Espace d'états atteignables utilisant les MDD pour modèles SAN avec fonctions

Afonso Sales, Brigitte Plateau

{prénom.nom}@imag.fr

Laboratoire d'Informatique de Grenoble, France
Projet MESCAL

Plan

Objectif

Générer rapidement et efficacement l'*espace d'états atteignables* (RSS) d'un système modélisé

Utilisation des *Diagrammes de Décision Multi-valués* (MDD)

- Stockage compact
- Manipulation ensembliste efficace

Applications

- Vérification de modèles de systèmes (Model Checking)
- Évaluation de performance
  - stockage des fonctions de SAN (voir thèse Ihab Sbeity)
  - génération et stockage du RSS

Diagrammes de Décision Multi-valués (MDD)

- Extension des BDD, mais appliquée à une logique non-binaire
- Fonction caractéristique d'ensemble
  - $\{0, 1, \ldots, m_K\} \times \ldots \times \{0, 1, \ldots, m_1\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$
- Ordre et réduction
- Utilisation des opérations : *union* et *intersection*

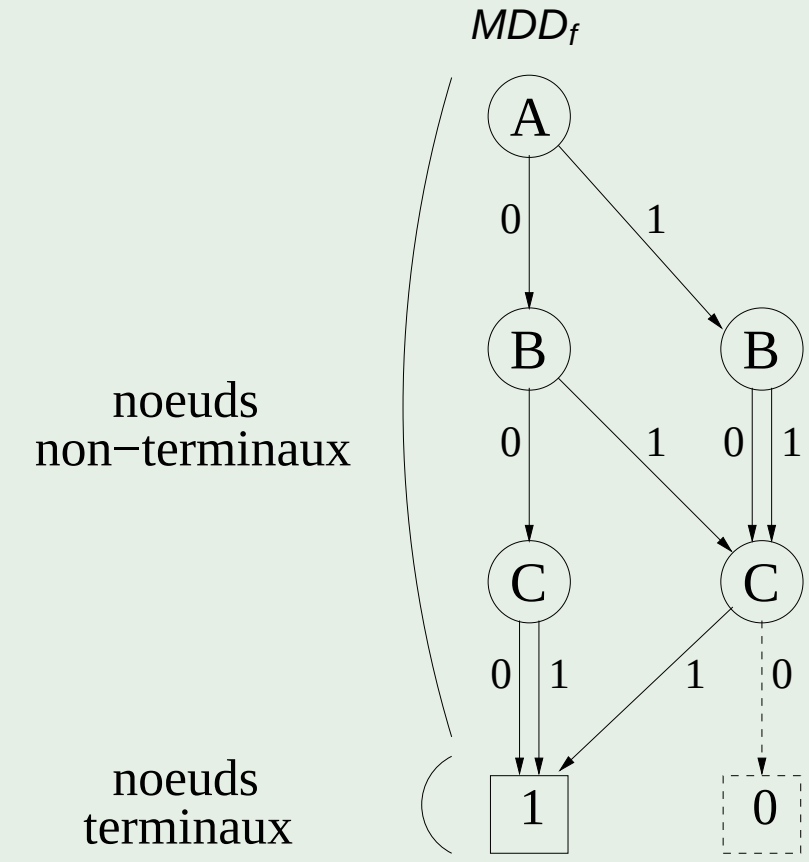
Diagrammes de Décision Multi-valués (MDD)

$f(A, B, C)$

$A = \{0, 1\}$

$B = \{0, 1\}$

$C = \{0, 1\}$



Etats

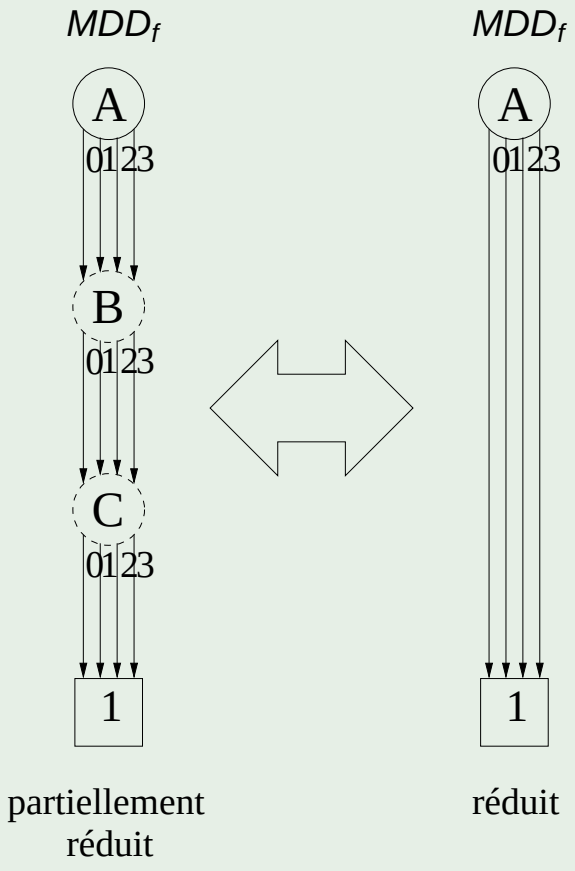
Espace produit = $2 \times 2 \times 2 = 8$ états

$f^{-1}(1) = \{000, 001, 011, 101, 111\}$

Diagrammes de Décision Multi-valués (MDD)

$$f(A,B,C)$$

$$A = \{0,1,2,3\}$$
$$B = \{0,1,2,3\}$$
$$C = \{0,1,2,3\}$$

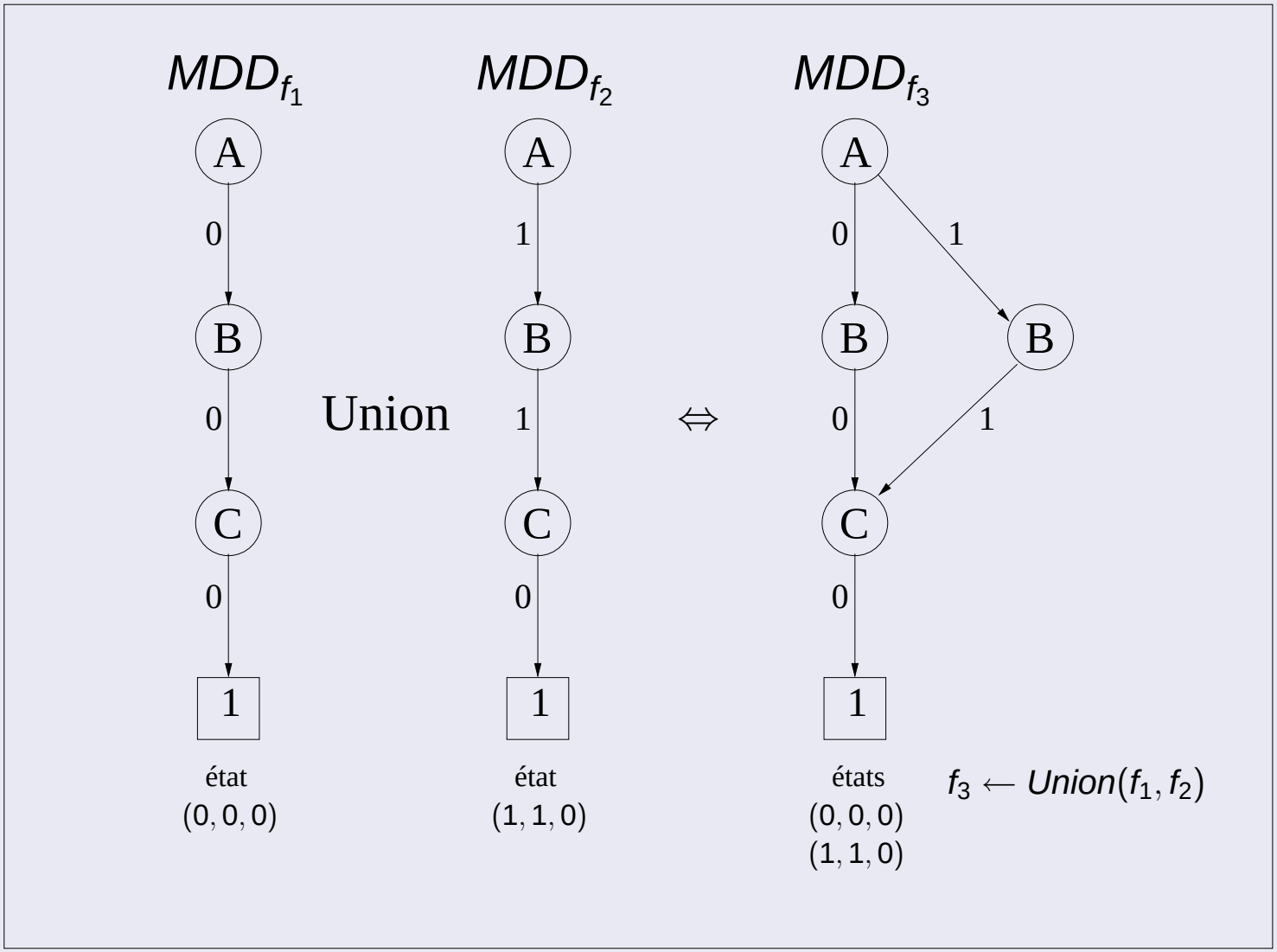


Etats

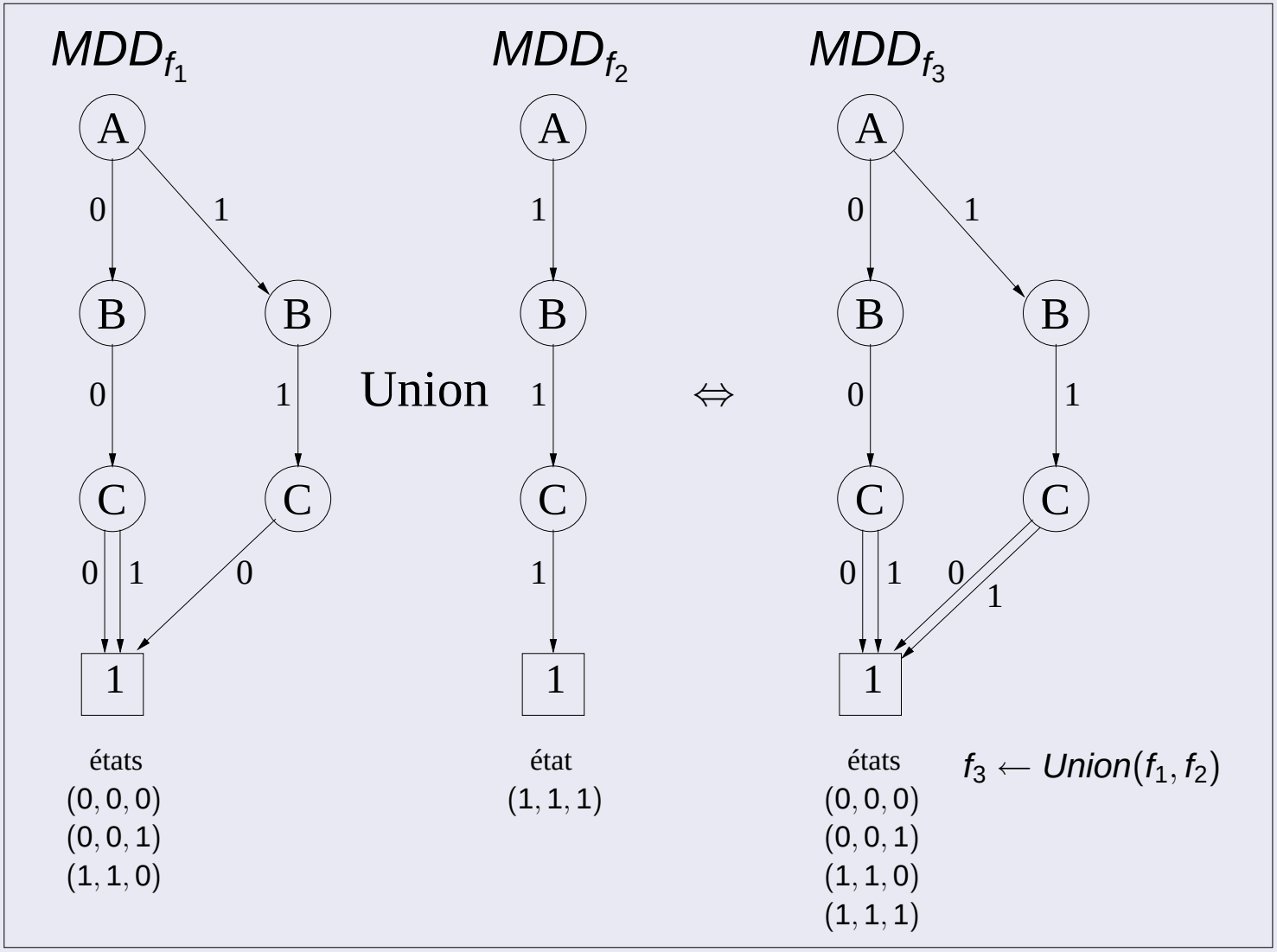
Espace produit = $4 \times 4 \times 4 = 64$ états

$$f^{-1}(1) = \{000, 001, 002, \ldots, 331, 332, 333\}$$

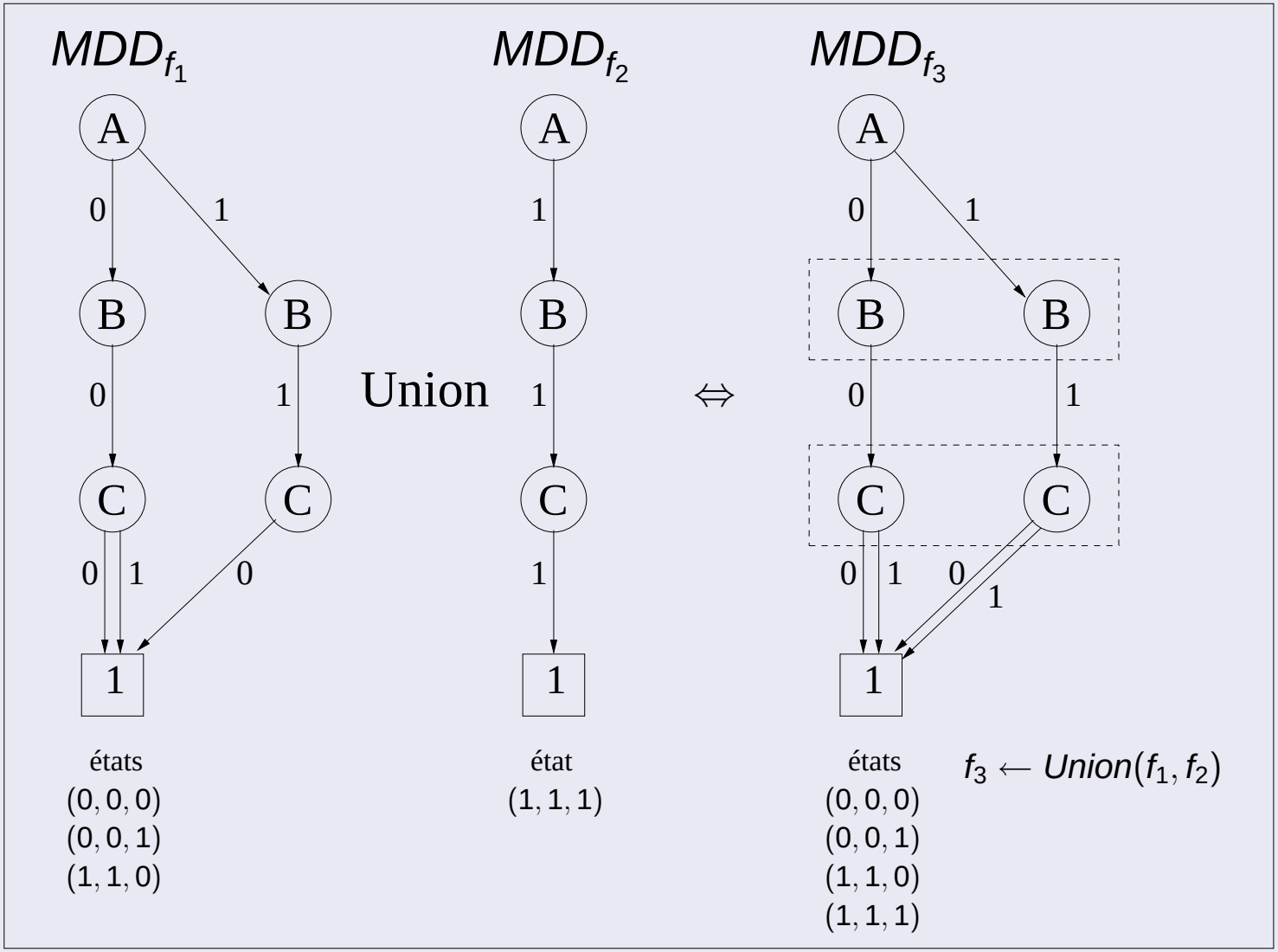
Opérations


MDD_{f_1}
MDD_{f_2}
MDD_{f_3}
A
B
C
1
0
1
Union
⇔
état
(0, 0, 0)
état
(1, 1, 0)
états
(0, 0, 0)
(1, 1, 0)
$f_3 \leftarrow Union(f_1, f_2)$

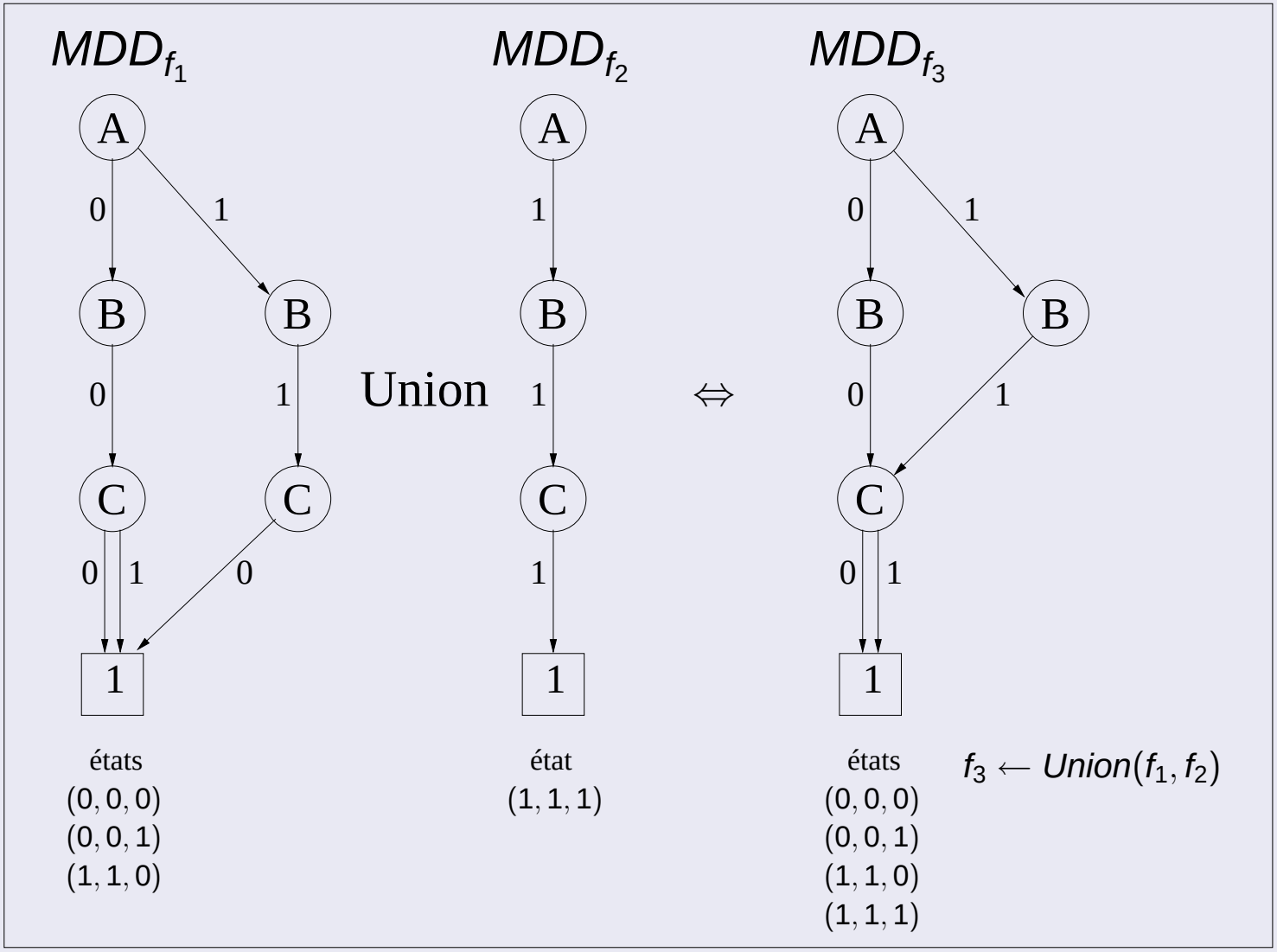
Opérations


MDD_{f_1}
MDD_{f_2}
MDD_{f_3}
A
B
C
1
0
1
Union
⇔
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
état
(1, 1, 1)
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
(1, 1, 1)
$f_3 \leftarrow Union(f_1, f_2)$

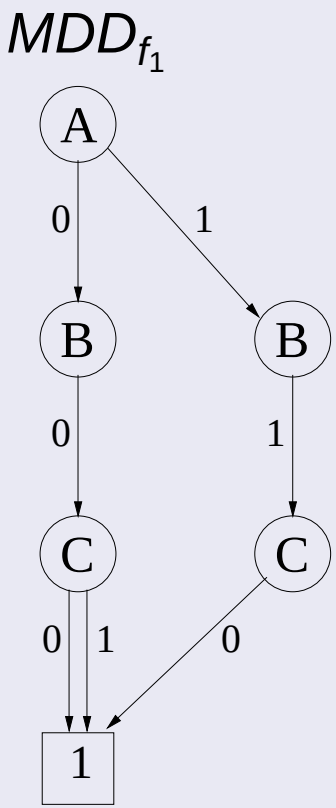
Opérations


MDD_{f_1}
MDD_{f_2}
MDD_{f_3}
A
B
C
1
0
1
Union
⇔
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
état
(1, 1, 1)
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
(1, 1, 1)
$f_3 \leftarrow Union(f_1, f_2)$

Opérations

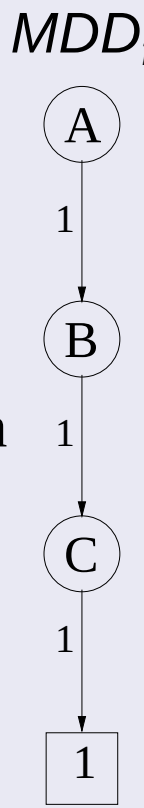

MDD_{f_1}
MDD_{f_2}
MDD_{f_3}
A
B
C
1
0
1
Union
⇔
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
état
(1, 1, 1)
états
(0, 0, 0)
(0, 0, 1)
(1, 1, 0)
(1, 1, 1)
$f_3 \leftarrow Union(f_1, f_2)$

Opérations



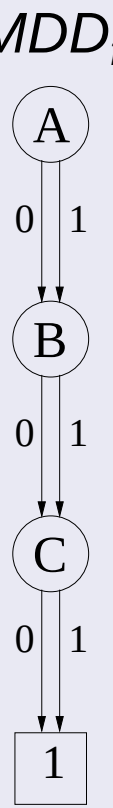
états
$(0,0,0)$
$(0,0,1)$
$(1,1,0)$

Union



état
$(1,1,1)$

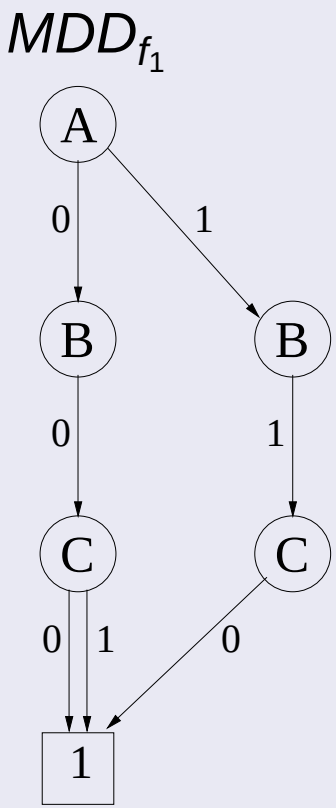
$\Leftrightarrow$



états
$(0,0,0)$
$(0,0,1)$
$(1,1,0)$
$(1,1,1)$

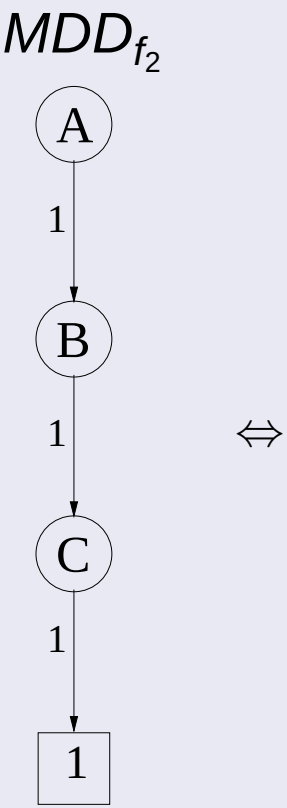
$f_3 \leftarrow Union(f_1, f_2)$

Opérations



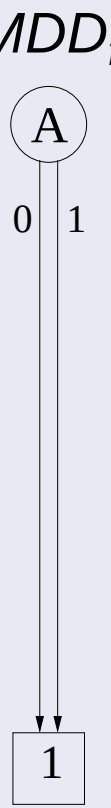
états
$(0,0,0)$
$(0,0,1)$
$(1,1,0)$

Union



état
$(1,1,1)$

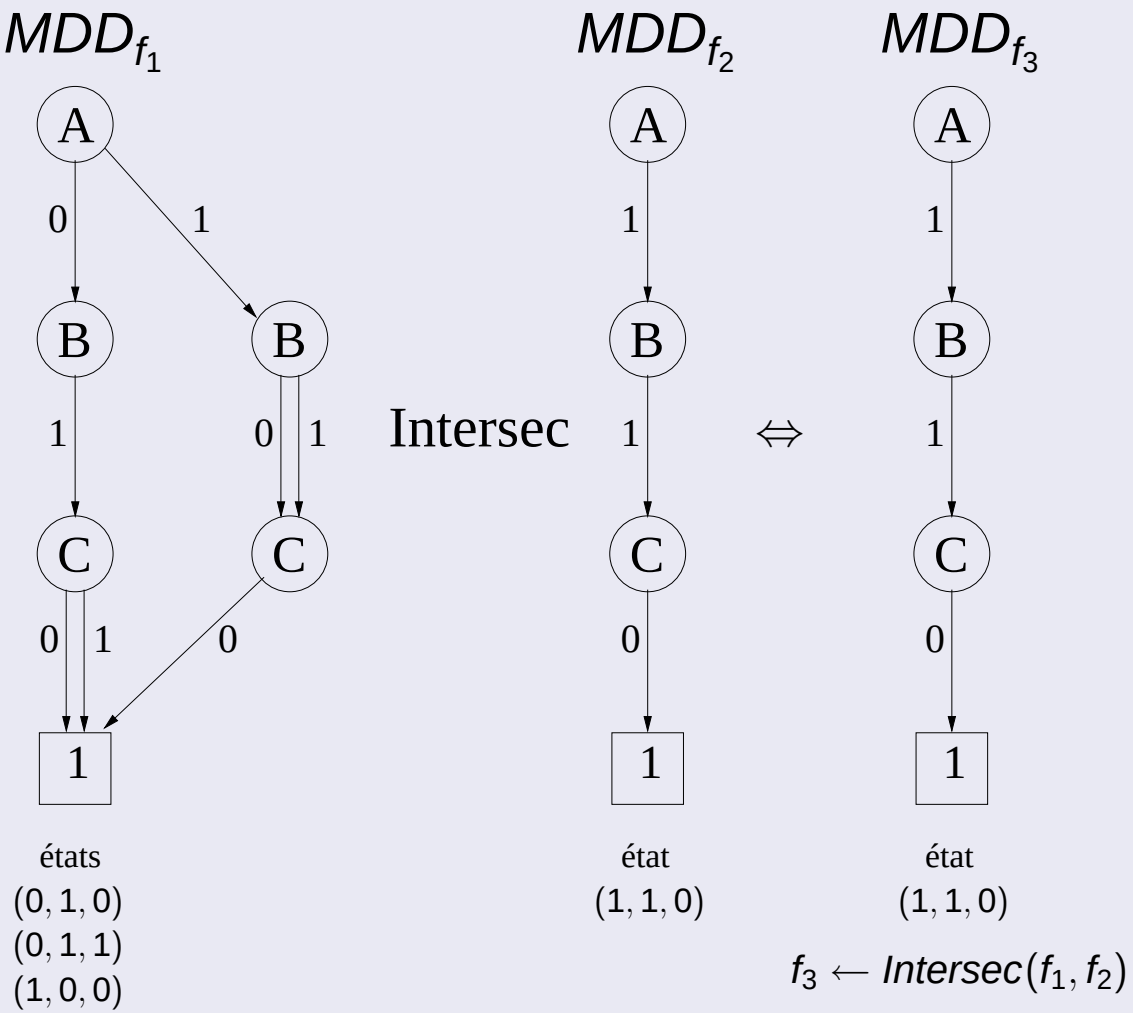
$\Leftrightarrow$



états
$(0,0,0)$
$(0,0,1)$
$(1,1,0)$
$(1,1,1)$

$f_3 \leftarrow Union(f_1, f_2)$

Opérations


MDD_{f_1}
MDD_{f_2}
MDD_{f_3}
A
B
C
1
0
1
Intersec
⇔
états
(0, 1, 0)
(0, 1, 1)
(1, 0, 0)
(1, 1, 0)
état
(1, 1, 0)
état
(1, 1, 0)
$f_3 \leftarrow Intersec(f_1, f_2)$

Implantation

- Les noeuds d'un MDD sont séparés en K groupes (niveaux)
- Chaque groupe est géré par une *table de hachage*
- Un temps constants d'accès pour chaque élément
- Un *cache* est utilisé pour réutiliser des opérations avec les mêmes paramètres

Applications

- Performante pour le stockage de sous-ensembles d'espace produit
- Utilisé pour la génération du RSS de modèles décrits par les *Réseaux de Petri Stochastiques* (SPN)
  - fonction caractéristique des espaces d'états locaux d'un modèle structuré

Réseaux d'Automates Stochastiques (SAN)

- Permet la modélisation de systèmes complexes à grand espace d'états
- Équivalent au formalisme des Chaînes de Markov
- Modélisation d'un système par des petits sous-systèmes presque indépendants
  - parallélisme : les automates n'interagissent pas
  - synchronisme : les automates interagissent

Réseaux d'Automates Stochastiques (SAN)

- Chaque sous-système est représenté par un automate stochastique
- Les événements locaux changent l'état local d'un seul automate
- Les événements synchronisants changent l'état local de deux ou plus automates simultanément

Réseaux d'Automates Stochastiques (SAN)

- L'interaction entre les automates :
  - Événements synchronisants
  - Taux et/ou probabilités fonctionnels

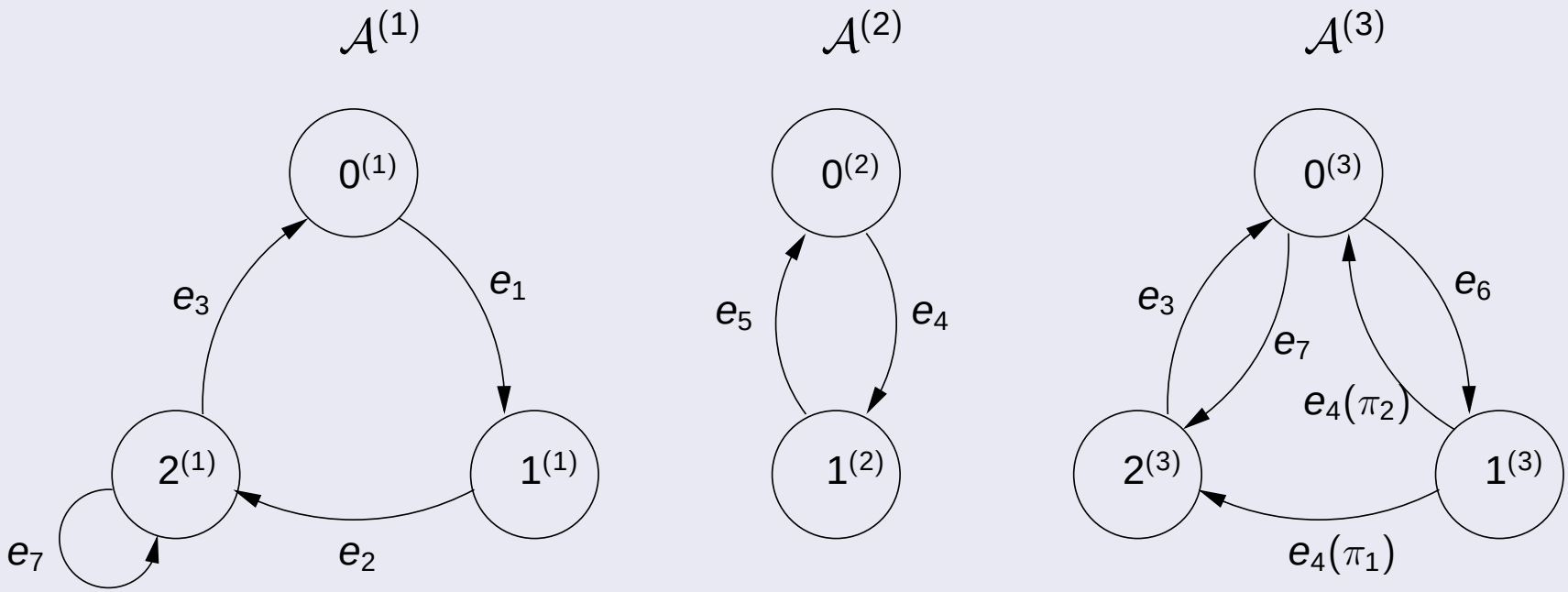
Réseaux d'Automates Stochastiques (SAN)

- L'interaction entre les automates :
  - Événements synchronisants
  - Taux et/ou probabilités fonctionnels

Taux et probabilités fonctionnels

Un des avantages de l'utilisation des fonctions est de décrire, d'une façon plus simple et efficace, des interactions complexes entre les composants du système modélisé.

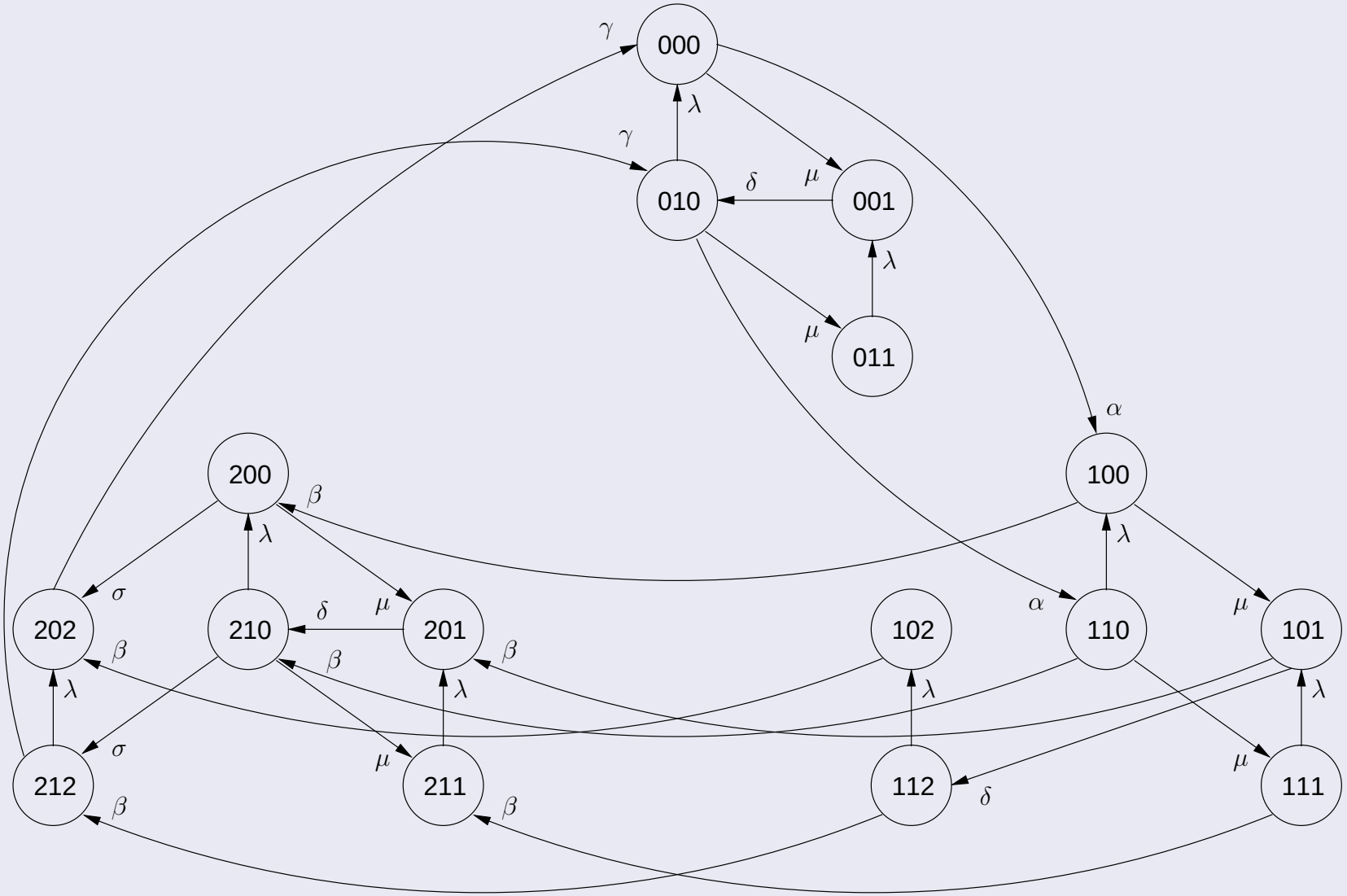
SAN (Exemple)



$f_1 = (st\mathcal{A}^{(3)} == 0) * \alpha$ $\pi_1 = (st\mathcal{A}^{(1)} == 1)$ $\pi_2 = 1 - \pi_1$

Type	Even.	Taux	Type	Even.	Taux	Type	Even.	Taux
loc	e_1	f_1	syn	e_4	δ	loc	e_6	μ
loc	e_2	β	loc	e_5	λ	syn	e_7	σ
syn	e_3	γ						

SAN (Markov)


000
010
001
011
100
200
202
210
201
102
110
101
212
211
112
111
γ
λ
δ
μ
α
β
σ

MDD pour les modèles SAN

Est-ce possible d'utiliser les MDD
pour obtenir le RSS d'un modèle SAN ?

MDD pour les modèles SAN

Est-ce possible d'utiliser les MDD
pour obtenir le RSS d'un modèle SAN ?

Démarche analogue

- Sous-systèmes (SPN) $\leftrightarrow$ Automates (SAN)

MDD pour les modèles SAN

Est-ce possible d'utiliser les MDD
pour obtenir le RSS d'un modèle SAN ?

Démarche analogue

- Sous-systèmes (SPN) $\leftrightarrow$ Automates (SAN)
- Transitions (SPN) $\leftrightarrow$ Événements (SAN)

MDD pour les modèles SAN

Alors, pourquoi ne peut-on pas utiliser les méthodes classiques de génération du RSS utilisant des MDD sur le formalisme SAN ?

MDD pour les modèles SAN

Alors, pourquoi ne peut-on pas utiliser les méthodes classiques de génération du RSS utilisant des MDD sur le formalisme SAN ?

Quel est la différence ?

MDD pour les modèles SAN

Alors, pourquoi ne peut-on pas utiliser les méthodes classiques de génération du RSS utilisant des MDD sur le formalisme SAN ?

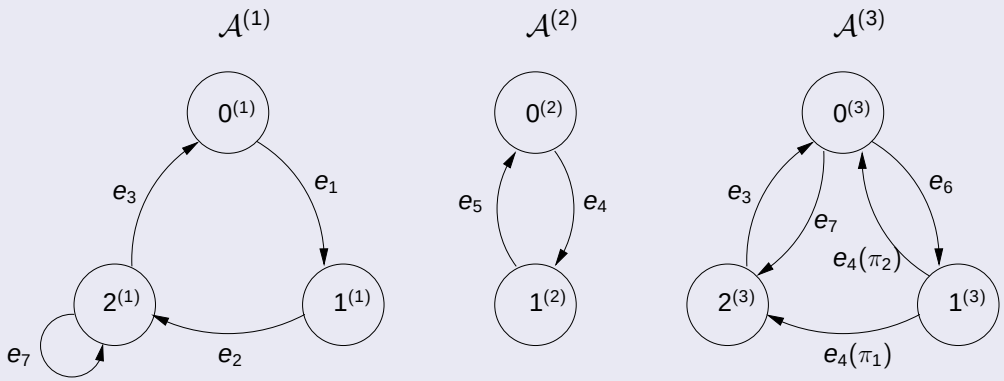
Quel est la différence ?

Les taux et probabilités fonctionnels

MDD pour les modèles SAN

- L'espace d'états d'un automate est représenté par un niveau (*groupe*) du MDD
- MDD_f représente l'ensemble d'états de la fonction f, où f ne s'annule pas
- L'évaluation de la fonction f
  - non-nulle : valeur 1 (*noeud terminal*)
  - nulle : valeur 0 (*non représenté*)

Fonction représentée par un MDD (Exemple)

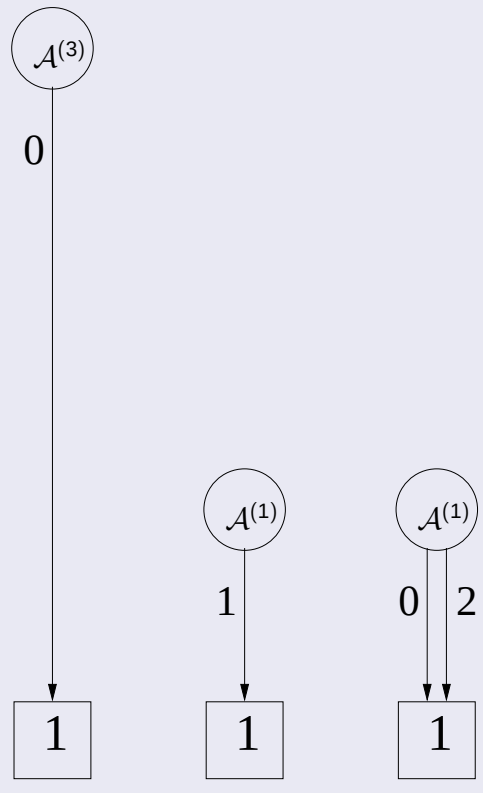


$$f_1 = (st(\mathcal{A}^{(3)}) == 0) * \alpha$$

$$\pi_1 = (st(\mathcal{A}^{(1)}) == 1)$$

$$\pi_2 = 1 - \pi_1$$

MDD_{f_1} MDD_{π_1} MDD_{π_2}



Génération du RSS de modèles avec fonctions

À partir d'un état initial S_{init} d'un modèle avec N sous-systèmes, on peut tirer exhaustivement tous les événements du modèle pour obtenir le RSS.

Génération du RSS de modèles avec fonctions

À partir d'un état initial S_{init} d'un modèle avec N sous-systèmes, on peut tirer exhaustivement tous les événements du modèle pour obtenir le RSS.

Définition : Noeud *saturé*

À partir d'un noeud saturé, le tirage d'un événement ne conduit pas à un état inexploré

Génération du RSS de modèles avec fonctions

À partir d'un état initial S_{init} d'un modèle avec N sous-systèmes, on peut tirer exhaustivement tous les événements du modèle pour obtenir le RSS.

Définition : Noeud *saturé*

À partir d'un noeud saturé, le tirage d'un événement ne conduit pas à un état inexploré

Saturation

- Explore les noeuds des niveaux bas vers les niveaux hauts du MDD
- Assure qu'un noeud saturé possède tous ses noeuds descendants saturés aussi

Algorithme de Saturation

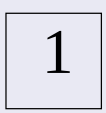
Generate(in s : vecteur $[1..K]$) : *node*
1. *MDD-Fonctions*();
2. $p \leftarrow 1$;
3. for $l = 1$ to K do
4. $\quad r \leftarrow NewNode(l)$;
5. $\quad r[s[l]] \leftarrow p$;
6. $\quad Saturate(r)$;
7. $\quad p \leftarrow r$;
8. return r;

Saturate(in,out p : *node*)
1. $\mathcal{F} \leftarrow \mathcal{E}_{p.lvl}$;
2. while $(\mathcal{F} \neq \emptyset)$ do
3. $\quad \alpha \leftarrow Pick(\mathcal{F})$;
4. $\quad$ for each $i \in Locals(\alpha, p)$ do
5. $\quad\quad f \leftarrow Fire(\alpha, p[i])$;
6. $\quad\quad$ for each $j \in \mathcal{N}_{p.lvl,\alpha}(i)$ do
7. $\quad\quad\quad u \leftarrow Union(f, p[j])$;
8. $\quad\quad\quad$ if $(u \neq p[j])$ then
9. $\quad\quad\quad\quad p[j] \leftarrow u$;
10. $\quad\quad\quad\quad \mathcal{F} \leftarrow \mathcal{E}_{p.lvl}$;
11. $p \leftarrow Check(p)$;

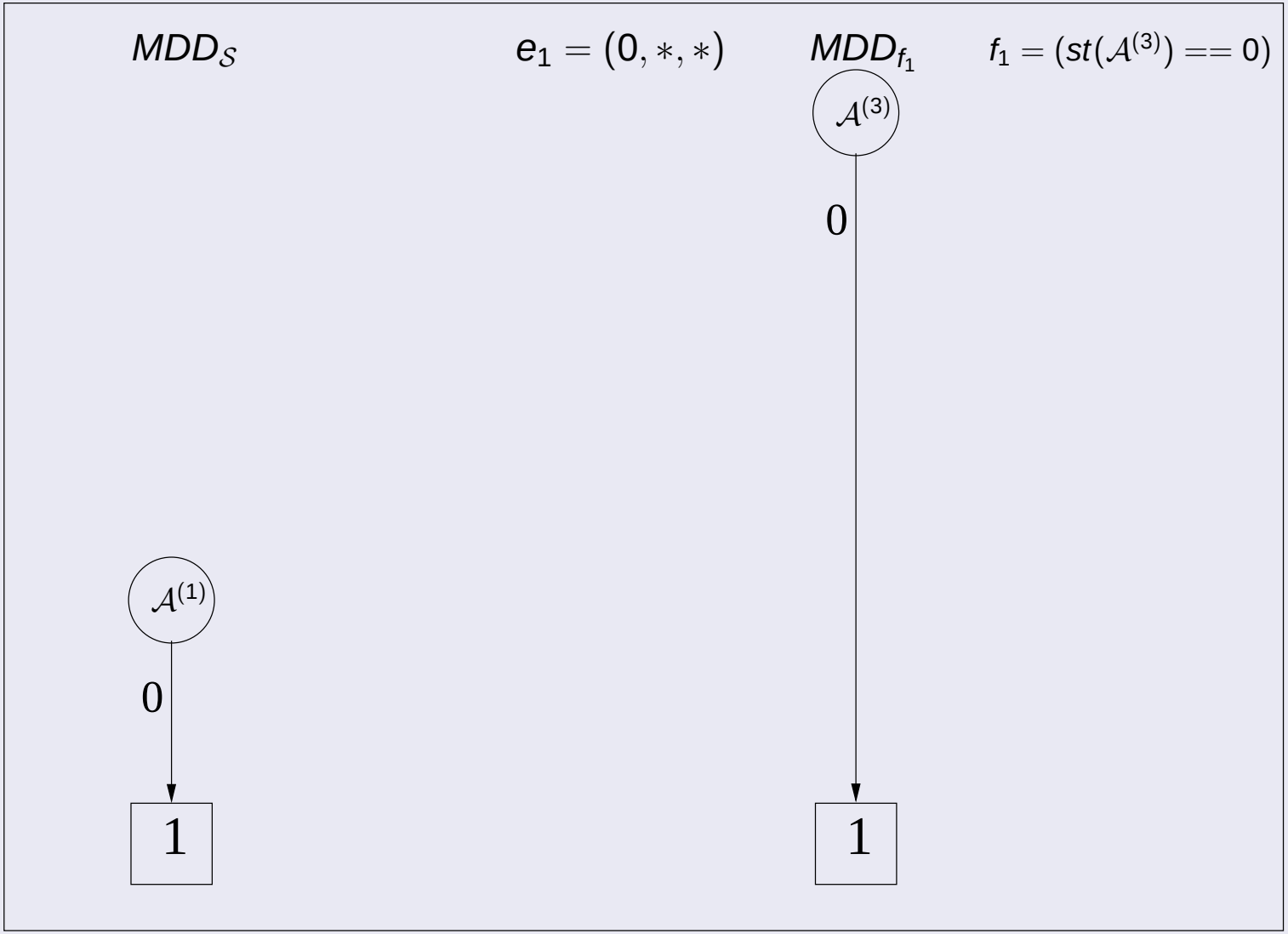
Fire(in α : *evnt*, q : *node*) : *node*
1. if $(q.lvl < Bot(\alpha))$ then return q;
2. if $(Find(FC[q.lvl], \{q, \alpha\}, s))$ then
3. $\quad$ return s;
4. $s \leftarrow NewNode(q.lvl)$;
5. for each $i \in Locals(\alpha, q)$ do
6. $\quad f \leftarrow Fire(\alpha, q[i])$;
7. $\quad$ for each $j \in \mathcal{N}_{l,\alpha}(i)$ do
8. $\quad\quad s[j] \leftarrow Union(f, s[j])$;
9. $Saturate(s)$;
10. $Insert(FC[q.lvl], \{q, \alpha\}, s)$;
11. return s;

Exemple : $S_{init} = (0, 0, 0)$

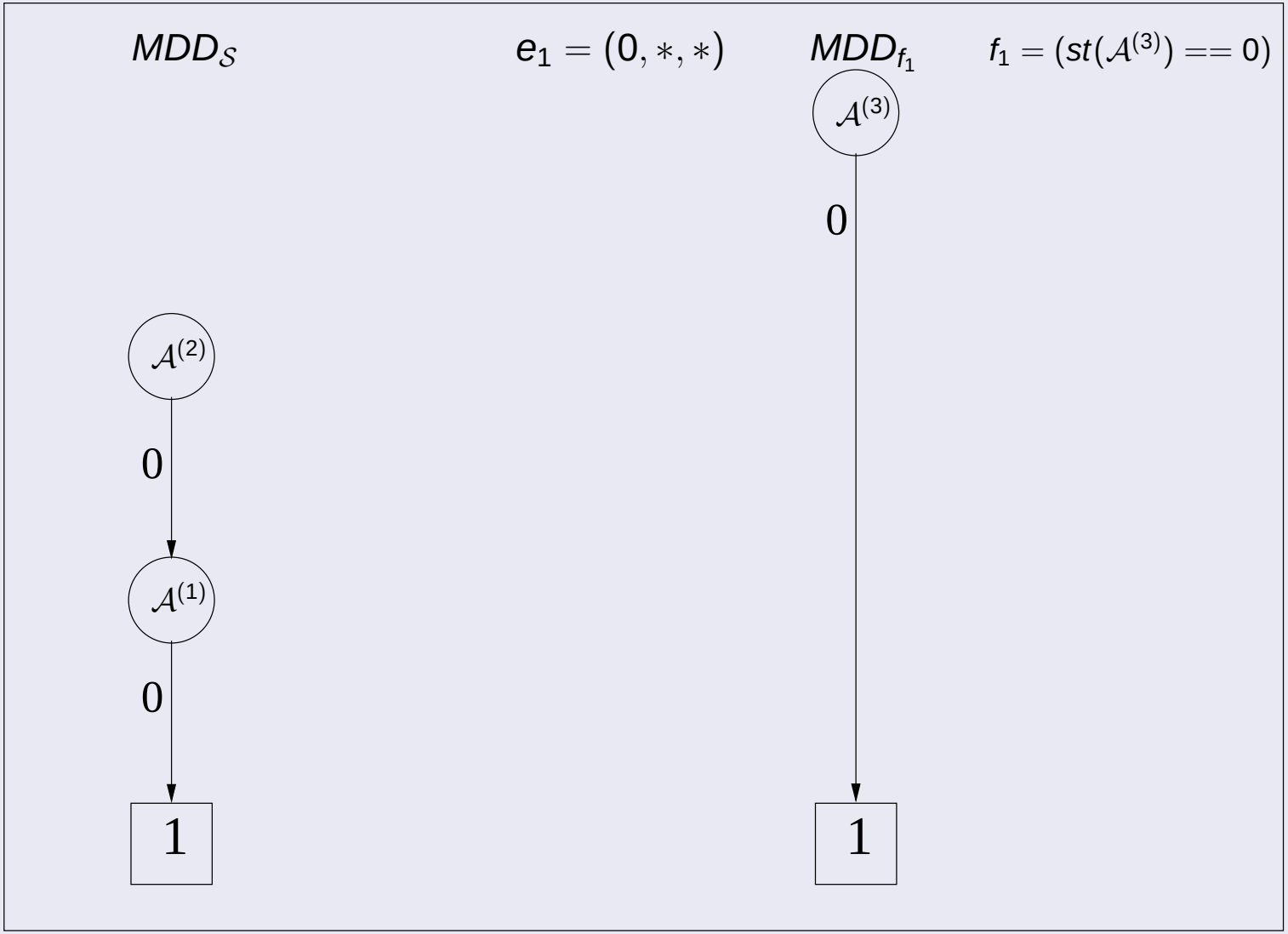
$MDD_{\mathcal{S}}$



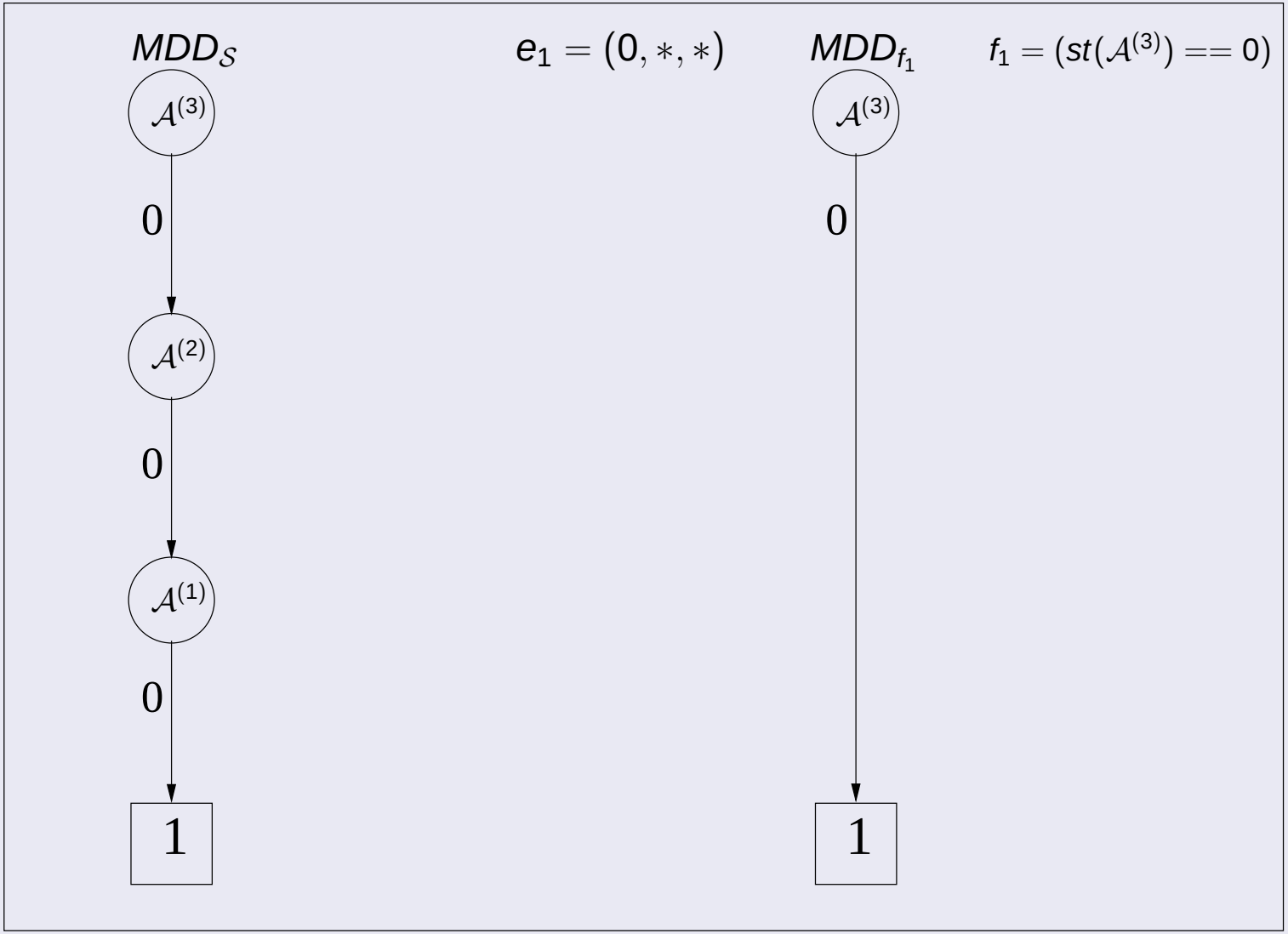
Exemple : $S_{init} = (0, 0, 0)$



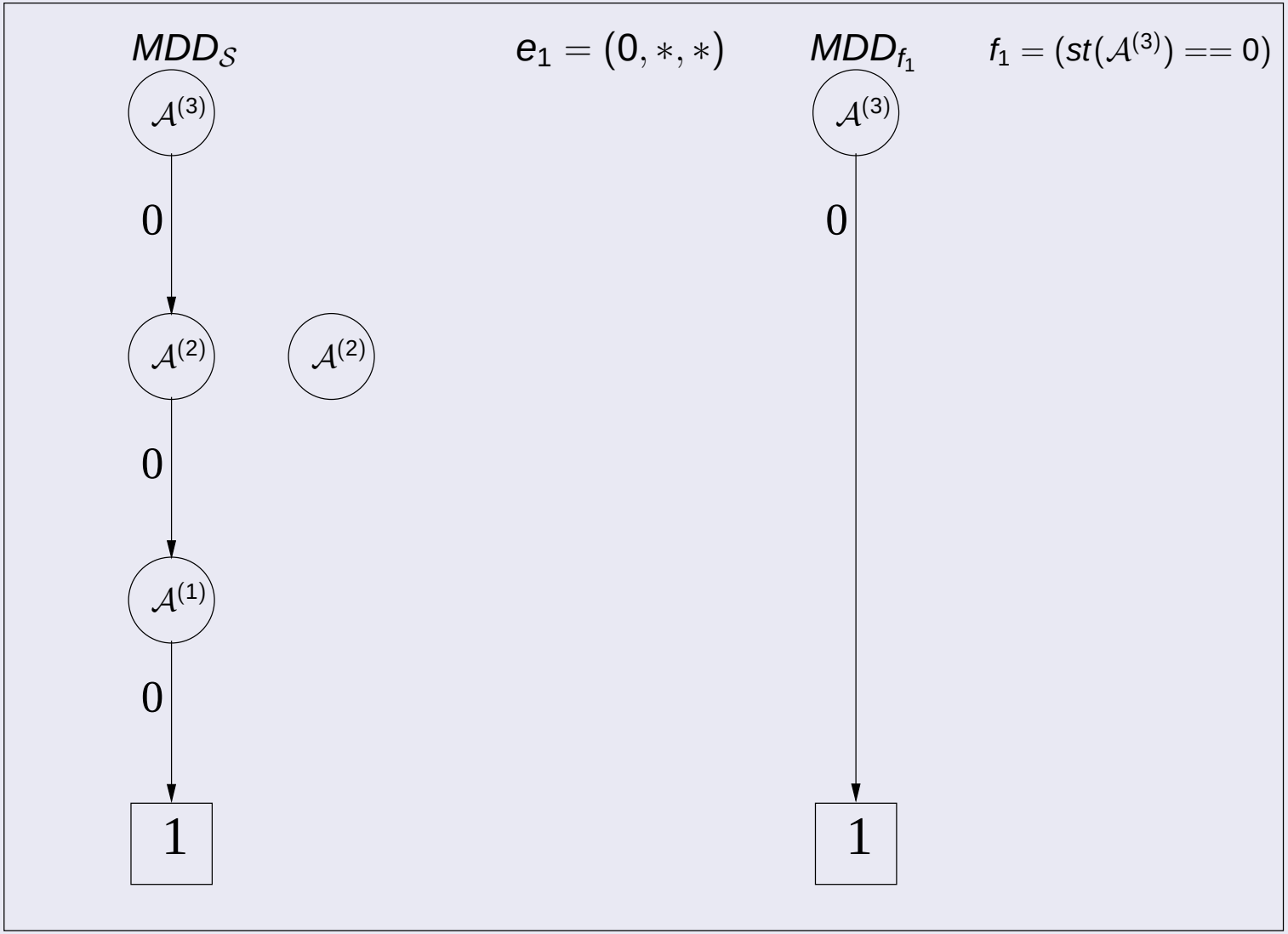
Exemple : $S_{init} = (0, 0, 0)$



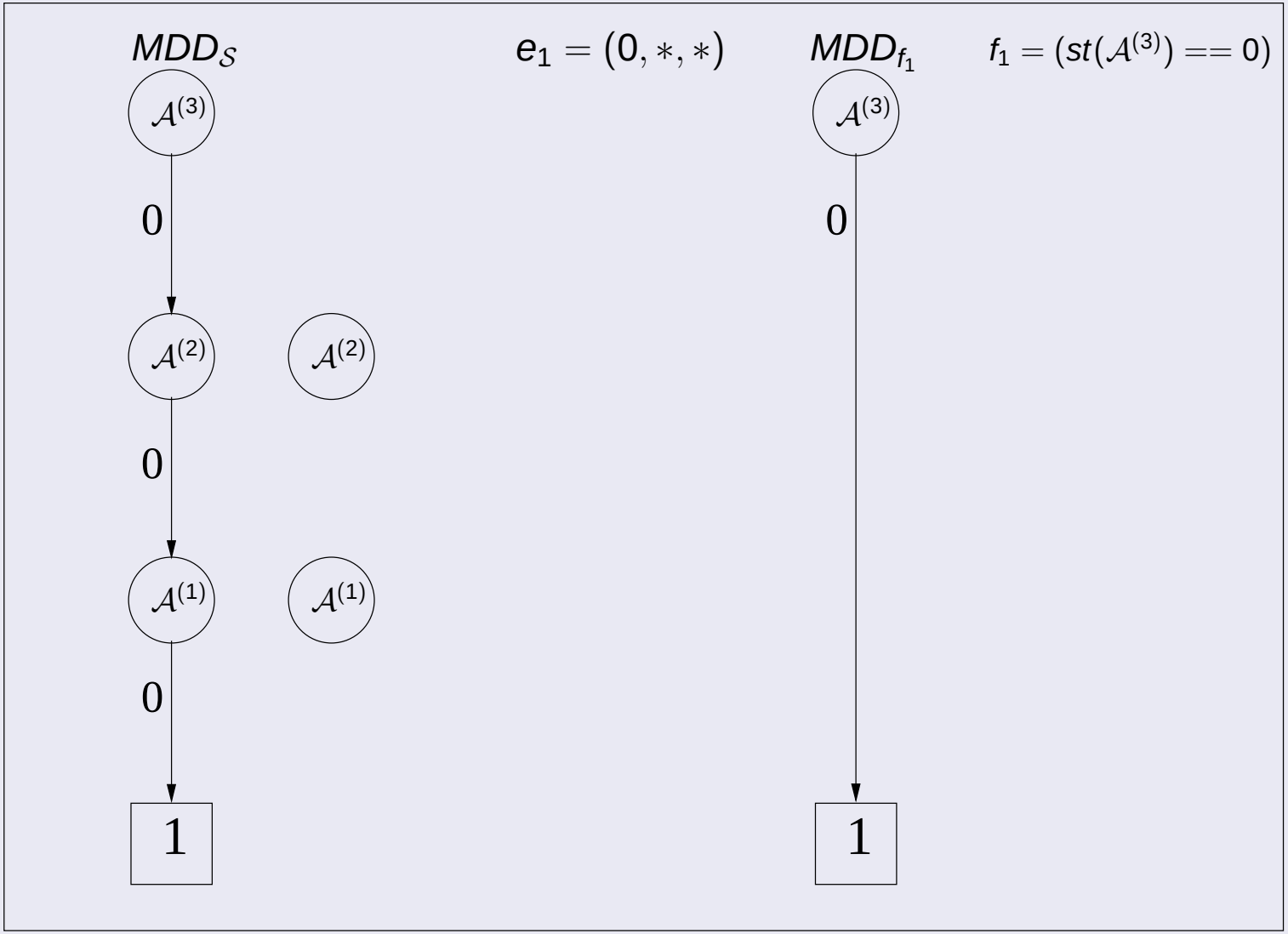
Exemple : $S_{init} = (0, 0, 0)$



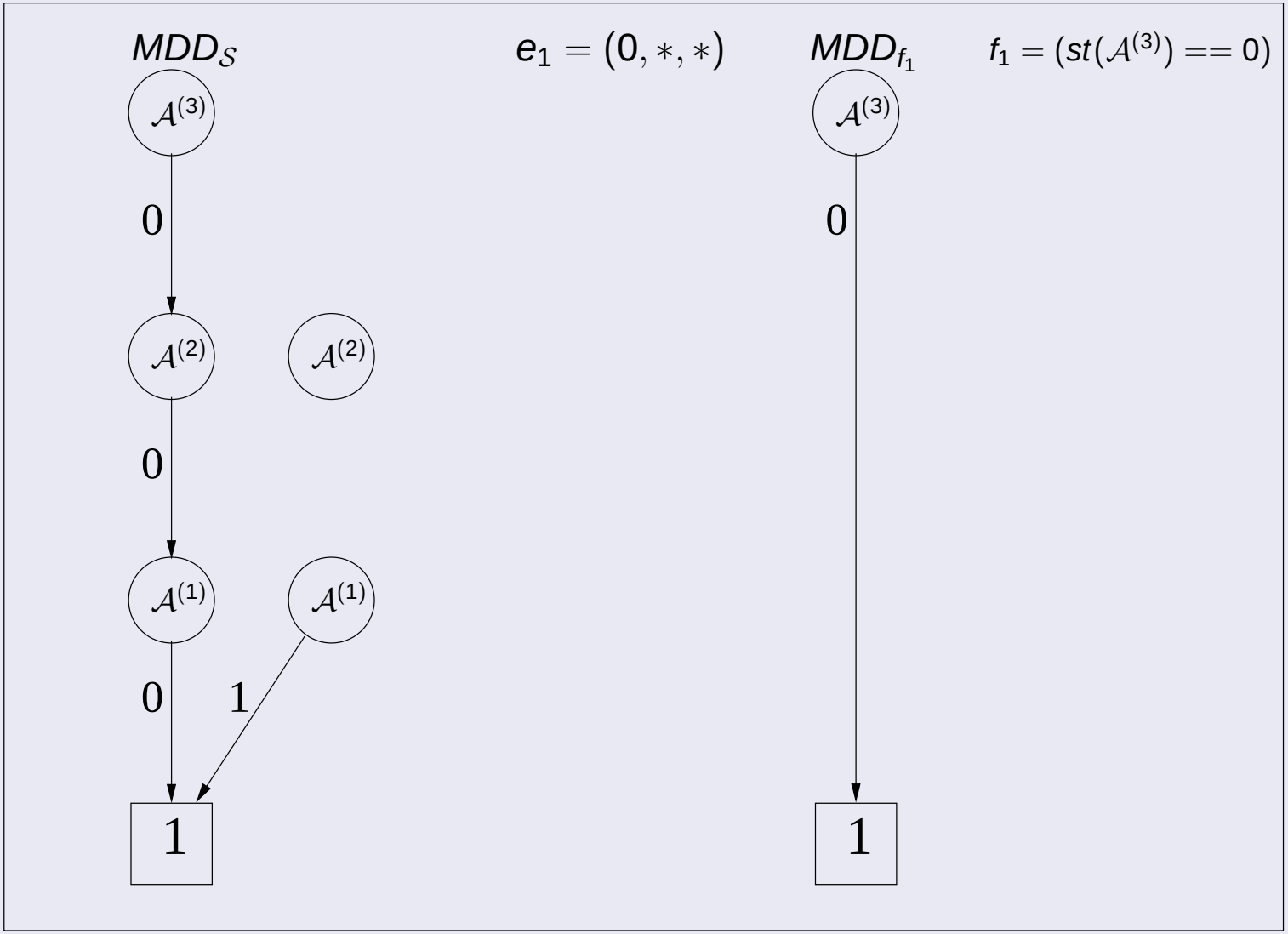
Exemple : $S_{init} = (0, 0, 0)$



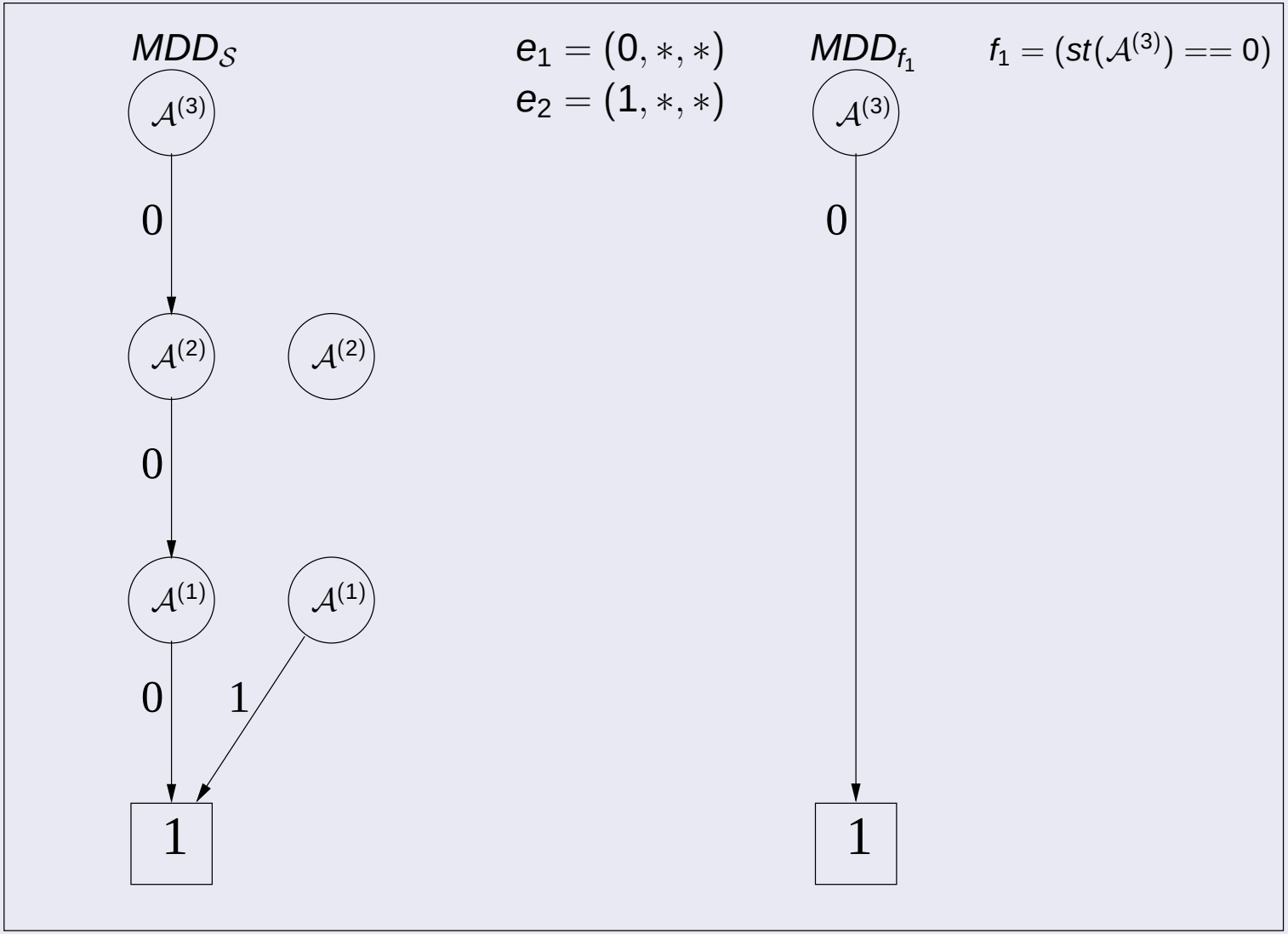
Exemple : $S_{init} = (0, 0, 0)$



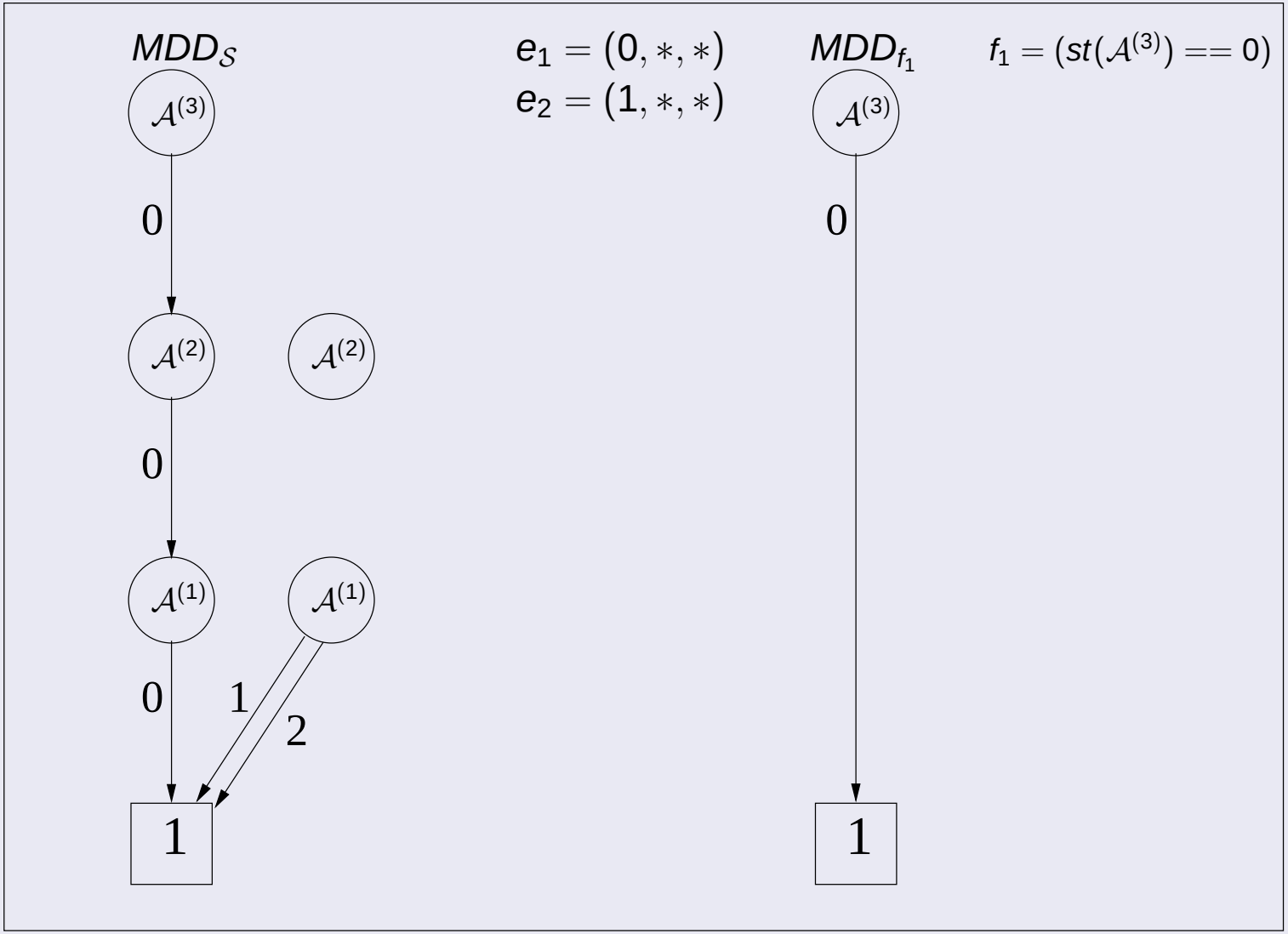
Exemple : $S_{init} = (0, 0, 0)$



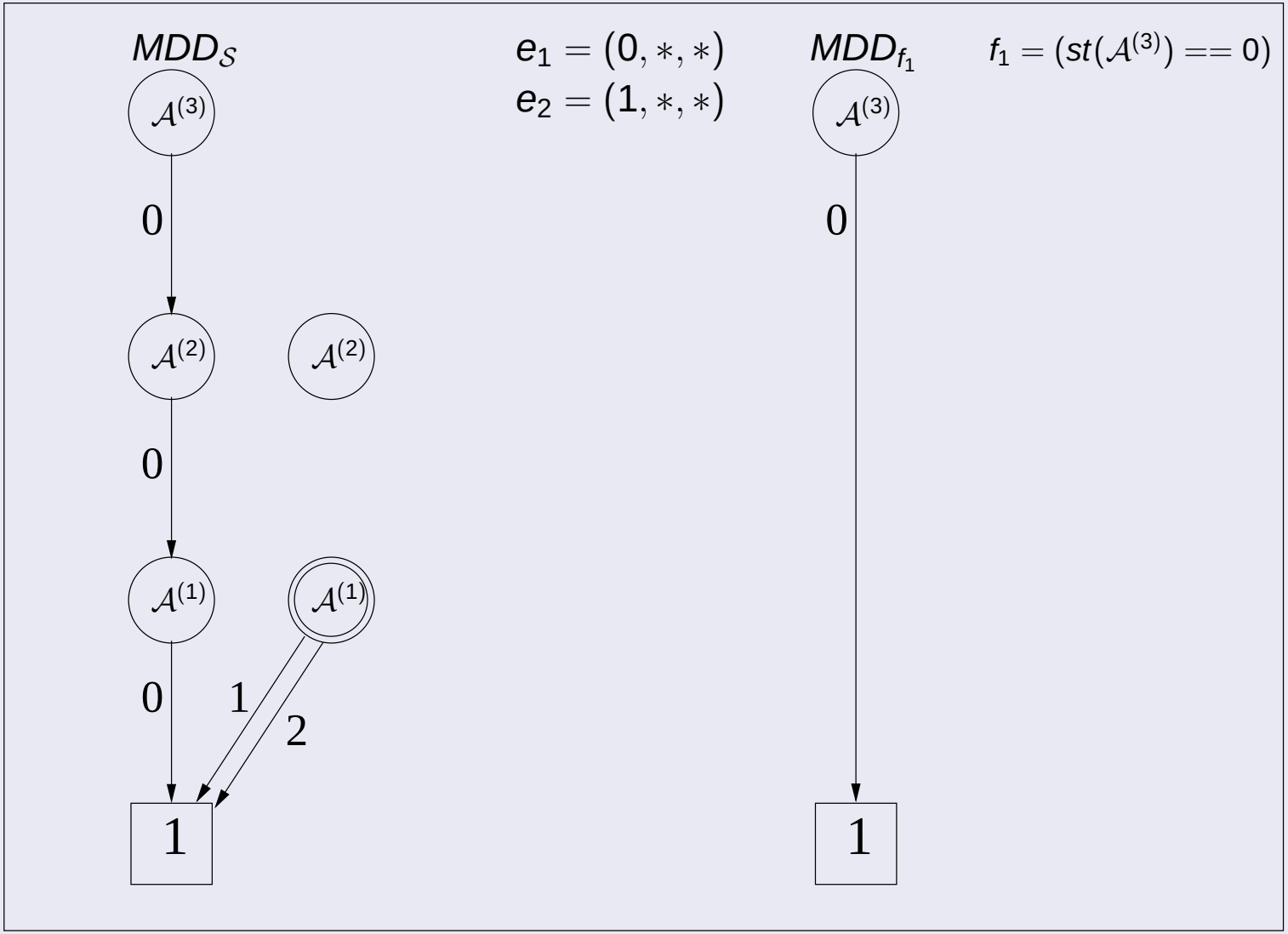
Exemple : $S_{init} = (0, 0, 0)$



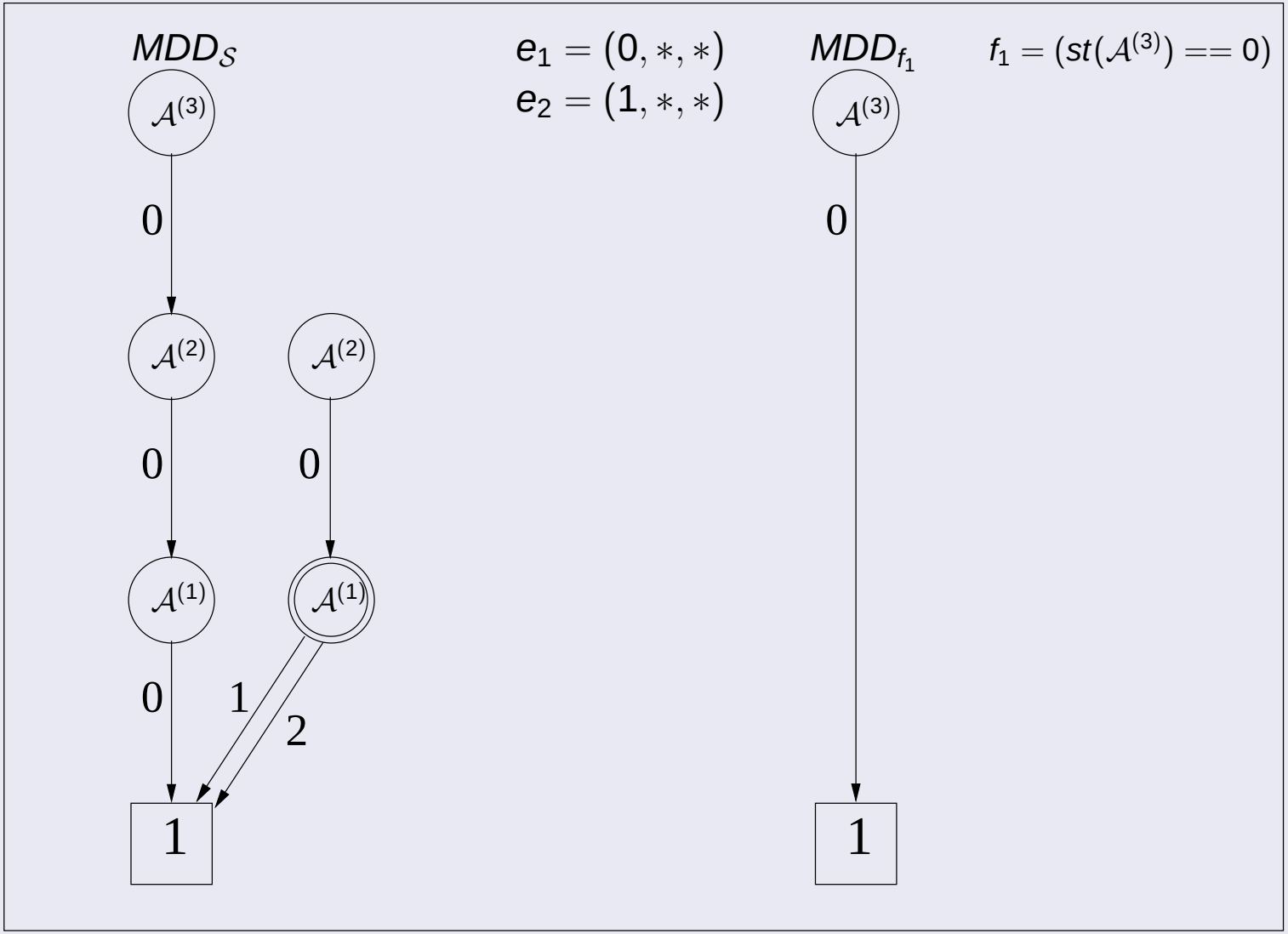
Exemple : $S_{init} = (0, 0, 0)$



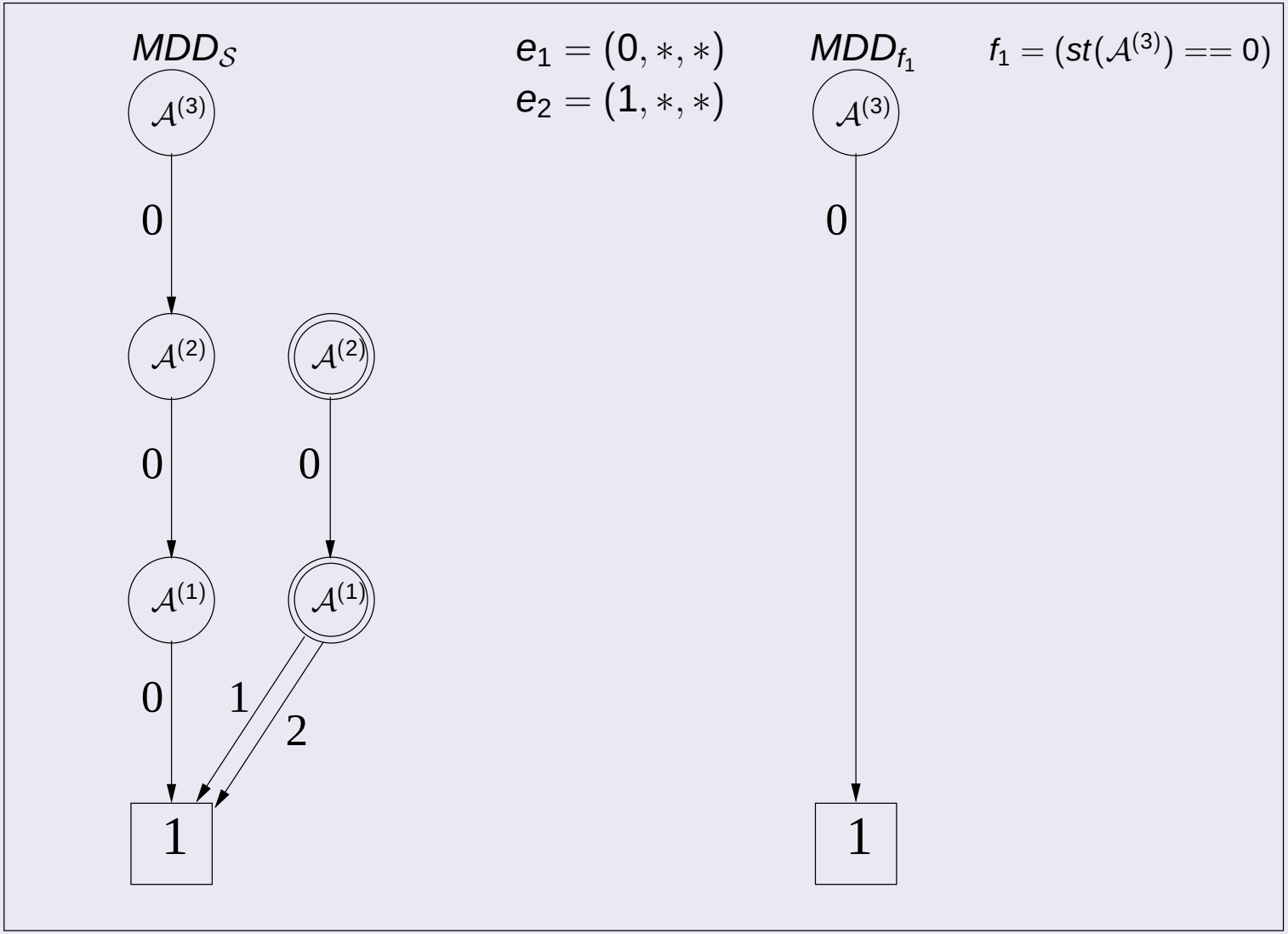
Exemple : $S_{init} = (0, 0, 0)$



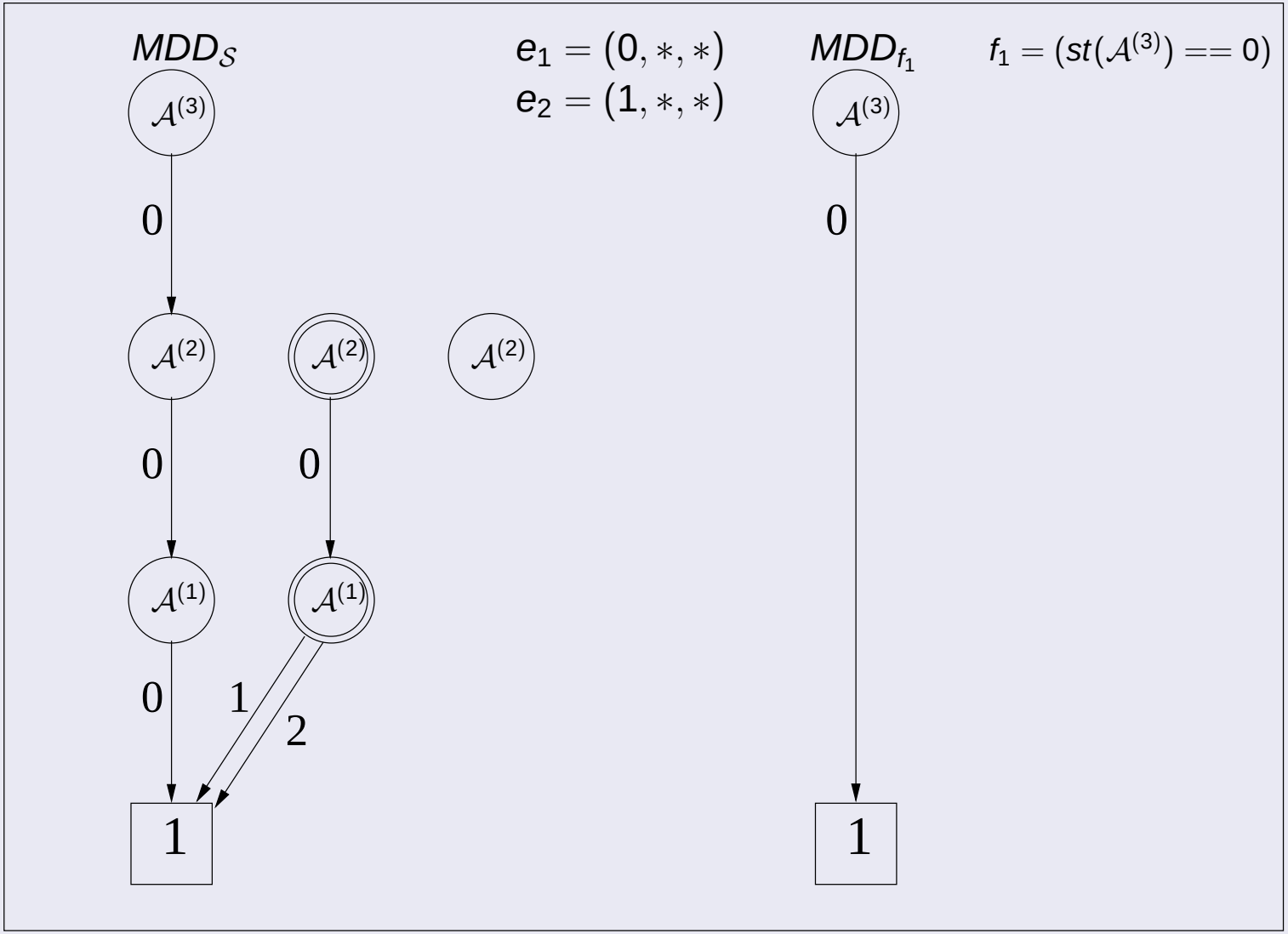
Exemple : $S_{init} = (0, 0, 0)$



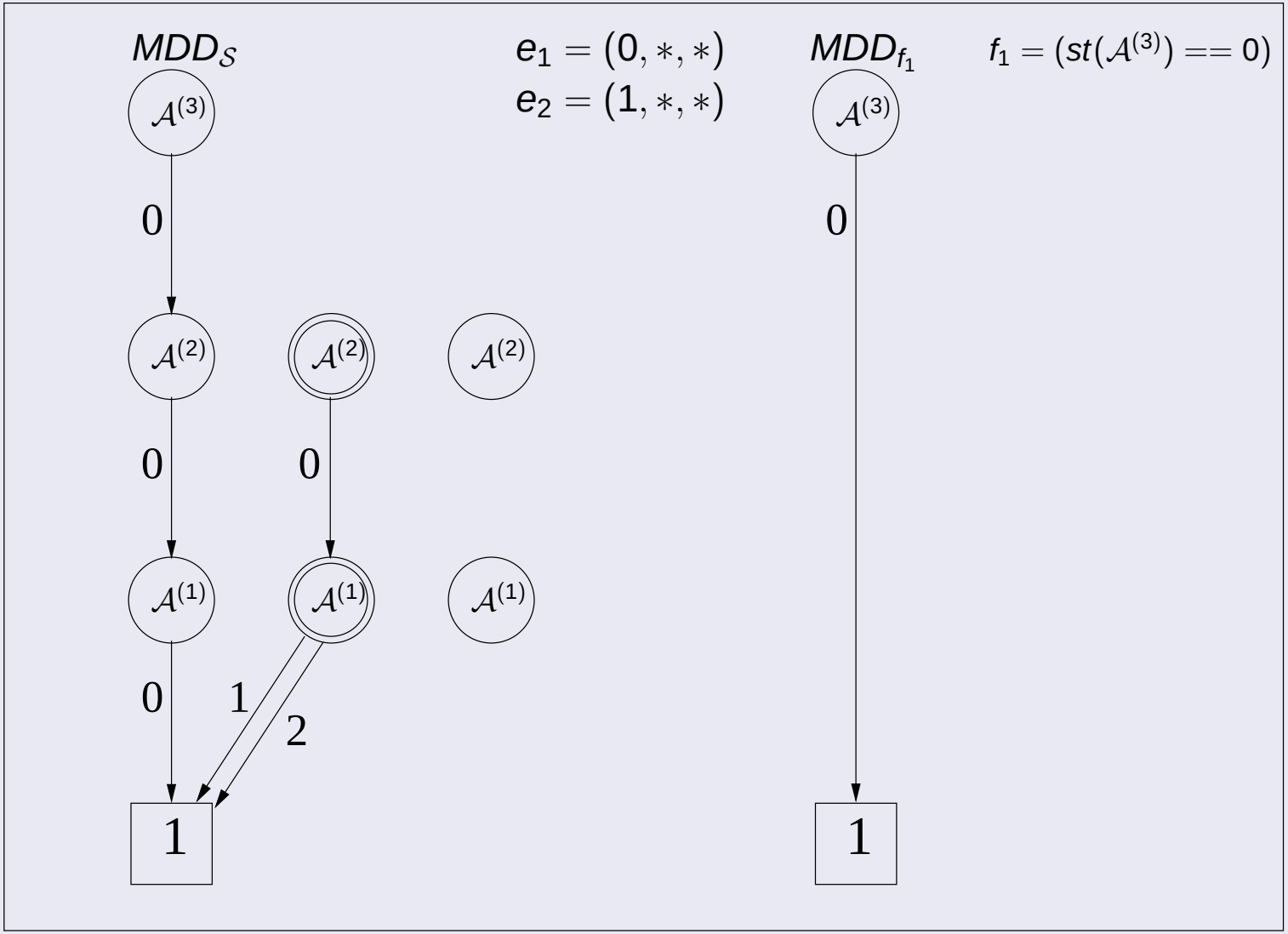
Exemple : $S_{init} = (0, 0, 0)$



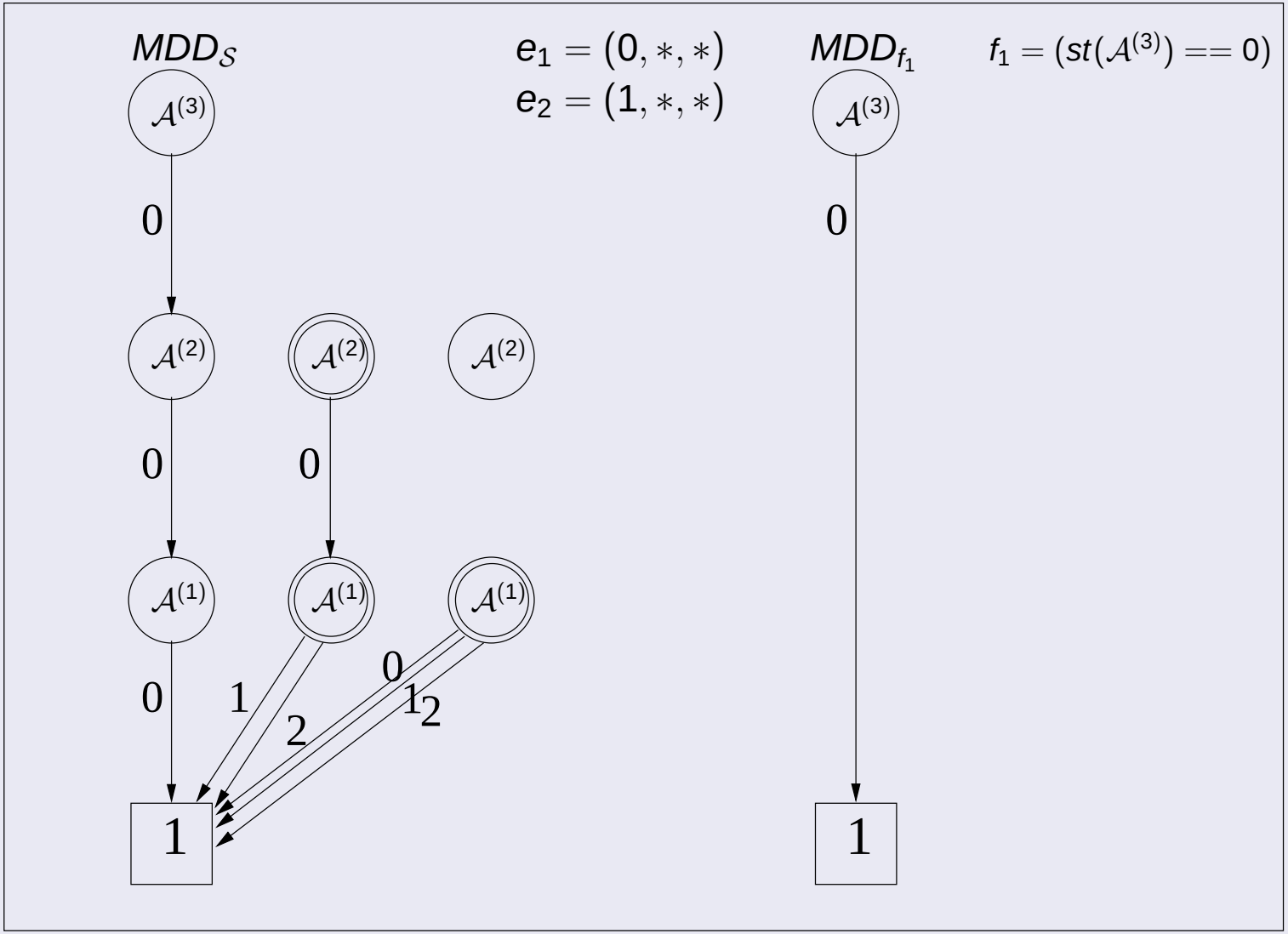
Exemple : $S_{init} = (0, 0, 0)$



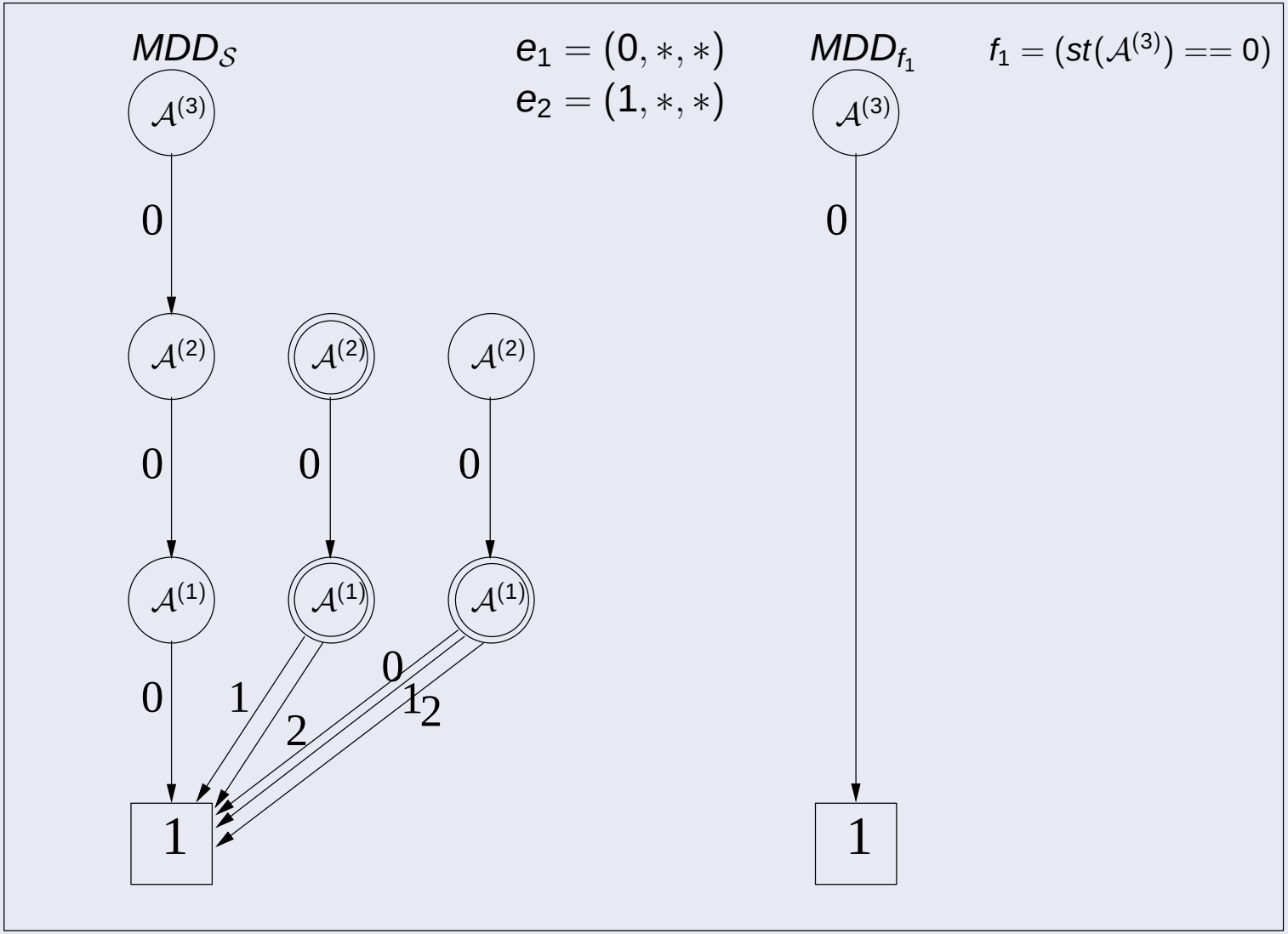
Exemple : $S_{init} = (0, 0, 0)$



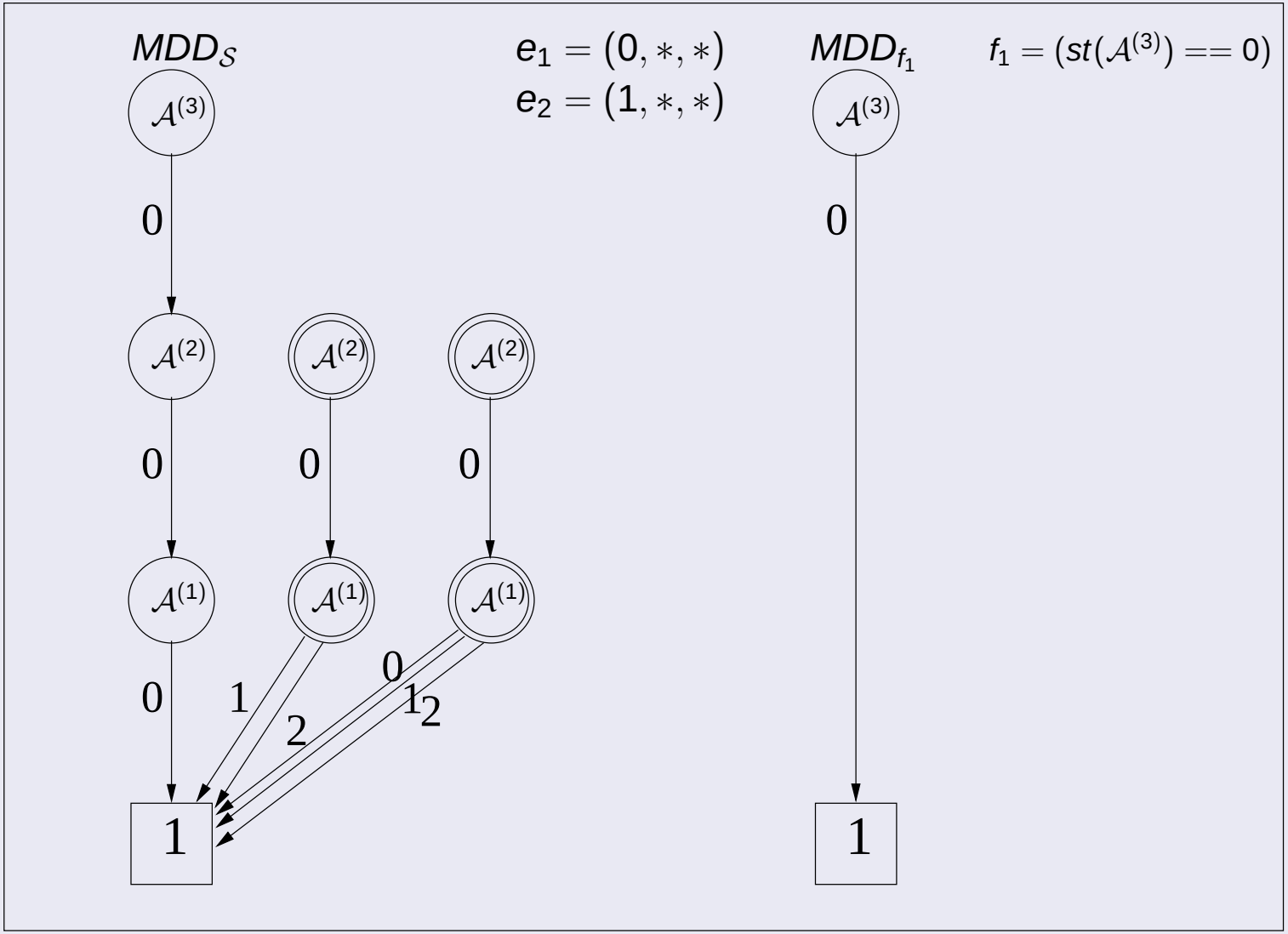
Exemple : $S_{init} = (0, 0, 0)$



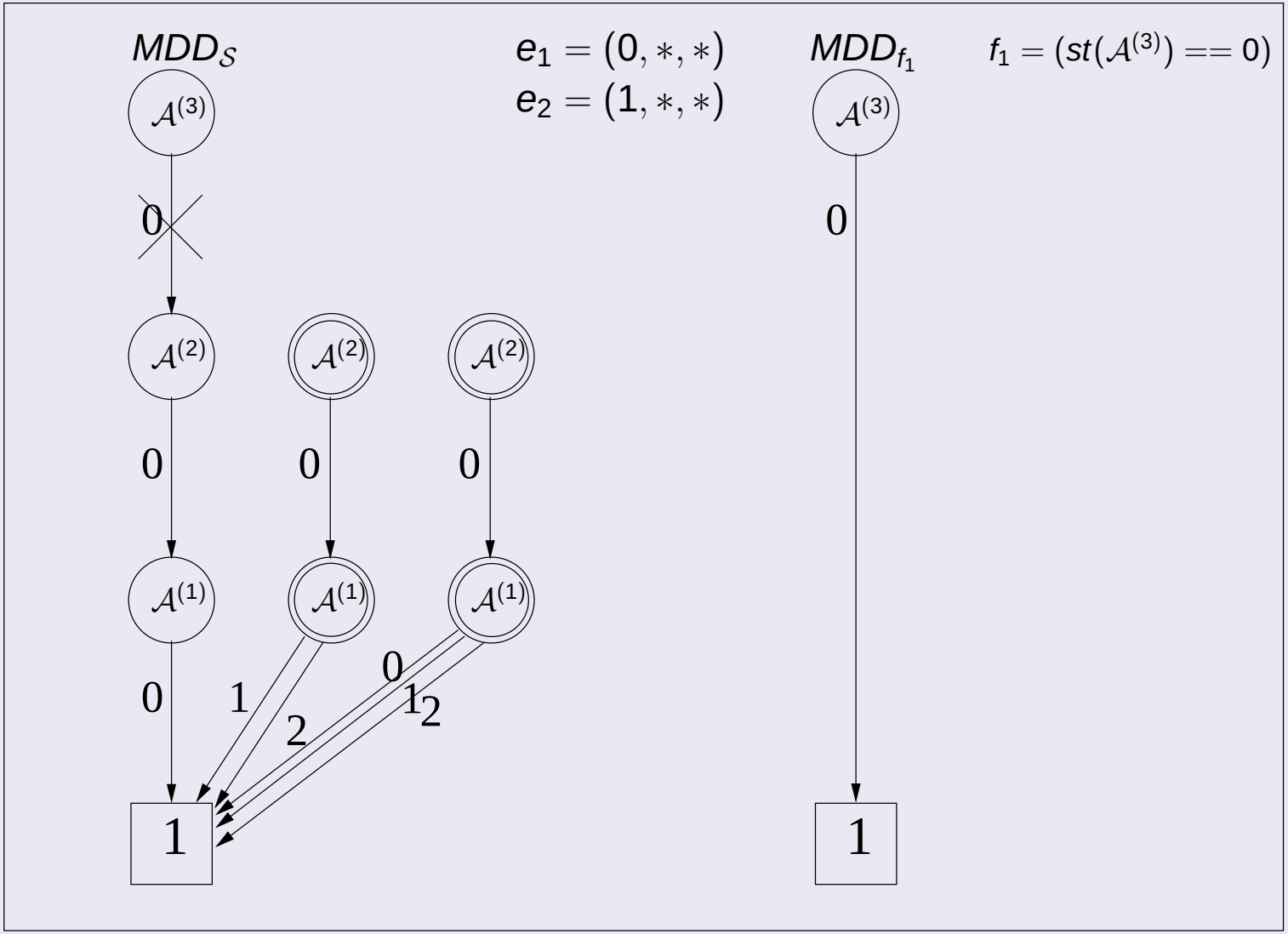
Exemple : $S_{init} = (0, 0, 0)$



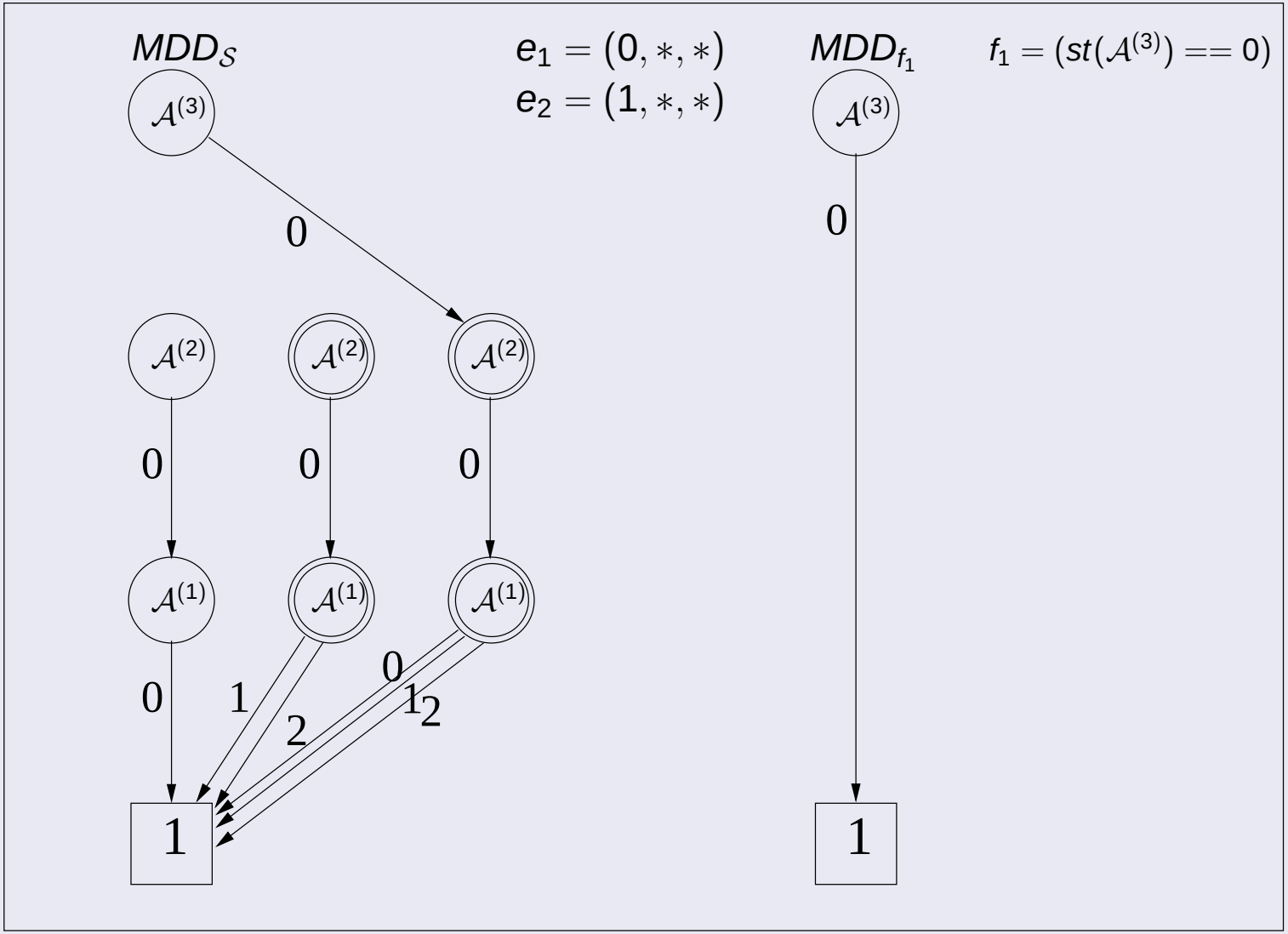
Exemple : $S_{init} = (0, 0, 0)$



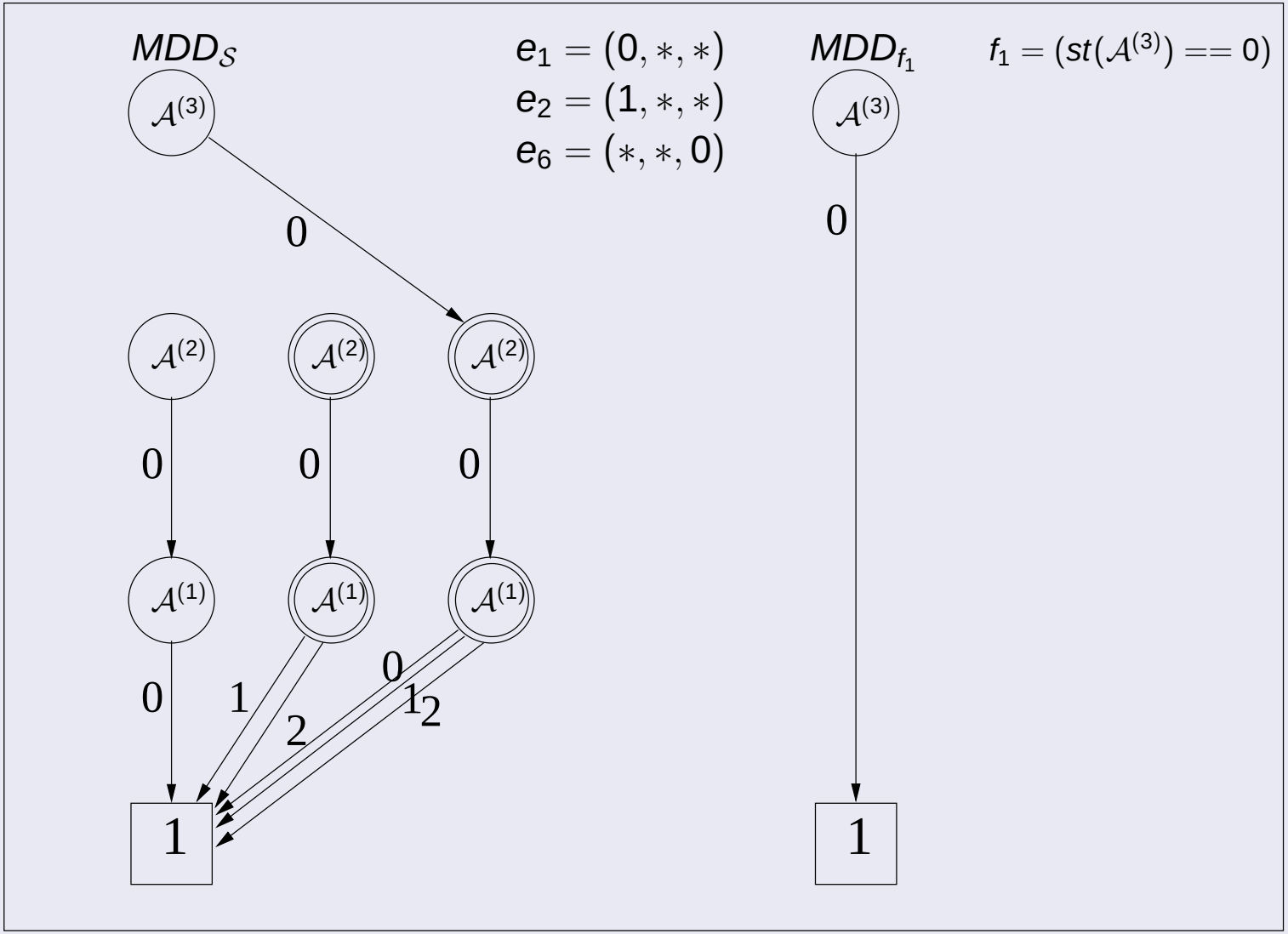
Exemple : $S_{init} = (0, 0, 0)$



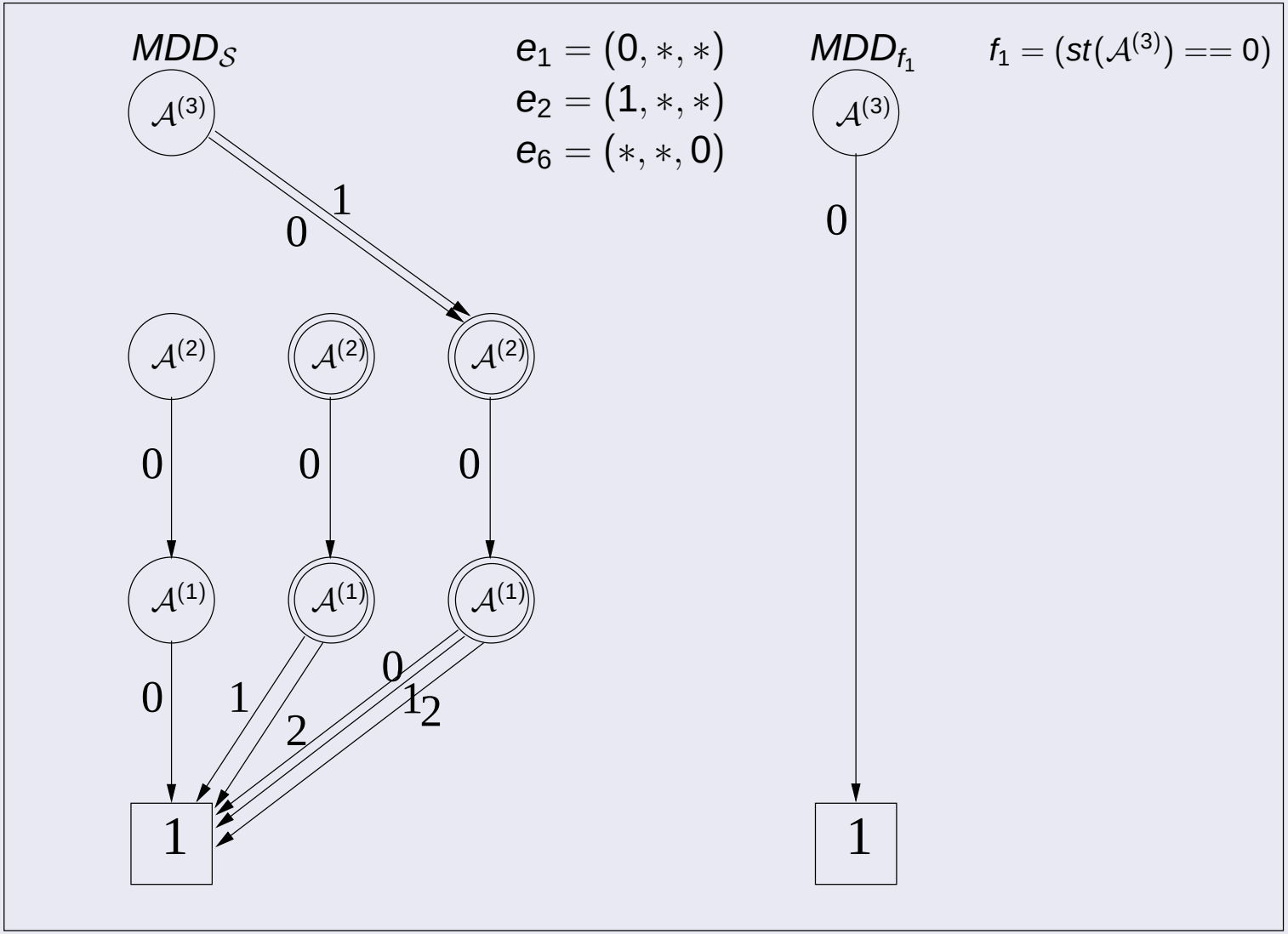
Exemple : $S_{init} = (0, 0, 0)$



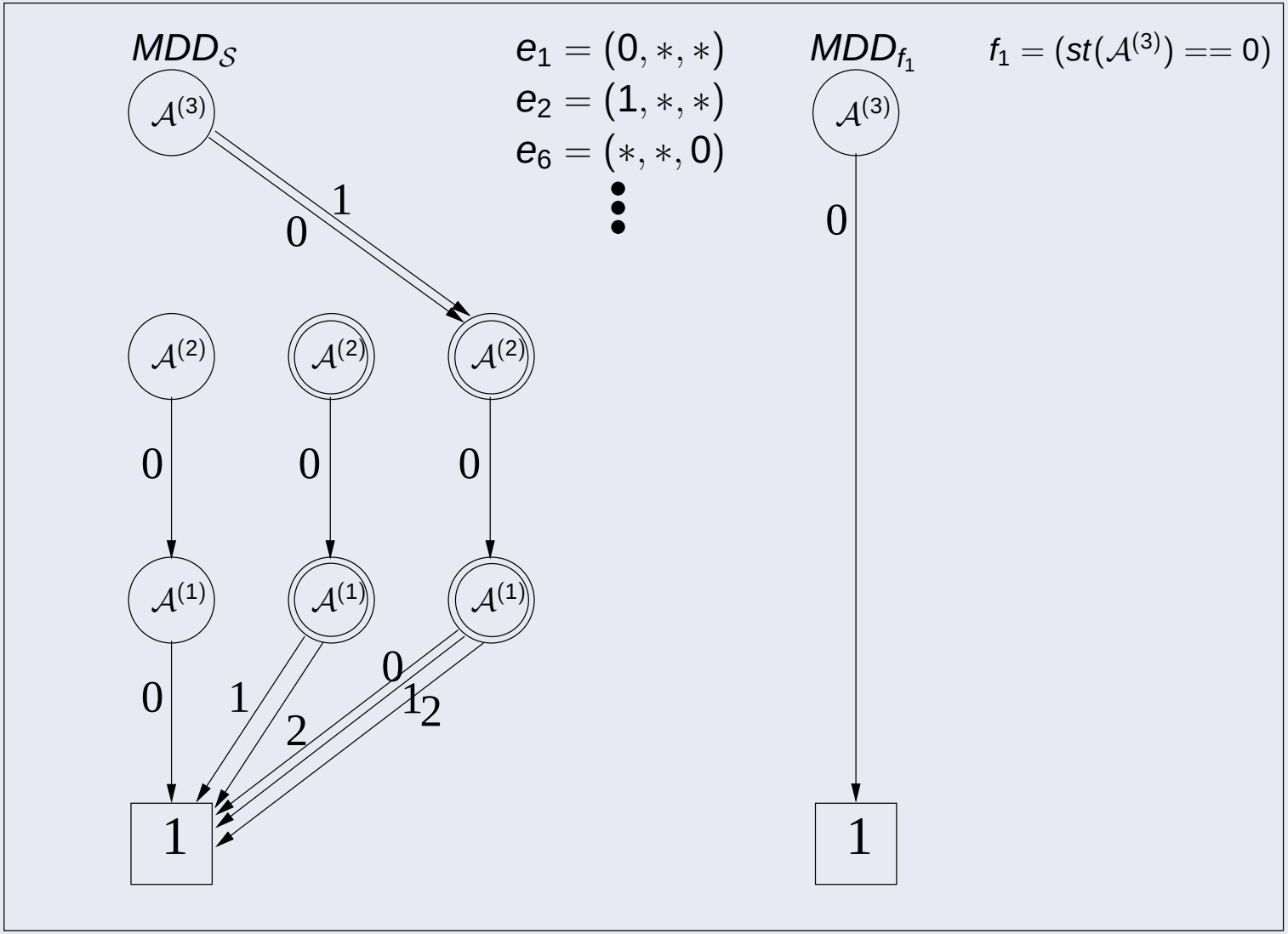
Exemple : $S_{init} = (0, 0, 0)$



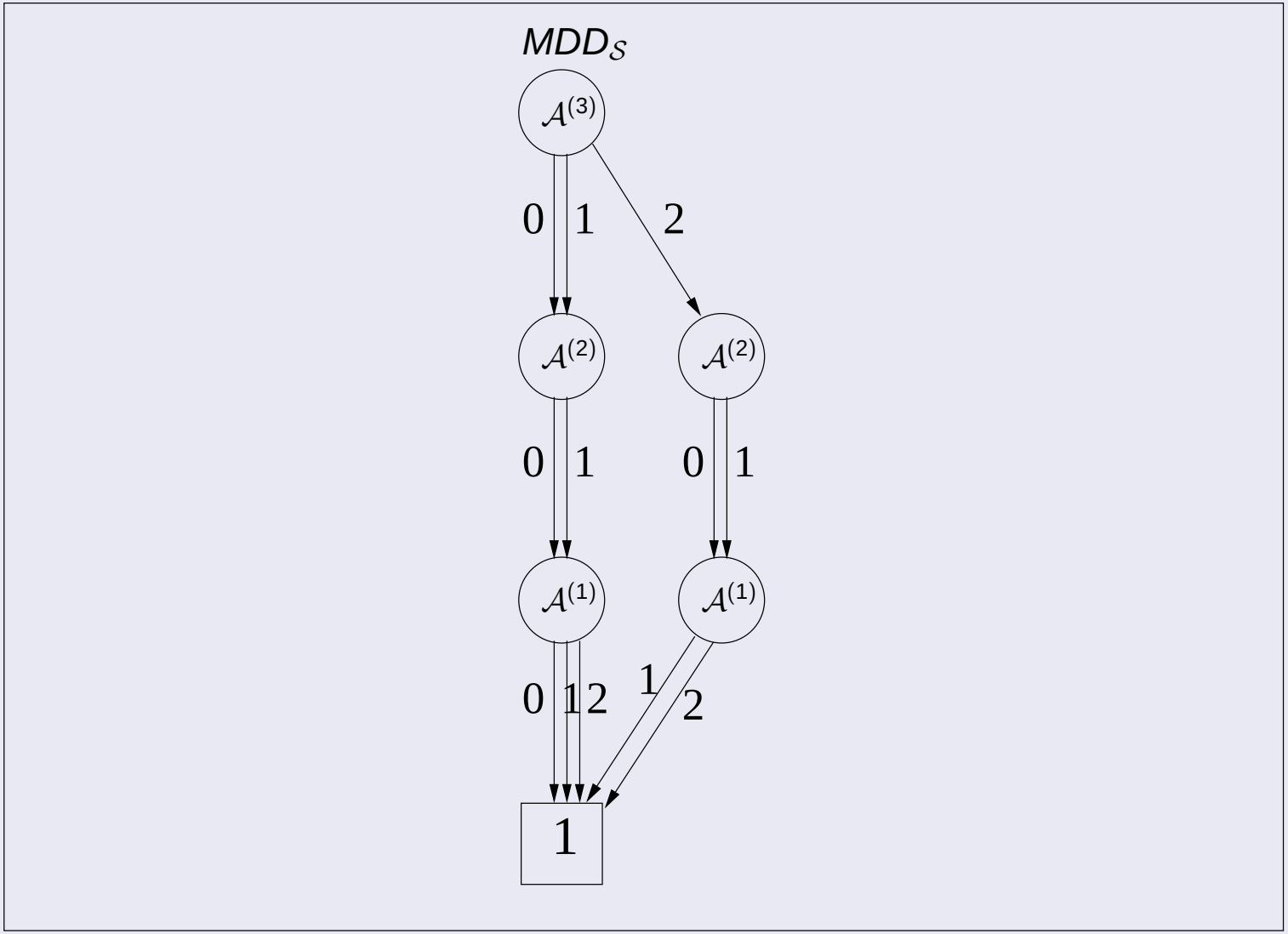
Exemple : $S_{init} = (0, 0, 0)$



Exemple : $S_{init} = (0, 0, 0)$



Exemple : $S_{init} = (0, 0, 0)$



Études de cas

- Réseau de files d'attente
- Partage de ressources
- Dîner des philosophes

Réseau de files d'attente (résultats)

$K_1 = 50\,;\, K_2 = 25\,;\, K_3 = 25\,;\, K_4 = 25\,;\, K_5 = 25$

K_{23} - K_{45}	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
$15-15$	35	4,351	7,628	1,258,342	9.43×10^{5}	2.33×10^{7}	3.06×10^{-3}	0.08
$25-25$	55	10,877	12,148	3,340,090	6.28×10^{6}	2.33×10^{7}	9.20×10^{-3}	0.32
$35-35$	35	13,972	8,988	4,501,122	1.58×10^{7}	2.33×10^{7}	5.63×10^{-3}	0.50
$45-45$	15	13,415	5,028	4,405,522	2.23×10^{7}	2.33×10^{7}	1.33×10^{-3}	0.50

$K_1 = 100\,;\, K_2 = 50\,;\, K_3 = 50\,;\, K_4 = 50\,;\, K_5 = 50$

K_{23} - K_{45}	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
$65-65$	75	4,227	22,108	1,665,604	3.92×10^{8}	6.83×10^{8}	4.74×10^{-2}	2.22×10^{-1}
$75-75$	55	2,317	17,748	958,932	5.23×10^{8}	6.83×10^{8}	2.09×10^{-2}	1.98×10^{-1}
$85-85$	35	1,007	12,588	448,548	6.22×10^{8}	6.83×10^{8}	8.76×10^{-3}	1.55×10^{-1}
$95-95$	15	297	6,628	147,828	6.75×10^{8}	6.83×10^{8}	2.66×10^{-3}	9.49×10^{-2}

Partage de ressources (résultats)

Evénements synchronisants

N = 10

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
1	21	42	5,344	30,096	1.10×10^{1}	2.05×10^{3}	1.04×10^{-3}
9	65	86	11,856	56,992	1.02×10^{3}	1.02×10^{4}	5.28×10^{-3}

N = 25

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
5	141	192	24,900	232,775	6.84×10^{4}	2.01×10^{8}	2.65×10^{-2}
24	350	401	56,136	511,847	3.36×10^{7}	8.39×10^{8}	8.84×10^{-2}

N = 100

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
5	591	792	100,800	3,170,750	7.94×10^{7}	7.61×10^{30}	5.85×10^{-1}
99	5,150	5,351	777,036	23,873,122	1.27×10^{30}	1.27×10^{32}	1.16×10^{1}

Partage de ressources (résultats)

Taux fonctionnels

N = 10

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
1	19	110	4,908	38,397	1.10×10^{1}	1.02×10^{3}	1.91×10^{-4}	8.66×10^{-4}
5	35	2,593	7,228	555,297	6.38×10^{2}	1.02×10^{3}	1.04×10^{-2}	3.53×10^{-3}
9	19	255	4,940	74,493	1.02×10^{3}	1.02×10^{3}	5.65×10^{-4}	2.09×10^{-3}

N = 20

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
5	95	63,937	17,384	12,724,827	2.17×10^{4}	1.05×10^{6}	8.02×10^{-1}	3.07×10^{-2}
10	120	2,782,111	20,968	564,781,503	6.17×10^{5}	1.05×10^{6}	6.19×10^{1}	4.09×10^{-2}
19	39	925	9,340	277,423	1.05×10^{6}	1.05×10^{6}	2.07×10^{-1}	9.15×10^{-3}

N = 25

R	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
5	125	186,434	22,408	36,664,142	6.84×10^{4}	3.36×10^{7}	9.13×10^{0}	5.79×10^{-2}
24	49	1,410	11,540	426,438	3.36×10^{7}	3.36×10^{7}	7.50×10^{0}	1.46×10^{-2}

Dîner des philosophes (résultats)

Evénements synchronisants							
N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)
	Final	Pic	Final	Pic	RSS	PSS	RSS
10	35	56	7,312	34,171	5.74×10^{3}	5.90×10^{4}	1.54×10^{-3}
50	195	296	36,752	417,251	1.17×10^{19}	7.18×10^{23}	8.96×10^{-3}
100	395	596	73,552	1,445,101	1.62×10^{38}	5.15×10^{47}	1.83×10^{-2}
1,000	3,995	5,996	735,952	124,256,401	5.09×10^{382}	1.32×10^{477}	2.16×10^{-1}

Taux fonctionnels								
N	Noeuds MDD		Mémoire (bytes)		Espace d'états		Temps (sec)	
	Final	Pic	Final	Pic	RSS	PSS	Fonc.	RSS
10	35	83	7,312	34,488	5.74×10^{3}	5.90×10^{4}	1.70×10^{-4}	1.23×10^{-3}
50	195	443	36,752	361,048	1.17×10^{19}	7.18×10^{23}	8.12×10^{-4}	7.32×10^{-3}
100	395	893	73,552	1,183,248	1.62×10^{38}	5.15×10^{47}	1.86×10^{-3}	1.52×10^{-2}
1,000	3,995	8,993	735,952	94,642,848	5.09×10^{382}	1.32×10^{477}	5.08×10^{-2}	1.89×10^{-1}

Conclusion et Perspectives

- Génération du RSS de modèles avec fonctions à très grand espace d'états
  - Stockage compact
  - Manipulation efficace
- Amélioration de la génération explicite des MDD des fonctions
- Développement d'un nouveau module pour le logiciel PEPS

Conclusion et Perspectives

- Génération du RSS de modèles avec fonctions à très grand espace d'états
  - Stockage compact
  - Manipulation efficace
- Amélioration de la génération explicite des MDD des fonctions
- Développement d'un nouveau module pour le logiciel PEPS

Conclusion et Perspectives

- Génération du RSS de modèles avec fonctions à très grand espace d'états
  - Stockage compact
  - Manipulation efficace
- Amélioration de la génération explicite des MDD des fonctions
- Développement d'un nouveau module pour le logiciel PEPS

Conclusion et Perspectives

- Génération du RSS de modèles avec fonctions à très grand espace d'états
  - Stockage compact
  - Manipulation efficace
- Amélioration de la génération explicite des MDD des fonctions
- Développement d'un nouveau module pour le logiciel PEPS

Conclusion et Perspectives

- Génération du RSS de modèles avec fonctions à très grand espace d'états
  - Stockage compact
  - Manipulation efficace
- Amélioration de la génération explicite des MDD des fonctions
- Développement d'un nouveau module pour le logiciel PEPS

Questions ?