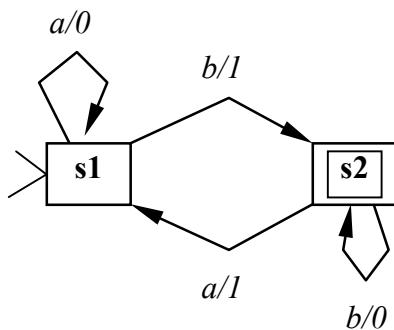


INTRODUCTION TO ALGEBRAIC SPECIFICATION NOTATION

A Finite State Machine is defined as $FSM=(S, I, O, IS, F, T)$, where

- S is the set of states
- I is the set of input symbols
- O is the set of output symbols
- IS is the initial state
- F is the set of final states
- T is the set of transitions ($T: I \times S \rightarrow O \times S$)

Example:



$S=\{s1,s2\}$
 $I=\{a,b\}$
 $O=\{0,1\}$
 $IS=s1$
 $FS=\{s2\}$

T	a	b
$s1$	$s1/0$	$s2/1$
$s2$	$s1/1$	$s2/0$

The algebraic notation is a compact way of writing down the transitions T of a FSM:

$$s1 = a.\bar{0}.s1 + b.\bar{1}.s2$$

$$s2 = a.\bar{1}.s1 + b.\bar{0}.s2$$

If someone does not wish to name some intermediate states (s)he could write:

$$S = b.a.\bar{1}.\bar{1}.S$$

ALGEBRAIC SPECIFICATIONS CAN BE PARAMETRIZED

A vending machine that WANTS 50 drs to deliver the Product, but also gives CHANGE if user inputs more than 50drs.

PAYMENT :

$$W50 = 10\text{drs}.W40 + 20\text{drs}.W30 + 50\text{drs}.\text{PAID}$$

$$W40 = 10\text{drs}.W30 + 20\text{drs}.W20 + 50\text{drs}.\text{O10}$$

$$W30 = 10\text{drs}.W20 + 20\text{drs}.W10 + 50\text{drs}.\text{O20}$$

$$W20 = 10\text{drs}.W10 + 20\text{drs}.\text{PAID} + 50\text{drs}.\text{O30}$$

$$W10 = 10\text{drs}.\text{PAID} + 10\text{drs}.\text{O10} + 50\text{drs}.\text{O40}$$

$$\text{PAID} = \overline{\text{Product}.\text{READY}}$$

$$\text{O40} = \overline{10\text{drs}.\text{O30}} + \overline{20\text{drs}.\text{O20}}$$

$$\text{O30} = \overline{10\text{drs}.\text{O20}} + \overline{20\text{drs}.\text{O10}}$$

$$\text{O20} = \overline{10\text{drs}.\text{O10}} + \overline{20\text{drs}.\text{PAID}}$$

$$\text{O10} = \overline{10\text{drs}.\text{PAID}}$$

Since there might be various types of Products which have various prices, one can have one parametrized version of the above:

PAYMENT(n, Product):

$$W_n = 10\text{drs}.W_{n-10} + 20\text{drs}.W_{n-20} + 50\text{drs}.W_{n-50} \quad \forall n > 0$$

$$W_n = O_{-n} \quad \forall n < 0$$

$$W_0 = \text{PAID}$$

$$\text{PAID} = \overline{\text{Product}.\text{READY}}$$

$$O_n = \overline{10\text{drs}.O_{n-10}} + \overline{20\text{drs}.O_{n-20}} \quad \forall n > 0$$

$$O_{-n} = \overline{10\text{drs}.\text{PAID}}$$

SYSTEMS AS COLLECTIONS OF OBJECTS

A systems consists of several objects $Obj_1, Obj_2, \dots, Obj_n$ each of one specified as a FSM which may run freely. We write:

$$\text{SYSTEM} = (Obj_1 | Obj_2 | \dots | Obj_n)$$

If the system was built as monolithic, then the possible states would have been k^n , where k the number of states in each object.

Objects could communicate and synchronise through handshakes, denoted as:

$$\text{SYSTEM} = (Obj_1 | Obj_2 | \dots | Obj_n) \setminus \{hs_1, hs_2, \dots, hs_m\}$$

Example 1 (MASTER/SLAVE):

Let M, S1 and S2 three processes

Version I: No synchronisation

$$M = \text{task.solution}_1.\text{solution}_2.M$$

$$S1 = \text{subtask}_1.S1$$

$$S2 = \text{subtask}_2.S2$$

$$\text{SYSTEM} = (M | S1 | S2)$$

Let M be the master and S1 and S2 be the two slave processes:

Version II: Waking up slaves in order

$$M = \text{task}.\overline{hs_1}.\overline{hs_2}.\text{solution}_1.\text{solution}_2.M$$

$$S1 = hs_1.\text{subtask}_1.S1$$

$$S2 = hs_2.\text{subtask}_2.S2$$

$$\text{SYSTEM} = (M | S1 | S2) \setminus \{hs_1, hs_2\}$$

Version III: Waking up slaves in arbitrary order

$$M = \text{task}.\overline{hs}.\overline{hs}.\text{solution}_1.\text{solution}_2.M$$

$$S1 = hs.\text{subtask}_1.S1$$

$$S2 = hs.\text{subtask}_2.S2$$

$$\text{SYSTEM} = (M | S1 | S2) \setminus \{hs\}$$

Version IV: Waking up slaves in arbitrary order and notify master when finished

$$M = \text{task}.\overline{\text{hs}}.\overline{\text{hs}}.\text{sh}.\text{sh}.\text{solution}_1.\text{solution}_2.M$$
$$S1 = \text{hs}.\text{subtask}_1.\overline{\text{sh}}.S1$$
$$S2 = \text{hs}.\text{subtask}_2.\overline{\text{sh}}.S2$$
$$\text{SYSTEM} = (M \mid S1 \mid S2) \setminus \{\text{hs}, \text{sh}\}$$

Alternatively (with different meaning though):

$$M = \text{task}.\overline{\text{hs}}.\overline{\text{hs}}.\text{sh}.\text{solution}_1.\text{sh}.\text{solution}_2.M$$
$$S1 = \text{hs}.\text{subtask}_1.\overline{\text{sh}}.S1$$
$$S2 = \text{hs}.\text{subtask}_2.\overline{\text{sh}}.S2$$
$$\text{SYSTEM} = (M \mid S1 \mid S2) \setminus \{\text{hs}, \text{sh}\}$$

Note: Solution_n should be interpreted as the n^{th} solution coming in and not as the solution of the n^{th} subsystem.

Example 2 (PRODUCER/CONSUMER):

Let P1, P2 and P3 three processes:

Version I: No synchronisation

$$P1 = \text{task1}.P1$$
$$P2 = \text{task2}.P2$$
$$P3 = \text{task3}.P3$$
$$\text{SYSTEM} = (P1 \mid P2 \mid P3)$$

Let P1, P2 and P3 be working in a pipeline

Version II: Pipelining

$$P1 = \text{task1}.\overline{\text{hs1}}.P1$$
$$P2 = \text{hs1}.\text{task2}.\overline{\text{hs2}}.P2$$
$$P3 = \text{hs2}.\text{task3}.P3$$
$$\text{SYSTEM} = (P1 \mid P2 \mid P3) \setminus \{\text{hs1}, \text{hs2}\}$$