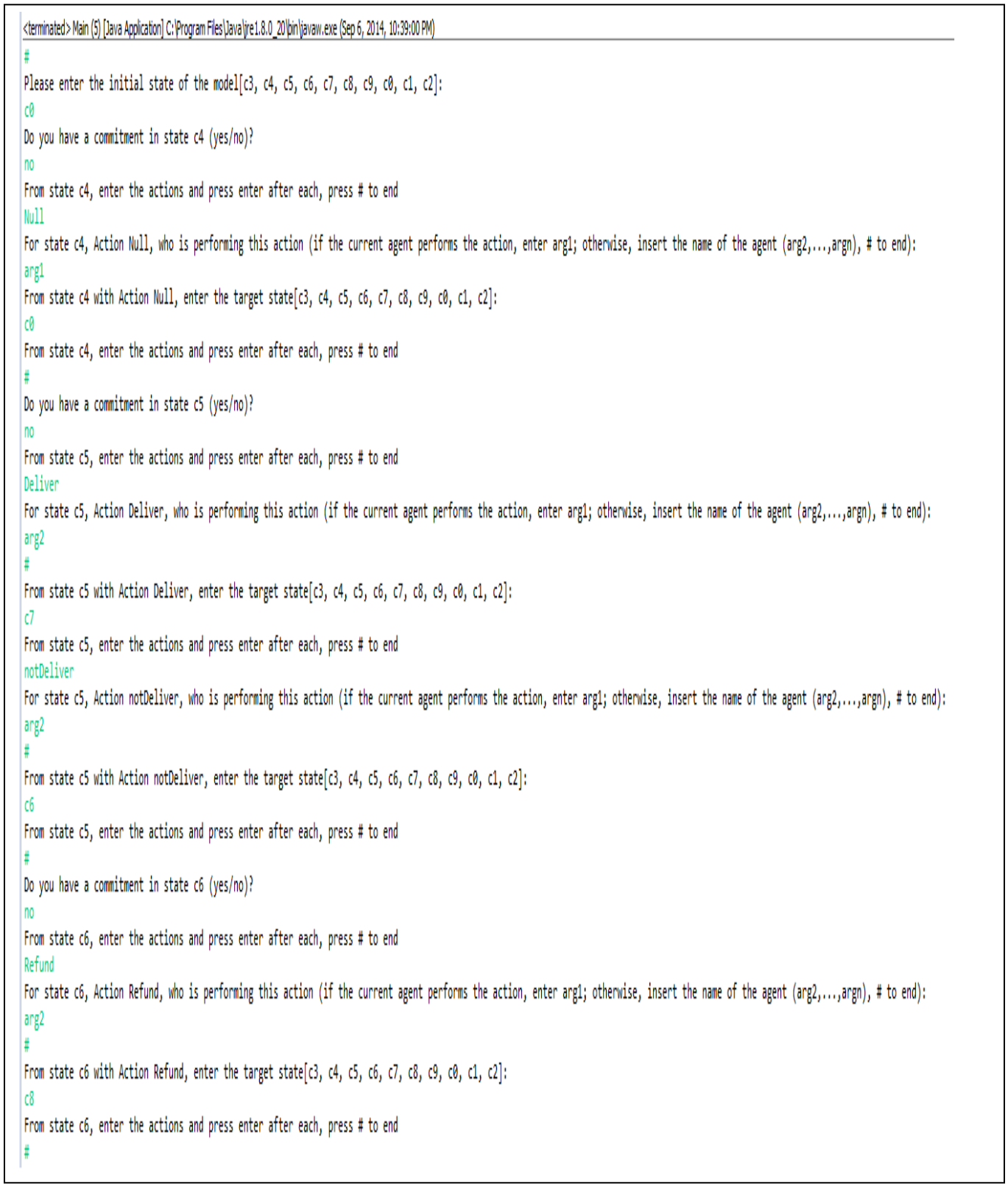
To simplify the use of JTT, we present screenshots for a complete example running under this tool. This example illustrates the case that we have 1 Merchant and 3 Customers. The tool starts by asking the name of the model, its instance(s) and states as in the figure below.

```
<terminated> Main (5) [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Sep 6, 2014, 10:39:00 PM)
***Java Tranformer Tool Started***

Please insert the Name of the model (# to end):
Customer
Please enter the name of each instance of this model and press enter after each name: (# to end)
cus1
cus2
cus3
#
Please enter the states of the model and press enter after each state[]: (# to end)
c0
Please enter the states of the model and press enter after each state[c0]: (# to end)
c1
Please enter the states of the model and press enter after each state[c0, c1]: (# to end)
c2
Please enter the states of the model and press enter after each state[c0, c1, c2]: (# to end)
c3
Please enter the states of the model and press enter after each state[c3, c0, c1, c2]: (# to end)
c4
Please enter the states of the model and press enter after each state[c3, c4, c0, c1, c2]: (# to end)
c5
Please enter the states of the model and press enter after each state[c3, c4, c5, c0, c1, c2]: (# to end)
c6
Please enter the states of the model and press enter after each state[c3, c4, c5, c6, c0, c1, c2]: (# to end)
c7
Please enter the states of the model and press enter after each state[c3, c4, c5, c6, c7, c0, c1, c2]: (# to end)
c8
Please enter the states of the model and press enter after each state[c3, c4, c5, c6, c7, c8, c0, c1, c2]: (# to end)
c9
Please enter the states of the model and press enter after each state[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]: (# to end)
#
Please enter the initial state of the model[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c0
Do you have a commitment in state c4 (yes/no)?
no
From state c4, enter the actions and press enter after each, press # to end
Null
For state c4, Action Null, who is performing this action (if the current agent performs the action, enter arg1; otherwise, insert the name of the agent (arg2,...,argn), # to end)
arg1
From state c4 with Action Null, enter the target state[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c0
From state c4, enter the actions and press enter after each, press # to end
#
```

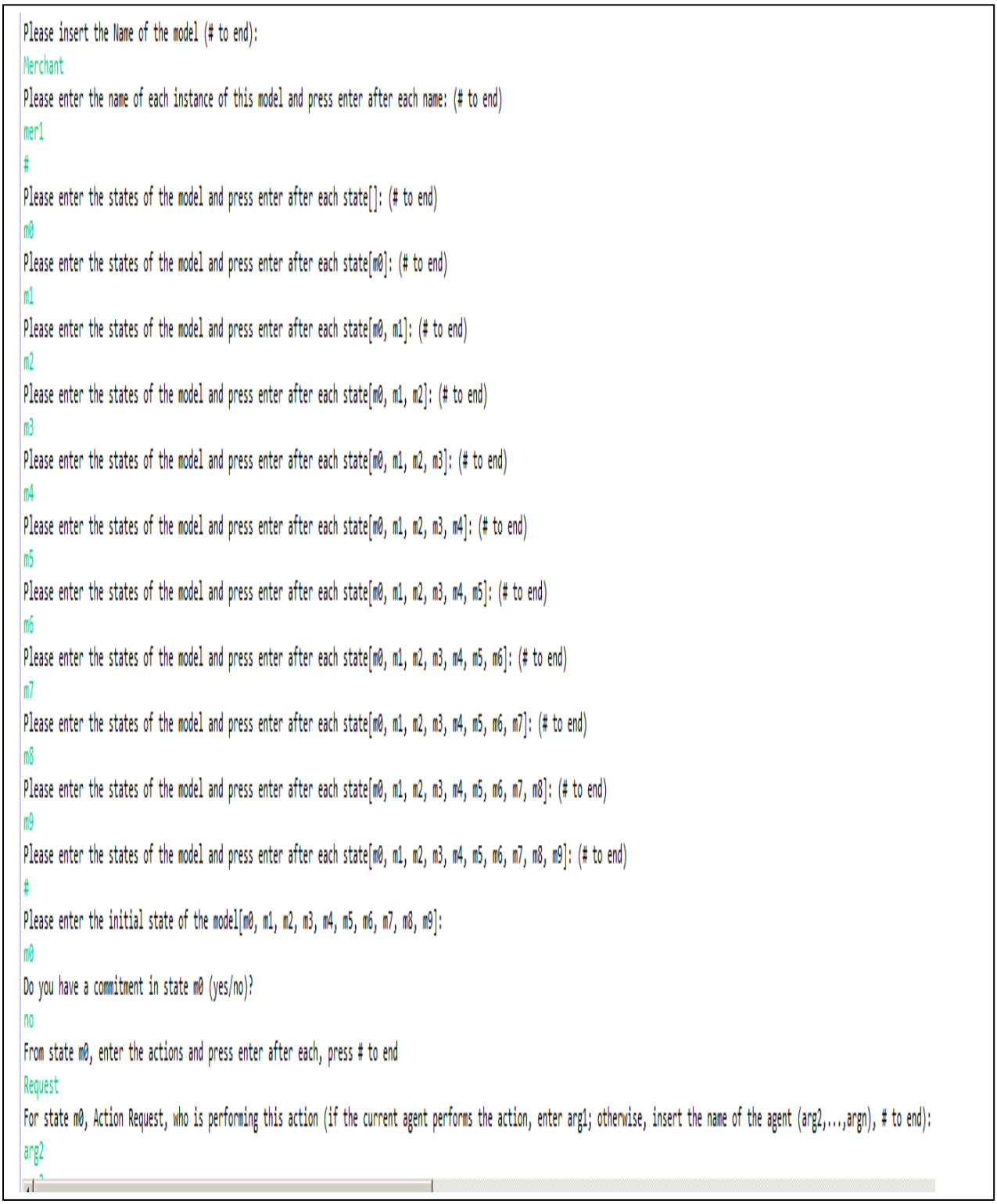
After that, the tool asks the user to insert the information of each state in the model as in the figure below.



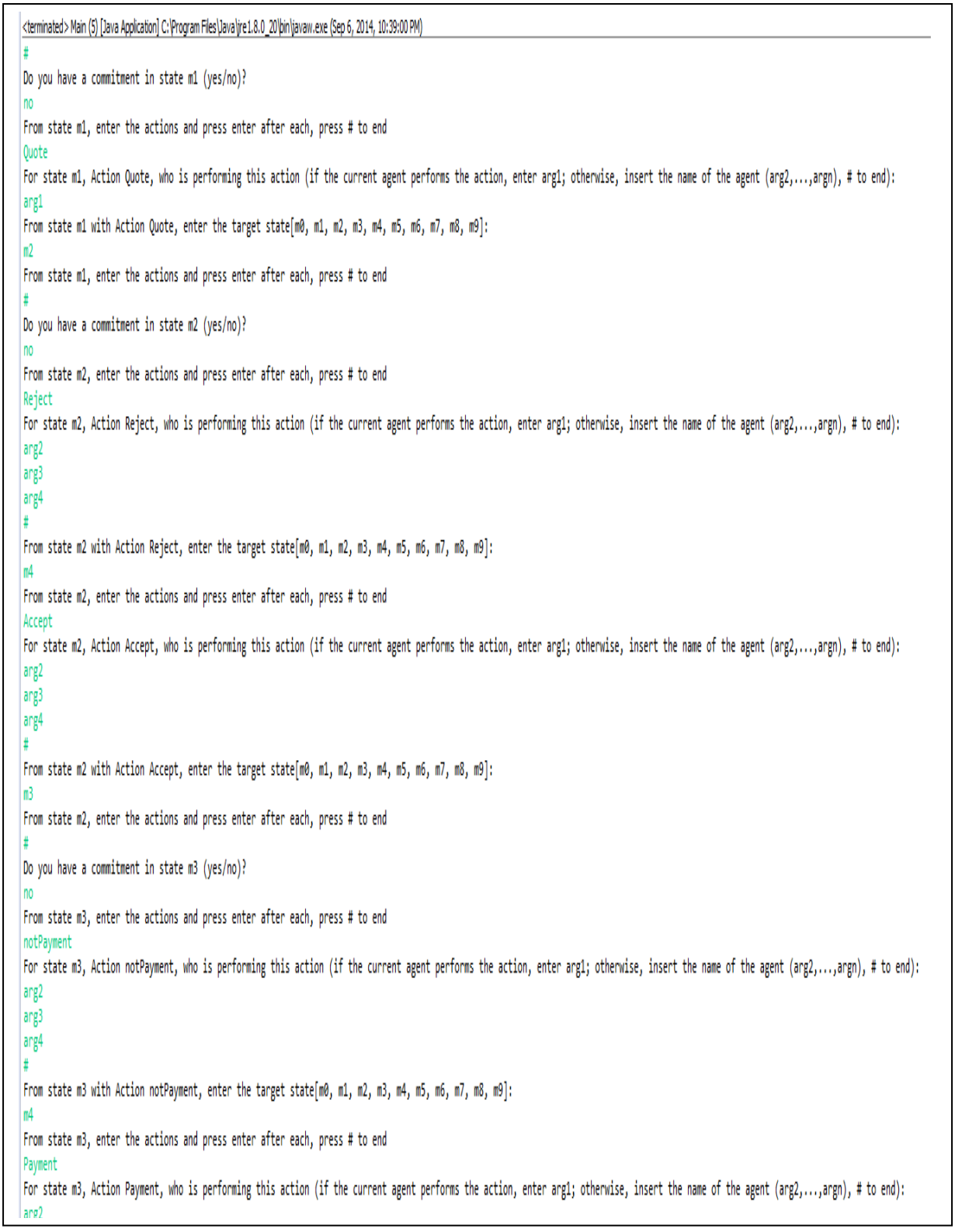
If a state has commitment then the user should insert where this commitment is fulfilled. After that, user has to insert the atomic propositions of this model as in the figure below.

```
Do you have a commitment in state c3 (yes/no)?
yes
What is the state where this commitment is fulfilled[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]?
c5
From state c3, enter the actions and press enter after each, press # to end
notPayment
For state c3, Action notPayment, who is performing this action (if the current agent performs the action, enter arg1; otherwise, insert the name of the agent (arg2,...,argn), # to e
arg1
From state c3 with Action notPayment, enter the target state[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c4
From state c3, enter the actions and press enter after each, press # to end
Payment
For state c3, Action Payment, who is performing this action (if the current agent performs the action, enter arg1; otherwise, insert the name of the agent (arg2,...,argn), # to end)
arg1
From state c3 with Action Payment, enter the target state[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c5
From state c3, enter the actions and press enter after each, press # to end
#
Please enter module atomic proposition (# to end):
Pay
Please enter the instances of this atomic proposition (press # to end):
cus1
cus2
cus3
#
Please enter the state of this atomic proposition[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c5
Please enter module atomic proposition (# to end):
Deliver
Please enter the instances of this atomic proposition (press # to end):
cus1
cus2
cus3
#
Please enter the state of this atomic proposition[c3, c4, c5, c6, c7, c8, c9, c0, c1, c2]:
c7
Please enter module atomic proposition (# to end):
#
```

This process continues until inserting all information for all states in the model. Hereafter, the user can start inserting the second model as in the figure below.



In a similar way to the previous model, user should insert the states of the model with their information as in the figure below.



After that, the user has two choices to insert the specifications; either using console, or using an input text file. In the figure below, the user inserts the specifications through the input file (formula.txt).

```
Do you have a commitment in state m8 (yes/no)?
no
From state m8, enter the actions and press enter after each, press # to end
Null
For state m8, Action Null, who is performing this action (if the current agent performs the action, enter arg1; otherwise, insert the name of the agent (arg2,...,argn), # to end):
arg1
From state m8 with Action Null, enter the target state[m0, m1, m2, m3, m4, m5, m6, m7, m8, m9]:
m0
From state m8, enter the actions and press enter after each, press # to end
#
Do you have a commitment in state m9 (yes/no)?
no
From state m9, enter the actions and press enter after each, press # to end
Null
For state m9, Action Null, who is performing this action (if the current agent performs the action, enter arg1; otherwise, insert the name of the agent (arg2,...,argn), # to end):
arg1
From state m9 with Action Null, enter the target state[m0, m1, m2, m3, m4, m5, m6, m7, m8, m9]:
m0
From state m9, enter the actions and press enter after each, press # to end
#
Please enter module atomic proposition (# to end):
#
Please insert the Name of the model (# to end):
#
Specifictions, do you want to enter specs from console or have them read them from a file (file,console)?
file
Please enter input filename (default: formula.txt)?
formula.txt
```

When user finishes inserting the specifications, the program will display the output extended NuSMV code as in the following figures and store it in output.txt file.

```
Module main
Var
cus1 : process <Customer>(cus1,mer1);
cus2 : process <Customer>(cus2,mer1);
cus3 : process <Customer>(cus3,mer1);
mer1 : process <Merchant>(mer1,cus1,cus2,cus3);

---------------------------------------
-- Atomic Propositions
---------------------------------------
DEFINE DEF_Pay := (cus1.state = c5 | cus2.state = c5 | cus3.state = c5);
DEFINE DEF_Deliver := (cus1.state = c7 | cus2.state = c7 | cus3.state = c7);

---------------------------------------
-- The definition of Customer Agent (cus1,cus2,cus3)
---------------------------------------
MODULE Customer (arg1,arg2)
VAR state: {c3,c4,c5,c6,c7,c8,c9,c0,c1,c2};
IVAR action : {Reject,Null,Payment,Accept,Alpha_cus,notPayment,Request,Beta_cus,Gamma_cus};
INIT (state = c0)
    TRANS(next(state)= case
        (arg1.state = c4 & arg1.action = Null) : c0;
        (arg1.state = c5 & arg2.action = notDeliver) : c6;
        (arg1.state = c5 & arg1.action = Gamma_cus) : c3;
        (arg1.state = c5 & arg2.action = Deliver) : c7;
        (arg1.state = c5 & arg1.action = Beta_cus) : c5;
        (arg1.state = c6 & arg2.action = Refund) : c8;
        (arg1.state = c7 & arg2.action = Receipt) : c9;
        (arg1.state = c8 & arg1.action = Null) : c0;
        (arg1.state = c9 & arg1.action = Null) : c0;
        (arg1.state = c0 & arg1.action = Request) : c1;
        (arg1.state = c1 & arg2.action = Quote) : c2;
        (arg1.state = c2 & arg1.action = Reject) : c4;
        (arg1.state = c2 & arg1.action = Accept) : c3;
        (arg1.state = c3 & arg1.action = Alpha_cus) : c5;
        (arg1.state = c3 & arg1.action = Payment) : c5;
        (arg1.state = c3 & arg1.action = notPayment) : c4;
        (arg1.state = c3 & arg1.action = Beta_cus) : c3;
    esac)
```

```
----------------------------------------
-- The definition of Merchant Agent (mer1)
----------------------------------------
MODULE Merchant (arg1,arg2,arg3,arg4)
VAR state: {m0,m1,m2,m3,m4,m5,m6,m7,m8,m9};
IVAR action : {Quote,Deliver,Null,Beta_mer,Receipt,Refund,notDeliver,Gamma_mer,Alpha_mer};
INIT (state = m0)
    TRANS(next(state)= case
        (arg1.state = m0 & (  arg3.action = Request | arg4.action = Request | arg2.action = Request )) : m1;
        (arg1.state = m1 & arg1.action = Quote) : m2;
        (arg1.state = m2 & (  arg3.action = Reject | arg2.action = Reject | arg4.action = Reject )) : m4;
        (arg1.state = m2 & (  arg3.action = Accept | arg4.action = Accept | arg2.action = Accept )) : m3;
        (arg1.state = m3 & (  arg4.action = notPayment | arg3.action = notPayment | arg2.action = notPayment )) : m4;
        (arg1.state = m3 & (  arg3.action = Payment | arg4.action = Payment | arg2.action = Payment )) : m5;
        (arg1.state = m4 & arg1.action = Null) : m0;
        (arg1.state = m5 & arg1.action = Deliver) : m7;
        (arg1.state = m5 & arg1.action = notDeliver) : m6;
        (arg1.state = m5 & arg1.action = Alpha_mer) : m7;
        (arg1.state = m5 & arg1.action = Beta_mer) : m5;
        (arg1.state = m6 & arg1.action = Refund) : m8;
        (arg1.state = m7 & arg1.action = Beta_mer) : m7;
        (arg1.state = m7 & arg1.action = Gamma_mer) : m5;
        (arg1.state = m7 & arg1.action = Receipt) : m9;
        (arg1.state = m8 & arg1.action = Null) : m0;
        (arg1.state = m9 & arg1.action = Null) : m0;
    esac)

SPEC AG!((EAX(Cus.action = Gamma_Cus)(AAX(Cus.action = Alpha_Cus)(AAX(Cus.action = Beta_Cus)(Def_Pay)&AAX(Mer.action = Beta_Mer)(Def_Pay))||EAX(Cus.action = B
SPEC AG!((EAX(Cus.action = Gamma_Cus)(AAX(Cus.action = Alpha_Cus)(AAX(Cus.action = Beta_Cus)(Def_Pay)&AAX(Mer.action = Beta_Mer)(Def_Pay))||EAX(Cus.action = B
SPEC AG!(( AAX(Cus.action = Alpha_Cus)(AAX(Mer.action = Beta_Mer)(Def_Pay)&AAX(Cus.action = Beta_Cus)(Def_Pay)))&AG(! AAX(Cus.action = Beta_Cus)( AAX(Cus.acti
SPEC AG( AAX(Mer.action = Alpha_Mer)(AAX(Cus.action = Beta_Cus)(Def_Deliver)&AAX(Mer.action = Beta_Mer)(Def_Deliver))->EF(EAX(Mer.action = Gamma_Mer)(AAX(Mer.
SPEC EF AAX(Mer.action = Alpha_Mer)(AAX(Cus.action = Beta_Cus)(Def_Deliver)&AAX(Mer.action = Beta_Mer)(Def_Deliver))
SPEC AG!( AAX(Mer.action = Alpha_Mer)(AAX(Cus.action = Beta_Cus)(Def_Deliver)&AAX(Mer.action = Beta_Mer)(Def_Deliver)))
```