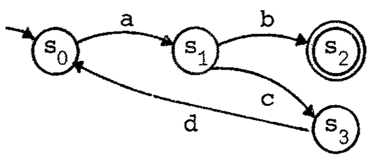
CHAPTER 1

Experimenting on nondeterministic machines

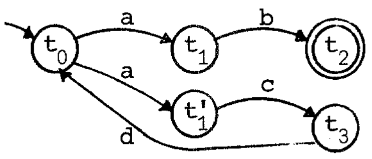
1.1 Traditional equivalence of finite state acceptors

Take a pair S,T of nondeterministic acceptors over the alphabet $\Sigma = \{a,b,c,d\}$:

S



T



The accepting states of S and T are s_2 and t_2 respectively; in non-deterministic acceptors we can always make do, as here, with a single 'dead' accepting state.

A standard argument that S and T are equivalent, meaning that they accept the same language (set of strings), runs as follows. Taking s_i (resp t_i) to represent the language accepted starting from state s_i (resp t_i), we get a set of equations for S , and for T :

$$\begin{array}{ll}
s_0 = as_1 & t_0 = at_1 + at'_1 \\
s_1 = bs_2 + cs_3 & t_1 = bt_2 \\
s_2 = \varepsilon & t'_1 = ct_3 \\
s_3 = ds_0 & t_2 = \varepsilon \\
 & t_3 = dt_0
\end{array}$$

Here as usual + stands for union of languages, ε for the language $\{\varepsilon\}$ containing only the empty string, and we can think of the symbol a standing for a function over languages: $as = a(s) = \{a\sigma;\ \sigma \in s\}$.

Now by simple substitution we deduce

$$s_0 = a(b\varepsilon + cds_0) \ .$$

By applying the distributive law $a(s + s') = as + as'$ we deduce

$$s_0 = ab\varepsilon + acds_0 \ ,$$

and we can go further, using a standard rule for solving such equations known as Arden's rule, to get

$$s_0 = (acd)^*ab\varepsilon \ .$$

For T it is even simpler; we get directly (without using distributivity)

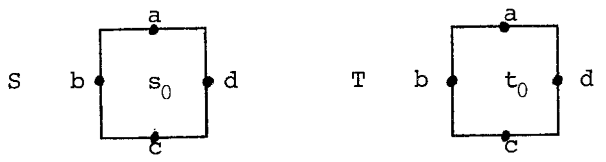
$$t_0 = ab\varepsilon + acdt_0$$

and the unique solvability of such equations tells us that $s_0 = t_0$, so S and T are equivalent acceptors.

But are they equivalent, in all useful senses?

1.2 Experimenting upon acceptors

Think differently about an acceptor over $\{a,b,c,d\}$. It is a black box, whose behaviour you want to investigate by asking it to accept symbols one at a time. So each box has four buttons, one for each symbol:



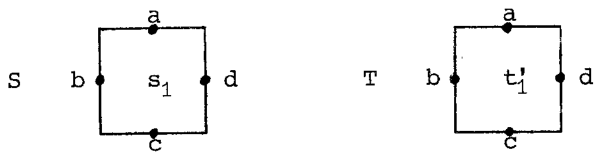
There are four atomic experiments you can do, one for each symbol. Doing an a-experiment on S (secretly in state s_0 , but you don't know that) consists in trying to press the a-button, with two possible outcomes in general:

(i) Failure - the button is locked;

(ii) Success - the button is unlocked, and goes down (and secretly a state transition occurs).

In fact we cannot distinguish between S and T , in their initial states, by any single atomic experiment; the a-experiment succeeds in each case, and the other three fail.

After a successful a-experiment on each machine, which may yield



we may try another atomic experiment, in our aim to see if the machines are equivalent or not. Clearly a b-experiment now succeeds for S and fails for T , though the other three experiments fail to distinguish them. After trying the b-experiment, then, can we conclude that S and T are <u>not</u> equivalent?

No, because S's response to the a-experiment could have been different (for all we know) and locked the b-button, while T's response could have been different (for all we know - and it could indeed!) and unlocked the b-button. Following this argument further, we may feel forced to admit that <u>no</u> finite amount of experiment could prove to us that S and T are, or are not, equivalent!

But suppose

(i) It is the <u>weather</u> at any moment which determines the choice of transition (in case of ambiguity, e.g. T at t_0 under an a-experiment) ;

(ii) The weather has only finitely many states - at least as far as choice-resolution is concerned ;

(iii) We can control the weather .

For some machines these assumptions are not so outrageous; for example, one of two pulses may always arrive first <u>within</u> a certain temperature range, and <u>outside</u> this range the other may always arrive first. (At the boundary of the range we have the well-known <u>glitch</u> problem, which we shall ignore here.)

Now, by conducting an a-experiment on S and T under <u>all</u> weather conditions (always in their start states, which we have to assume are recoverable), we can find that S's b-button is always unlocked, but that T's b-button is sometimes locked, and we can conclude that <u>the machines are not equivalent</u>.

Is this sense of equivalence, in which S and T are not equivalent, a meaningful one? We shall find that we can make it precise and shall adopt it - partly because it yields a nice theory, partly because it is a finer (smaller) equivalence relation than the standard one (which we can always recover by introducing the distributive law used in §1.1), but more for the following reason. Imagine that the b-buttons on S and T are hidden. Then <u>in all weathers</u> every successful experiment upon S unlocks some visible button:

<u>S (with b hidden) is not deadlockable</u>,

while in some weathers, and after some experiments, all of T's visible buttons will be locked:

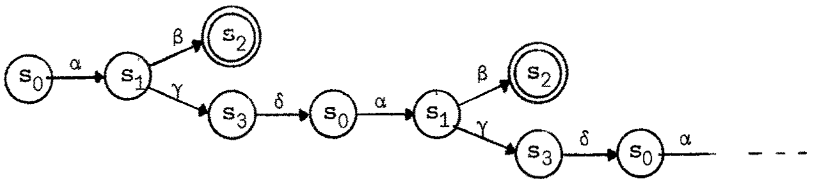
T (with b hidden) is deadlockable.

We wish to think of a nondeterministic choice in such machines as being resolved irreversibly, at a particular moment, by information flowing into the system from an unseen source; if a deadlock can thus arise in one machine but not in another, we do not regard them as behaviourally equivalent.

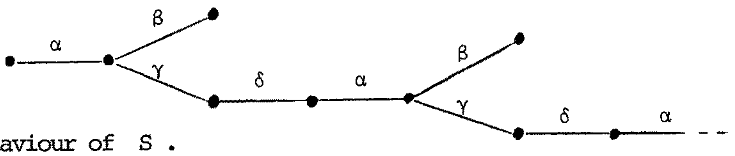
1.3 Behaviour as a tree

Because we reject the distributive law $a(x + y) = ax + ay$, we can no longer take languages (sets of strings) as the behaviours of our machines. We proceed to an alternative. From now on we will use NIL instead of ε to stand for a behaviour which can do nothing (= admits no experiment) ; we shall also use Greek letters for our symbols - i.e. names of buttons - so you should consider $\alpha,\beta,\gamma,\delta$ as replacements for a,b,c,d in our simple example.

First, take the transition graph for S and unfold it into a tree with states as node labels and symbols as arc labels:



Because state names are present we have lost no information; the state transition graph can be recovered from such a tree. But the experimenter cannot see the state - he can only see the transitions. This leads us to drop the node labels, and take the infinite tree



as the behaviour of S .

Definition A label is a member of a given (fixed) label set Λ .

We are using $\alpha,\beta,\gamma,..$ to stand for labels. (The use of the word 'label' in place of 'symbol' will be further motivated later.)

<u>Definition</u> A <u>sort</u> is a subset of Λ .

We shall usually use L,M,N,.. to stand for sorts. We shall also often use the word <u>agent</u> in place of 'machine' or 'acceptor', so

'S is an acceptor over the alphabet Σ'

becomes

'S is an agent of sort L' .

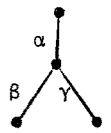
<u>Definition</u> A <u>Rigid Synchronization Tree</u> (RST) of sort L is a rooted, unordered, finitely branching tree each of whose arcs is labelled by a member of L .

Thus the tree in the last diagram is an RST of sort $\{\alpha, \beta, \gamma, \delta\}$. (It is also an RST of any larger sort.)

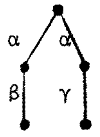
Why 'rigid'? Because it is the behaviour of a rigid agent - one which can make no transition except that corresponding to an atomic experiment. We shall soon meet other transitions.

Why 'synchronization'? Because we shall later see how the communication of two agents can be represented in forming their joint tree from their separate trees. Then the joint tree will <u>not</u> be rigid, in general, since intercommunication between component agents is not observable.

Notice that <u>finite</u> RSTs can be represented as expressions:

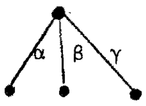


is $\alpha(\beta NIL + \gamma NIL)$



is $\alpha\beta NIL + \alpha\gamma NIL$

and usually there is more than one natural expression:



is $\alpha NIL + (\beta NIL + \gamma NIL)$, or
$(\alpha NIL + \beta NIL) + \gamma NIL$.

Indeed, + is both commutative and associative, since we declared RSTs to be unordered trees - and NIL is easily seen to be a zero for summation. To justify these remarks we now define the algebra of RSTs.

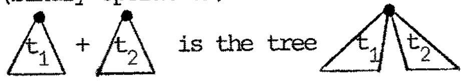
1.4 Algebra of RSTs

Ignoring sorts for a moment, we have an elementary algebra over RSTs, whose operations are:

NIL (nullary operation)

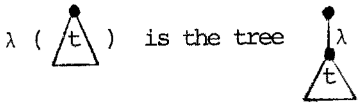
NIL is the tree ● ;

$+$ (binary operation)



(identify roots) ;

λ (unary operation, for each $\lambda \in \Lambda$)



.

They obey the following laws, as you can easily see:

Associativity	$x + (y + z) = (x + y) + z$
Commutativity	$x + y = y + x$
Nullity	$x + \mathrm{NIL} = x$

In fact, these laws are <u>complete</u>: any true equation between RST expressions can be deduced from them.

If we consider sorts, and let RST_L be the set of RSTs of sort L, then NIL is of sort L for any L:

$$\mathrm{NIL} \in \mathrm{RST}_L .$$

Further, $+$ takes trees of sort L,M respectively to a tree of sort $L\cup M$:

$$+ \in \mathrm{RST}_L \times \mathrm{RST}_M \to \mathrm{RST}_{L\cup M} ,$$

and λ takes a tree of sort L to a tree of sort $L\cup\{\lambda\}$:

$$\lambda \in \mathrm{RST}_L \to \mathrm{RST}_{L\cup\{\lambda\}} .$$

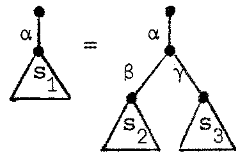
We shall usually forget about sorts for the present, but there are times later when they will be essential.

Consider now solving recursive equations over RSTs. We wish the equations for our agent S of §1.1

$$s_0 = \alpha s_1 \qquad s_1 = \beta s_2 + \gamma s_3$$
$$s_2 = \mathrm{NIL} \qquad s_3 = \delta s_0$$

to define the (infinite) behaviour of S as an RST of sort $\{\alpha,\beta,\gamma,\delta\}$.

This set of equations has a unique solution for the variables $s_0,..,s_3$; you can see this by the fact that the entire tree can be developed top-down to any depth:

s_0 = 

= ... and so on.

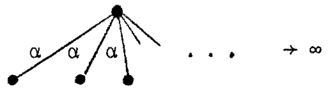
Warning. Not every set of recursive equations has a unique solution; consider the simple equation

$$s = s$$

which is satisfied by any RST (or anything else, for that matter). Again, some sets of equations define no RST at all. Consider the equation

$$s = s + \alpha\mathrm{NIL} \ ;$$

a solution would have to be infinitely branching at the root. Even if we allowed infinitely branching RSTs, so that

s_0 = 

would be a solution, it would not be unique since $s_0 + t$ would also be a solution for any t . We defer this problem to Chapter 5.

Exercise 1.1 Can you find a condition on a set of equations

$$\begin{aligned} s_0 &= \ \dots \\ s_1 &= \ \dots \\ &\dots \\ s_n &= \ \dots \end{aligned}$$

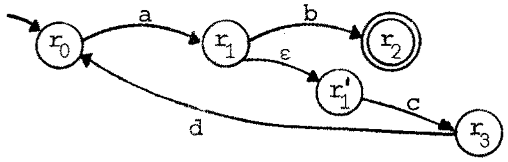
(with RST expressions involving $s_0,..,s_n$ on the right-hand sides)

which ensures that it possesses a unique solution in RSTs?
(Hint: consider cycles of ε-transitions in transition graphs.)

1.5 Unobservable actions

Under the conventional definition, a nondeterministic acceptor may have transitions labelled by ε , the null string. Consider R , a modification of our S of §1.1 (reverting briefly to Roman alphabet):

R



(The loop formed by the d-transition is irrelevant to our comparison.) In the conventional sense, R and S are equivalent. But what does the ε-transition mean, in our more mechanistic interpretation? It means that R in state r_1 (i.e. after the a-button has been pressed) may at any time move silently to state r_1', and that if a b-experiment is never attempted it will do so.

Thus, if we attempt a b-experiment on R, after the successful a-experiment, there are some weather conditions in which we find the b-button permanently locked; if on the other hand we attempt a c-experiment (after the a-experiment) we shall in all weather conditions find the c-button eventually unlocked (eventually, because although R may take a little time to decide on its ε-transition, it will do so since no b-experiment is attempted).

Exercise 1.2 Use this as the basis of an argument that no pair of R, S and T are equivalent. A rigorous basis for the argument will be given later.

Let us return to our Greek alphabet, and ask how we should write the equations specifying R's behaviour. We choose the symbol τ in place of ε (to avoid confusion with the null string), and use it as a new unary operation upon behaviours. The equations determining the behaviours $r_0,..,r_3$ are:

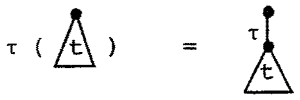
$$r_0 = \alpha r_1 \qquad r_1 = \beta r_2 + \tau r_1' \qquad r_1' = \gamma r_3$$
$$r_2 = \mathrm{NIL} \qquad r_3 = \delta r_0$$

We are assuming that $\tau \notin \Lambda$ (the fixed label set).

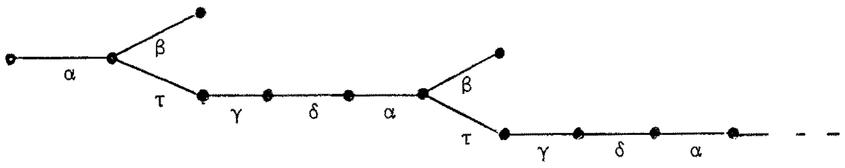
Definition A Synchronization Tree (ST) of sort L is a rooted, unordered, finitely branching tree each of whose arcs is labelled by a member of $L\cup\{\tau\}$.

Thus a rigid ST (an RST) is just an ST with no arcs labelled τ; it is the behaviour of an agent which can make no silent transitions.

Since we are taking the unary operation τ over STs to be given by

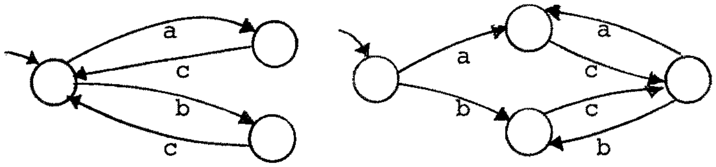


we can of course deduce the ST-behaviour of R. It is



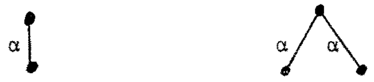
STs are a simple and useful notion of behaviour. They are just the unfoldings of behaviour equations, which in turn follow directly from transition graphs. Of course in this way different transition graphs can yield the same ST, from which we can be certain that they are indistinguishable by experiment.

Exercise 1.3 Convince yourself that the transition graphs



have the same unfolding.

However, different STs (or transition graphs yielding different STs) may be indistinguishable by experiment. This is true even for RSTs; consider the simple pair

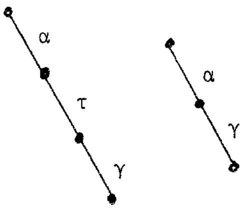


each of which admits a single α-experiment and then nothing else.

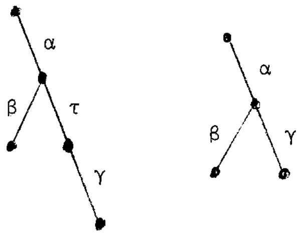
But it is even more true in the case of unobservable actions. Later we shall study an equivalence relation, observation equivalence, over STs, which can (for finite STs) be axiomatized by a finite set of equations added to those given in §1.4 above. To get a foretaste of the equivalence consider the following exercise.

Exercise 1.4 Examine each of the following pairs of simple STs and try to decide by informal argument, as in Exercise 1.2 above, which are observation equivalent (i.e. indistinguishable by experiment). You may reasonably conclude that four pairs are equivalent, or that six pairs are equivalent, but you should also find that the notion of equivalence is not yet precise. The point of this exercise is that it is not trivial to capture our informal arguments by a precise notion.

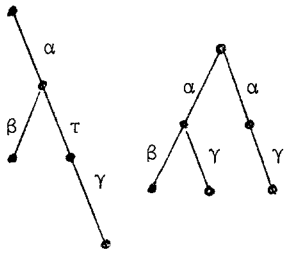
(i)



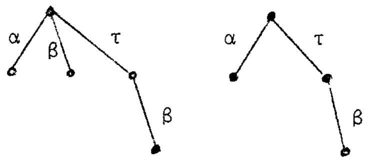
(ii)



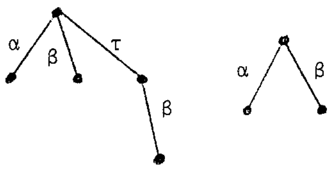
(iii)



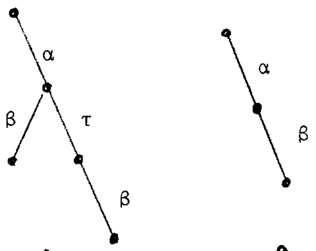
(iv)



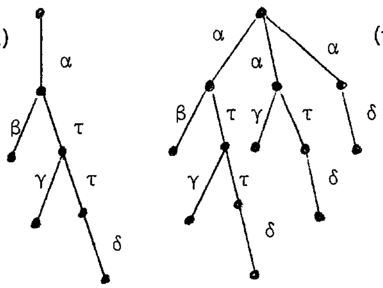
(v)



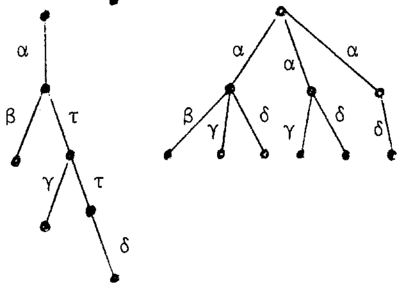
(vi)



(vii)



(viii)



Can you think of some equational axioms of observation equivalence?