CS E6831 04f S. H. 10/3/04

NOTES ON MACHINE STRUCTURE AND STATE ASSIGNMENTS. Part 2

FINDING CLOSED PARTITIONS

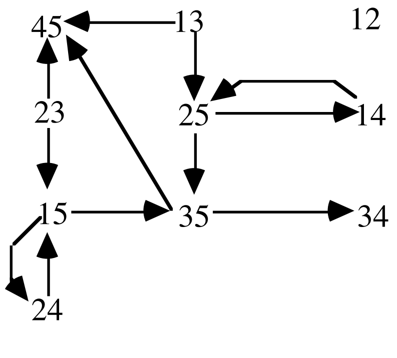
Use as a running example, the table below:

	A	B	C
1	2	5	1
2	1	5	1
3	5	4	1
4	5	3	1
5	4	3	1

Definition: pab is the partition equating only a and b. Example: p13=(13, 2, 4, 5)

Definition: mab is the smallest closed partition that equates a and b , i.e., mab is closed, includes a block containing ab, and is smaller than any other partition with these properties. Such partitions are called *basic* partitions. Example: m35=(1, 2, 345). This was obtained by noting that, if the partition equates 3 and 5 (we will use notation 3~5 for this), then the next states for 3 and 5 under any input must also be equated, and looking at the flow table, we see that this implies 3~4. So, by transitivity, we have 3, 4, and 5 in the same block. There are no further implications.

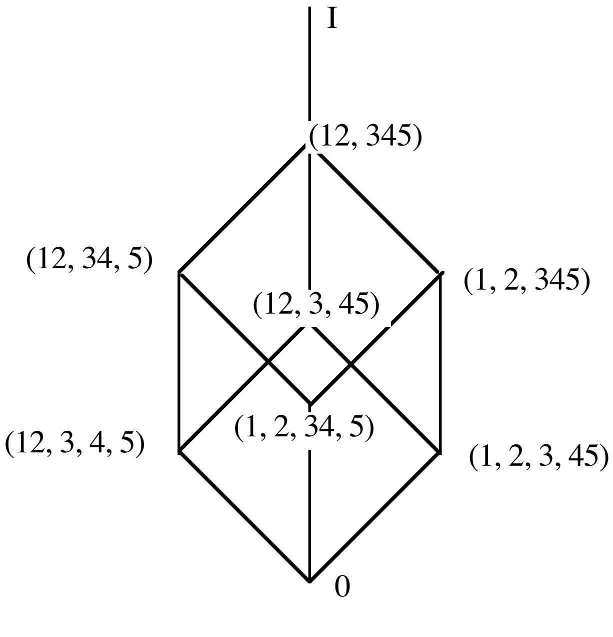
If partitions p and q are both closed, then partition p+q, is also closed (as is pq). All closed partitions can be found as sums of one or more basic partitions. So a good starting point for finding the set of all closed partitions is to find all the basic partitions. A good way to do this is to generate a directed graph showing all the pairwise implications. This graph for the above table is shown below. Recall that I is the trivial partition which equates all states (puts them all in the same block). Also, the implication graph has been simplified by the deletion of redundant arcs. That is, if there are arcs from a to b and from b to c, we can delete an arc from a to c because of transitivity. In the example, we deleted, among other arcs, the arc from 15 to 35, because the existing arcs from 15 to 25 and from 25 to 35 made it redundant.



Now, using the information in the implication graph, we generate the mij's as below:
m12=(12, 3, 4, 5)
m13=m14=m15=m23=m24=m25=I
m34=(1, 2, 34, 5)
m35=(1, 2, 345)
m45=(1, 2, 3, 45)

m12+ m34 gives us (12, 34, 5)
m12+ m35 gives us (12, 345)
m12+ m45 gives us (12, 3, 45)

No other nontrivial closed partitions can be produced. We can arrange the pp's in lattice form as below. (A lattice is a structure where a partial ordering exists among the elements and, for every pair of elements, there is a unique glb and a unique lub.) From the lattice, we can easily find the product or sum of any pair of the partitions in it. For example, to find the product of (12, 34, 5) and (1, 2, 345), find the first intersection of downward lines from their positions in the lattice. This occurs at (1, 2, 34, 5), which is the product, or glb of those partitions. We can find the sum (lub) of any pair by finding the first intersection of upward lines from the pair. We can, for example find in this manner that (12, 3, 4, 5) + (1, 2, 345) = (12, 345)



With the aid of the lattice, we can see that a valid state assignment could be constructed using (12, 34, 5) and (1, 2, 3, 45). But, it would appear, at least at first, that we would have to use 4 state variables, since we need 2 for each of these partitions. However, a look at the lattice tells us that both have a common factor, since there are downward paths from (12, 345) to each of them. Unfortunately, there are no closed 2-block partitions that can serve as cofactors. We can, however, easily find non-closed 2-block partitions that can do the job. We are looking for 2-block partitions p and q such that

(12, 345)p = (12, 34, 5), and (12, 345)q = (1, 2, 3, 45). We could formally specify the requirements as p = (12, 34, 5)/ (12, 345) and q = (1, 2, 3, 45). / (12, 345). It is not hard to find solutions p = (125, 34) and q = (13, 245). There are other solutions. Thus, a good state assignment would consist of y1 corresponding to (12, 345), y2 corresponding to (125, 34), and y3 corresponding to (13, 245). Generating the logic would verify that Y1 is independent of y2 and y3, that Y2 is independent of y3, and that Y3 is independent of y2.

PARTITION PAIRS (pp's)
An ordered pair of partitions [p1, p2] is a partition pair (pp) if, given the input and the block of p1 containing the present state, the block of p2 containing the next state is determined.

Example 1: The machine below has a pp [(125,34), (15, 234)]. For example, if we are told that X=0 and that the present state is in the block 125, then we know that the next state is in the block 234. So, if we use the given state assignment, Y2 will be a function only of X and y1. In fact, Y2=y1'+ X.

Another pp for this table is [(14, 2, 3, 50, (123, 4, 5)].

	X			
	0	1	y1	y2
1	3	2	0	0
2	4	3	0	1
3	5	2	1	1
4	1	3	1	1
5	3	4	0	0

Example 2: [(124, 35), (135, 24)] is a pp for the table below. So is
[(13, 2, 4, 5), (134, 25)]

	AB			
	00	01	11	10
1	3	1	4	2
2	5	3	2	4
3	3	4	3	5
4	5	1	4	2
5	5	4	3	5

PROPERTIES OF PARTITION PAIRS (pp's)
Suppose t, t', and u are all partitions on the set of states for some flow table.

Recall that t>u means that every block of u is included in a block of t, and that t<u means that every block of t is included in a block of u. So (12, 3, 456) > (12, 3, 46, 5) and
(1, 23, 45, 6) < (123, 456)

--if [t, t'] is a pp, and if u < t, then [u, t'] is also a pp.

--if [t, t'] is a pp, and if v > t', then [t, v] is also a pp.

--if [t1, t1'] and [t2, t2'] are pp's, then so are [(t1+t2), (t1'+t2')] and [(t1xt2), (t1'xt2')]
(We will usually omit the multiplication sign, writing the last pp as [(t1t2), (t1't2')])

--If [u, u] is a pp, then p is CLOSED, i.e., closed partitions are a special case of partition pairs.

Definition: if t is a partition, then m(t) is the smallest partition such that [t, m(t)] is a pp.
Note that, for every partition p, [p, I] is a pp.

Definition: M(t) is the largest partition such that [M(t), t] is a pp.
Note that, for every partition p, [0, p] is a pp.

Definition: pab is the partition equating *only* a and b.

mab = m(pab) is the smallest partition that equates every pair directly implied by ab.
For example, in the table of Example 2 above, m13 = (134, 25).

Clearly, [pab, mab] is a pp.

Generating the pp's for a Flow Table
1. For ever pair of states i, j, find mij.

2. Add all combinations of the mij's repeatedly until nothing new is found. This set of partitions are the m-partitions.

3. For each m-partition u, find M(u). The resulting set of pp's, {M(u), u} are called the Mm pairs. All pp's for the given machine are either Mm pairs or can be found from Mm pairs by making the first members smaller or the second member larger (or both). For example, if [(1234, 567, 8), (1, 35, 45, 78)] is an Mm pair, then other pp's include
[(12, 34, 5, 67, 8), (1, 35, 45, 78)], [(1234, 567, 8), (135, 4578)], and
[(123, 4, 567, 8), (135, 45, 78)]. So our goal is to find the Mm pairs, which compactly give us all the pp's.

The methods used to implement the above approach will be developed by processing the following flow table to find a good state assignment.

	AB			
	00	01	11	10
1	1	1	4	5
2	3	3	4	5
3	4	1	1	5
4	2	1	4	5
5	5	3	1	5

(The only non-trivial closed partition is (1234, 5))

First, we find the mij's, which are listed below.

m12=(13, 2, 4, 5)
m13=(14, 2, 3, 5)
m14=(12, 3, 4, 5)
m15=(1345, 2)
m23=(134, 2, 5)
m24=(123, 4, 5)
m25=(14, 2, 35)
m34=(124, 3, 5)
m35=(13, 2, 45)
m45=(134, 25)

Of course, for each of mij, we have a pp, [pij, mij].

A partition p is an m-partition if for some partition q, [q, p] is a pp and there is no x<p such that [q, x] is also a pp. Next, we add them in various combinations to generate additional m-partitions. This yields 3 more m-partitions:
m14 + m23 = (1234, 5)
m14 + m25 = (124, 35)
m14 + m35 = (123, 45)

For each of the above 13 m-partitions p, we must now find M(p). One way is to add all the pij's such that mij<p. So, for example, we can find
M(13, 2, 45) = p13+p35 = (135, 2, 4).
Also, M(123, 45) = p12 + p14 + p24 + p35 = (124, 35).

Another way is to find, for each input column, i, a partition ri such that the column-I NS entries for the blocks of ri map into blocks of p. The product of the ri partitions for all the columns is the desired M(p).

A third method is to assign a unique label to each block of p, and then to replace each NS entry in the flow table with the label of the u-block containing that entry. Then specify as a block of M(p) each set of rows with identical patterns of labels.

The resulting list of Mm partitions is:

[(12, 3, 4, 5), (13, 2, 4, 5)]
[(13, 2, 4, 5), (14, 2, 3, 5)]
[(14, 2, 3, 5), (12, 3, 4, 5)]

[(1235, 4), (1345, 2)]
[(123, 4, 5), (134, 25)]
[(124, 3, 5), (123, 4, 5)]
[(13, 25, 4), (14, 2, 35)]
[(134, 2, 5), (124, 3, 5)]
[(1, 2, 35, 4), (13, 2, 45)]
[(123, 45), (134, 25)]
[(1234, 5), (1234, 5)]
[(134, 25), (124, 35)]
[(124, 35), (123, 45)]

Examining this set, we can see a fortunate "cyclic sequence" of pp's involving exactly 3 2-block partitions, namely, (123, 45), (124, 35), (134, 25). The pp's are
[(124, 35), (123, 45)], [(134, 25), (124, 35)], [(123, 45), (134, 25)]

If we choose these to construct our state assignment (which is possible, since their product is the 0-partition), assigning y1, y2, and y3 to them in the order given, then we benefit from the fact that (in addition to A and B), Y1 depends only on y2, Y2 depends only on y3, and Y3 depends only on y1.

The result is as shown below:

	AB						
	00	01	11	10	y1	y2	y3
1	1	1	4	5	0	0	0
2	3	3	4	5	0	0	1
3	4	1	1	5	0	1	0
4	2	1	4	5	1	0	0
5	5	3	1	5	1	1	1

Y1=B'y2 + AB'
Y2=A'y3 + AB'
Y3=B'y1 + AB'

Another example, with only a sample of the work done is:

	AB			
	00	01	11	10
1	1	3	4	6
2	3	2	6	5
3	1	2	6	4
4	5	6	2	3
5	5	4	3	2
6	4	6	2	1

pp's include
from m(12), [(12, 3, 4, 5, 6), (123, 456)]
from m(14) [(14, 2, 3, 5, 6)(15, 24, 36)]

from m(26) [(1, 26, 3, 4, 5)(15, 26, 34)]
from m(35) [(1, 2, 35, 4, 6)(15, 24, 36)]

Adding m(14)+m(26)+m(35)=[(14, 26, 35), (15, 2346)]

Also, [(14, 2, 35, 6), (15, 2346)] is also a pp

DECOMPOSITION OF MACHINES

Using closed partitions.

Example 1:

	X			
	0	1	y1	y2
1	3	2	0	0
2	4	2	0	1
3	1	2	1	1
4	2	1	1	0

(12, 34) is a closed partition. We can define a machine, M1, based on this partition, with input X, and one state variable, which distinguishes the blocks of this partition. Its output is y1, which is fed to a second machine, M2, corresponding to the (non-closed) partition, (14, 23), which can be used to complete the state assignment. (The only requirement for the second partition is that its product with the first partition is 0.)

The flow table for M1 (where row-1 represents the first block of (12, 34) and row-2 the second block) is:

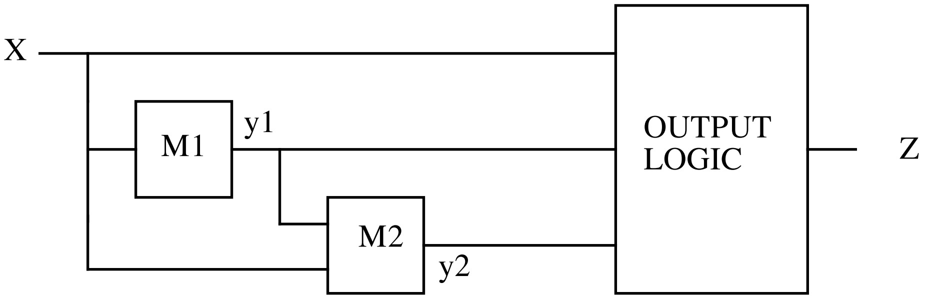
		X		
		0	1	y1
(12)	1	2	1	0
(34)	2	1	1	1

M1

		Xy1				
		00	01	11	10	y2
(14)	1	2	2	1	2	0
(23)	2	1	1	2	2	1

M2

The resulting decomposed circuit is shown below. Note that resulting logic is the same as if we treated this as a straightforward synthesis using the given state assignment. The difference is only conceptual, in that we treat the system as the interconnection of two clearly defined machines. The rightmost block generates the output logic, given the input signals and the state variables.



Our second example involves the machine below, which is the same machine used above to illustrate finding the lattice of closed partitions. We will use the results obtained there.

In particular we use the fact that the closed 2-block partition (12, 345) is a factor of the two closed 4-block partitions, (12, 34, 5) and (1, 2, 3, 45), Recall that we found 2 non-closed 2-block partitions that when multiplied by (12, 345) yielded the two 4-block partitions. We will define 3 machines. First M1, that implements the common factor partition (12, 345). The flow table for this machine, with state-variable y1 assigned, is:

		A	B	C	y1
(12)	1	1	2	1	0
(345)	2	2	2	1	1

M1

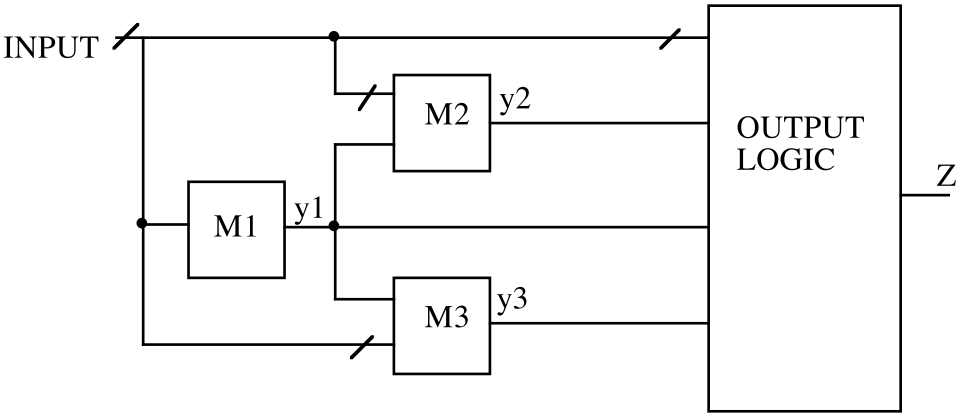
		Ay1'	Ay1	By1'	By1	Cy1'	Cy1	y2
(125)	1	1	2	1	2	1	1	0
(34)	2	-	1	-	2	-	1	1

M2

		Ay1'	A y1	By1'	By1	Cy1'	Cy1	y3
(145)	1	2	1	1	2	1	2	0
(23)	2	1	1	1	1	1	1	1

M3

The block diagram of the circuit is shown below.


INPUT
M2
y2
M1
y1
M3
y3
OUTPUT
LOGIC
Z