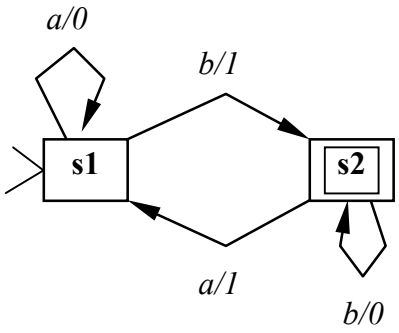
INTRODUCTION TO ALGEBRAIC SPECIFICATION NOTATION

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×S→O×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:

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

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

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

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

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 :

$W50 = 10drs.W40 + 20drs.W30 + 50drs.PAID$
$W40 = 10drs.W30 + 20drs.W20 + 50drs.O10$
$W30 = 10drs.W20 + 20drs.W10 + 50drs.O20$
$W20 = 10drs.W10 + 20drs.PAID + 50drs.O30$
$W10 = 10drs.PAID + 10drs.O10 + 50drs.O40$

$PAID = \overline{Product}.READY$

$O40 = \overline{10drs}.O30 + \overline{20drs}.O20$
$O30 = \overline{10drs}.O20 + \overline{20drs}.O10$
$O20 = \overline{10drs}.O10 + \overline{20drs}.PAID$
$O10 = \overline{10drs}.PAID$

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

PAYMENT(n, Product):

$W_n = 10drs.W_{n-10} + 20drs.W_{n-20} + 50drs.W_{n-50}$ $\forall n>0$
$W_n = O_{-n}$ $\forall n<0$
$W_0 = PAID$

$PAID = \overline{Product}.READY$

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

SYSTEMS AS COLLECTIONS OF OBJECTS

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

$$SYSTEM = (Obj_1 \mid Obj_2 \mid \ldots. \mid Obj_n)$$

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:

$$SYSTEM = (Obj_1 \mid Obj_2 \mid \ldots. \mid Obj_n) \setminus \{hs_1, hs_2, \ldots, hs_m\}$$

Example 1 (MASTER/SLAVE):

Let M, S1 and S2 three processes

Version I: No synchronisation

$M = task.solution_1.solution_2.M$
$S1 = subtask_1.S1$
$S2 = subtask_2.S2$

$SYSTEM = (M \mid S1 \mid S2)$

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

Version II: Waking up slaves in order

$M = task.\overline{hs_1}.\overline{hs_2}.solution_1.solution_2.M$
$S1 = hs_1.subtask_1.S1$
$S2 = hs_2.subtask_2.S2$

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

Version III: Waking up slaves in arbitrary order

$M = task.\overline{hs}.\overline{hs}\ .\ solution_1.solution_2.M$
$S1 = hs.subtask_1.S1$
$S2 = hs.subtask_2.S2$

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

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

$M = task.\ \overline{hs}.\ \overline{hs}\ .\ sh.\ sh.\ solution_1.solution_2.M$
$S1 = hs.subtask_1.\ \overline{sh}.\ S1$
$S2 = hs.subtask_2.\ \overline{sh}.\ S2$

$SYSTEM = (M \mid S1 \mid S2) \setminus \{hs, sh\}$

Alternatively (with different meaning though):

$M = task.\ \overline{hs}.\ \overline{hs}\ .\ sh.\ solution_1.\ sh.\ solution_2.M$
$S1 = hs.subtask_1.\ \overline{sh}.\ S1$
$S2 = hs.subtask_2.\ \overline{sh}.\ S2$

$SYSTEM = (M \mid S1 \mid S2) \setminus \{hs, sh\}$

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

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

Let P1, P2 and P3 be working in a pineline

Version II: Pipelining

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