coffee vending machine v.2

-NuSMV-

202172247 허윤아

DSLAB

MODULE coin

```
MODULE coin(cvm_state, menu_choice, cur_coin)
VAR
    coin_in : {0, 1, 5, 10};
    menu_val : {0, 1, 2, 3, 4, 5};
ASSIGN
    init(coin_in) := 0;
    next(coin_in) :=
        case
            cvm_state = on : {0, 1, 5, 10};
            TRUE : 0;
        esac;
    init(menu_val) := 0;
    next(menu_val) :=
        case
            cvm_state = on & menu_choice = espresso : 1;
            cvm_state = on & menu_choice = black : 2;
            cvm_state = on & menu_choice = milk : 3;
            cvm_state = on & menu_choice = latte : 4;
            cvm_state = on & menu_choice = mocha : 5;
            TRUE : 0;
        esac;
DEFINE
    main.current_coin := cur_coin + coin_in - menu_val;
```

MODULE stock

```
MODULE stock(machine_state, choose_menu, cur_coin2)
VAR
    current_espresso : 0..10;
    current_water : 0..50;
    current_milk : 0..50;
    current_choco : 0..10;

    out_of_service : process alarm(current_espresso, current_water, current_milk, current_choco);
ASSIGN
    init(current_espresso) := 10;
    next(current_espresso) :=
        case
            machine_state = on & choose_menu = espresso & cur_coin2 >= 1 & current_espresso >= 2 : current_espresso - 2;
            machine_state = on & choose_menu = black & cur_coin2 >= 2 & current_espresso >= 2 : current_espresso - 2;
            machine_state = on & choose_menu = latte & cur_coin2 >= 4 & current_espresso >= 2 : current_espresso - 2;
            machine_state = on & choose_menu = mocha & cur_coin2 >= 5 & current_espresso >= 2 : current_espresso - 2;
            machine_state = filling : 10;
            TRUE : current_espresso;
        esac;
    init(current_water) := 50;
    next(current_water) :=
        case
            machine_state = on & choose_menu = black & cur_coin2 >= 2 & current_water >= 8 : current_water - 8;
            machine_state = filling : 50;
            TRUE : current_water;
        esac;
    init(current_milk) := 50;
    next(current_milk) :=
        case
            machine_state = on & choose_menu = milk & cur_coin2 >= 3 & current_milk >= 10 : current_milk - 10;
            machine_state = on & choose_menu = latte & cur_coin2 >= 4 & current_milk >= 8 : current_milk - 8;
            machine_state = on & choose_menu = mocha & cur_coin2 >= 5 & current_milk >= 7 : current_milk - 7;
            machine_state = filling : 50;
            TRUE : current_milk;
        esac;
    init(current_choco) := 10;
    next(current_choco) :=
        case
            machine_state = on & choose_menu = mocha & cur_coin2 >= 5 & current_choco >= 1 : current_choco - 1;
            machine_state = filling : 10;
            TRUE : current_choco;
        esac;
```

MODULE alarm

```
MODULE alarm(cur_espresso, cur_water, cur_milk, cur_choco)
VAR
    espresso_out : boolean;
    black_out : boolean;
    milk_out : boolean;
    latte_out : boolean;
    mocha_out : boolean;
ASSIGN
    init(espresso_out) := FALSE;
    next(espresso_out) :=
        case
            cur_espresso < 2 : TRUE;
            TRUE : FALSE;
        esac;
    init(black_out) := FALSE;
    next(black_out) :=
        case
            cur_espresso < 2 | cur_water < 8 : TRUE;
            TRUE : FALSE;
        esac;
    init(milk_out) := FALSE;
    next(milk_out) :=
        case
            cur_milk < 10 : TRUE;
            TRUE : FALSE;
        esac;
    init(latte_out) := FALSE;
    next(latte_out) :=
        case
            cur_espresso < 2 | cur_milk < 8 : TRUE;
            TRUE : FALSE;
        esac;
    init(mocha_out) := FALSE;
    next(mocha_out) :=
        case
            cur_espresso < 2 | cur_choco < 1 | cur_milk < 7 : TRUE;
            TRUE : FALSE;
        esac;
```

MODULE main

```
MODULE main
VAR
    on_command : boolean;
    refund_command : boolean;
    fill_command : boolean;

    state : {off, on, refund, filling};
    menu : {none, espresso, black, milk, latte, mocha};

    check_coin : process coin(state, menu, current_coin);
    make_menu : process stock(state, menu, current_coin);

    current_coin : 0..50;

ASSIGN
    init(on_command) := FALSE;
    init(refund_command) := FALSE;
    init(state) := off;
    next(state) :=
        case
            on_command = TRUE & state = off : on;
            state = on & refund_command = TRUE : refund;
            state = refund : on;
            fill_command = TRUE : filling;
            on_command = FALSE & state = on : off;
            TRUE : state;
        esac;
    init(menu) := none;
    next(menu) :=
        case
            state = on : {espresso, black, milk, latte, mocha};
            TRUE : none;
        esac;
    init(current_coin) := 0;
    next(current_coin) :=
        case
            state = off | state = filling | state = on : current_coin;
            state = refund : 0;
        esac;
```

CTL Properties

Property	Description	Expected Output	Result
SPEC AG EX TRUE	deadlock freeness	TRUE	TRUE
SPEC AG(state = off & on_command -> AX(state = on))	전원을 켜면 on 상태에 돌입한다	TRUE	FALSE
SPEC AG(state = on & !on_command -> AX(state = off))	전원을 끄면 off 상태에 돌입한다	TRUE	FALSE
SPEC EF(current_coin = 5 & refund_command & menu = mocha)	환불 버튼과 모카 버튼을 동시에 선택할 수 있다	FALSE	FALSE
SPEC EF(menu = mocha & menu = black)	한 번에 두 개 이상의 음료를 선택할 수 있다	FALSE	FALSE
SPEC AG(current_coin = 0 & check_coin.coin_in = 0 -> AX(current_coin = 0))	돈을 넣지 않으면 돈이 추가되지 않는다	TRUE	TRUE
SPEC AG(current_coin = 50 & check_coin.coin_in = 10 -> AX(current_coin = 50))	돈을 넣어도 5000원 이상으로는 돈을 받지 않는다	TRUE	TRUE
SPEC AG(current_coin = 0 & menu = espresso & make_menu.current_espresso = 4 -> AX(make_menu.current_espresso = 4))	돈을 넣지 않은 상태에서 메뉴를 선택하면 음료가 제조되지 않는다	TRUE	TRUE
SPEC AG(current_coin = 10 & menu = black & make_menu.current_espresso = 10 & make_menu.current_water = 50 -> AX(current_coin = 8 & make_menu.current_espresso = 8 & make_menu.current_water = 42))	돈을 넣은 상태에서 메뉴를 선택하면 선택한 메뉴만큼 돈과 재료가 차감되며 음료가 제조된다	TRUE	TRUE
SPEC AG(current_coin = 8 & refund_command -> AX(current_coin = 0))	돈이 들어있는 상태에서 환불 버튼을 누르면 돈이 환불된다	TRUE	TRUE
SPEC AG(state = off & menu = espresso & make_menu.current_espresso = 10 -> AX(make_menu.current_espresso = 10))	전원이 꺼져 있는 상태에서는 메뉴 버튼을 눌러도 음료가 제조되지 않는다	TRUE	TRUE
SPEC AG(current_coin = 3 & menu = mocha & make_menu.current_espresso = 6 & make_menu.current_choco = 10 & make_menu.current_milk = 20 -> AX(make_menu.current_espresso = 6 & make_menu.current_choco = 10 & make_menu.current_milk = 20))	재료가 충분하더라도 돈이 충분하지 않으면 음료가 만들어지지 않는다	TRUE	TRUE
SPEC AG(current_coin = 10 & menu = mocha & make_menu.current_milk = 3 -> AX(make_menu.current_milk = 3))	재료가 충분하지 않으면 음료는 만들어지지 않는다	TRUE	TRUE
SPEC AG(make_menu.current_milk = 5 -> AX(make_menu.out_of_service.milk_out & make_menu.out_of_service.latte_out & make_menu.out_of_service.mocha_out))	재료가 다 떨어진 경우 메뉴가 없음을 알리는 alarm이 표시된다	TRUE	TRUE
SPEC AG(make_menu.out_of_service.milk_out & menu = milk & make_menu.current_milk = 3 -> AX(make_menu.current_milk = 3))	재료가 충분하지 않으면 음료는 만들어지지 않는다	TRUE	TRUE

CTL Properties – results (wrong)

```
-- specification AG ((state = off & on_command) -> AX state = on)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    on_command = FALSE
    refund_command = FALSE
    fill_command = FALSE
    state = off
    menu = none
    check_coin.coin_in = 0
    check_coin.menu_val = 0
    make_menu.current_espresso = 10
    make_menu.current_water = 50
    make_menu.current_milk = 50
    make_menu.current_choco = 10
    make_menu.out_of_service.espresso_out = FALSE
    make_menu.out_of_service.black_out = FALSE
    make_menu.out_of_service.milk_out = FALSE
    make_menu.out_of_service.latte_out = FALSE
    make_menu.out_of_service.mocha_out = FALSE
    current_coin = 0
    check_coin.main.current_coin = 0
  -> Input: 1.2 <-
    _process_selector_ = main
    running = TRUE
    make_menu.running = FALSE
    make_menu.out_of_service.running = FALSE
    check_coin.running = FALSE
  -> State: 1.2 <-
    on_command = TRUE
  -> Input: 1.3 <-
    _process_selector_ = make_menu
    running = FALSE
    make_menu.running = TRUE
  -> State: 1.3 <-
    on_command = FALSE
```

```
-- specification AG ((state = on & !on_command) -> AX state = off)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    on_command = FALSE
    refund_command = FALSE
    fill_command = FALSE
    state = off
    menu = none
    check_coin.coin_in = 0
    check_coin.menu_val = 0
    make_menu.current_espresso = 10
    make_menu.current_water = 50
    make_menu.current_milk = 50
    make_menu.current_choco = 10
    make_menu.out_of_service.espresso_out = FALSE
    make_menu.out_of_service.black_out = FALSE
    make_menu.out_of_service.milk_out = FALSE
    make_menu.out_of_service.latte_out = FALSE
    make_menu.out_of_service.mocha_out = FALSE
    current_coin = 0
    check_coin.main.current_coin = 0
  -> Input: 2.2 <-
    _process_selector_ = main
    running = TRUE
    make_menu.running = FALSE
    make_menu.out_of_service.running = FALSE
    check_coin.running = FALSE
  -> State: 2.2 <-
    on_command = TRUE
  -> Input: 2.3 <-
  -- Loop starts here
  -> State: 2.3 <-
    on_command = FALSE
    state = on
  -> Input: 2.4 <-
    _process_selector_ = make_menu
    running = FALSE
    make_menu.running = TRUE
  -> State: 2.4 <-
```

Proposal

- Platoon system
  - 차량 간에 서로 통신을 주고받으며 군집주행
  - 과제에서 해당 시스템을 활용할 예정이나, 시스템에 대한 명확한 SRS나 모델 등이 없으므로, 필요성을 느낌
  - 또한, 준비 중인 논문에서 근거 자료로 활용하려면 해당 시스템에 대한 V&V와 그 결과가 필요함

- 모델링의 목적
  - Platoon system에 관한 대략적인 SRS을 기반으로 하여 모델링을 진행
  - SRS의 digital twin을 만들어 모순이 없는지를 검증
  - SRS에 모순이나 빠진 내용이 있다면, 이를 모델링을 통해 보충