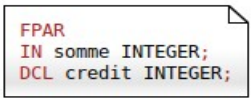
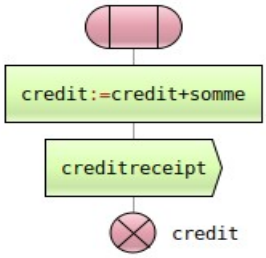
Lydia KHEMSINE

1-MODELISATION

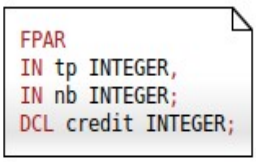
a.

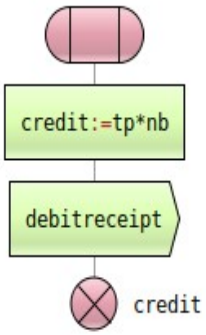
Procédure creditproc : crédite un compte d'un utilisateur et retourne un signal creditreceipt





Procédure debitproc : débite une somme d'un compte et retourne debitreceipt



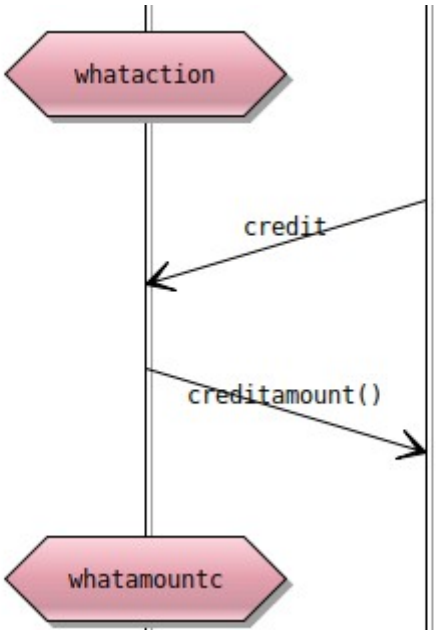


b.

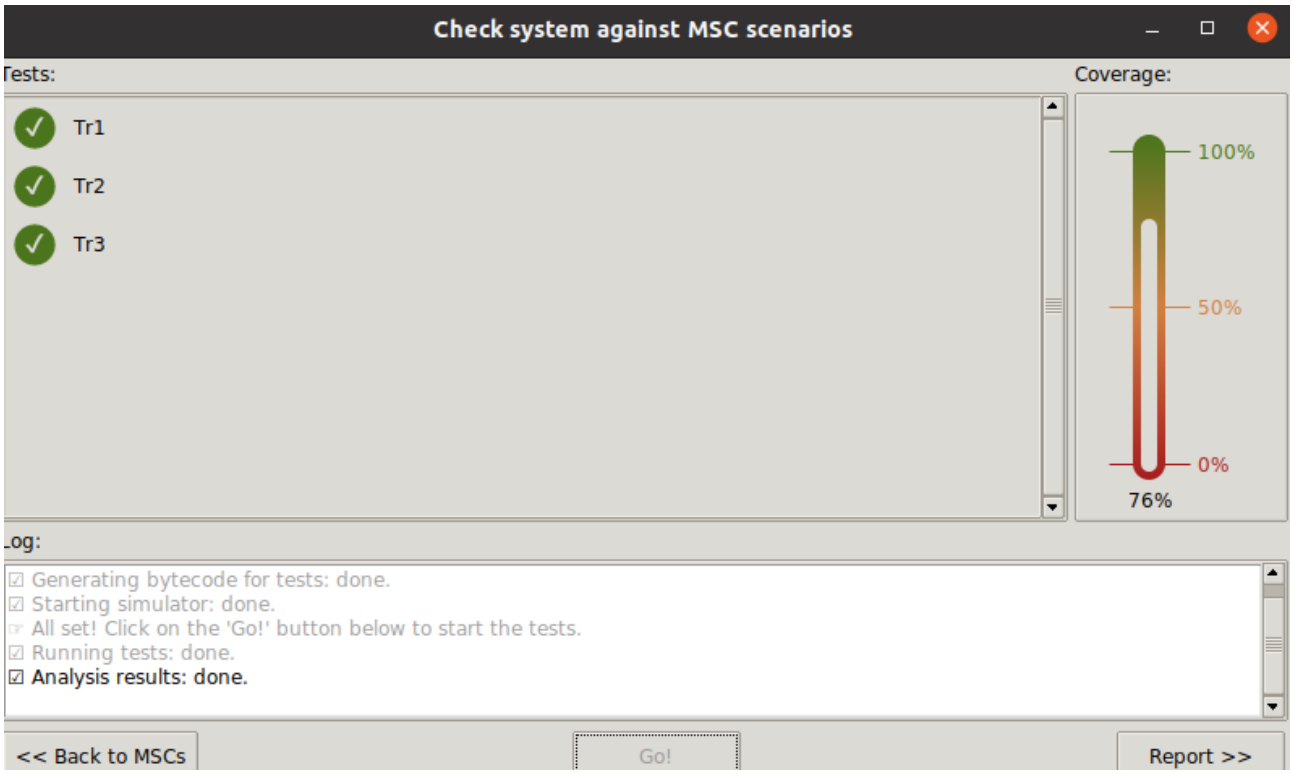
- Trace MSC qui correspond à une séquence de test selon la stratégie UIO :

TSUIO(whataction,credit/creditamount,whatamountc) :

Aprés *whataction* on demande de créditer un compte(signal : credit), et on reçoit *creditamount* et on arrive sur l'état whatamountc comme le montre la capture suivante :



- **les 3 traces qui couvre >65 % :**

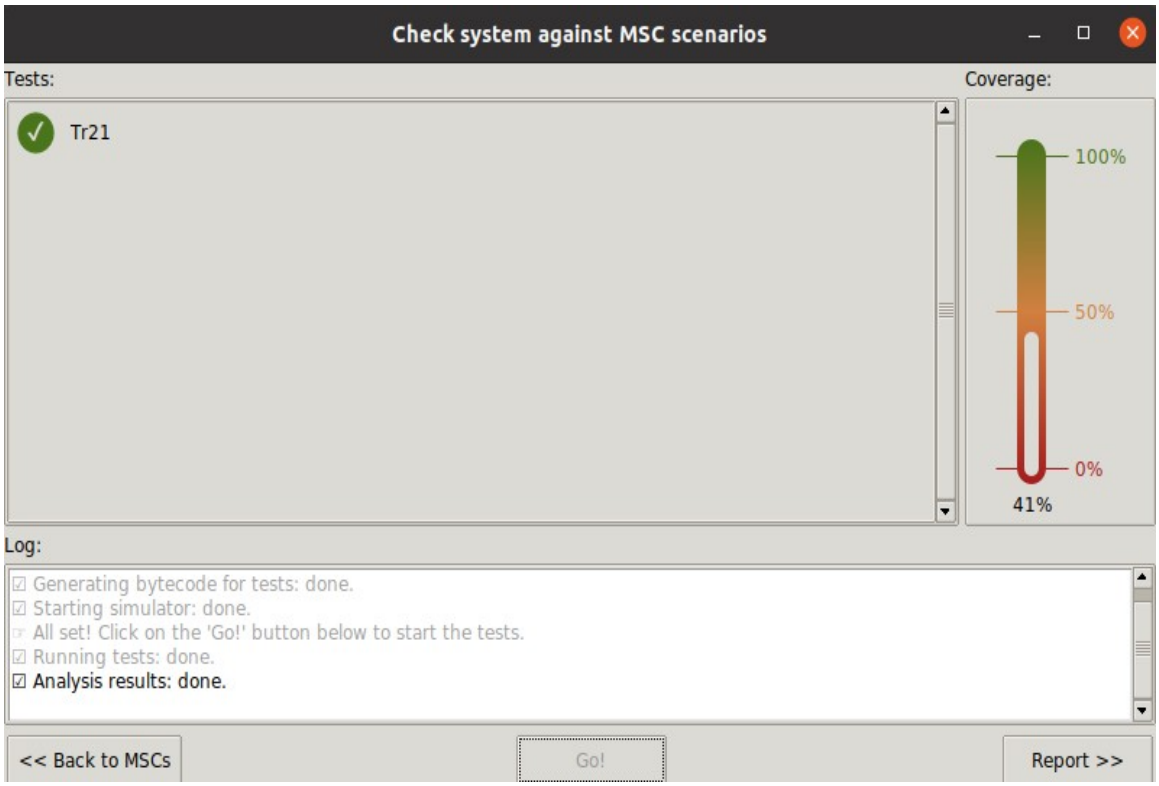


on a 3 traces Tr1,Tr2 et Tr3 :

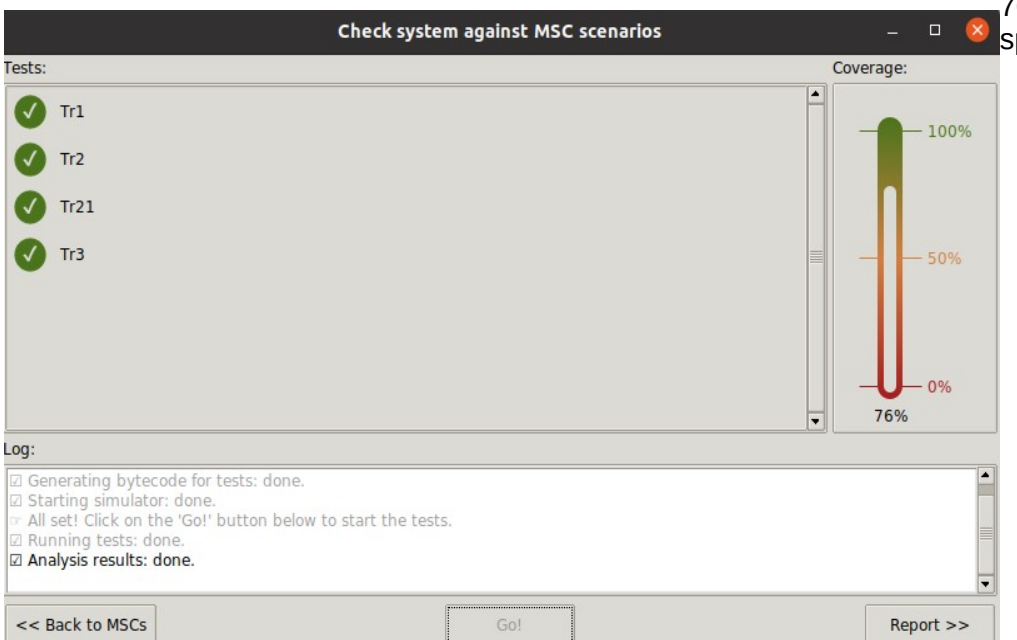
- Dans Tr1, on a crédité un montant correspondant à mon numéro d'étudiant (71811034)
- Dans Tr2, on a débité nbkn billets de 50 euros où nbkn vaut 71811034 mod 50=34.
- Dans Tr3, on fait insertcard, insertion d'un code faux (0) , aprés on insere un code correct(1234), donc dans ce cas on couvre les deux cas de la spécification.

Ensuite, j'ai fais 4 fois débit avec 4 billet(a10,a20,a50 et a100) et donc on couvre les 4 cas sur débit.
==>avec les3 traces ensemble, j'ai pu tester l'insertion de code (faux et correct), crédit et débit avec les 4 billets.
==>donc avec ça j'ai réussi à couvrir 76 % de la spécification.
Il en reste des comportements dans la spécification que mes traces ne couvrent pas comme le cas où l'utilisateur ne rentre pas de code et donc tempo sera écoulé.

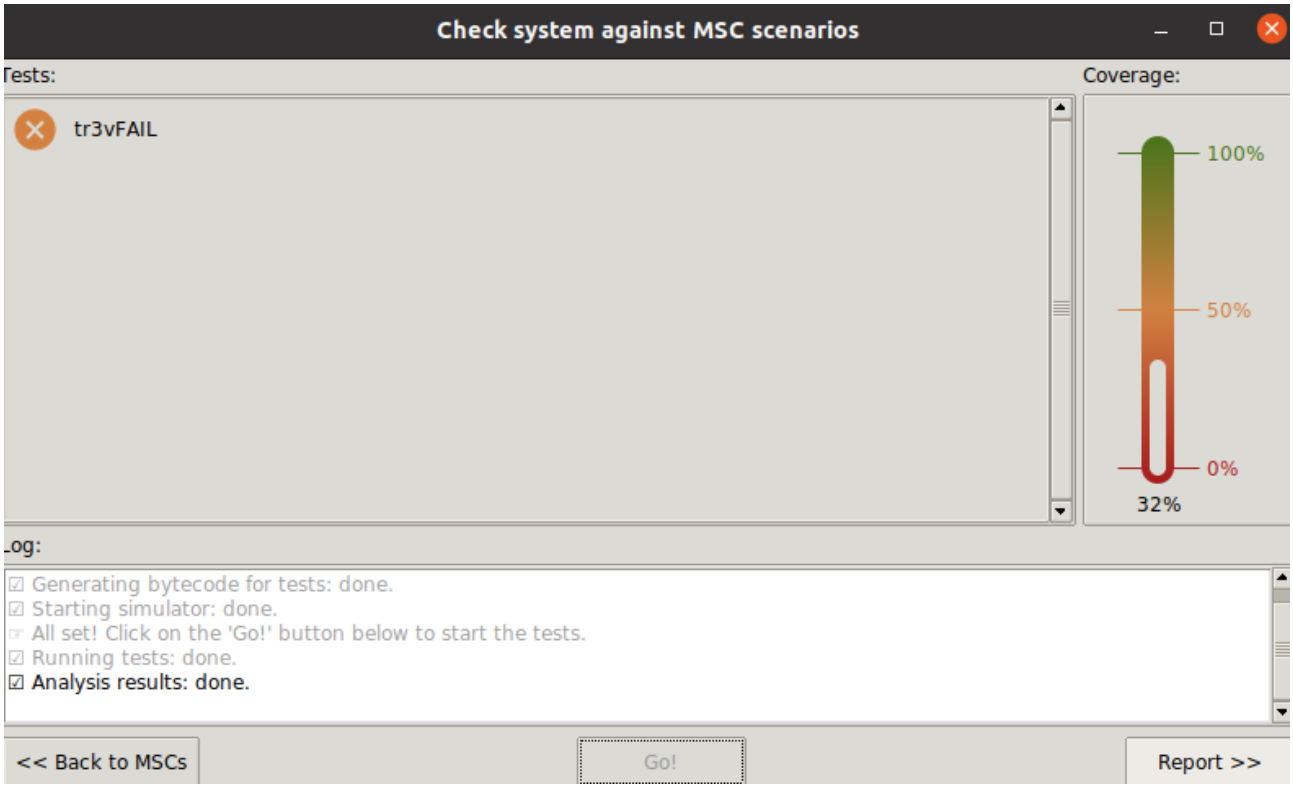
2.
a. ckeck avec la trace d'un camarade, j'ai eu un **verdit pass** et la trace couvre 41 % de la spécification.
car la spécification reste la même. Plus Si on regarde sa trace on trouve que le comportement pour lequel cette trace est faite est couvert par ma spécification, et elle contient aucun comportement qui n'est pas présent dans ma spécification.



On ajoutant la trace d'un camarade à mes traces j'ai eu un verdict pass et les traces couvrent 76 % de la spec :

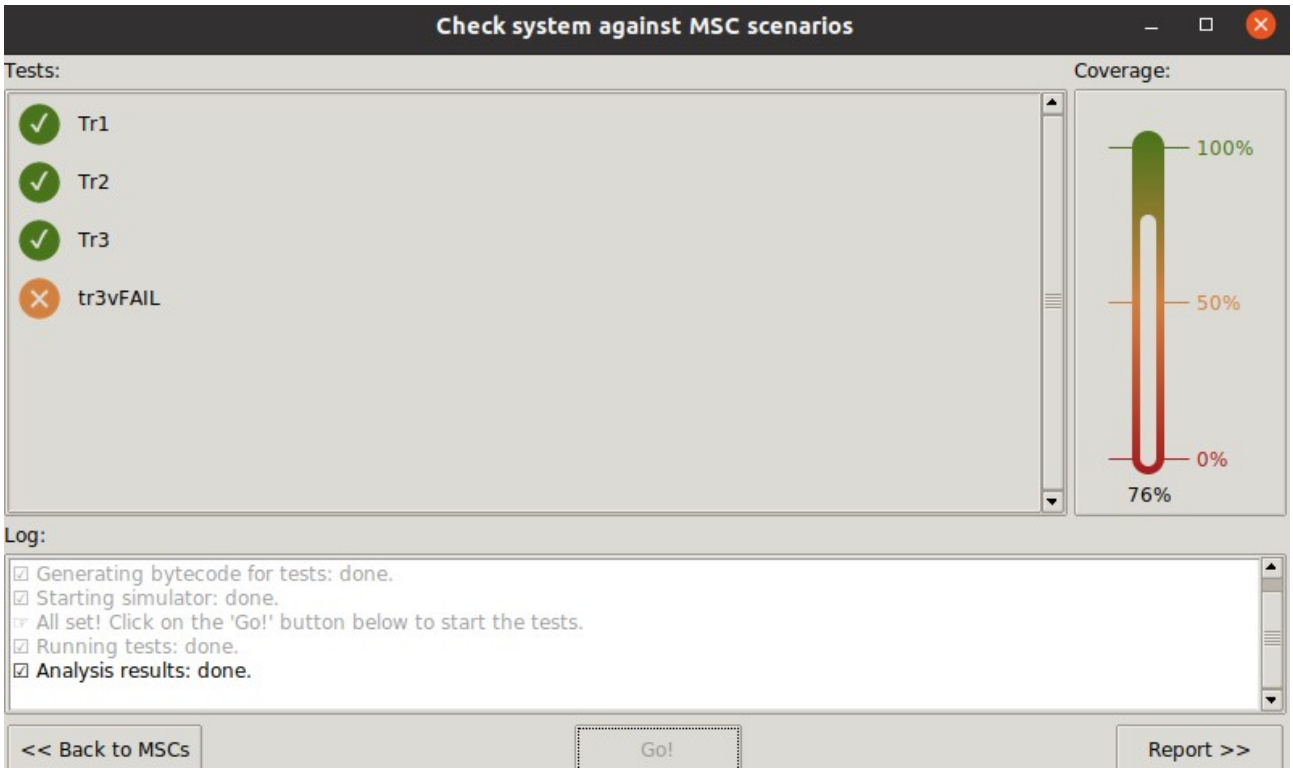


b. check de tr3FAIL sur ma spécification :



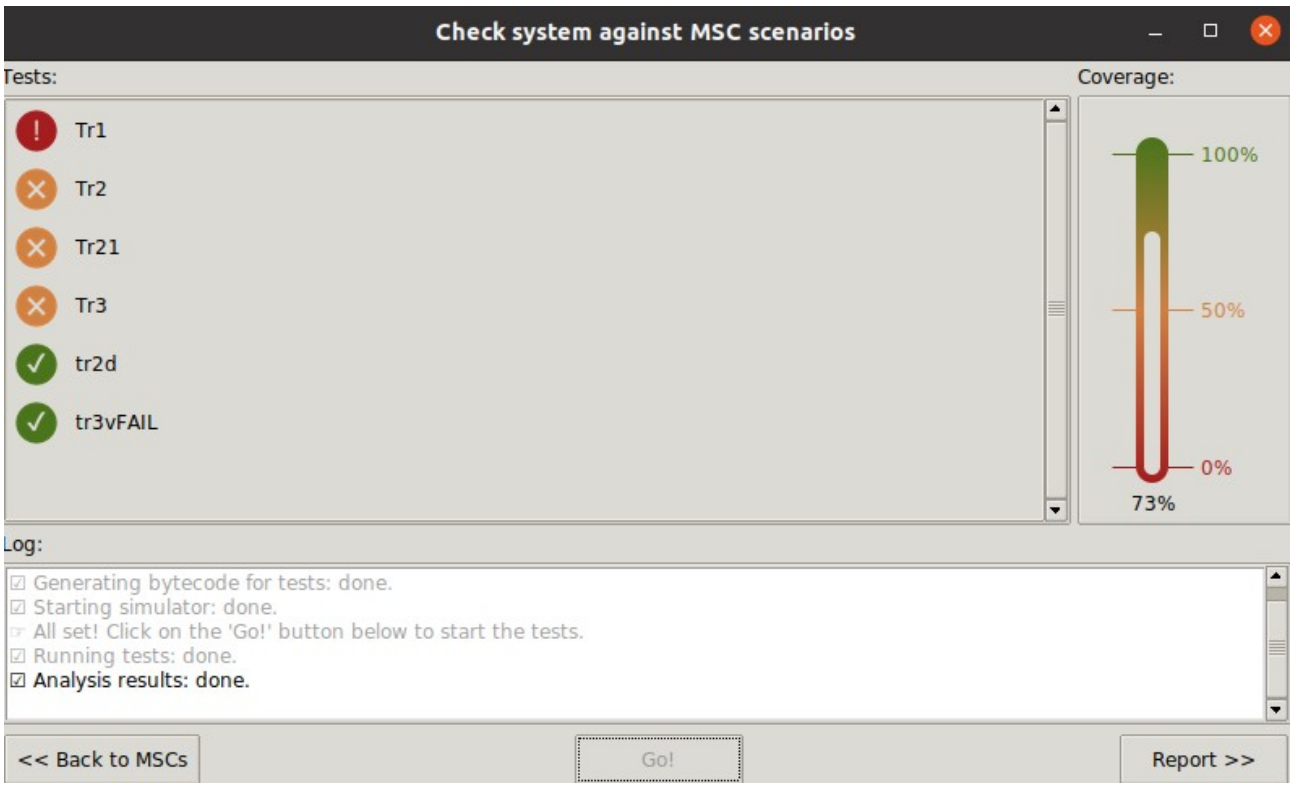
le check de tr3FAIL nous retourne un **verdict fail**
on a reçu deux signaux *bknote20* juste après la saisie de nombre de billets qu'on veut débiter, ce signal existe dans la spécification. Mais le comportement n'est pas souhaité puisque on a donné un paramètre de **numberbanknote=3** ce qui fait que le check échoue sur cette trace.
La trace couvre 32% de la spécification car cette trace couvre quelques comportements de notre spécification : nsertcard, debit …ect

c. check de tr3FAIL avec mes trace



le check échoue même en l'ajoutant à mes traces car le comportement de tr3vFAIL n'est pas présent dans ma spécification, et les traces ensemble couvre toujours 76 % de celle-ci.
Les comportement couvert par tr3vFAIL est déjà couvert par mes traces de début donc la couverture reste toujours la même (76%).

d.



Tr2 et Tr21 nous renvoie un verdict **fail :**

aprés avoir insérer la carte et le code, on débite avec numberbanknote=34 et 2 respectivement. mais sans le signal bknote qui doit être bknote50 dans notre cas que la spécification exige.

Tr3 nous renvoie un verdict **fail** :

pour la même raison que Tr2, en plus on a débiter plusieurs fois avec un paramètre *numberbankonote* diffèrent de 0 mais le signal *bknote* et non pressent.

Tr2d nous renvoie un verdict **PASS** :

tous les comportement qui sont représentés dans cette trace sont présents dans la spécification, et aucun comportement n'est étrange à la spécification. On remarque que contrairement aux traces précédentes dans cette trace on a le signal bknote :

o Au début on a débiter 2 billet de a10 et on a deux fois le signal bknote10.

o Puis 3 billet de a20 et on a trois fois le signal bknote20.

Tr1 nous retourne un verdict **error :**

cela est dû au fait qu'à la fin de ces traces on a *okcancel* et à laquelle on a une réponse *returncard* dans la première spécification, par contre dans la nouvelle spécification on aura pas de réponses pour cette requête , parce que *returncard* n'est pas présent dans la spécification.

Tr3VFAIL **pass :**

cette trace nous retourne **un pass** en faisant check avec les autres traces , on regardent le comportement de cette trace, on remarque en faisant debit on donne comme paramètre ***bknote=3****. Dans notre sépcification, on doit avoir 3 signaux bknote20(), alors qu'on a que 2 signaux bknote20().*

*Et puisque **Tr3VFAIL** s'exécute au même temps que **tr2d** dans laquelle on a **bknote=3** aussi. Donc, à la fin de test, on aura 3 signaux bknote20() pour les 2 trace mais qui ne sont pas visible dans Tr3vFAIL.*

```
module TTCN_TestsAndControl
  {
  import from TTCN_Declarations all;

  import from TTCN_Templates all;


  altstep RTDS_fail() runs on runsOn_atmsys
    {
    []cardchan.receive
      {
      setverdict(fail, "Fail in default altstep!");
      stop;
      };
    }

  testcase TC_Tr1() runs on runsOn_atmsys system atmsys
    {
    activate(RTDS_fail());
    map(self:cardchan, system:cardchan);
    cardchan.send(insertcard_tr2dMSG1);
    cardchan.send(insertcode_Tr1MSG2);
    cardchan.receive(errorcode_Tr1MSG3);
    cardchan.send(insertcode_tr2dMSG2);
    cardchan.send(credit_Tr1MSG5);
    cardchan.receive(creditamount_Tr1MSG6);
    cardchan.send(credita_Tr1MSG7);
    cardchan.receive(creditreceipt_Tr1MSG8);
    cardchan.receive(OKop_tr2dMSG10);
    cardchan.send(cancelop_tr2dMSG20);
    cardchan.receive(cancelopquestion_tr2dMSG21);
    cardchan.send(OKcancel_tr2dMSG22);
    cardchan.receive(returncard_Tr1MSG13);
    setverdict(pass);
    }

  testcase TC_Tr2() runs on runsOn_atmsys system atmsys
    {
    activate(RTDS_fail());
    map(self:cardchan, system:cardchan);
    cardchan.send(insertcard_tr2dMSG1);
    cardchan.send(insertcode_Tr1MSG2);
    cardchan.receive(errorcode_Tr1MSG3);
    cardchan.send(insertcode_tr2dMSG2);
    cardchan.send(debit_tr2dMSG3);
    cardchan.receive(debitamount_tr2dMSG4);
    cardchan.send(a50_Tr2MSG7);
```

```
testcase TC_tr2d_tr3vFAIL() runs on runsOn_atmsys system atmsys
  {
  activate(RTDS_fail());
  map(self:cardchan, system:cardchan);
  cardchan.send(insertcard_tr2dMSG1);
  cardchan.send(insertcode_tr2dMSG2);
  cardchan.send(debit_tr2dMSG3);
  cardchan.receive(debitamount_tr2dMSG4);
  cardchan.send(a10_tr2dMSG5);
  cardchan.send(numberbanknotes_tr2dMSG6);
  cardchan.receive(bknote10_tr2dMSG7);
  cardchan.receive(bknote10_tr2dMSG7);
  cardchan.receive(debitreceipt_tr2dMSG9);
  cardchan.receive(OKop_tr2dMSG10);
  cardchan.send(debit_tr2dMSG3);
  cardchan.receive(debitamount_tr2dMSG4);
  cardchan.send(a20_tr2dMSG13);
  cardchan.send(numberbanknotes_tr2dMSG14);
  cardchan.receive(bknote20_tr2dMSG15);
  cardchan.receive(bknote20_tr2dMSG15);
  alt {
    []cardchan.receive(bknote20_tr2dMSG15) {
      cardchan.receive(debitreceipt_tr2dMSG9);
      cardchan.receive(OKop_tr2dMSG10);
      cardchan.send(cancelop_tr2dMSG20);
      cardchan.receive(cancelopquestion_tr2dMSG21);
      cardchan.send(OKcancel_tr2dMSG22);
      }
    []cardchan.receive(debitreceipt_tr2dMSG9) {
      cardchan.receive(OKop_tr2dMSG10);
      cardchan.send(cancelop_tr2dMSG20);
      cardchan.receive(cancelopquestion_tr2dMSG21);
      cardchan.send(OKcancel_tr2dMSG22);
      cardchan.receive(returncard_Tr1MSG13);
      }
    }
  setverdict(pass);
  }
```

2.d)
Dans le code de TTCN3, on a un cas de test pour chaque trace :
Dans TR1 Dont on a tout les messages envoyé avec send et les messages reçus avec receive et soit :

-on reçoit **un verdict pass** à la fin de la trace.
-soit on déclenche RTDS_fail qui nous retourne un fail.

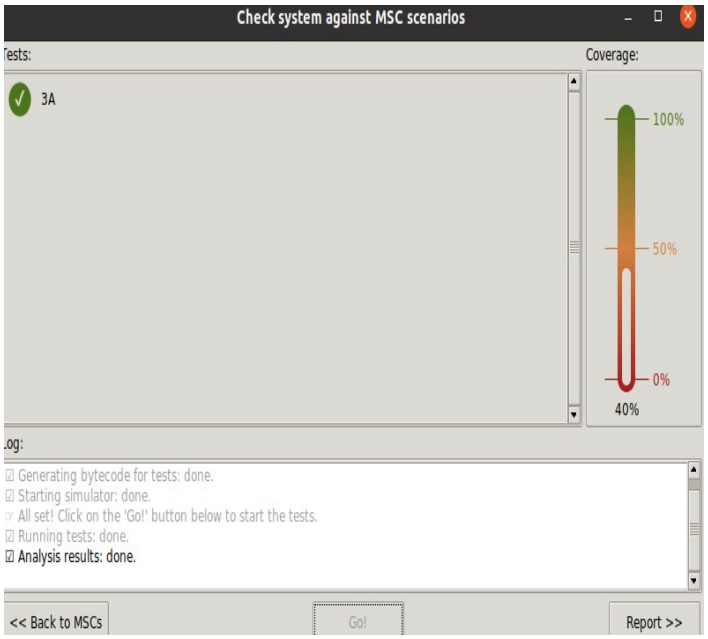
Dans TC_tr2d_tr3vFAIL : on a des send et des receive aussi mais avec deux alternative qui nous mènent vers un **verdict pass** ou sinon on active RTDS_fail() qui nous return le **verdict fail**
les 2 alternative sont :

-soit on reçoit le signal **debitreceipt** et dans ce cas la carte est retourné,donc ici on a le comportement de la trace **tr3vFAIL.**

-soit on reçoit **bknote20** et dans ce cas on reçoit le signal **okcancel.** Donc ici on a le comportement de la trace **tr2d.**

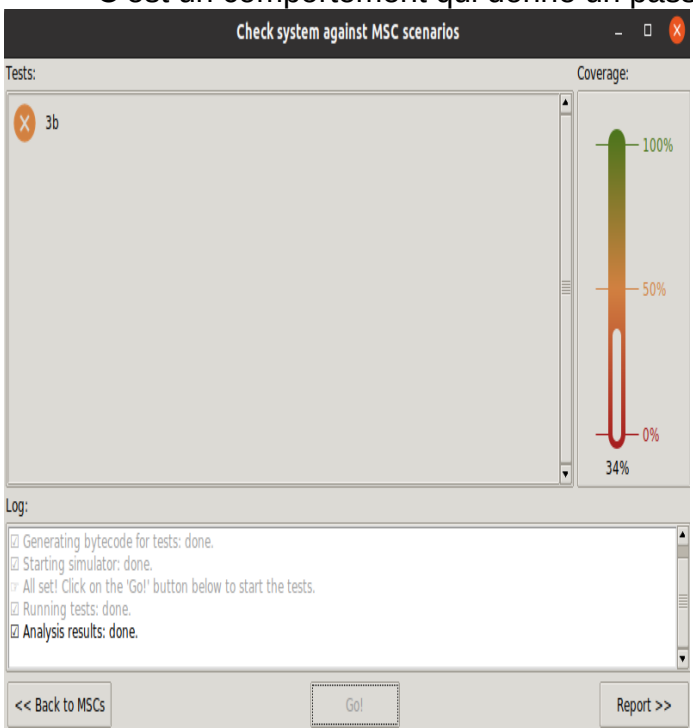
3-SCRIPTS TTCN3

a. PASS



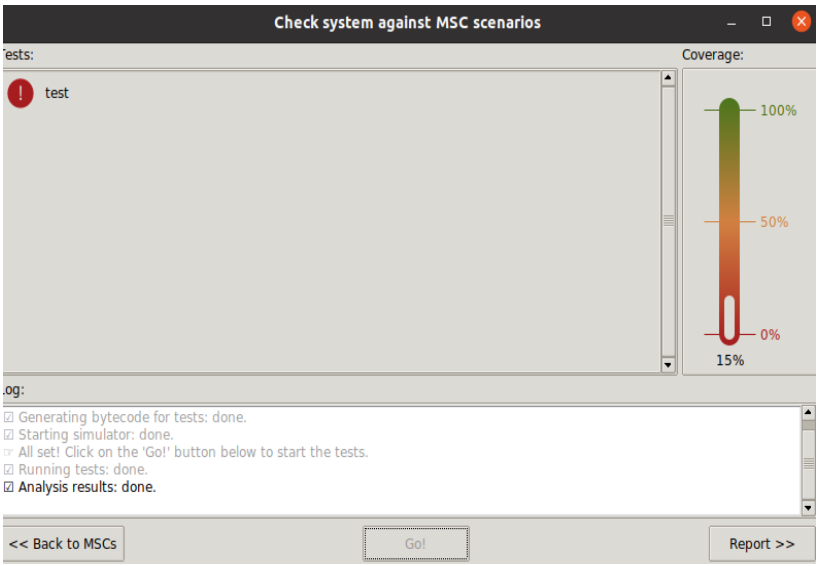

```
module TTCN_TestsAndControl
  {
  import from TTCN_Declarations all;

  import from TTCN_Templates all;


  altstep RTDS_fail() runs on runsOn_atmsys
    {
    []cardchan.receive
      {
      setverdict(fail, "Fail in default altstep!");
      stop;
      };
    }

  testcase TC_3A() runs on runsOn_atmsys system atmsys
    {
    activate(RTDS_fail());
    map(self:cardchan, system:cardchan);
    cardchan.send(insertcard_3AMSG1);
    cardchan.send(insertcode_3AMSG2);
    cardchan.send(debit_3AMSG3);
    cardchan.receive(debitamount_3AMSG4);
    cardchan.send(a10_3AMSG5);
    cardchan.send(numberbanknotes_3AMSG6);
    cardchan.receive(bknote10_3AMSG7);
    cardchan.receive(bknote10_3AMSG7);
    cardchan.receive(debitreceipt_3AMSG9);
    cardchan.receive(OKop_3AMSG10);
    cardchan.send(cancelop_3AMSG11);
    cardchan.receive(cancelopquestion_3AMSG12);
    cardchan.send(OKcancel_3AMSG13);
    cardchan.receive(returncard_3AMSG14);
    setverdict(pass);
    }
```

b.fail : Pour cette trace on a un fail car si un client demande un billet de 50 il ne doit pas recevoir un billet de 20 et c'est un comportement non attendu de notre SPEC3.
C'est un comportement qui donne un pass dans la 1ere specification.




```
testcase TC_3b() runs on runsOn_atmsys system atmsys
  {
  activate(RTDS_fail());
  map(self:cardchan, system:cardchan);
  cardchan.send(insertcard_3AMSG1);
  cardchan.send(insertcode_3AMSG2);
  cardchan.send(debit_3AMSG3);
  cardchan.receive(debitamount_3AMSG4);
  cardchan.send(a50_3bMSG5);
  cardchan.send(numberbanknotes_3bMSG6);
  cardchan.receive(debitreceipt_3AMSG9);
  cardchan.receive(OKop_3AMSG10);
  setverdict(pass);
  }
```

d.error : dans cette trace on a un error car dans la spécification SPEC3 on peut insérer un code erroné qu'un seul fois. Sinon on se bloque. Par contre la première spécification, on peut insérer le code 2 fois.
C'est pour cela qu'on a un error dans la spécification SPEC3.




```
module TTCN_TestsAndControl
  {
  import from TTCN_Declarations all;

  import from TTCN_Templates all;


  altstep RTDS_fail() runs on runsOn_atmsys
    {
    []cardchan.receive
      {
      setverdict(fail, "Fail in default altstep!");
      stop;
      };
    }

  testcase TC_test() runs on runsOn_atmsys system atmsys
    {
    activate(RTDS_fail());
    map(self:cardchan, system:cardchan);
    cardchan.send(insertcard_testMSG1);
    cardchan.send(insertcode_testMSG2);
    cardchan.receive(errorcode_testMSG3);
    cardchan.send(insertcode_testMSG4);
    setverdict(pass);
    }

  control
    {
    execute(TC_test());
    }
  }
```