Projet SF

RAPPORT

Modèlisation et Vérification du comportement de véhicules automatiques sur un pont à voie unique

Réalisé par ***DOAN Cao Sang***

5 décembre 2015

Chapitre 1

Propriété

*L*e programme doit satisfaire les propriétés suivantes :

1. Il n'y a pas de collision (i.e. deux véhicules circulants en sens inverse) sur le pont.
2. Un véhicule qui arrive est certain de passer sur le pont à l'issue d'une durée bornée.

Chapitre 2

Explication

2.1 Question 1.1

2.1.1 Vaa

VerificationA est une place sortie, qui lie le composant *Vaa* avec le contrôlleur *CTRLP*, lorsque un véhicule A veut traverser le pont, d'abord il communique avec le contrôlleur *CTRLP* pour avoir son autorisation.

OKControlleurA est une place entrée qui lie entre *Vaa* et *CTRLP*, lorsque le *Vaa* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vaa* puisse passer.

OKPont est une place entrée qui lie le *P* avec le *Vaa*, le *P* marque cette place lors qu'il reçoit l'autorisation de contrôlleur *CTRLP*.

AuRevoirControlleurA est une place sortie, le *Vaa* marque cette place dès qu'il sort du *P*.

2.1.2 Vab

VerificationB est une place sortie, qui lie le composant *Vab* avec le contrôlleur *CTRLP*, lorsque un véhicule B veut traverser le pont, d'abord il communique avec le contrôlleur *CTRLP* pour avoir son autorisation.

OKControlleurB est une place entrée qui lie entre *Vab* et *CTRLP*, lorsque le *Vab* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vab* puisse passer.

OKPont est une place entrée qui lie le *P* avec le *Vab*, le *P* marque cette place lors qu'il reçoit l'autorisation de contrôlleur *CTRLP*.

AuRevoirControlleurB est une place sortie, le *Vab* marque cette place dès qu'il sort du *P*.

2.1.3 Pont

CapaciteControlleur est une place sortie, qui lie le composant *P* avec le contrôlleur *CTRLP* pour lui signaler sa capacité restante.

OKPont est une place sortie qui lie le *P* avec les 2 composants *Vaa* et *Vab*, le *P* marque cette place lorsque sa capacité reste suffisante.

2.1.4 CTRLP

VerificationA est une place entrée, qui lie le composant *Vaa* avec le contrôlleur *CTRLP*, lorsque un véhicule A veut traverser le pont, le *CTRLP* doit d'abord, recevoir sa demande.

VerificationB est une place entrée, qui lie le composant *Vab* avec le contrôlleur *CTRLP*, lorsque un véhicule B veut traverser le pont, le *CTRLP* doit d'abord, recevoir sa demande.

OKControlleurA est une place sortie qui lie entre *Vaa* et *CTRLP*, lorsque le *Vaa* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vaa* puisse passer.

OKControlleurB est une place sortie qui lie entre *Vab* et *CTRLP*, lorsque le *Vab* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vab* puisse passer.

CapaciteControlleur est une place entrée, qui lie le composant *P* avec le contrôlleur *CTRLP* pour savoir la capacité restante du *P*.

AuRevoirControlleurA est une place entrée, cette place est marqué par le *Vaa*.

AuRevoirControlleurB est une place entrée, cette place est marqué par le *Vab*.

2.2 Question 1.2

Cette composant contient 8 places, dont 4 ont pour le rôle d'interface, et 3 transitions.

La place **VehiculeA** modélise le véhicule A lors qu'il arrive devant le pont. S'il veut passer sur le pont, d'abord, il communique avec le contrôlleur par la transition *demanderEntrerA*, cette transition met un jeton dans l'interface **VerificationA** qui lie avec le contrôlleur *CTRLP* et passe à l'état **AttenteAutorisationA**. Lorsqu'il reçoit l'autorisation du contrôlleur *CTRLP* via l'interface **OKControlleurA** et celle du pont *P* via l'interface **OKPont**, il peut traverser le pont, en passant à l'état **DansPontA**. Ensuite, quand il sort, il marque la place d'interface **AuRevoirControlleurA** et aussi passe à l'état **FiniA** par la transition *sortPontA*.

La sécurité est garantie car, pour passer à l'état **AttenteAutorisationA** le véhicule doit communiquer avec le contrôlleur *CTRLP*. Pour entrer, il doit avoir l'autorisation du contrôlleur *CTRLP* et consommer le jeton dans la place d'interface **OKPont**, cette place modélise la consommation de la capacité du pont *P*. Enfin, il signale sa sortie du pont au contrôlleur *CTRLP* en mettant un jeton dans la place **AuRevoirControlleurA**.

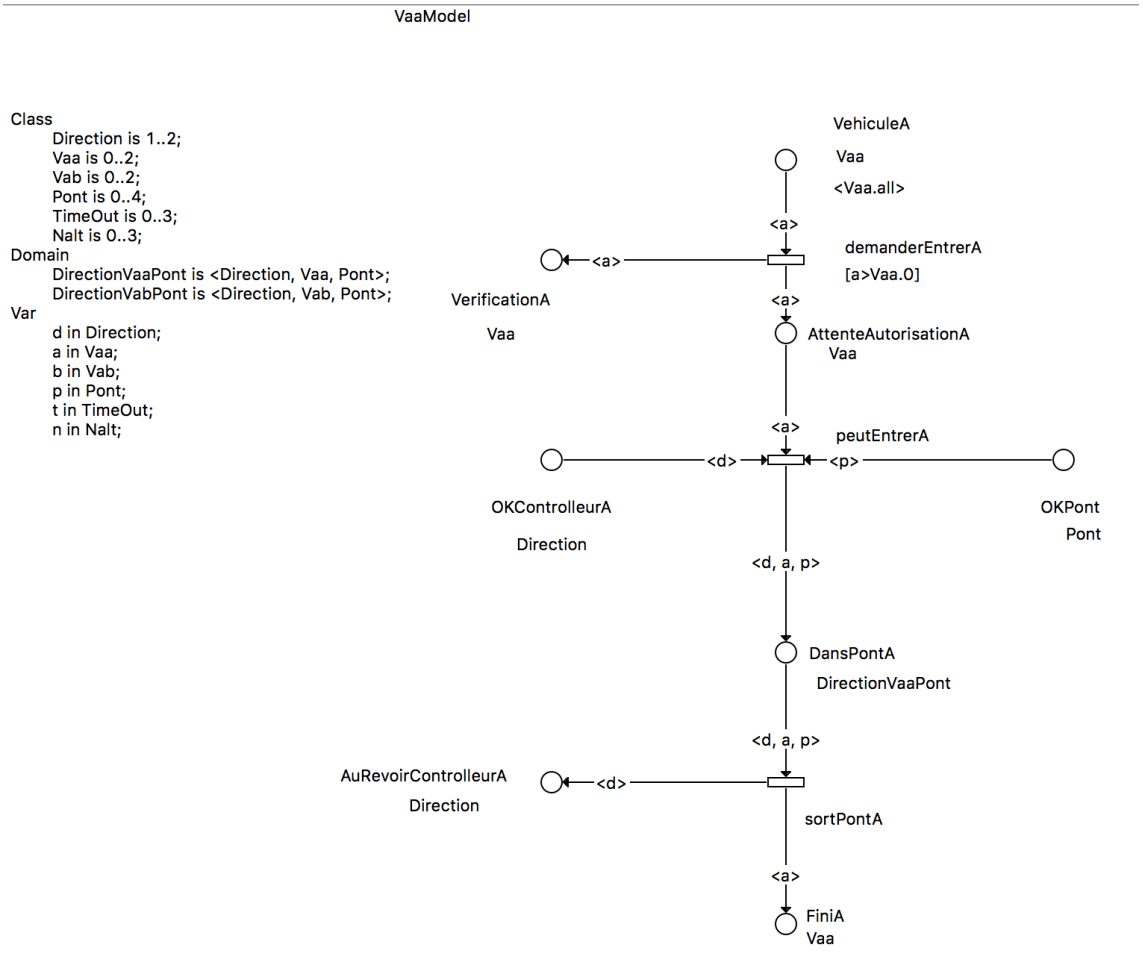

FIGURE 2.1 – Le composant modélise le véhicule automatisée A.

2.3 Question 1.3

Cette composant est identique avec celle de véhicule A, contient 8 places, dont 4 ont pour le rôle d'interface, et 3 transitions.

La place **VehiculeB** modélise le véhicule B lors qu'il arrive devant le pont. S'il veut passer sur le pont, d'abord, il communique avec le contrôlleur par la transition *demanderEntrerB*, cette transition met un jeton dans l'interface **VerificationB** qui lie avec le contrôlleur *CTRLP* et passe à l'état **AttenteAutorisationB**. Lorsqu'il reçoit l'autorisation du contrôlleur *CTRLP* via l'interface **OKControlleurB** et celle du pont *P* via l'interface **OKPont**, il peut traverser le pont, en passant à l'état **DansPontB**. Ensuite, quand il sort, il marque la place d'interface **AuRevoirControlleurB** et aussi passe à l'état **FiniB** par la transition *sortPontB*.

La sécurité est garantie car, pour passer à l'état **AttenteAutorisationB** le véhicule doit communiquer avec le contrôlleur *CTRLP*. Pour entrer, il doit avoir l'autorisation du contrôlleur *CTRLP* et consommer le jeton dans la place d'interface **OKPont**, cette place modélise la consommation de la capacité du pont *P*. Enfin, il signale sa sortie du pont au contrôlleur *CTRLP* en mettant un jeton dans la place **AuRevoirControlleurB**.

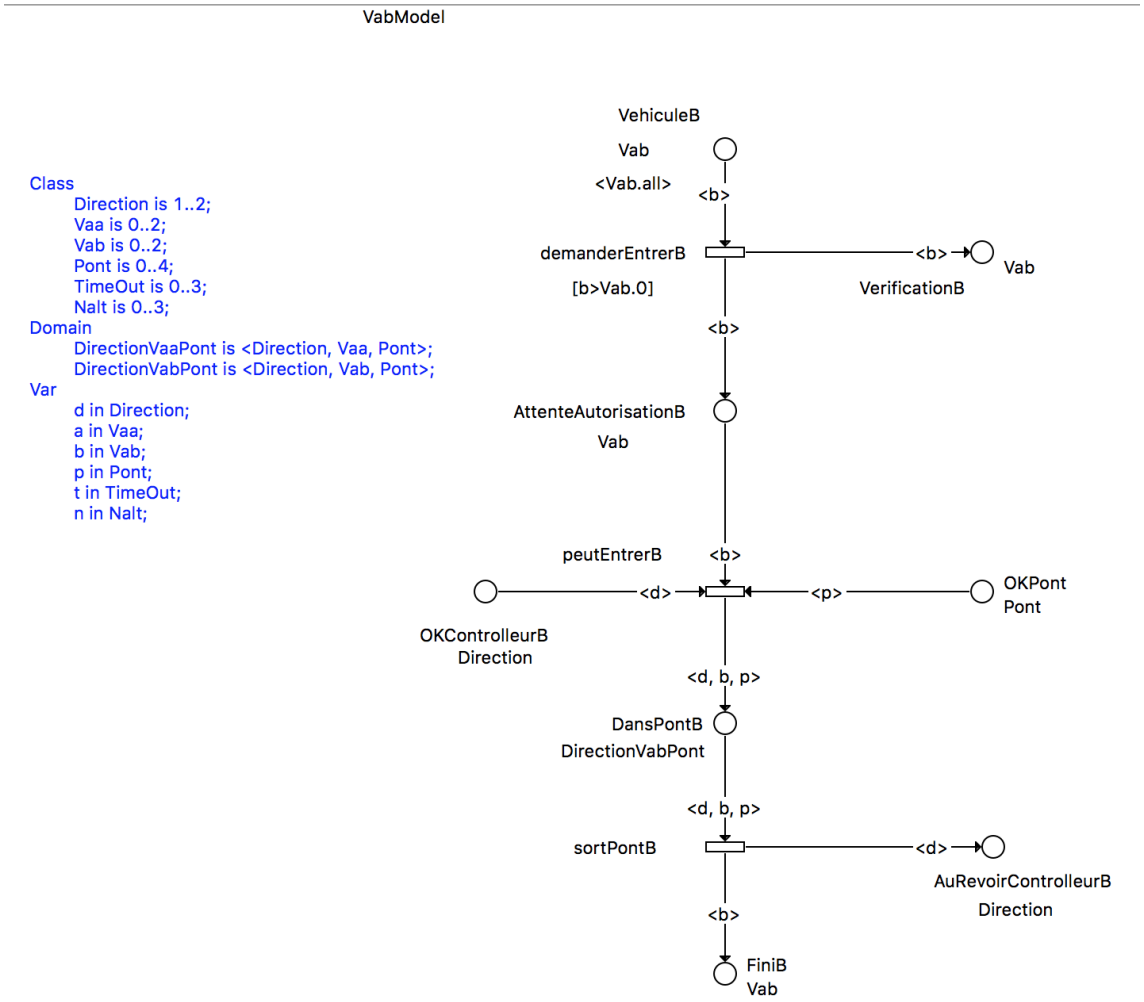

FIGURE 2.2 – Le composant modélise le véhicule automatisée B.

2.4 Question 1.4

Le comportement d'un pont est simple, il notifie sa capacité au contrôlleur *CTRLP* et diminue si consomé par un véhicule. Donc, il contient 3 places, dont 2 interfaces et une transition.

La transition *notifieCapacite* marque 2 interfaces **CapaciteControlleur** et **OKPont** par 2 jetons, 1 pour chaque.

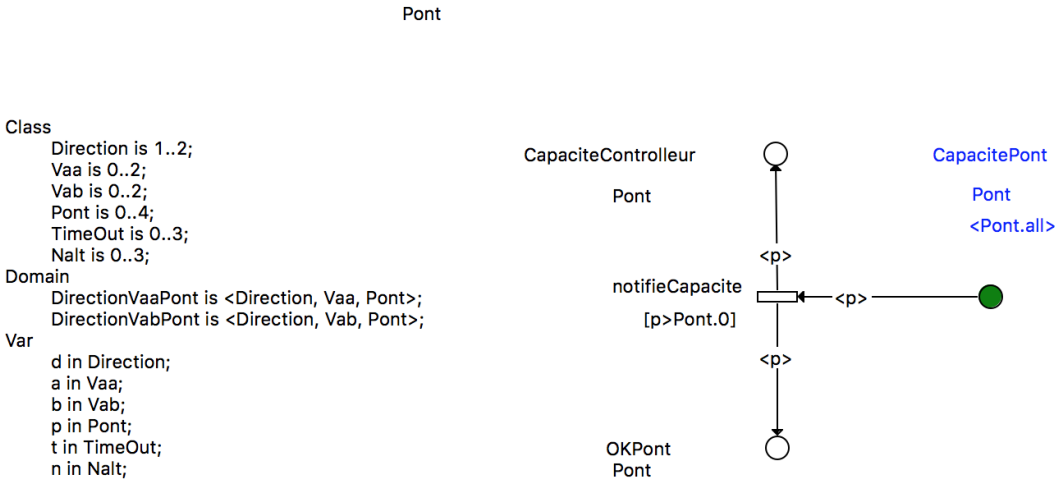

FIGURE 2.3 – Le composant modélise le pont.

2.5 Question 1.5

Ce composant contient 12 places dont 7 interfaces, et 6 transitions.

Lorsque le *CTRLP* reçoit une demande d'entrer d'une véhicule *Vaa* (idem, *Vab*), il vérifie le sens actuel (initilise à 1, direction de A vers B) est correspondant à celui de *Vaa*, le timeout restant, le nombre *Nalt* restant et la capacité *CapaP* restant du pont. Si ces conditions sont satisfaits, alors le contrôlleur *CTRLP* donne l'autorisation au véhicule en attente. Quand le véhicule sort du pont, le contrôlleur reçoit son signal.

Le timeout est modélisé par une place nommée *TimeOut* avec une seule transition *ecoule*. Le *TimeOut* décrémente une unité de temps automatique indépendamment avec le système. Lors de l'expiration de *TimeOut*, il réinitialise le *Nalt*, lui même, et change la direction actuelle. Idem pour le *Nalt*, si $Nalt = 0$, il réinitialise le *TimeOut*, *Nalt* et change la direction.

Quand la capacité du pont est à 0, alors aucun véhicule peut utiliser ce pont.

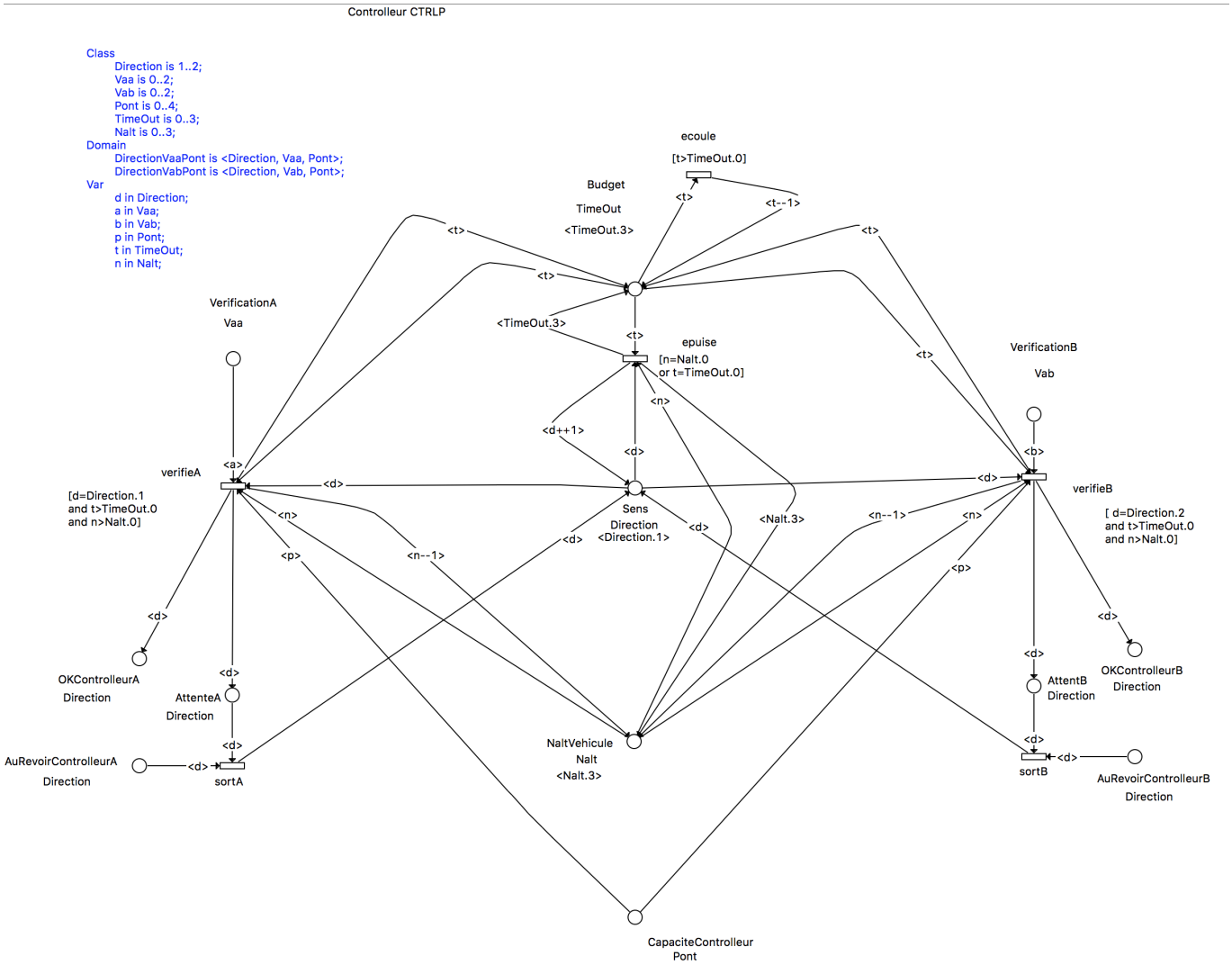

FIGURE 2.4 – Le composant modélise le pont.

2.6 Question 1.6

[Assemblage]

```
query node ((card(DansPontA)==1) and (card(DansPontB)==1))



============================================================
This service relies on the prod model checker developped
at the Helsinki University of technology  (laboratory of
computer  science,  http://www.tcs.hut.fi/Software/prod)
============================================================

Integration in CosyVerif by F. Kordon (UPMC, 2013)

Installing Prod
Installing gph2dot
Installing model data for persistent storage
PN syntax checker by X. Bonnaire 1992-1996, A. Diagne 1996-1997, I. Mounier 1993-2002, J.-L. M
Statistics for this net
Number of places      22
Number of Queues      0
Number of Transitions 13
Number of arcs        55
Generating prod format
Generating and compiling code for the model
Generating the state space (this may take a while;-)

---
Statistics about the reachability graph provided by prod are shown below
Number of nodes: 283688
Number of arrows: 1033700
Number of terminal nodes: 0
Number of nodes that have been completely processed: 283688
Number of strongly connected components: 229305
Number of nontrivial terminal strongly connected components: 1
Evaluating the provided CTL formula
0 paths
Built set %1
------------------------------------------------

============================================================
Service completed OK
============================================================

Debug information
============================================================
CosyVerif directory  -> /tmp/alligator-5454896919309414835
```

```
Persostent directory -> /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b
============================================================
Execution trace follows below
+ '[' '!' -f prod_model ']'
+ echo 'PN syntax checker by X. Bonnaire 1992-1996, A. Diagne 1996-1997, I. Mounier 1993-2002,
+ /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b/FkSandrine -s -i prod_8763807886865996
+ '[' '!' -f .stats ']'
++ wc -c
++ grep colores .qualif
+ CLASS=8
+ sed -e s/://
+ cat .stats
+ echo 'Generating prod format'
+ /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b/matrice -s -i camix_model -m rdp_matri
++ tr -s ' '
++ cut -d ' ' -f 2
++ wc -l rdp_matrix
+ CHECK=rdp_matrix
+ '[' rdp_matrix = 0 ']'
++ grep NAME_NOT_IDENTIFIER matrice_output
+ '[' '' ']'
+ '[' -f prod_model.gph ']'
+ echo 'Generating and compiling code for the model'
+ prod prod_model.init
+ '[' -f prod_model ']'
+ echo 'Generating the state space (this may take a while;-)'
+ ./prod_model -m 1024000

./prod_model.aws is used directly

./prod_model.gph is used directly

./prod_model.adr is used directly
+ echo statistics
+ strong prod_model
+ sed -e s/.#//g
+ probe prod_model
+ echo ---
+ echo 'Statistics about the reachability graph provided by prod are shown below'
+ cat statistics.data
+ echo 'Evaluating the provided CTL formula'
+ cat ctl.prb prod_801592428106530532.ctlprod
+ sed -e 's/.*#//g'
+ probe prod_model
+ cat prod_formula_result

query node (AttenteAutorisationA == FiniA) // OK trace
```

```
=========================================================
This service relies on the prod model checker developped
at the Helsinki University of technology  (laboratory of
computer  science,  http://www.tcs.hut.fi/Software/prod)
=========================================================

Integration in CosyVerif by F. Kordon (UPMC, 2013)

Prod binaries checked OK
Gph2dot checked OK
Reachability Graph already computed, retrieving data
---
Statistics about the reachability graph provided by prod are shown below
Number of nodes: 283688
Number of arrows: 1033700
Number of terminal nodes: 0
Number of nodes that have been completely processed: 283688
Number of strongly connected components: 229305
Number of nontrivial terminal strongly connected components: 1
Evaluating the provided CTL formula


26672 paths
Built set %1
-----------------------------------------------

=========================================================
Service completed OK
=========================================================

Debug information
=========================================================
CosyVerif directory  -> /tmp/alligator-2047358861068064411
Persostent directory -> /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b
=========================================================
Execution trace follows below
+ '[' '!' -f prod_model ']'
+ '[' -f prod_model.gph ']'
+ echo 'Reachability Graph already computed, retrieving data'
+ echo ---
+ echo 'Statistics about the reachability graph provided by prod are shown below'
+ sed -e s/0#// statistics.data
+ echo 'Evaluating the provided CTL formula'
+ cat ctl.prb prod_1231987064332949103.ctlprod
+ sed -e 's/.*#//g'
+ probe prod_model
+ cat prod_formula_result
```

```
query node (card(Sens) == 2) // oK 0 path


=========================================================
This service relies on the prod model checker developped
at the Helsinki University of technology  (laboratory of
computer  science,  http://www.tcs.hut.fi/Software/prod)
=========================================================

Integration in CosyVerif by F. Kordon (UPMC, 2013)

Prod binaries checked OK
Gph2dot checked OK
Reachability Graph already computed, retrieving data
---
Statistics about the reachability graph provided by prod are shown below
Number of nodes: 283688
Number of arrows: 1033700
Number of terminal nodes: 0
Number of nodes that have been completely processed: 283688
Number of strongly connected components: 229305
Number of nontrivial terminal strongly connected components: 1
Evaluating the provided CTL formula
0 paths
Built set %1
-----------------------------------------------


=========================================================
Service completed OK
=========================================================

Debug information
=========================================================
CosyVerif directory  -> /tmp/alligator-8223995637312865004
Persostent directory -> /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b
=========================================================
Execution trace follows below
+ '[' '!' -f prod_model ']'
+ '[' -f prod_model.gph ']'
+ echo 'Reachability Graph already computed, retrieving data'
+ echo ---
+ echo 'Statistics about the reachability graph provided by prod are shown below'
+ sed -e s/0#// statistics.data
+ echo 'Evaluating the provided CTL formula'
+ cat ctl.prb prod_1904795993131222933.ctlprod
+ sed -e 's/.*#//g'
+ probe prod_model
+ cat prod_formula_result
```

```
query node ((card(FiniA) == 2) and (card(FiniB) == 2)) // OK

=========================================================
This service relies on the prod model checker developped
at the Helsinki University of technology  (laboratory of
computer  science,  http://www.tcs.hut.fi/Software/prod)
=========================================================

Integration in CosyVerif by F. Kordon (UPMC, 2013)

Prod binaries checked OK
Gph2dot checked OK
Reachability Graph already computed, retrieving data
---
Statistics about the reachability graph provided by prod are shown below
Number of nodes: 283688
Number of arrows: 1033700
Number of terminal nodes: 0
Number of nodes that have been completely processed: 283688
Number of strongly connected components: 229305
Number of nontrivial terminal strongly connected components: 1
Evaluating the provided CTL formula

40 paths
Built set %1
-----------------------------------------------

=========================================================
Service completed OK
=========================================================

Debug information
=========================================================
CosyVerif directory  -> /tmp/alligator-8573433729245045077
Persostent directory -> /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b
=========================================================
Execution trace follows below
+ '[' '!' -f prod_model ']'
+ '[' -f prod_model.gph ']'
+ echo 'Reachability Graph already computed, retrieving data'
+ echo ---
+ echo 'Statistics about the reachability graph provided by prod are shown below'
+ sed -e s/0#// statistics.data
+ echo 'Evaluating the provided CTL formula'
+ cat ctl.prb prod_2927488966804921096.ctlprod
+ sed -e 's/.*#//g'
```

```
+ probe prod_model
+ cat prod_formula_result


query node ((DansPontA == <.1.> and DansPontB == <.1.>) or (DansPontA == <.1.> and DansPontB =

============================================================
This service relies on the prod model checker developped
at the Helsinki University of technology  (laboratory of
computer  science,  http://www.tcs.hut.fi/Software/prod)
============================================================

Integration in CosyVerif by F. Kordon (UPMC, 2013)

Prod binaries checked OK
Gph2dot checked OK
Reachability Graph already computed, retrieving data
---
Statistics about the reachability graph provided by prod are shown below
Number of nodes: 283688
Number of arrows: 1033700
Number of terminal nodes: 0
Number of nodes that have been completely processed: 283688
Number of strongly connected components: 229305
Number of nontrivial terminal strongly connected components: 1
Evaluating the provided CTL formula
0 paths
Built set %1
------------------------------------------------


============================================================
Service completed OK
============================================================

Debug information
============================================================
CosyVerif directory  -> /tmp/alligator-6888013607560646498
Persostent directory -> /tmp/ProdInCosyVerif_f369dd251b6b1eac4a8afd256ef6d33b
============================================================
Execution trace follows below
+ '[' '!' -f prod_model ']'
+ '[' -f prod_model.gph ']'
+ echo 'Reachability Graph already computed, retrieving data'
+ echo ---
+ echo 'Statistics about the reachability graph provided by prod are shown below'
+ sed -e s/0#// statistics.data
+ echo 'Evaluating the provided CTL formula'
+ cat ctl.prb prod_3538640120971840321.ctlprod
```

```
+ sed -e 's/.*#//g'
+ probe prod_model
+ cat prod_formula_result
```