

FINITE-STATE MODELING OF SOFTWARE

Why Make a Finite-State Model:

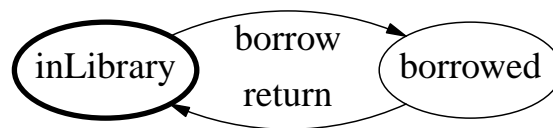
- FSM is a *finite* description of a potentially *infinite* number of behavior-sequences (similar to a program).
- It is programming language independent.
- Can generate semi-automated code from the FSM.
- Can easily build subsystem-models and higher level (architecture) models from FSM.

Remarks:

- (1) Converting a program P to an FSM $M(P)$ can be automated.
- (2) Building an FSM from the problem-statement or requirements, without the software, cannot be fully automated, and remains very much a human-centered activity.

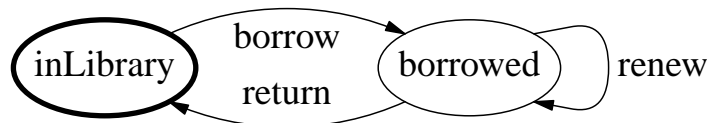
AN EXAMPLE

- Shown below is a simple FSM for the borrow-return operations of a library book; it shows that
 - (1) the two operations must alternate, and
 - (2) the start-operation is borrow.



Question: What makes "borrow" the start-operation?

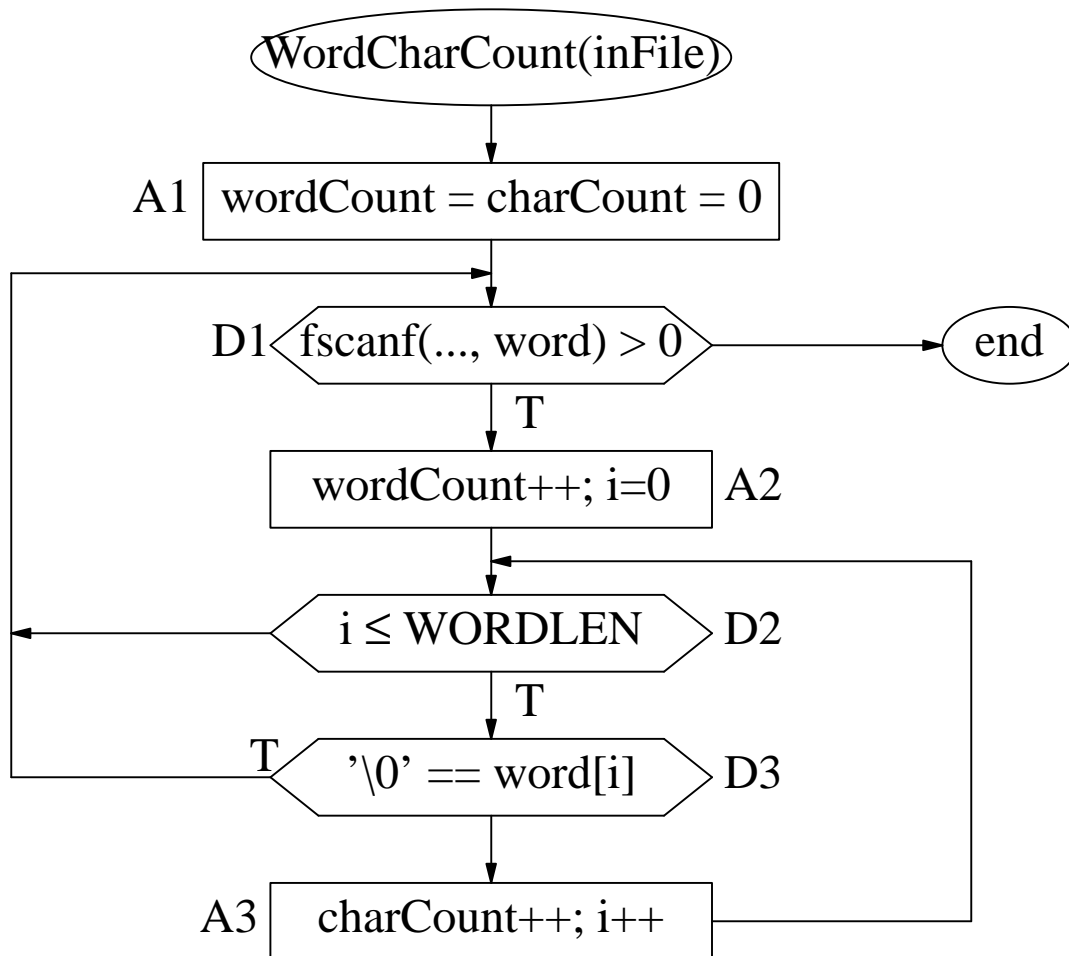
- A modified FSM to include the renew-operation.



Questions

- ? What distinguishes renew-operation from return-operation and how is it reflected/captured in the FSM?
- ? Do you see any shortcoming in this model?
- ? What would be the new FSM if we assume that one can renew the book at most 2 times? (Is there a need for such a restriction?)
- ? How to model the fact that the person borrowing the book is the person renewing it? (Is this restriction necessary?)

FLOWCHART OF WordCharCount



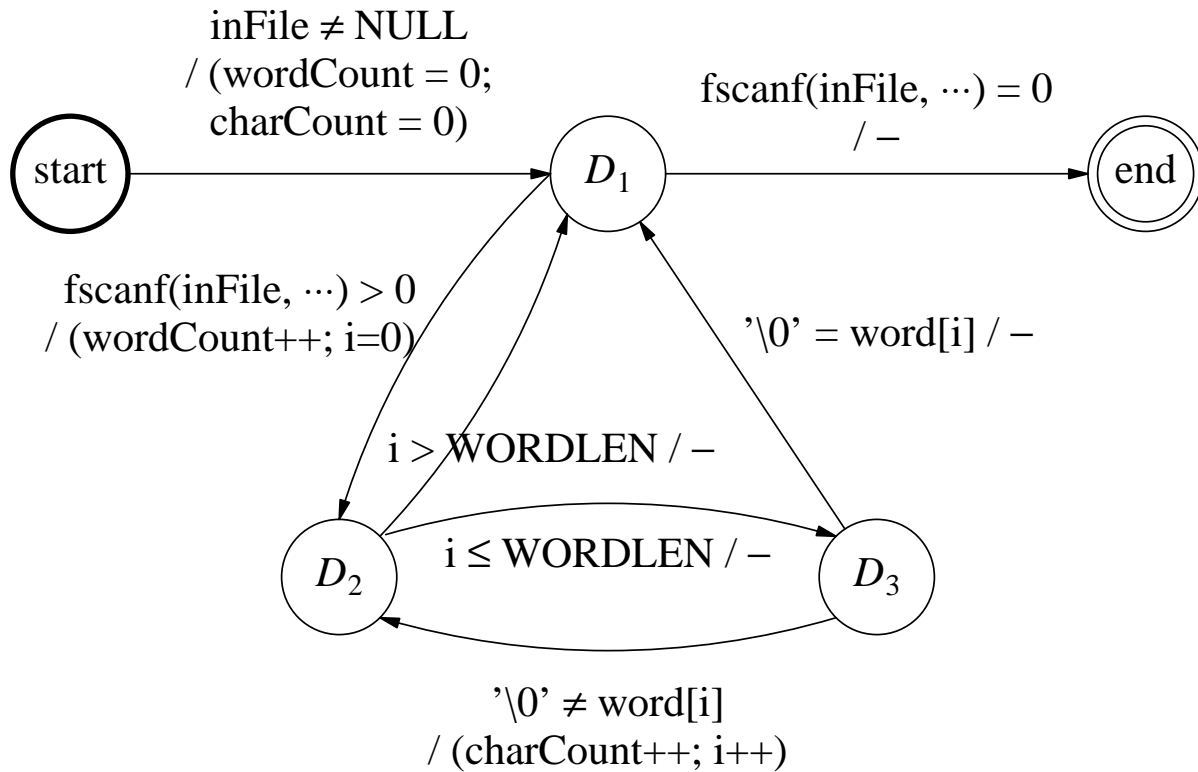
Decision to decision path (DD-path):

The "chunk" of activities, if any, between two consecutive branch-points on a path from start to end.

- Start → A1 (→ D1)
- D1 → A2 (→ D2); D1 → end;
- D2 (→ D1); D2 (→ D3)

Question: What is the relationship between the #(DD-paths) and #(decision-points) in the program? (Assume that each decision is two-way: true and false.)

FINITE-STATE MODEL FROM DD-PATHS



Conditions and actions: c_{ij}/a_{ij} .

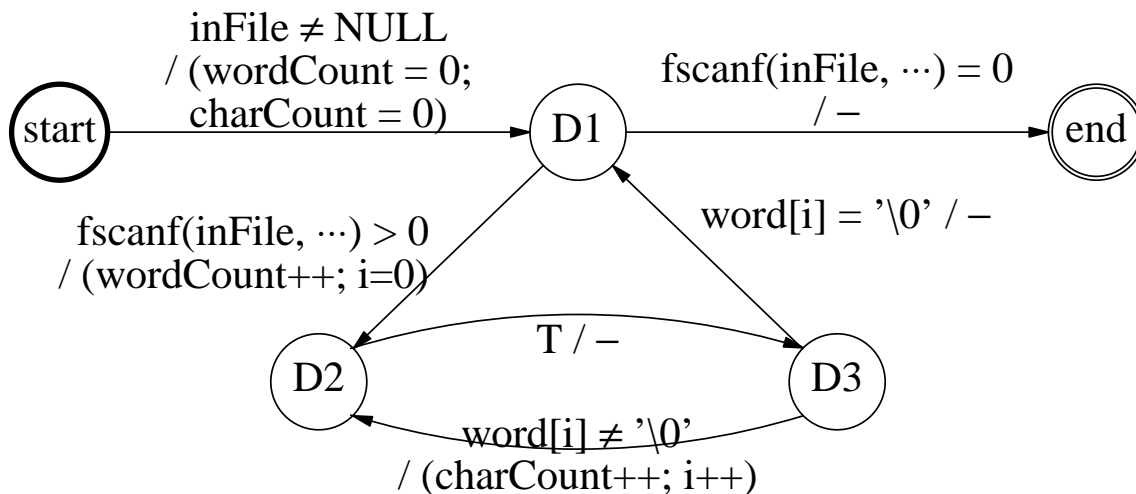
- Each node, other than the start and end nodes, has two transitions from it.
- Only one of the conditions c_{ij} will hold at each state at any point (deterministic behavior).
- Cycles in the flowchart give cycles in the FSM.

Each decision-node plus current values of local/global vars.
gives an abstraction of the history of computation which
determines the future computations from that point on.

SOME SIMPLIFICATIONS

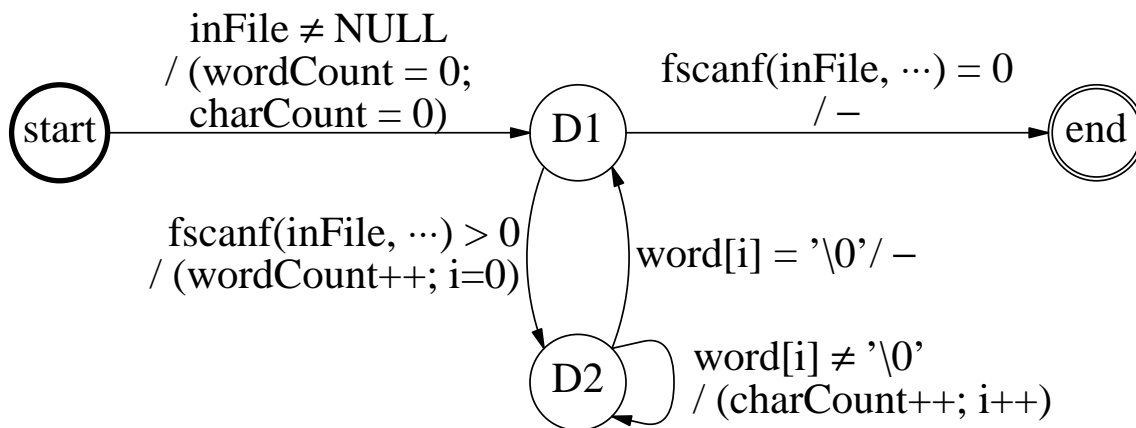
Elimination of unused transition: " $i > \text{WORDLEN}/-$ "

- The condition " $i \leq \text{WORDLEN}$ " becomes "T" (true).



Elimination of the transition with Condition "T":

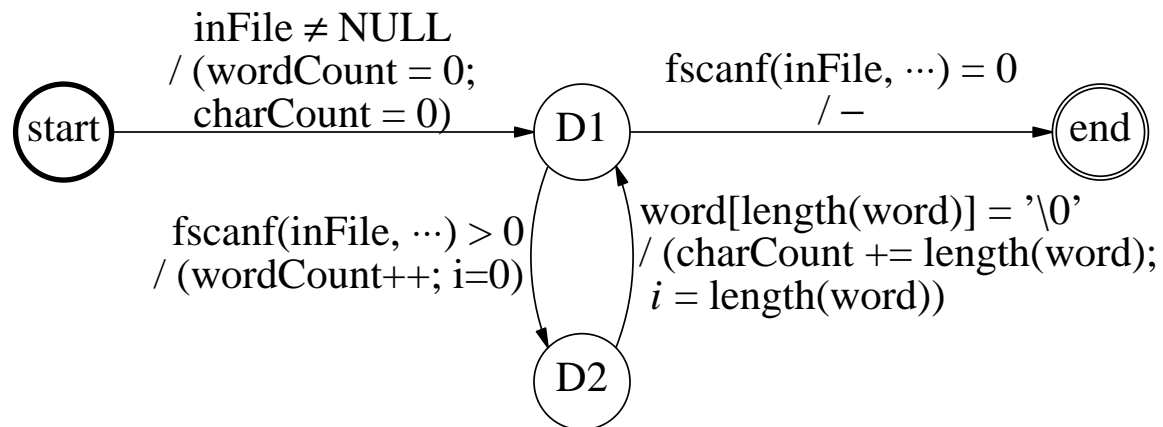
- This affects both (D3, D2) and (D3, D1).



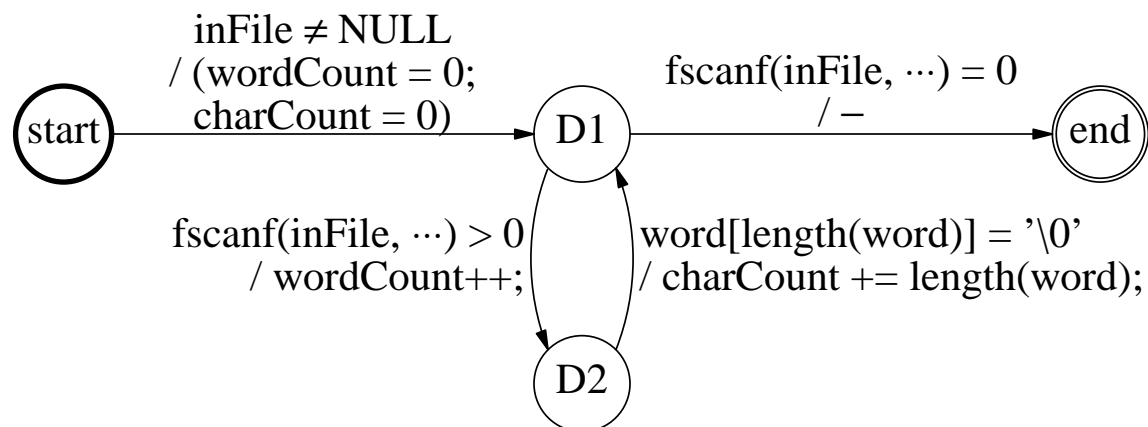
On loop termination: $i = \text{length}(\text{word})$

LOOP ELIMINATION

Loop Elimination:



Elimination of Variable *i*:



Simplified Code:

```

void WordCharCount(FILE *inFile)
{ char word[WORDLEN+1]; //1 for end of string

  wordCount = charCount = 0;
  while (fscanf(inFile, "%s", word) > 0) {
    wordCount++;
    charCount += length(word);
  }
}

```

SUMMARY

All computations can be modeled,
at any desired level, by FSMs
using condition-guards on the transitions.

Remarks:

- This is basically a restatement of the Church-Turing hypothesis:

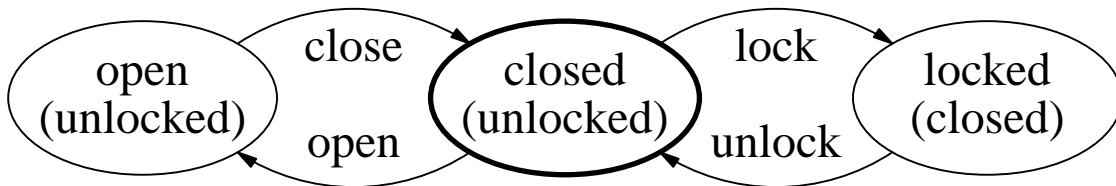
Each computation/algorithm can be
modeled by a Turing Machine.

- Construction of an FSM, without having a program in hand, is often a non-trivial task.

MORE EXAMPLES OF FSM

Window with a Lock:

- Four operations: open, close, lock, and unlock.
- Constraints:
 - can be opened only if it closed and unlocked.
 - can be closed only if it opened.
 - can be locked only if it is closed and unlocked.
 - can be unlocked only if it is locked.
- Initially closed and unlocked.



Proper state-names help us to easily identify the applicable actions at a state.

- There are no condition-guards for the transitions here (why?).
- There are no final states here because the operations can be continued for ever, without termination.
- We allow transitions to the start-state to keep the number of states small.

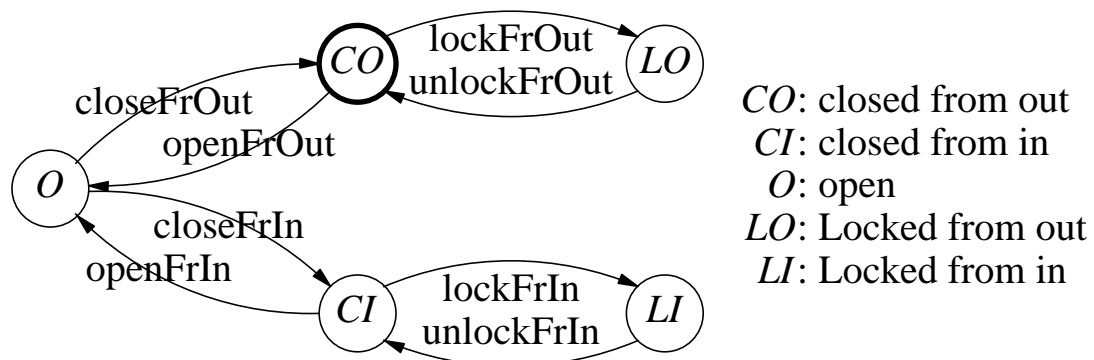
MORE EXAMPLES OF FSM

Door With Two-sided Lock:

- Eight operations: openFromIn, closeFromIn, lockFromIn, and unlockFromIn, and similar operations from out.

Imagine a person moving in and out of the room when the door is open; the person's moves are not modeled.

- Initially, the door is closed from out and unlocked.
- Constraints:
 - Similar to those for the window for the operations from in and for the operations from out.
 - "Inside" operations can occur only after the operation openFromOut, and likewise for "outside" operations.
- The door can be closed/locked from one side at a time.

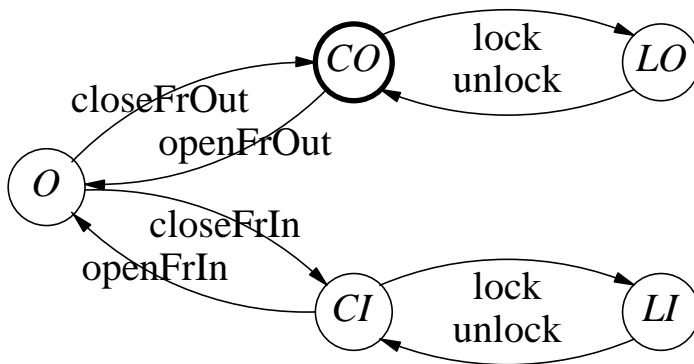


EXERCISE

1. Show the new FSM after we add the operations goIn and goOut to model the person's move.

THE CHOICE OF OPERATION-NAMES CAN AFFECT THE FSM

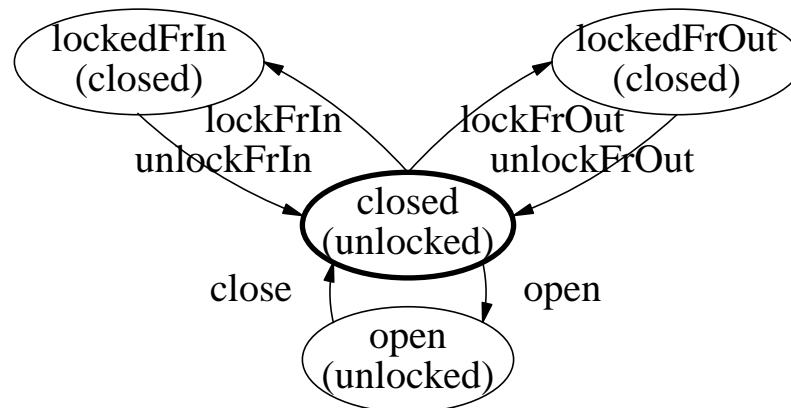
- Imagine a single lock-function that takes into account the state of the door-with-two-sided-lock and performs the appropriate operation lockFrIn or lockFrOut as needed.
- Similarly for the unlock-function.



Cannot use the same name "close" for both closeFrIn and closeFrOut because it causes non-determinism.

EXERCISE

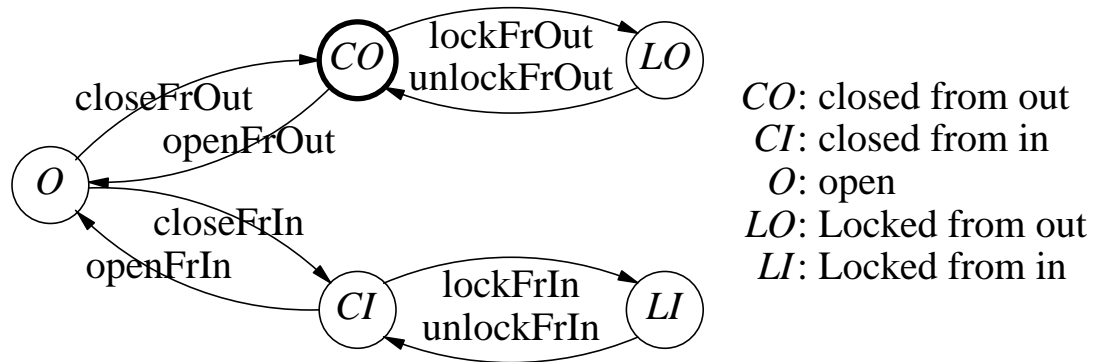
1. Show the new FSM when we use "close" both for closeFrIn and closeFrOut and similarly for open, but keep different names lockFrIn and lockFrOut. (Hint: you may need more states; following FSM is no good - why?)



2. How many ways a state-diagram can fail to represent a proper FSM?
3. How many FSM's are possible with n states and m actions/events if we do not use any guards and have no final states? (Remember that there may not be a transition for every state-event pair.)

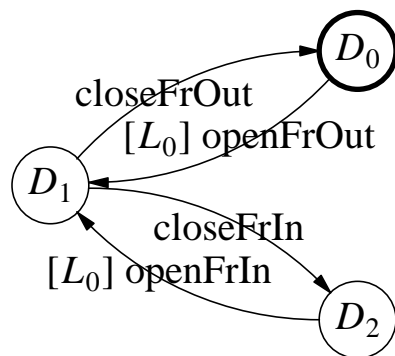
DECOMPOSING AN FSM

The FSM for Door With Two-sided Lock:

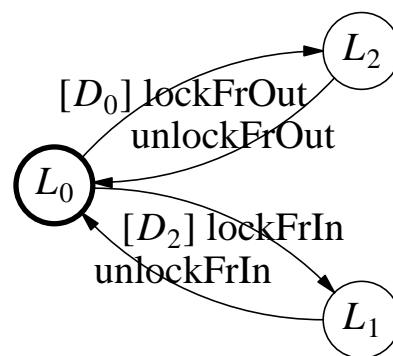


Decomposition into Two FSMs:

- We use guards to coordinate the interaction between them.
- The composition $M(D) \times M(L)$ gives the original FSM.



The finite-state model
 $M(D)$ for door.

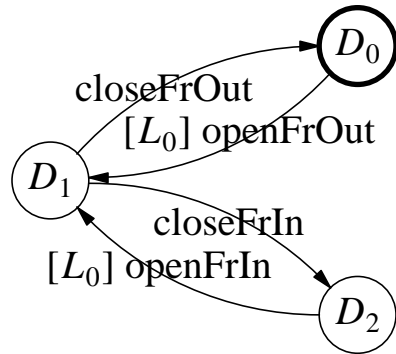


The finite-state model
 $M(L)$ for two-sided-lock.

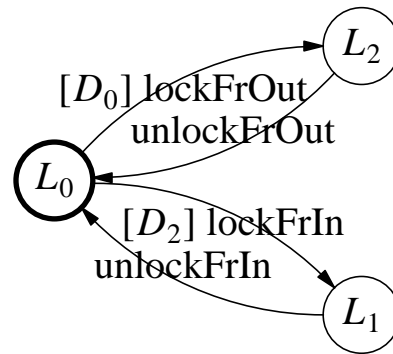
Question: What do the states in $M(D)$ and $M(L)$ look like in terms of the states in the original FSM?

FORMING THE COMPOSITION $M(D) \times M(L)$

Starting FSMs $M(D)$ and $M(L)$:



$M(D)$ for door.



$M(L)$ for two-sided-lock.

Composition $M(D) \times M(L)$:

- The dashed transitions are not present due to guards.
- The shaded states are not there because they are not reachable from the start-state D_0L_0 .

