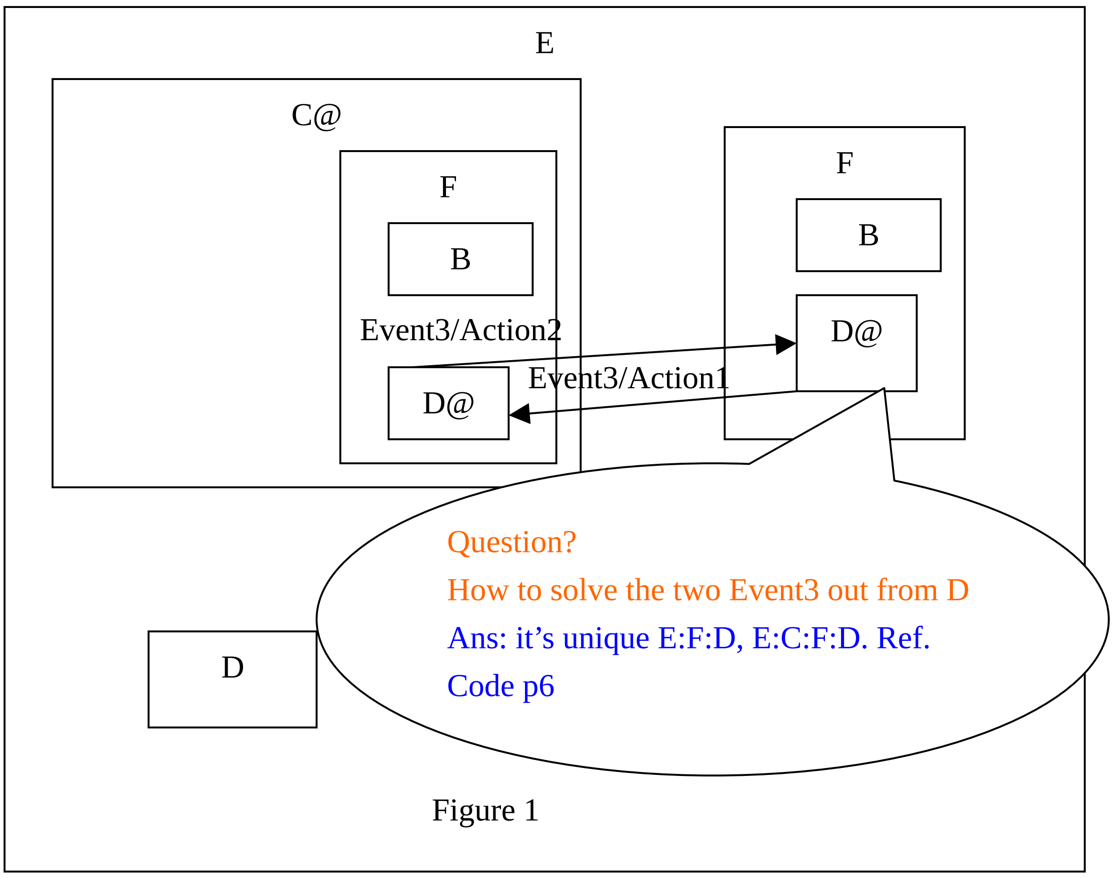

E
C@
F
B
Event3/Action2
D@
Event3/Action1
F
B
D@
D
Question?
How to solve the two Event3 out from D
Ans: it’s unique E:F:D, E:C:F:D. Ref.
Code p6

Figure 1

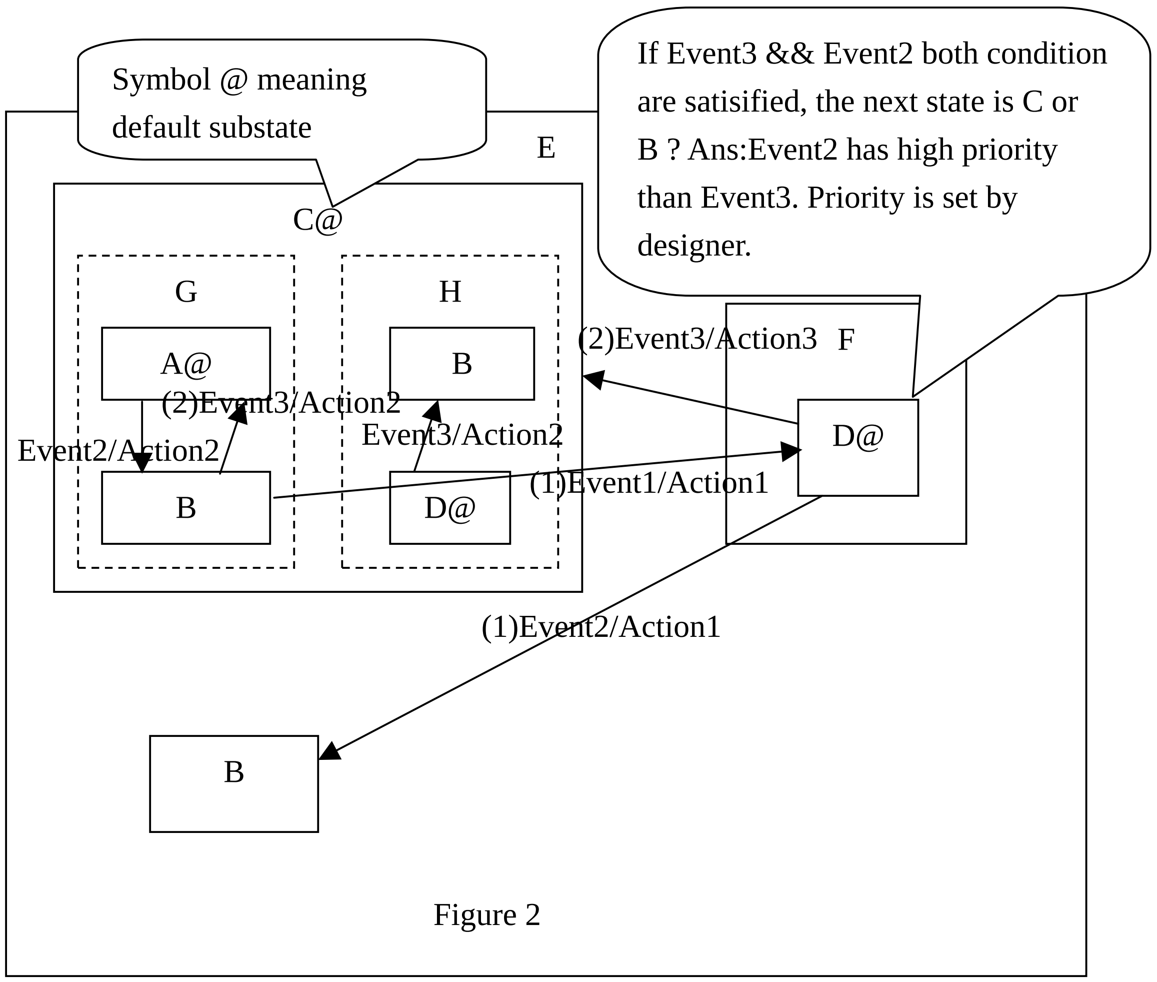

Symbol @ meaning default substate
If Event3 && Event2 both condition are satisisfied, the next state is C or B ? Ans:Event2 has high priority than Event3. Priority is set by designer.
E
C@
G
H
A@
B
(2)Event3/Action2
Event2/Action2
Event3/Action2
B
D@
(2)Event3/Action3
F
D@
(1)Event1/Action1
(1)Event2/Action1
B

Figure 2

*note : G is Interaction Thread; H is Background Thread.

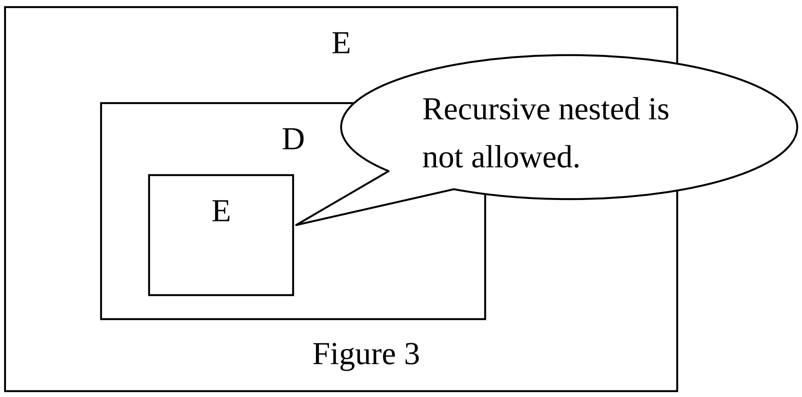


Figure 3

I explain my algorithm using the Figure 4.

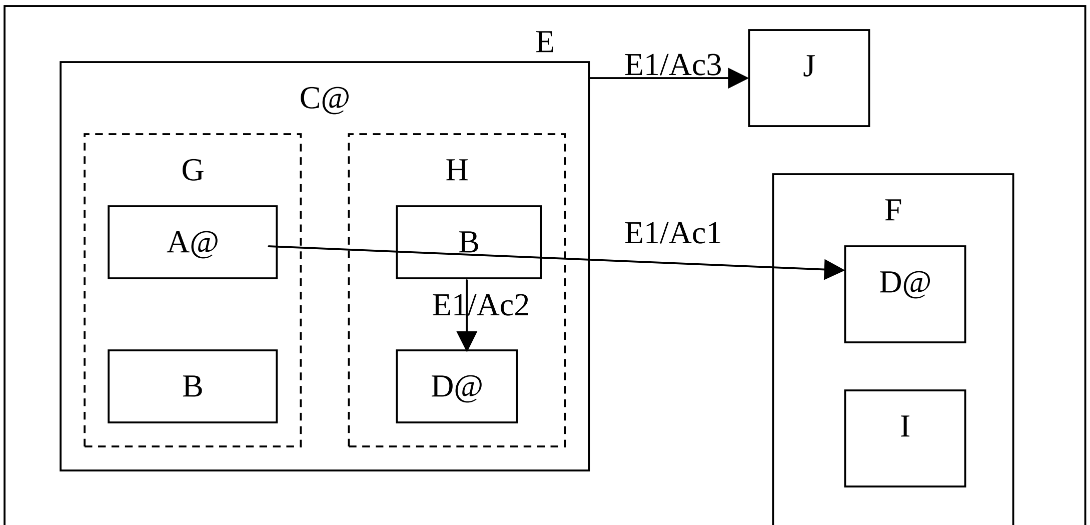
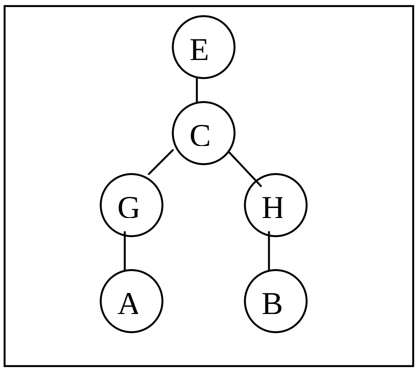


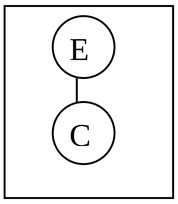
Figure 4

The tree is:



```
case A:
    if (E1 && Parent(1) == G && Parent(2) == C && Parent(3) == E) {
        Ac1();
    }
case B:
    if (E1 && Parent(1) == H) {
        Ac2();
    }
case C:
    if (E1 && Parent(1) == E) {
        Ac3();
    }
```

When e1 occurs:

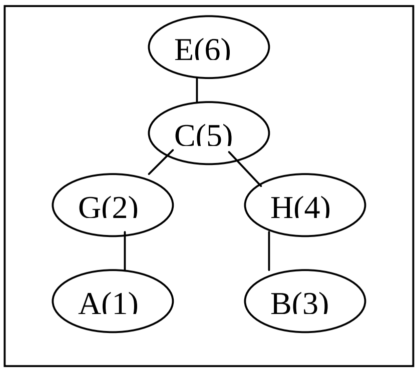
Result = 

EFSM_E:

State	do
E:F:D	Ac1
E:C:H:D	Ac2
E:J	Ac3

So the algorithm is:

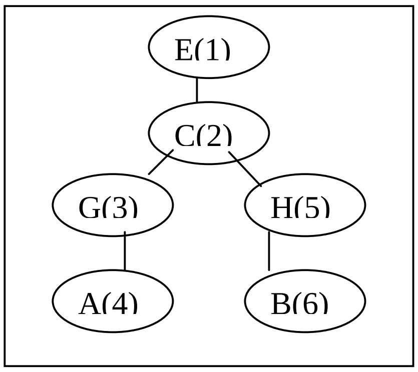
1. The order of do Action is bottom_up



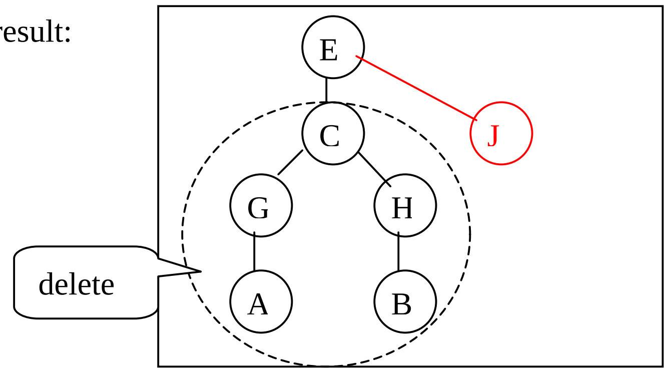
We change the state by this order.

And if we have changed the state, we stop, and go to Pivot and try to next thread or parent.

Eg. In figure 4, when we were at C, and meet e1, we change to E:J, and we stop try to meet e1 at A and B.



The result:



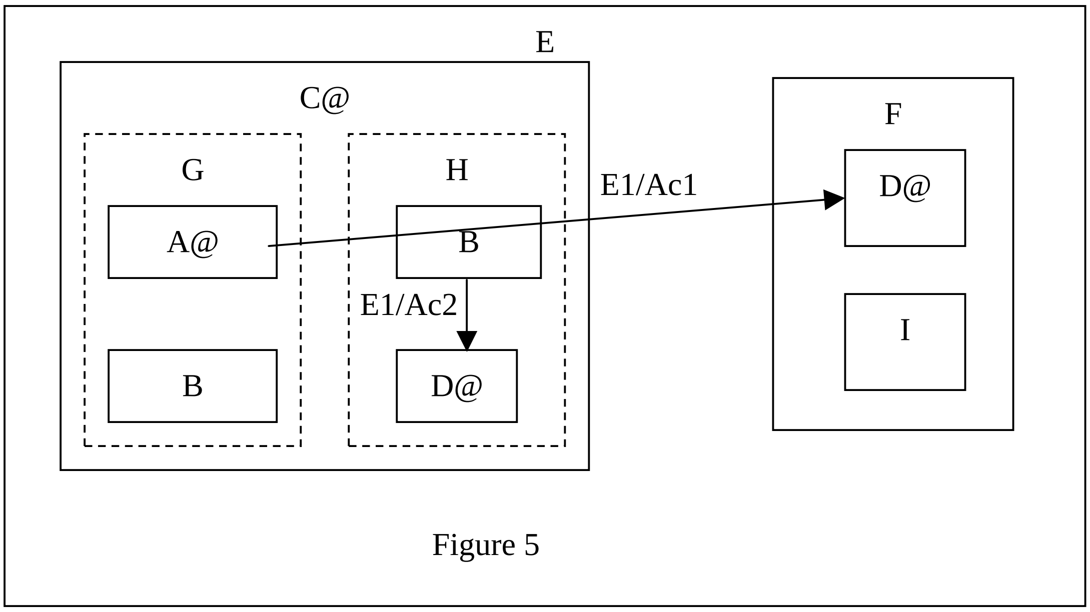
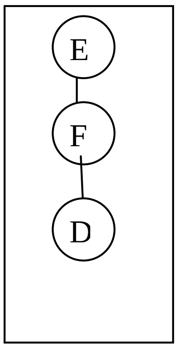


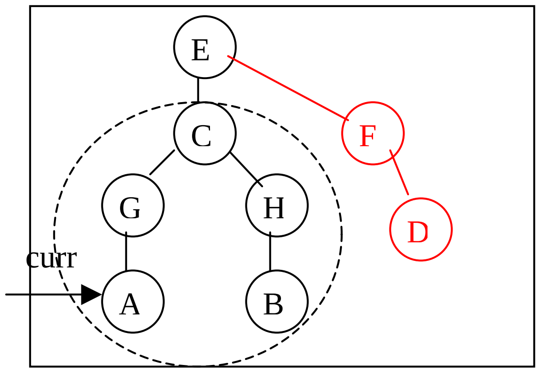
Figure 5

When e1 occurs:

Result =



So, according algorithm:



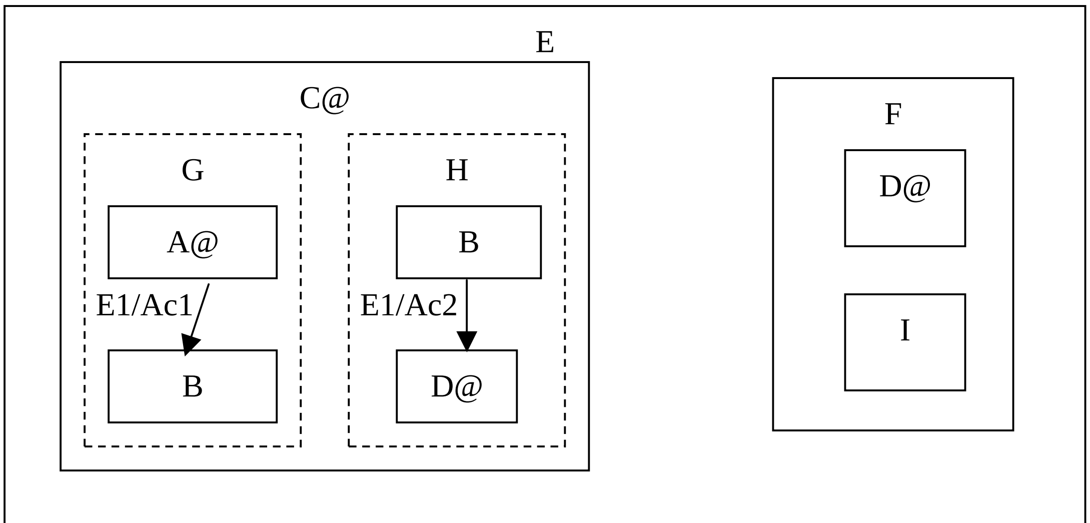
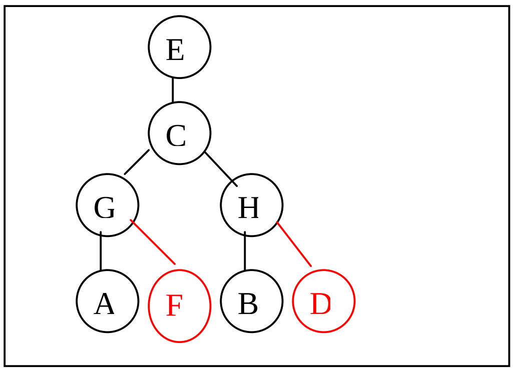


Figure 6

So, according algorithm:



Proof:

1. It's right for those cases of no Background Thread.
2. Limit: State in Back Ground Thread can't escape outside. (It's reasonable for modeling the reality.)
3. So, we only need consider (1).state in the Interaction Thread (Foreground Thread) escape to outside of Interaction Thread and (2).didn't escape to outside of Interaction Thread.

   My algorithm is work for these two cases.