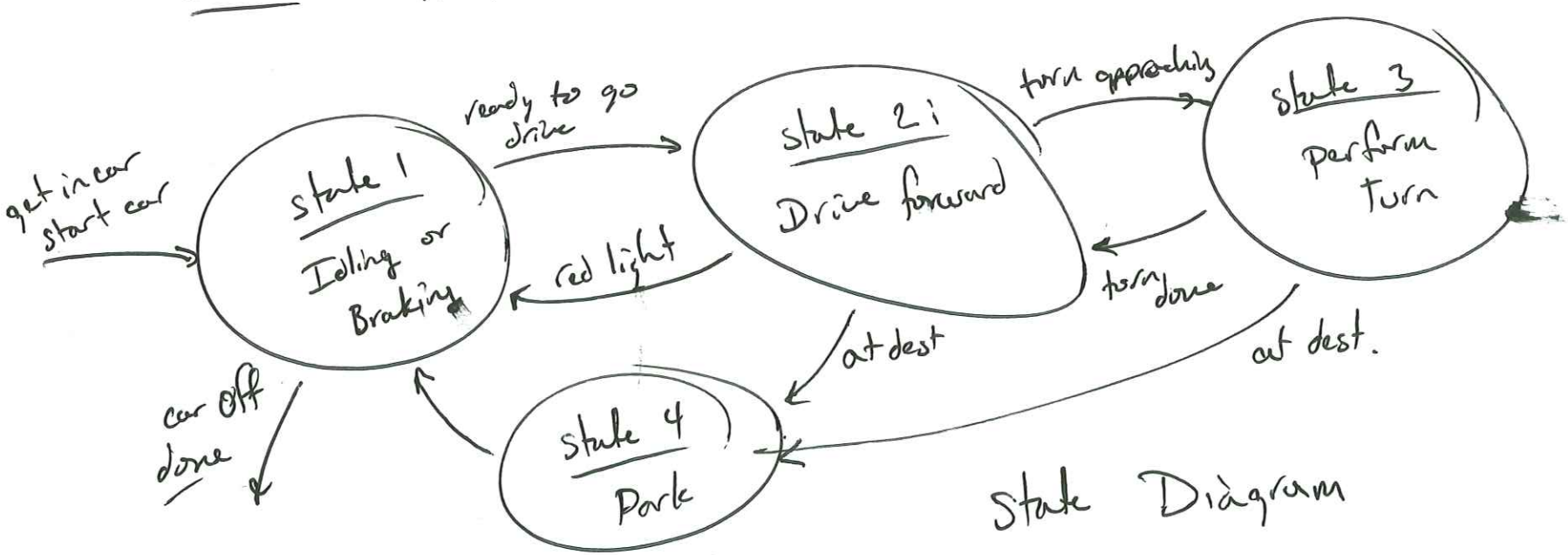
Algorithm Development

States: A distinct operational state or configuration.

Events: Triggers/occurrences that cause a state change



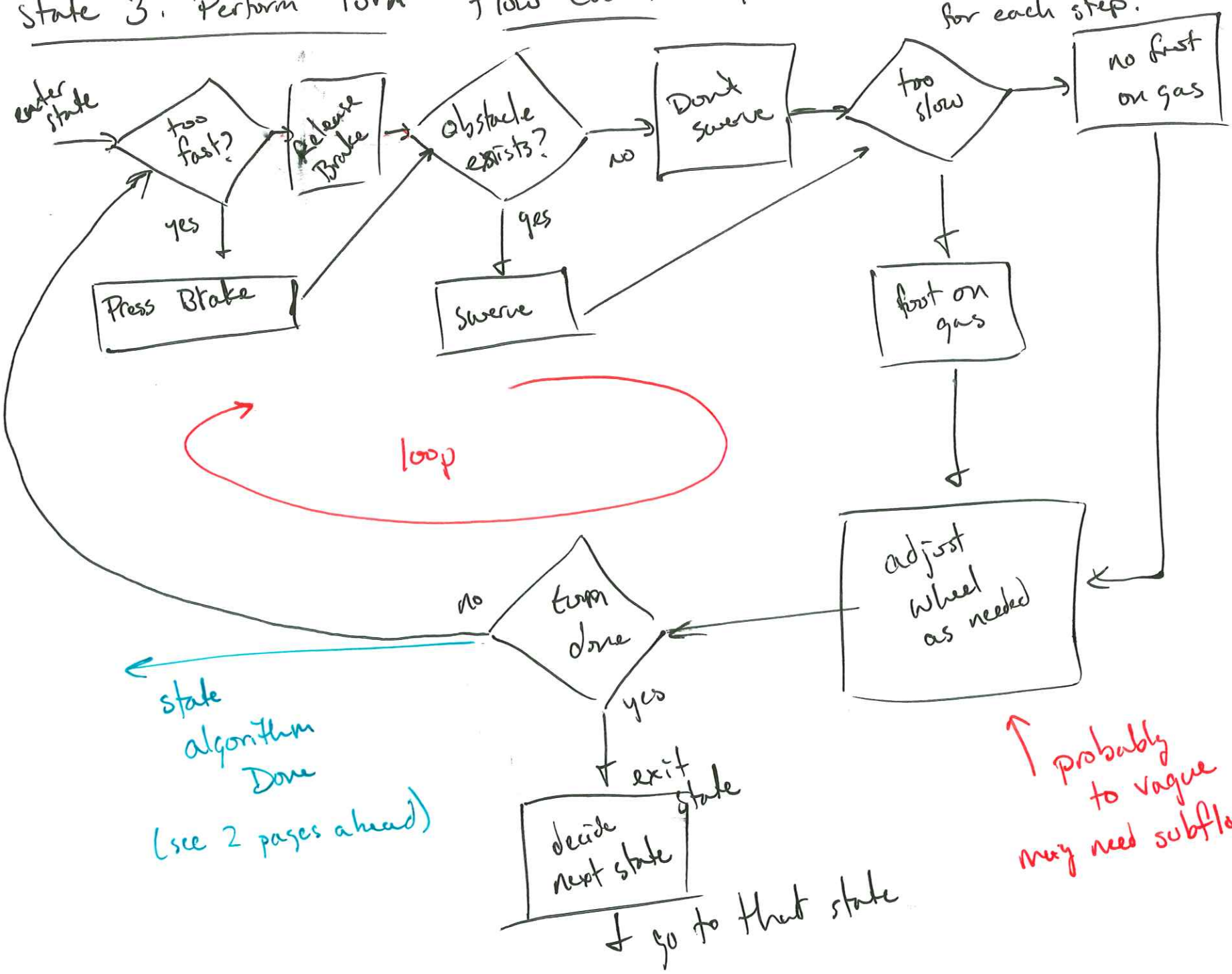
State Diagram

At a high-level, I build a state diagram to simplify a program...

State Diagrams are typically high-level

High-Level: Big Picture

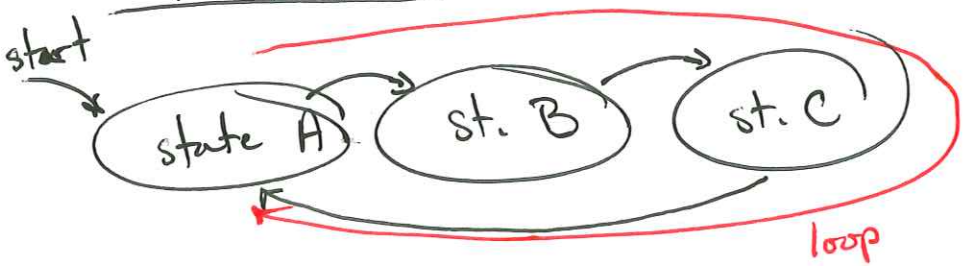
State 3: Perform Turn flow Chart → low-level explicit operations for each step.



Programming States

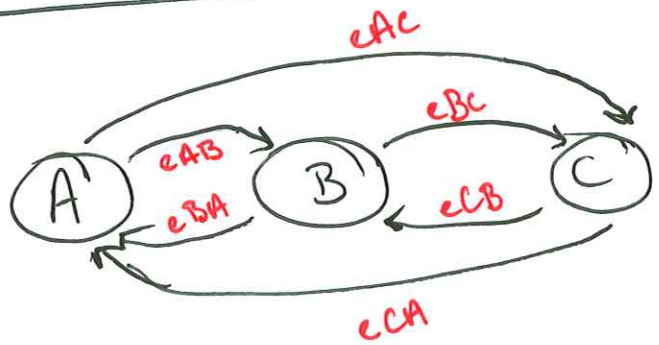
Assume states are written as functions with internal loops.

Simple program:




```
while(1){
    stateA();
    stateB();
    stateC();
}
```

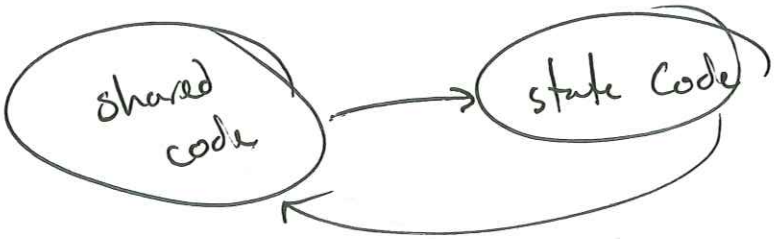
More Complicated:




```
uint8_t state = 'A';
while(1){
    if( state == 'A'){
        stateA();
        if(eAB)  state = 'B';
        if (eAC)  state = 'C';
    } else
    if (state == 'B'){
        stateB();
        if(eBC)  state = 'C';
        if (eBA)  state = 'A';
    }
    ;
}
```

Alternative: Assume functions represent states with NO INTERNAL LOOP

then I can do:




```
state = 'A';
while (1){
    sharedCode();
    if( state == 'A'){
        // stuff for state A ....
        // state transitions
    }
    .
    .
    .
}
```

← for example: read all important inputs

← should be quick! No loop

finite State Machine "fsm"

System defined by states & inputs OR events.

State $\in \{A, B, C, D, \ldots\}$

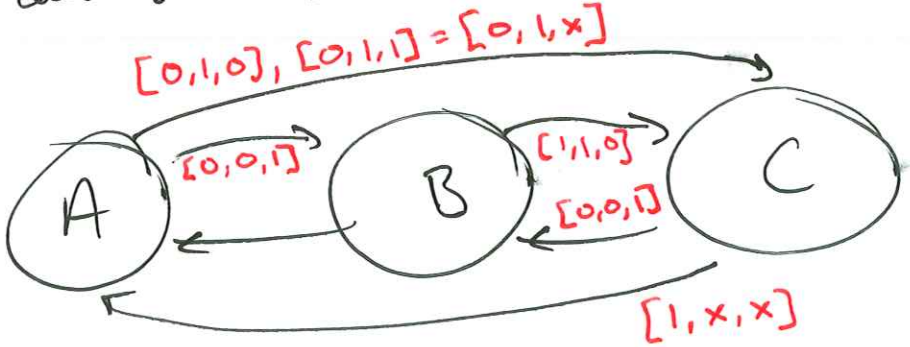
inputs : {pb1, pb2, ss, temp, ...}

for example: A 3-input system: [pb1, pb2, ss]

inputs can be: [0,0,0], [0,0,1], [0,1,0] .... [1,1,0], [1,1,1]

x = don't care.

3 States:




```
state = 'A';
while(1){
   readInputs();
   ; other shared stuff
   if (state == 'A'){
      stateA();
      if(inputs == [0,0,1]) state = 'B';
      if( inputs == [0,1,x]) state = 'C';
   } ;
     ;
```

Not valid code.

Pseudocode

code written in easy to understand language.
could easily (usually) be also represented as a flow chart.

for example: state 3: Perform turn:

```
infinite loop:
        Going too fast?
                yes: press brake
                no : release brake
        Obstacle in way?
                yes: swerve
                no : don't swerve
        :
        :
```

Indentity is extremely important to understanding!