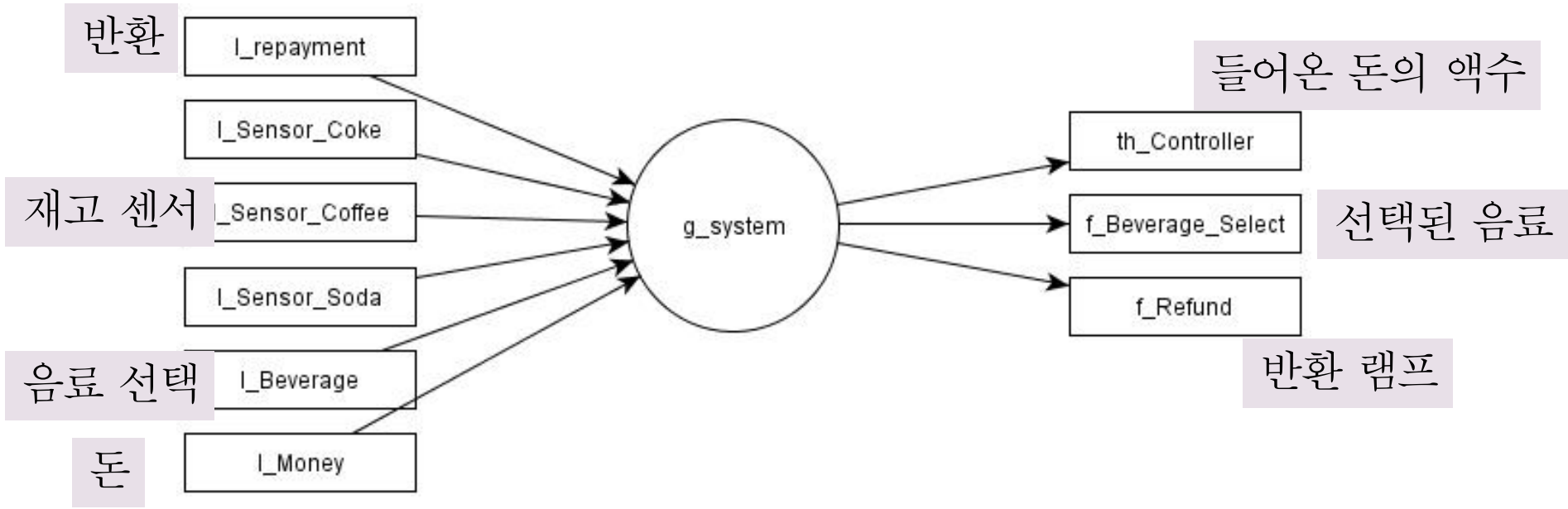
Software Verification
- Coffee Vending Machine-

Team 2: 200611453 권지숙
200611463 김지원
200611466 류아름
200611513 임주희

구현 과제

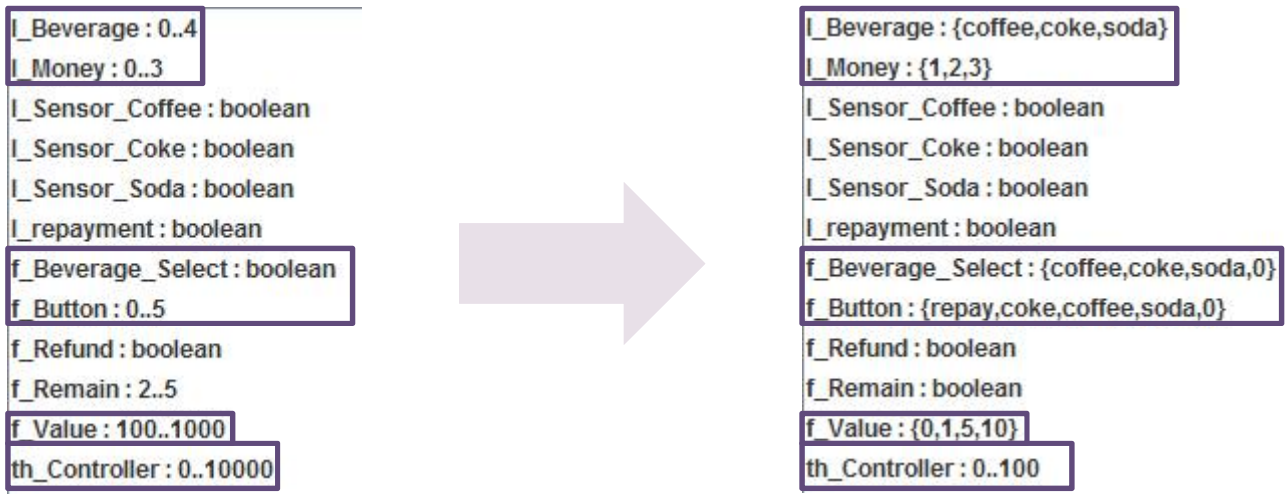
- NuSRS와 SMV를 이용한 커피 자판기 모델링
- 요구 사항
  - 3가지 종류의 음료(coffee, coke, soda)
  - 각 음료의 재고 상황을 알리는 센서 3개
  - 반환 램프
  - 버튼 : 4개 (각 음료 선택 / 반환 버튼)
  - 돈 입력 : 100, 500, 1000원만 받는 것으로 가정
  - 잔액이 10000원을 넘지 않도록 함

Mapping


반환
I_repayment
I_Sensor_Coke
재고 센서
_Sensor_Coffee
I_Sensor_Soda
음료 선택
I_Beverage
돈
I_Money
g_system
들어온 돈의 액수
th_Controller
f_Beverage_Select
선택된 음료
f_Refund
반환 램프

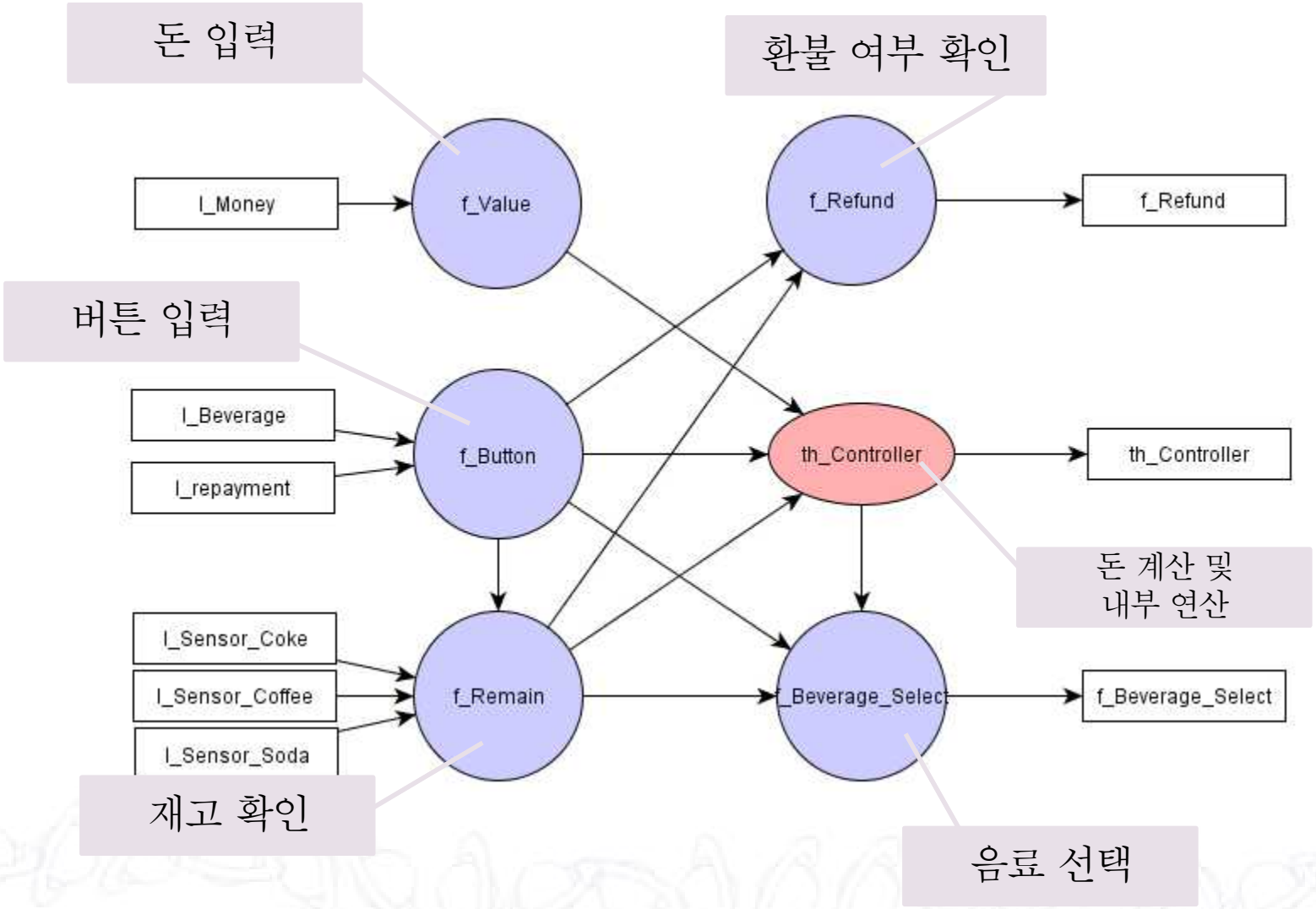
- f_Value : {0, 1, 5, 10}
- f_Button : {0, repay, coffee, coke, soda}
- f_Remain : boolean
- f_Refund : boolean
- F_Beverage_Select : {0, coffee, coke, soda}
- th_Controller : 0 ... 100
- k_Cost : 음료 값을 나타내는 상수 (5)
- th_Controller_t0 : th_Controller의 전 사이클 값을 나타내기 위한 값

g_System (1)



- Enum type 사용
- 숫자 범위 축소 (→ transition 개수 감소)

g_System (2)


돈 입력
환불 여부 확인
l_Money
f_Value
f_Refund
f_Refund
버튼 입력
l_Beverage
l_repayment
f_Button
th_Controller
th_Controller
돈 계산 및
내부 연산
l_Sensor_Coke
l_Sensor_Coffee
l_Sensor_Soda
f_Remain
f_Beverage_Select
f_Beverage_Select
재고 확인
음료 선택

f_Remain (1)

Conditions	1	2	3	4
(f_Button = 2) & (I_Sensor_Coffee = true)	T	-	-	F
(f_Button = 3) & (I_Sensor_Coke = true)	-	T	-	F
(f_Button = 4) &(I_Sensor_Soda = true)	-	-	T	F
f_Button = 0 | f_Button = 1	F	F	F	T
Action	**1**	**2**	**3**	**4**
f_Remain := true	O	O	O	
f_Remain := false				O

f_Remain (2)

Conditions	1	2	3	4	5	6	7
(f_Button = coffee) & (I_Sensor_Coffee = true)	T	F	F	F	F	F	F
(f_Button = coke) & (I_Sensor_Coke = true)	F	T	F	F	F	F	F
(f_Button = soda) & (I_Sensor_Soda = true)	F	F	T	F	F	F	F
(f_Button = coffee) & (I_Sensor_Coffee = false)	F	F	F	T	F	F	F
(f_Button = coke) & (I_Sensor_Coke = false)	F	F	F	F	T	F	F
(f_Button = soda) & (I_Sensor_Soda = false)	F	F	F	F	F	T	F
f_Button = 0 | f_Button = repay	F	F	F	F	F	F	T
Action	**1**	**2**	**3**	**4**	**5**	**6**	**7**
f_Remain := true	O	O	O				O
f_Remain := false				O	O	O	

Property – f_Remain (1)

U : for until

- 버튼을 누른 음료에 대한 재고가 존재하거나, 반환 버튼을 눌렀거나, 버튼을 누르지 않으면 f_Remain = true.

➡ f_Remain = true가 나오는 경우는, 재고가 있거나 반환 버튼 또는 버튼 입력이 없는 경우

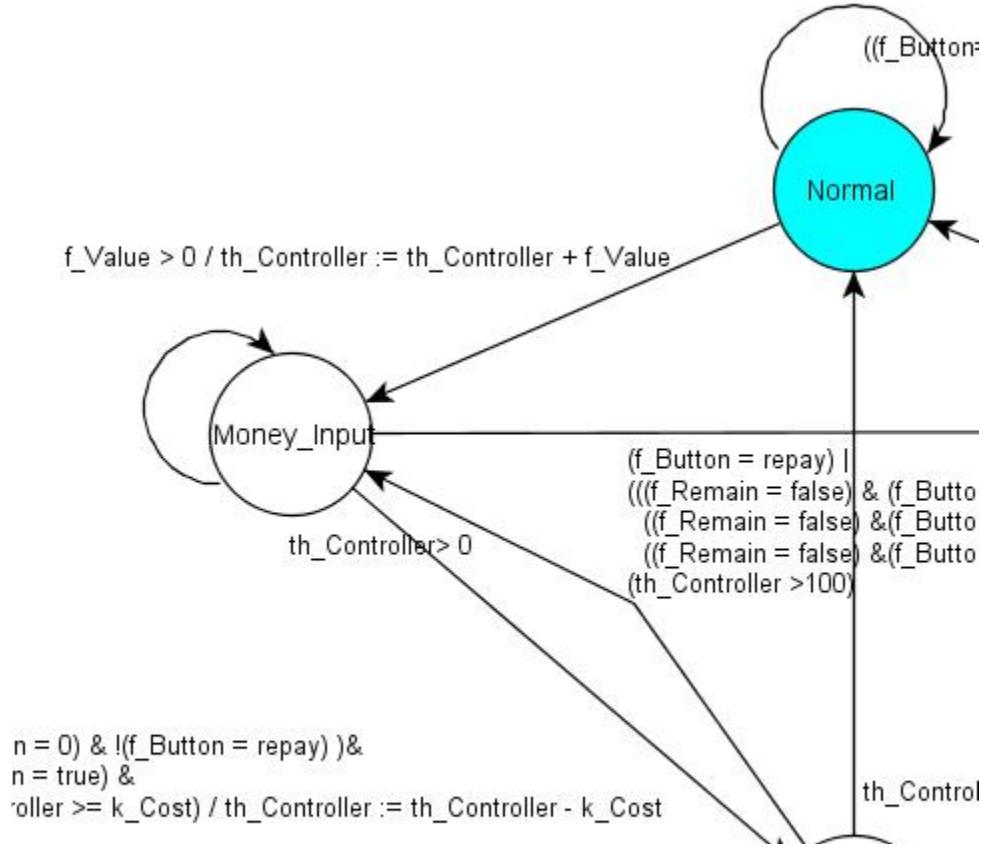
- SPEC A(f_Remain = true U

(((f_Button = coffee) & (I_Sensor_Coffee = true)) |
((f_Button = soda) & (I_Sensor_Soda = true)) |
((f_Button = coke) & (I_Sensor_Coke = true)) |
!((f_Button = repay) | (f_Button = 0))))

Property – f_Remain (2)

- f_Remain에서 false가 나오는 경우는 버튼과 센서가 일치하지 않는 경우 뿐
  (repay버튼이나 아무것도 안 누른 상태는 true로 간주됨)
- SPEC A(f_Remain = false U
  (((f_Button = coffee) & (I_Sensor_Coffee = false)) |
  ((f_Button = soda) & (I_Sensor_Soda =false)) |
  ((f_Button = coke) & (I_Sensor_Coke = false)) |
  !(f_Button = repay) | !(f_Button = 0)))

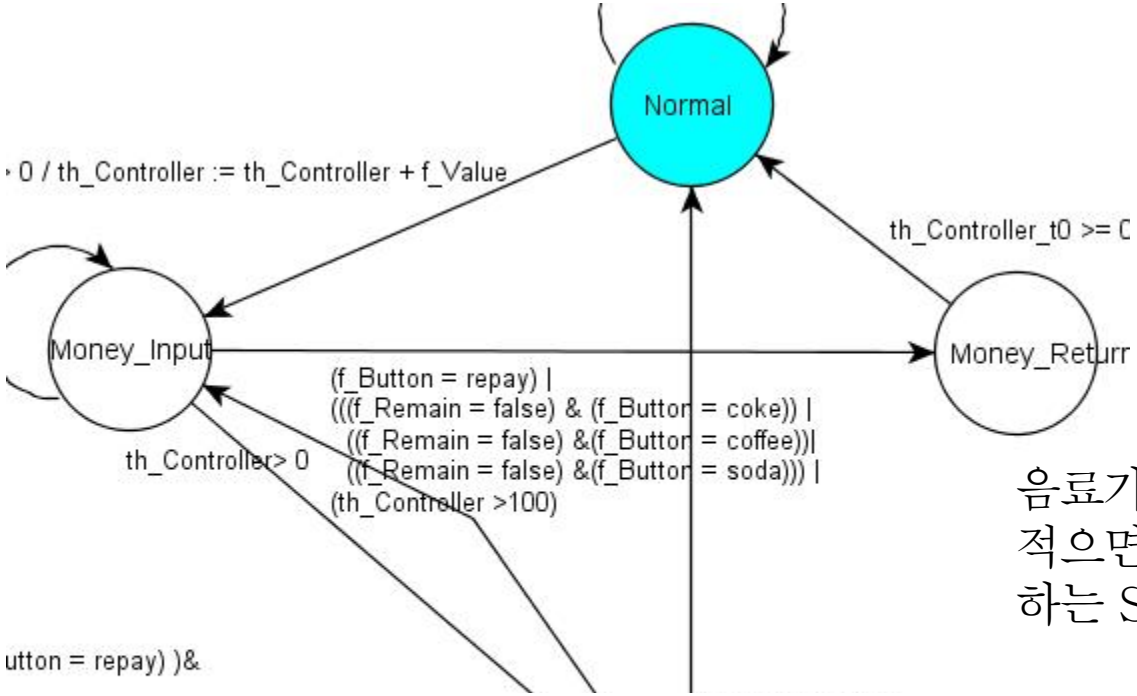
Property - th_Controller (1)-1



돈이 들어와서 음료 1개 값 이상의 잔액이 남으면, (음료가 나간 후) 반드시 그 다음에 Money_Input 상태로 돌아가야 함.

SPEC AG(FROM-Money_Input-To-Beverage_Out-taken & (th_Controller > k_Cost) -> AX(FROM-Beverage_Out-TO-Money_Input-taken)

Property - th_Controller (1)-2



음료가 나간 후 잔액이 음료 값보다 적으면 돈 입력을 다시 받거나 반환하는 State로 반드시 돌아가야 함.

SPEC AG((FROM-Beverage_Out-TO-Money_Input-taken) &(th_Controller <k_Cost)) -> AX(FROM-Money_Input-TO-Money_Input-taken &(FROM-Money_Input-TO-Money_Input-enabled | FROM-Money_Input-TO-Money_Return_Without_Beverage-enabled))

Property - g_System (1)

- 검증하고자 하는 내용 :

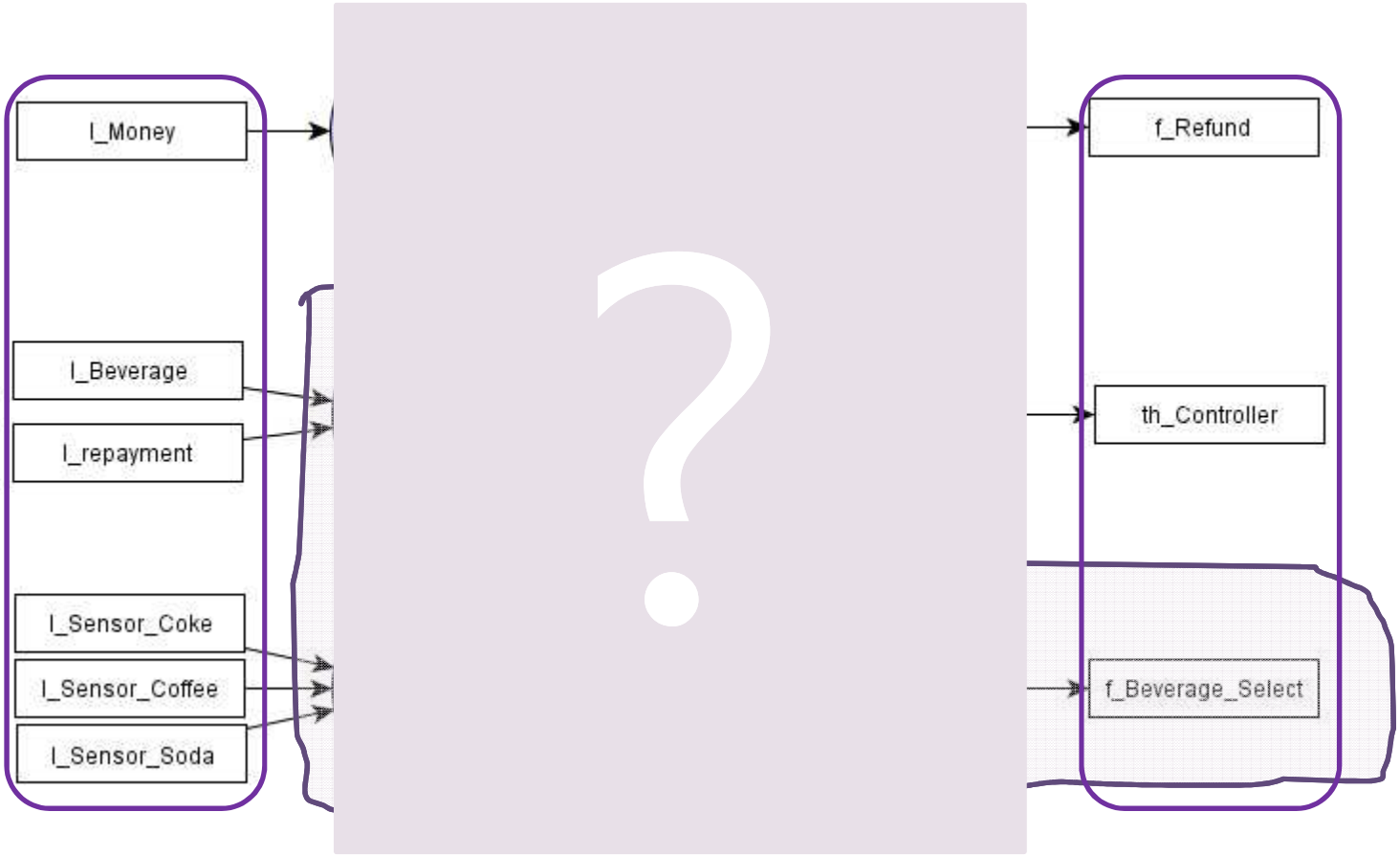
  버튼 입력, 재고가 있고, 지불가에 돈이 음료 값 이상으로 들어있을 경우, 나온 음료는 버튼 선택과 같아야 한다

  SPEC AG ((f_Button.f_Button = coffee) & (f_Remain.f_Remain = 1)
  &(th_Controller.th_Controller_t0 >= th_Controller.k_Cost) ->
  EX(f_Beverage_Select.f_Beverage_Select = coffee))

  ⬇

  SPEC AG ((f_Button.f_Button = coffee) & (f_Remain.f_Remain = 1)
  &(th_Controller.th_Controller_t0 >= th_Controller.k_Cost) &
  (th_Controller.FROM-Money_Input-TO-Beverage_Out-taken) ->
  EX(f_Beverage_Select.f_Beverage_Select = coffee))

Property - g_System (2)



Property – g_System (3)

- SPEC AG ((I_Money = 3) & (I_repayment = 0) & (I_Beverage = coffee) & (I_Sensor_Coffee = 1) & (th_Controller.FROM-Money_Input-TO-Beverage_Out-taken) -> EX(f_Beverage_Select.f_Beverage_Select =coffee))

Failure to verify T_T

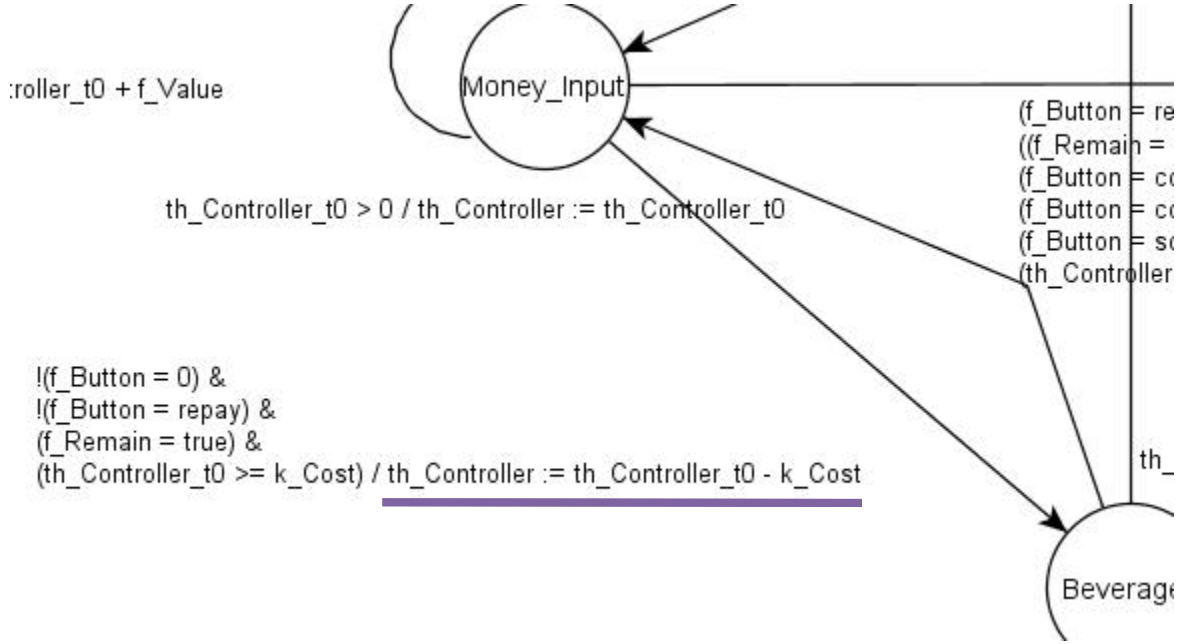
Tracing (1)

Source | Trace | Log

File Edit Run View

	9	10	11	12	13	14	15	16
\FROM-Normal-TO-Money_Input-taken	0	0	0	0	0	0	0	0
STATE	Money_Ir	Money_Ir	Money_Ir	Beverage	Money_Ir	Money_Ir	Money_Ir	Money_Ir
f_Button	0	0	soda	0	0	0	0	0
f_Remain	0	0	1	0	0	0	0	0
f_Value	1	1	1	1	1	1	1	1
false	0	0	0	0	0	0	0	0
\in-Beverage_Out	0	0	0	1	0	0	0	0
\in-Money_Input	1	1	1	0	1	1	1	1
\in-Money_Return_Without_Beverage	0	0	0	0	0	0	0	0
\in-Normal	0	0	0	0	0	0	0	0
k_Cost	5	5	5	5	5	5	5	5
k_Delay	3	3	3	3	3	3	3	3
th_Controller	4	5	5	0	5	5	6	6
th_Controller_t0	4	4	5	5	0	5	5	6
true	1	1	1	1	1	1	1	1

Tracing (2)


:roller_t0 + f_Value
Money_Input
(f_Button = re
((f_Remain =
(f_Button = c
(f_Button = c
(f_Button = s
(th_Controller
th_Controller_t0 > 0 / th_Controller := th_Controller_t0
!(f_Button = 0) &
!(f_Button = repay) &
(f_Remain = true) &
(th_Controller_t0 >= k_Cost) / th_Controller := th_Controller_t0 - k_Cost
th_
Beverage

Thank you! :)