so is their fair merge A||B.
The concurrency operator is easily proved commutative and associative using obvious isomorphisms between state sets.
[For associativity, note that $\mathcal{M}||\mathcal{M}'$ inherits properties (1) - (2)]

7. Simulations on Automata:

In [11], the problem of fairness is viewed in the context of a "fixpoint" approach; the novelty lies in introducing expressions with both sorts of fixpoint operator in them. The novelty raises the question how known proof principles can be brought to bear. In fact, technical difficulties arise just with the maximal operator, in the context described here. Although the Scott induction principle dualises, its scope is considerably more limited - since combinations which are continuous in the conventional sense may cease to be so in the dual sense (when the lattice is "turned upside down"). This was seen in Section 4, with the example

$$F(X) = (2(0+1)^* + 0)\ X$$

showing that concatenation between extended languages is not cocontinuous - so, without special justification, forms involving concatenation cannot appear in the hypotheses for dualised Scott induction.

This provides extra motivation to be interested in proof principles for automata such as those involved here - even though their utility for the purposes of (operational) semantics of programs is obviously limited.

The sort of rule to be discussed can be seen to develop from the known decision procedures for problems concerning these automata. But in the form given here, they

are best related to notions of "weak homomorphism"[6] or "simulation"[10].

Firstly, we give definitions for the notion applied to finite automata $M = \langle S, s_o, M, F\rangle$, $M' = \langle S', s_o', M', F'\rangle$.

Say that M <u>simulates</u> M' via R - in symbols $M \underset{R}{\sim\!>} M'$, or just $M \sim\!> M'$ - if $R \subseteq S \times S'$, and, writing $s \sim\!> s'$ for $\langle s,s'\rangle \in R$,

1) $s_o \sim\!> s_o'$

2) $s \in F,\ s \sim\!> s' \Rightarrow s' \in F$

3) $\sigma \in \Sigma \cup \{\lambda\},\ s_1 \sim\!> s_1',\ s_2 \in M(s_1,\sigma)$

$\Rightarrow s_2 \sim\!> s_2'$ for some $s_2' \in M'(s_1',\sigma)$

Say that M <u>bisimulates</u> M' via R (in symbols $M \underset{R}{<\!\sim\!>} M'$, etc.) if M simulates M' via R, and M' simulates M via $R = \{\langle s',s\rangle \mid \langle s,s'\rangle \in R\}$.

<u>Theorem 7.1</u>: For finite automata M, M'

a) If $M \sim\!> M'$ then $T(M) \subseteq T(M')$

b) If M' is deterministic, then $T(M) \subseteq T(M')$ iff $M \sim\!> M'$

c) If M, M' are deterministic, then $T(M) = T(M)$ iff $M <\!\sim\!> M'$

<u>Proof</u>: Let T(M,s) denote the tapes accepted by M with start state changed to s.

a) prove, by induction on length of tapes w that
$s \sim\!> s'$, $w \in T(M,s) \Rightarrow w \in T(M',s')$
$T(M) \subseteq T(M')$ follows, putting $s = s_o$, $s' = s_o'$.

b) one direction follows from (a).
Suppose $T(M) \subseteq T(M')$; define $\underset{R}{\sim\!>}$ by

$s \underset{R}{\sim\!>} s'$ iff $s \in \overline{M}(s_o,w)$, $s' \in \overline{M}'(s_o',w)$ for some w
and $T(M, s) \subseteq T(M', s')$

c) define $\underset{R}{<\!\sim\!>}$ by

$s \underset{R}{<\!\sim\!>} s'$ iff $s \in \overline{M}(s_o, w)$, $s' \in \overline{M}'(s_o', w)$ for some w
and $T(M, s) = T(M', s')$.

For simulations on M-automata, we need the following definition:

Call $X \subseteq S$ <u>accessible</u> (via $s_1 s_2 \ldots\ldots s_n$) if there exist w, $w' \in \Sigma^*$ with $s_1 \in \overline{M}(s_o, w)$, $s_1 s_2 \ldots\ldots s_n s_1 \in \text{paths}(s_1, w')$, and $X = \{s_1, s_2, \ldots\ldots s_n\}$.

So X is accessible iff X consists of accessible states, and is generated by some cycle, and iff $X = \text{In}(\pi)$ for some infinite path π. Accessibility is decidable; since if X is accessible, it is accessible via a path of length $< k^2$, where k is the size of X. [The k^2 bound follows from an elementary argument; suppose $X = \{x_1, x_2, \ldots\ldots x_k\}$; for each i there is a path from x_i to x_{i+1} of length $\leq k$ -- the bound can be reduced by more careful analysis, but is still $O(k^2)$.]

For M-automata M, M' define $_R$ to be a simulation relation by adding a fourth condition to (1) - (3) above:

4) Let R be an automaton with state set the given relation $R = \{\langle s, s'\rangle \mid s \rightsquigarrow s'\}$, with start state $\langle s_o, s'_o\rangle$, and with transitions for each $s_1 \rightsquigarrow s'_1$, and each $\sigma \in \Sigma \cup \{\lambda\}$
$M_R(\langle s_1, s'_1\rangle, \sigma) = \{\langle s_2, s'_2\rangle \mid s_2 \rightsquigarrow s'_2, s_2 \in M(s_1, \sigma),$ and $s'_2 \in M'(s'_1, \sigma)\}$
Then if $X \subseteq R$ is accessible, and $\pi_1(X) \in \mathbb{M}$, then $\pi_2(X) \in \mathbb{M}'$.

[The reader might care to check some equivalences in which (4) plays a non-trivial role. Try the obvious automata corresponding to the following expressions:

$$(0^*11^*0)^\omega \qquad ((00^*1) + (11^*0))^\omega \qquad (0^\omega \,||\, 1^\omega) \qquad]$$

Theorem 7.2: (a) - (c) of 7.1 hold for M-automata M, M'.

Proof: So far as finite sequences are concerned, all properties are clear from 7.1. Otherwise:

a) Suppose $w \in \inf(M)$; then $In(\pi) \in \mathbb{D}$, for some $\pi \in paths(s_o, w)$. Suppose $\pi = s_o s_1 s_2 \ldots\ldots$; From (1), (3) there exists $\pi' \in paths(s'_o, w)$, with $s_i \rightsquigarrow s'_i$ for all i -- by induction on i. So $\pi'' = \langle s_o, s'_o\rangle\langle s_1, s'_1\rangle\langle s_2, s'_2\rangle\ldots.$ is an infinite path in R. Choosing a suitably long and late finite segment of π'' which cycles in R, we must get an accessible X with $\pi_1(X) = In(\pi)$, $\pi_2(X) = In(\pi')$; and $w' \in \inf(M')$ then follows from (4).

b) Define $\tilde{R}>$ as in 7.1(b). Let X be any accessible subset of the resulting R ; choose $\langle s, s'\rangle \in X$, and a path π which cycles through X, $\pi \in paths(\langle s, s'\rangle, w)$; if $\pi_1(X) \in \mathbb{D}$, then $w^\omega \in T(M, s)$, so $w^\omega \in T(M', s')$, since $s \rightsquigarrow s'$; so $\pi_2(X) \in \mathbb{D}$.

c) is clear from (a) and (b), defining $<\tilde{R}>$ as in 7.1(c).

Note: for deterministic M-automata, the decision method which results from 7.2 is closely related to the classical decidability results dependent on closure under the boolean operations on languages. The latter suggest the construction of an automaton M'', such that

$$T(M'') = \overline{T(M)} \cup T(M')$$

Then $T(M) \subseteq T(M')$ iff $T(M'') = \Sigma^\dagger$. The automaton R needed for (4) is just M'' restricted to accessible states. Given (1), (3); (2) holds iff each accessible state accepts (in the finite sense); and (4) holds iff $\inf(M'') = \Sigma^\omega$.

For nondeterministic automata, the method is not always applicable, even for finite automata. For example, consider the automata M, M' with state diagrams below:

(B, D are the accept states.) $M \rightsquigarrow M'$ sending A $\rightsquigarrow$ C, A $\rightsquigarrow$ D, B $\rightsquigarrow$ D ; but there is no simulation $M' \rightsquigarrow M$.

In general, therefore, translation to deterministic automata may be essential to establish equivalence etc.. But actually, in the context of programming constructs, nondeterminism is usually controllable -- for example, by adopting a regime in which "control characters" in path expressions act as terminators for particular component automata in concatenations and iterations. We might hope because of such devices for a system where only deterministic recognizing automata for path expressions need be considered. But in the presence of concurrency such hopes are unrealistic. We must check that concurrency is "well-behaved" in a more subtle sense.

Lemma 7.3: If X is accessible in $M || M'$ then, for i = 1, 2, either $\pi_i(X)$ is accessible, or $\pi_i(X)$ is a singleton.

Theorem 7.4: If $M_1 \tilde{R}> M_2$, $M_1' \tilde{S}> M_2'$ then $M_1 || M_1' \tilde{T}> M_2 || M_2'$ for some T.

Proof: Note first that the construction in (1) of 6.1 does not spoil simulations. If s' is introduced to remove a self-loop of state s, then setting s' $\rightsquigarrow$ s" whenever s $\rightsquigarrow$ s" (or s" $\rightsquigarrow$ s' whenever s $\rightsquigarrow$ s") does not affect the simulation. So we may assume none of the automata involved has self-loops. Now define T by

$$\langle s_1, s_1' \rangle \tilde{T}> \langle s_2, s_2' \rangle \quad \text{iff} \quad s_1 \rightsquigarrow s_2 \text{ and } s_1' \rightsquigarrow s_2'$$

Let R, S, T be the automata which correspond to R, S, T. T is isomorphic to $R || S$, by the map which exchanges 2nd and 3rd elements of the quadruples for T. So if X is any accessible set of T, Lemma 7.3 can be applied to project it into accessible or singleton sets in R and S. The result then follows from the simulation conditions on R and S , and the definition of the concurrency operator.

Simulations can also be shown to be well-behaved with respect to suitable choices corresponding to the regular operators of Section 3. (Using ε-moves, these constructions can be made straightforward. The construction corresponding to concatenation, for example, can be based on ε-moves between accept and start states.)

8. Unresolved problems.

1) Transitivity of simulations is not preserved on passing from finite automata to omega-automata. Consider the following M_1, M_2, M_3:

$D_1 = \{\{A\}\}$ $D_2 = \{\{C\}\}$ $D_3 = \{\{E\}\}$

There are simulations $M_1 \rightsquigarrow M_2$ and $M_2 \rightsquigarrow M_3$, but no simulation $M_1 \rightsquigarrow M_3$. Is there a modification of (4) to cure this ?

2) Identities between standard regular expressions can be elegantly provable by setting up suitable bisimulations, using the rules for forming derivatives of regular expressions [1]to verify simulation conditions (related to Ginzburg's procedure [5].) One looks for a similar approach to identities between ω-regular expressions, derivative rules for which are straightforward. But the relationship between derivative structure and deterministic recognizer structure is not analogous -- one cannot necessarily identify equivalent states of M-automata. Consider

$D = \{\{A, B\}\}$

States A,B are equivalent, but cannot be identified without accepting all infinite sequences. Perhaps a reformulation of the M-automaton notion would remove this awkwardness? One idea is to talk in terms of transitions taken infinitely often.

3) The idea of (2) is related to a question open at the time of writing. Define $A^+ = AA^*$. Is it always the case that, if $S = (BC)^\omega = (CB)^\omega$, then $S = (B^+C^+)^\omega$?

Acknowledgements:

The ideas of Section 4 arose from collaboration with Jerzy Tiuryn. I have also benefited much from discussion with Michael Paterson, Robin Milner, Gordon Plotkin and Sven Skyum.

References:

1. Brzozowski, J.A. Derivatives of regular expressions. J. ACM 11, 481-494(1964)

2. Büchi, J.R. Weak second order arithmetic and finite automata. Z. Math. Logik u Grund. Math. 6, 66-92 (1960).

3. Büchi, J.R.On a decision method in restricted second order arithmetic. Proc. Int. Congr. Logic, Method., Phil. of Sci. 1960. Stanford U. Press, 1962.

4. Choueka, Y. Theories of automata on ω-tapes: a simplified approach. J. Comp. Sys. Sci. 8, 117-141 (1974).

5. Ginzburg, A. A procedure for checking equality of regular expressions. J. ACM 14, 355-362 (1967).

6. Ginzburg, A. "Algebraic Theory of Automata." Academic Press 1968.

7. Landweber, L. Decision problems for omega-automata. Math. Sys. Thy. 3 376-384 (1969).

8. McNaughton, R. Testing and generating infinite sequences by a finite automaton. Inf. & Control 9, 521-530 (1966).

9. Muller, D.E. Infinite sequences and finite machines. Proc. 4th Ann. Symp. on Switching Circuit Theory and Logical Design, 3-16. IEEE 1963.

10. Milner, R. An algebraic definition of simulation between programs. Proc. 2nd Int. Joint. Conf. on Artificial Intelligence. British Comp. Soc. 1971.

11. Park, D. On the semantics of fair parallelism. 504-526 in "Abstract Software Specifications" (Proc. 1979 Copenhagen Winter School) ed. D. Bjørner. Springer Lec. Notes in Comp. Sci. 86. (1980).

12.Rabin,M. & Scott, D. Finite automata and their decision problems. IBM J of Res. & Dev. 3, 114-125 (1959).

13. Salomaa, A. Two complete axiom systems for the algebra of regular events. J. ACM 13, 158-169 (1966).