# State machines

Peter Hancock

Preliminary version: 23 Nov 2000

These are working notes. The aim is to arrange and relate some notions connected with state machines and their specifications. The points of interest so far are:

1. that Mealy machines and Moore machines are complementary; they are 'servers' and 'clients' in a certain sense. This may be obvious, or at least well-known, but I haven't heard or seen it pointed out before.
2. that we can generalise both kinds of machine to dependent types and thereby shed some light on the notion of termination.
3. that a Moore machine generalised in this way is the same as a transition system 'localised' to a (dependent) family of sets.

## 1 Machines

Let $\langle C, R\rangle$ be a pair of sets. (In section 2 we generalise this to a dependent pair $\langle C, \{R(c) \,|\, c : C\}\rangle$.)

In the following the exponential notation $S^R$ may be regarded as a synonym for the function space notation $R \to S$. (For extra generality we may work in a category with binary products, a monad $(+1)$, and exponentials.)

- A *Mealy machine* with input $C$ and output $R$ consists of a set $S$ whose elements are called states, a function $m : S \to (R \times (1 + S))^C$, and an initial state $s_0 : S$. The function $m$ can be written in the form $\langle f, g\rangle$, where $f : S \to R^C$ and $g : S \to (1 + S)^C$. We may call $f$ the response function, and $g$ the side effect function.

  The use of $(+1)$ in the type of $f$ models the situation that after certain transitions, the system terminates, or ceases to engage in any further interaction, moving to the 'non-state' **inl**. It is 'pure bureaucracy'.

  When $S$ is finite it is convenient to draw the state-graph of a Mealy machine by taking the states to be nodes, and drawing for each $c : C$ an arrow from $s$, labelled by $\langle c, f(s, c)\rangle$ to the 'extended state' $g(s, c)$ . The output is 'on the arrows'.

- A *Moore machine* with output $C$ and input $R$ consists of a set $S$ whose elements are called states, a function $m : S \to C \times (1+S)^R$ (or equivalently a pair of functions $\rho : S \to C$, $\delta : S \to (1+S)^R$), and an initial state $s_0 : S$.

  The use of of (1+) in the type of $\delta$ models the situation that the machine may halt for certain inputs (have no next state). [Sometimes $C$ is the booleans, so $\rho$ is in effect a (decidable) subset of $S$. In that case, the machine is called a deterministic automaton. With dependent types we may consider $C = \{ack, accept\}$, $R(accept) = \{\,\}$, $R(ack) = V$. Such an automaton accepts a subset of $V^*$. The regular languages are the languages accepted by some finite state automaton.]

  When $S$ is finite it is convenient to draw the state-graph of the Moore machine by taking the nodes to be states, where $s$ is labelled with $\rho(s)$, and drawing for each $r : R$ an arrow from $s$, labelled by $r$, to the 'extended state' $\delta(s, r)$ . The output of the graph is 'on the nodes'.

We write $Me(C, R)$ for the class of Mealy machines with input $C$ and output $R$, and $Mo(C, R)$ for the class of Moore machines with output $C$ and input $R$. Note the asymmetry: for Mealy $C$ is input, whereas for Moore it is output. In some poetic sense, Mealy machines are *passive*. They wait for an input, and respond with an output as they move to their new state. On the other hand, Moore machines are *active*, in the sense that they (spontaneously, unsolicitedly) send an output (in $C$) and proceed to the next state only on a response. Moore machines are clients and Mealy machines are servers.

A Mealy machine is often called simply a *state machine*, *transducer* or *deterministic automaton*. Very often the state set $S$ is required to be finite. Here we don't need to impose any cardinality restriction as part of the definition – state machines of either kind may have an infinite state set. Sometimes there is additionally a special state, or a special subset of states to express notions of *termination*. (Instead of making such an addition, we represent halting by generalising the basic definition to dependent types, allowing that certain commands have no response.)

Another way to describe these machines (ignore halting) is that they are pointed coalgebras (coalgebras with a distinguished element) for the following endofunctors on the category of sets. (We leave the action on morphisms implicit.)

$$\begin{array}{rcl} F_{Me}(S) & = & (R \times S)^C \\ F_{Mo}(S) & = & C \times S^R \end{array}$$

To say that they are pointed coalgebras means that they can be represented as elements of (*i.e.* morphisms from a terminal object into) the final coalgebras for the respective functors.

We can picture machines as trees or forests of trees (which need not be wellfounded, but may have paths of infinite length.

The notation $\nu\_$ in the following stands for an operator which returns for each endofunction $F : \mathsf{Set} \to \mathsf{Set}$ which is sufficiently well-behaved a final coalgebra.

The question of exactly what constitutes good behaviour is difficult; however polynomial functors such as $Me$ and $Mo$ are certainly well-behaved.

- A Mealy machine is an element of $\nu F_{Me}$. It can be pictured as $C$-indexed forest of $C$-branching trees in which every node is labelled with an element of $R$.

- A Moore machine is an element of $\nu F_{Mo}$. It can be pictured as a $R$-branching tree in which every node is labelled with an element of $C$. Another way to think of these objects is as generalised powerseries, where the coefficients are elements of $C$, and the variables are elements of $R$. The 'constant term' is the label of the root, and for $r : R$, the partial derivative by $r$ consists of the powerseries whose coefficient at $w : R^*$ is the coefficient of the original series at $r : w$, or $r$ pushed on the front of $w$. If the coefficients have a semi-ring structure, then this lifts to the powerseries.

**Examples.** One important example of a Mealy machine is the finite automaton part of a Turing machine (see for example [2]). In this case, the input set $C$ consists of the symbols which may appear in the tape cells, while (according to a common convention) the output set $R = 2 + C$ represents the actions of moving the head left, moving the head right, or rewriting the cell currently under the head so that it holds a certain symbol from alphabet $C$. (There are many slight variations.)

One important example of a Moore machine is the tape (and transport part – it does not matter whether it moves the tape, or moves the machine) mechanism of a Turing machine. In this case, the input set is the set $R$ above, while the output set is the set $C$ above. The set of states of this device can be represented in various ways; one may use the pair of the stream of characters to the left of the head, and the stream of characters under or the right of the head. (An advantage of the final coalgebra is that we have an abstract state.)

(There is some Haskell code which could go here.)

## 2 Dependent versions

Here we start from a *family* $\langle C, \{R(c) \mid c : C\}\rangle$ of sets rather than a mere pair of sets. In this case, the functors are as follows:

$$\begin{array}{rcl} F_{Me}(S) & = & (\Pi\, c : C)(R(c) \times S) \\ F_{Mo}(S) & = & (\Sigma\, c : C)S^{R(c)} \end{array}$$

We call elements of the final coalgebras for these functors *generalised* machines.

It is allowed that for certain commands $c : C$ the set of responses $R(c)$ is empty. This sheds some light on the termination of machines. Termination is sometimes represented by adjoining to the basic coalgebra either a special *terminal* state, or a predicate of *terminal states*. In the dependent version, the commands $c : C$ for which $R(c)$ is empty are defined to be *terminal commands.*

**An example of a generalised Moore machine.** Martin-Löf's W-type $W(C,R)$ is the least solution $W$ of the equation $W = F_{Mo}(W)$, with typical element $\sqcup_c f$ where $c : C$ and $f : R(c) \to W$. Then the map $\phi : \sqcup_c f \mapsto \langle c, f \rangle$ provides us with a coalgebra $(W, \phi)$ for $F_{Mo}$, and so any element $t : W$ is the initial state of a generalised Moore machine. (As $W$ is the least fixed point of $F_{Mo}$, such a machine eventually terminates, in the sense that it eventually issues a command for which no response is possible – ie a command $c$ for which the set $R(c)$ is empty.)

**another example** $C$ is the set of natural numbers, $R(c)$ is the $c$-th canonical finite set.

# 3 Transition systems

A transition system is a triple $(A, B, .[.])$ where $A$ is a set of states, $B$ is a function which to $a : A$ assigns a set $B(a)$, and $.[.]$ is a binary function which to $a : A$ and $b : B(a)$ assigns $a[b] : A$.

$$\begin{array}{lcl} A & : & \mathsf{Set} \\ B & : & A \to \mathsf{Set} \\ .[.] & : & (\Pi\, a : A) B(a) \to A \end{array}$$

A *pointed* transition system is a transition system $(A, \ldots)$ togther with an element $a_0 : A$.

Note that a transition system is a coalgebra in the (super-large) category of types (large sets) for the (covariant) functor *Fam*, where $\mathit{Fam}(X) = (\Sigma\, I : \mathsf{Set}) X^I$. This is because a typical element of the type $A \to \mathit{Fam}(A)$ has the form

$$\lambda\, a. \langle B(a), \lambda\, b : B(a). a[b] \rangle$$

Given a transition system $(A, B, .[.])$, we can define two predicate transformers, from predicates over $A$ to predicates over $A$.

$$\begin{array}{lcl} P^{\Box} & = & \{ a \mid (\Pi\, b : B(a)) P(a[b]) \} \\ P^{\Diamond} & = & \{ a \mid (\Sigma\, b : B(a)) P(a[b]) \} \end{array}$$

The *accessible* elements of $A$ are those in the least prefixed point of $.^{\Box}$.

$$\mathit{Acc} = \mu P. P^{\Box} \subseteq P$$

That an element is accessible means that all (type-correct) chains of transitions eventually end in deadlock.

The *potentially infinite* elements of $A$ are those in the greatest postfixed point of $\_^{\Diamond}$.

$$\mathit{Inf} = \nu P. P \subseteq P^{\Diamond}$$

That an element (start state)is potentially infinite means that there is an infinite stream, in which successive entries belong to a certain set that depends on the finite prefix of the stream up to that entry, and ultimately on the start state.

These are rather subtle notions.

Observe that an element of $Acc(a)$ for some $a : A$ has the form $\sqcup f$, where $f(b : B(a)) : Acc(a[b])$. An element of $Inf(a)$ for some $a : A$ has the form $\langle b, t\rangle$ where $b : B(a)$ and $t : Inf(a[b])$;

**Localisation to a universe.** Let $\langle C, R\rangle$ be a universe (*i.e.* a set-indexed family of sets). Then we can define for each set $S$ a set $fam_{\langle C,R\rangle}(S)$.

$$fam_{\langle C,R\rangle}(S) = (\Sigma\, c : C)S^{R(c)}$$

The elements of this set are *internal families* of elements of $S$, in the sense that they represent the index sets of families via codes in the set $C$ and the decoding $R : C \to \mathsf{Set}$ of its elements as sets rather than by directly using the type $\mathsf{Set}$. Each internal family $\langle c, f\rangle$ represents the (external) family $\{f(r) \,|\, r : R(c)\}$ of elements of $S$.

Now we can define a localised version of a transition system; if a (global) transition system is a coalgebra for *Fam*, then a transition system local to a family of sets $\langle C, R\rangle$ is a coalgebra for $fam_{\langle C,R\rangle}$.

**Observation** . A $\langle C, R\rangle$-localised transition system (with initial state) is just the same thing as a generalised Moore machine with command set $C$, and response family $\{R(c) \,|\, c : C\}$. This machine issues as a command a code for a set, and then moves to a new state on receipt of an element of the set so coded.

(This might be regarded as a 'foundation' game. I give you a set; you choose an element.)

Say something about accessibility and termination of the Moore machine. Accessibility means inevitable termination when interacting with *any* Mealy machine.

It is natural now to look at coalgebras for the functor

$$S \mapsto (\Pi\, c : C)R(c) \times S$$

that is, at generalised Mealy machines.

Also, look at algebras as well as coalgebras.

# 4 Interaction systems

A interaction system is a quadruple $(A, B, C, .[./.])$ where $A$ is a set of states, $B$ is a function which assigns a set $B(a)$ to each element $a : A$, $C$ is a function which to $a : A$ and $b : B(a)$ assigns a set $C(a, b)$, and $.[./.]$ is a ternary function which to $a : A$ and $b : B(a)$ assigns $a[b/c] : A$.

$$\begin{array}{lcl} A & : & \mathsf{Set} \\ B & : & A \to \mathsf{Set} \\ C & : & (\Pi\, a : A)B(a) \to \mathsf{Set} \\ .[./.] & : & (\Pi\, a : A, b : B(a))C(a, b) \to A \end{array}$$

Note that a interaction system is a coalgebra in the category of types (large sets) for the functor $Fam^2$. This is because a typical element of the type $A \to Fam^2(A)$ has the form

$$\lambda\, a.\langle B(a), \lambda\, b.\langle C(a,b), \lambda\, c.a[b/c]\rangle\rangle$$

Given such a structure, we can define two predicate transformers.

$$\begin{array}{rcl} P^{\Box\Diamond} & = & \{a \mid (\Pi\, b : B(a))(\Sigma\, c : C(a,b))P(a[b/c])\} \\ P^{\Diamond\Box} & = & \{a \mid (\Sigma\, b : B(a))(\Pi\, c : C(a,b))P(a[b/c])\} \end{array}$$

The *barred*[1] elements of $A$ are those in the least prefixed point of $\_^{\Diamond\Box}$.

$$Bar = \mu P.P^{\Diamond\Box} \subseteq P$$

We may generalise the notion of barredness, to barred relative to a predicate $X$. Such a state is one from which there is a strategy which ensures that $X$ should hold *eventually*, if only the opponent follows a strategy which evades deadlock.

The *positive* elements of $A$ are those in the greatest postfixed point of $\_^{\Box\Diamond}$.

$$Pos = \nu P.P \subseteq P^{\Box\Diamond}$$

We may generalise the notion of positivity, to positivity relative to a predicate $X$. Such a state is one in which $X$ holds, and moreover there is a perfect strategy for responding to the player so that $X$ will continue to hold, now and evermore. Such a strategy evades deadlock.

Observe that

- an element of $Bar(a)$ for some $a : A$ has the form

  $$\sqcup_b f$$

  where $b : B(a)$ and $f : (\Pi\, c : C(a,b))Bar(a[b/c])$.

- an element of $Pos(a)$ for some $a : A$ has the form

  $$\lambda\, b : B(a).\langle \rho(b), \delta(b)\rangle$$

  where $\rho : (\Pi\, b : B(a))C(a,b)$ and $\delta : (\Pi\, b : B(a))Pos(a[b/\rho(c)])$.

One can think of the elements of $Bar(a)$ as client programs that can be used in state $a$ to induce server deadlock. That we have a least fixed point means we have a way of showing that certain relations are full. The relations are between states and client-side programs to induce deadlock in the server starting in that state. To show that a relation $X(a,t)$ between $a : A$ and $t : Bar(a)$ is full we have to show that $X$ satisfies an 'algebraic' closure property

$$\begin{array}{l} (\Pi\, a : A) \\ (\Pi\, t = \sqcup_b f : Bar(a)) \\ \quad ((\Pi\, c : C(a,b))X(a[b/c], f(c))) \to X(a,t) \end{array}$$

One can think of the elements of $Pos(a)$ as programs for the server, or Mealy machines to evade deadlock starting in state $a$.

The situation here does not some to be dual.

[1] I use some terminology suggested by Martin-Löf, from whom I first learnt about positivity.

**Localisation.** It is not clear how to localise the notion of interaction system. We have the option of changing universes.

## 5 Allegories

From the perspective of dependent type-theory, there are two notions of 'subset' of a set $A$, namely propositional function $P : \mathit{Pow}(A)$ and indexed family $(T, t) : \mathit{Fam}(A)$. There are therefore two[2] notions of 'binary relation' between sets $A$ and $B$, namely $A \to \mathit{Pow}(B)$ and $A \to \mathit{Fam}(B)$. The former admits a natural notion of converse ($P^{\sim}(a, b) = P(b, a)$) while the latter does not.

The notion of *allegory* arose as an analysis of the notion of relation, and a generalisation of relational calculus to a categorical setting ([3], [1, Chap. 4]). Briefly, an allegory is a category of a certain kind in which the hom-sets are equipped with a partial order and binary meets, together with a converse operator $(.)^{\sim} : \mathrm{hom}(A, B) \to \mathrm{hom}(B, A)$.

It is natural to ask how close we can get to a notion of converse in type theory starting from the weak notion of relation $A \to \mathit{Fam}(B)$. One idea is to consider a category in which the objects are sets and the morphisms in $\mathrm{hom}(A, B)$ consist of *pairs* $(\phi, \phi^*)$ where $\phi : A \to \mathit{Fam}(B)$ and $\phi^* : B \to \mathit{Fam}(A)$. This is reminiscent of a Conway game, except that Conway games have to be wellfounded. It should be noted that the notion of interaction system (which is in a sense asymmetrical) can be recast in the (more symmetric) form of such a pair. (The difference is that an interaction system has one starting point, while one of these 'games' has two. Now the question arises of how to define a partial order on such pairs.

## References


[1] R. Bird and O. de Moor. *Algebra of Programming.* Prentice Hall, 1997.

[2] M. Davis. *Computability and Unsolvability.* Dover, 1982. Originally published McGraw-Hill, 1958.

[3] P. Freyd and A. Ščedrov. *Categories, Allegories.* Number 39 in Mathematical Library. North Holland, 1990.


[2] At least. There is also the notion $\mathit{Fam}(A \times B)$; an object of that type is close to what is commonly called a 'span' between $A$ and $B$.