Dépliage efficace de réseaux de Petri colorés

Alban Linard
Alban.Linard@lip6.fr

Laboratoire d'Informatique de Paris 6
Systèmes Répartis et Coopératifs

2 décembre 2005

Plan

Introduction

Contexte

- modélisation par réseaux colorés (Well-Formed)
- certains outils nécessitent des réseaux P/T
- l'opération passant d'un réseau coloré à P/T : le **dépliage**

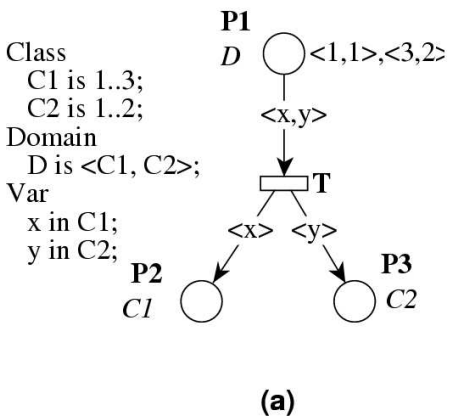
Problème traité

une explosion du nombre de places et transitions dépliées est possible

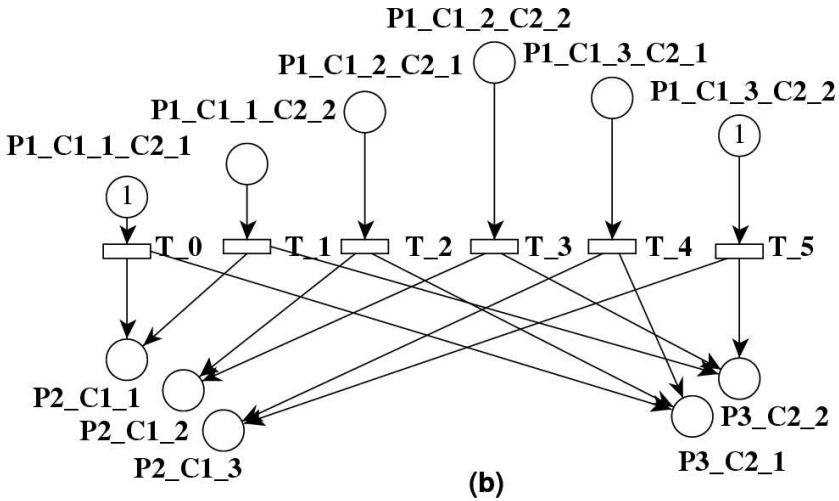
Solution

- certaines parties du réseau déplié sont mortes
- le réseau est optimisé en supprimant ces parties

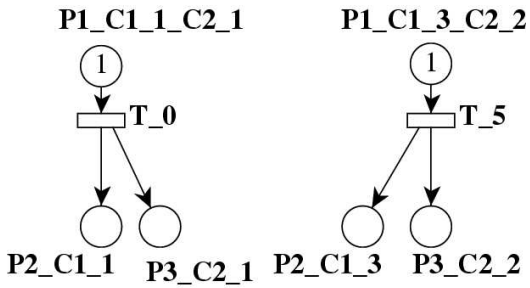
Exemple de dépliage



(a)



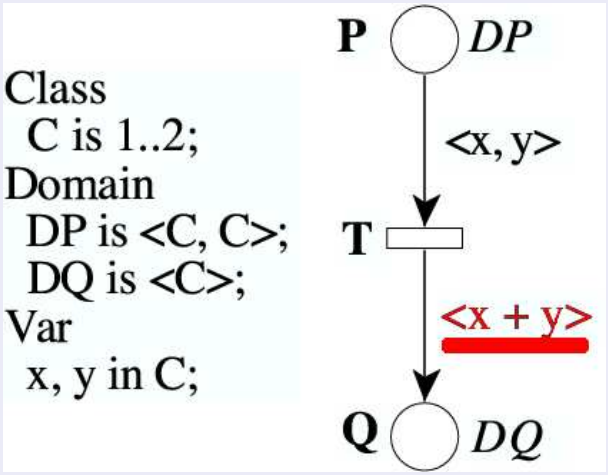
(b)



(c)

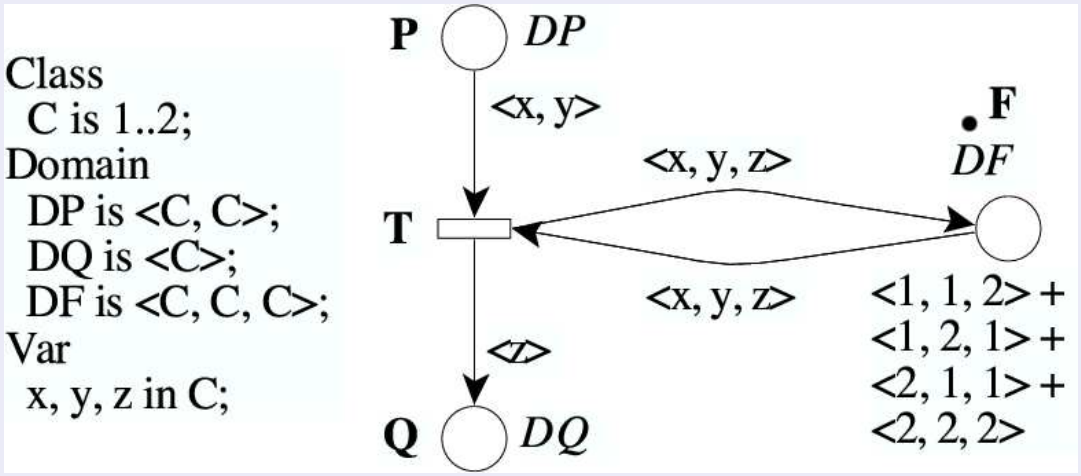
Exemple de problème

Problème



- représentation de fonctions dans un réseau de Petri

Solution



- le dépliage génère beaucoup de transitions : en n^3 (n nombre d'éléments d'une classe)

Solution proposée

Problème

- l'optimisation nécessite une représentation de l'ensemble du réseau déplié
- la représentation concrète est possible dans certains cas

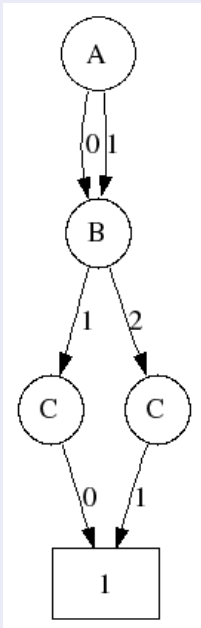
- représentation symbolique du réseau déplié
- optimisations appliquées à cette représentation :
  - suppression des transitions de garde fausse
  - **verrou maximal non marqué :**
    - ★ ensemble de places qui ne peut pas gagner de jetons de l'extérieur
    - ★ pas de marquage initial
      $\Rightarrow$ jamais marqué
      $\Rightarrow$ partie morte du réseau
  - suppression des places marquées orphelines
  - ...
- puis génération du réseau déplié concret

Data Decision Diagrams

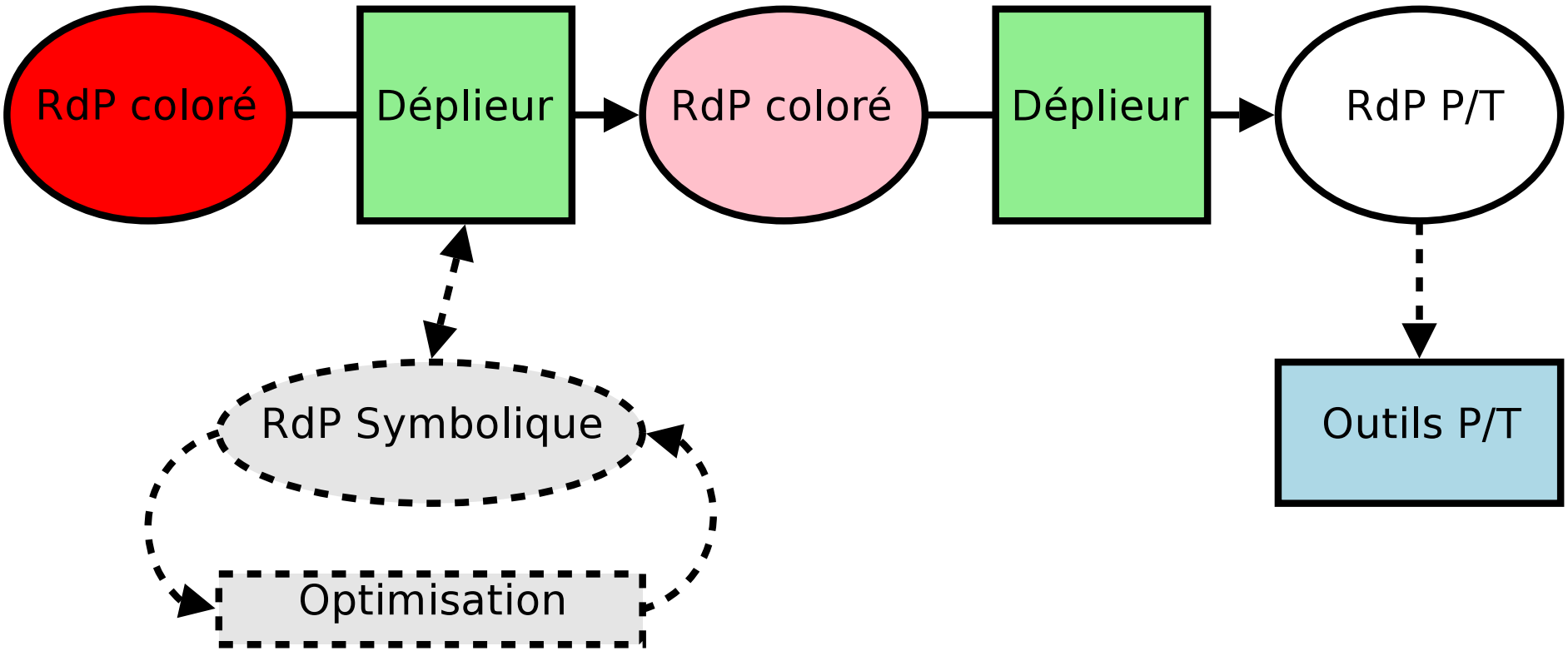
- représentation compacte de gros ensembles de vecteurs d'entiers
- bibliothèque (LIP6/LABRI) : J.-M. Couvreur (LABRI), E. Encrenaz (LIP6) et D. Poitrenaud (LIP6)
- deux types d'opérations :
  - opérations ensemblistes :
    - ★ union ($+$)
    - ★ intersection ($*$)
    - ★ difference ($\backslash$)
  - opérations locales (homomorphismes) :
    - ★ fonction pour chaque nœud (var, val) de l'arbre
    - ★ DDD pour le cas 1
    - ★ opérations d'union ($+$) et composition ($\circ$)

Exemple

- $a \xrightarrow{0} b \xrightarrow{1} c \xrightarrow{0} 1$
- $a \xrightarrow{1} b \xrightarrow{1} c \xrightarrow{0} 1$
- $a \xrightarrow{0} b \xrightarrow{2} c \xrightarrow{1} 1$
- $a \xrightarrow{1} b \xrightarrow{2} c \xrightarrow{1} 1$



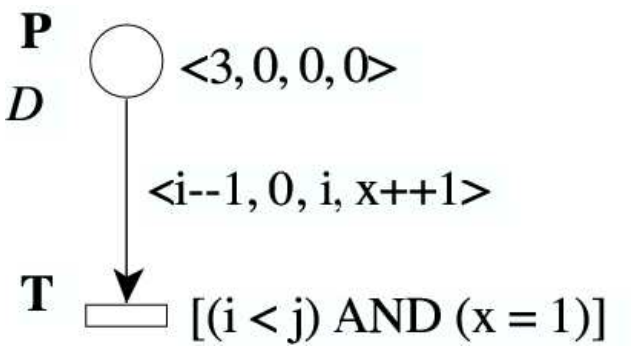
Contexte logiciel



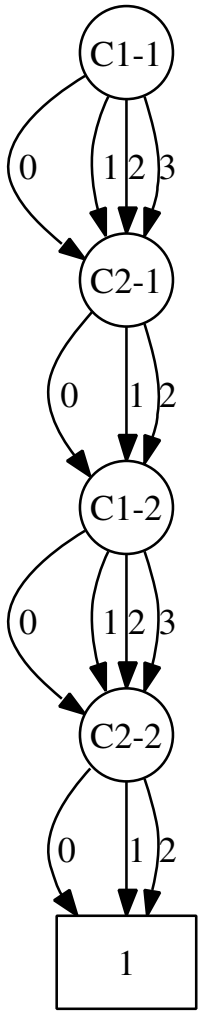
Représentation symbolique

Place

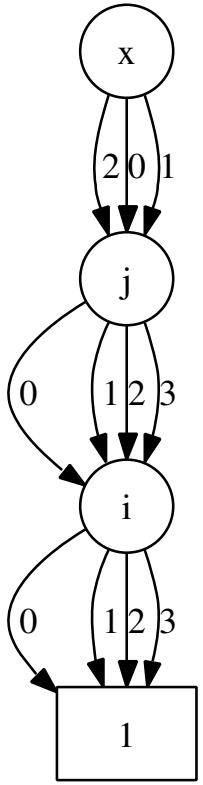
- une variable de DDD par classe du domaine de la place
- une variable de DDD par variable de transition
- chaque chemin correspond à une place ou une transition dépliée



Place **P** :



Transition **T** :



Construction de la représentation concrète

Plusieurs étapes à réaliser :

- pour chaque chemin d'un DDD de place,
  construire une place dépliée correspondante
- pour chaque chemin d'un DDD de transition,
  construire une transition dépliée correspondante
- pour chaque chemin d'un DDD de place initialement marquée,
  si la place dépliée existe, la marquer initialement
- pour chaque couple (p,t) du réseau coloré lié par un arc,
  pour chaque couple (p',t') dépliées à partir de p et t
  si la valuation de l'arc est cohérente avec p' et t',
  créer un arc dans le réseau déplié

Suppression des transitions de garde fausse

Algorithme

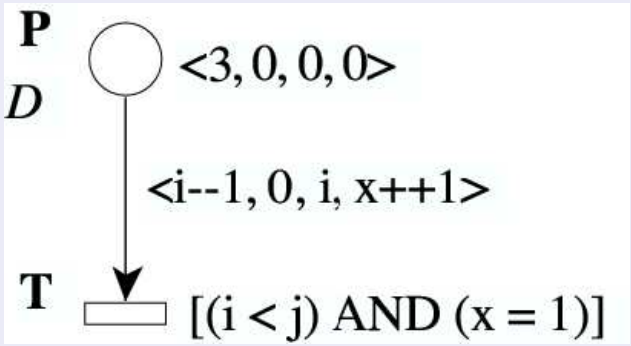
- pour chaque transition :
  - identification des couleurs ne satisfaisant pas la garde
  - suppression des chemins correspondant dans le DDD de la transition

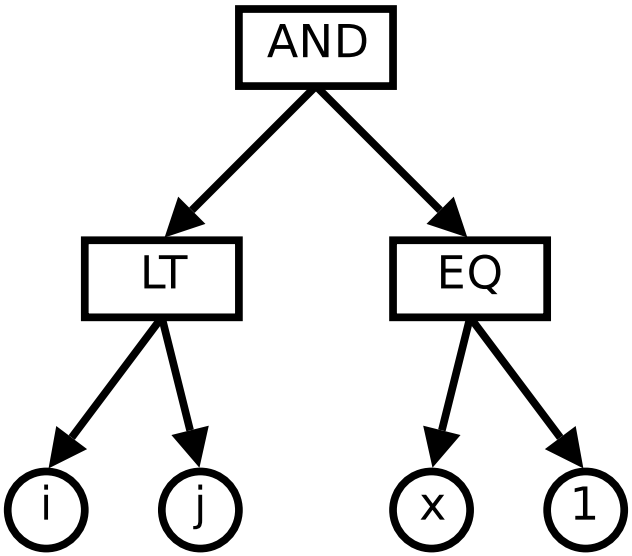
Implémentation

- une garde est représentée par un arbre
- son parcours construit un homomorphisme :
  - $\texttt{AND} \longrightarrow \circ$.
  - $\texttt{OR} \longrightarrow +$,
  - reste $\longrightarrow$ homomorphisme.

Exemple

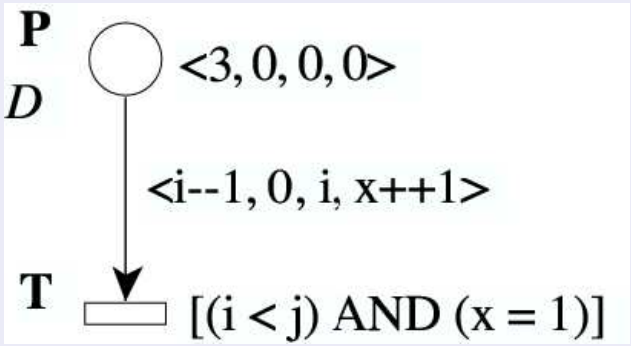
Transition

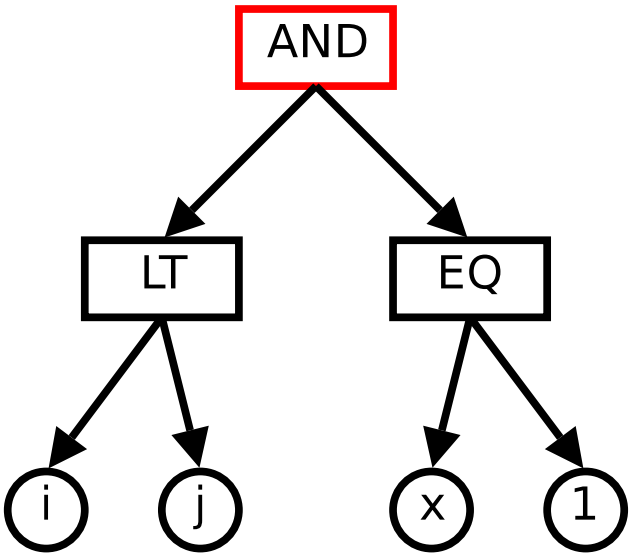


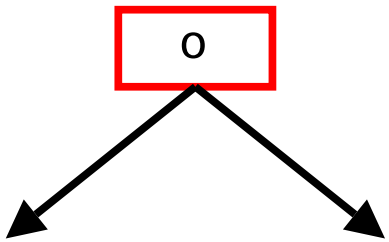


Exemple

Transition

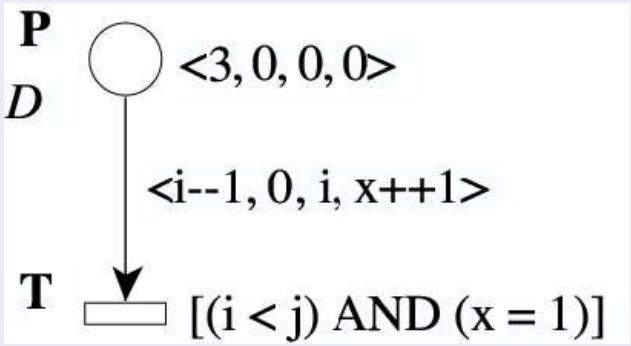


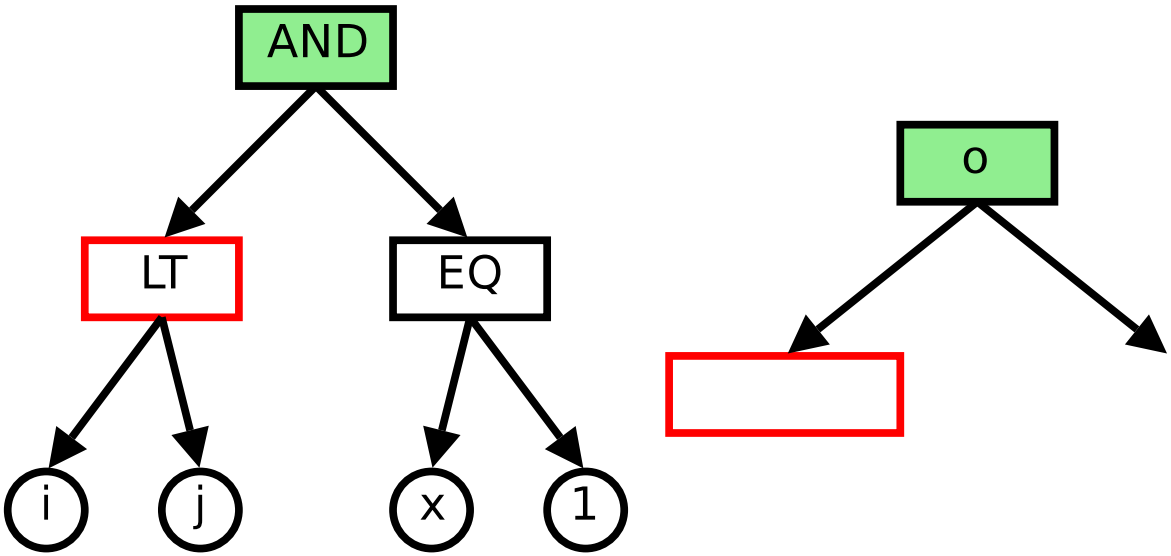




Exemple

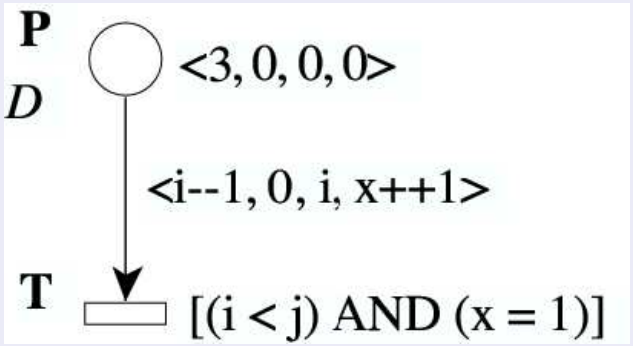
Transition

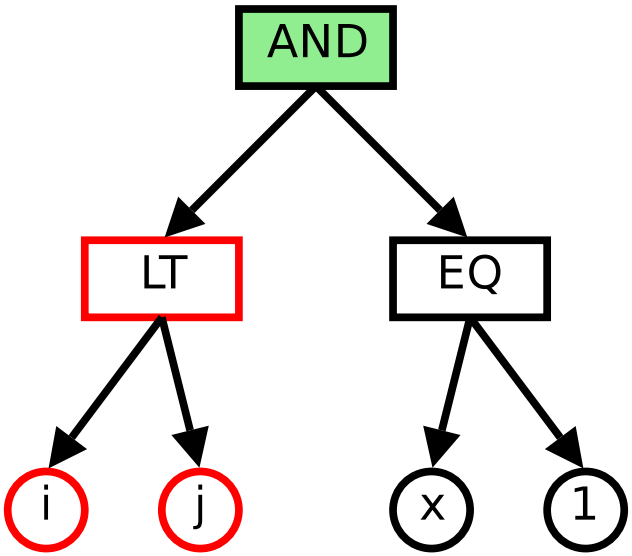


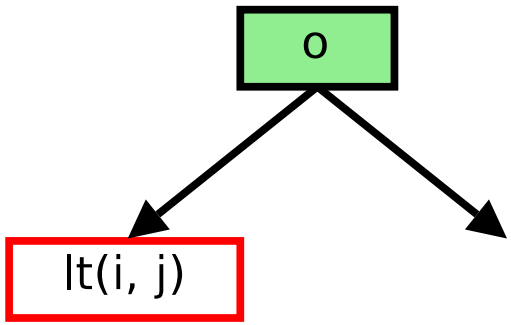


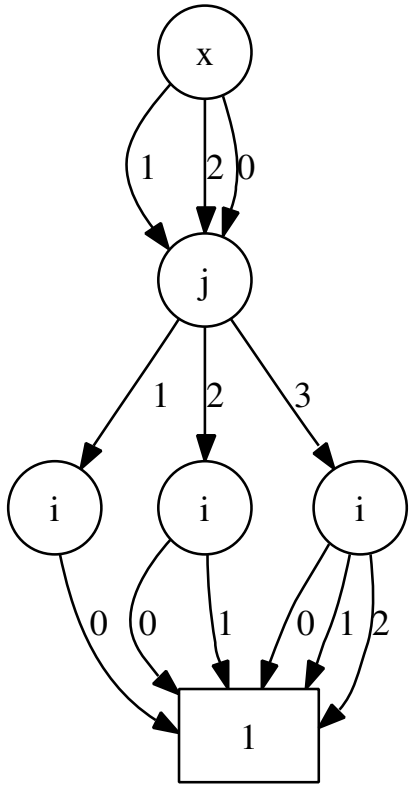
Exemple

Transition



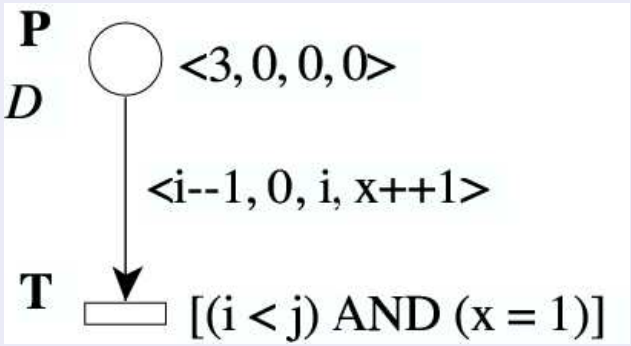


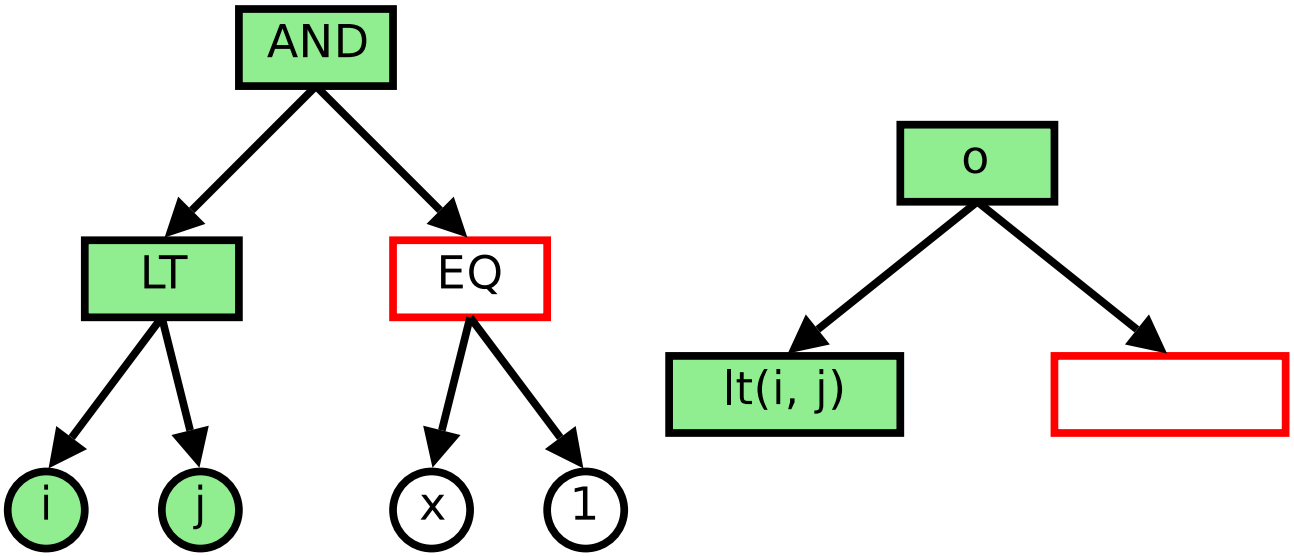




Exemple

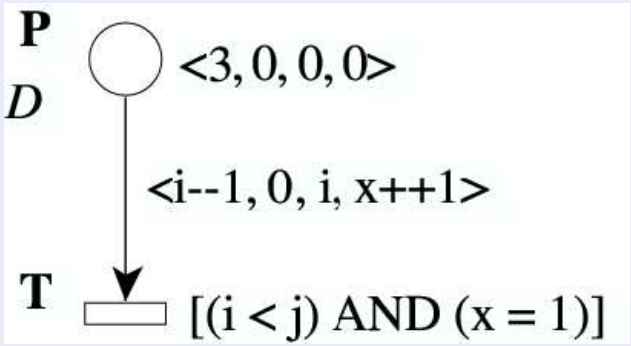
Transition

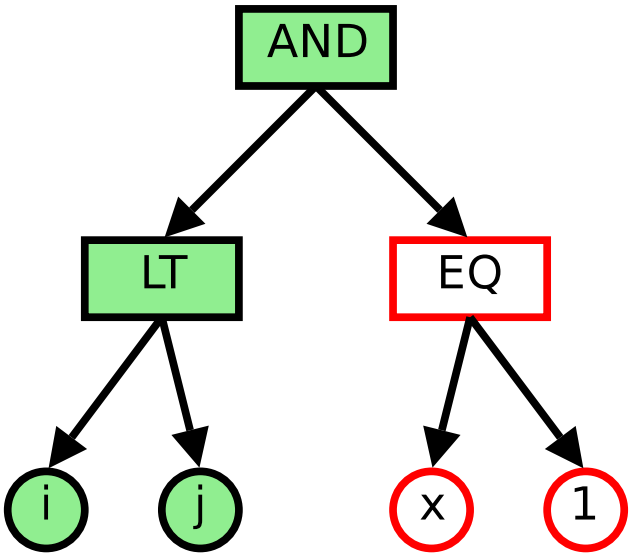


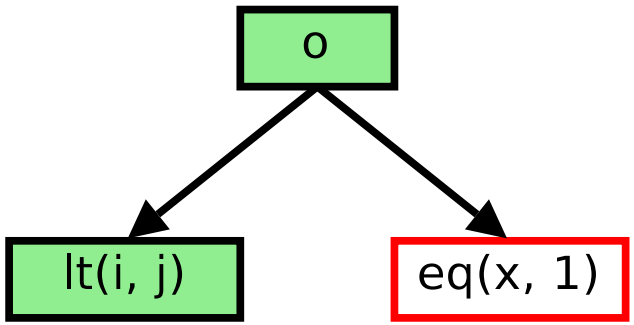


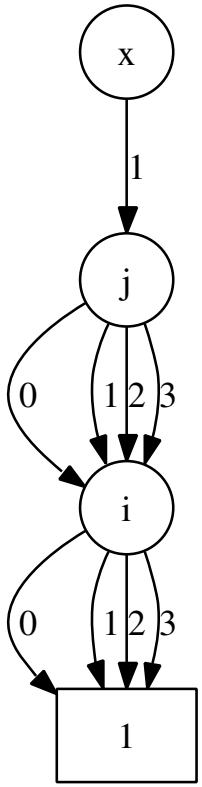
Exemple

Transition



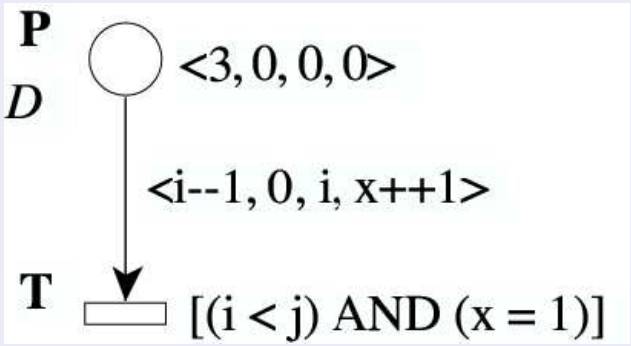


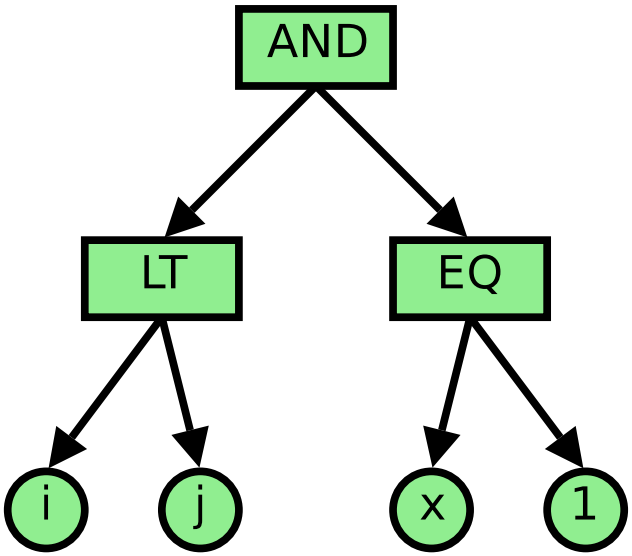


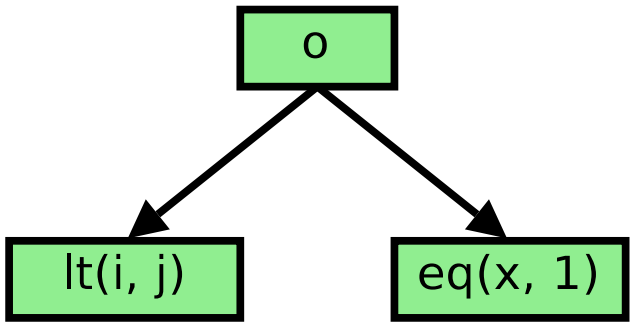


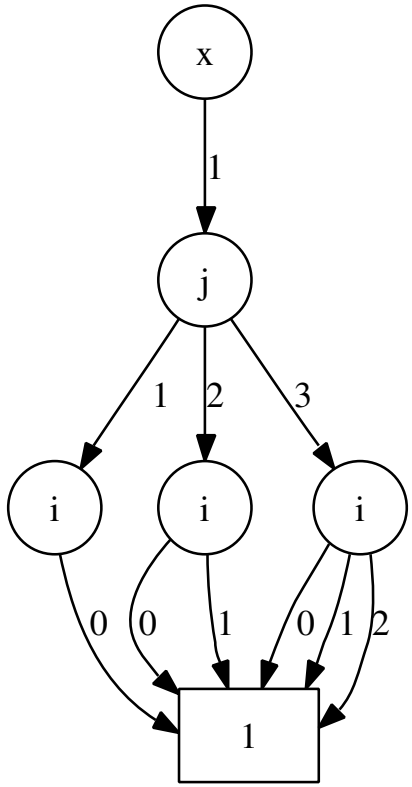
Exemple

Transition





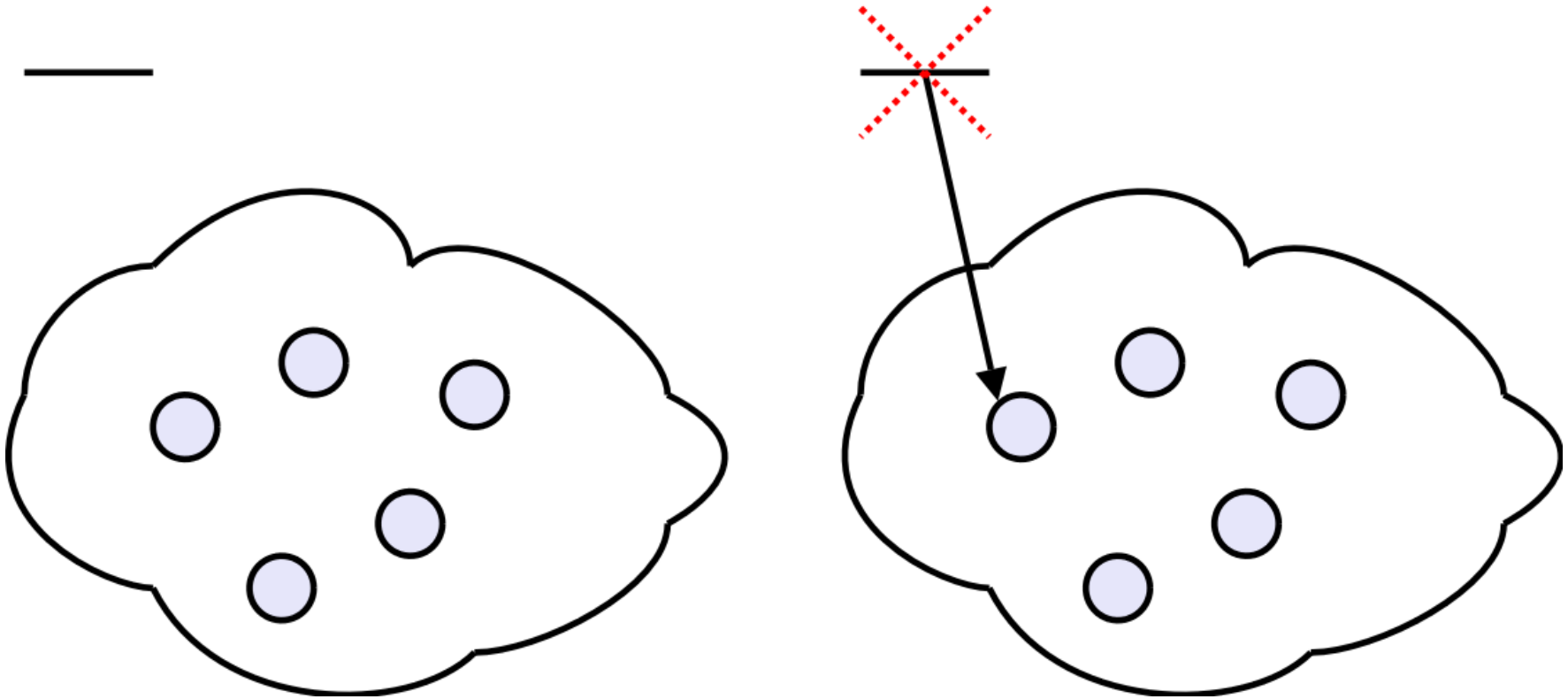




Verrou maximal non marqué

Principe

ensemble de places ne pouvant pas gagner de jeton de l'extérieur



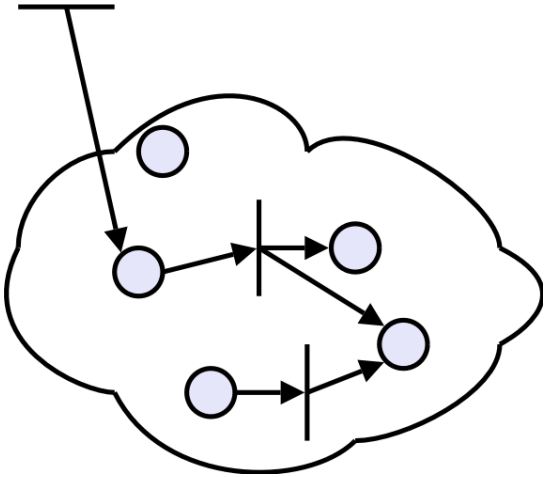
Verrou maximal non marqué

Principe

ensemble de places ne pouvant pas gagner de jeton de l'extérieur

Algorithme :

1. initialement le verrou contient toutes les places
2. suppression des places initialement marquées
3. itération (jusqu'à stabilité) :



4. suppression du verrou obtenu du réseau déplié
5. suppression des transitions ayant une place du verrou en entrée

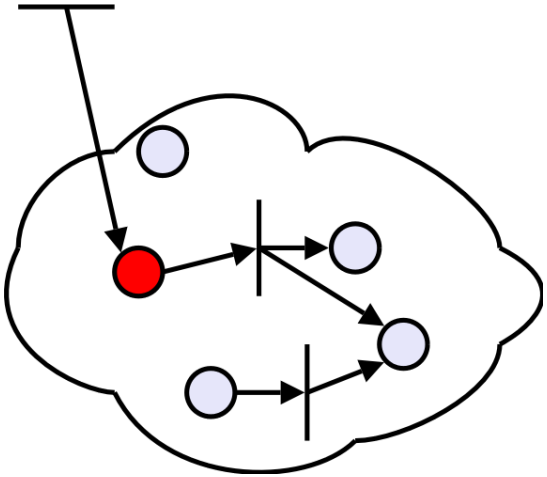
Verrou maximal non marqué

Principe

ensemble de places ne pouvant pas gagner de jeton de l'extérieur

Algorithme :

1. initialement le verrou contient toutes les places
2. suppression des places initialement marquées
3. itération (jusqu'à stabilité) :



4. suppression du verrou obtenu du réseau déplié
5. suppression des transitions ayant une place du verrou en entrée

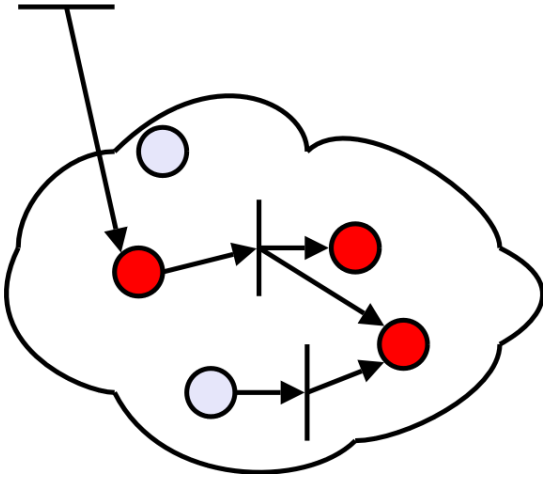
Verrou maximal non marqué

Principe

ensemble de places ne pouvant pas gagner de jeton de l'extérieur

Algorithme :

1. initialement le verrou contient toutes les places
2. suppression des places initialement marquées
3. itération (jusqu'à stabilité) :



4. suppression du verrou obtenu du réseau déplié
5. suppression des transitions ayant une place du verrou en entrée

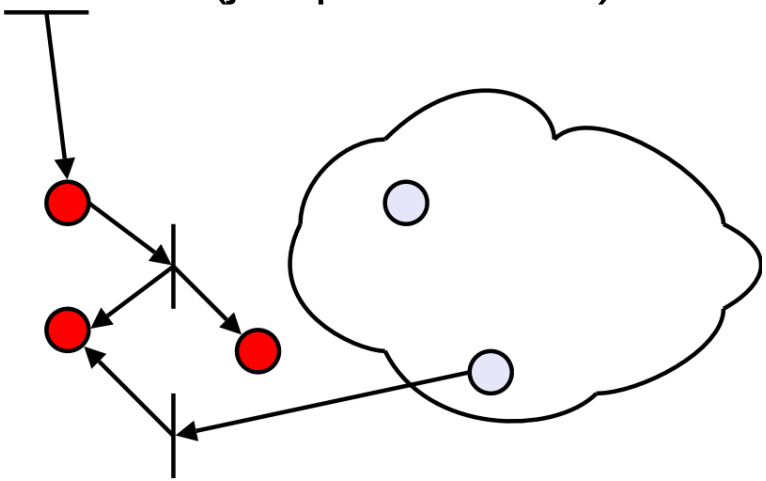
Verrou maximal non marqué

Principe

ensemble de places ne pouvant pas gagner de jeton de l'extérieur

Algorithme :

1. initialement le verrou contient toutes les places
2. suppression des places initialement marquées
3. itération (jusqu'à stabilité) :

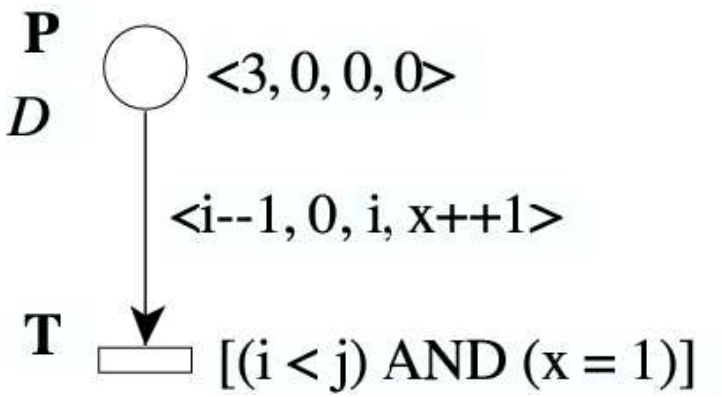


4. suppression du verrou obtenu du réseau déplié
5. suppression des transitions ayant une place du verrou en entrée

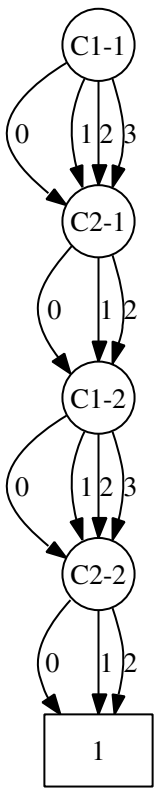
Suppression des places initialement marquées

Principe

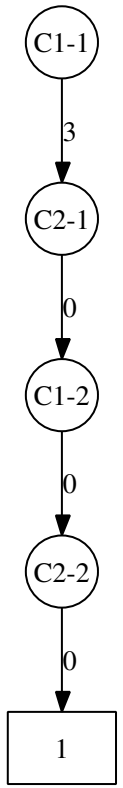
- à partir du marquage d'une place colorée
- construction du DDD représentant les places dépliées marquées
- opération : $DDD_{place} - DDD_{marques}$



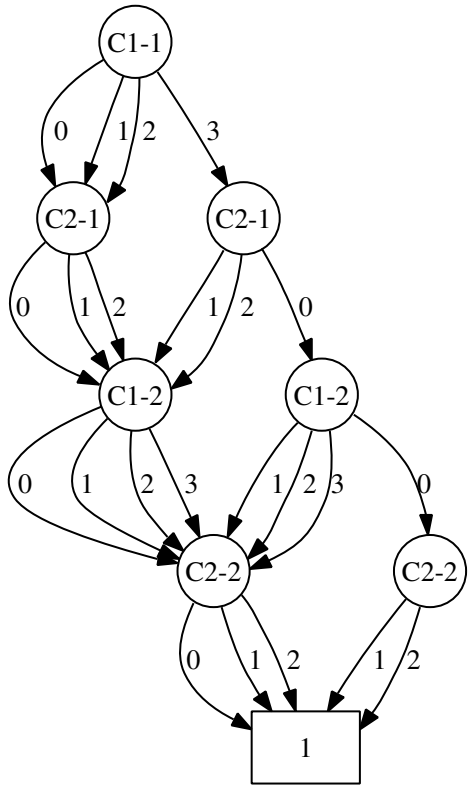
P



Marque(p)



Résultat



Algorithme

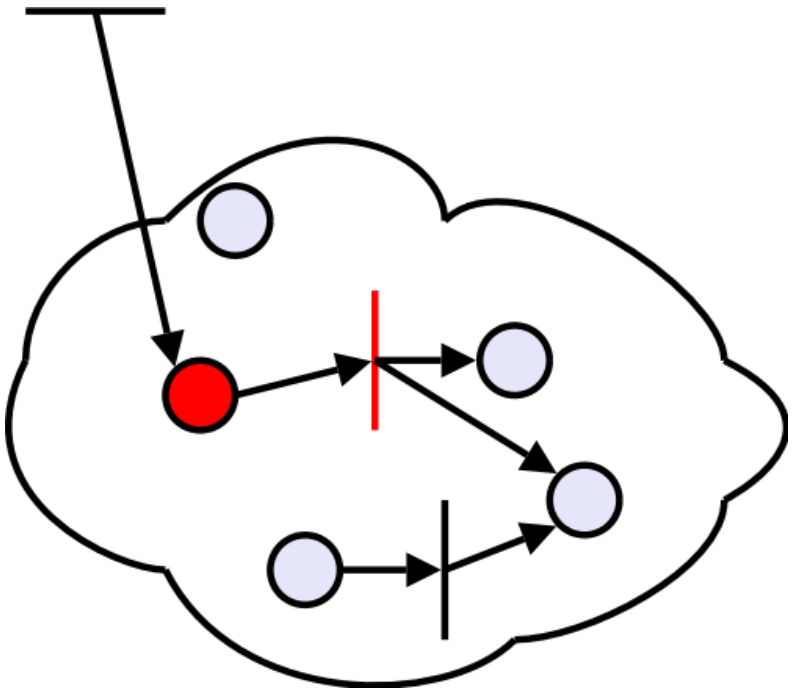
- initialement :
  - on considère un verrou “élargi” :
    initialement toutes les places et transitions dépliées
  - calcul par réduction jusqu’à stabilité
- supprimer les places initialement marquées
- itérer jusqu’à stabilité :
  - pour chaque transition dépliée si toutes ses places en entrée sont hors du verrou, la transition et ses places en sortie sortent du verrou
- supprimer le verrou calculé du réseau déplié

Problème

- algorithme pour chaque transition dépliée
- mais représentation symbolique des transitions dépliées
  $\Rightarrow$ quelle est la représentation symbolique des transitions dépliées à supprimer parmi les candidates ?

Construction de l'algorithme d'itération

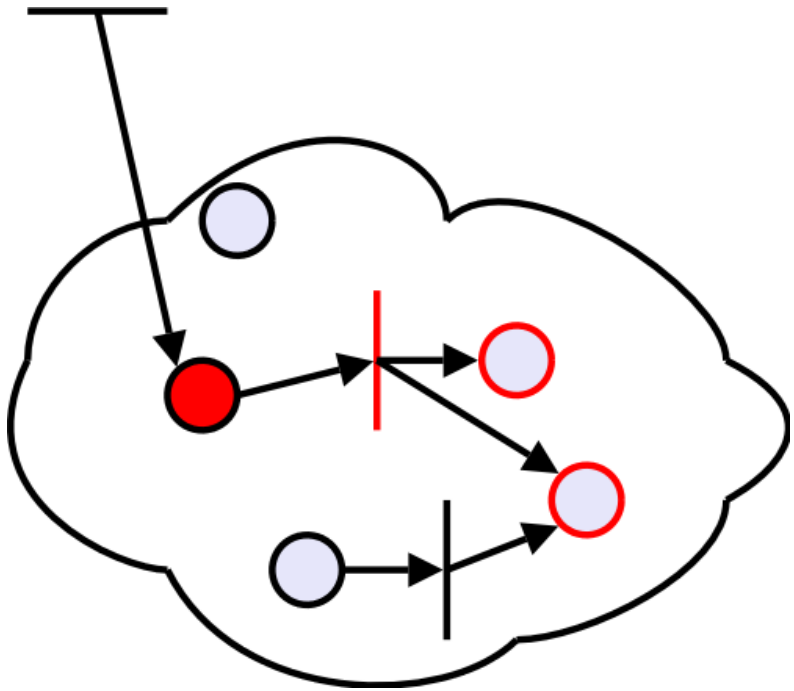
Dans le verrou en cours de calcul :



3. obtenir les transitions à supprimer

Construction de l'algorithme d'itération

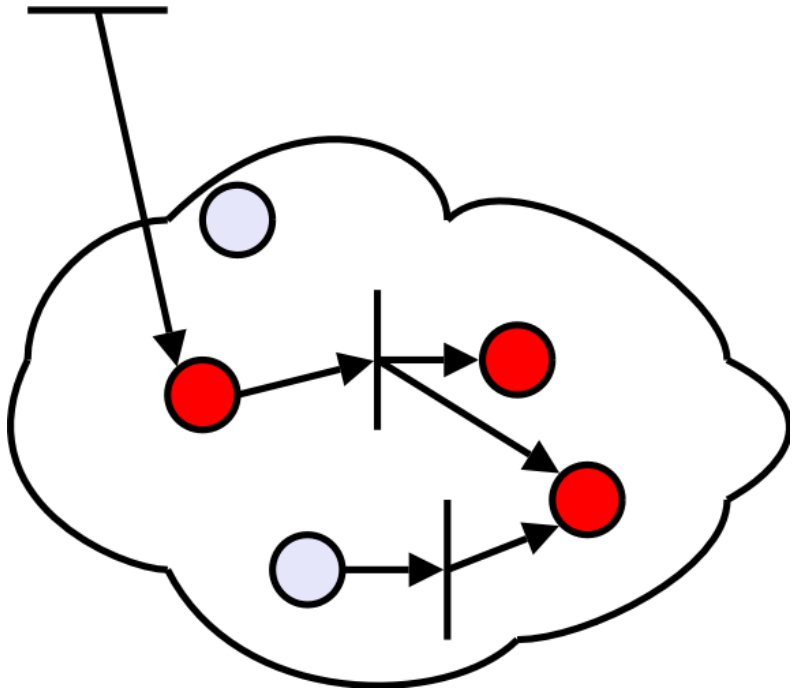
Dans le verrou en cours de calcul :



3. obtenir les transitions à supprimer
4. obtenir les places *Post* des transitions à supprimer

Construction de l'algorithme d'itération

Dans le verrou en cours de calcul :

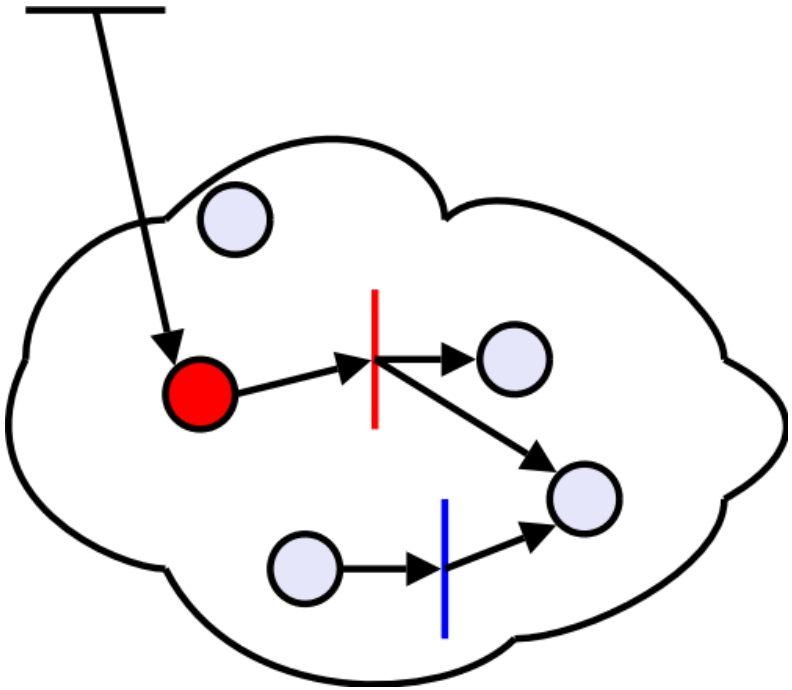


3. obtenir les transitions à supprimer
4. obtenir les places *Post* des transitions à supprimer
5. supprimer du verrou les transitions et places sélectionnées

Construction de l'algorithme d'itération

Dans le verrou en cours de calcul :

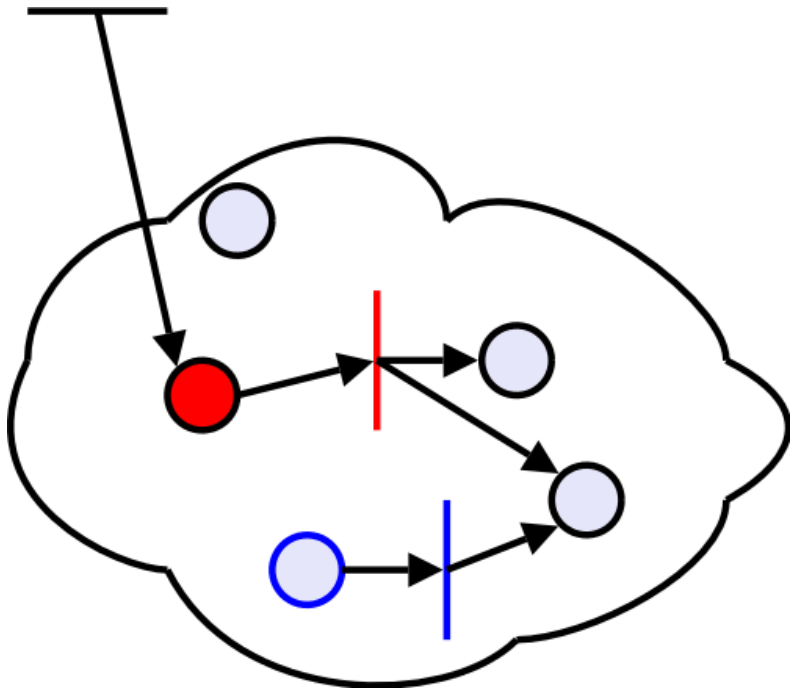
2. obtenir les transitions ayant au moins une place en entrée dans le verrou
3. obtenir les transitions à supprimer
4. obtenir les places *Post* des transitions à supprimer
5. supprimer du verrou les transitions et places sélectionnées



Construction de l'algorithme d'itération

Dans le verrou en cours de calcul :

1. pour chaque transition t,
   pour chaque place $p \in Pre(t)$,
   obtenir les transition dépliées de t
   liées en entrée aux places
   dépliées de p encore dans le
   verrou
2. obtenir les transitions ayant au
   moins une place en entrée dans le
   verrou
3. obtenir les transitions à supprimer
4. obtenir les places *Post* des
   transitions à supprimer
5. supprimer du verrou les
   transitions et places sélectionnées



Construction de l'algorithme d'itération

Dans le verrou en cours de calcul :

1. pour chaque transition t,
   pour chaque place $p \in Pre(t)$,
   obtenir les transition dépliées de t
   liées en entrée aux places
   dépliées de p encore dans le
   verrou
2. obtenir les transitions ayant au moins une place en entrée dans le verrou
3. obtenir les transitions à supprimer
4. obtenir les places *Post* des transitions à supprimer
5. supprimer du verrou les transitions et places sélectionnées

Solution

interpréter les valuations des arcs

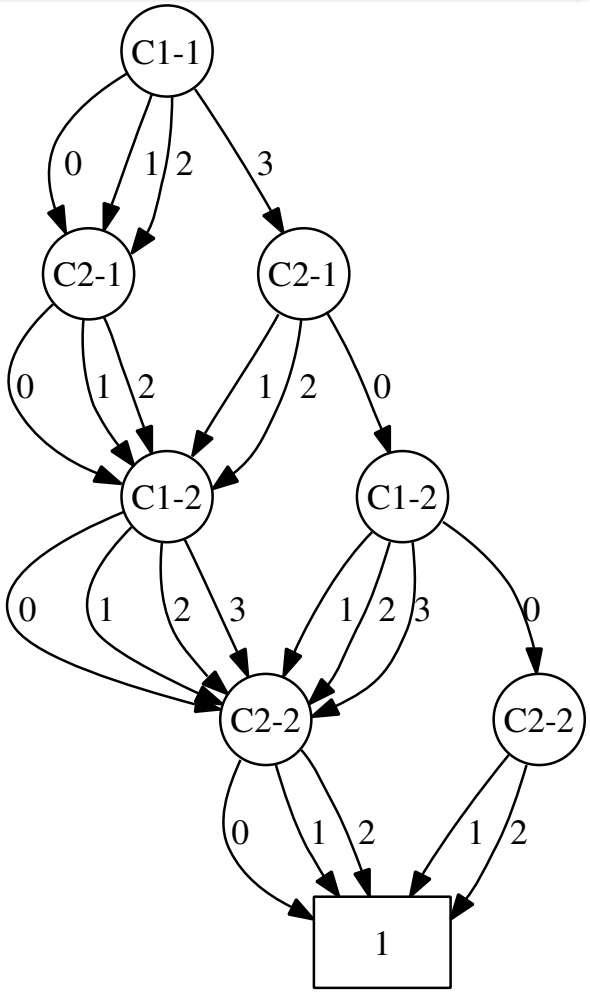
$Pre(t)$

1. sélection des places valides vis-à-vis des constantes de la valuation
2. suppression des variables de DDD correspondant à ces constantes
3. sélection des places cohérentes avec la valuation concernant les successeurs des variables de transition apparaissant plusieurs fois
4. suppression des occurences supplémentaires de ces variables
5. application des successeurs
6. renommage des variables de DDD vers les variables de DDD correspondant à des variables de la valuation
7. remise en ordre des variables de DDD pour correspondre à l'ordre du DDD de transition
8. ajout des variables de transition manquantes

$Pre(t)$ pas-à-pas

Etape

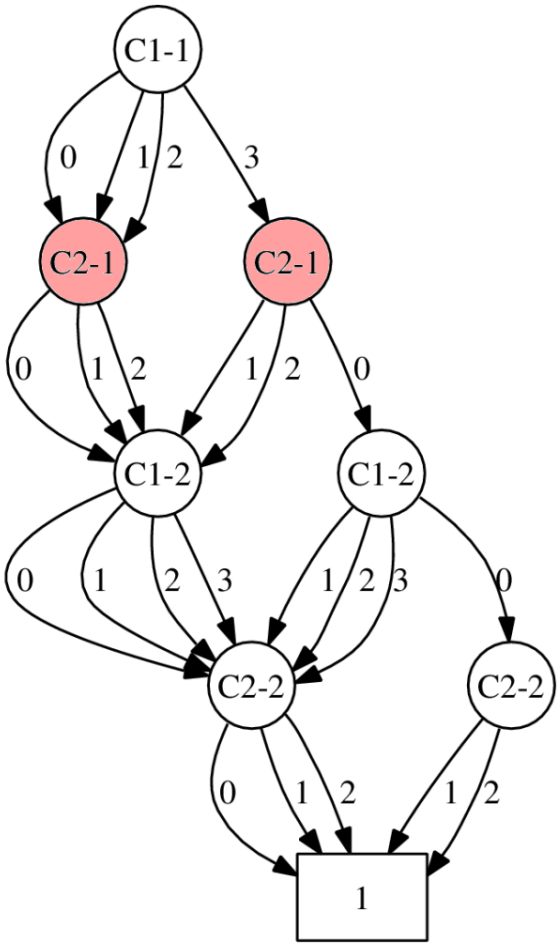
initialement : places non marquées

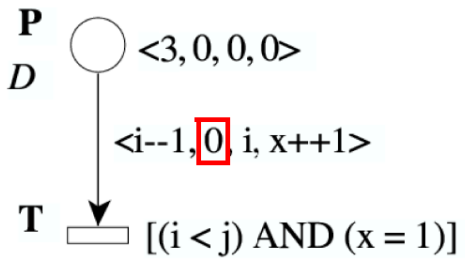


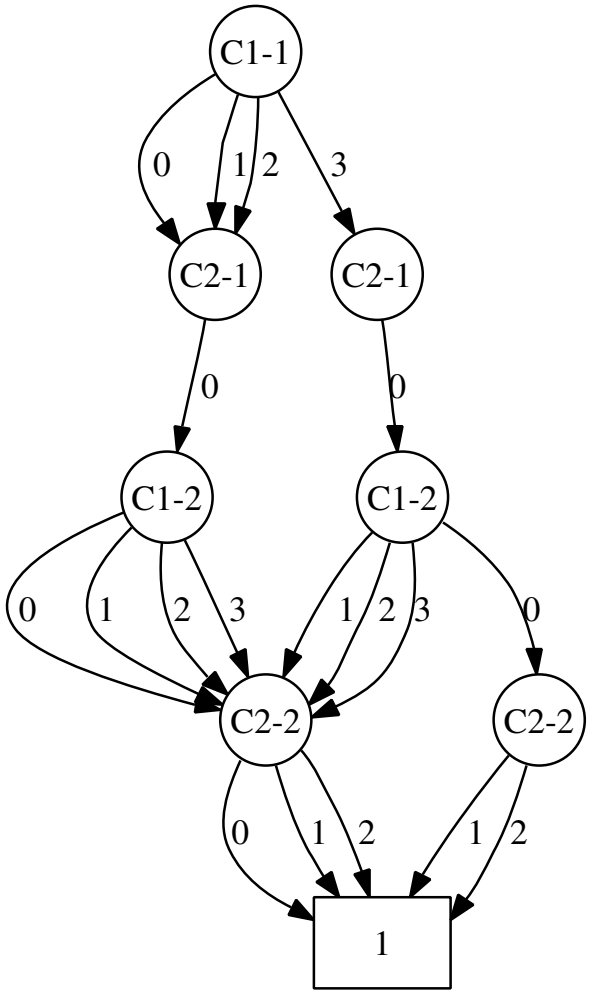
$Pre(t)$ pas-à-pas

Etape

sélection des places valides vis-à-vis des constantes de la valuation



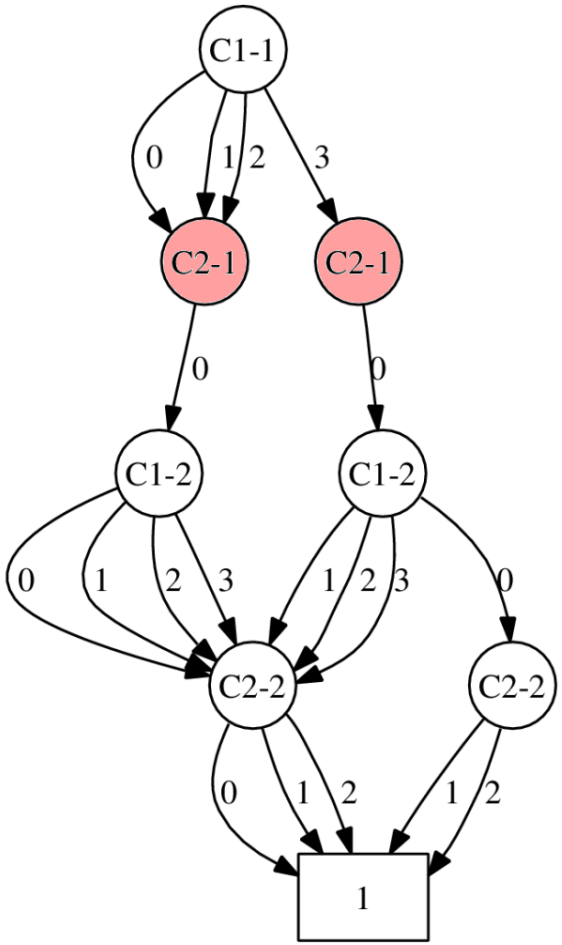


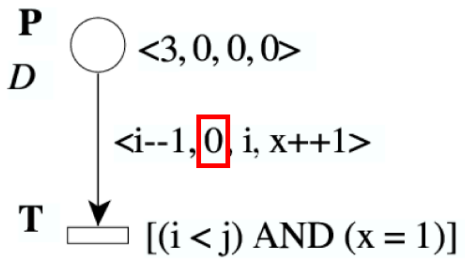


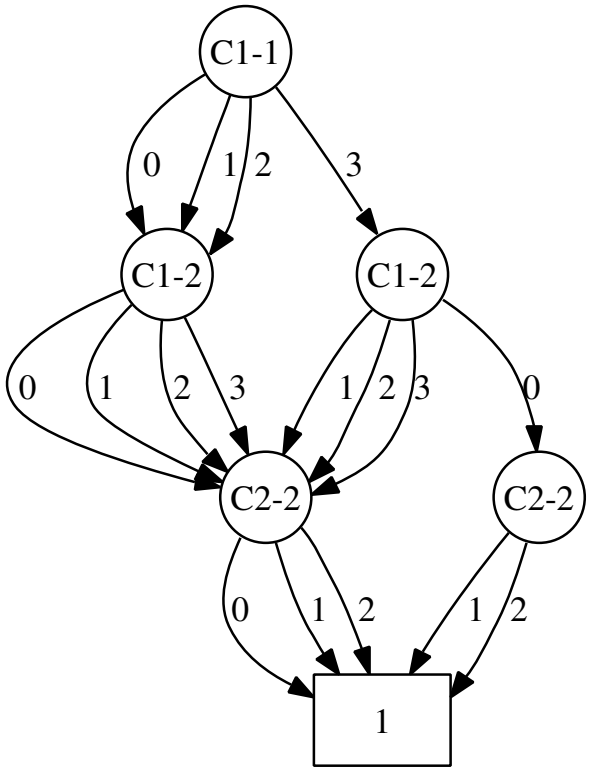
$Pre(t)$ pas-à-pas

Etape

suppression des variables de DDD correspondant à ces constantes



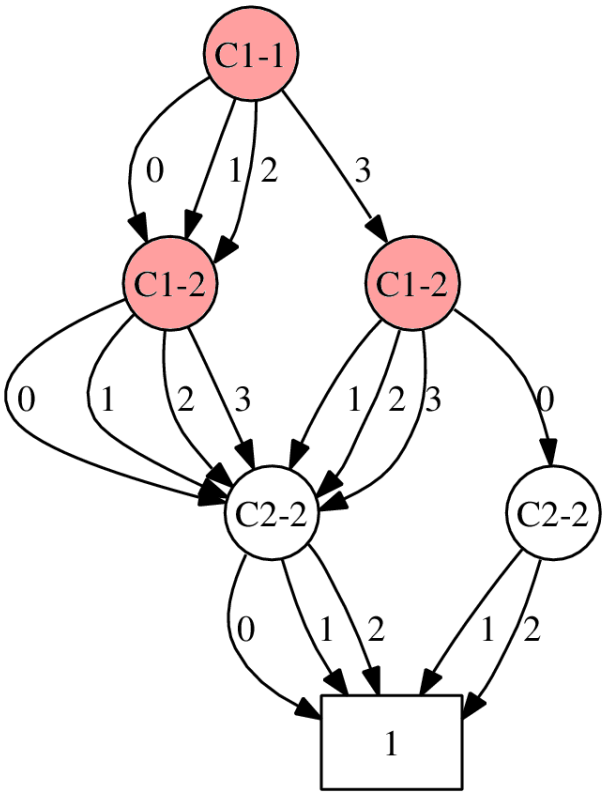


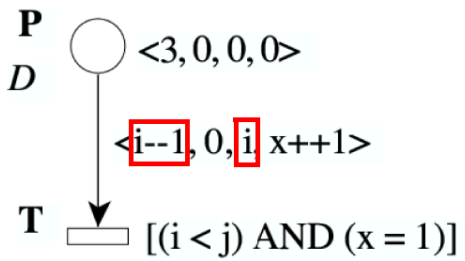


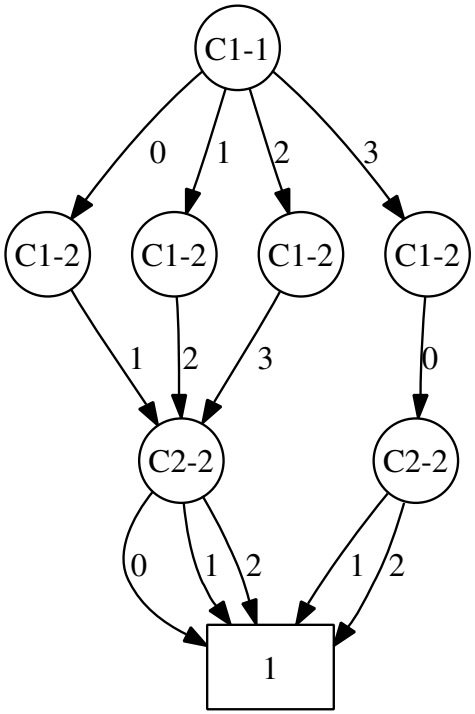
$Pre(t)$ pas-à-pas

Etape

sélection des places cohérentes avec la valuation concernant les successeurs des variables de transition apparaissant plusieurs fois



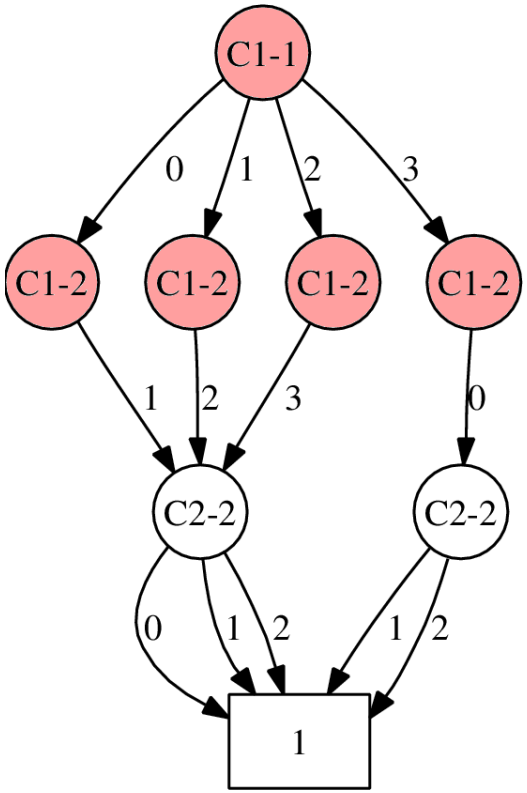


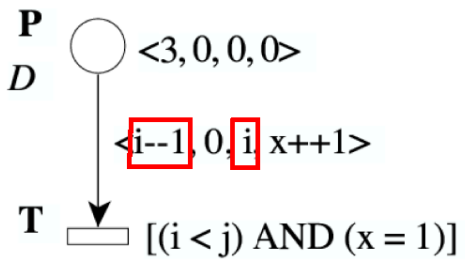


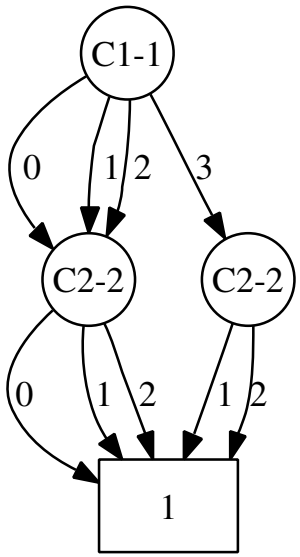
$Pre(t)$ pas-à-pas

Etape

suppression des occurences supplémentaires de ces variables



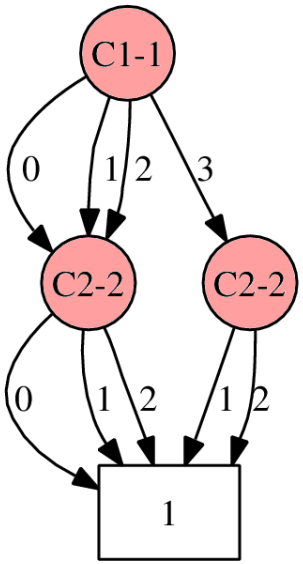


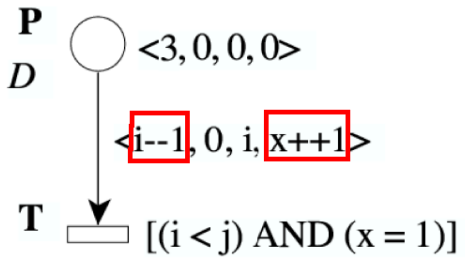


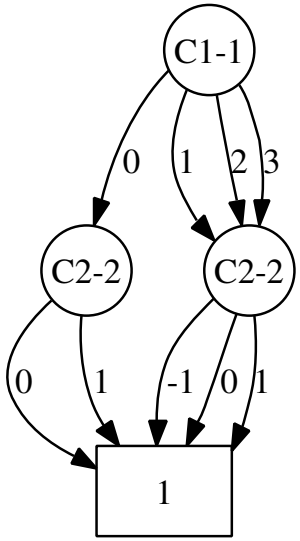
$Pre(t)$ pas-à-pas

Etape

application des successeurs



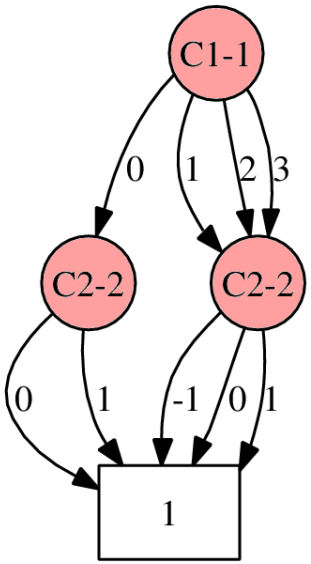


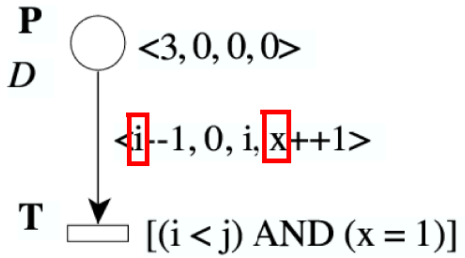


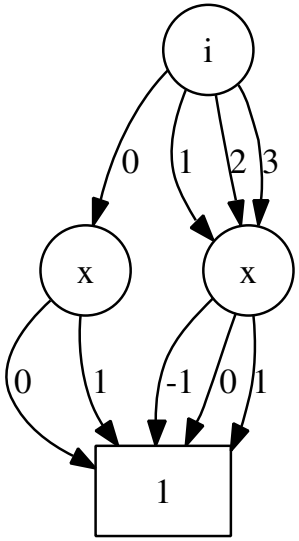
$Pre(t)$ pas-à-pas

Etape

renommage des variables de DDD vers les variables de DDD correspondant à des variables de la valuation



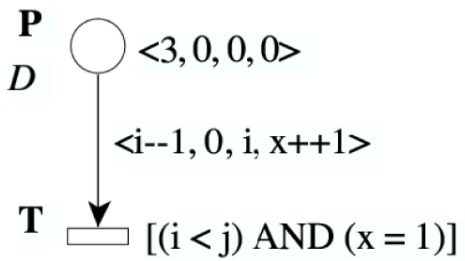


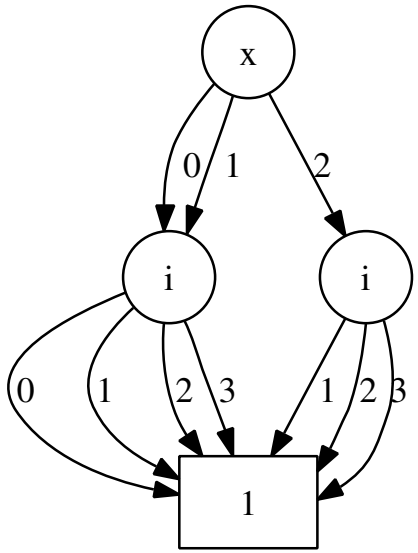


$Pre(t)$ pas-à-pas

Etape

remise en ordre des variables de DDD pour correspondre à l'ordre du DDD de transition

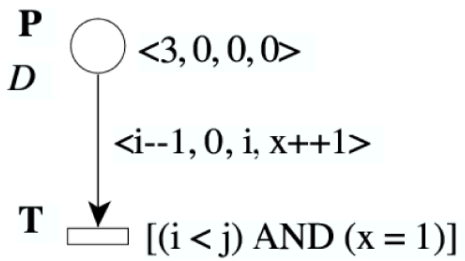


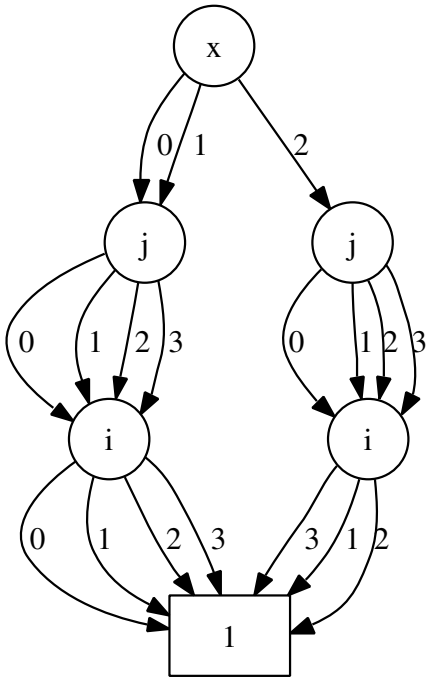


$Pre(t)$ pas-à-pas

Etape

ajout des variables de transition manquantes





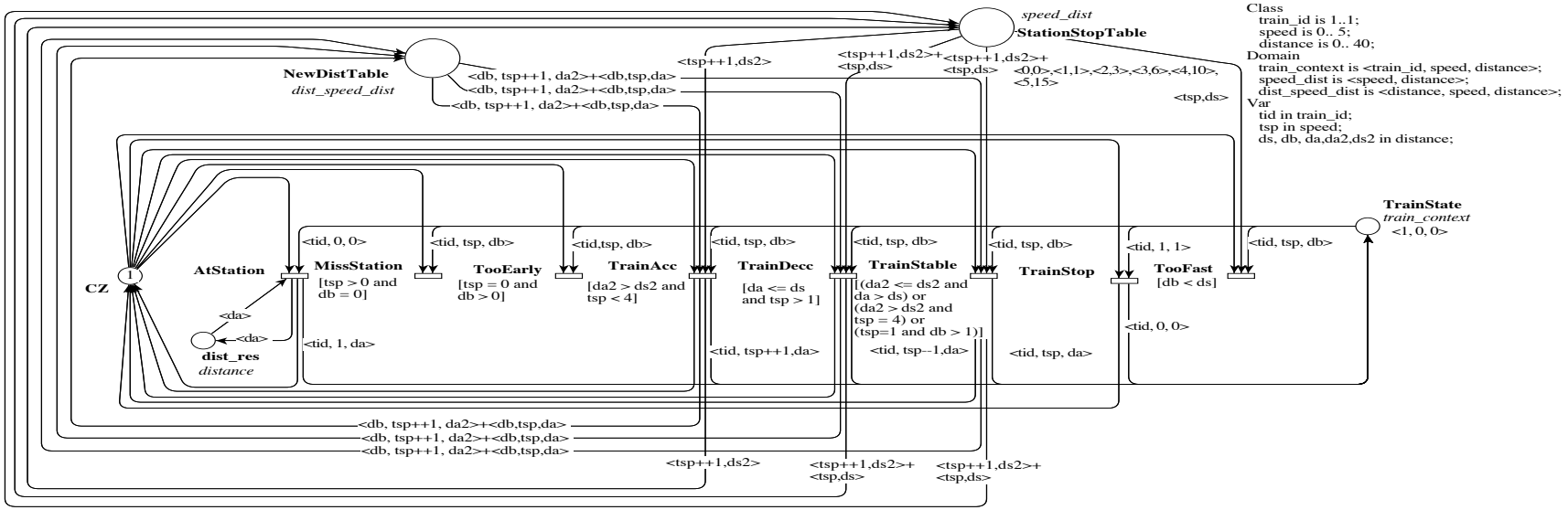
$Post(t)$

1. copie des variables de DDD correspondant à des variables de transition présentes dans la valuation
2. suppression des autres variables de DDD
3. application des successeurs
4. insertion des constantes de la valuation

Résultats

Model	Initial		Optimized			
	Places	Transitions	Time	Memory	Places	Transitions
PolyORB_20	4.964	3.392.800	0,91	9	4.964	3.392.040
PolyORB_40	9.344	6.786.800	1,39	14	9.344	6.784.860
PolyORB_60	13.724	10.182.400	2,07	21	13.724	10.178.480
PolyORB_80	18.104	13.579.600	3,15	31	18.104	13.572.900
Telephon_25	4.800	55.575	0,36	3	4.200	39.325
Telephon_50	17.100	408.650	1,48	14	14.650	281.150
Telephon_75	36.900	1.340.475	4,19	35	31.350	912.975
Telephon_100	64.200	3.132.300	9,22	72	54.300	2.122.300
Peterson_5	1.150	58.850	0,77	8	470	7.810
Peterson_10	9.100	1.835.400	5,23	52	3.890	313.220
Peterson_15	30.600	13.838.400	17,44	163	13.260	2.576.730
Peterson_20	72.400	58.121.600	44,47	378	31.580	11.308.840
Train	10.620	$1.407.10^9$	7,51	71	343	202

Résultats



<40>,<39>,<38>,<37>,<36>,<35>,<34>,<33>,<32>,<31>,<30>,
<29>,<28>,<27>,<26>,<25>,<24>,<23>,<22>,<21>,<20>,<19>,
<18>,<17>,<16>,<15>,<14>,<13>,<12>,<11>,<10>,<9>,<8>,
<7>,<6>,<5>

<1,0,1>,<1,1,0>,<2,0,2>,<2,1,1>,<2,2,0>,<3,0,3>,<3,1,2>,<3,2,1>,<3,3,0>,<4,0,4>,<4,1,3>,<4,2,2>,<4,3,1>,<4,4,0>,<5,0,5>,
<5,1,4>,<5,2,3>,<5,3,2>,<5,4,1>,<5,5,0>,<6,0,6>,<6,1,5>,<6,2,4>,<6,3,3>,<6,4,2>,<6,5,1>,<7,0,7>,<7,1,6>,<7,2,5>,<7,3,4>,<7,4,3>,
<7,5,2>,<8,0,8>,<8,1,7>,<8,2,6>,<8,3,5>,<8,4,4>,<8,5,3>,<9,0,9>,<9,1,8>,<9,2,7>,<9,3,6>,<9,4,5>,<9,5,4>,<10,0,10>,<10,1,9>,<10,2,8>,
<10,3,7>,<10,4,6>,<10,5,5>,<11,0,11>,<11,1,10>,<11,2,9>,<11,3,8>,<11,4,7>,<11,5,6>,<12,0,12>,<12,1,11>,<12,2,10>,<12,3,9>,<12,4,8>,
<12,5,7>,<13,0,13>,<13,1,12>,<13,2,11>,<13,3,10>,<13,4,9>,<13,5,8>,<14,0,14>,<14,1,13>,<14,2,12>,<14,3,11>,<14,4,10>,<14,5,9>,<15,0,15>,
<15,1,14>,<15,2,13>,<15,3,12>,<15,4,11>,<15,5,10>,<16,0,16>,<16,1,15>,<16,2,14>,<16,3,13>,<16,4,12>,<16,5,11>,<17,0,17>,
<17,1,16>,<17,2,15>,<17,3,14>,<17,4,13>,<17,5,12>,<18,0,18>,<18,1,17>,<18,2,16>,<18,3,15>,<18,4,14>,<18,5,13>,<19,0,19>,
<19,1,18>,<19,2,17>,<19,3,16>,<19,4,15>,<19,5,14>,<20,0,20>,<20,1,19>,<20,2,18>,<20,3,17>,<20,4,16>,<20,5,15>,<21,0,21>,
<21,1,20>,<21,2,19>,<21,3,18>,<21,4,17>,<21,5,16>,<22,0,22>,<22,1,21>,<22,2,20>,<22,3,19>,<22,4,18>,<22,5,17>,<23,0,23>,
<23,1,22>,<23,2,21>,<23,3,20>,<23,4,19>,<23,5,18>,<24,0,24>,<24,1,23>,<24,2,22>,<24,3,21>,<24,4,20>,<24,5,19>,<25,0,25>,
<25,1,24>,<25,2,23>,<25,3,22>,<25,4,21>,<25,5,20>,<26,0,26>,<26,1,25>,<26,2,24>,<26,3,23>,<26,4,22>,<26,5,21>,<27,0,27>,
<27,1,26>,<27,2,25>,<27,3,24>,<27,4,23>,<27,5,22>,<28,0,28>,<28,1,27>,<28,2,26>,<28,3,25>,<28,4,24>,<28,5,23>,<29,0,29>,
<29,1,28>,<29,2,27>,<29,3,26>,<29,4,25>,<29,5,24>,<30,0,30>,<30,1,29>,<30,2,28>,<30,3,27>,<30,4,26>,<30,5,25>,<31,0,31>,
<31,1,30>,<31,2,29>,<31,3,28>,<31,4,27>,<31,5,26>,<32,0,32>,<32,1,31>,<32,2,30>,<32,3,29>,<32,4,28>,<32,5,27>,<33,0,33>,
<33,1,32>,<33,2,31>,<33,3,30>,<33,4,29>,<33,5,28>,<34,0,34>,<34,1,33>,<34,2,32>,<34,3,31>,<34,4,30>,<34,5,29>,<35,0,35>,
<35,1,34>,<35,2,33>,<35,3,32>,<35,4,31>,<35,5,30>,<36,0,36>,<36,1,35>,<36,2,34>,<36,3,33>,<36,4,32>,<36,5,31>,<37,0,37>,
<37,1,36>,<37,2,35>,<37,3,34>,<37,4,33>,<37,5,32>,<38,0,38>,<38,1,37>,<38,2,36>,<38,3,35>,<38,4,34>,<38,5,33>,<39,0,39>,
<39,1,38>,<39,2,37>,<39,3,36>,<39,4,35>,<39,5,34>,<40,0,40>,<40,1,39>,<40,2,38>,<40,3,37>,<40,4,36>,<40,5,35>

Suppression de :

- 99,99999985 % des transitions
- 99,995 % des places

Résultats

Model	Initial		Guards		Syphon		Orphans	
	Places	Transitions	% P	% T	% P	% T	% P	% T
PolyORB_20	4.964	3.392.800	0	100	0	0	0	0
PolyORB_40	9.344	6.786.800	0	100	0	0	0	0
PolyORB_60	13.724	10.182.400	0	100	0	0	0	0
PolyORB_80	18.104	13.579.600	0	100	0	0	0	0
Telephon_25	4.800	55.575	0	100	100	0	0	0
Telephon_50	17.100	408.650	0	100	100	0	0	0
Telephon_75	36.900	1.340.475	0	100	100	0	0	0
Telephon_100	64.200	3.132.300	0	100	100	0	0	0
Peterson_5	1.150	58.850	0	55	100	45	ε	0
Peterson_10	9.100	1.835.400	0	59	100	41	ε	0
Peterson_15	30.600	13.838.400	0	61	100	39	ε	0
Peterson_20	72.400	58.121.600	0	62	100	38	ε	0
Train	10.620	$1.407.10^9$	0	61	99	39	1	0

Dépliage partiel

Principe

- ne pas déplier toutes les couleurs
- appliquer les optimisations

- la représentation symbolique ne représente que les classes à déplier
- les variables de valuations dont la classe ne doit pas être dépliée sont ignorées
- pas un problème théorique, mais lourdeur dans l'implémentation
- mais actuellement les gardes ne sont pas générées

Perspectives

Prioritaire

- générer les gardes dans les réseaux partiellement dépliés

Interfaçage avec d'autres outils

- sortie sur PNDDD
- sortie sur SPOT
- sortie permettant d'accéder à chaque place et transition dépliée sans représentation concrète du réseau déplié

Nouvelles optimisations

- suppression des transitions liées uniquement par des arcs doubles

Autre utilisation

- débogage du réseau pendant la modélisation

Conclusion

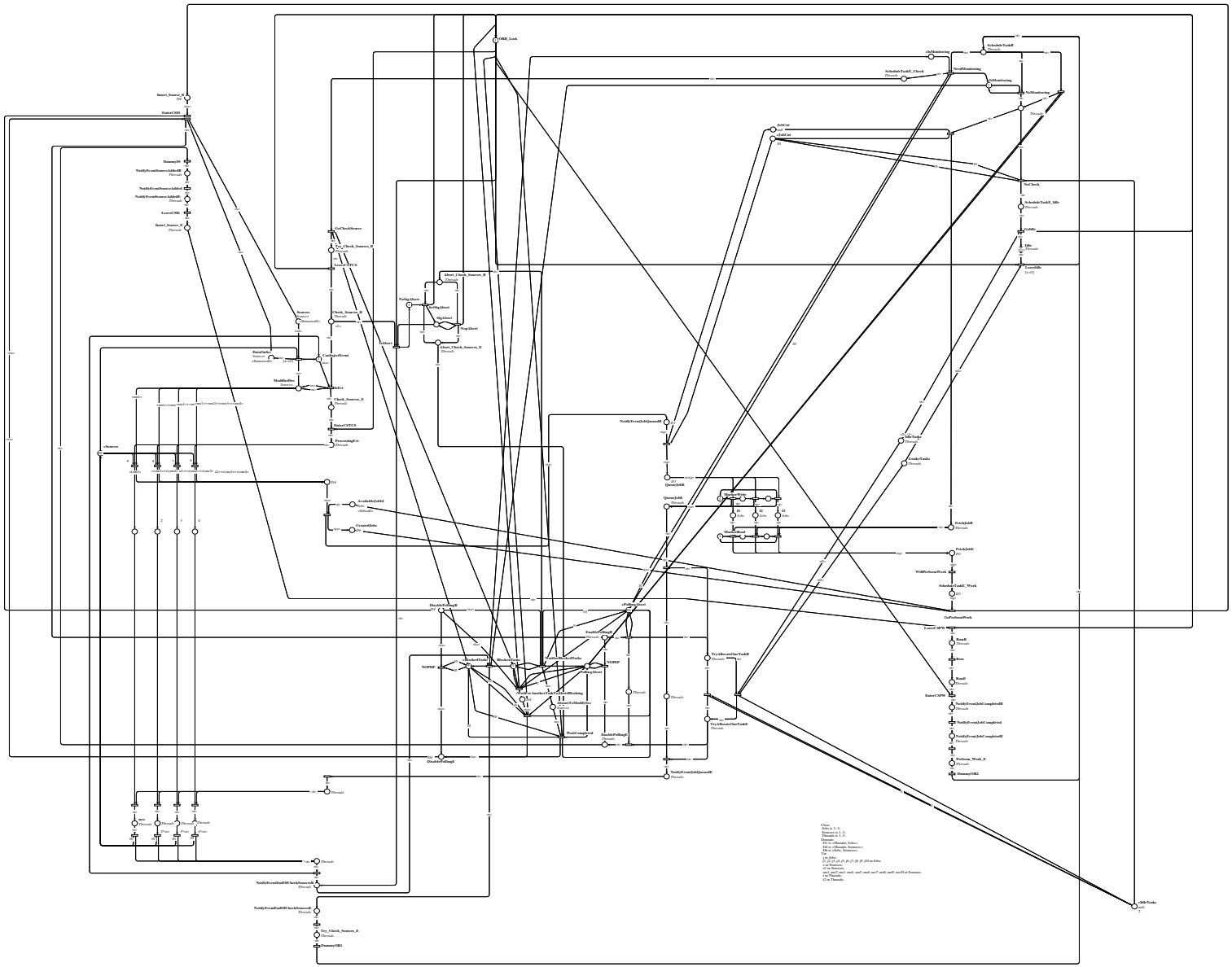
- le dépliage de réseaux ayant une énorme partie morte fonctionne
  - le dépliage des autresd modèles n'est pas pénalisé
  - quelle que soit l'efficacité du dépliage, on peut envisager d'accéder au réseau déplié sans représentation concrète
- utilisation originale des Data Decision Diagrams :
  - représentation du réseau et non des états

Questions ?

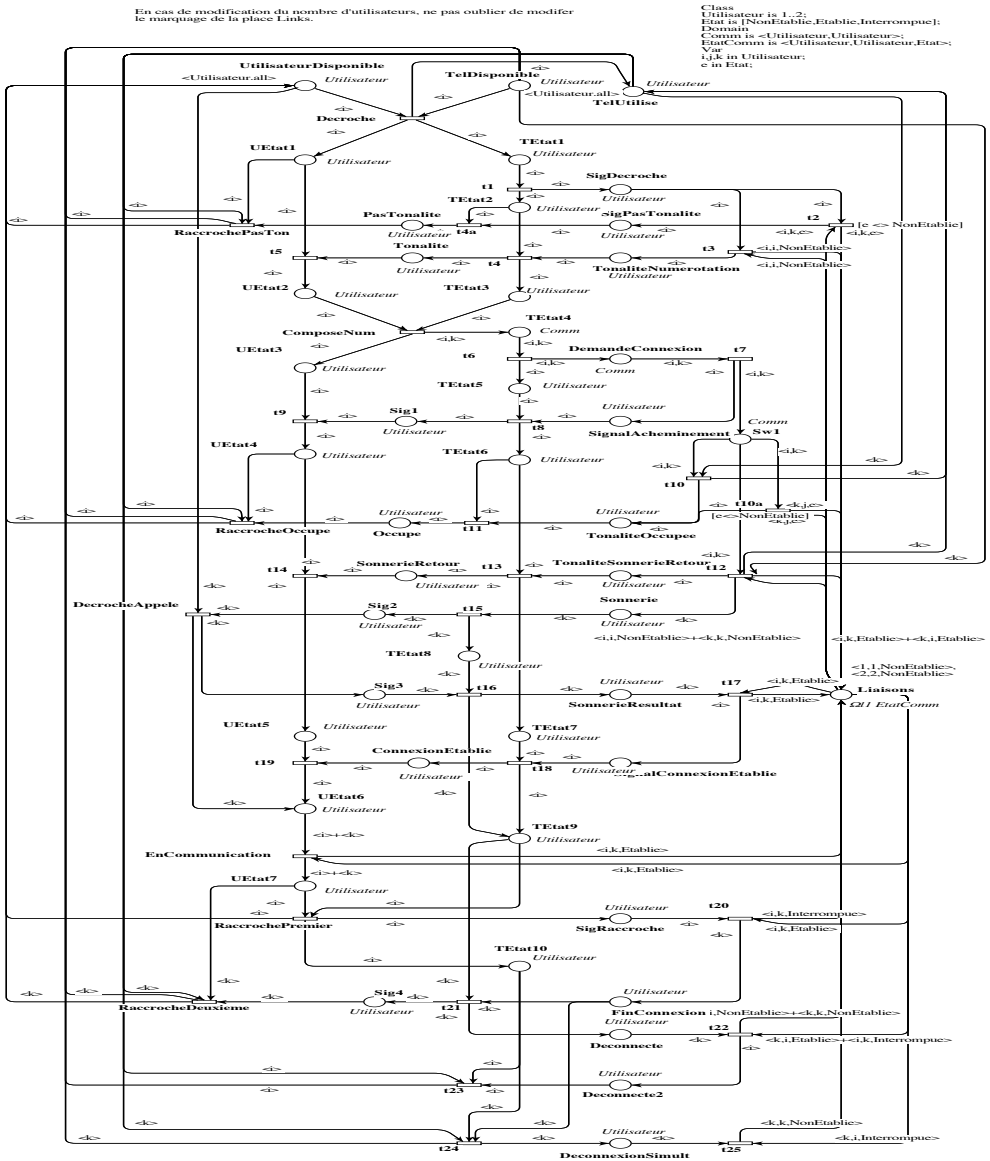

A. de Groot, J. Hooman, F. Kordon, E. Paviot-Adet, I. Vernier-Mounier, M. Lemoine, G. Gaudiere, V. Winter, and D. Kapur.
A survey : Applying formal methods to a software intensive system.
In *The 6th IEEE Int. Symposium on High-Assurance Systems Engineering*, pages 55–64. IEEE Computer Society, 2001.


F. Kordon, A. Linard, and E. Paviot-Adet.
Optimized Colored Nets Unfolding.
Submitted to ICATPN'2006.

PolyOrb



Telephon


En cas de modification du nombre d'utilisateurs, ne pas oublier de modifer le marquage de la place Links.
Class
Utilisateur is 1..2;
Etat is [NonEtablie,Etablie,Interrompue];
Domain
Comm is <Utilisateur,Utilisateur>;
EtatComm is <Utilisateur,Utilisateur,Etat>;
Var
i,j,k in Utilisateur;
e in Etat;
UtilisateurDisponible
TelDisponible
TelUtilise
Decroche
UEtat1
TEtat1
SigDecroche
t1
TEtat2
PasTonalite
sigPasTonalite
t2
RaccrochePasTon
t4a
Tonalite
t5
t4
t3
TonaliteNumerotation
UEtat2
TEtat3
ComposeNum
TEtat4
UEtat3
t6
DemandeConnexion
t7
TEtat5
t9
Sig1
t8
SignalAcheminement
Sw1
UEtat4
TEtat6
t10
t10a
RaccrocheOccupe
Occupe
t11
TonaliteOccupee
t14
SonnerieRetour
t13
TonaliteSonnerieRetour
t12
Sonnerie
DecrocheAppele
Sig2
t15
TEtat8
Sig3
t16
SonnerieResultat
t17
Liaisons
UEtat5
TEtat7
ConnexionEtablie
t19
t18
UEtat6
TEtat9
EnCommunication
UEtat7
RaccrochePremier
SigRaccroche
t20
TEtat10
RaccrocheDeuxieme
Sig4
t21
FinConnexion
Deconnecte
t22
t23
Deconnecte2
t24
DeconnexionSimult
t25

PetersonCompact

```
Class
  Process is 1..n;
  Level is 0..n-1;
  State is [IDLE, ASKFORSECTION, TESTTURN, BEGINLOOP, TESTIDENTITY, TESTALONE, ISENDLOOP, ENDTURN, CS];

Domain
  ProcStateLevelProc is <Process, State, Level, Process>;
  LevelProc is <Level, Process>;

Var
  Id, Id1, IdLoop in Process;
  Lev, Lev1 in Level;
  St in State;
```