- MODULE main{VAR}
- Include states and onoff/refund/lackOfMaterial commands
- Three process about coin_check,alarm_on and material_add
- Current water,milk,coffee and coin

```
MODULE main

VAR
    state : {off, on, refund,coin,alarm,addMaterial};
    onoff_command : boolean;
    refund_command : boolean;
    lackOfMaterial: boolean;
    coffeeMaking_command : {NONE, america, latte, cappuccino};

    coin_check : process coin(state, current_coin, coffeeMaking_command);
    alarm_on : process alarm(lackOfMaterial, alarm_timeout);
    material_add : process addMaterial(alarm_on)

    current_water : 0..1000;
    current_milk : 0..1000;
    current_coffee : 0..1000;
    current_coin : 0..2000;
```

- MODULE main {ASSIGN}

```
ASSIGN
    init(state) := off;
    next(state) :=
        case
        state = off & onoff_command = TRUE : on;
        state = on & onoff_command = FALSE : off;
        state = on & refund_command = TRUE : refund;
        state = refund : on;
        TRUE : state;
        esac;

    init(current_milk) := 1000;
    next(current_milk) :=
        case
        current_coin >= 500 & state = on & coffeeMaking_command = latte & current_milk >= 20 : current_milk - 20;
        current_coin >= 500 & state = on & coffeeMaking_command = cappuccino & current_milk >= 10 : current_milk - 10;
        state = on & coffeeMaking_command = latte & current_milk < 20 : current_milk;
        TRUE : current_milk;
        esac;

    init(current_water) := 1000;
    next(current_water) :=
        case
        current_coin >= 500 & state = on & coffeeMaking_command = latte & current_water >= 30 : current_water - 30;
        current_coin >= 500 & state = on & coffeeMaking_command = cappuccino & current_water >= 40 : current_water - 40;
        current_coin >= 300 & state = on & coffeeMaking_command = america & current_water >= 50 : current_water - 50;
        TRUE : current_water;
        esac;

    init(current_coffee) := 1000;
    next(current_coffee) :=
        case
        current_coin >= 500 & state = on & coffeeMaking_command = latte & current_coffee >= 10 : current_coffee - 10;
        current_coin >= 500 & state = on & coffeeMaking_command = cappuccino & current_coffee >= 10 : current_coffee - 10;
        current_coin >= 300 & state = on & coffeeMaking_command = america & current_coffee >= 10 : current_coffee - 10;
        TRUE : current_coffee;
        esac;

    init(coffeeMaking_command) := NONE;
    next(coffeeMaking_command) :=
        case
        state = on : {NONE, latte, america ,cappuccino};
        state != on : NONE;
        esac;
```

- MODULE main {ASSIGN}

```
init(lackOfMaterial) := FALSE;
next(lackOfMaterial) :=
    case
        current_milk < 10 | current_coffee < 10 | current_water < 30 : TRUE;
        coffeeMaking_command = america & current_water < 50 : TRUE;
        coffeeMaking_command = latte & (current_milk < 20 | current_water < 30 ) : TRUE;
        coffeeMaking_command = cappuccino & (current_milk < 10 | current_water < 40 ) : TRUE;
        TRUE : FALSE;
    esac;

init(current_coin) := 0;
next(current_coin) :=
    case
    state = off | state = on : current_coin;
    state = refund : 0;
    esac;
```

- MODULE main {SPEC}

```
SPEC AG EX TRUE
SPEC AG(coffeeMaking_command = america & current_coin<300 -> coffeeMaking_command = NONE)

SPEC EF (state = off -> state = on)
SPEC EF (state = on -> current_coin !=0)

SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = latte -> (current_coin = 0))
SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = cappuccino -> (current_coin = 0))
SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = america -> (current_coin = 200))

SPEC AF (current_milk = 1000 & state = on & coffeeMaking_command = latte -> EX(current_milk = 980))
SPEC AF (current_milk = 1000 & state = on & coffeeMaking_command = cappuccino -> EX(current_milk = 990))
```

- MODULE addMaterial
- Module for adding material when material is not enough
  - When material_state is true and user select fill command ,this module will fill the selected material to fill

```
MODULE addMaterial(alarm_on)
VAR
    material_state : boolean;

ASSIGN
    init(material_state) := FALSE;
    next(material_state) :=
        case
        material_state = TRUE & fill_command = fill_water : fill_water;
        material_state = TRUE & fill_command = fill_coffee : fill_coffee;
        material_state = TRUE & fill_command = fill_milk : fill_milk;
        esac;
```

- MODULE alarm
- Module for alarm in case of material shortage
  - Alarm switches to on when lackOfMaterial is true
  - Alarm switches to off after a period of time

```
MODULE alarm(lackOfMaterial, timeout)
VAR
    alarm_on : boolean;
ASSIGN
    init(alarm_on) := FALSE;
    next(alarm_on) :=
        case
        lackOfMaterial = TRUE : TRUE;
        timeout.timeout_alarm = TRUE :FALSE;
        TRUE : FALSE;
        esac;
```

- MODULE coin
- Reduce or insert coin' module
  - Insert : user can select 0,100,500,1000 to insert coin
  - Reduce : reduce by the price of coffee[America:300,cappuccino:500,latte:500]

```
MODULE coin(machine_state,current_coin2,coffeeMaking_command)
VAR
    coin_value : {0,100,500,1000};
    coin_reduce : {0,200,300};
ASSIGN
    init(coin_value) := 0;
    next(coin_value) :=
        case
        machine_state = on : {0,100,500,1000};
        TRUE : 0;
        esac;

    init(coin_reduce) := 0;
    next(coin_reduce) :=
        case
        machine_state = on & coffeeMaking_command = cappuccino : 500;
        machine_state = on & coffeeMaking_command = latte : 500;
        machine_state = on & coffeeMaking_command = america : 300;
        TRUE : 0;
        esac;
DEFINE
    main.current_coin := current_coin2 + coin_value - coin_reduce;
```

CTL property

No.	CTL SPEC	Description	Excepted Output	Result
1	SPEC AG EX TRUE	Deadlock	True	True
2	SPEC AG (coffee_Making_command = america & current_coin<300 -> coffeeMaking_command = None)	When the number of coins is less than 300 and the user selects coffee, the user will not get coffee	False	False
3	SPEC EF (state = off -> state = on)	It can be on when the machine is off	True	True
4	SPEC EF (state = on -> current_coin !=0)	Coins can be increased when the machine is on	True	True
5	SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = latte -> (current_coin = 0))	When you insert 500 coin and request latte,the coin becomes 0	True	True
6	SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = america -> (current_coin = 200))	When you insert 500 coin and request america ,the coin becomes 200	True	True

No.	CTL SPEC	Description	Excepted Output	Result
7	SPEC AF (current_milk = 1000 & state = on & coffeeMaking_command = latte -> EX(current_milk = 980))	Requesting latte reduces milk by 20	True	True
8	SPEC AF (current_milk = 1000 & state = on & coffeeMaking_command = cappuccino -> EX(current_milk = 990))	Requesting cappuccino reduces milk by 10	True	True
9	SPEC AX (current_coin = 500 & state = on & coffeeMaking_command = cappuccino -> (current_coin = 0))	When you insert 500 coin and request cappuccino,the coin becomes 0	True	True

Model Checking commands

```
E:\NuSMV-2.6.0-win64\NuSMV-2.6.0-win64\bin>NuSMV -int
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

NuSMV > read_model -i CVM.smv
NuSMV > go
WARNING *** Processes are still supported, but deprecated.      ***
WARNING *** In the future processes may be no longer supported. ***

WARNING *** The model contains PROCESSes or ISAs. ***
WARNING *** The HRC hierarchy will not be usable. ***
NuSMV > compute_reachable
The computation of reachable states has been completed.
The diameter of the FSM is 1105.
```

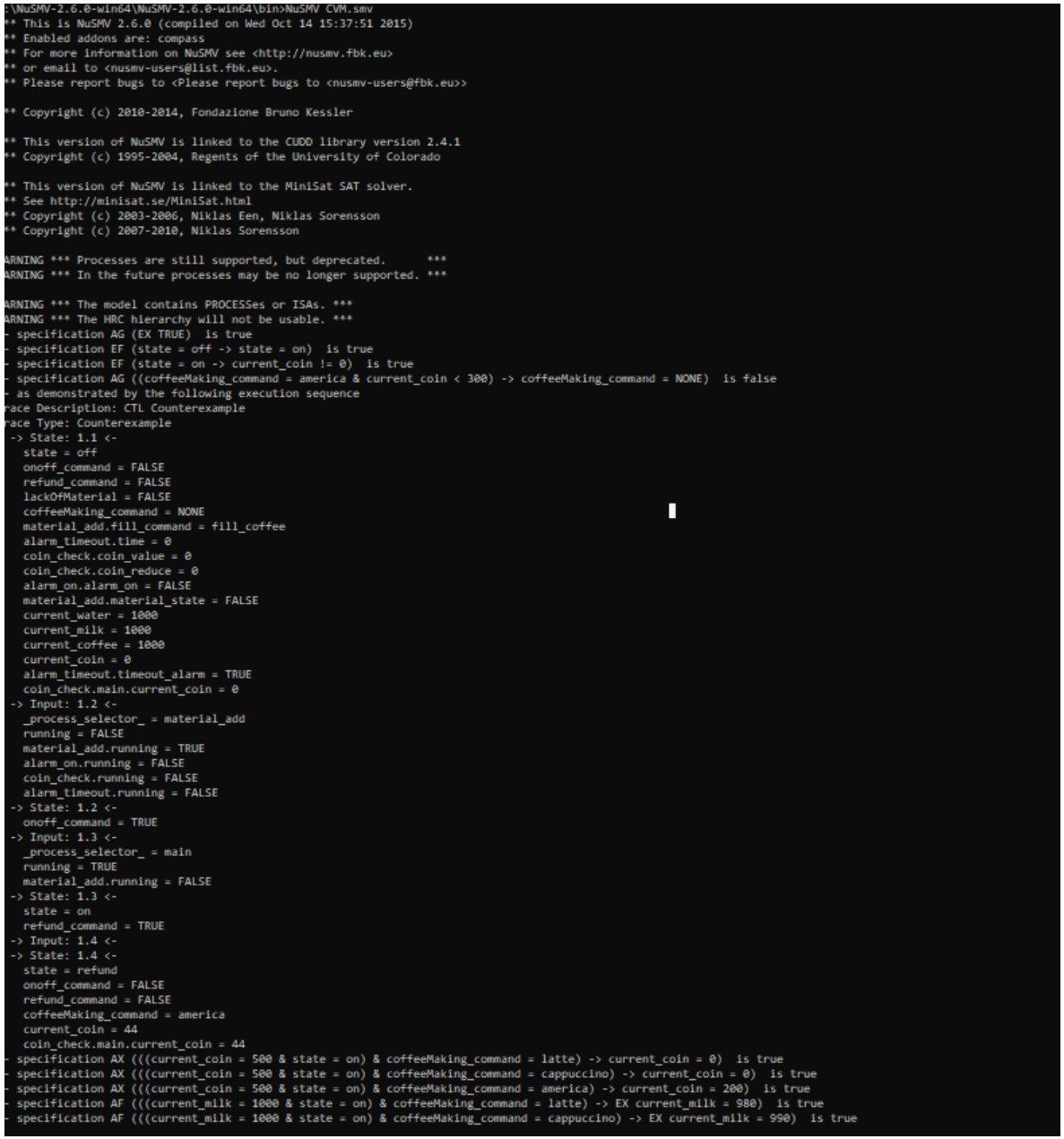
Verification Result

```
NuSMV > check_ctlspec
-- specification AG (EX TRUE)  is true
-- specification EF (state = off -> state = on)  is true
-- specification EF (state = on -> current_coin != 0)  is true
```

```
-- specification AX (((current_coin = 500 & state = on) & coffeeMaking_command = latte) -> current_coin = 0)  is true
-- specification AX (((current_coin = 500 & state = on) & coffeeMaking_command = cappuccino) -> current_coin = 0)  is true
-- specification AX (((current_coin = 500 & state = on) & coffeeMaking_command = america) -> current_coin = 200)  is true
-- specification AF (((current_milk = 1000 & state = on) & coffeeMaking_command = latte) -> EX current_milk = 980)  is true
-- specification AF (((current_milk = 1000 & state = on) & coffeeMaking_command = cappuccino) -> EX current_milk = 990)  is true
```

```
-- specification AG ((coffeeMaking_command = america & current_coin < 300) -> coffeeMaking_command = NONE)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    state = off
    onoff_command = FALSE
    refund_command = FALSE
    lackOfMaterial = FALSE
    coffeeMaking_command = NONE
    material_add.fill_command = fill_coffee
    alarm_timeout.time = 0
    coin_check.coin_value = 0
    coin_check.coin_reduce = 0
    alarm_on.alarm_on = FALSE
    material_add.material_state = FALSE
    current_water = 1000
    current_milk = 1000
    current_coffee = 1000
    current_coin = 0
    alarm_timeout.timeout_alarm = TRUE
    coin_check.main.current_coin = 0
  -> Input: 1.2 <-
    _process_selector_ = material_add
    running = FALSE
    material_add.running = TRUE
    alarm_on.running = FALSE
    coin_check.running = FALSE
    alarm_timeout.running = FALSE
  -> State: 1.2 <-
    onoff_command = TRUE
  -> Input: 1.3 <-
    _process_selector_ = main
    running = TRUE
    material_add.running = FALSE
  -> State: 1.3 <-
    state = on
    refund_command = TRUE
  -> Input: 1.4 <-
  -> State: 1.4 <-
    state = refund
    onoff_command = FALSE
    refund_command = FALSE
    coffeeMaking_command = america
    current_coin = 44
    coin_check.main.current_coin = 44
```

SMV model checking –Batch mode



Proposal- Recommender System

- The task of recommender system
  - Help users find what they want (music, video, merchandise)
  - Improve the utilization of people's information and reduce information overload
  - Improve site click-through rate and conversion rate
  - Deepen understanding of users and provide users with personalized customized services
- Algorithms for recommender systems

Popularity-Based Algorithms, Collaborative Filtering Algorithms, Content-Based Algorithms, Model-Based Algorithms, Hybrid Algorithms

To learn a recommendation system, you need to first learn the frameworks and concepts related to big data, and you can draw valuable conclusions for personalized recommendation from massive data, including data collection, analysis, and visualization. Knowledge, but also machine learning knowledge To realize the algorithm mentioned above, it can also be used to discover the laws of data. Recommendation systems have been widely used in the industry, and learning recommendation systems is very helpful for the construction of knowledge systems.