



---

A large portion of today's programming is for embedded microprocessors. Every dish-washer, alarm clock, washing machine, car, phone, etc. has one or more microprocessors inside.



A large portion of today's programming is for embedded microprocessors. Every dish-washer, alarm clock, washing machine, car, phone, etc. has one or more microprocessors inside.

Some decades ago, an elevator was controlled by relays. Today, there is a small computer that does everything.



---

A large portion of today's programming is for embedded microprocessors. Every dish-washer, alarm clock, washing machine, car, phone, etc. has one or more microprocessors inside.

Some decades ago, an elevator was controlled by relays. Today, there is a small computer that does everything.

Electrical engineers spend a lot of time programming embedded devices.

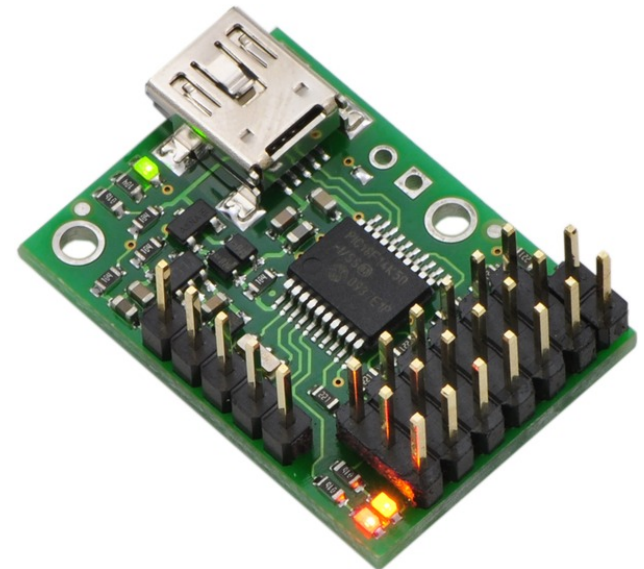


A large portion of today's programming is for embedded microprocessors. Every dish-washer, alarm clock, washing machine, car, phone, etc. has one or more microprocessors inside.

Some decades ago, an elevator was controlled by relays. Today, there is a small computer that does everything.

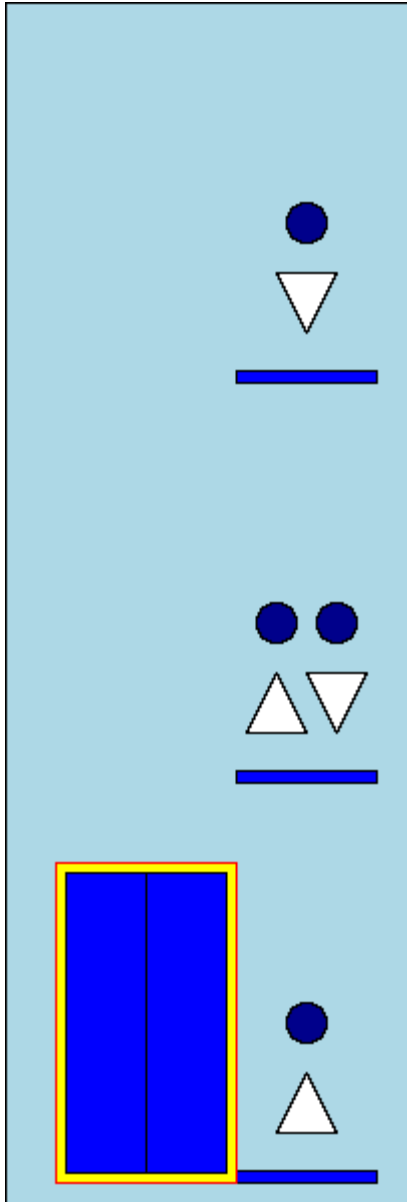
Electrical engineers spend a lot of time programming embedded devices.

You can buy off-the-shelf devices that allow you to control hardware using a USB connection.





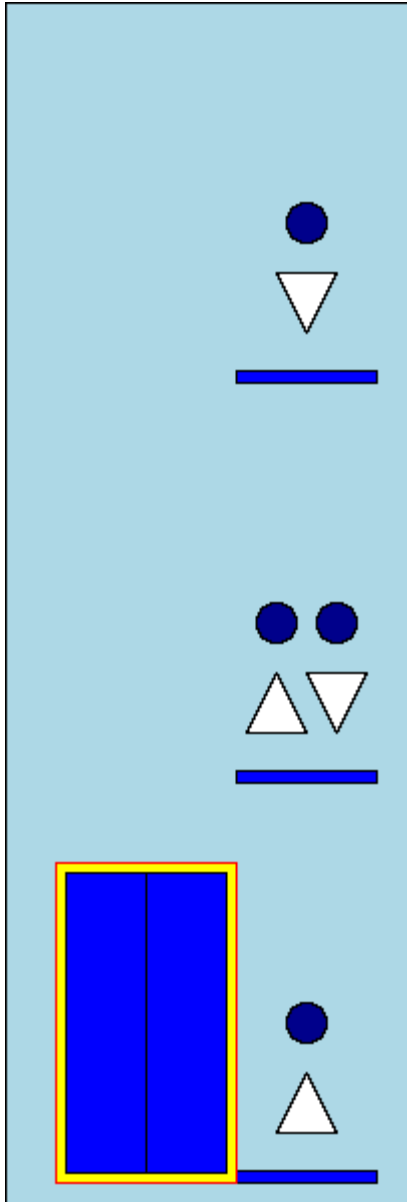
An elevator for three floors.





An elevator for three floors.

**Hardware:** Elevator motor, three floor sensors, four call buttons, four call lights, elevator door.



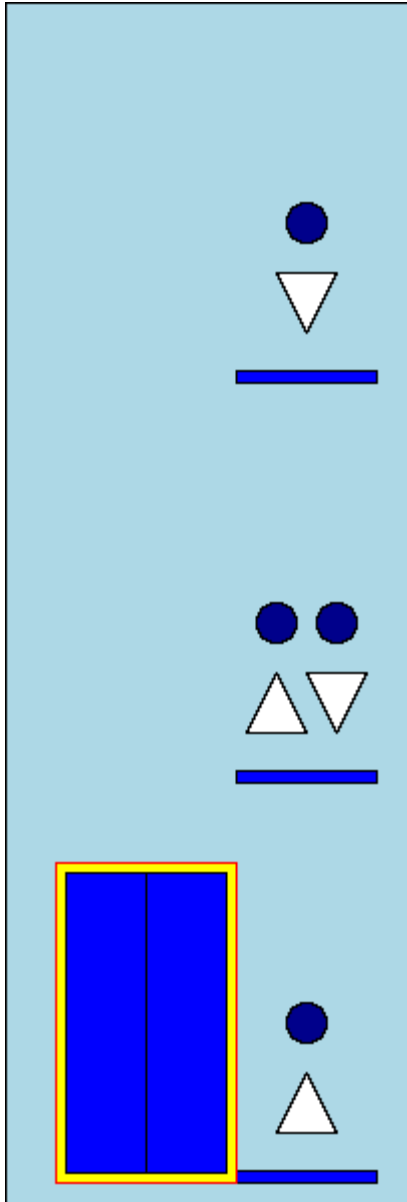


An elevator for three floors.

**Hardware:** Elevator motor, three floor sensors, four call buttons, four call lights, elevator door.

**Software interface:**

```
from elevator import *  
init_hardware()  
set_motor(speed)           # -127..127  
set_light(num, state)      # num 0..3  
open_door()  
close_door()  
get_button(num)            # num 0..3  
get_sensor(floor)          # floor 1..3
```





## Controlling the motor

---

When the motor is running, we have to continuously check the floor sensors to stop the motor when we arrive at the right floor (or the elevator will shoot out of the shaft and fall off the roof).





When the motor is running, we have to continuously check the floor sensors to stop the motor when we arrive at the right floor (or the elevator will shoot out of the shaft and fall off the roof).

```
def up_to_two():  
    set_motor(40)  
    while not get_sensor(2):  
        pass  
    set_motor(0)
```



When the motor is running, we have to continuously check the floor sensors to stop the motor when we arrive at the right floor (or the elevator will shoot out of the shaft and fall off the roof).

```
def up_to_two():
    set_motor(40)
    while not get_sensor(2):
        pass
    set_motor(0)

def move_to_floor(speed, floor):
    set_motor(speed)
    while not get_sensor(floor):
        pass
    set_motor(0)
```



---

```
# Elevator is at this floor
```

```
current_floor = 1
```

```
def goto_floor(floor):
```

```
    global current_floor
```

```
    if floor == current_floor:
```

```
        return
```

```
    speed = 80
```

```
    if floor < current_floor:
```

```
        speed = -80
```

```
    move_to_floor(speed, floor)
```

```
    current_floor = floor
```



## Checking buttons and sensors

---

When a call button is pressed, the call light should go on immediately, even if the computer is currently “busy” monitoring the elevator movement.



When a call button is pressed, the call light should go on immediately, even if the computer is currently “busy” monitoring the elevator movement.

We need to keep calling `get_button` during cabin moves:

```
def check_buttons():
    for i in range(4):
        if get_button(i):
            pending[i] = True
            set_light(i, True)

def move_to_floor(speed, floor):
    set_motor(speed)
    while not get_sensor(floor):
        check_buttons()
    set_motor(0)
```



---

The target floor of a cabin move can change **while** the cabin is moving.



---

The target floor of a cabin move can change **while** the cabin is moving.

Let's start again at the beginning. At any time, the elevator is in one of the following eleven states:



---

The target floor of a cabin move can change **while** the cabin is moving.

Let's start again at the beginning. At any time, the elevator is in one of the following eleven states:

wait1

wait2

wait3

waiting at a floor





The target floor of a cabin move can change **while** the cabin is moving.

Let's start again at the beginning. At any time, the elevator is in one of the following eleven states:

wait1

wait2

wait3

move12

move23

move21

move32

waiting at a floor    moving between floors



The target floor of a cabin move can change **while** the cabin is moving.

Let's start again at the beginning. At any time, the elevator is in one of the following eleven states:

wait1

move12

open1

wait2

move23

open2up

wait3

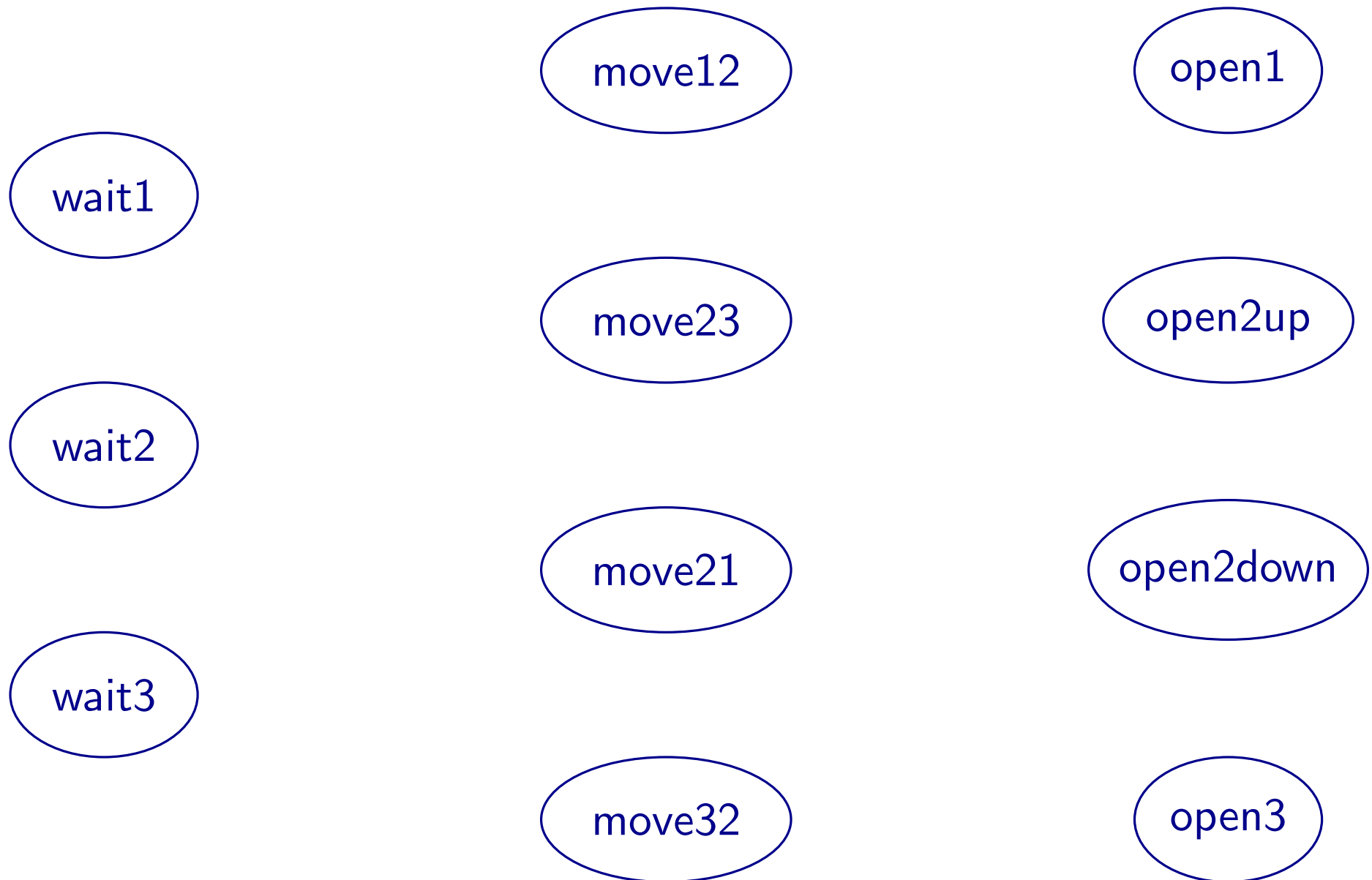
move21

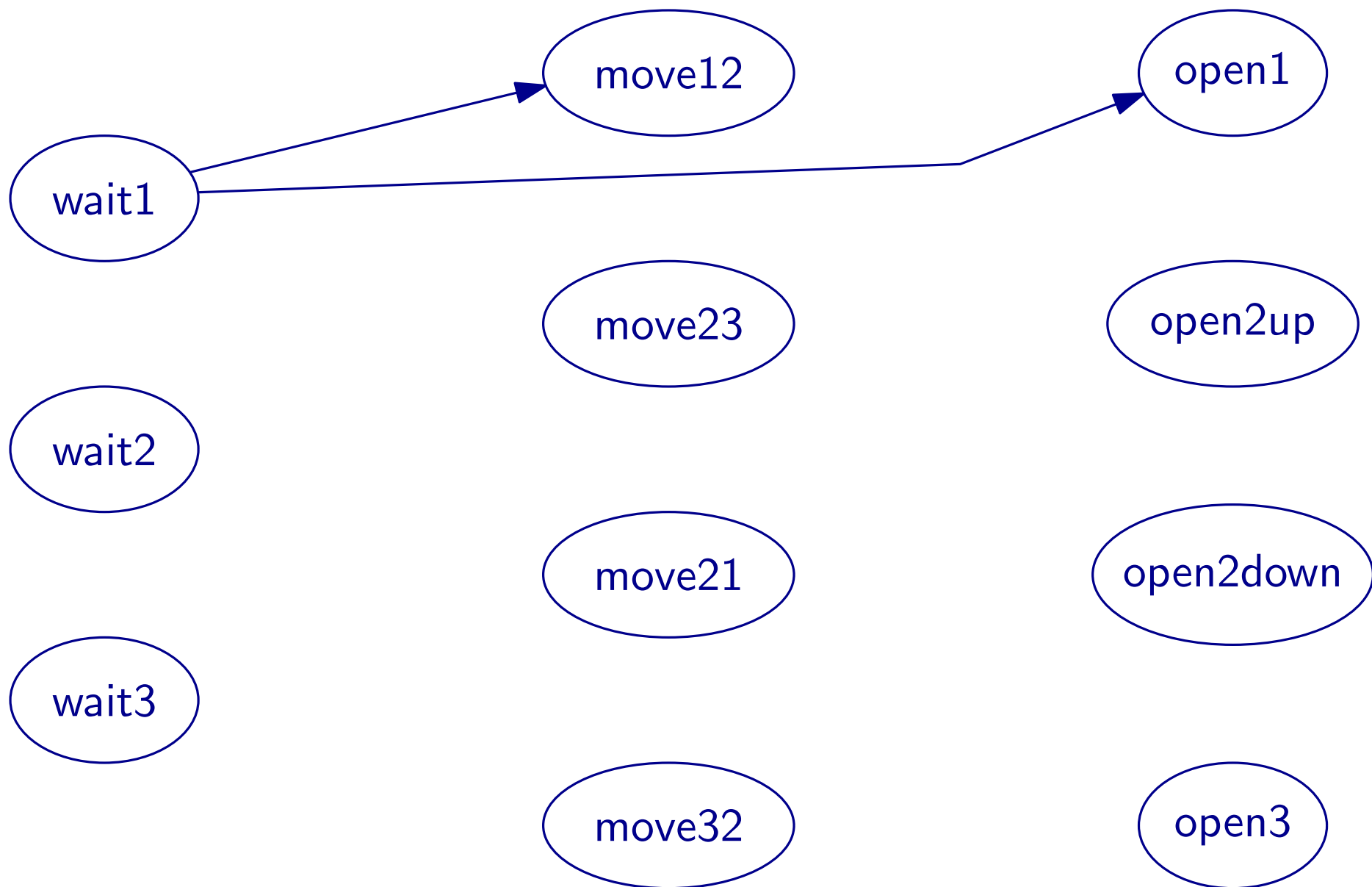
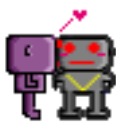
open2down

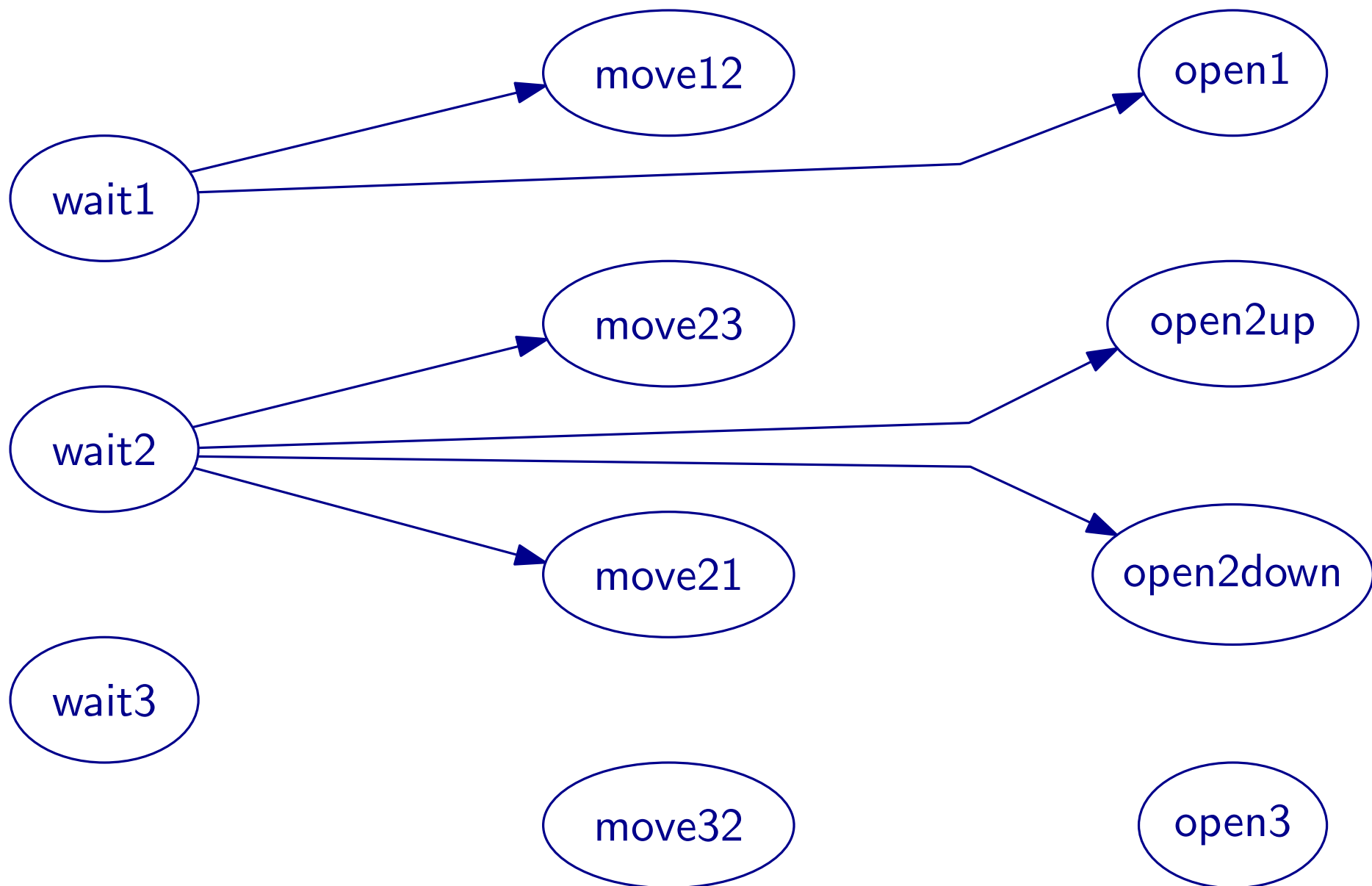
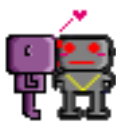
move32

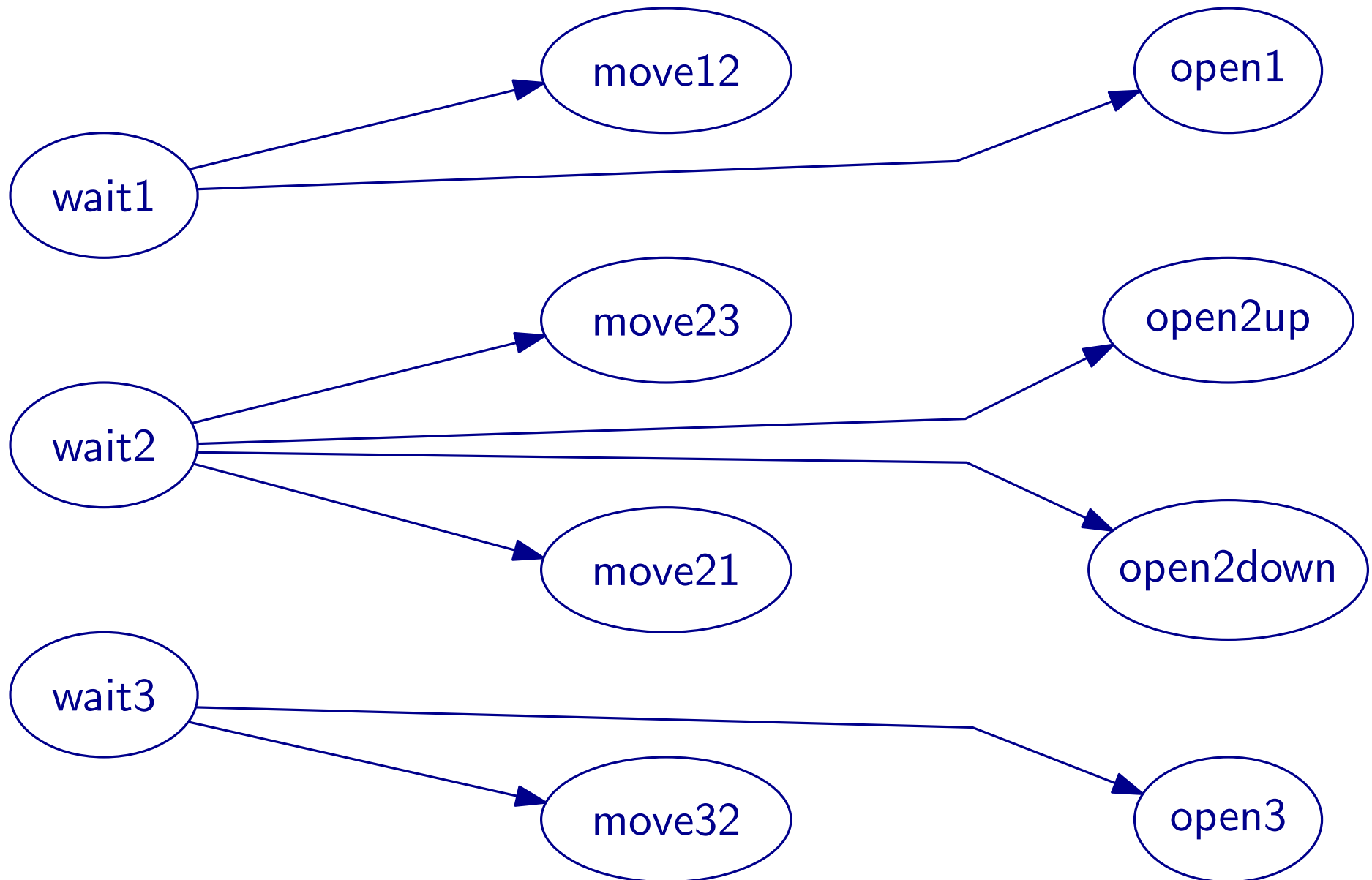
open3

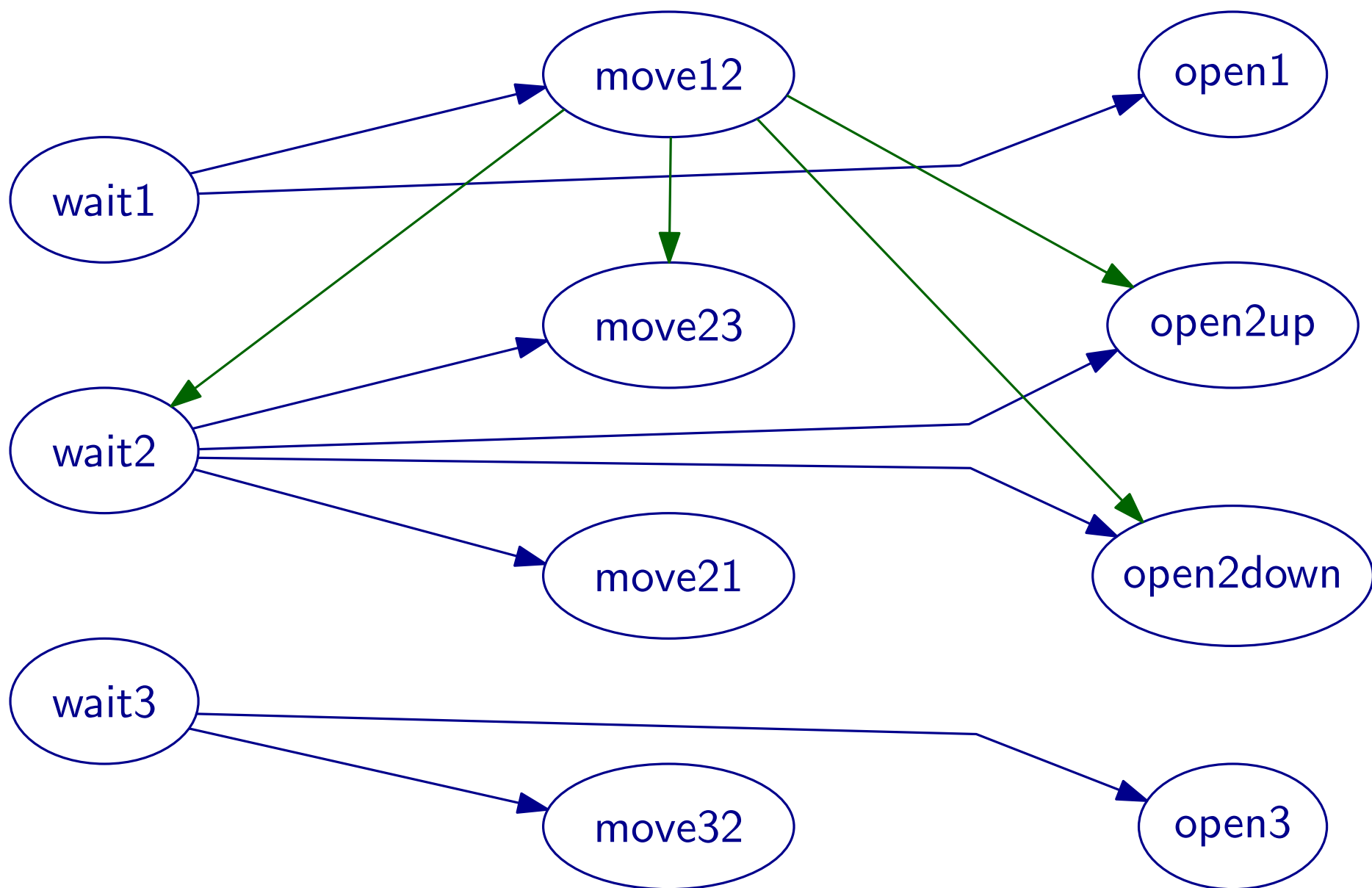
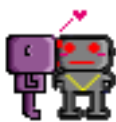
waiting at a floor	moving between floors	door open
--------------------	-----------------------	-----------

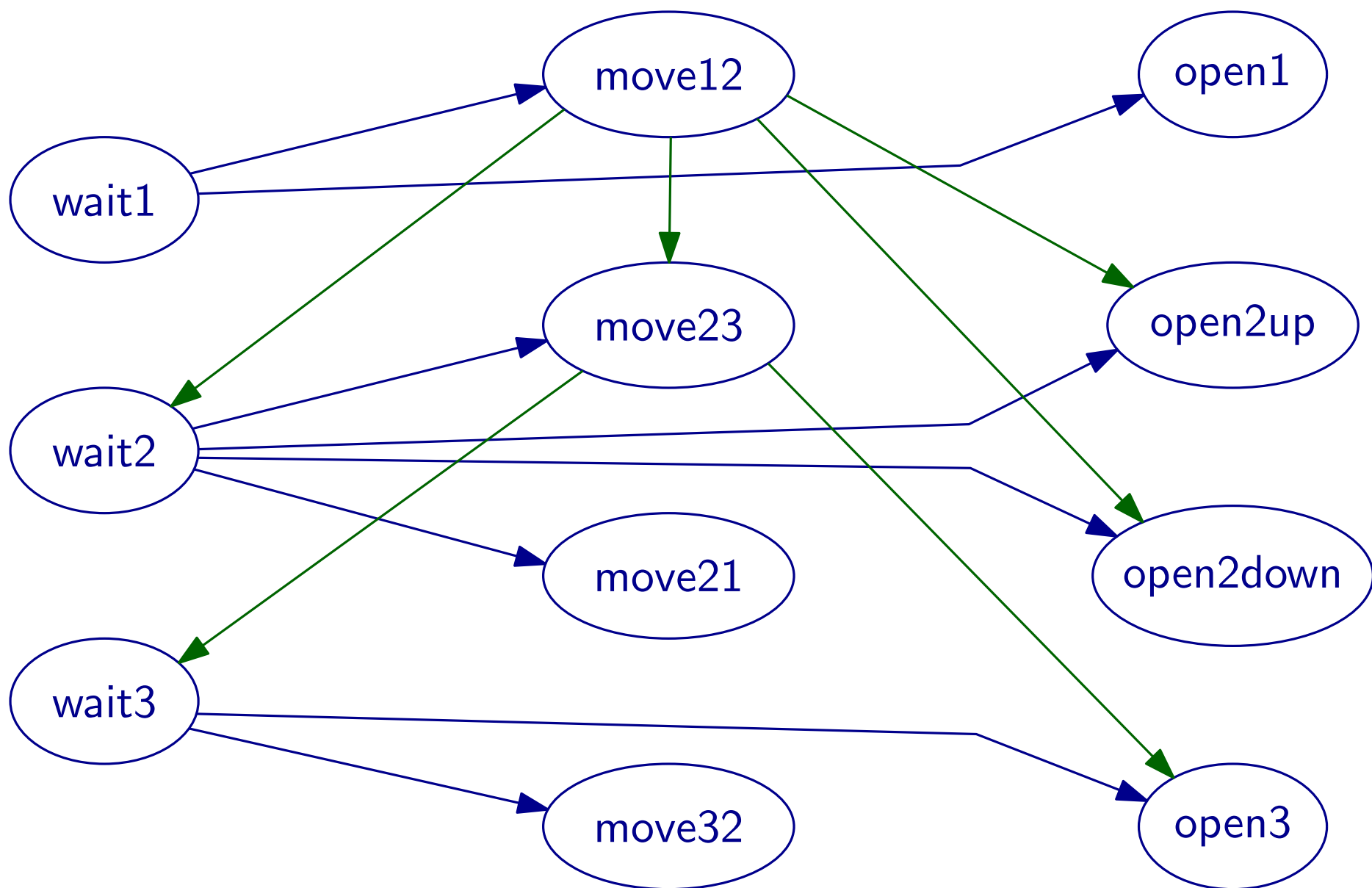
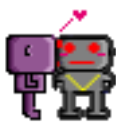




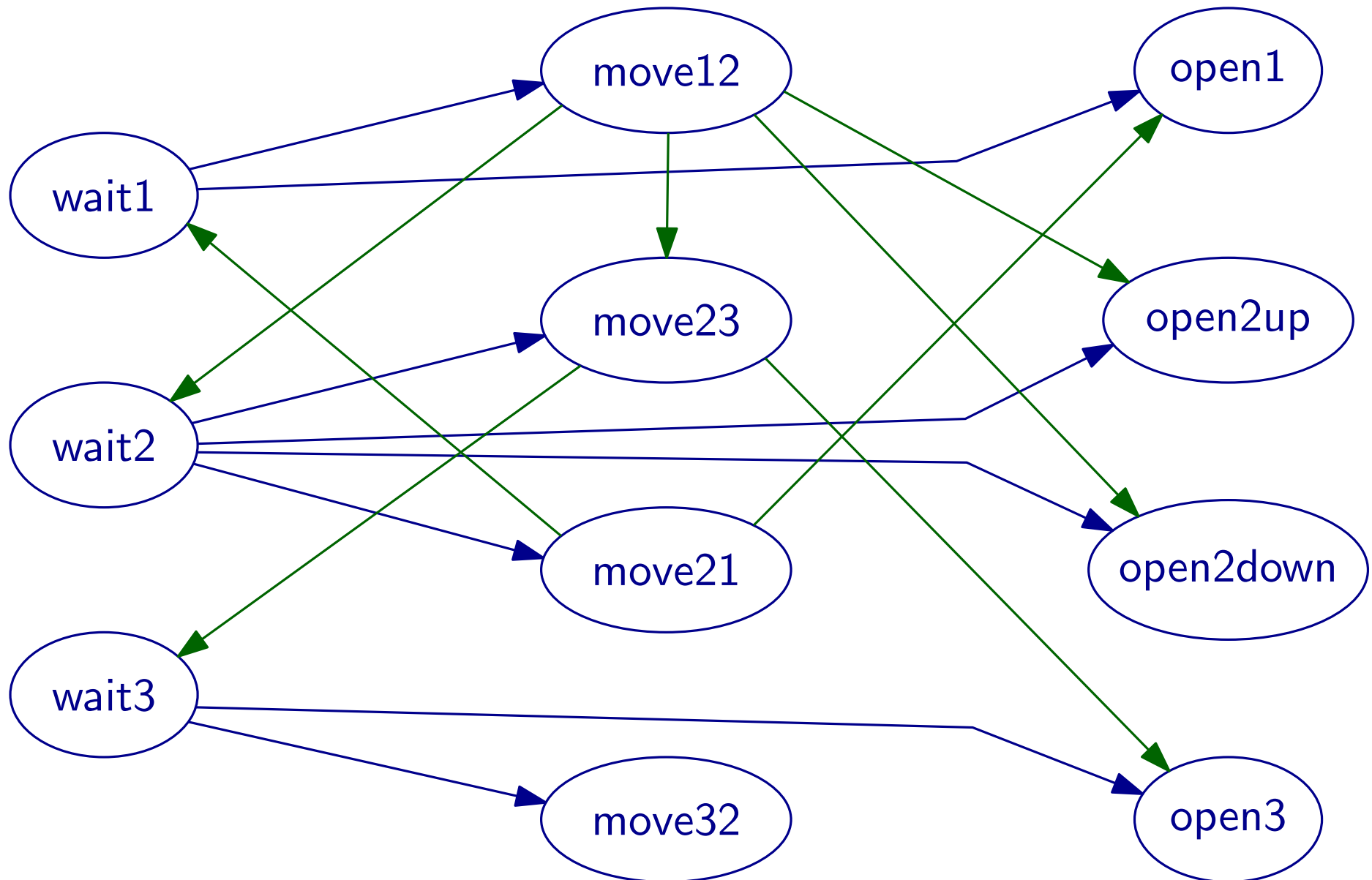


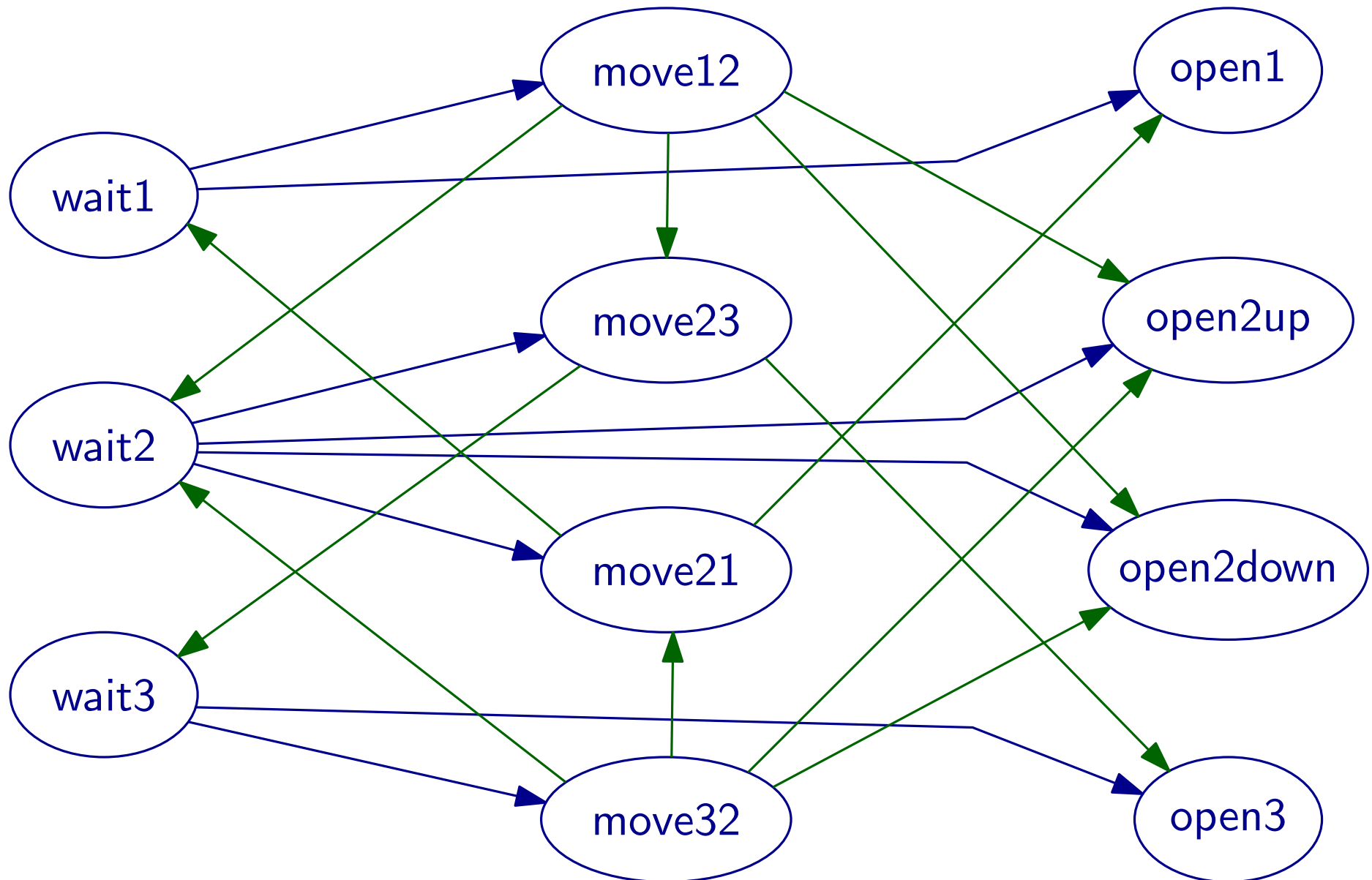


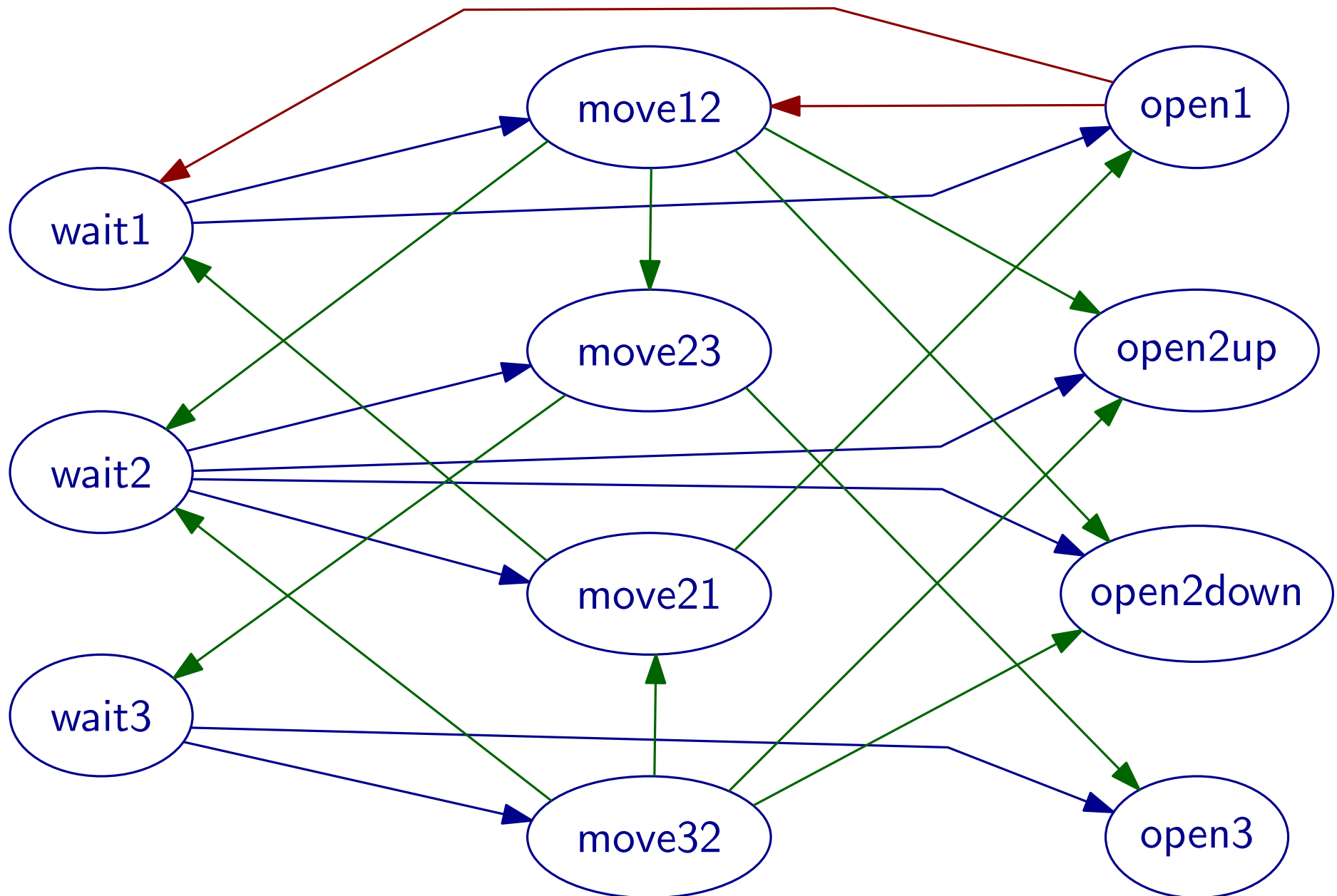


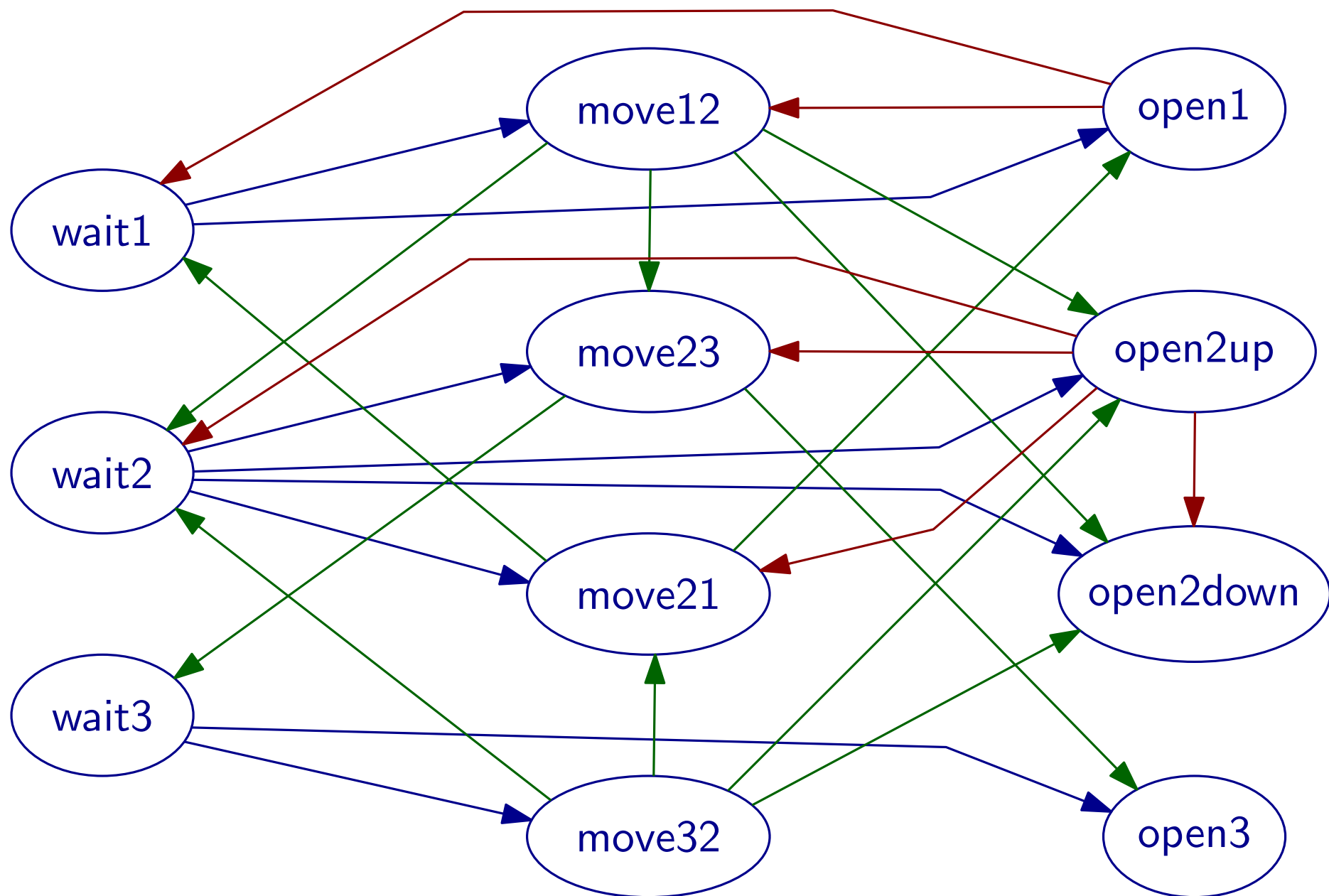
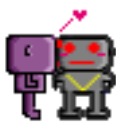


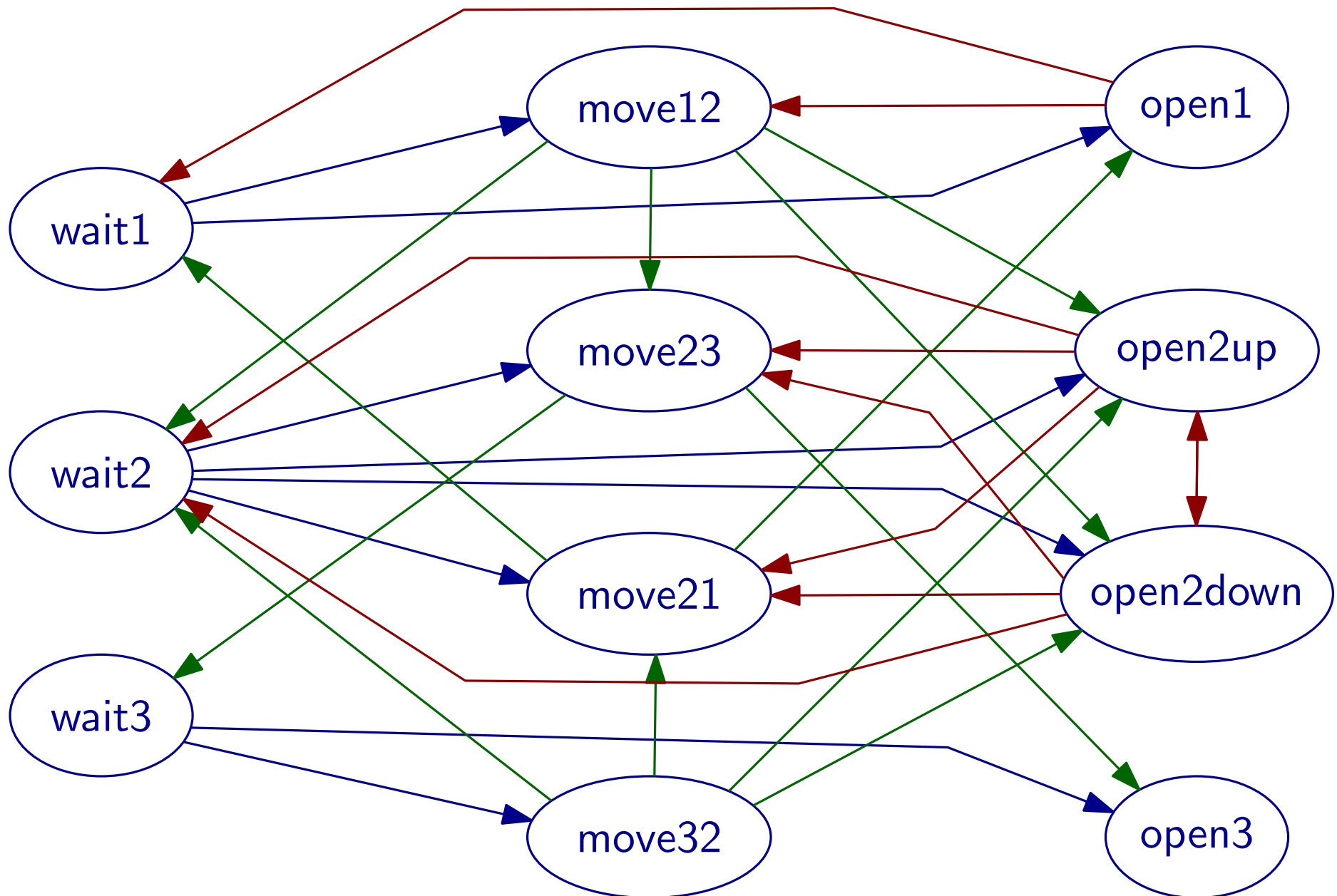
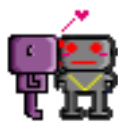


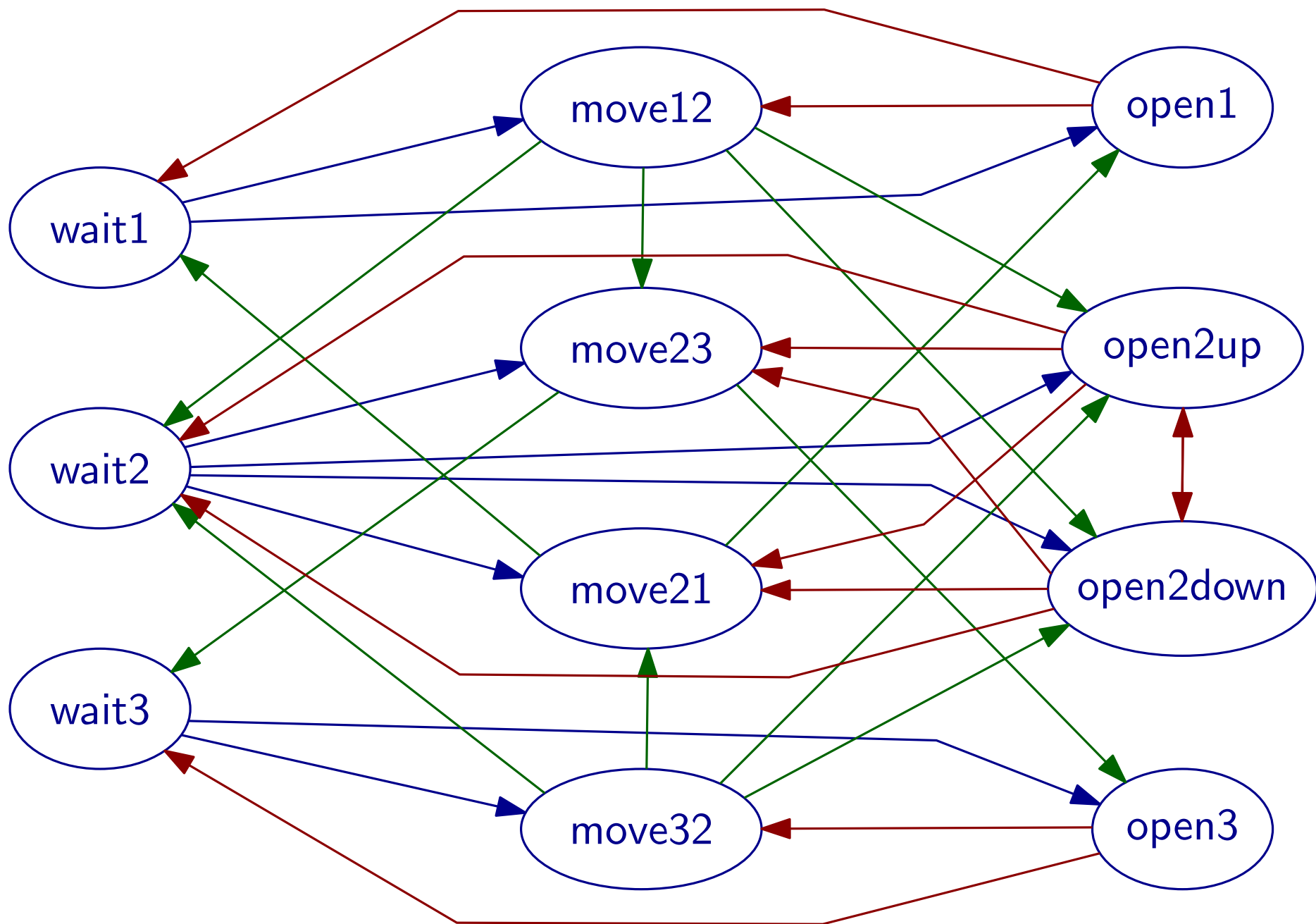
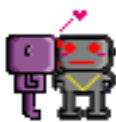








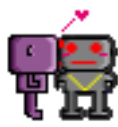






# Implementing the state diagram

We write a **handler** function for each state. This function will be called repeatedly as long as the elevator is in this state. When the state changes, we start calling the next handler function.



# Implementing the state diagram

We write a **handler** function for each state. This function will be called repeatedly as long as the elevator is in this state. When the state changes, we start calling the next handler function.

The main loop has become very simple:

```
handler = [ wait1, wait2, wait3,  
            move12, move21, move23, move32,  
            open1, open2up, open2down, open3 ]  
  
state = WAIT1  
while True:  
    handler[state]()
```





---

Each handler function must check all call buttons so that the elevator is responsive. It then checks if the state changes, and updates motor, lights, and doors correspondingly.



Each handler function must check all call buttons so that the elevator is responsive. It then checks if the state changes, and updates motor, lights, and doors correspondingly.

```
def move12():
    global state
    check_buttons()
    if not get_sensor(2): return
    if pending[2]:
        request_done(2)
        start_open(OPEN2UP)
    elif pending[3]:
        state = MOVE23
        return
    elif pending[0] or pending[1]:
        request_done(1)
        start_open(OPEN2DOWN)
    else:
        start_wait(WAIT2)
```