Vending Machine Model Checking - Spin

진정하 / 김우빈
MBC Lab
2015. 5. 18.

Vending Machine

- System
  - sell product
    - can drink : 500원 X 2ea , 600원, 700원, 1,000원
    - coffee : 200원 X 2ea, 300원 X 3ea
- Model Checkers
  - Spin

constraint-1

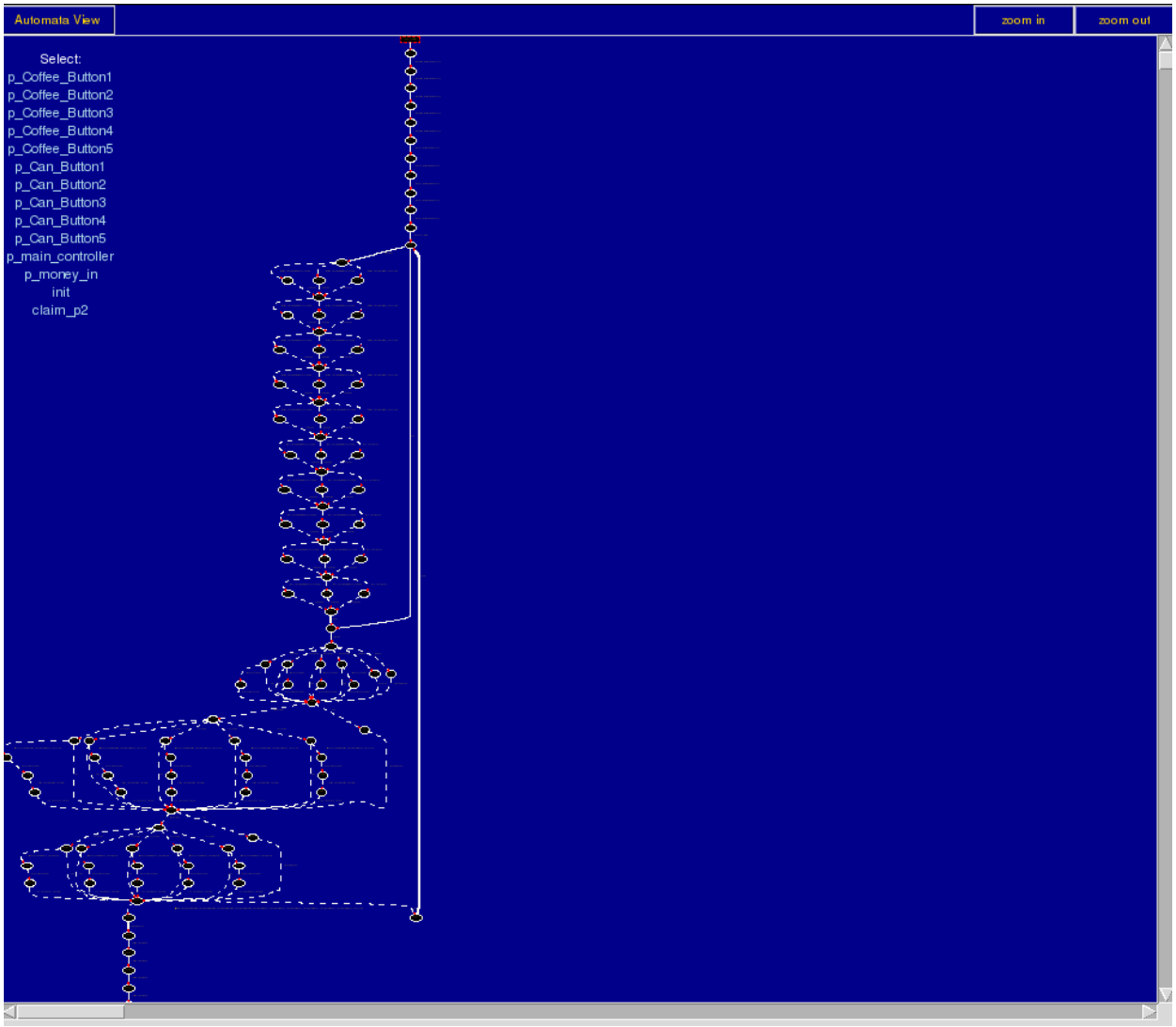
- coin max : 5,000원
  - 동전의 최소 단위 : 50원
  - 지폐사용 : 1,000원
- button
  - 동시에 같은 상품 종류 버튼을 누를 때 하나의 상품에 대해서만 동작함
  - 상품버튼을 누르면서 반환버튼을 동시에 누를 시 잔액이 refund 됨 (금액관련 우선순위 : coin_in>refund>coffee>can)
  - 버튼에 램프가 들어와 있지 않는 경우 눌러도 아무런 동작을 하지 않음

constraint-2

- 기타 제약사항
  - 전역변수 사용안함
  - 각 프로세스들은 메시지를 주고받아 실제 자판기와 같이 각 모듈들이 메모리를 공유하지 않고 메시지를 주고받는 형태로 구현

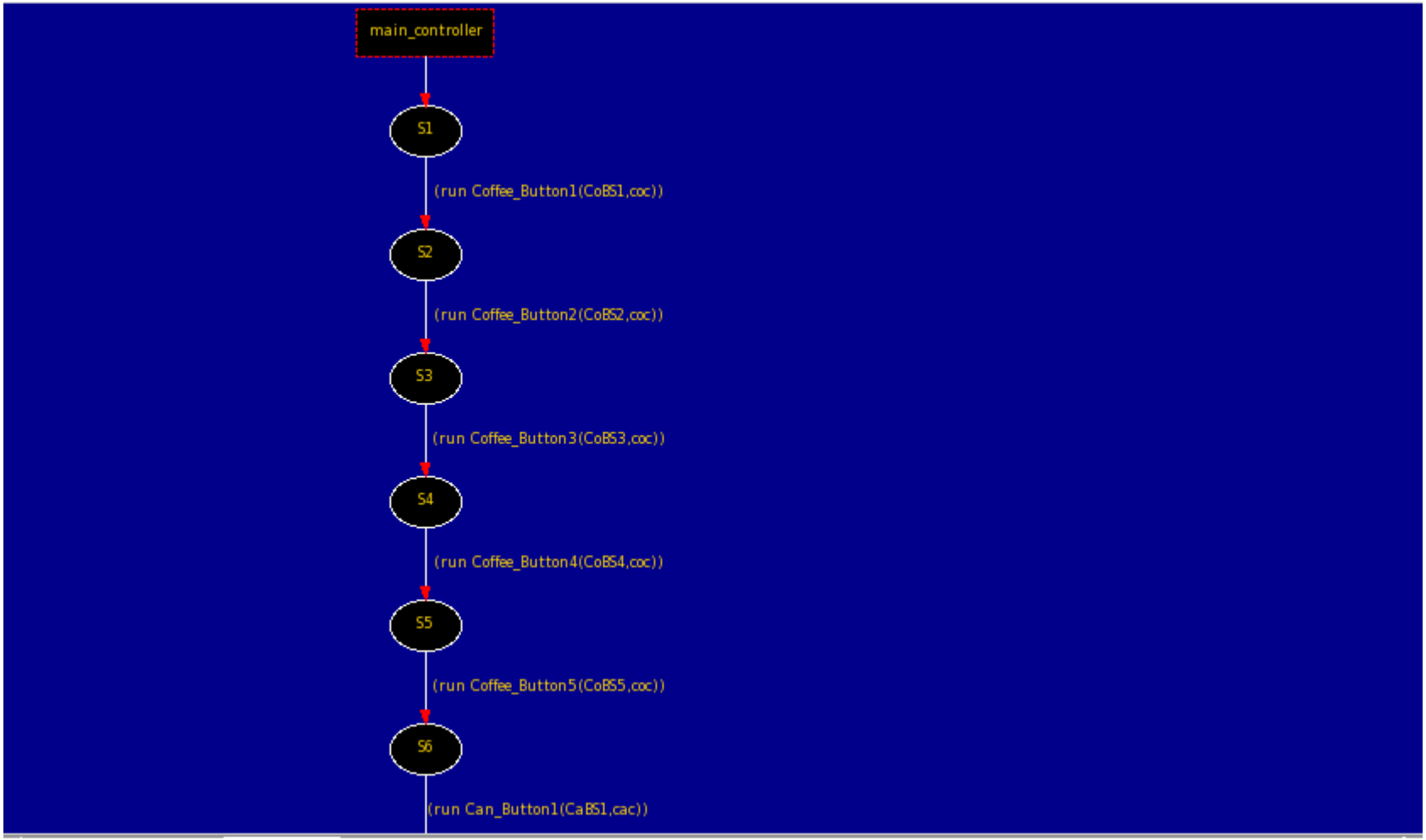
Automata

- Main_controller



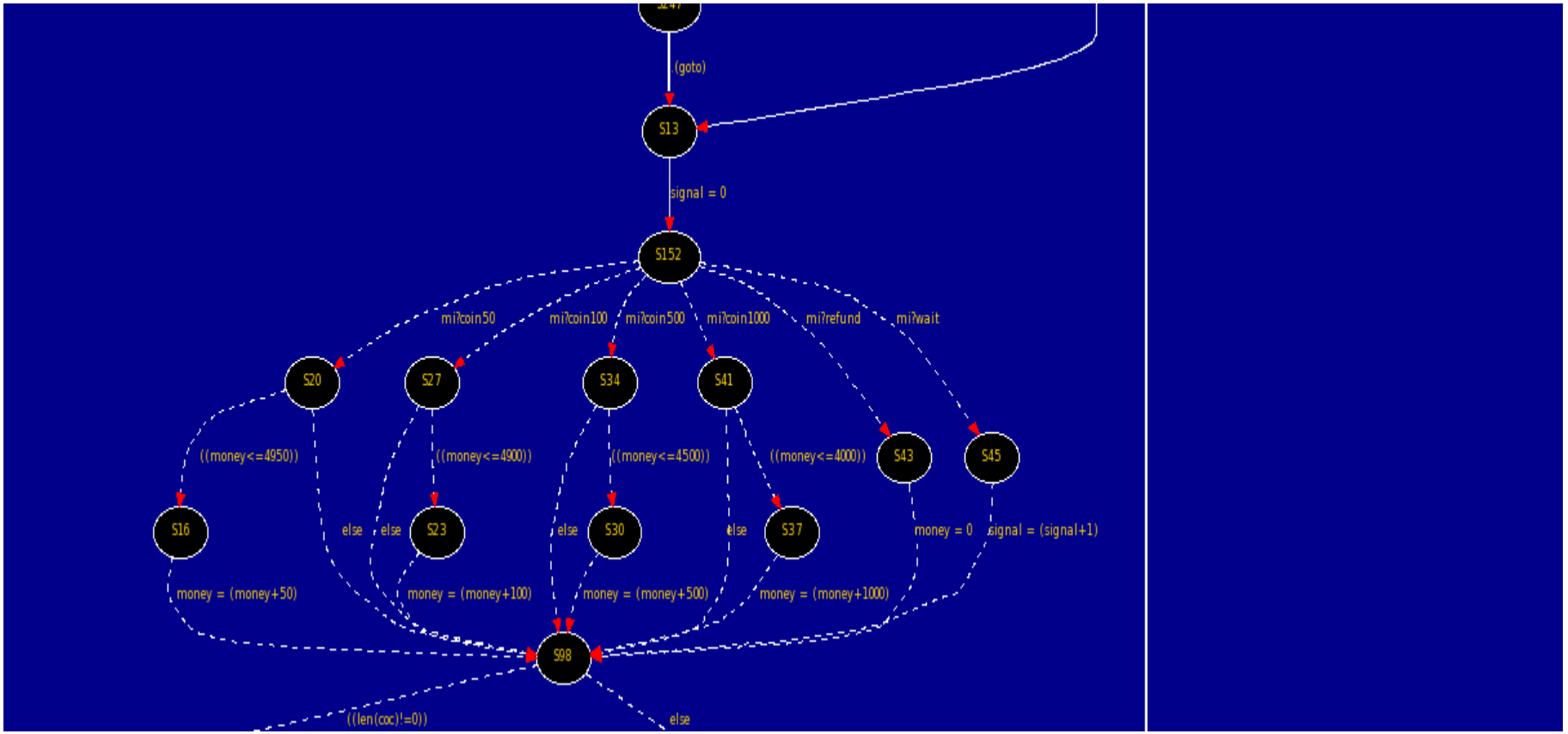
Automata

- 처음은 프로세스를 run으로 시작



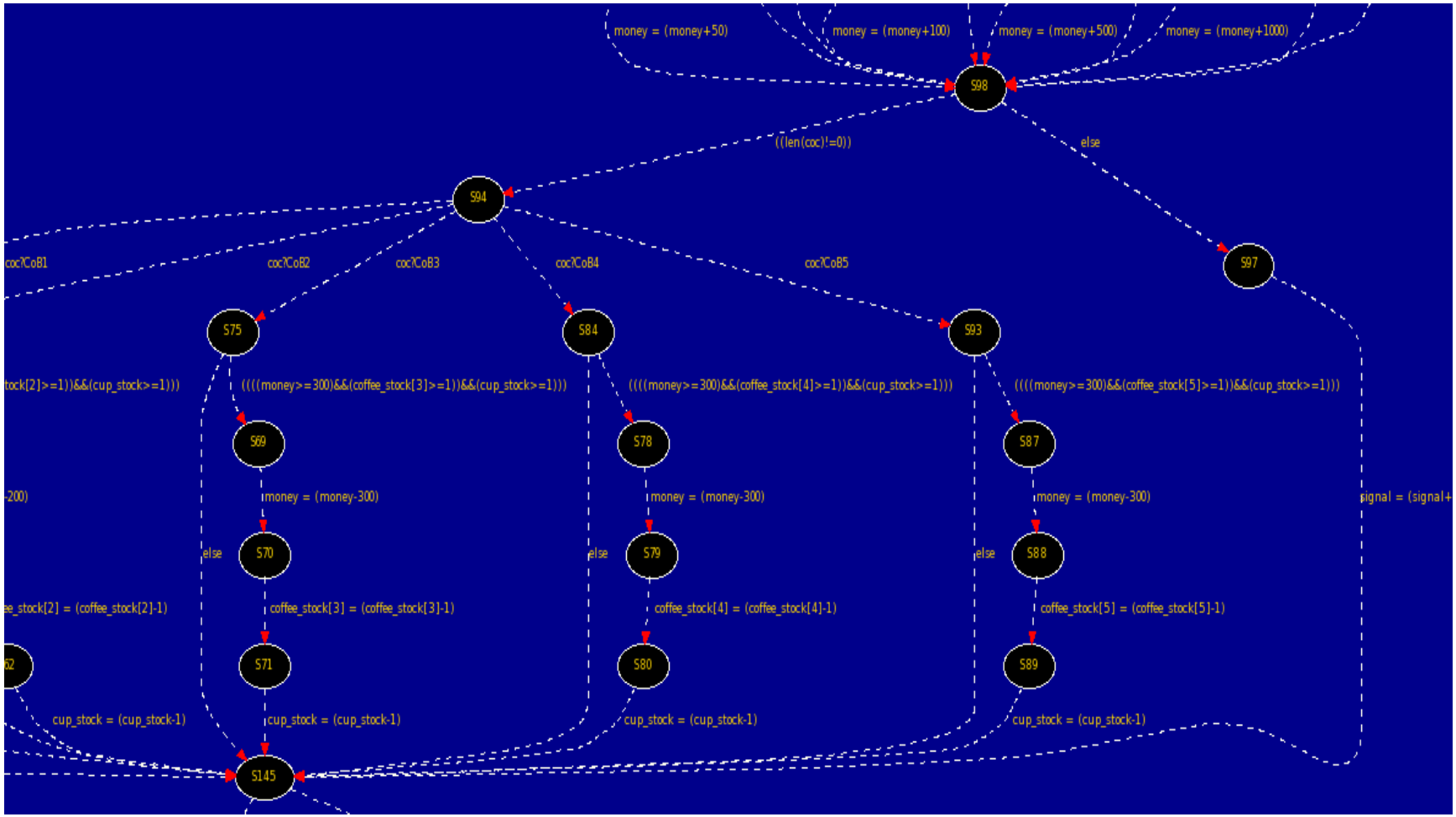
Automata

- 코인이 들어오거나 refund 되는지 검사



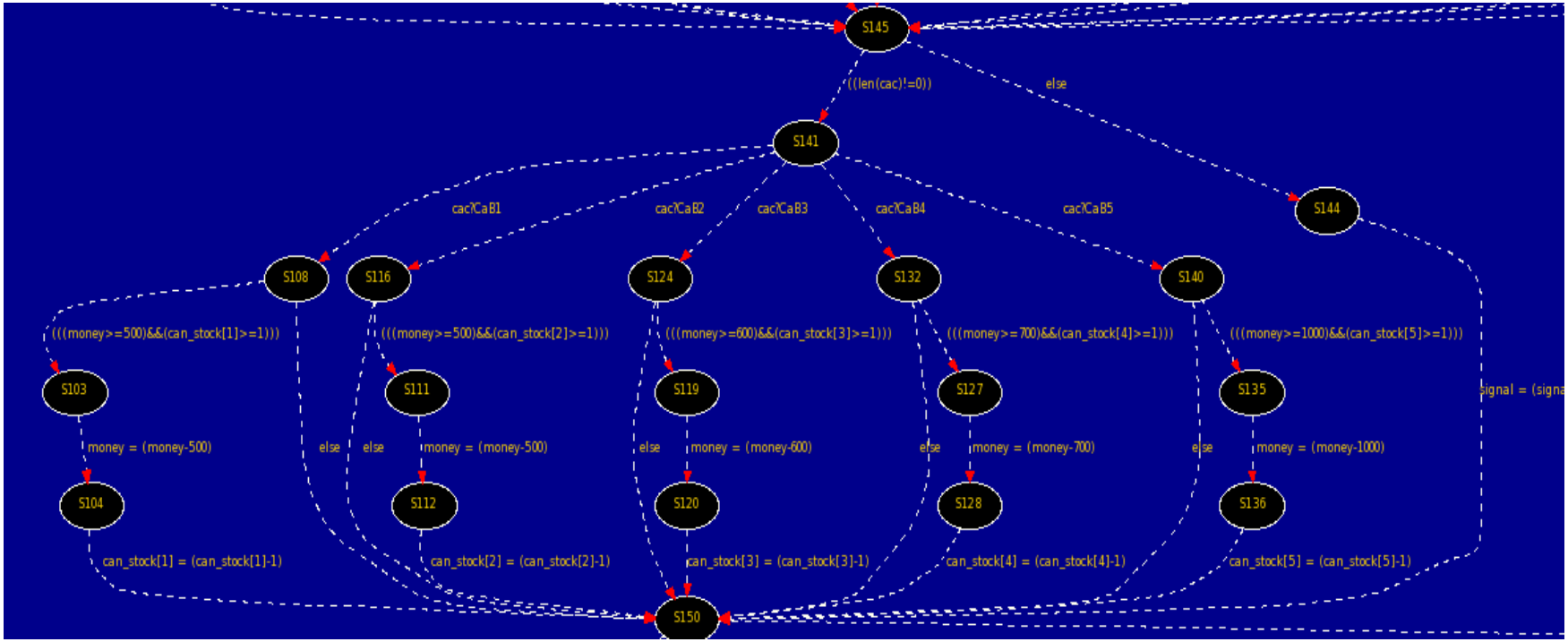
Automata

- Coffee버튼을 검사



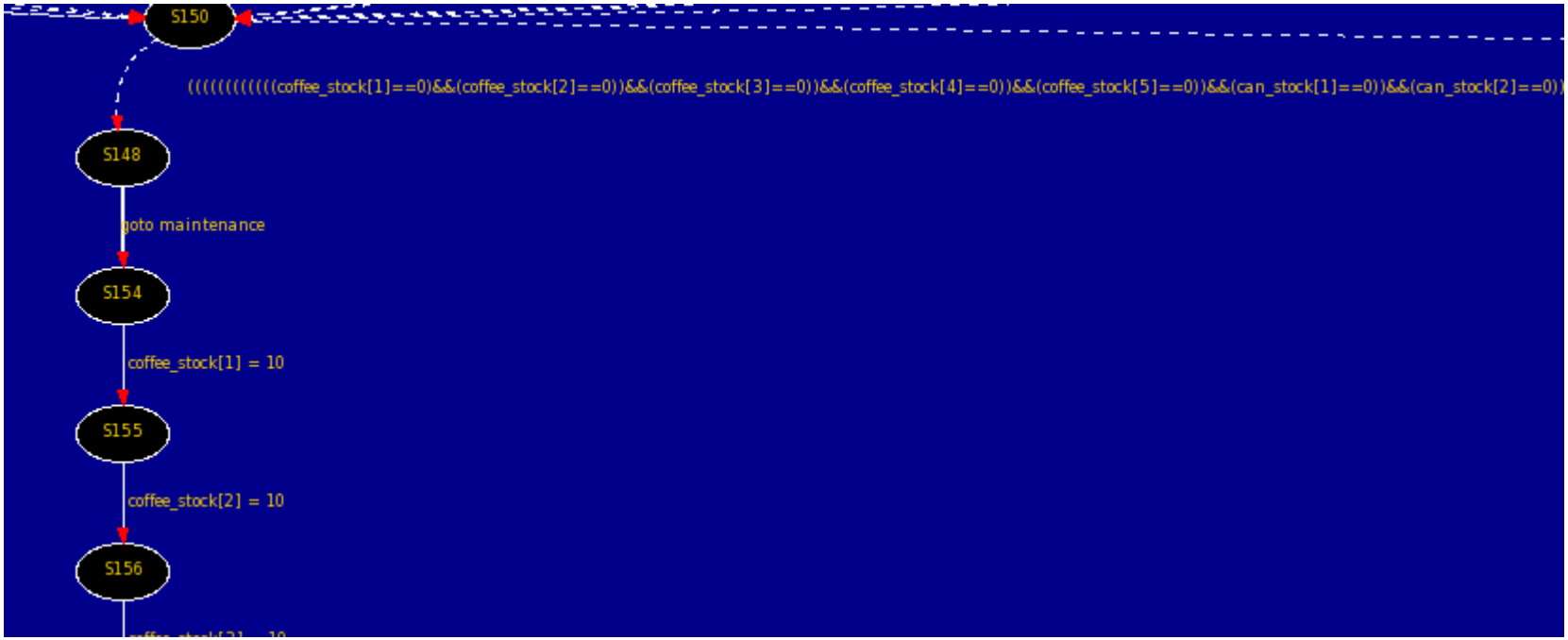
Automata

- Can버튼을 검사



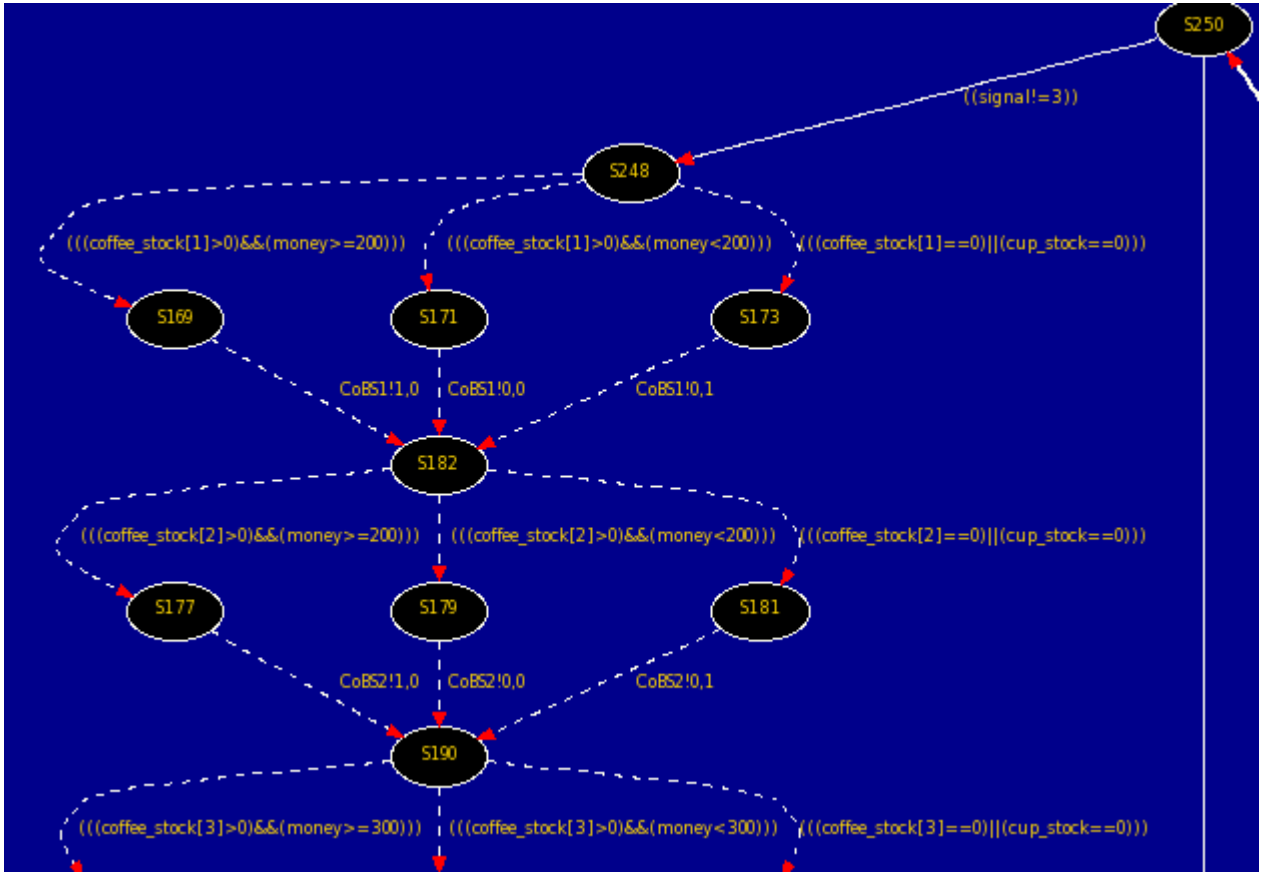
Automata

- 재고를 모두 검사하고 재고가 모두 없으면 maintenance로 들어가며 이외의 경우에는 위쪽으로 복귀



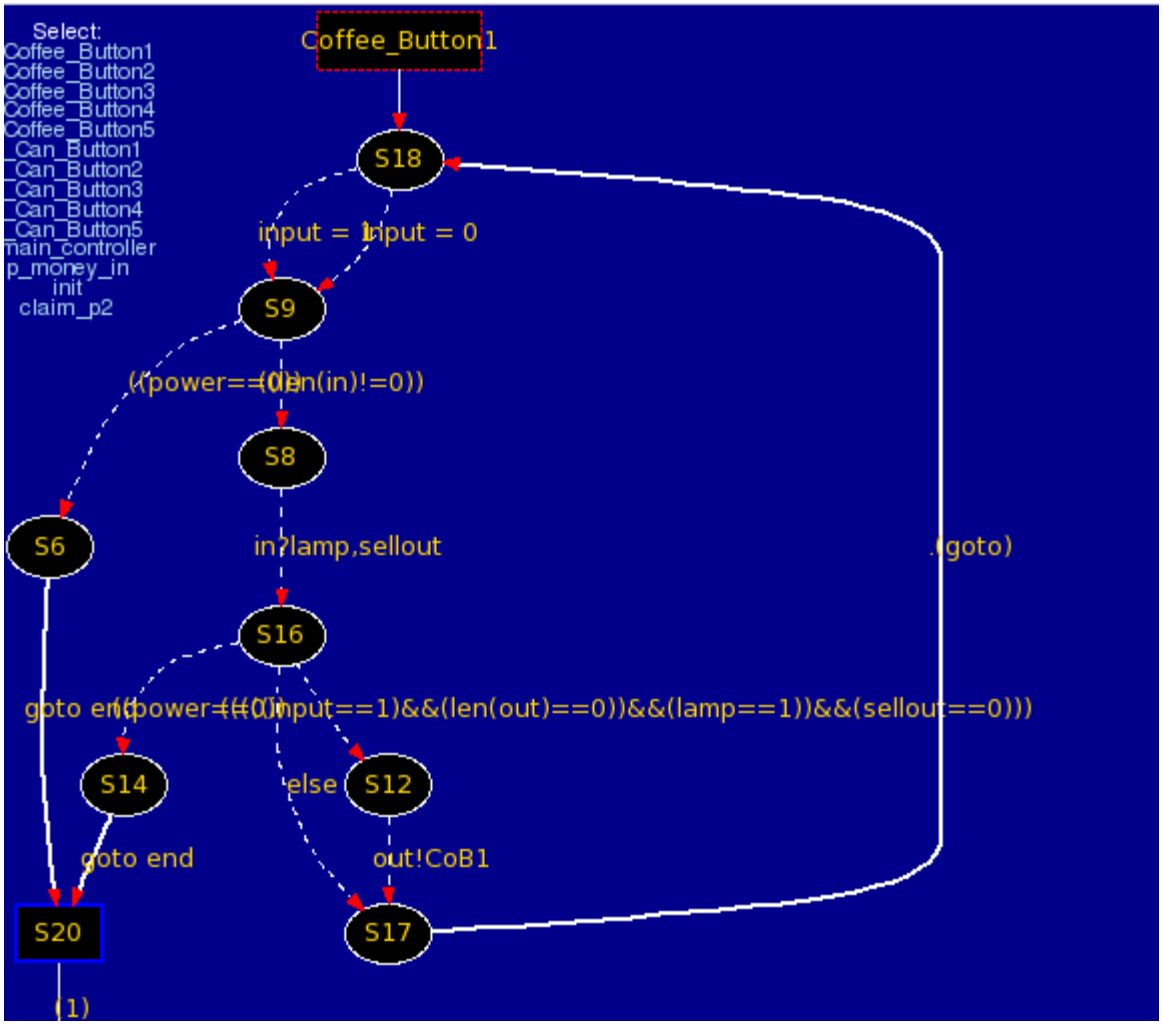
Automata

- Money나 재고에 변화가 생기면 버튼들에게 나타내야 할 상태를 보내줌



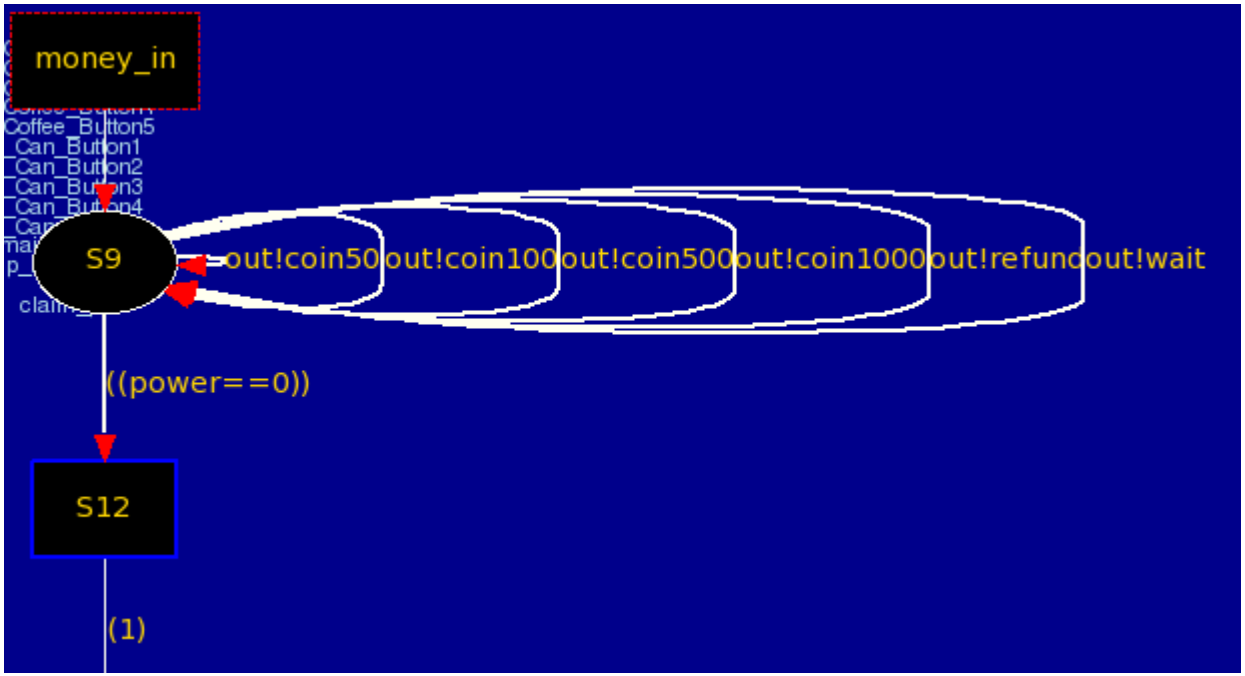
Automata

- 버튼들의 오토마타
  - 버튼은 따로 계산을 수행하지 않음
  - 누르는 입력과 컨트롤러에서 보내오는 메시지에 따라 상태가 바뀜
  - 버튼입력은 랜덤하게 생성



Automata

- 동전 투입구 오토마타
  - 동전투입구는 받은 코인에 대해 컨트롤러에 메시지만 전송함



Source code - 1

- 전역변수 power
- mtype으로 메시지들을 정의

```
bool power = 1;

mtype = { coin50, coin100, coin500, coin1000, refund, wait,
          CoB1, CoB2, CoB3, CoB4, CoB5,
          CaB1, CaB2, CaB3, CaB4, CaB5}
```

- init
  - 메인 컨트롤러만 불러옴
  - 나머지는 컨트롤러 안에서 생성됨

```
init
{
    run main_controller();
}
```

Source code - 2

- Main controller()
  - 각종 변수들을 생성
  - coc : coffee 버튼 입력
  - cac : can 버튼입력
  - Signal : 변동사항체크 변수
  - mi : 코인입력
  - CoBS등 : 컨트롤러에서 버튼에게 보내는 메시지

```
proctype main_controller()
{
    int money = 0;
    int cup_stock = 50;
    chan coc = [1] of { mtype };
    chan cac = [1] of { mtype };
    int coffee_stock[6] = {0,10,10,10,10,10};
    int can_stock[6] = {0,10,10,10,10,10};
    int signal = 0;
    chan mi = [0] of { mtype };
    chan CoBS1 = [1] of { bool ,bool };
    chan CoBS2 = [1] of { bool ,bool };
    chan CoBS3 = [1] of { bool ,bool };
    chan CoBS4 = [1] of { bool ,bool };
    chan CoBS5 = [1] of { bool ,bool };
    chan CaBS1 = [1] of { bool ,bool };
    chan CaBS2 = [1] of { bool ,bool };
    chan CaBS3 = [1] of { bool ,bool };
    chan CaBS4 = [1] of { bool ,bool };
    chan CaBS5 = [1] of { bool ,bool };
```

Source code - 3

- Main controller()
  - 버튼들과 투입구를 실행
  - 시작시 버튼들의 상태들을 세팅
  - 이후 돈이나 재고의 변동에 따라 이 부분으로 돌아오게 됨

```
    run Coffee_Button1(CoBS1,coc);
    run Coffee_Button2(CoBS2,coc);
    run Coffee_Button3(CoBS3,coc);
    run Coffee_Button4(CoBS4,coc);
    run Coffee_Button5(CoBS5,coc);
    run Can_Button1(CaBS1,cac);
    run Can_Button2(CaBS2,cac);
    run Can_Button3(CaBS3,cac);
    run Can_Button4(CaBS4,cac);
    run Can_Button5(CaBS5,cac);
    run money_in(mi);
start :
    goto buttonmanage
```

Source code - 4

- Main controller()
  - again : 변동이 없을 시 이곳으로 돌아옴
  - signal : 변동체크변수 초기화
  - 먼저 코인이 들어오거나 반환이 되는지를 체크하여 알맞은 작업을 수행

```
again :
    signal = 0;
atomic{
    if
    :: mi?coin50 -> atomic{if
                    :: (money <= 4950) -> money = money + 50
                    :: else
                    fi}
    :: mi?coin100 -> atomic{if
                    :: (money <= 4900) -> money = money + 100
                    :: else
                    fi}
    :: mi?coin500 -> atomic{if
                    :: (money <= 4500) -> money = money + 500
                    :: else
                    fi}
    :: mi?coin1000 -> atomic{if
                    :: (money <= 4000) -> money = money + 1000
                    :: else
                    fi}
    :: mi?refund -> money = 0
    :: mi?wait -> signal++
    fi
```

Source code - 5

- Main controller()
  - Coffee버튼에 대한 작업수행

```
if
:: (len(coc)!=0) ->if
    :: coc?CoB1 -> atomic{if
                    :: ((money >= 200) && (coffee_stock[1] >=1) && (cup_stock >= 1)) -> mone
                    :: else
                   fi}
    :: coc?CoB2 -> atomic{if
                    :: ((money >= 200) && (coffee_stock[2] >=1) && (cup_stock >= 1)) -> mone
                    :: else
                   fi}
    :: coc?CoB3 -> atomic{if
                    :: ((money >= 300) && (coffee_stock[3] >=1) && (cup_stock >= 1)) -> mone
                    :: else
                   fi}
    :: coc?CoB4 -> atomic{if
                    :: ((money >= 300) && (coffee_stock[4] >=1) && (cup_stock >= 1)) -> mone
                    :: else
                   fi}
    :: coc?CoB5 -> atomic{if
                    :: ((money >= 300) && (coffee_stock[5] >=1) && (cup_stock >= 1)) -> mone
                    :: else
                   fi}
    fi
:: else -> signal++
fi
```

Source code - 6

- Main controller()
  - Can버튼에 대한 작업수행

```
if
:: (len(cac)!=0) ->if
    :: cac?CaB1 -> atomic{if
                    :: ((money >= 500) && (can_stock[1] >=1)) -> money = money - 500
                    :: else
                   fi}
    :: cac?CaB2 -> atomic{if
                    :: ((money >= 500) && (can_stock[2] >=1)) -> money = money - 500
                    :: else
                   fi}
    :: cac?CaB3 -> atomic{if
                    :: ((money >= 600) && (can_stock[3] >=1)) -> money = money - 600
                    :: else
                   fi}
    :: cac?CaB4 -> atomic{if
                    :: ((money >= 700) && (can_stock[4] >=1)) -> money = money - 700
                    :: else
                   fi}
    :: cac?CaB5 -> atomic{if
                    :: ((money >= 1000) && (can_stock[5] >=1)) -> money = money - 10
                    :: else
                   fi}
    fi
:: else -> signal++
fi
```

Source code - 7

- Main controller()
  - 재고에 따라 maintenance나 다시 되돌아감

```
if
:: ((coffee_stock[1] == 0)&&(coffee_stock[2] == 0)&&(coffee_stock[3] == 0)&&(coffee_stock[4] == 0)&&(coffee_stock[5] == 0)&&
   (can_stock[1] == 0)&&(can_stock[2] == 0)&&(can_stock[3] == 0)&&(can_stock[4] == 0)&&(can_stock[5] == 0))&&(cup_stock == 0) -> goto maintenance
::else
fi
}
goto start
```

Source code - 8

- Main controller()
  - 버튼들의 상태를 체크해줌
  - 가격이나 재고에 변동이 없었을 시 바로 빠져 나감

```
buttonmanage :
if
:: (signal!=3) ->
    atomic
    {
        if
        ::((coffee_stock[1]>0)&&(money>=200)) -> CoBS1!1(0)
        ::((coffee_stock[1]>0)&&(money<200)) -> CoBS1!0(0)
        ::((coffee_stock[1]==0)||(cup_stock==0)) -> CoBS1!0(1)
        fi
        if
        ::((coffee_stock[2]>0)&&(money>=200)) -> CoBS2!1(0)
        ::((coffee_stock[2]>0)&&(money<200)) -> CoBS2!0(0)
        ::((coffee_stock[2]==0)||(cup_stock==0)) -> CoBS2!0(1)
        fi
        if
        ::((coffee_stock[3]>0)&&(money>=300)) -> CoBS3!1(0)
        ::((coffee_stock[3]>0)&&(money<300)) -> CoBS3!0(0)
        ::((coffee_stock[3]==0)||(cup_stock==0)) -> CoBS3!0(1)
```

Source code - 9

- Money in
  - 동전 투입구
  - 각 50원, 100원, 500원, 1000원, 반환, 기다림 등을 마음대로 선택하여 컨트롤러에 메시지를 보냄
  - 자판기의 전원이 꺼지면 프로세스 종료

```
proctype money_in(chan out)
{
    do
    :: out!coin50
    :: out!coin100
    :: out!coin500
    :: out!coin1000
    :: out!refund
    :: out!wait
    :: (power==0) -> goto end
    od
end :
}
```

Source code - 9

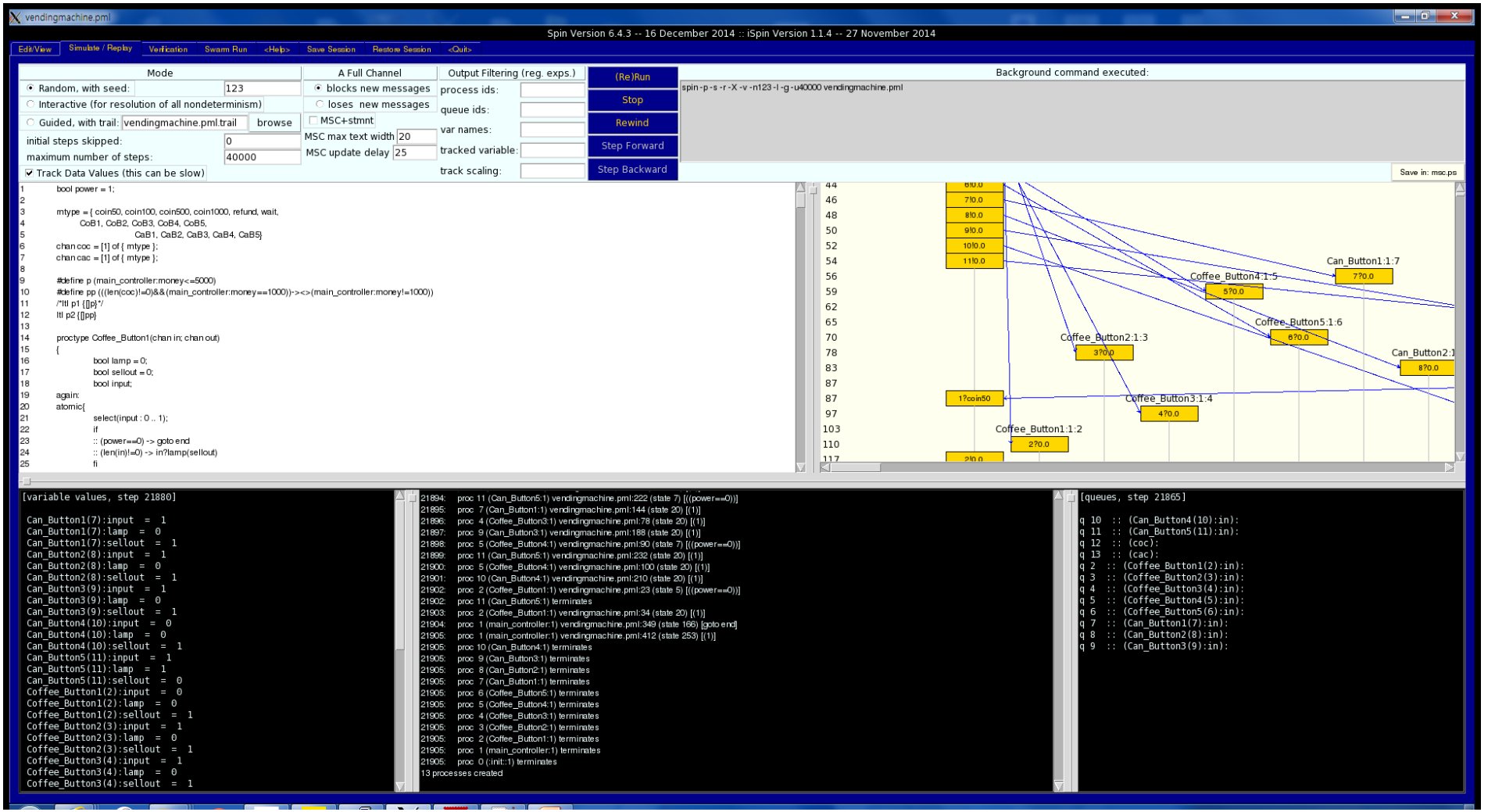
- Button
  - 버튼 들의 프로세스
  - 처음에 눌림 상태를 랜덤으로 생성

```
proctype Coffee_Button1(chan in; chan out)
{
    bool lamp = 0;
    bool sellout = 0;
    bool input;
again:
atomic{
    select(input : 0 .. 1);
    if
    :: (power==0) -> goto end
    :: (len(in)!=0) -> in?lamp(sellout)
    fi
    if
    :: ((input == 1) && (len(out)==0) && (lamp == 1) && (sellout == 0)) -> out!CoB1
    :: (power==0) -> goto end
    :: else
    fi
    }
    goto again
end:
}
```

메인 컨트롤러에서 보내준 메시지에 따라 상태를 바꾸어줌

버튼이 눌리고 조건을 만족하면 자판기에게 메시지 전달

vending machine simulation



vending machine verification - 1

- 검증에서의 문제
  - ltl 문에서 proctype내의 chan의 내부 내용을 불러오지 못함 -> coc?CoB1 등 사용불가
  - len(main_controller:coc)와 같이 함수에서 proctype 내부변수 사용불가 -> 전역변수로 하여야만 가능
  - 변수 등을 이용하여 검증시 다음단계는 chan으로 보내는 것이고 chan에서 받은 뒤 스테이트가 바뀌기 때문에 변수를 이용하여 검증불가
  - <>(eventually)를 사용하면 정말로 그 상태로 인해 그것이 되었는지 알 수 없음 -> 부정확한 검증

vending machine verification - 2

- ltl p1 {[] (main_controller:money<=5000)}
  - 항상 금액이 5000원 이하이다.
  - 자판기에 5000원 이상의 금액을 넣으면 반환
  - true

```
ltl p1: [] ((main_controller:money<=5000))

  choose which one with ./pan -a -N name (defaults to -N p1)
  or use e.g.: spin -search -ltl p1 vendingmachine.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DNOCLAIM -w -o pan pan.c
./pan -m20000000  -a

State-vector 268 byte, depth reached 3981385, errors: 0
  1160418 states, stored
   427927 states, matched
  1588345 transitions (= stored+matched)
  4290353 atomic steps
hash conflicts:      7828 (resolved)
```

vending machine verification - 3

- ltl p2 {[] (((len(coc)!=0)&&(main_controller:money==1000))->(main_controller:money!=1000))
  - Main controller의 coc를 전역으로 하여 검증
  - Money 변화의 영향을 받지 않도록 can, coin_in 제거 후 검증 돈은 초기에 한번 입력하여줌
  - 돈이 1000원 있고 Coffee버튼입력이 있으면 돈이 빠져나간다.
  - (돈이 있고 coffee버튼을 누르면 커피가 나온다)
  - true

```
ltl p2: [] ((! (((len(coc)!=0)) && ((main_controller:money==1000)))) || ((main_controller:money!=1000)))
```

```
/pan -m20000000  -a -N p2
```

```
State-vector 276 byte, depth reached 4968771, errors: 0
 1123942 states, stored
  400516 states, matched
 1524458 transitions (= stored+matched)
 4110556 atomic steps
hash conflicts:      6701 (resolved)
```

vending machine verification - 4

- ltl p2 {[] (((len(cac)!=0)&&(main_controller:money==1000))->(main_controller:money!=1000))
  - Main controller의 cac를 전역으로 하여 검증
  - Money 변화의 영향을 받지 않도록 coffee, coin_in 제거 후 검증 돈은 초기에 한번 입력하여줌
  - 돈이 1000원 있고 can버튼입력이 있으면 돈이 빠져나간다.
  - (돈이 있고 can버튼을 누르면 can이 나온다)
  - true

```
ltl p3: [] ((! (((len(cac)!=0)) && ((main_controller:money==1000)))) || ((main_controller:money!=1000)))
```

```
./pan -m20000000 -a -N p3
```

```
State-vector 276 byte, depth reached 4968771, errors: 0
  1123942 states, stored
   400516 states, matched
  1524458 transitions (= stored+matched)
  4110556 atomic steps
hash conflicts:      6969 (resolved)
```