UNIVERSITÉ DE LORRAINE

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Construction de comportements intelligents

Présentation ISN 2017 - 30/03/2017

Vincent THOMAS – Université de Lorraine
vincent.thomas@loria.fr

https://members.loria.fr/VThomas/mediation/ISN_planif

Principe

- Construire bribes intelligence
  - Relativement simple
  - Solution à des besoins étudiants
  - Nombreux exemples

- Agent intelligent / prise de décision

- 3 idées fortes
  - Représenter un problème => *MDP deterministe*
  - Propriétés de la solution => *Equation Bellman*
  - Résolution du problème => *Value iteration*

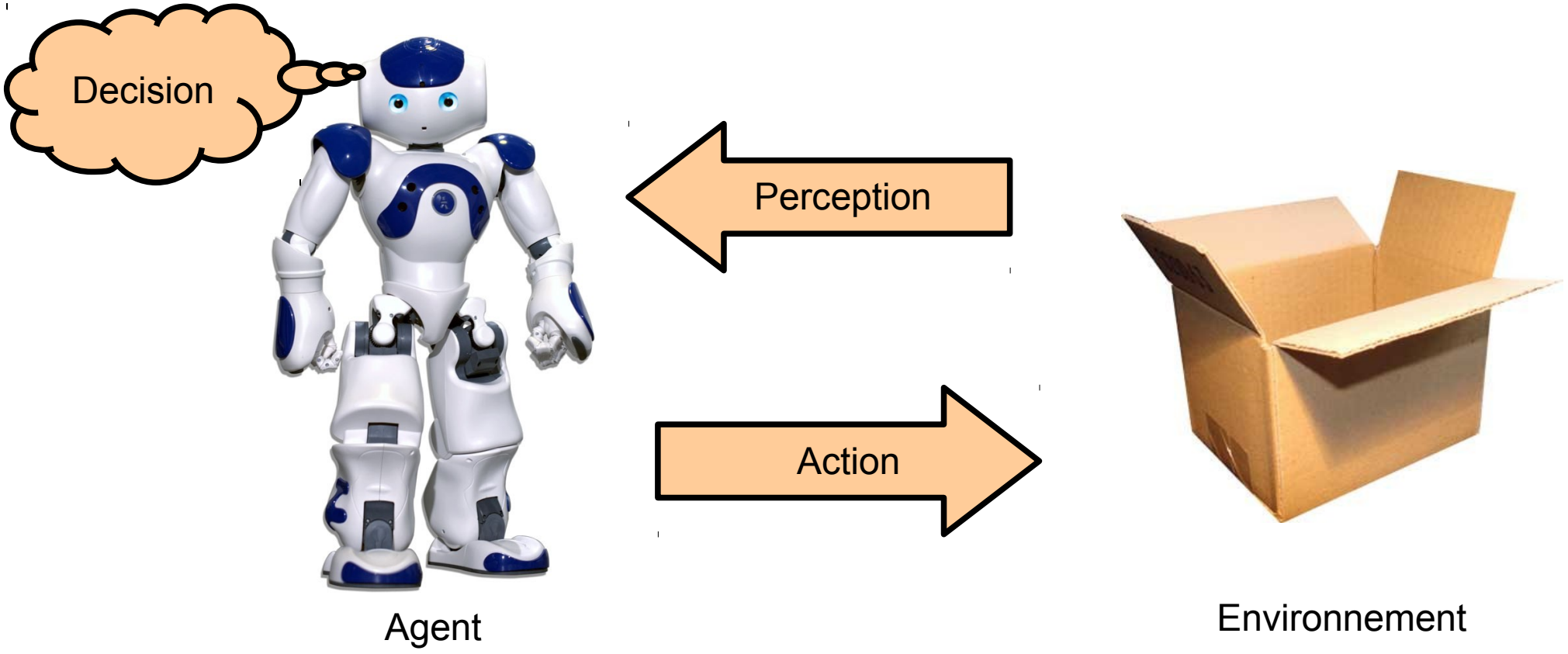
Plan

- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

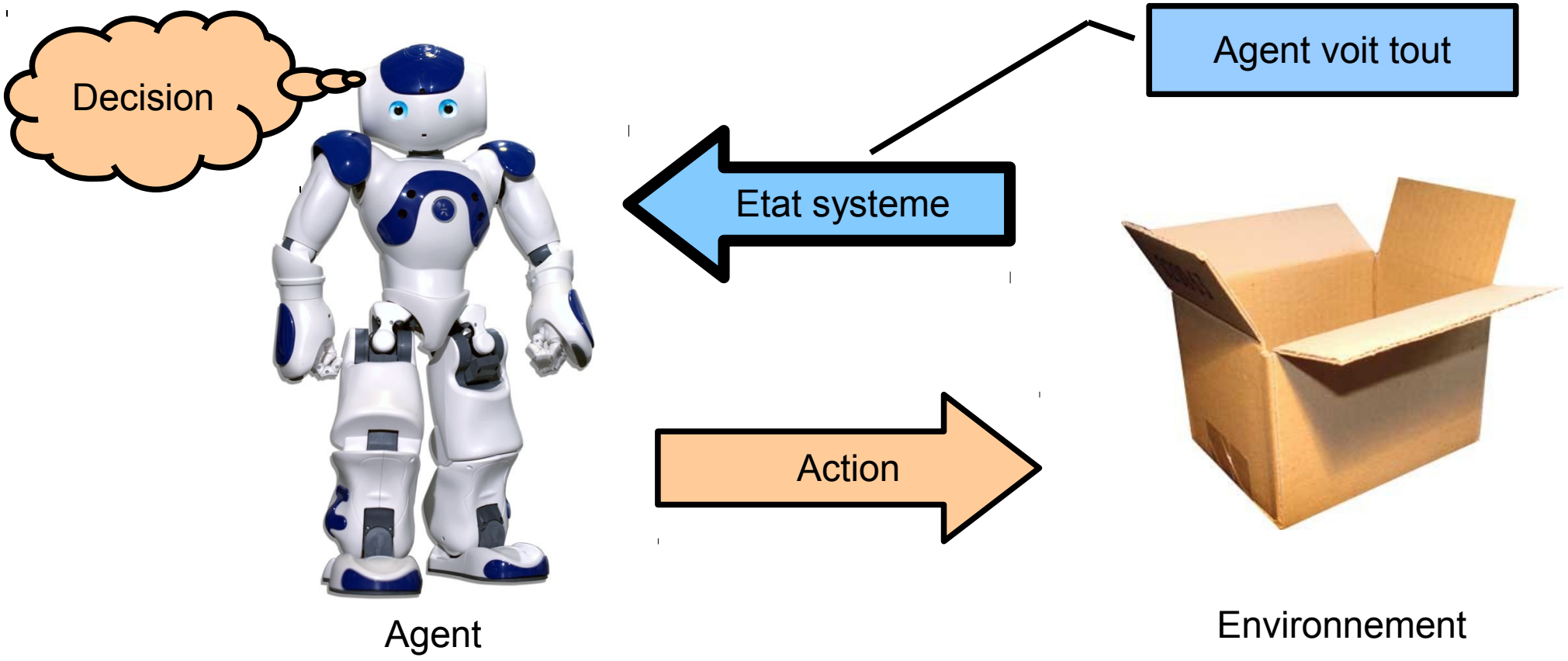
- Perspectives

Boucle perception-action



Un **agent** intelligent est une entité réelle ou artificielle, dotée de **capteurs** et **d'effecteurs**, capable d'agir de manière **autonome** grâce à une **fonction de décision**

Boucle perception-action (simple)



Un **agent** intelligent est une entité réelle ou artificielle, dotée de **capteurs** et **d'effecteurs**, capable d'agir de manière **autonome** grâce à une **fonction de décision**

Plan

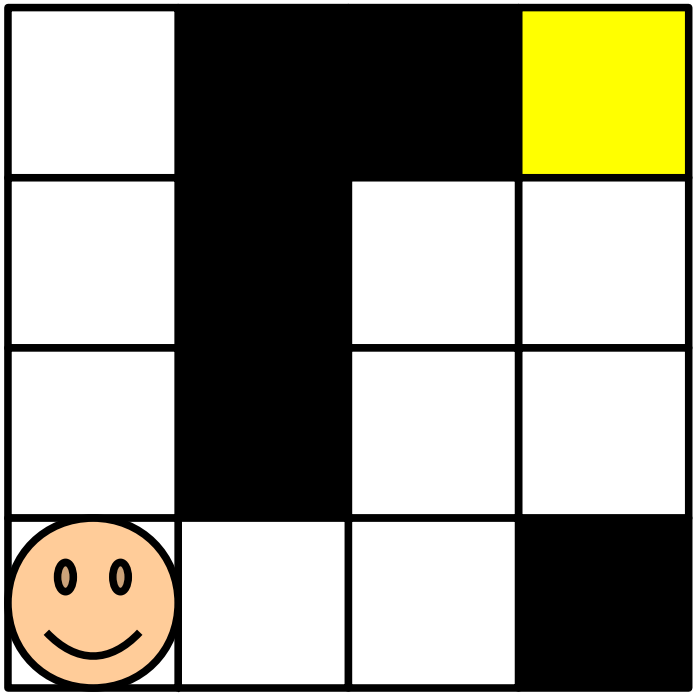
- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

- Perspectives

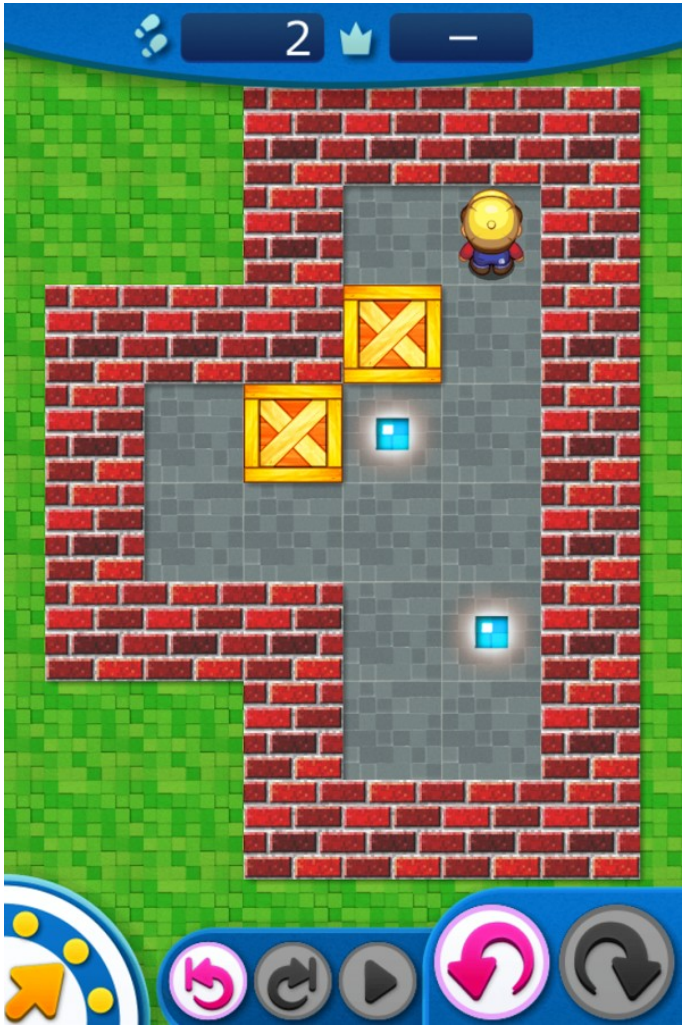
Exemple - Labyrinthe

- Labyrinthe
  - Déplacer
- Atteindre sortie
- Prise décision sequentielle



Exemple - Sokoban

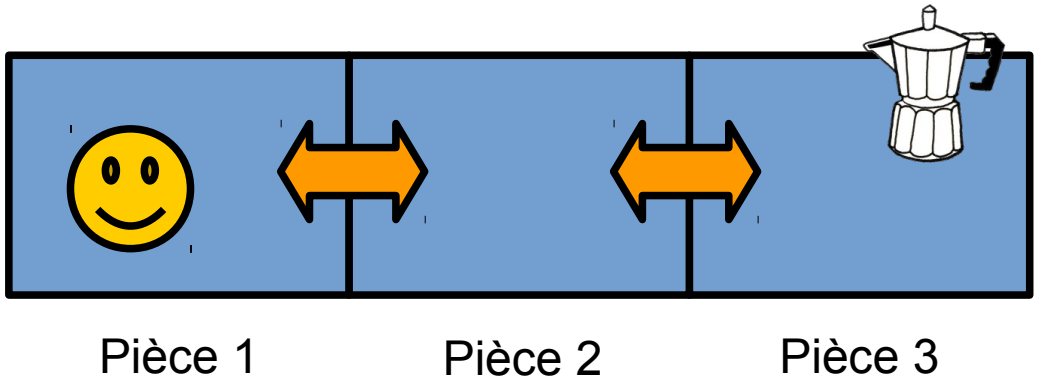
- Sokoban
  - Déplacer
  - Pousser caisses
- Amener objectif
- Prise de décision séquentielle



Sokoban touch

Exemple – Robot café

- Robot cafe
  - 3 salles
  - Robot se déplace
  - Café dans cuisine



- Objectif
  - Ramener un café

- Prise de décision sequentielle

Plan

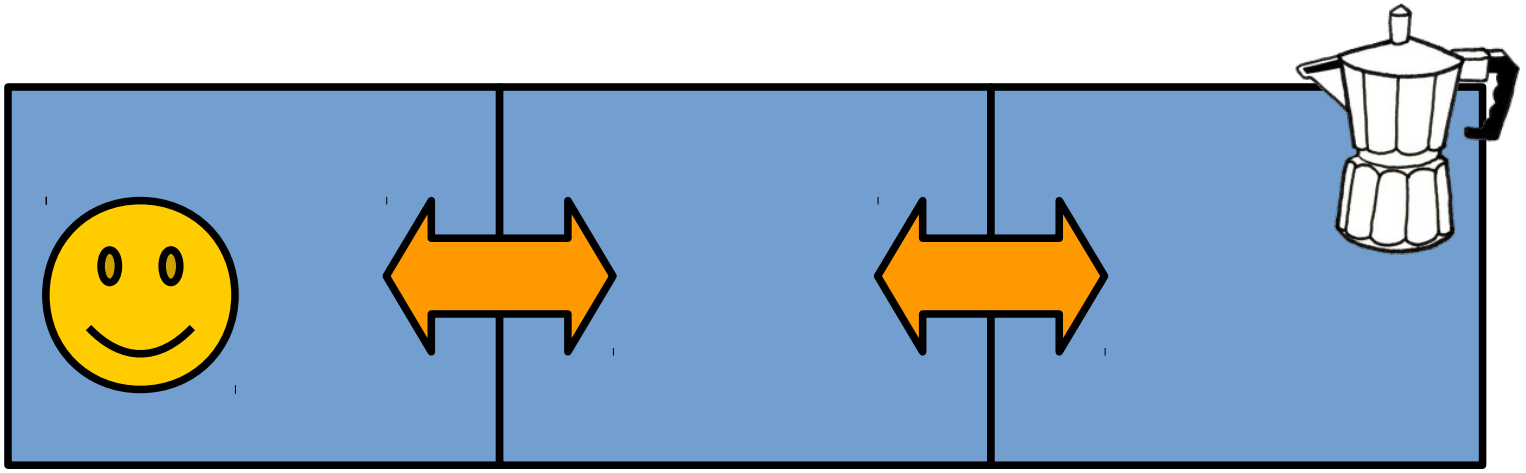
- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

- Perspectives

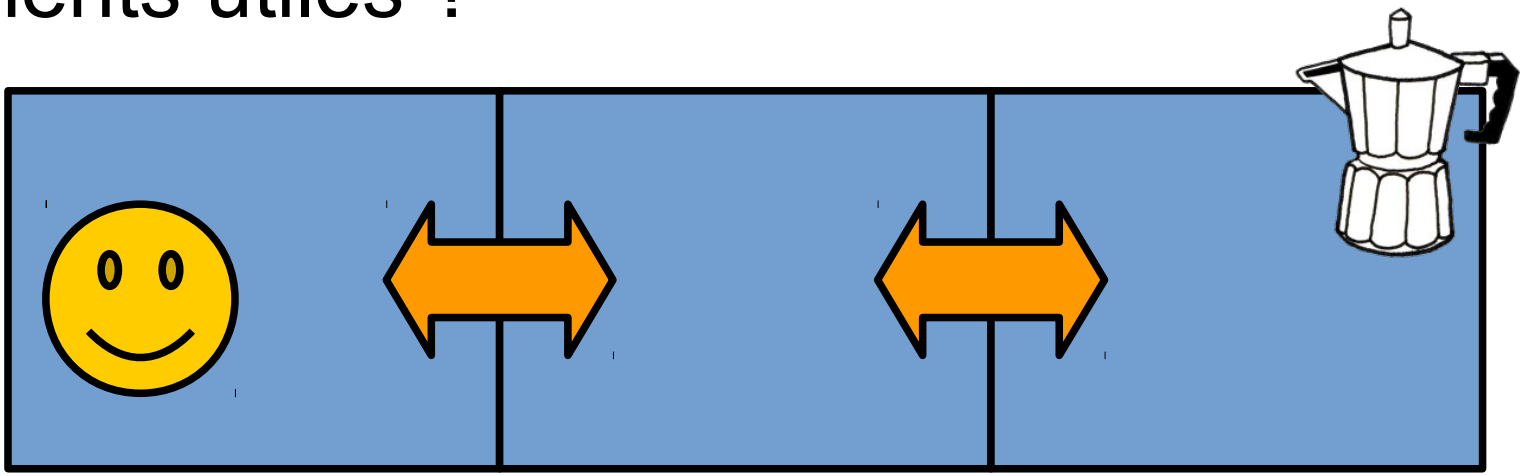
Exemple robot-Cafe

- 3 pièces qui se suivent
  - Cuisine au bout du couloir
  - Pièce représentée par indices
  - Ramener café pièce 1



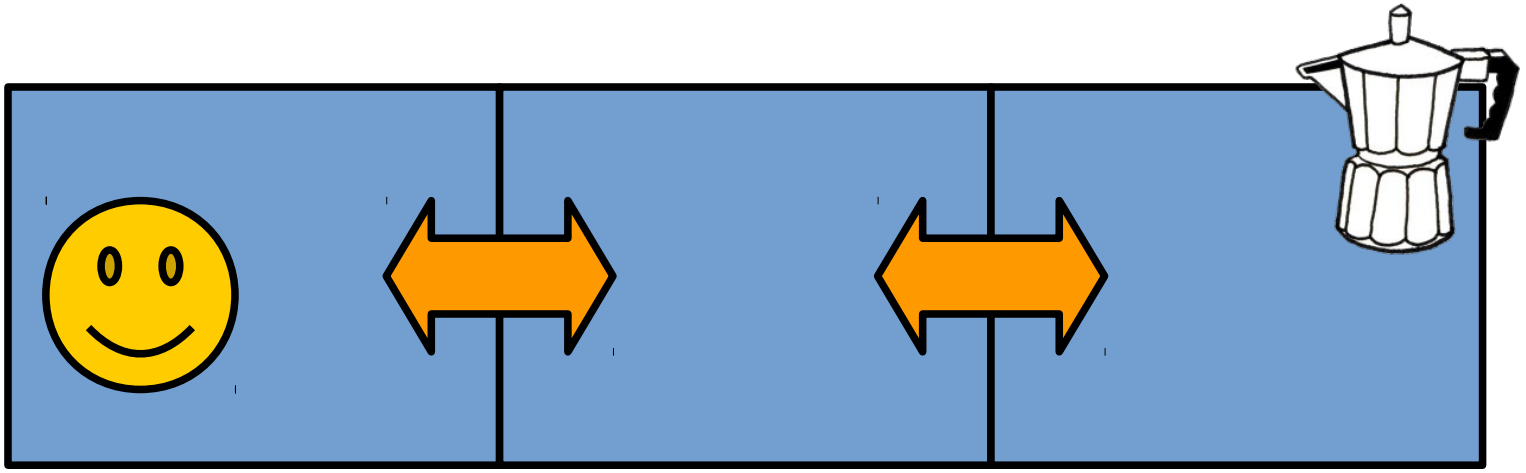
Exemple robot-Cafe

- Elements utiles ?



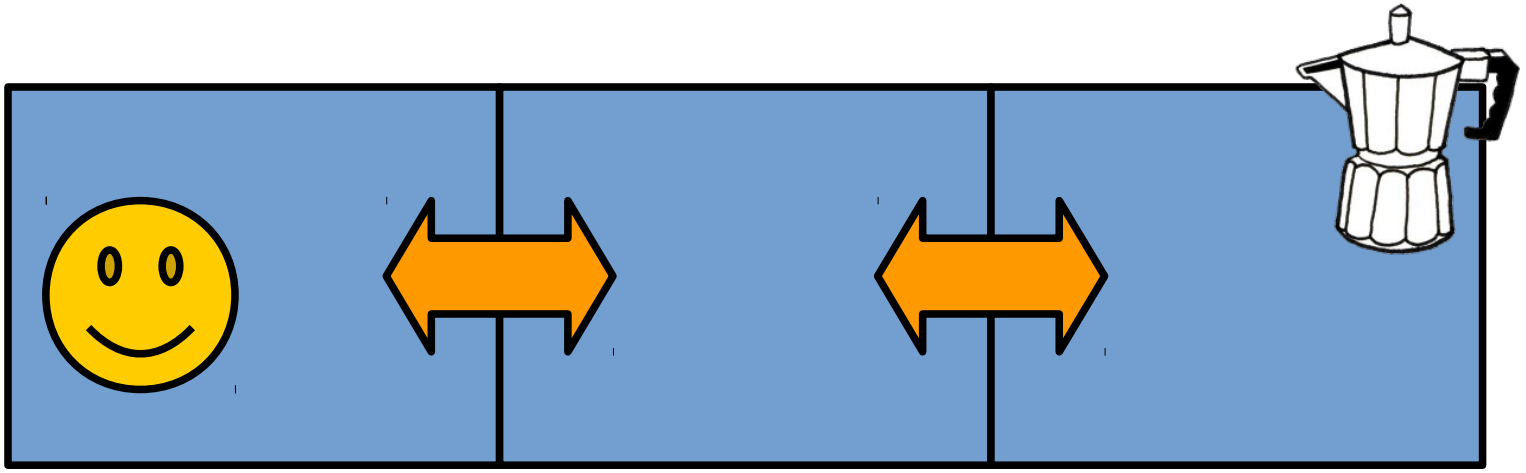
Exemple robot-Cafe

- Elements utiles ?
  - Actions
  - Etats
  - Dynamique
  - Objectif



Exemple robot-Cafe

- Elements utiles ?
  - Actions => gauche, droite, prendre, poser
  - Etats => (p1,0) (p2,0) (p3,0) (p1,1) (p2,1) (p3,1)
  - Dynamique => fonction T : S x A --> S
  - Objectif => fonction rec : S x A --> Reel



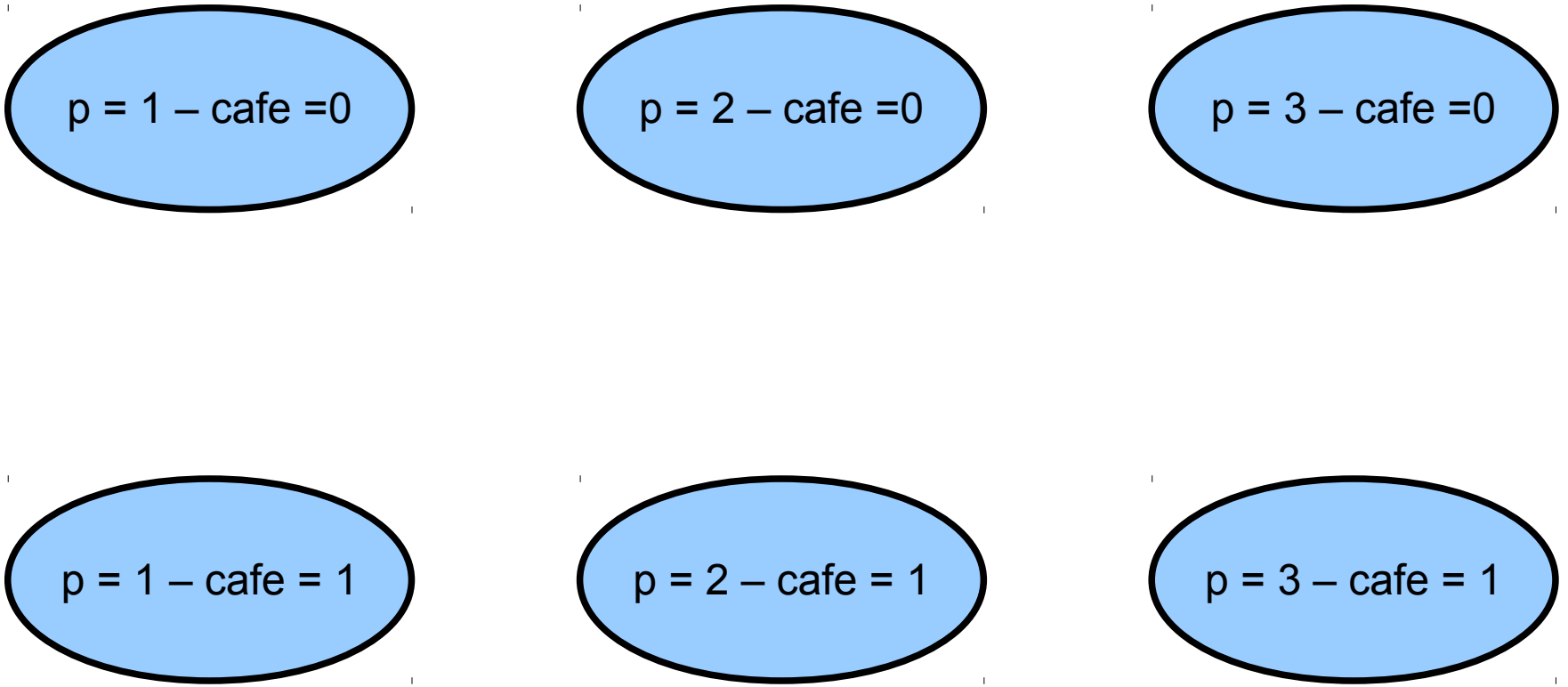
Problème générique

```
class Probleme:
    """permet de definir un probleme"""

    def actions(self):
        """returne la liste d'actions"""
        return ([])

    def etats(self):
        """returne la liste d'etats"""
        return ([])


    def transition(self,s,a):
        """definit les consequence d une action"""
        return(s)


    def recompense(self,s,a,sarr):
        """definit la recompense obtenue"""
        return(0)
```

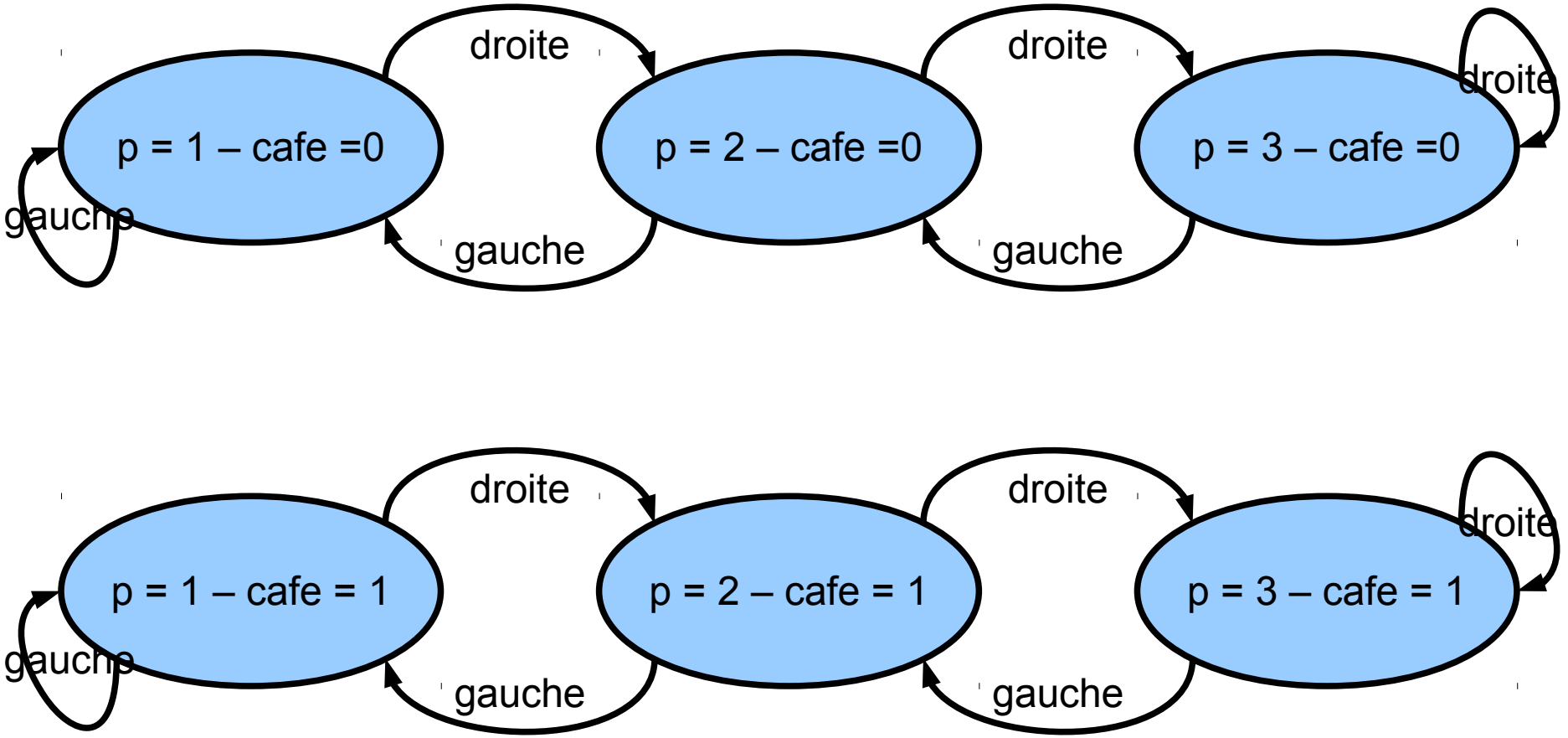
Problème RobotCafe

```
class Cafe:

    def actions(self):
        return(['gauche','droite','prendre','poser'])

    def etats(self):
        return([(1,0),(1,1),(2,0),(2,1),(3,0),(3,1)])

    def transition(self,s,a):
        pos=s[0];
        cafe=s[1];
        if (a=='gauche'):
            if (pos>1):
                pos=pos-1
        if (a=='droite'):
            if (pos<3):
                pos=pos+1
        if (a=='prendre'):
            if (pos==3):
                cafe=1
        if (a=='poser'):
            cafe=0
        return(pos,cafe)

    def recompense(self,s,a,sarr):
        if (s[0]==1) and (s[1]==1) and (a=='poser'):
            return(100)
        return(-1)
```

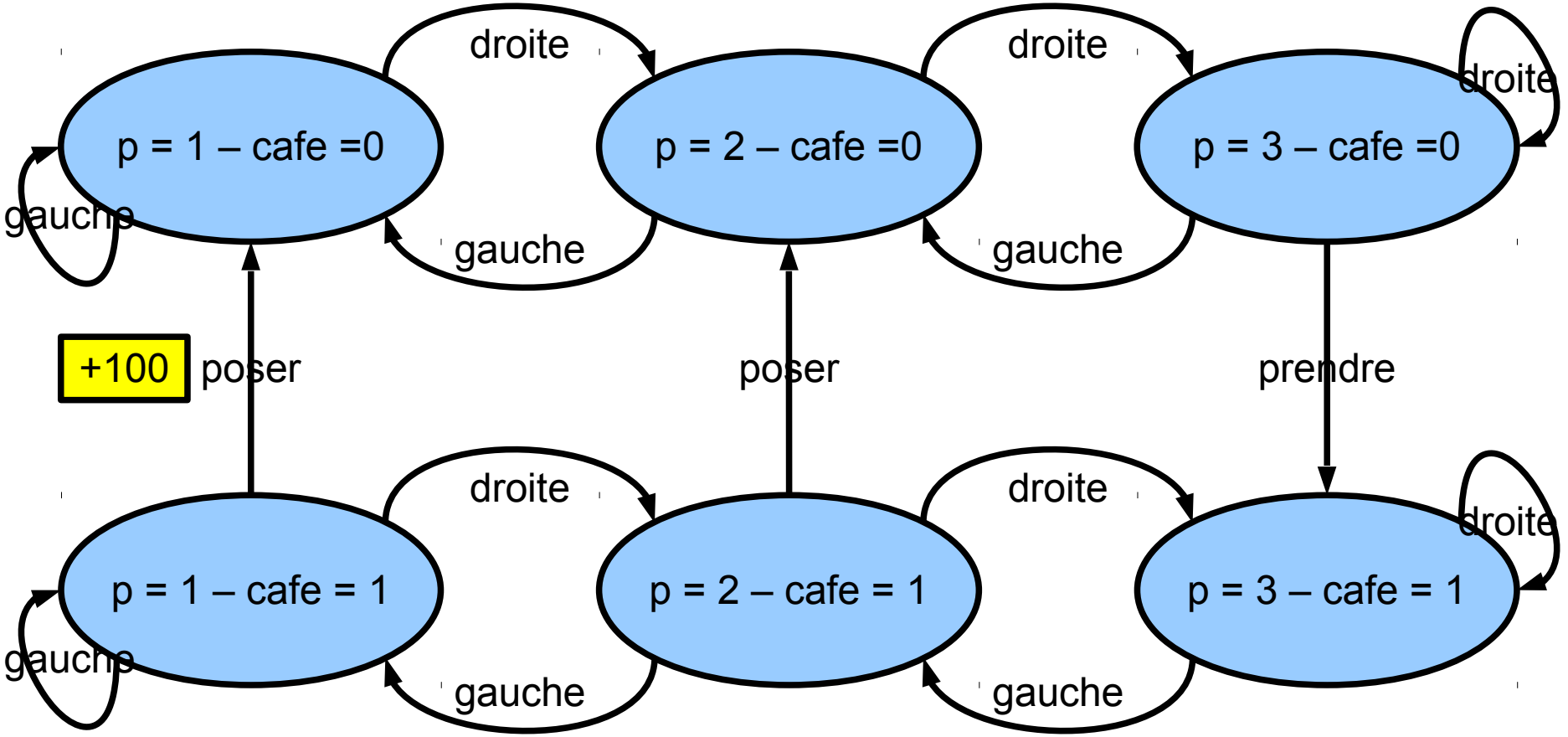
Représentation graphe



Représentation graphe



Représentation graphe

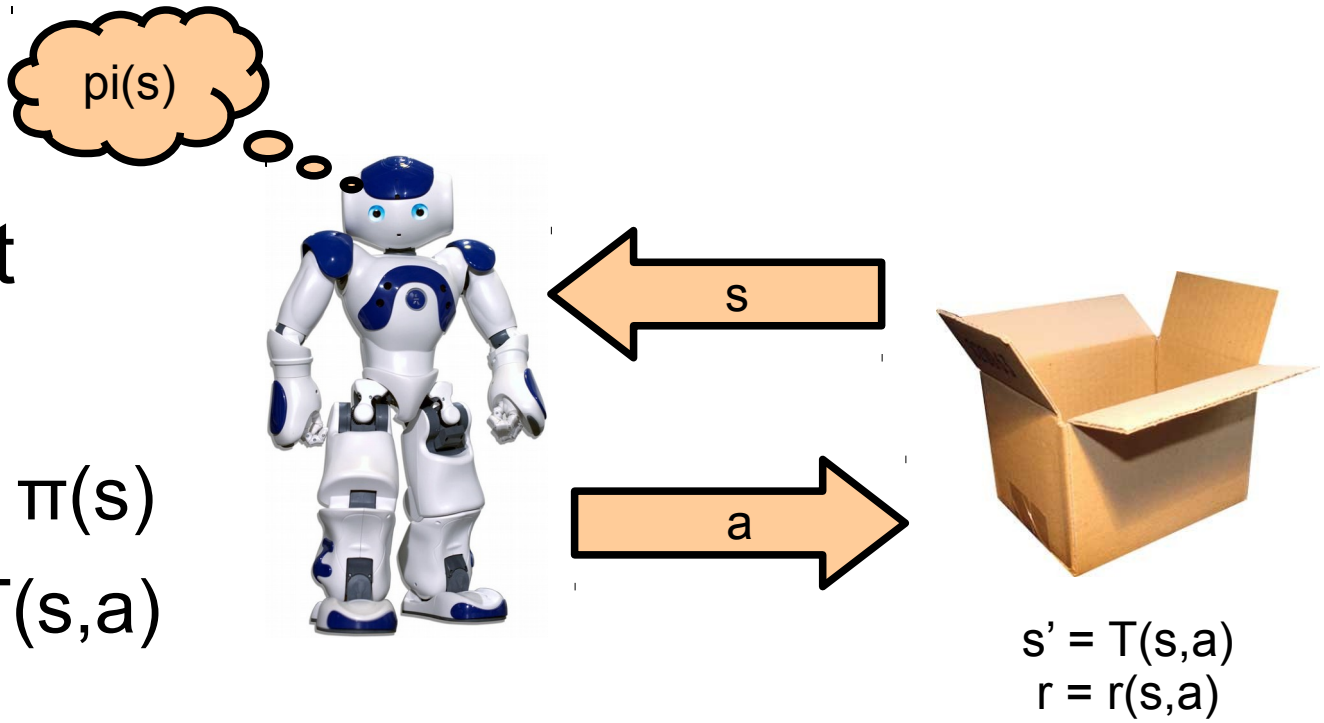


Comportement agent

- Fonction politique π : S -> A
  - Boucle fermée
  - Un exemple avec cafeRobot

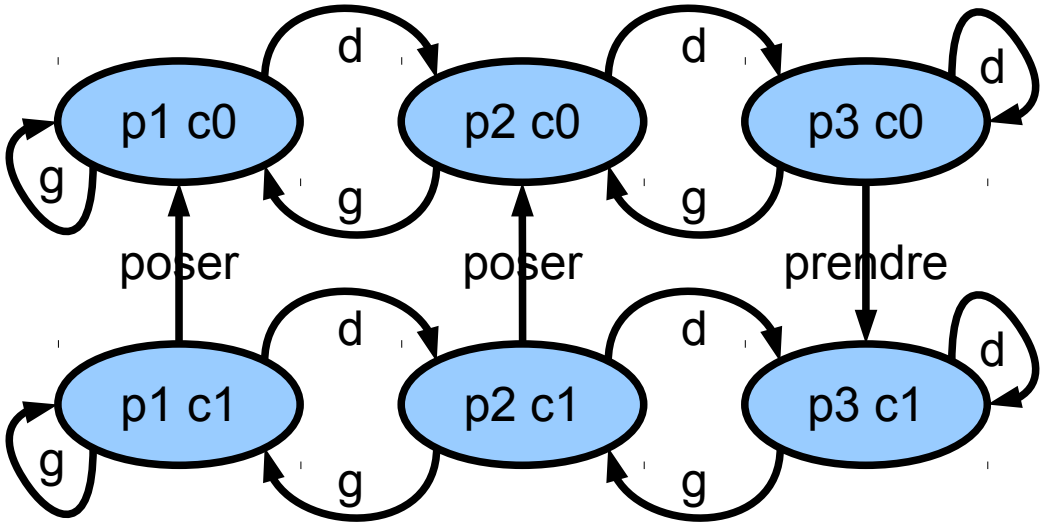
Etat ->	Action
P1, c0	« Droite »
P1, c1	« Gauche »
P2, c0	« Droite »
P2, c1	« Droite »
P3, c0	« Prendre »
P3, c1	«Gauche »

- Comportement
  - Dans s
  - Choix a selon π(s)
  - Etat s' selon T(s,a)



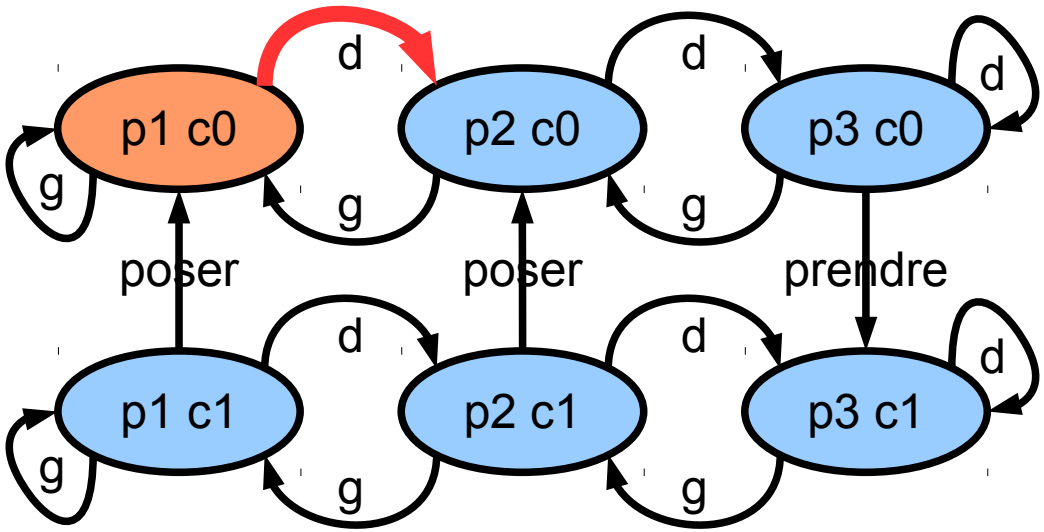
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



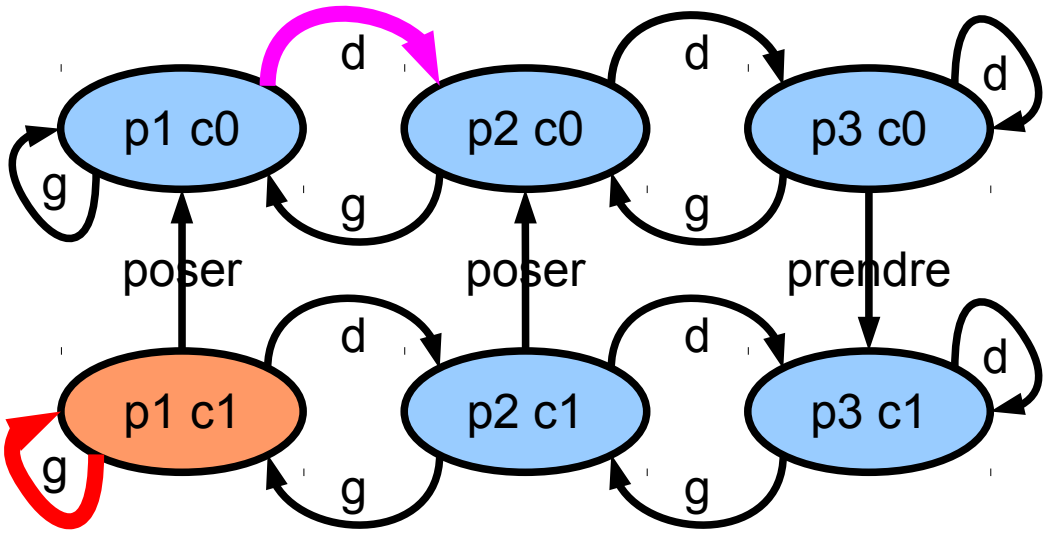
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



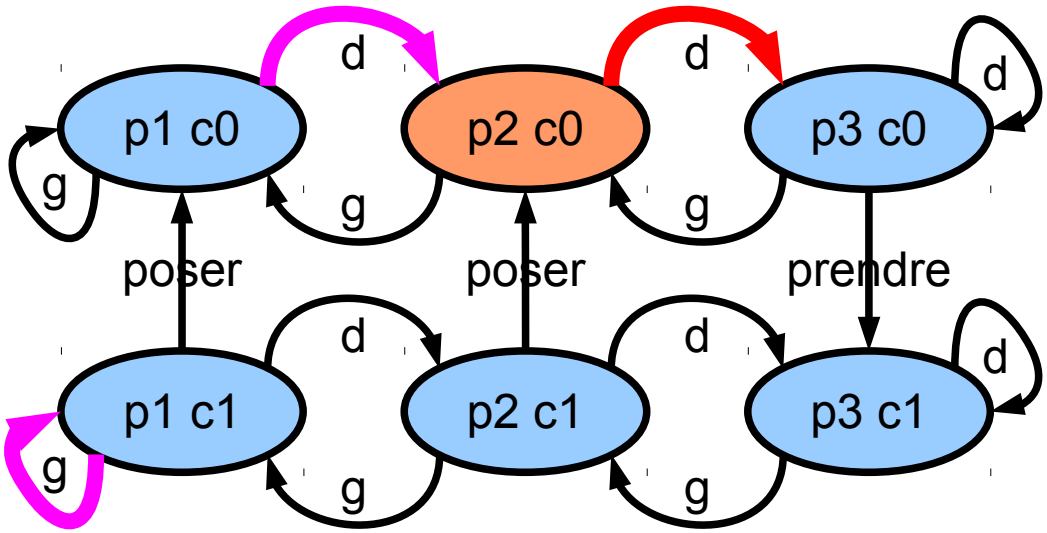
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



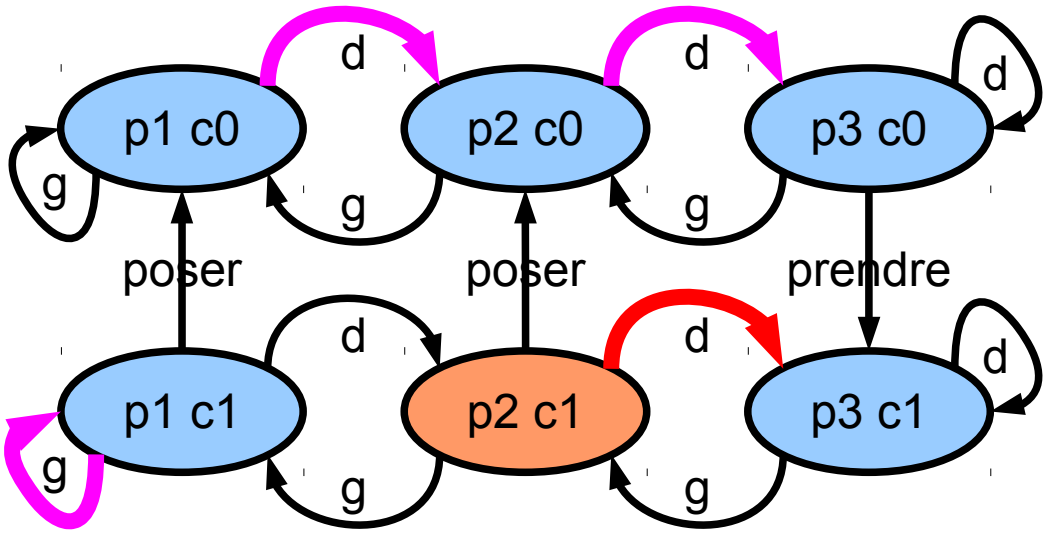
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



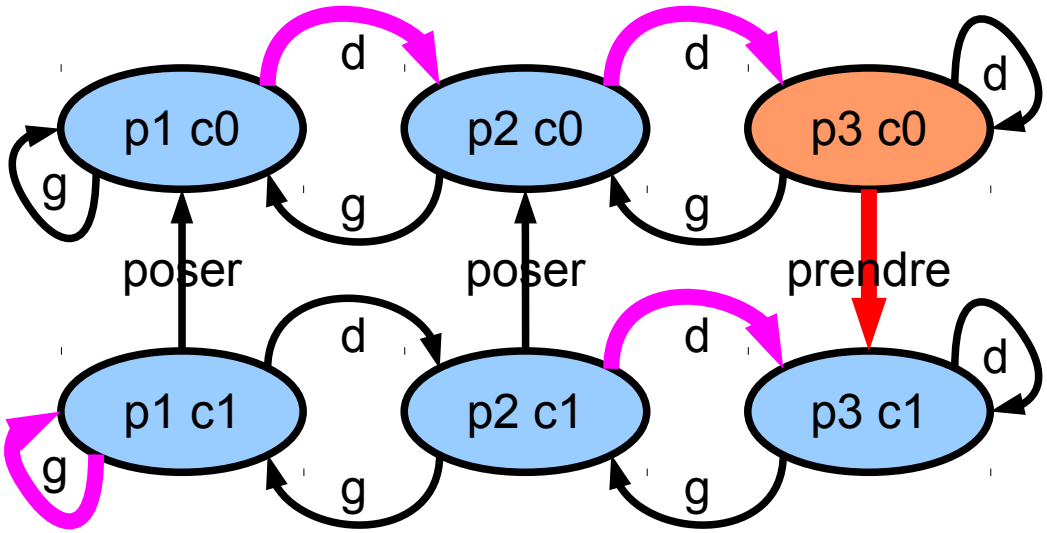
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



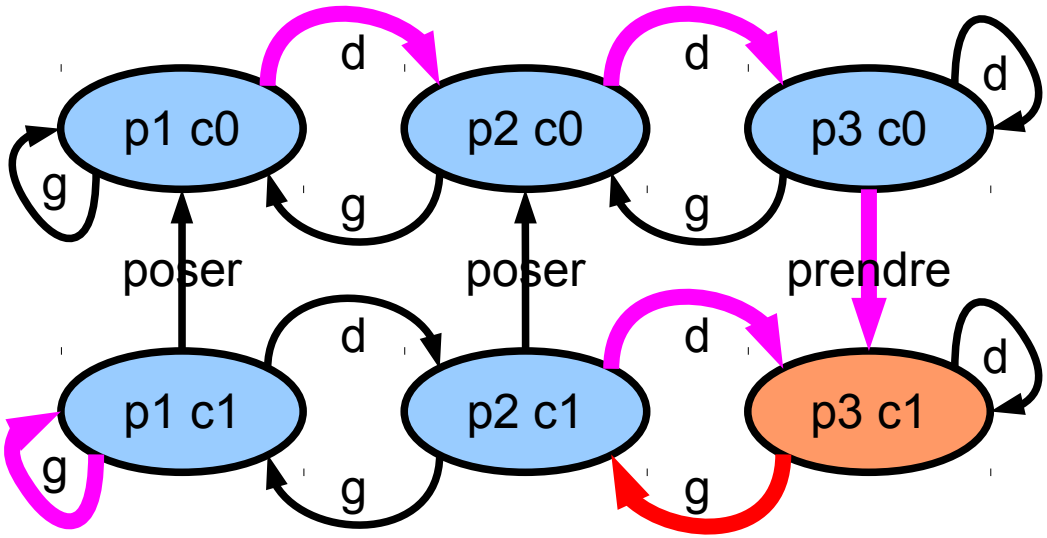
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



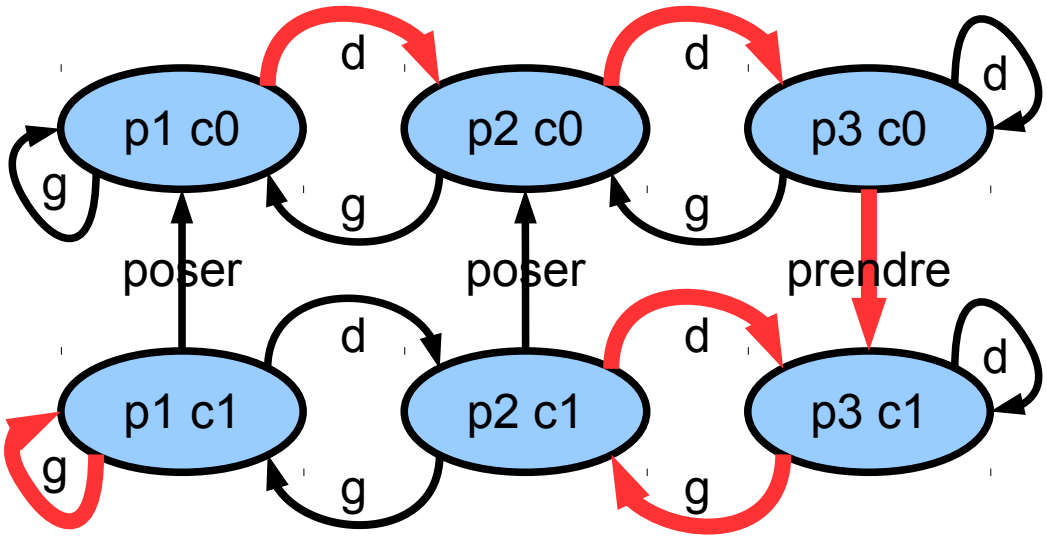
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```

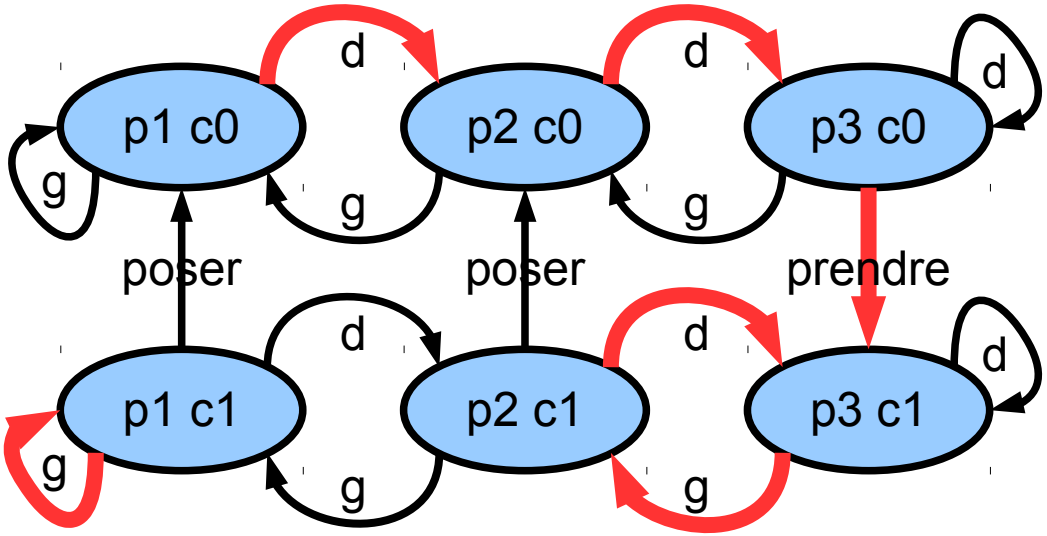


Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```

```
class SystemeExecute:

    def __init__(self,pb):
        self.pb=pb

    def executerPi(self,pi,depart,nb):
        s=depart
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            print(s," -> ",action," : ",sFin)
            s=sFin
```

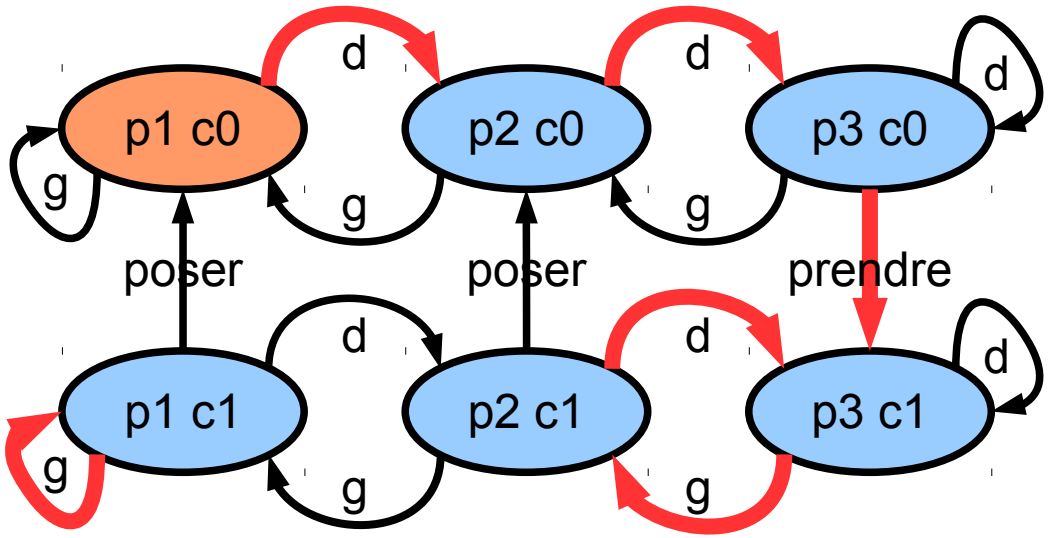
Code python execution agent




```
class SystemeExecute:

    def __init__(self,pb):
        self.pb=pb

    def executerPi(self,pi,depart,nb):
        s=depart
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            print(s," -> ",action," : ",sFin)
            s=sFin
```

Code python execution agent

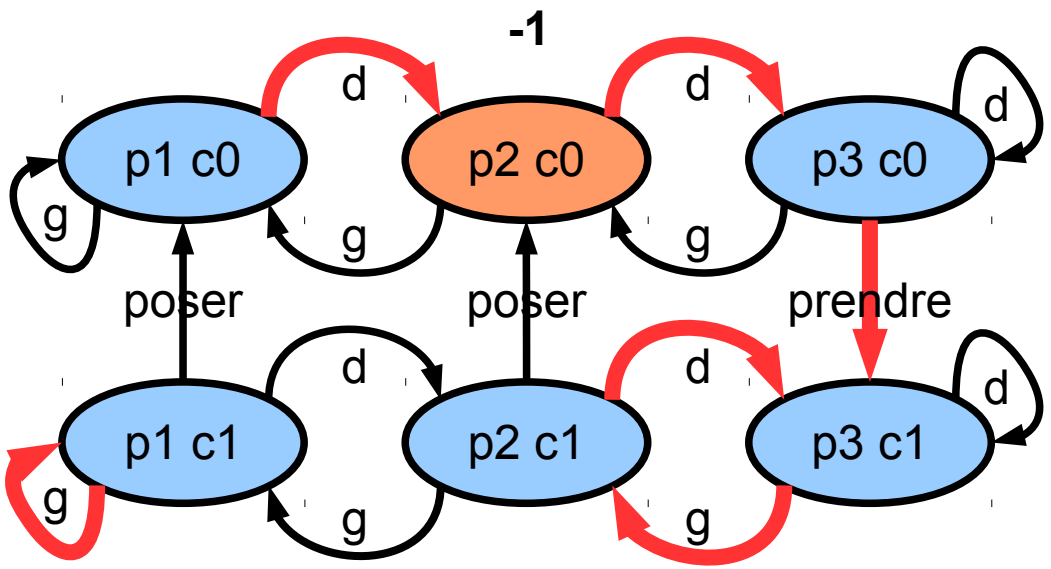
```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



Code python execution agent

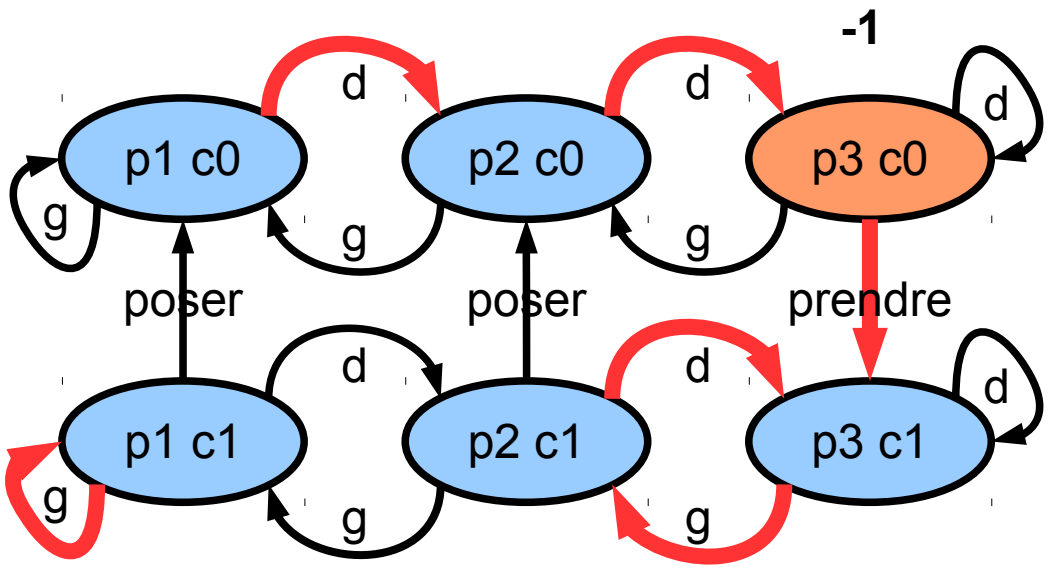
```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```

Récompense reçue



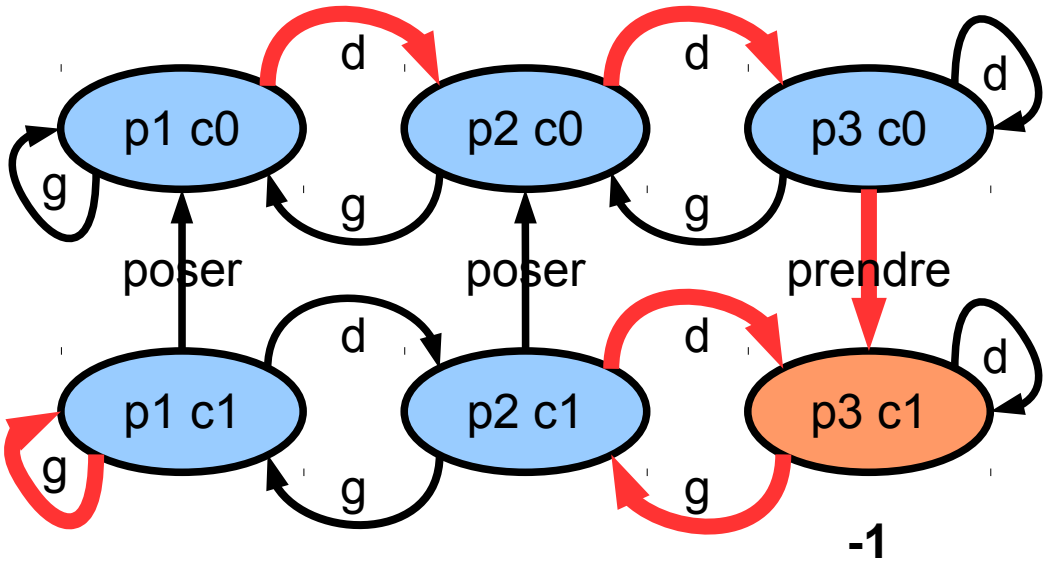
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



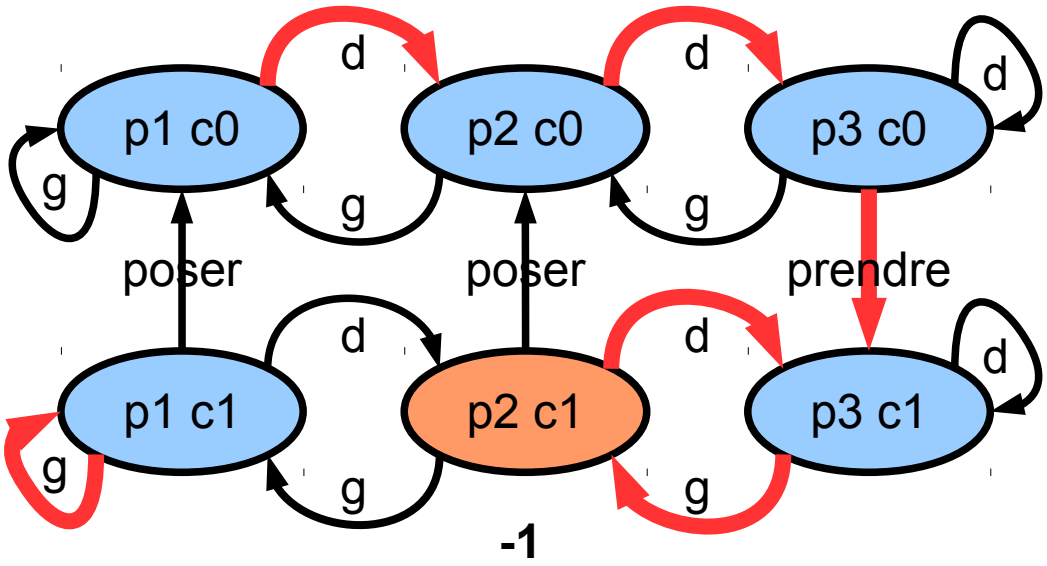
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



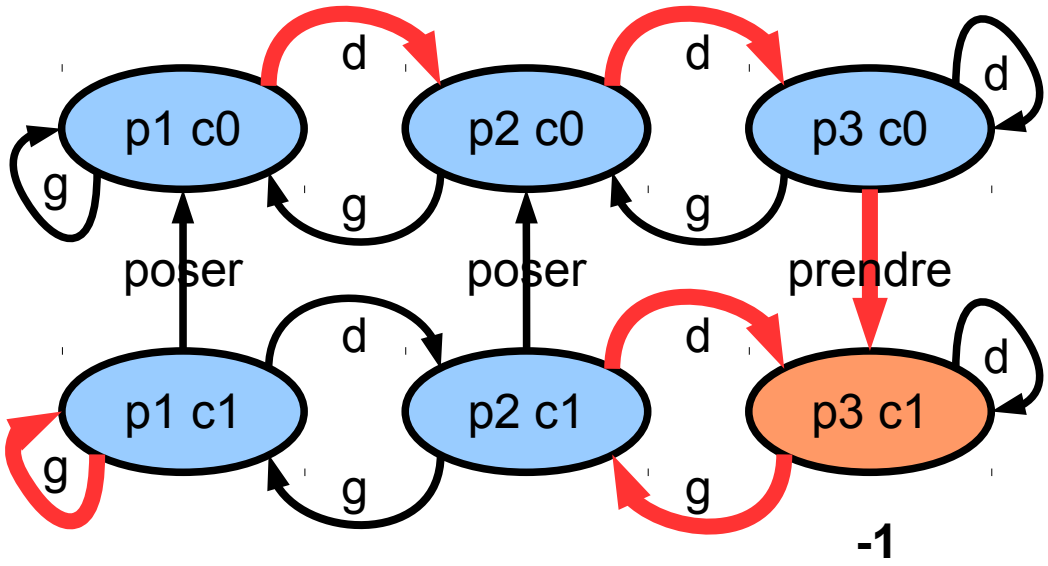
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



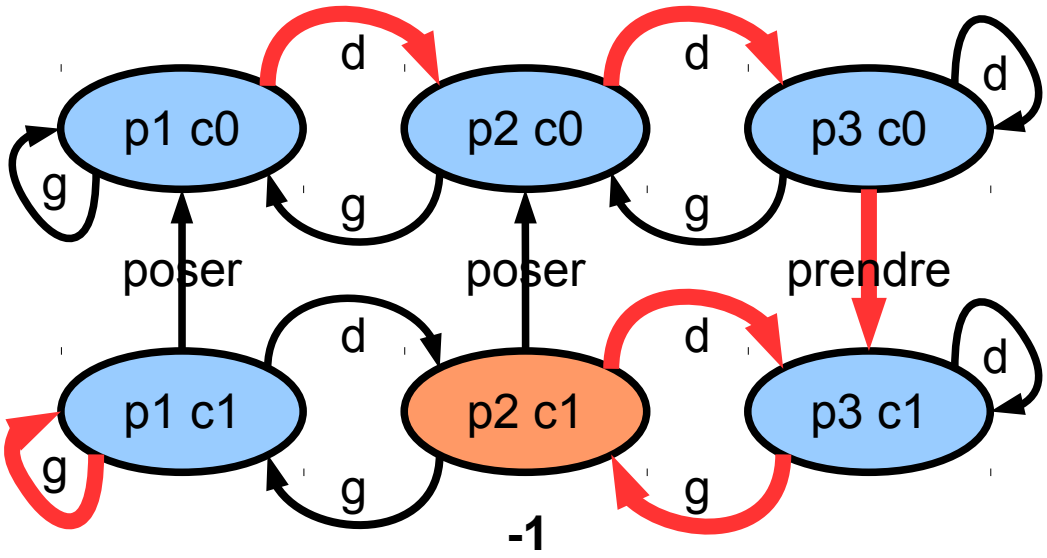
Code python execution agent

```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



Code python execution agent

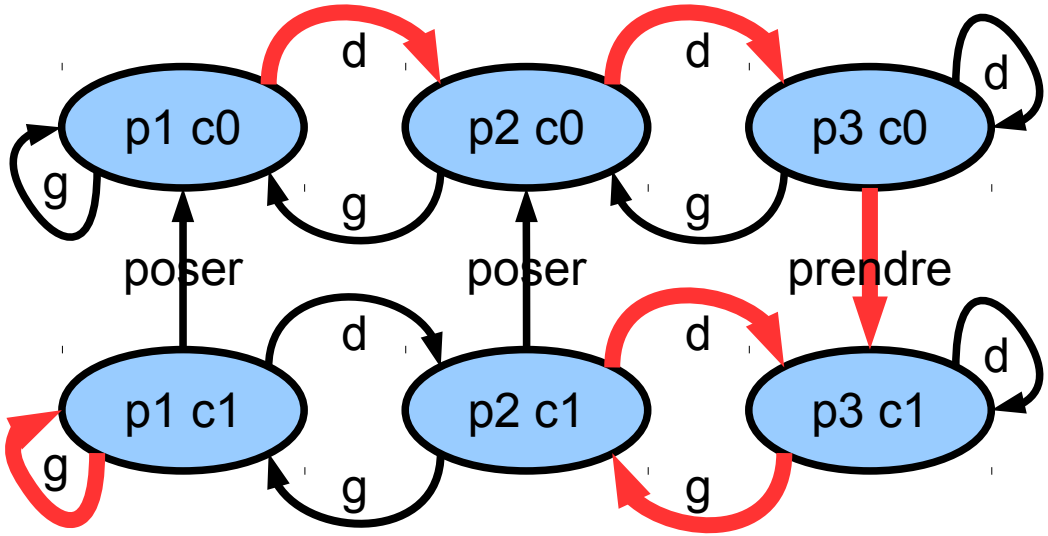
```
pb=Cafe()

pi={}
pi[(1,0)]='droite'
pi[(1,1)]='gauche'
pi[(2,0)]='droite'
pi[(2,1)]='droite'
pi[(3,0)]='prendre'
pi[(3,1)]='gauche'

print("*** test execution ***")
systemExec = SystemeExecute(pb)
systemExec.executerPi(pi,(1,0),10)
```



Code python execution politique

- Récupére la somme des récompenses

- Définition
  - π donné, performance $Perf(\pi, s_0) = \sum_{trajectoire} \gamma^t * r_t$



$$Perf(\pi,(1,0)) = -1 + (-1)*\gamma + (-1)*\gamma^2 + (-1)*\gamma^3 + \ldots$$

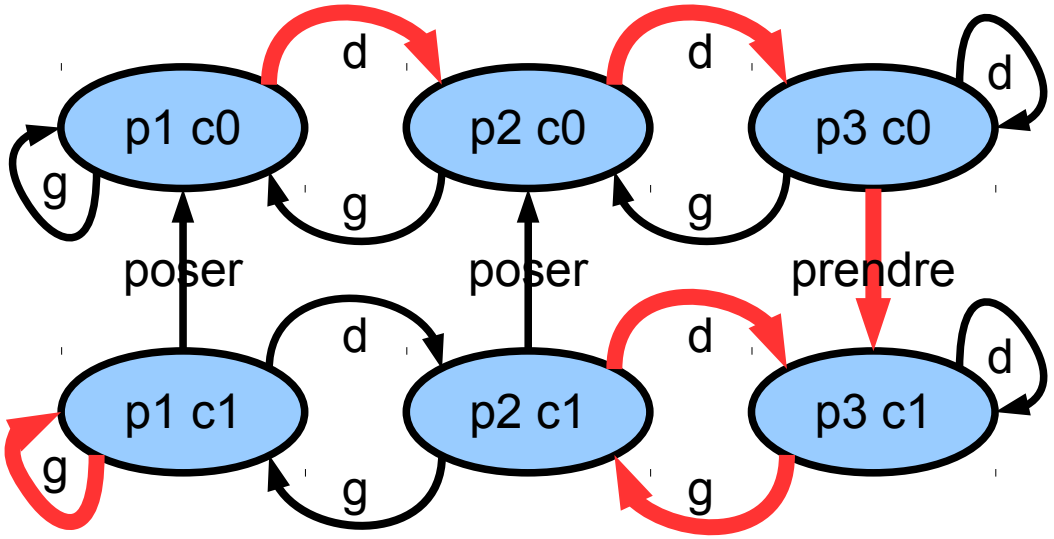
Code python execution politique

```
class SystemeExecute:
    def __init__(self,pb):
        self.pb=pb

    def executerPi(self,pi,depart,nb):
        s=depart
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            print(s," -> ",action," : ",sFin)
            s=sFin

    def executerPiRec(self,pi,depart,nb):
        s=depart
        gamma=0.99
        somme=0
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            r=pb.recompense(s,action,sFin)
            somme=somme + (gamma**i)*r
            print(s," -> ",action," : ",sFin,"<",r,">")
            s=sFin
        return somme
```

Code python execution politique

```
class SystemeExecute:
    def __init__(self,pb):
        self.pb=pb

    def executerPi(self,pi,depart,nb):
        s=depart
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            print(s," -> ",action," : ",sFin)
            s=sFin

    def executerPiRec(self,pi,depart,nb):
        s=depart
        gamma=0.99
        somme=0
        for i in range(nb):
            action=pi[s]
            sFin=pb.transition(s,action)
            r=pb.recompense(s,action,sFin)
            somme=somme + (gamma**i)*r
            print(s," -> ",action," : ",sFin,"<",r,">")
            s=sFin
        return somme
```

Resultat execution

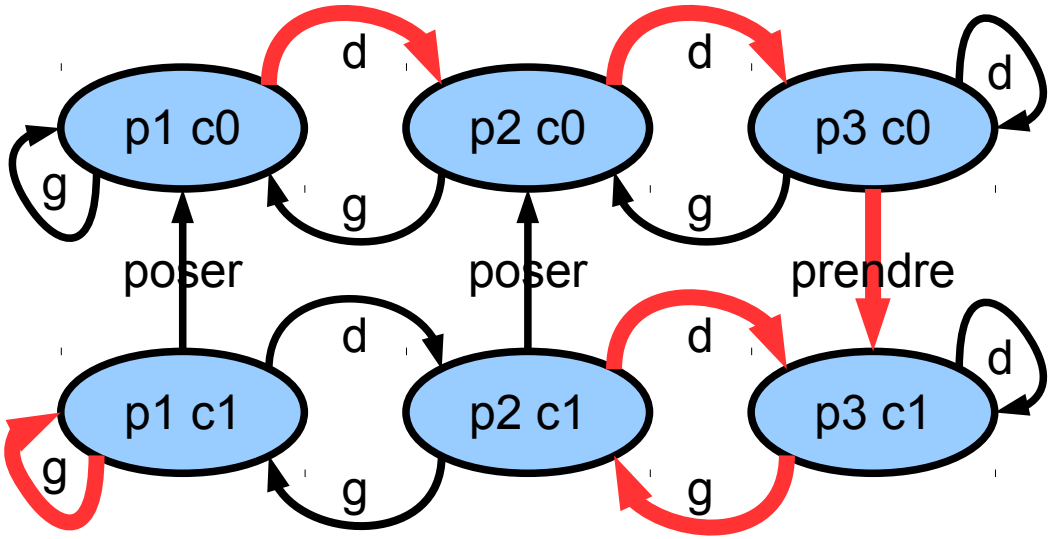


Etat ->	Action
P1, c0	« Droite »
P1, c1	« Gauche »
P2, c0	« Droite »
P2, c1	« Droite »
P3, c0	« Prendre »
P3, c1	«Gauche »

```
*** test execution recompense ***
(1, 0)   ->  droite    :  (2, 0) < -1 >
(2, 0)   ->  droite    :  (3, 0) < -1 >
(3, 0)   ->  prendre   :  (3, 1) < -1 >
(3, 1)   ->  gauche    :  (2, 1) < -1 >
(2, 1)   ->  droite    :  (3, 1) < -1 >
(3, 1)   ->  gauche    :  (2, 1) < -1 >
(2, 1)   ->  droite    :  (3, 1) < -1 >
(3, 1)   ->  gauche    :  (2, 1) < -1 >
(2, 1)   ->  droite    :  (3, 1) < -1 >
(3, 1)   ->  gauche    :  (2, 1) < -1 >
somme:   -9.561792499119552
```

Politique

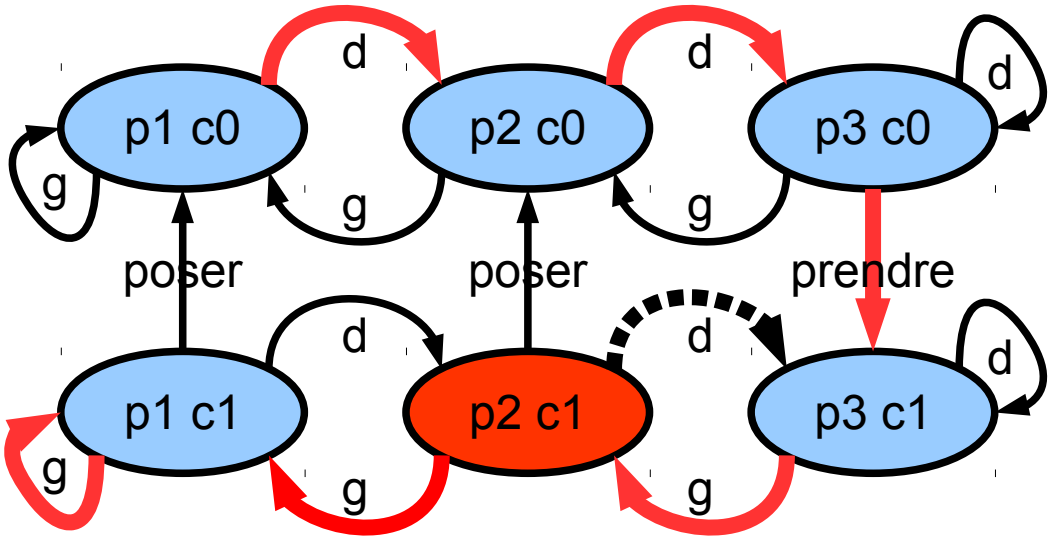
- Meilleure politique ?



Etat ->	Action
P1, c0	« Droite »
P1, c1	« Gauche »
P2, c0	« Droite »
P2, c1	« Droite »
P3, c0	« Prendre »
P3, c1	« Gauche »

Politique

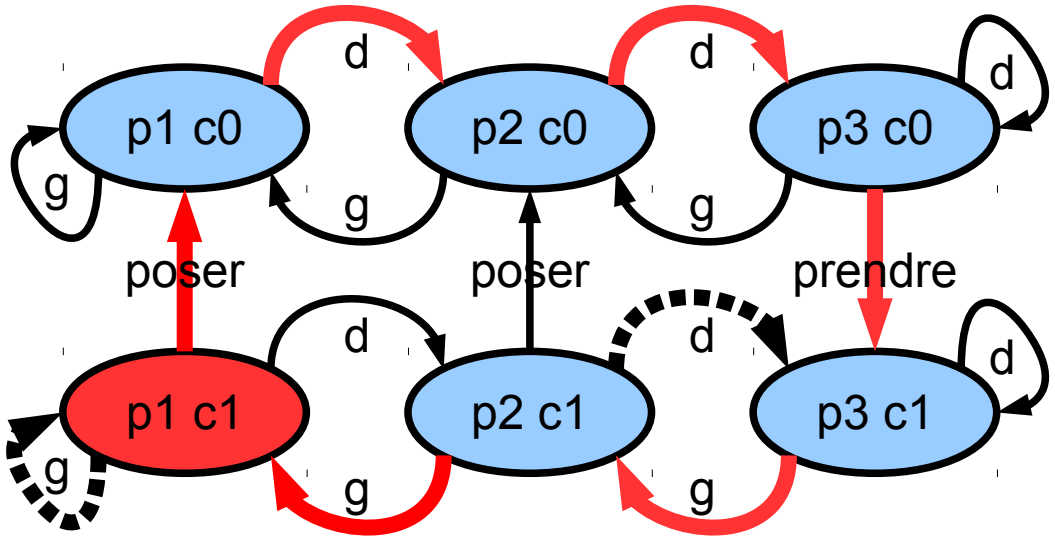
- Meilleure politique ?



Etat ->	Action
P1, c0	« Droite »
P1, c1	« Gauche »
P2, c0	« Droite »
P2, c1	« **Gauche** »
P3, c0	« Prendre »
P3, c1	« Gauche »

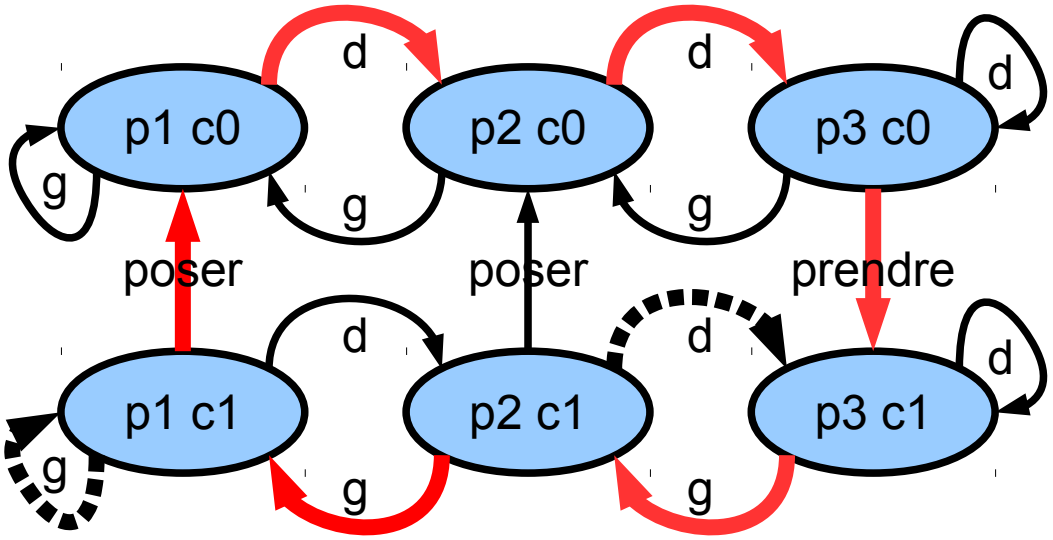
Politique

- Meilleure politique ?



Etat ->	Action
P1, c0	« Droite »
P1, c1	« **Poser** »
P2, c0	« Droite »
P2, c1	« Gauche »
P3, c0	« Prendre »
P3, c1	« Gauche »

Resultat execution

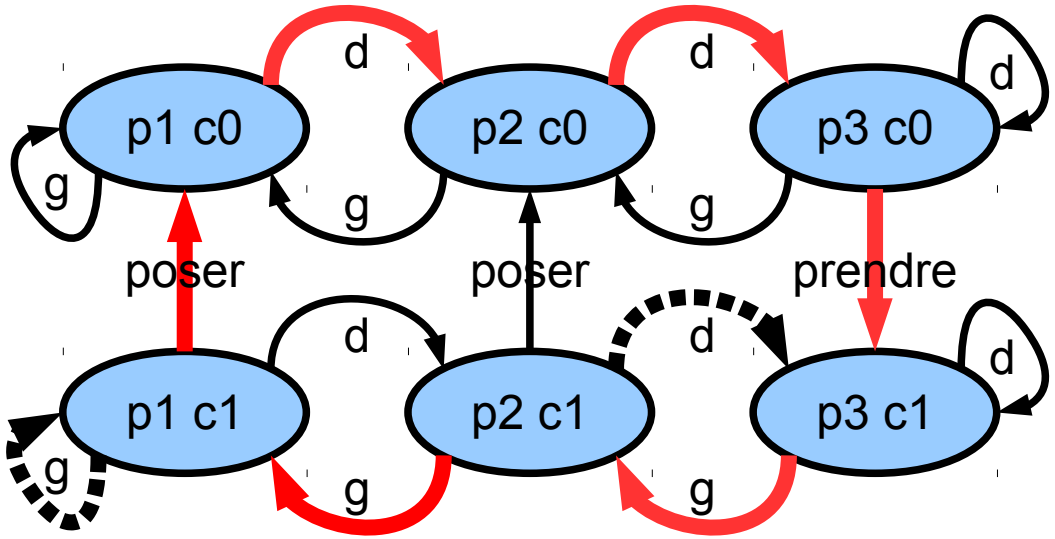


Etat ->	Action
P1, c0	« Droite »
P1, c1	« Poser »
P2, c0	« Droite »
P2, c1	« Gauche »
P3, c0	« Prendre »
P3, c1	« Gauche »

```
*** test execution recompense ***
(1, 0)   ->  droite    :   (2, 0) < -1 >
(2, 0)   ->  droite    :   (3, 0) < -1 >
(3, 0)   ->  prendre   :   (3, 1) < -1 >
(3, 1)   ->  gauche    :   (2, 1) < -1 >
(2, 1)   ->  gauche    :   (1, 1) < -1 >
(1, 1)   ->  poser     :   (1, 0) < 100 >
(1, 0)   ->  droite    :   (2, 0) < -1 >
(2, 0)   ->  droite    :   (3, 0) < -1 >
(3, 0)   ->  prendre   :   (3, 1) < -1 >
(3, 1)   ->  gauche    :   (2, 1) < -1 >
somme:   86.48820254078045
```

Politique

- Meilleure politique ?



Etat ->	Action
P1, c0	« Droite »
P1, c1	« **Poser** »
P2, c0	« Droite »
P2, c1	« Gauche »
P3, c0	« Prendre »
P3, c1	« Gauche »

Notre objectif : Algorithme pour construire automatiquement

Idee forte – représenter pb

- Problème (Markov Decision Process déterministe)
  - Ensemble d'états S
  - Ensemble d'actions A
  - Fonction transition T : S x A -> S
  - Fonction récompense r : S x A -> Reel

- Cherche π : S -> A
  - Maximise somme $Perf(\pi, s_0) = \sum_{trajectoire} \gamma^t * r_t$

- Représente de trés nombreux problèmes

Idee forte – représenter pb

- Problème (Markov Decision Process déterministe)
  - Ensemble d'états S
  - Ensemble d'actions A
  - Fonction transition T : S x A -> S
  - Fonction récompense r : S x A -> Reel

Idée 1

- Cherche π : S -> A
  - Maximise somme $Perf(\pi, s_0) = \sum_{trajectoire} \gamma^t * r_t$

- R

Question BONUS

Combien de politiques différentes ???

Plan

- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

- Perspectives

Equation de bellman

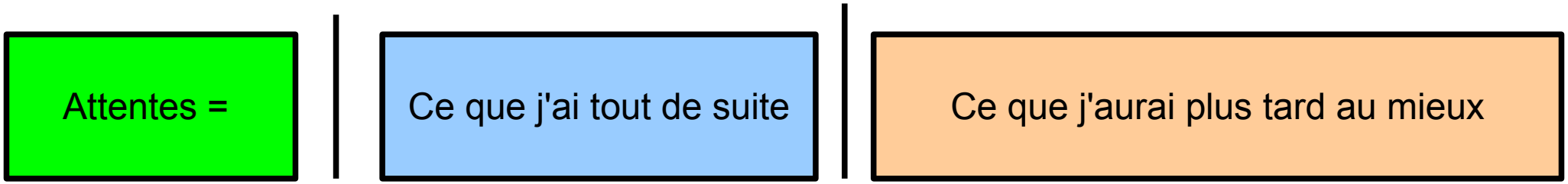
- Principe d'optimalité
  - Propriété que vérifie la politique
  - Fonction de valeur optimale Q*(s,a)
    - Ce que je peux attendre en partant de s en faisant a
    - Puis en suivant la meilleure politique

- Equation optimalité de bellman

Equation de bellman

- Principe d'optimalité
  - Propriété que vérifie la politique
  - Fonction de valeur optimale Q*(s,a)
    - Ce que je peux attendre en partant de s en faisant a
    - Puis en suivant la meilleure politique

- Equation optimalité de bellman



Equation de bellman

- Principe d'optimalité
  - Propriété que vérifie la politique
  - Fonction de valeur optimale Q*(s,a)
    - Ce que je peux attendre en partant de s en faisant a
    - Puis en suivant la meilleure politique

- Equation optimalité de bellman

$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma \max_{a'} Q^*(T(s,a),a')$$

Attentes = | Ce que j'ai tout de suite | Ce que j'aurai plus tard au mieux

Facteur actuation

- Facteur actuation gamma < 1
  - Ratio aujourd'hui / demain

- Deuxieme Idée forte
  - Equation optimalité Bellman

$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Facteur actuation

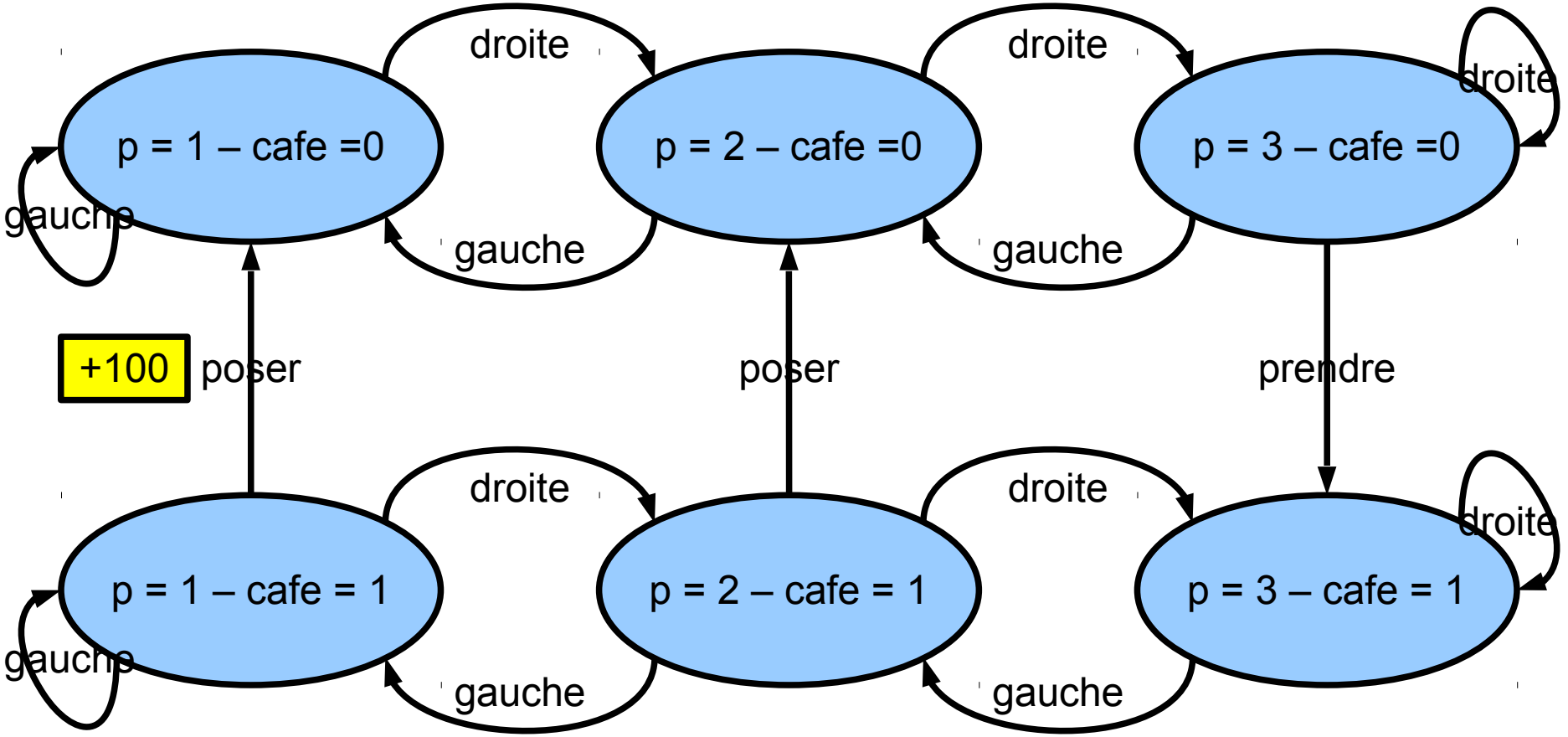
- Facteur actuation gamma < 1
  - Ratio aujourd'hui / demain

- Deuxieme Idée forte
  - Equation optimalité Bellman

$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Dérouler pour quelques couples (s,a)

Représentation graphe



Plan

- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

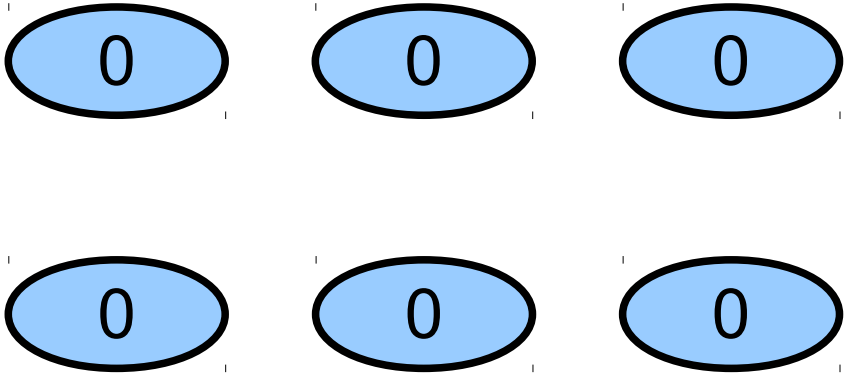
- Perspectives

Résolution

- 1) Résolution simple
  - Énumérer toutes les politiques
  - Mais beaucoup politiques => S^A
- 2) Utiliser équation de Bellman
  - Calcul récursif
  - Mais Recalculer plein de choses (cf Markov)
  - Et Plein de boucles
- 3) Utiliser la programmation dynamique

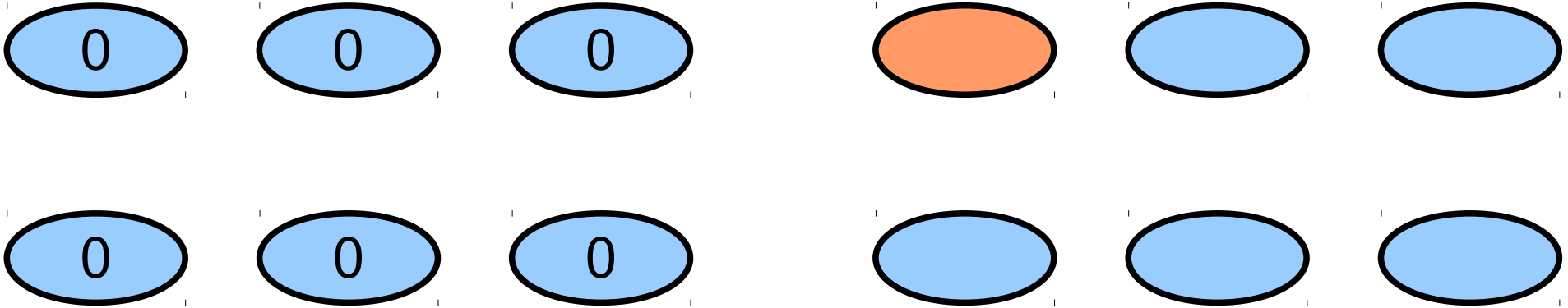
Principe algorithme

- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



Principe algorithme

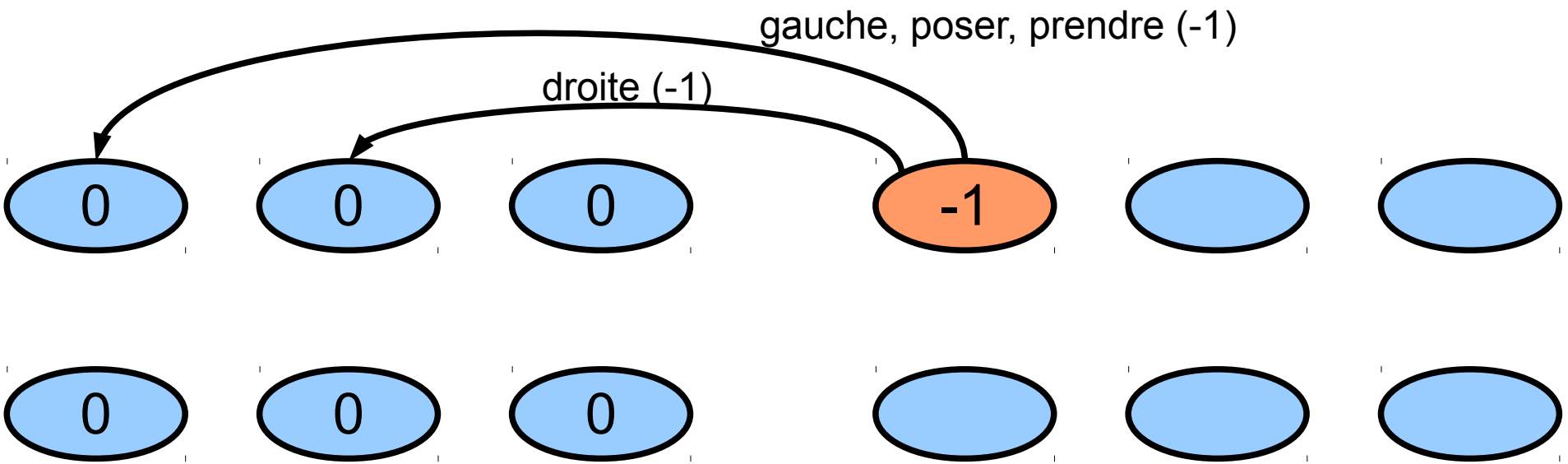
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

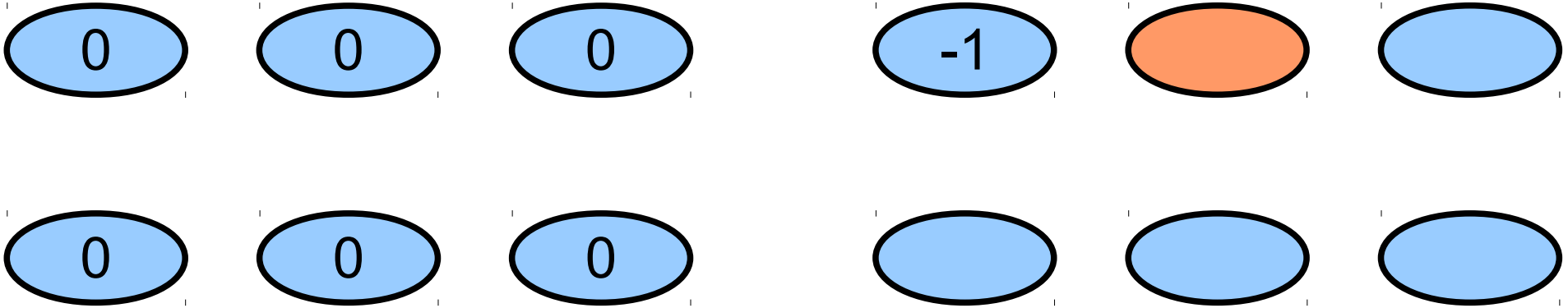
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

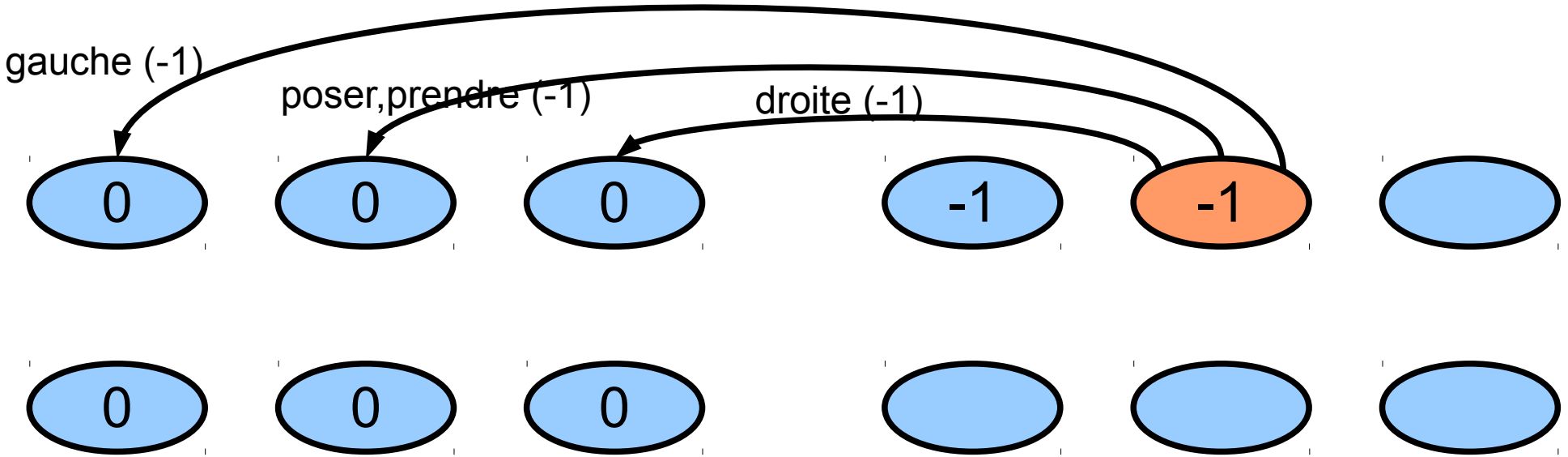
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

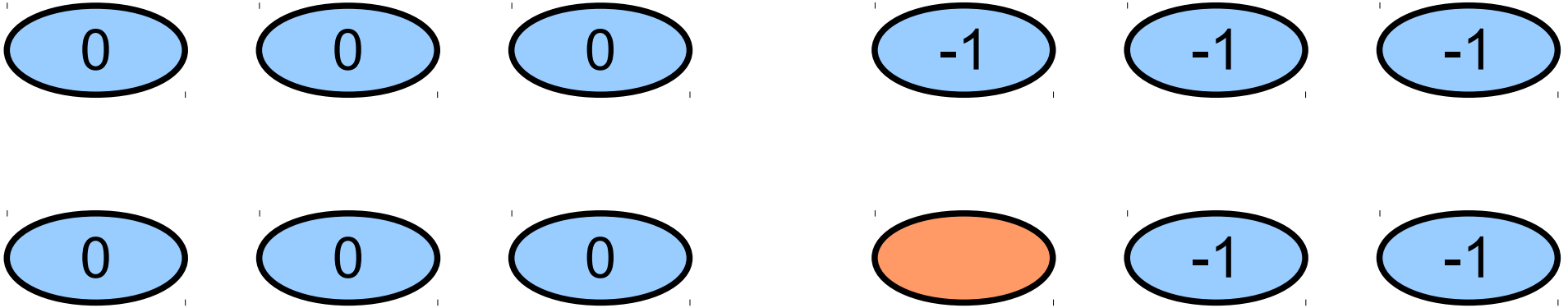
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

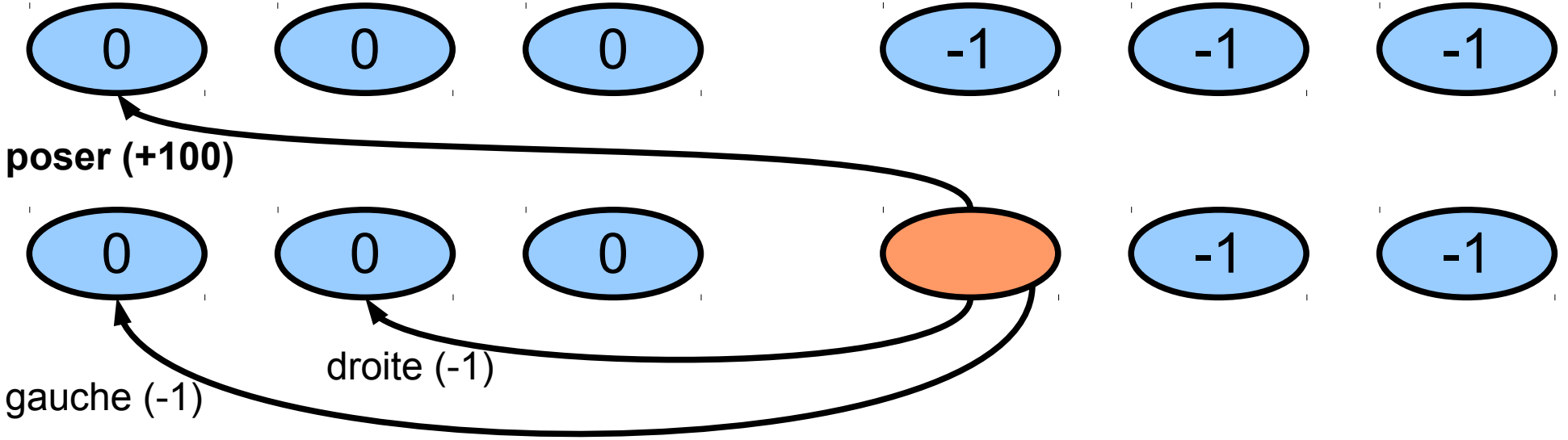
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

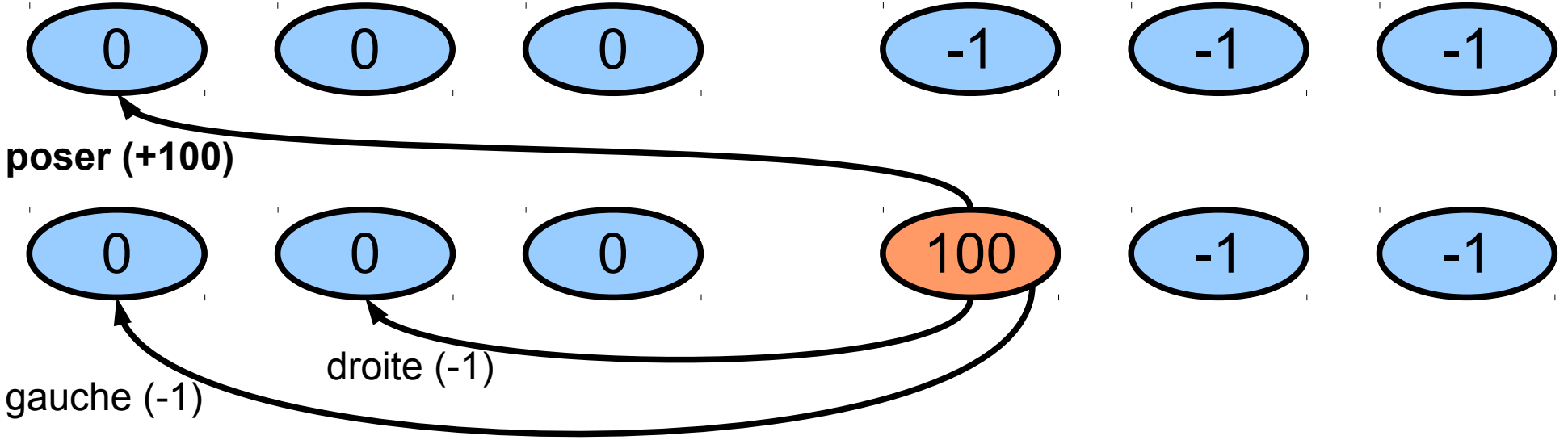
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

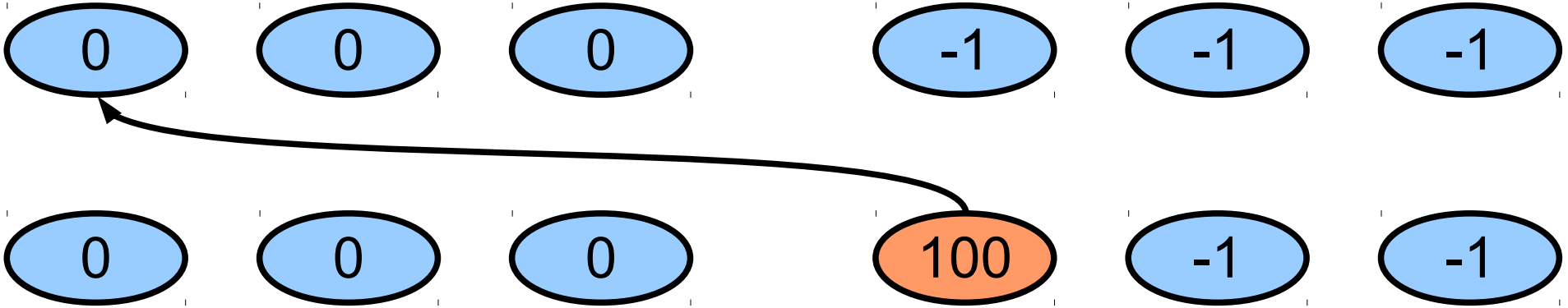
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

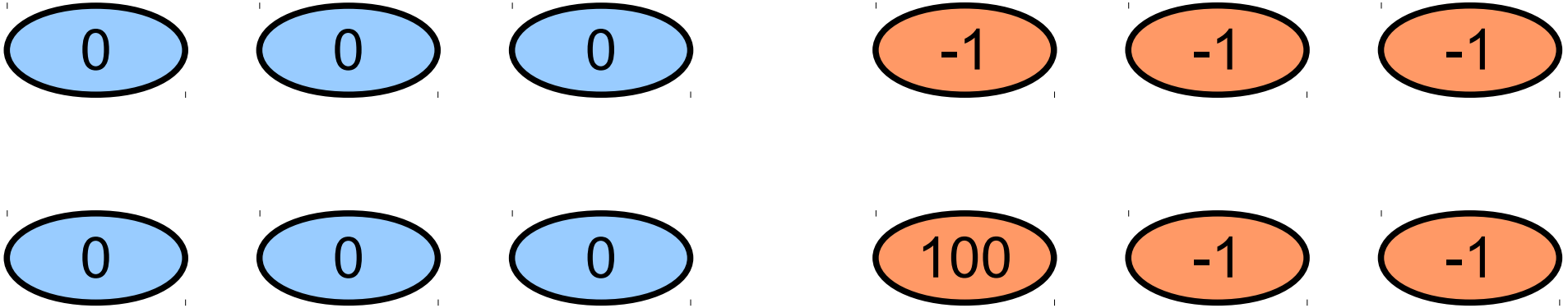
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

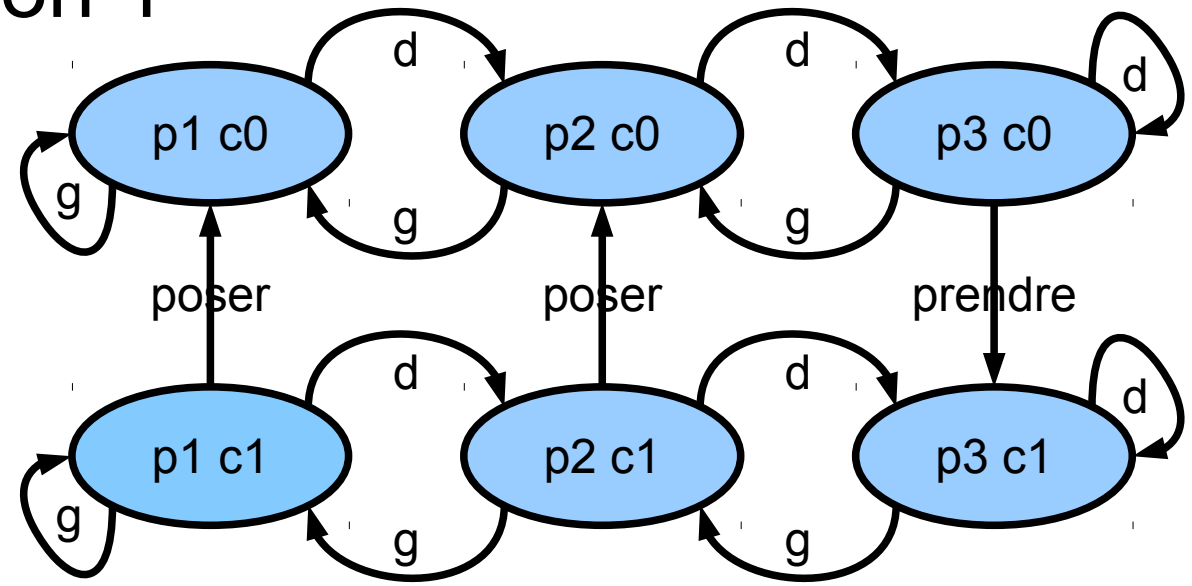
- Part de la fin
  - Action dispo = 0
  - Performance = 0
- Remonte temps
  - Action dispo = 1



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Exemple RobotCafe

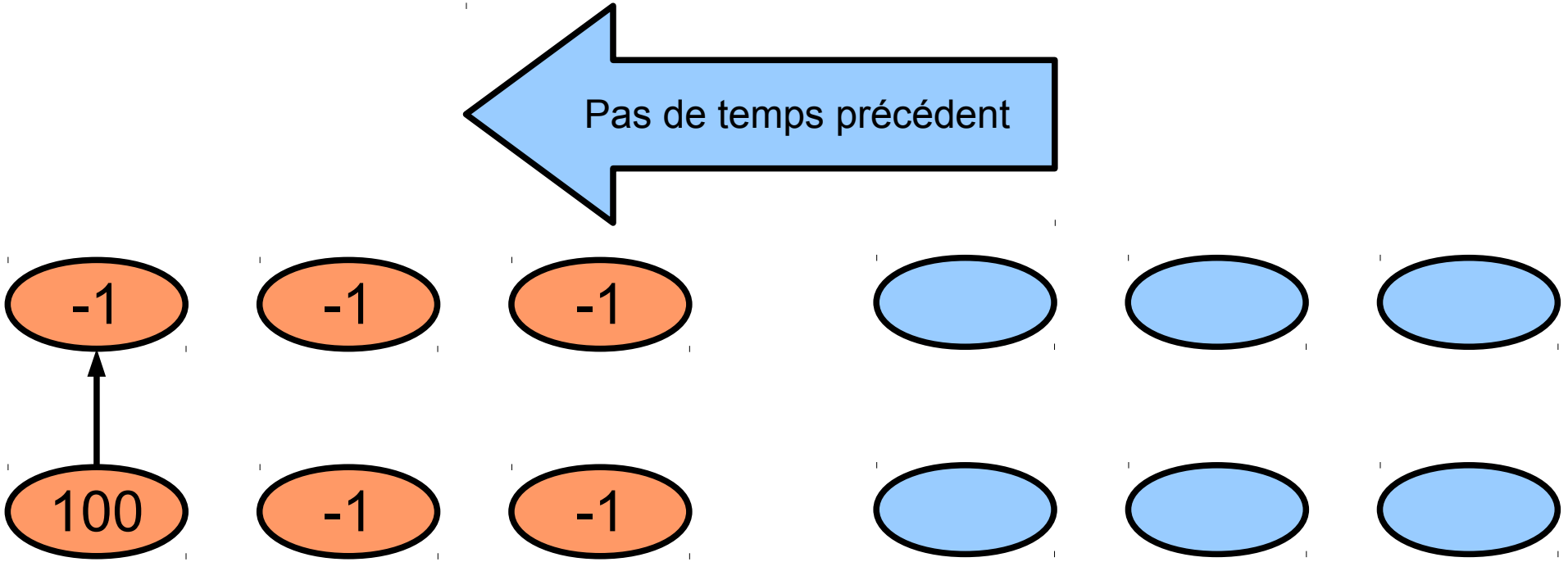
- Iteration 1




```
(1, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(1, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> 100.0

(2, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(2, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(3, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(3, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0
```

Principe algorithme

- Part de la fin
  - Action dispo = 1
- Remonte temps
  - Action dispo = 2

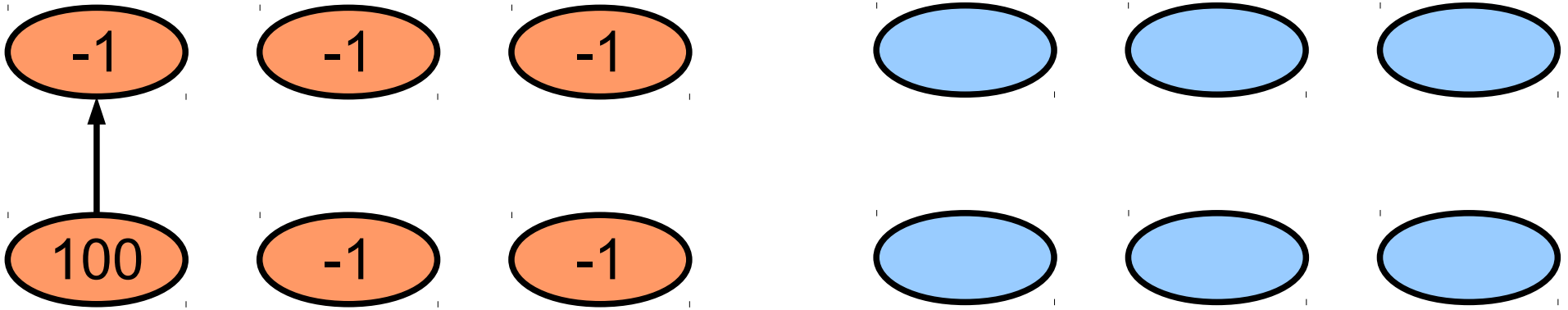


$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

- Part de la fin
  - Action dispo = 1
- Remonte temps
  - Action dispo = 2

Quel état intéressant t=2 ?

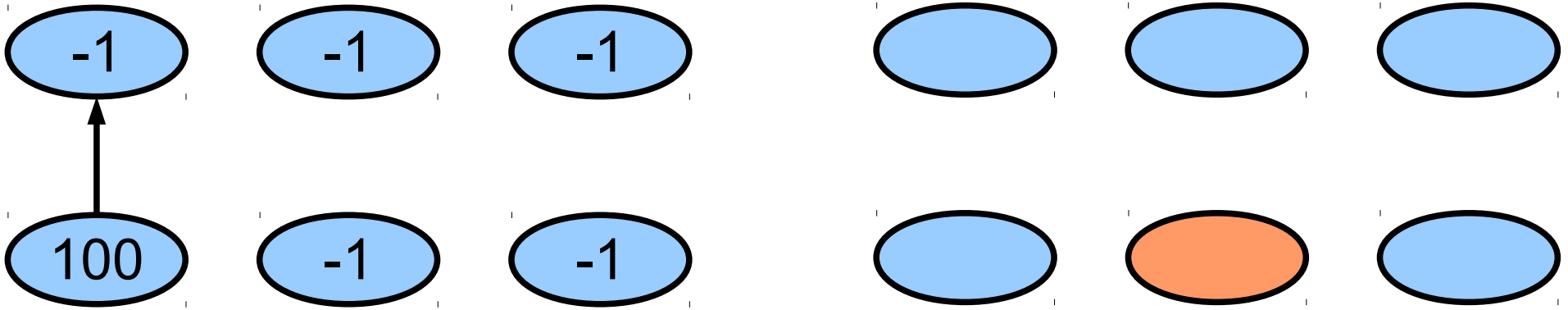


$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

- Part de la fin
  - Action dispo = 1
- Remonte temps
  - Action dispo = 2

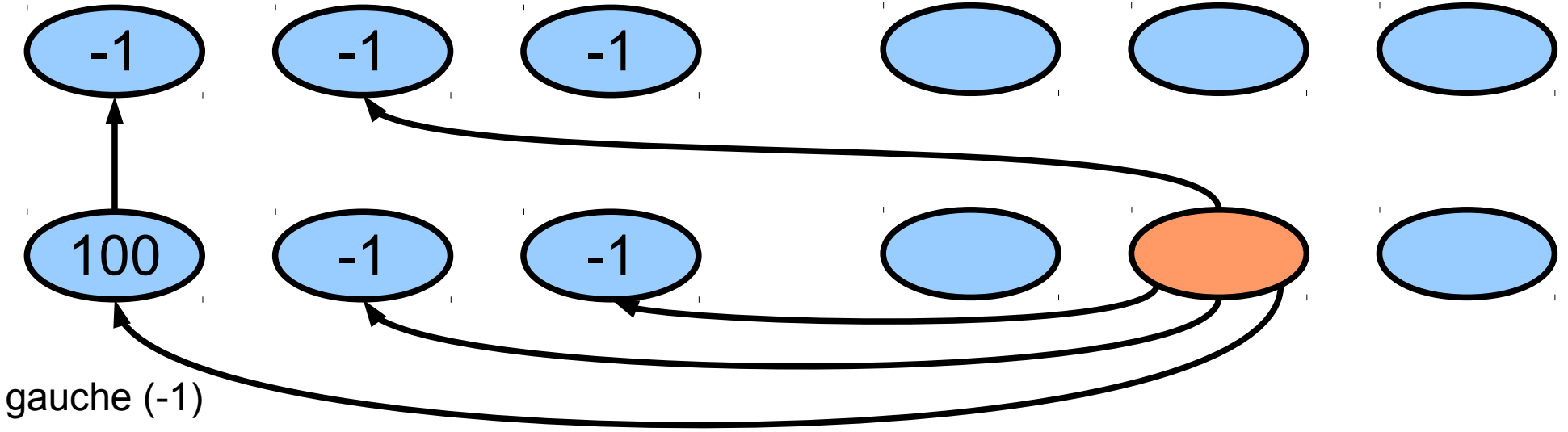
Quel état intéressant t=2 ?



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

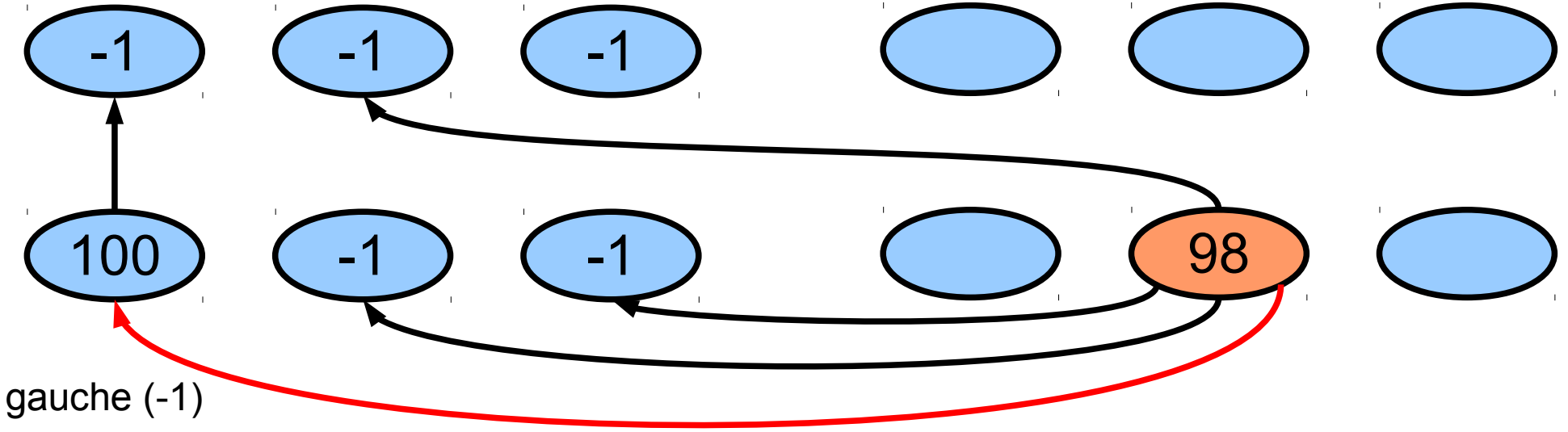
- Part de la fin
  - Action dispo = 1
- Remonte temps
  - Action dispo = 2



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

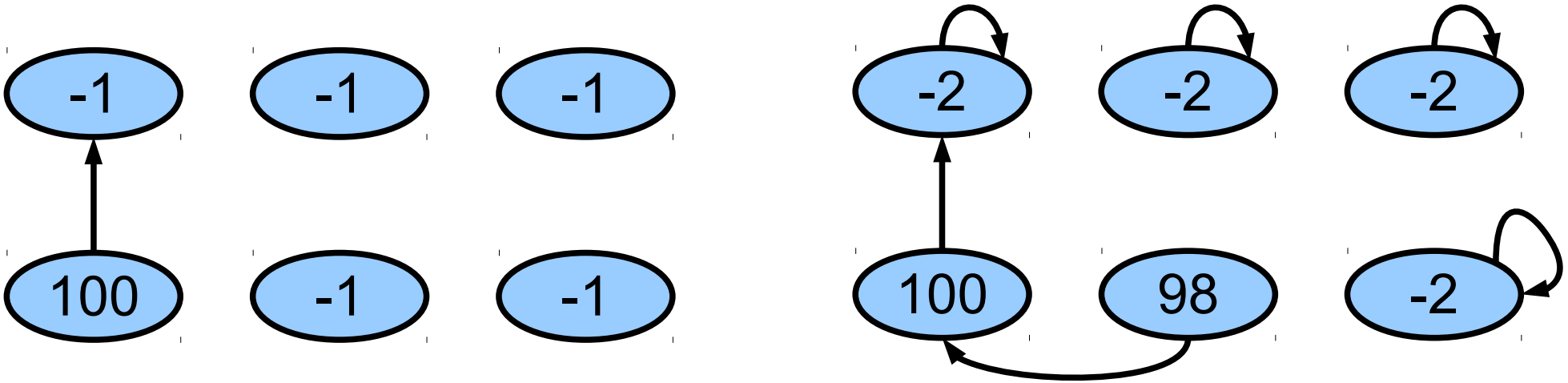
- Part de la fin
  - Action dispo = 1
- Remonte temps
  - Action dispo = 2



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Principe algorithme

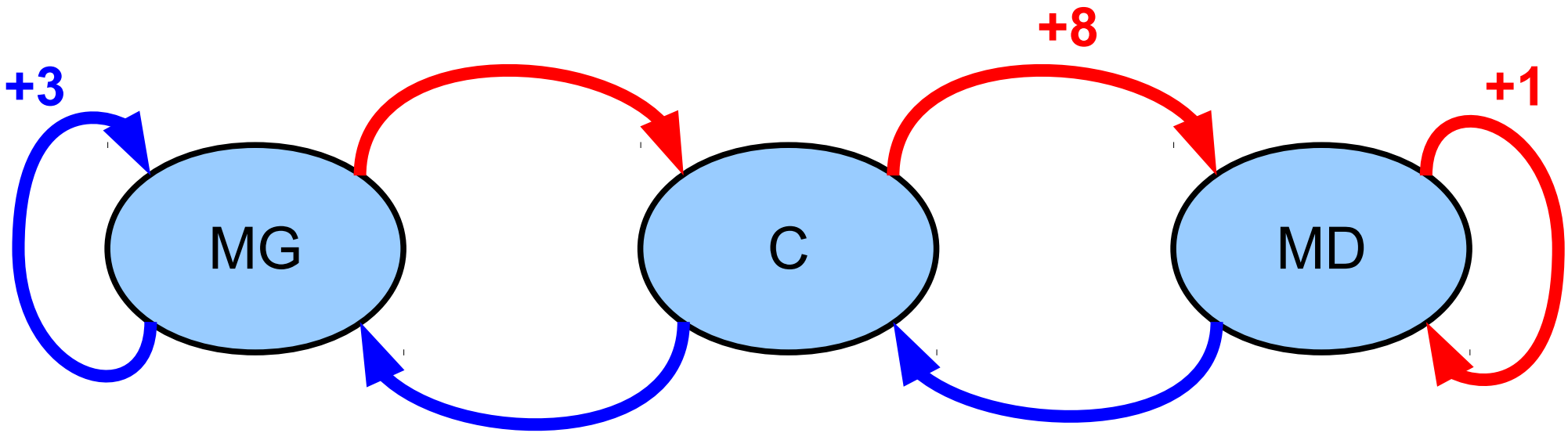
- Continue à remonter le temps
  - À chaque pas de temps, **applique Bellman**



$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*(T(s,a),a')$$

Exemple2 – chercheur or (fin diapos)

- 3 etats:
  - Chemin (C), Mine gauche/droite (MG/MD)
- 2 actions:
  - Gauche (G), droite (D)



Value iteration

- Repeter pour t
  - Chaque état et chaque action
    - $Q^*_{t+1}(s,a) = R(s,a,T(s,a)) + \gamma . max_{a'} Q^*_t(T(s,a),a')$

- Principe
  - Initialiser avec Q*=0
  - Remonter le temps
    - Q*(t+1) dépend de la valeur de l'état d'arrivée
  - Preuve de converge (pas magique => discussion)

- Python : Q = dictionnaire (etat,action) -> valeur

ValueIteration - commentaires

```
def valueIteration(self,nb,gamma):
        # initialiser Qvaleurs
        # faire nb iterations
                # executer une Maj de Q
        # retourner Q
```

```
def executerUneIteration(self,Q,gamma):
        # initialiser Q2
        # pour chaque etat
                # pour chaque action
                        # calculer arrivee et recompense
                        # cherche max arrivee
                        # Mise à jour Q2
                        Q2[(etat,action)]=r+gamma*max
        # retourne resultat Q2
```

```
def calculerMax(self,Q,sArriv):
        # pour chaque action
                # si c'est mieux que max, stocker max
        # retourner max
```

ValueIteration

```
def valueIteration(self,nb,gamma):
        Q=self.initialiserQ()
        #nb iteration iteration
        for i in range(0,nb):
            Q=self.executerUneIteration(Q,gamma)
        return(Q)
```

```
def initialiserQ(self):
        Q={}
        for s in self.pb.etats():
            for a in self.pb.actions():
                Q[(s,a)]=0
        return Q
```

ValueIteration

```
def executerUneIteration(self,Q,gamma):
        Q2={}
        #pour chaque etat,action
        for etat in self.pb.etats():
            for action in self.pb.actions():
                #calculer arrivee
                sArriv=self.pb.transition(etat,action)
                r=self.pb.recompense(etat,action,sArriv)
                # cherche max arrivee
                max=self.calculerMax(Q,sArriv)
                #mise à jour
                Q2[(etat,action)]=r+gamma*max
         #retourne resultat
        return(Q2)
```

```
def calculerMax(self,Q,sArriv):
        max=-100000
        for actionMax in self.pb.actions():
            if (Q[(sArriv,actionMax)]>max):
                max=Q[(sArriv,actionMax)]
        return(max)
```

```
def valueIteration(self,nb,gamma):
        Q={}
        for s in self.pb.etats():
            for a in self.pb.actions():
                Q[(s,a)]=0

        #une iteration
        for i in range(0,nb):
            Q2={}
            #pour chaque etat,action
            for etat in self.pb.etats():
                for action in self.pb.actions():
                    #calculer arrivee
                    sArriv=self.pb.transition(etat,action)
                    r=self.pb.recompense(etat,action,sArriv)
                    # cherche max arrivee
                    max=-100000
                    for actionMax in self.pb.actions():
                        if (Q[(sArriv,actionMax)]>max):
                            max=Q[(sArriv,actionMax)]

                    #mise à jour
                    Q2[(etat,action)]=r+gamma*max

            #on augmente iteration
            Q=Q2
        return(Q)
```

Politique optimale

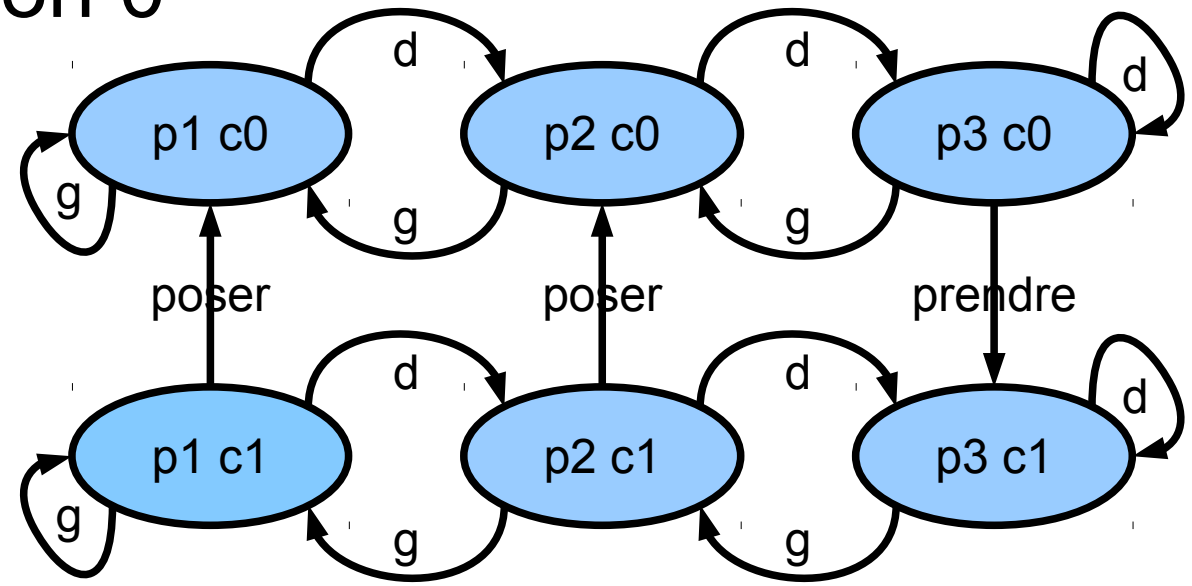
- Théorème : Value iteration converge
  - Preuve : Fonction recurrente contractante

- Politique optimale
  - Choisir action max Q* ==> π(s) = Argmax (Q*(s,a))

```
def politiqueFromQ(self,Q):
    pi={}
    # pour chaque état
    for etat in self.pb.etats():
        max = -100000
        amax = ''
        # cherche le maximum
        for actionMax in self.pb.actions():
            if (Q[(etat,actionMax)]>max):
                max=Q[(etat,actionMax)]
                amax=actionMax
        pi[etat]=amax
    return(pi)
```

Exemple RobotCafe

- Iteration 0



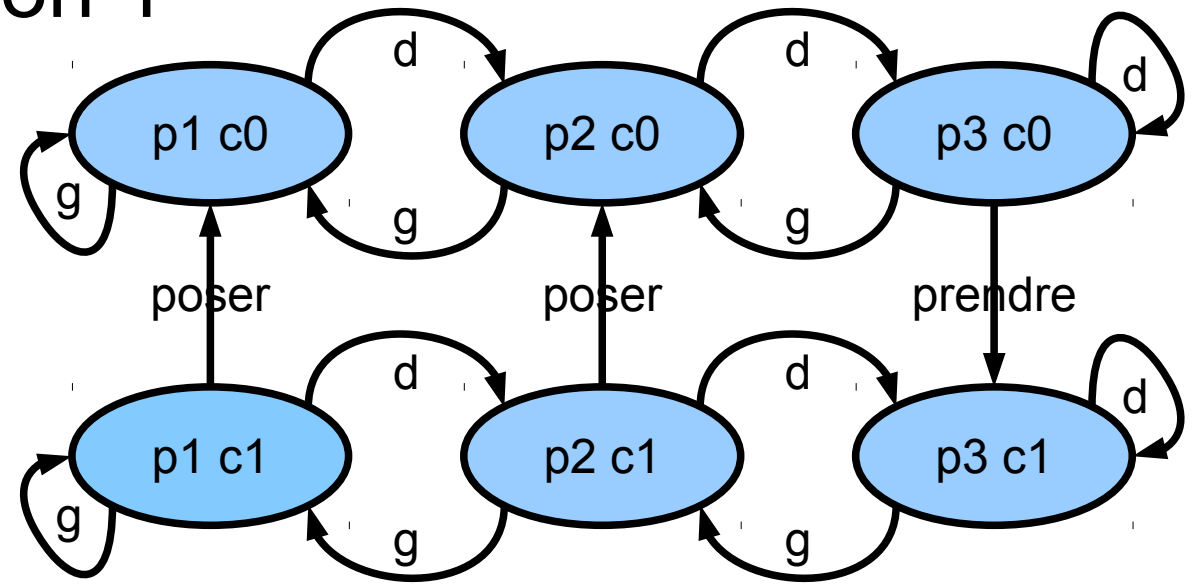

```
(1, 0)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0

(1, 1)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0
```

```
(2, 0)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0

(2, 1)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0
```

```
(3, 0)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0

(3, 1)
   - gauche -> 0
   - droite -> 0
   - prendre -> 0
   - poser -> 0
```

Exemple RobotCafe

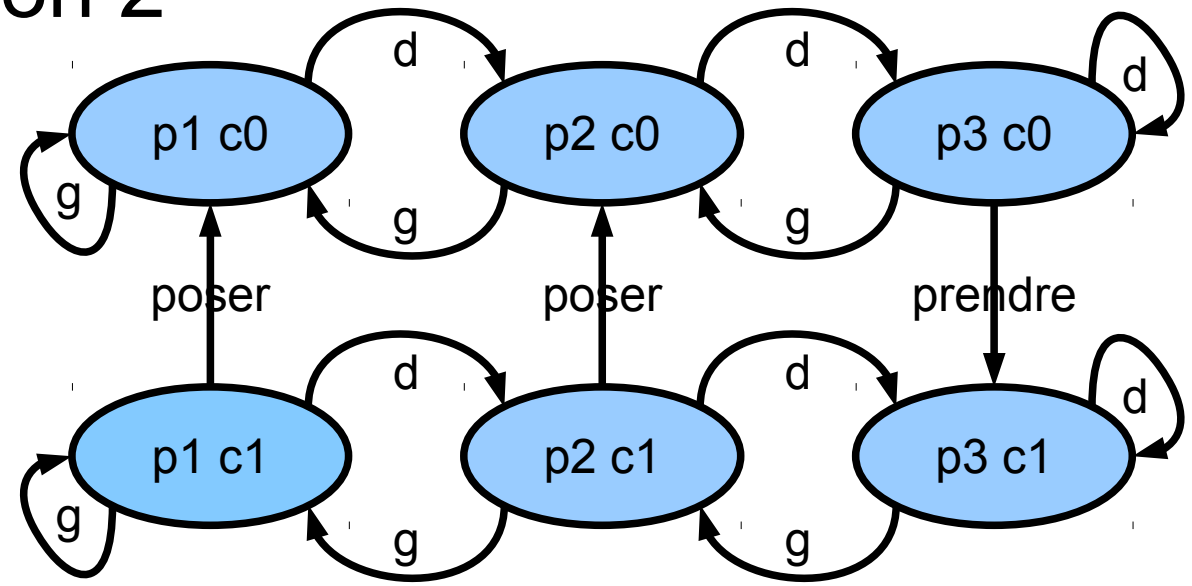
- Iteration 1




```
(1, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(1, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> 100.0

(2, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(2, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(3, 0)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0

(3, 1)
   - gauche -> -1.0
   - droite -> -1.0
   - prendre -> -1.0
   - poser -> -1.0
```

Exemple RobotCafe

- Iteration 2



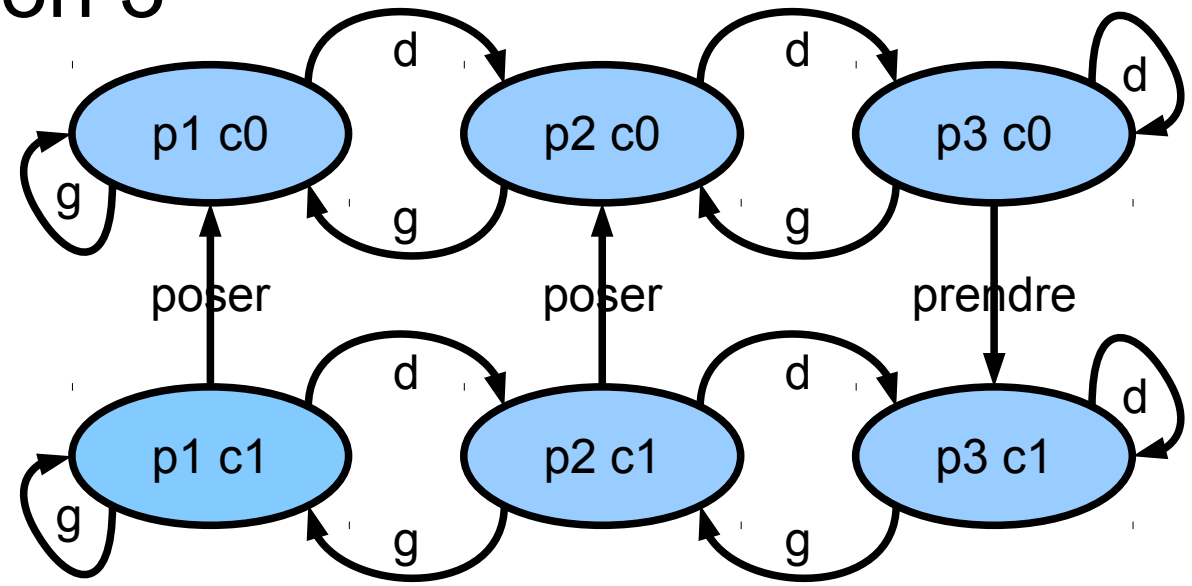

```
(1, 0)
   - gauche -> -1.99
   - droite -> -1.99
   - prendre -> -1.99
   - poser -> -1.99

(1, 1)
   - gauche -> 98.0
   - droite -> -1.99
   - prendre -> 98.0
   - poser -> 99.01
```

```
(2, 0)
   - gauche -> -1.99
   - droite -> -1.99
   - prendre -> -1.99
   - poser -> -1.99

(2, 1)
   - gauche -> 98.0
   - droite -> -1.99
   - prendre -> -1.99
   - poser -> -1.99
```

```
(3, 0)
   - gauche -> -1.99
   - droite -> -1.99
   - prendre -> -1.99
   - poser -> -1.99

(3, 1)
   - gauche -> -1.99
   - droite -> -1.99
   - prendre -> -1.99
   - poser -> -1.99
```

Exemple RobotCafe

- Iteration 3



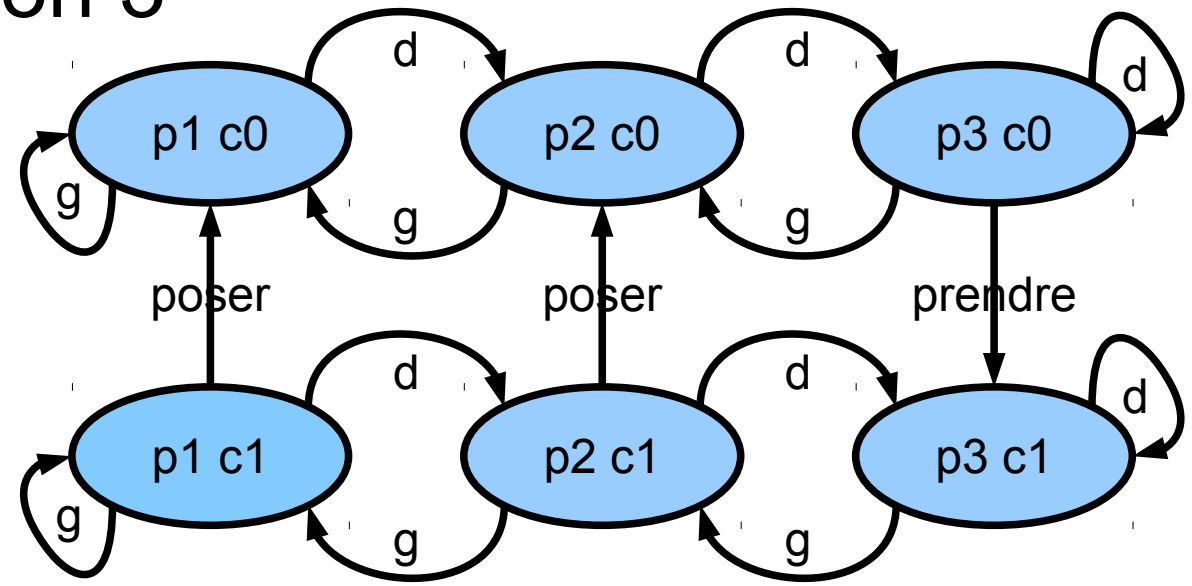

```
(1, 0)
   - gauche -> -2.97
   - droite -> -2.97
   - prendre -> -2.97
   - poser -> -2.97

(1, 1)
   - gauche -> 97.02
   - droite -> 96.02
   - prendre -> 97.02
   - poser -> 98.03
```

```
(2, 0)
   - gauche -> -2.97
   - droite -> -2.97
   - prendre -> -2.97
   - poser -> -2.97

(2, 1)
   - gauche -> 97.02
   - droite -> -2.97
   - prendre -> 96.02
   - poser -> -2.97
```

```
(3, 0)
   - gauche -> -2.97
   - droite -> -2.97
   - prendre -> -2.97
   - poser -> -2.97

(3, 1)
   - gauche -> 96.02
   - droite -> -2.97
   - prendre -> -2.97
   - poser -> -2.97
```

Exemple RobotCafe

- Iteration 3



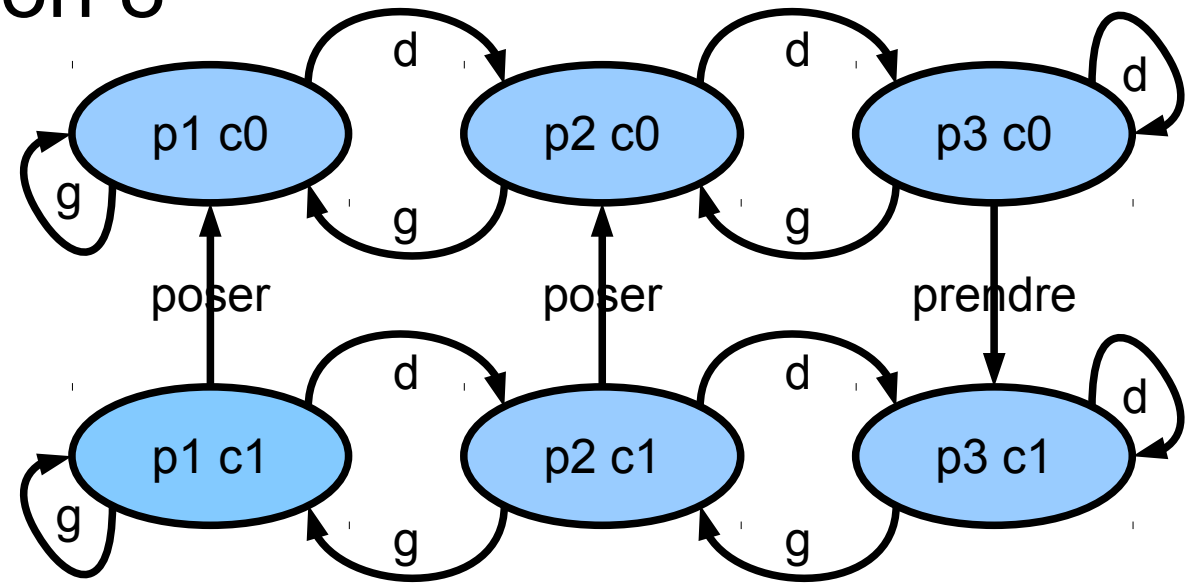

```
(1, 0)
   - gauche -> -3.94
   - droite -> -3.94
   - prendre -> -3.94
   - poser -> -3.94

(1, 1)
   - gauche -> 96.05
   - droite -> 95.05
   - prendre -> 96.05
   - poser -> 97.06
```

```
(2, 0)
   - gauche -> -3.94
   - droite -> -3.94
   - prendre -> -3.94
   - poser -> -3.94

(2, 1)
   - gauche -> 96.05
   - droite -> 94.06
   - prendre -> 95.05
   - poser -> -3.94
```

```
(3, 0)
   - gauche -> -3.94
   - droite -> -3.94
   - prendre -> 94.06
   - poser -> -3.94

(3, 1)
   - gauche -> 95.05
   - droite -> 94.06
   - prendre -> 94.06
   - poser -> -3.94
```

Exemple RobotCafe

- Iteration 8




```
(1, 0)
   - gauche -> 86.44
   - droite -> 87.4
   - prendre -> 86.44
   - poser -> 86.44

(1, 1)
   - gauche -> 185.48
   - droite -> 183.54
   - prendre -> 185.48
   - poser -> 187.44
```

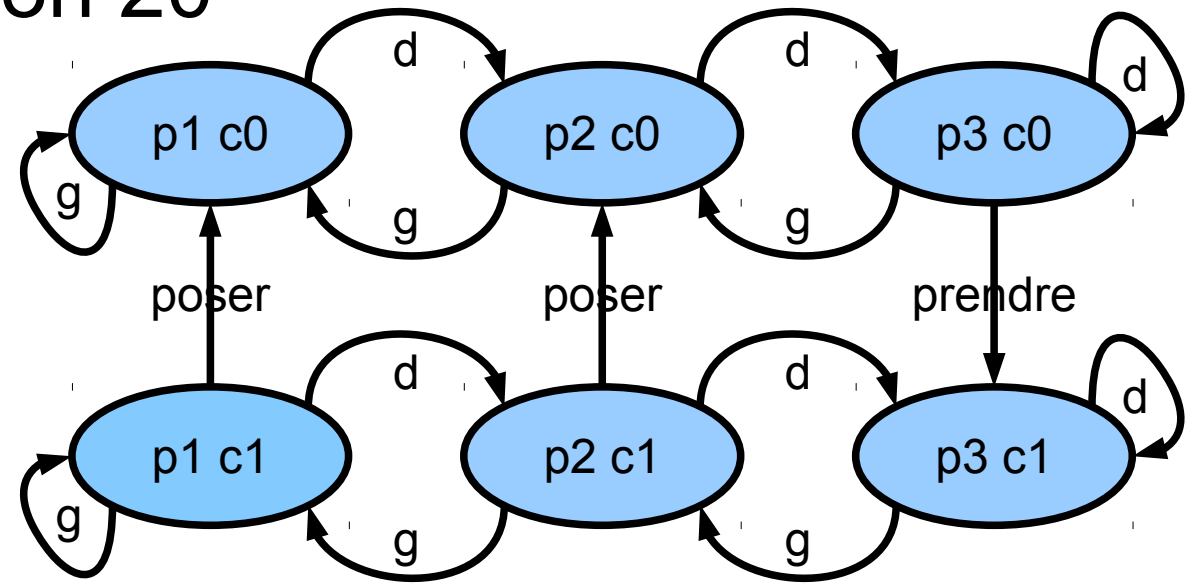
```
(2, 0)
   - gauche -> 86.44
   - droite -> 88.37
   - prendre -> 87.4
   - poser -> 87.4

(2, 1)
   - gauche -> 185.48
   - droite -> 89.35
   - prendre -> 183.54
   - poser -> 87.4
```

```
(3, 0)
   - gauche -> 87.4
   - droite -> 88.37
   - prendre -> 89.35
   - poser -> 88.37

(3, 1)
   - gauche -> 183.54
   - droite -> 89.35
   - prendre -> 89.35
   - poser -> 88.37
```

Surprenant ?

Exemple RobotCafe

- Iteration 20




```
(1, 0)
   - gauche -> 250.69
   - droite -> 253.41
   - prendre -> 250.69
   - poser -> 250.69

(1, 1)
   - gauche -> 347.99
   - droite -> 261.72
   - prendre -> 347.99
   - poser -> 351.69
```

```
(2, 0)
   - gauche -> 250.69
   - droite -> 256.15
   - prendre -> 253.41
   - poser -> 253.41

(2, 1)
   - gauche -> 347.99
   - droite -> 258.92
   - prendre -> 261.72
   - poser -> 253.41
```

```
(3, 0)
   - gauche -> 253.41
   - droite -> 256.15
   - prendre -> 258.92
   - poser -> 256.15

(3, 1)
   - gauche -> 261.72
   - droite -> 258.92
   - prendre -> 258.92
   - poser -> 256.15
```

Exemple RobotCafe

- Gerer tous les problèmes de cette forme

- Experimentations - Gère les compromis
  - Modifie gain
    - Recompense café = 4
  - Ajoute action rien
    - Action += { rien }

Plan

- Problème de prise de décision séquentiel
- Exemples

- (1) Représenter un problème
- (2) Equation de Bellman
- (3) Algorithme de résolution

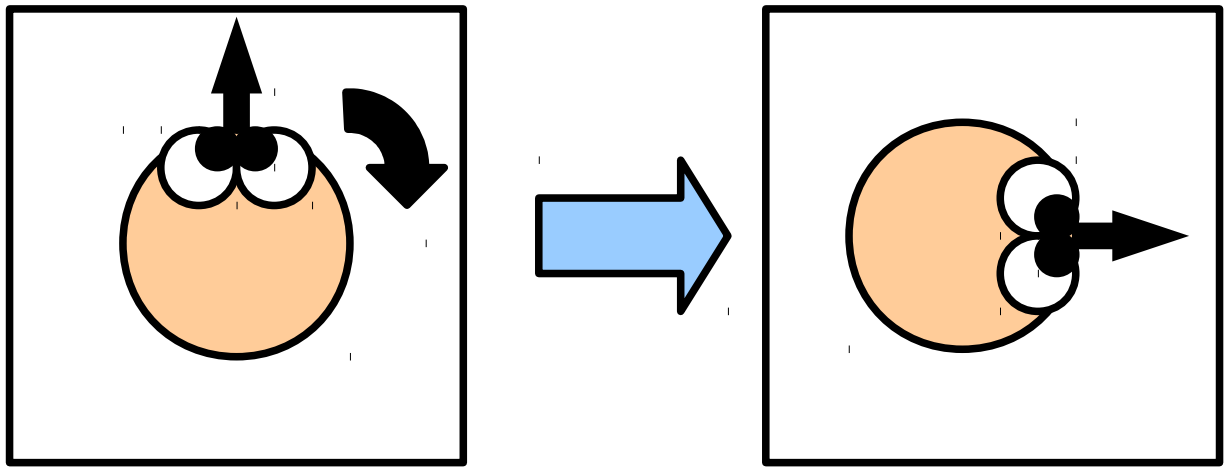
- Perspectives

Autres exemples

- Principe
  - Algorithme générique
  - Adapté à TOUS les problèmes (cadre)

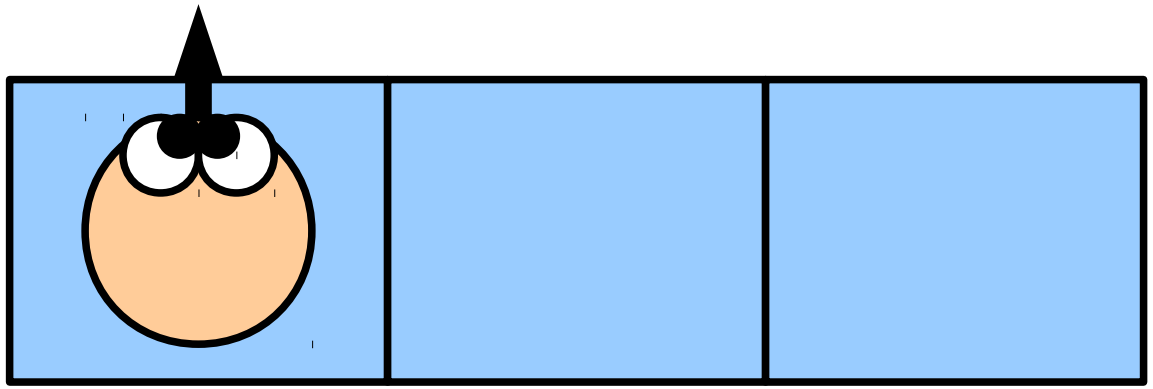
- Problème gestion orientation

Probleme Orientation



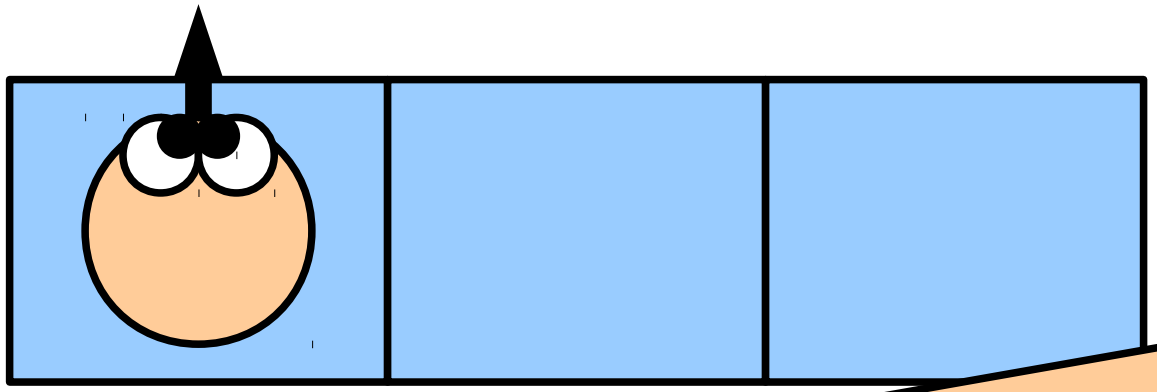
- Agent dispose d'une orientation
  - Tourner
  - Avancer
- Modifie espace d'état
  - Position + orientation

Probleme Orientation



- Robot Cafe (bis)
  - Action
    - avancer, tournerG, tournerD, prendre, poser
  - Etat
    - Position, orientation, cafe
  - Transition : S x A -> S
    - Gère orientation et déplacement
  - Recompense : S x A -> Reel
    - Amener café

Probleme Orientation



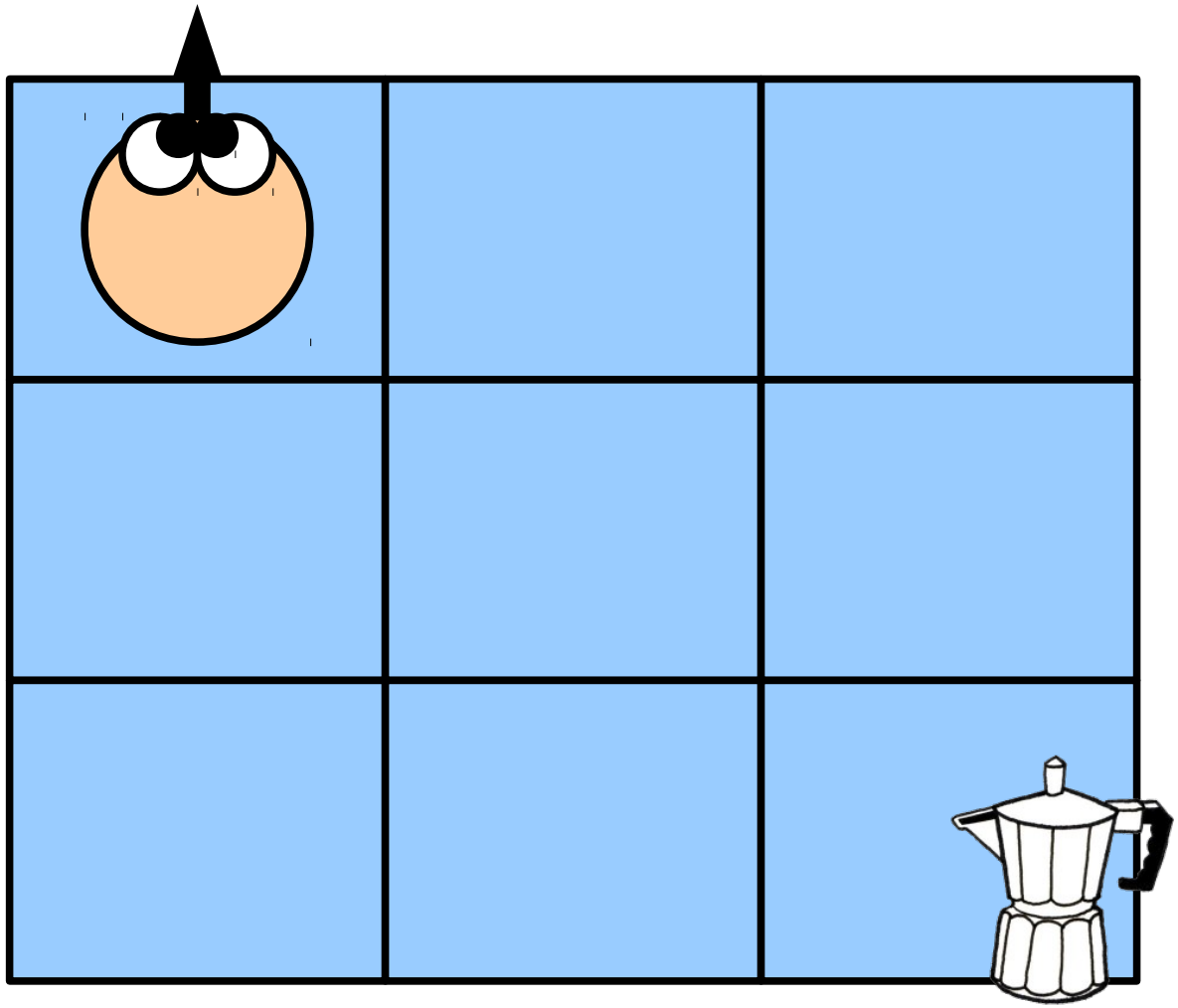
- Robot Cafe (bis)
- Gère orientation et déplacement
  - Recompense : S x A -> Reel
    - Amener café

La « preuve » (???) !

04_cafe_rotation

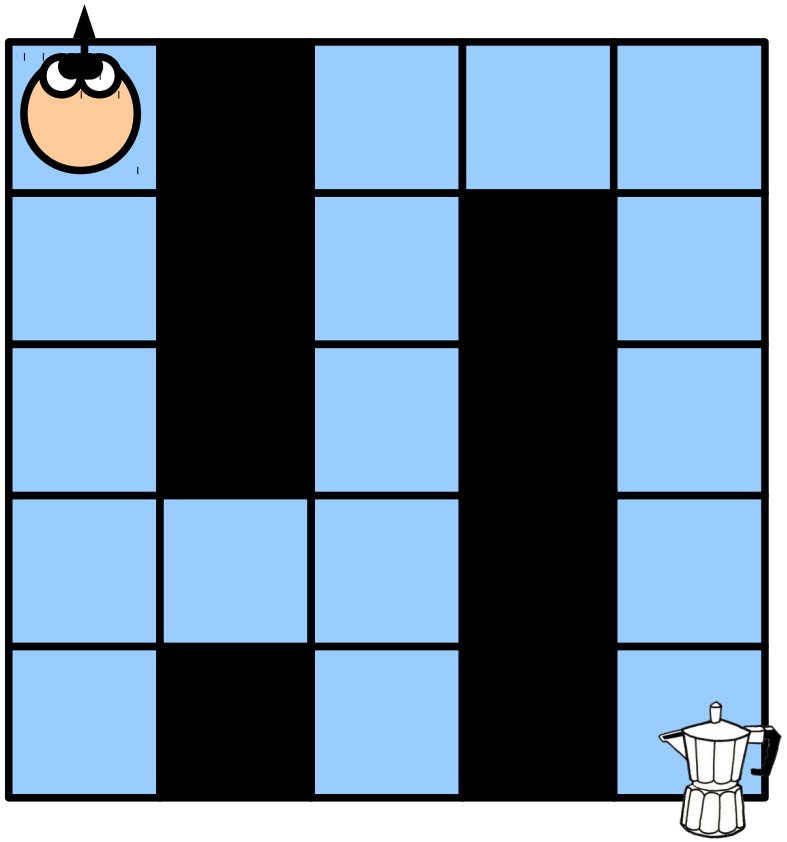
Probleme Orientation

- Passer en 2D
  - 06_cafeRotation2D



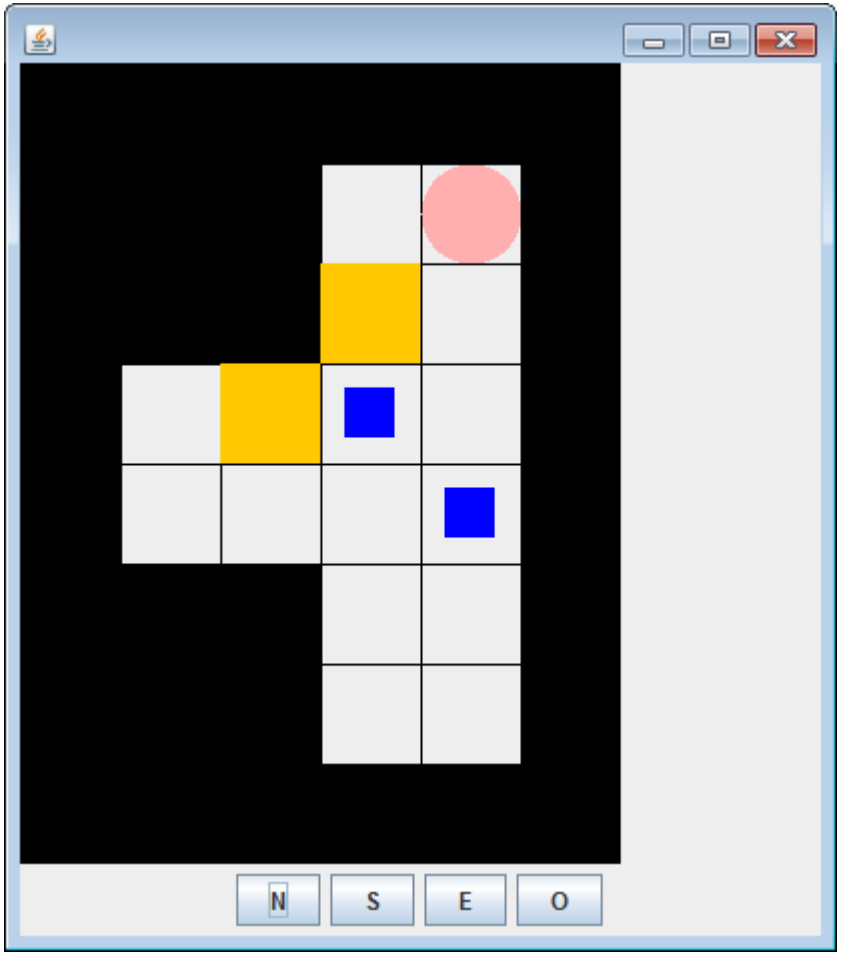
Probleme Orientation

- Ajouter des murs
  - 07_cafe_rotation_2D_murs



Sokoban

- Workspace Sokoban



Autres exemples

- Orientation
- Labyrinthe (cf slides bonus)
- Sokoban (cf code Java)
- Robot et plaque de pression
- Glissement sol (cf site)

- Autre exemple value iteration (cf slides bonus)

Perspectives

- Stochastique
  - MDP
- Taille de l'espace etat ou Continu
  - approximation, MCTS
- Modele inconnu
  - Apprentissage
- Guider la recherche A*
  - Heuristiques
- Observabilite partielle (wumpus)
  - POMDP

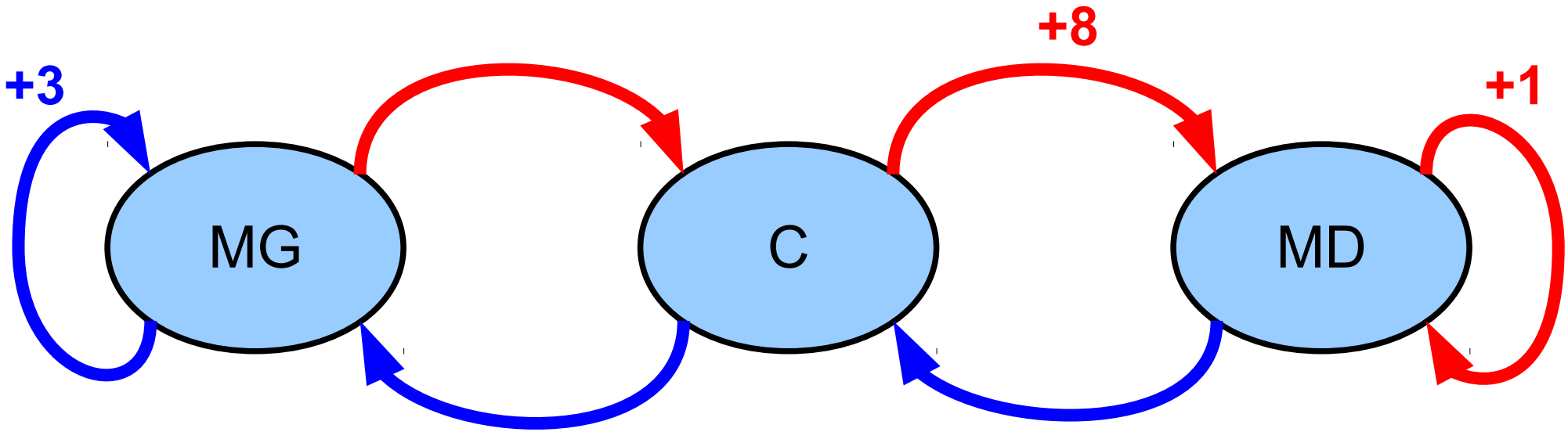
Slides Bonus

(issus principalement 2016)

Plan slides Bonus

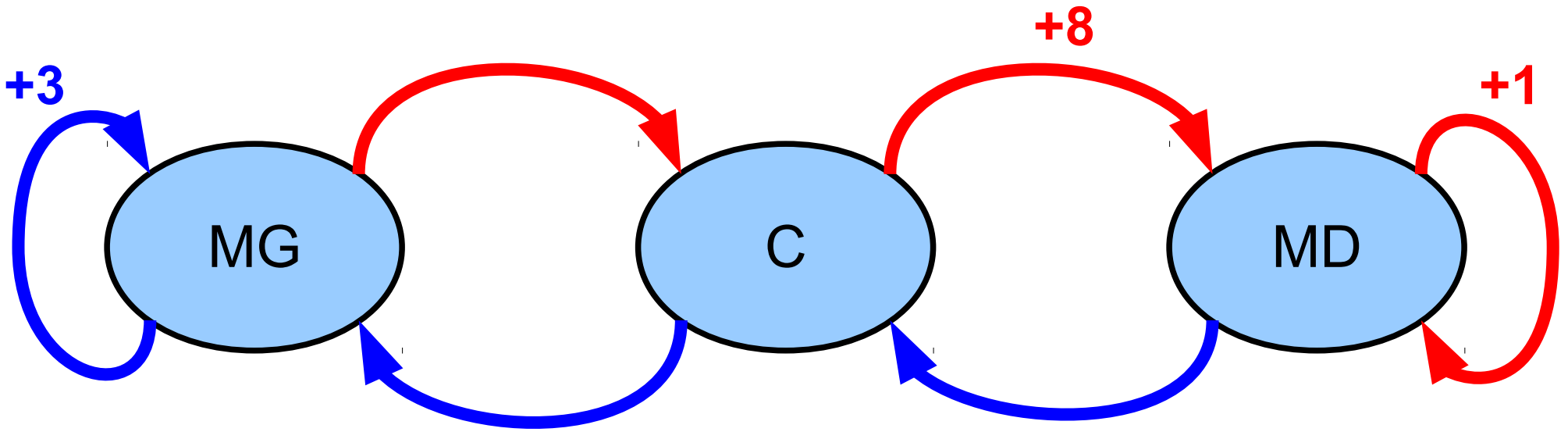
- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Observabilité partielle

Exemple2 – chercheur or



- Meilleure politique ?

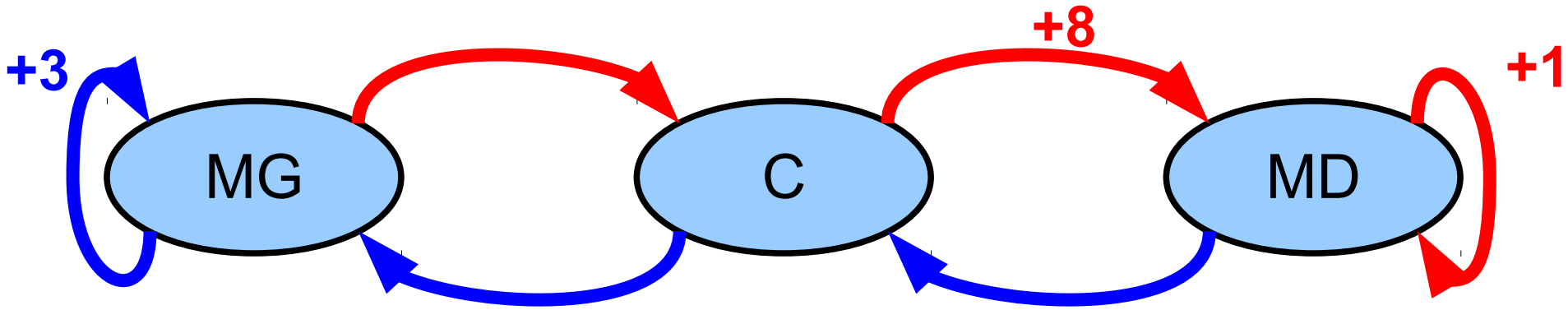
Exemple – chercheur or



- Récriture Equation Bellman

$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

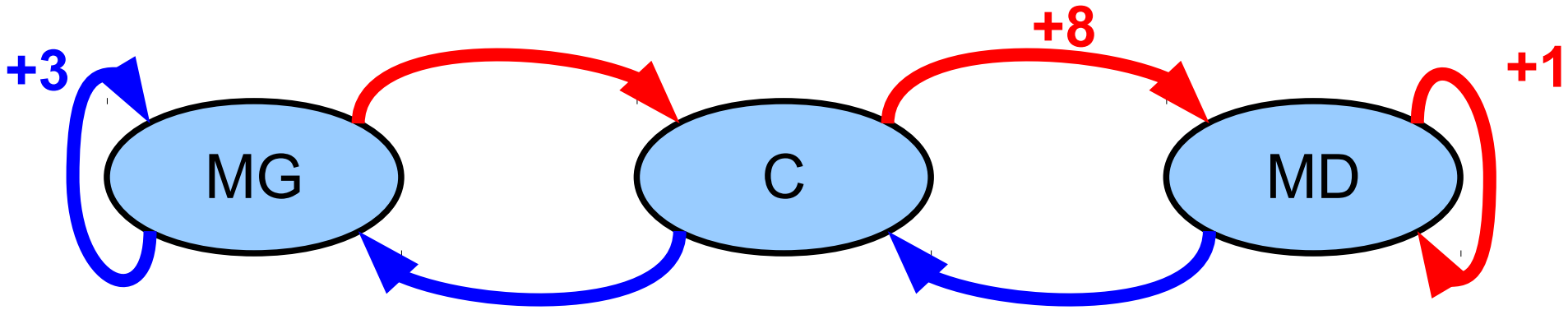
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

- 6 cas

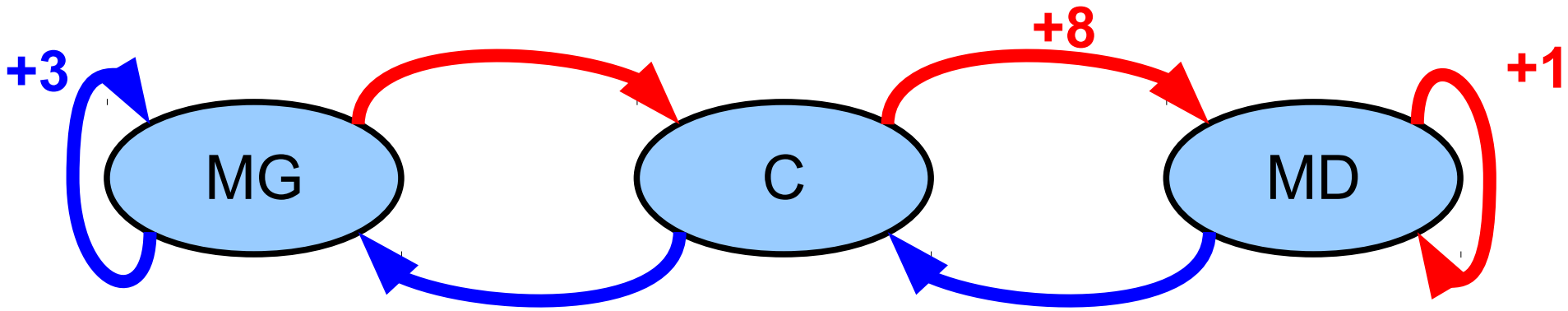
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

- 6 cas
  - MG + Gauche -> MG, +3
  - MG + Droite -> C, +0
  - C + Droite -> MD, +8
  - C + Gauche -> MG, +0
  - MD + Gauche -> C, +0
  - MD + Droite -> MD, +1

Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

$$Q^*(MG,G) = R(MG,G,MG) + 0{,}9.max(Q^*(MG,G),Q^*(MG,D))$$

$$Q^*(MG,D) = R(MG,D,C) + 0{,}9.max(Q^*(C,G),Q^*(C,D))$$

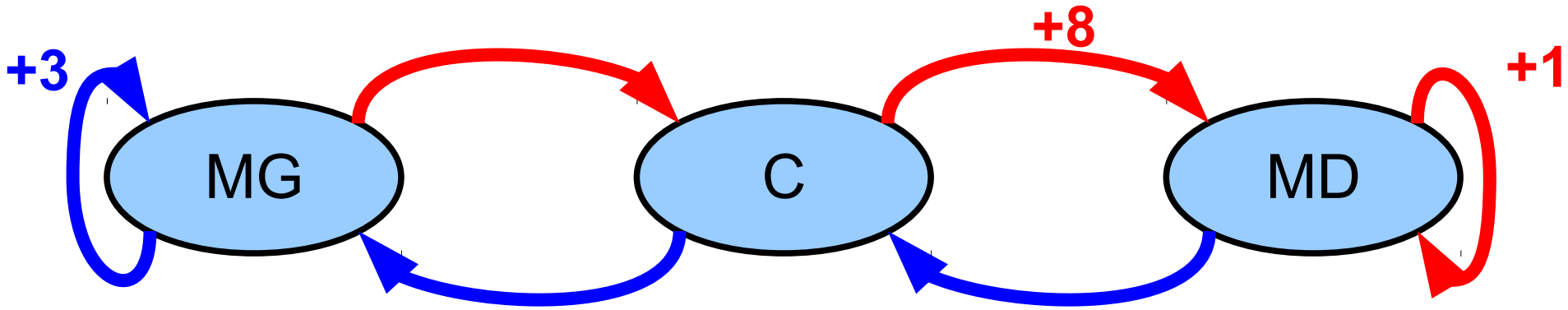
$$Q^*(C,G) = R(C,G,MG) + 0{,}9.max(Q^*(MG,G),Q^*(MG,D))$$

$$Q^*(C,D) = R(C,D,MD) + 0{,}9.max(Q^*(MD,G),Q^*(MD,D))$$

$$Q^*(MD,G) = R(MD,G,C) + 0{,}9.max(Q^*(C,G),Q^*(C,D))$$

$$Q^*(MD,D) = R(MD,D,MD) + 0{,}9.max(Q^*(MD,G),Q^*(MD,D))$$

Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

$$Q^*(MG,G) = 3 + 0{,}9.max(Q^*(MG,G), Q^*(MG,D))$$

$$Q^*(MG,D) = 0 + 0{,}9.max(Q^*(C,G), Q^*(C,D))$$

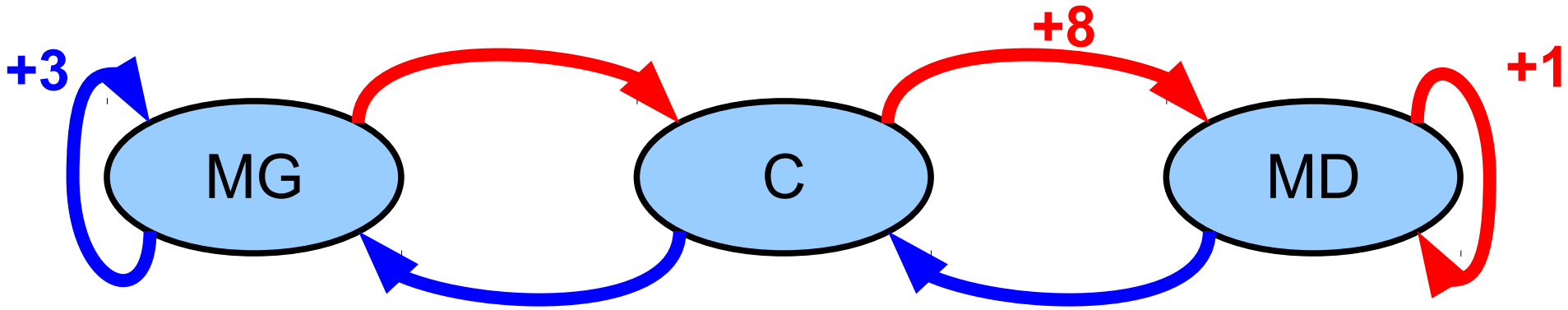
$$Q^*(C,G) = 0 + 0{,}9.max(Q^*(MG,G), Q^*(MG,D))$$

$$Q^*(C,D) = 8 + 0{,}9.max(Q^*(MD,G), Q^*(MD,D))$$

$$Q^*(MD,G) = 0 + 0{,}9.max(Q^*(C,G), Q^*(C,D))$$

$$Q^*(MD,D) = 1 + 0{,}9.max(Q^*(MD,G), Q^*(MD,D))$$

Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

$$Q^*(MG,G) = 3 + 0{,}9.max(Q^*(MG,G), Q^*(MG,D))$$

$$Q^*(MG,D) = 0{,}9.max(Q^*(C,G), Q^*(C,D))$$

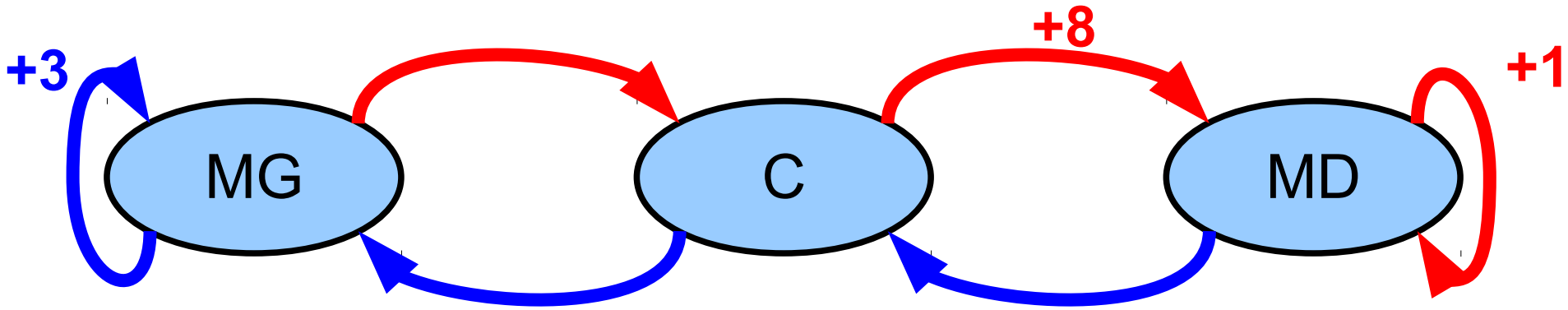
$$Q^*(C,G) = 0{,}9.max(Q^*(MG,G), Q^*(MG,D))$$

$$Q^*(C,D) = 8 + 0{,}9.max(Q^*(MD,G), Q^*(MD,D))$$

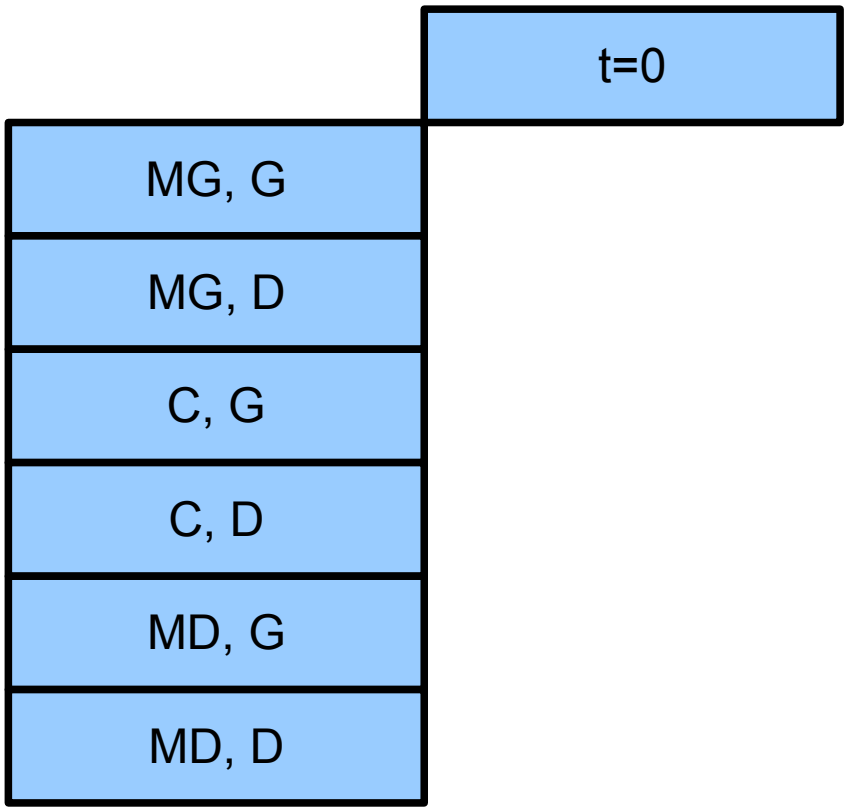
$$Q^*(MD,G) = 0{,}9.max(Q^*(C,G), Q^*(C,D))$$

$$Q^*(MD,D) = 1 + 0{,}9.max(Q^*(MD,G), Q^*(MD,D))$$

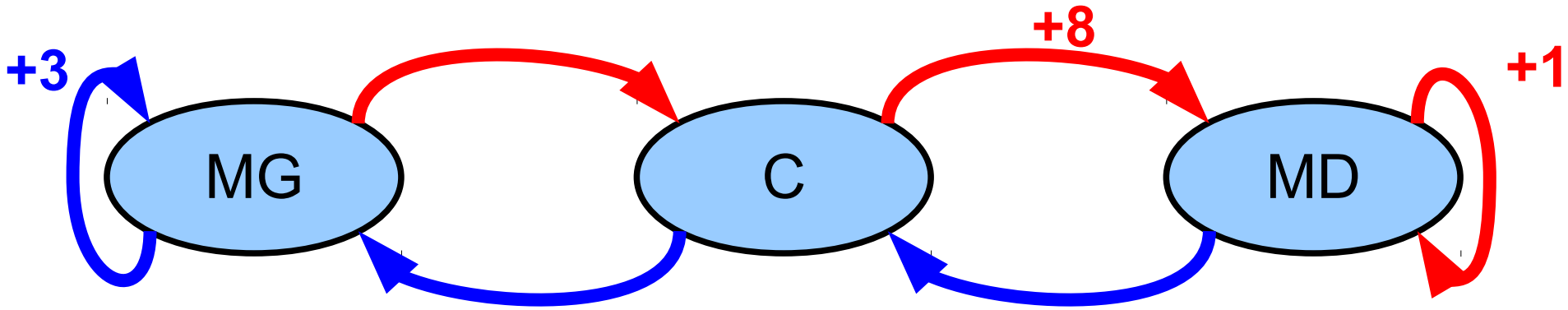
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$



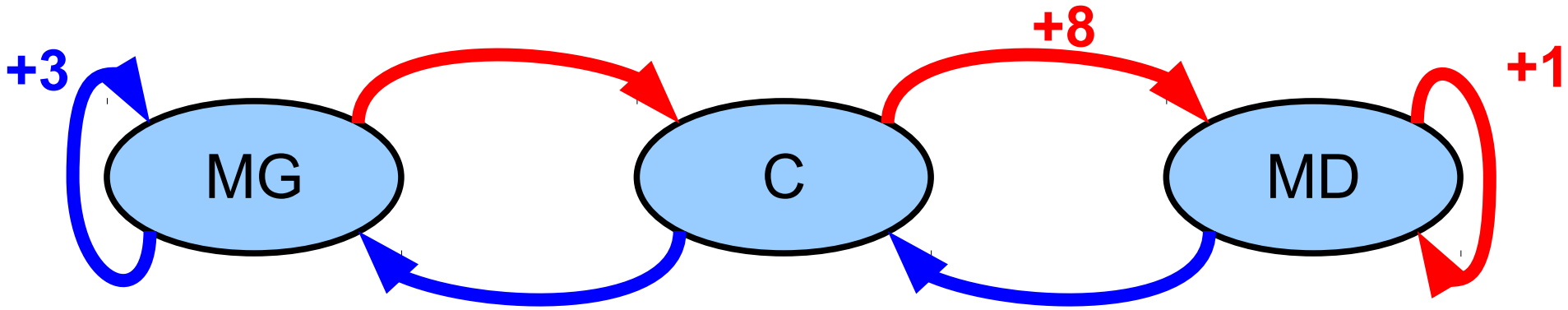
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0
MG, G	0
MG, D	0
C, G	0
C, D	0
MD, G	0
MD, D	0

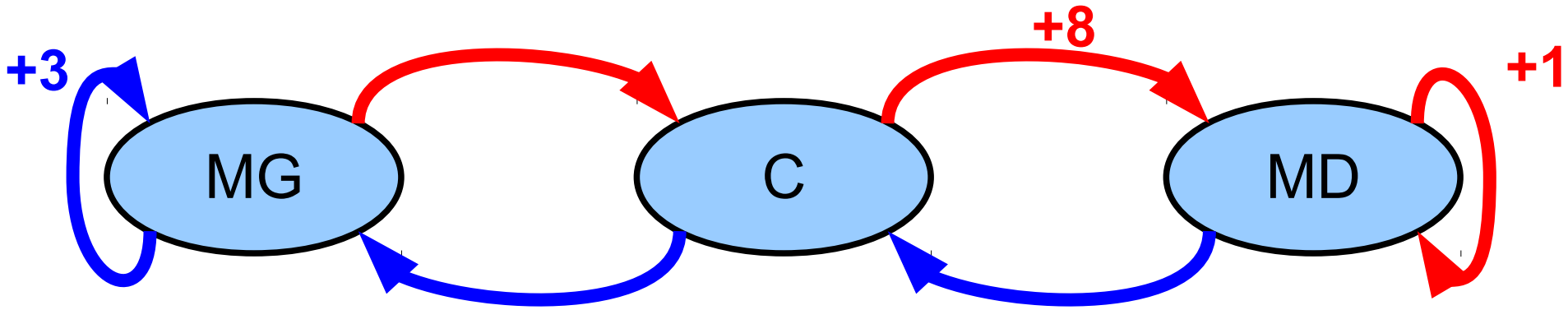
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	
MG, D	0	
C, G	0	
C, D	0	
MD, G	0	
MD, D	0	

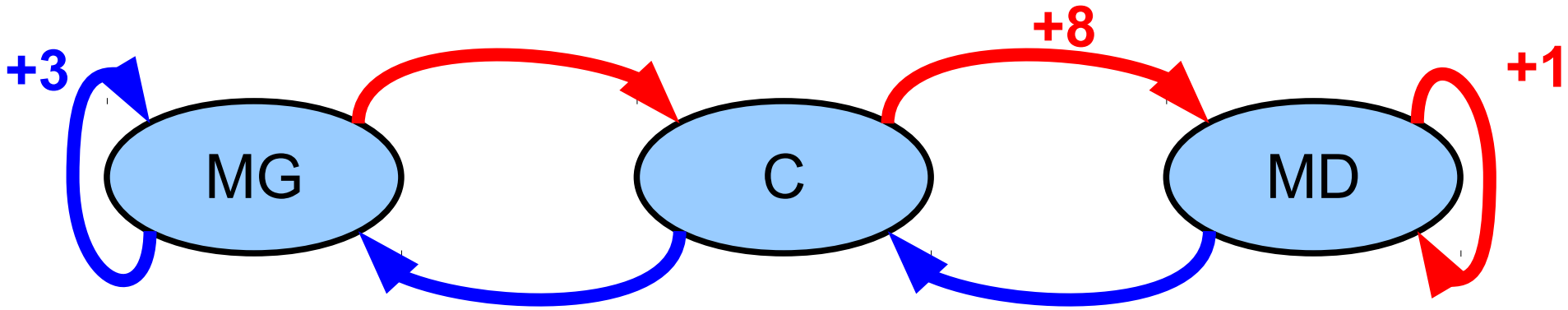
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	= 3 +0,9 max Q(MG,G), Q(MG,D)
MG, D	0	
C, G	0	
C, D	0	
MD, G	0	
MD, D	0	

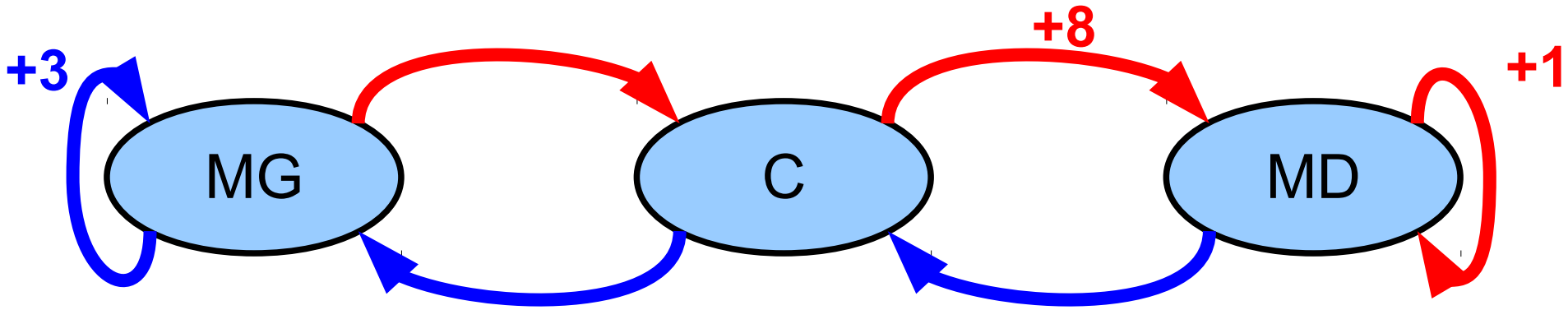
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	= 3
MG, D	0	
C, G	0	
C, D	0	
MD, G	0	
MD, D	0	

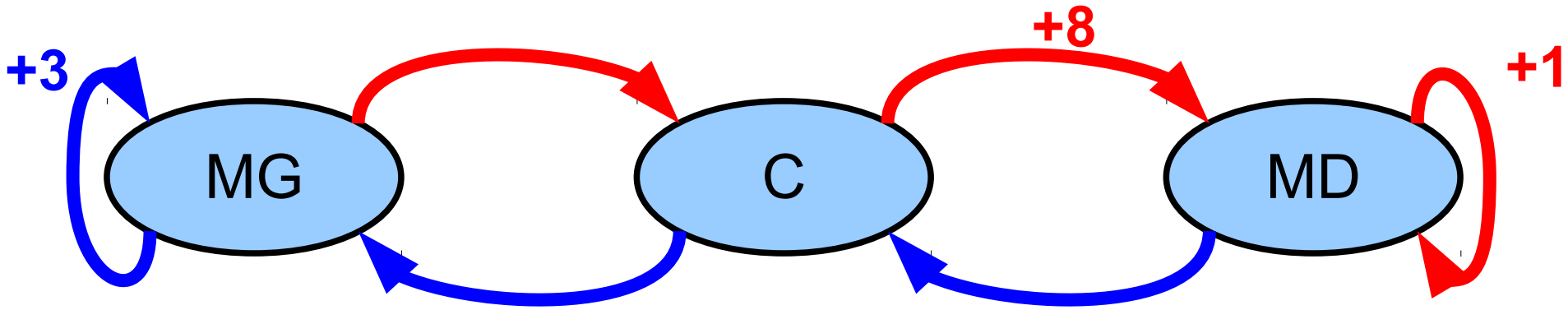
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	= 3
MG, D	0	= 0 + 0,9 max (Q (C,G), Q(C,D)
C, G	0	
C, D	0	
MD, G	0	
MD, D	0	

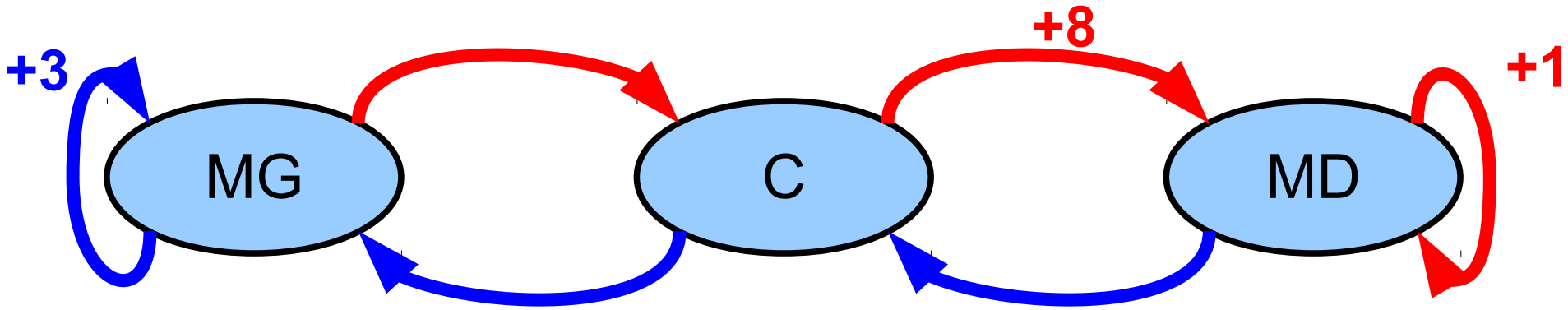
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	= 3
MG, D	0	= 0
C, G	0	= 0 + max (Q(MD, G) Q(MD,D))
C, D	0	= 8 + max (Q(MD,G), Q(MD,G))
MD, G	0	= 0 + max (Q(C,G), Q(C,D))
MD, D	0	= 1 + max (Q(MD,G),Q(MD,D))

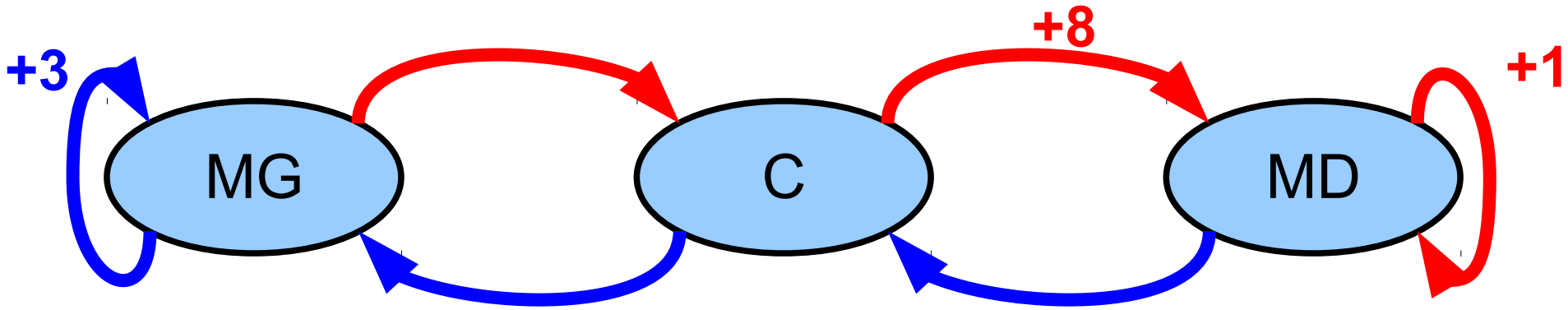
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	= 3
MG, D	0	= 0
C, G	0	= 0
C, D	0	= 8
MD, G	0	= 0
MD, D	0	= 1

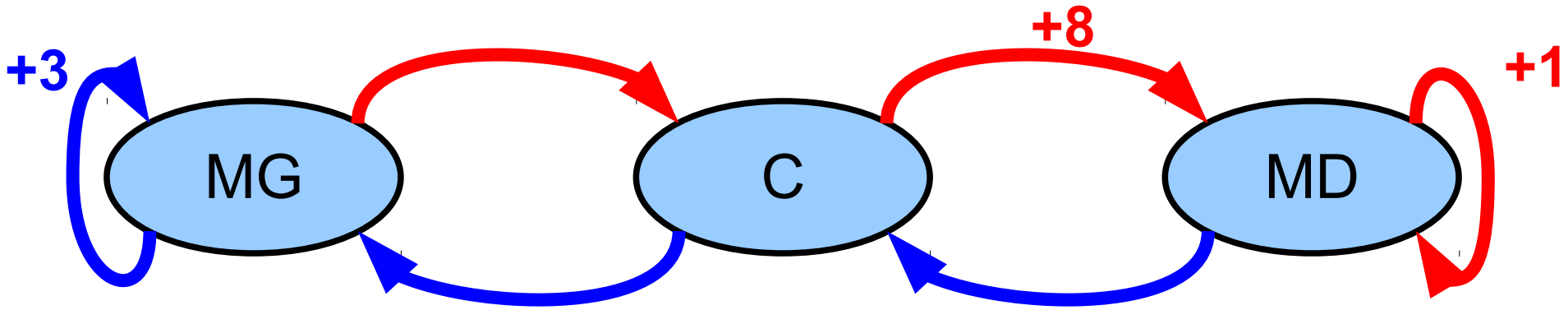
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1
MG, G	0	3
MG, D	0	0
C, G	0	0
C, D	0	8
MD, G	0	0
MD, D	0	1

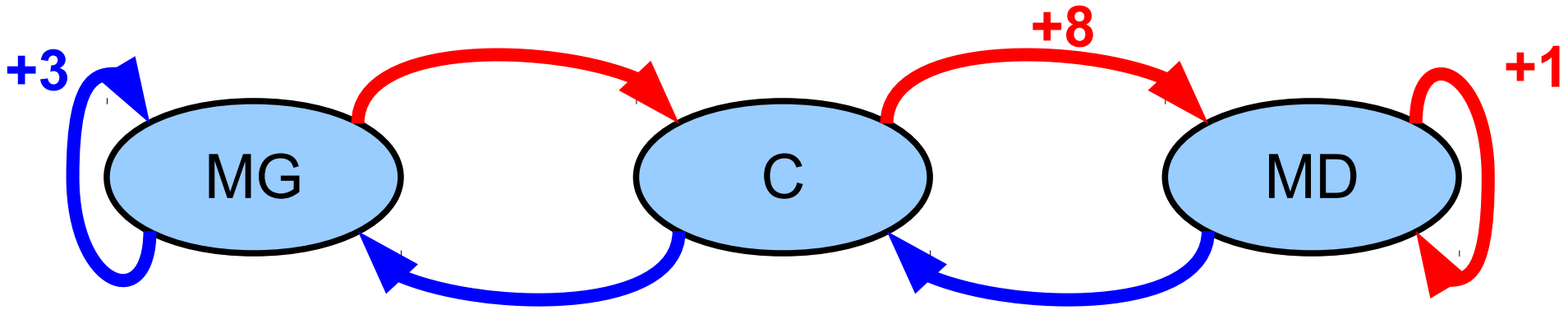
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1		t=2
MG, G	0	3	MG	= 3 + 0,9 . max Q(MG)
MG, D	0	0		= 0 + 0,9 . max (C)
C, G	0	0	C	= 0 + 0,9 . max (MG)
C, D	0	8		= 8 + 0,9 . max (MD)
MD, G	0	0	MD	= 0 + 0,9 . max (C)
MD, D	0	1		= 1 + 0,9 . max (MD)

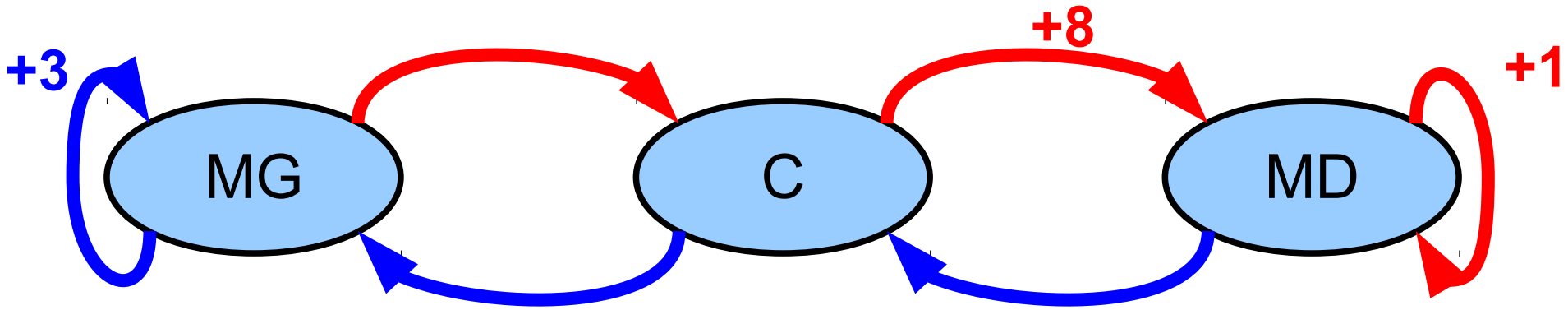
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1		t=2
MG, G	0	3	MG	= 3 + 0,9 * 3
MG, D	0	0		= 0 + 0,9 *8
C, G	0	0	C	= 0 + 0,9 * 3
C, D	0	8		= 8 + 0,9 * 1
MD, G	0	0	MD	= 0 + 0,9 * 8
MD, D	0	1		= 1 + 0,9 * 1

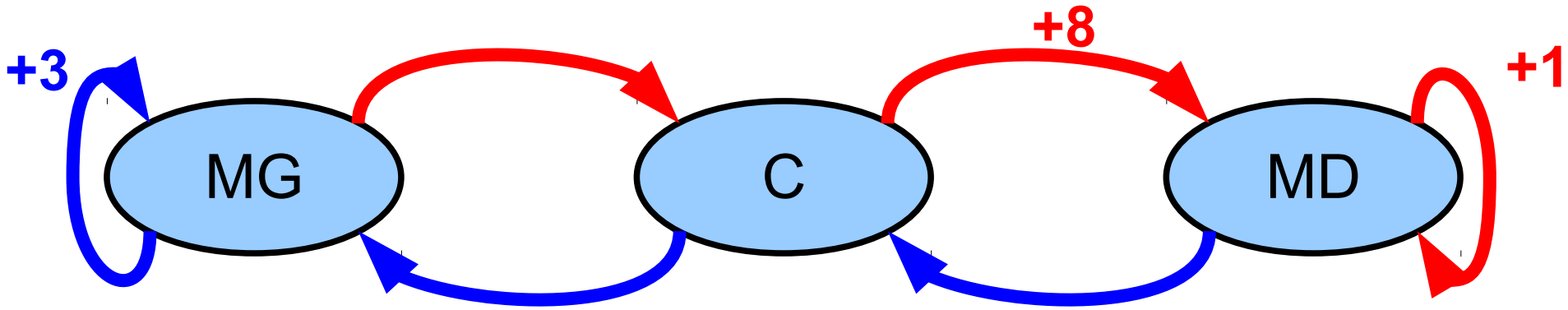
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1		t=2
MG, G	0	3	MG	= 5,7
MG, D	0	0		= 7,2
C, G	0	0	C	= 2,7
C, D	0	8		= 8,9
MD, G	0	0	MD	= 7,2
MD, D	0	1		= 1,9

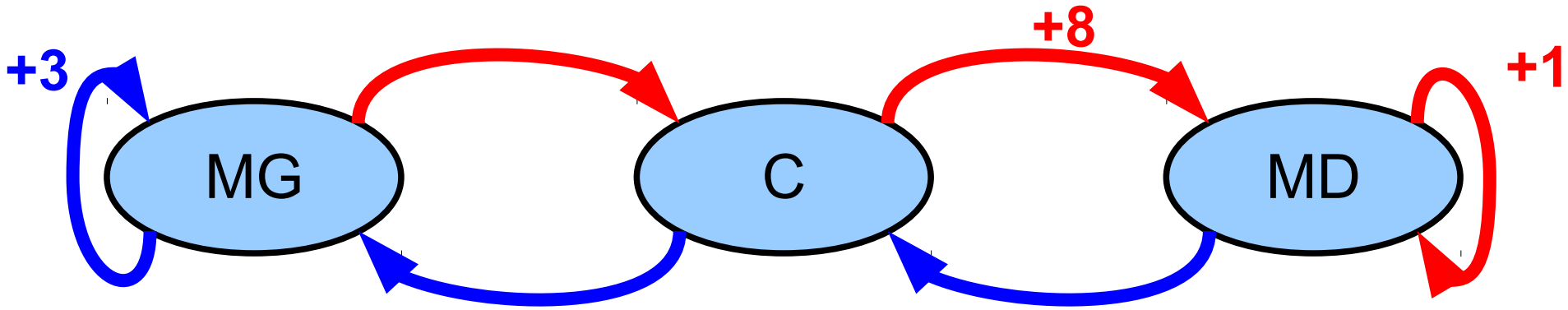
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1	t=2	
MG, G	0	3	5,7	MG
MG, D	0	0	7,2	
C, G	0	0	2,7	C
C, D	0	8	8,9	
MD, G	0	0	7,2	MD
MD, D	0	1	1,9	

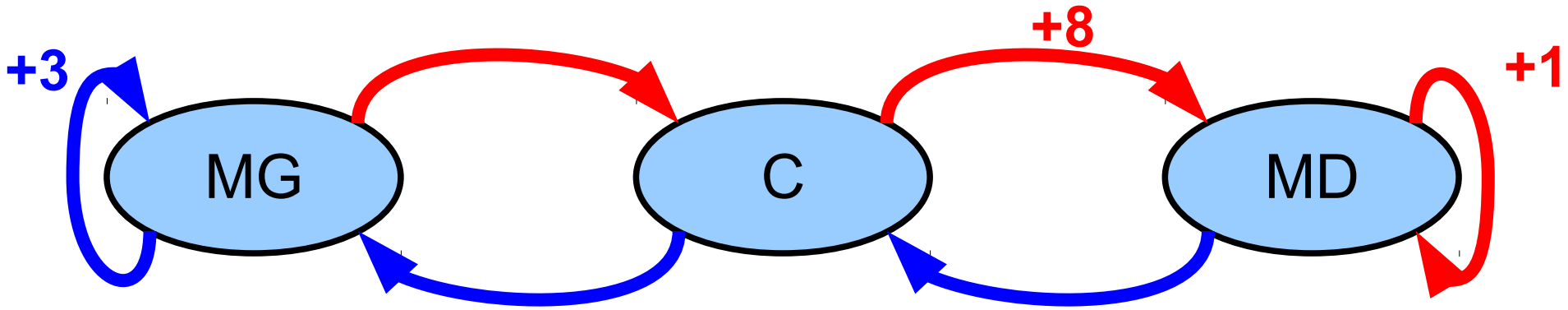
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1	t=2		t=3
MG, G	0	3	5,7	MG	3 + 0,9 * 7,2
MG, D	0	0	7,2		0 + 0,9 * 8,9
C, G	0	0	2,7	C	0 + 0,9 * 7,2
C, D	0	8	8,9		8 + 0,9 * 7,2
MD, G	0	0	7,2	MD	0 + 0,9 * 8,9
MD, D	0	1	1,9		1 + 0,9 * 7,2

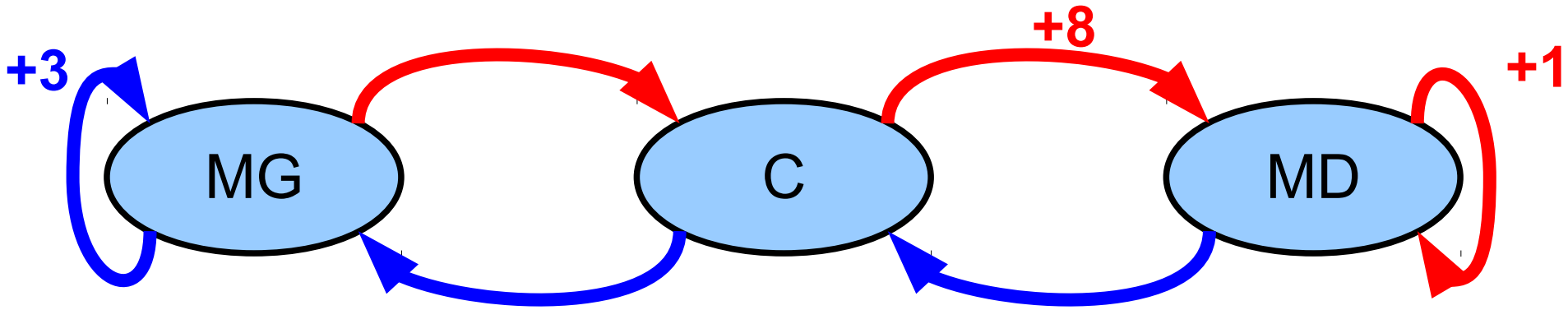
Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1	t=2		t=3
MG, G	0	3	5,7	MG	9,48
MG, D	0	0	7,2		8,01
C, G	0	0	2,7	C	6,48
C, D	0	8	8,9		14,48
MD, G	0	0	7,2	MD	8,01
MD, D	0	1	1,9		7,48

Exemple – chercheur or



$$Q^*(s,a) = R(s,a,T(s,a)) + max_{a'} Q^*(T(s,a),a')$$

	t=0	t=1	t=2	t=3	t=10	t=100
MG, G	0	3	5,7	9,48	23.18..	37.10...
MG, D	0	0	7,2	8,01	24.68..	37.89...
C, G	0	0	2,7	6,48	20.18..	34.10...
C, D	0	8	8,9	14,48	27.81..	42.10...
MD, G	0	0	7,2	8,01	24.68..	37.89...
MD, D	0	1	1,9	7,48	20.81..	35.10...

Plan slides Bonus

- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Observabilité partielle

Propagation dans labyrinthe

- Exemple labyrinthe
  - Appliquer l'algorithme fourni

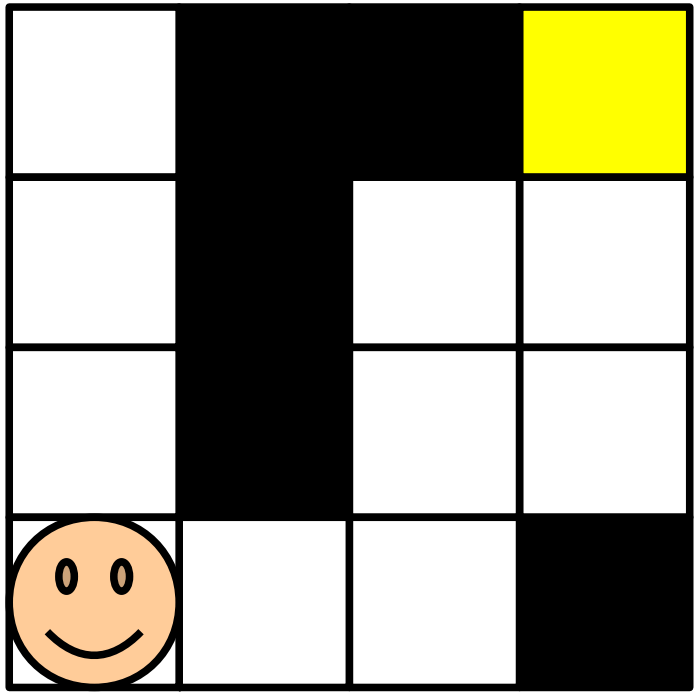
- Montrer application

Propagation dans labyrinthe

- Exemple labyrinthe
  - Appliquer l'algorithme fourni
- Montrer application
- Probleme de taille de l'espace d'état
  - Idée : ne construire Qval que états rencontrés
  - Exemple RaceTracker

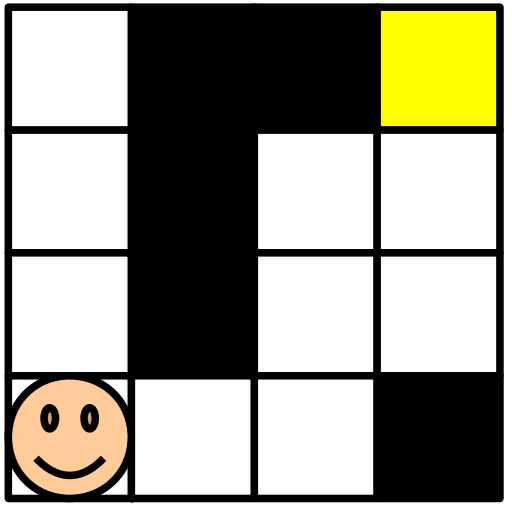
Probleme labyrinthe

- Labyrinthe à traverser avec des trous



Probleme labyrinthe

- Labyrinthe à traverser avec des trous



- Etat : position robot
- Action : N, S, E, O
- Transition :
  - position x action -> position
- Recompense
  - Position == trou => -5
  - Position == arrivee ==> +100
  - Deplacer ==> -1

Probleme labyrinthe

- Etats et actions

```
    def actions(self):
        return(['N','S','E','O'])

     def etats(self):
        etats=[]
        for i in range(0,11):
            for j in range(0,11):
                etats+=[(i,j)]

        #etat final
        etats+=[(-1,-1)]

        return(etats)
```

Probleme labyrinthe

- Transition

```
def transition(self,s,a):
        """fait evoluer d une unite le systeme"""
        x=s[0];
        y=s[1];

        #si on part de 10,10 on retourne en 0
        if (x==10) and (y==10):
            return(-1,-1)
        if (x==-1) and (y==-1):
            return(-1,-1)

        if (a=='N'):
            y=y-1
            if (y<0):
               y=0
        if (a=='S'):
            y=y+1
            if (y>10):
                y=10
        if (a=='E'):
            x=x+1
            if (x>10):
                x=10
        if (a=='O'):
            x=x-1
            if (x<0):
                x=0
        return((x,y))
```

Probleme labyrinthe

- Recompense

```
def recompense(self,s,a,sarr):

        """ok si on est en 10,10"""
        if ((sarr[0]==10)and(sarr[1]==10)):
            return(100)

        if (sarr in self.trou):
            return(-5)

        if (sarr[0]==-1)and (sarr[1]==-1):
            return(0)

        return(-1)
```

Probleme labyrinthe

- Recompense

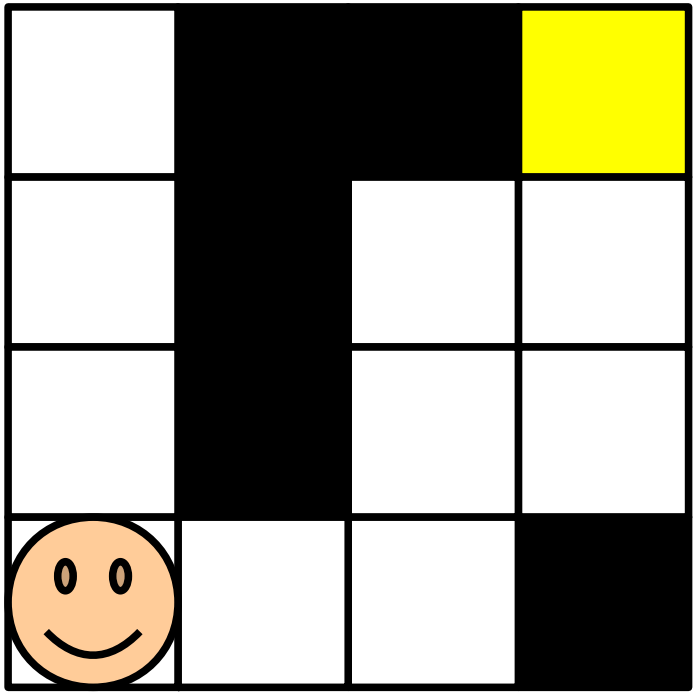
```
def recompense(self,s,a,sarr):

        """ok si on est en 10,10"""
        if ((sarr[0]==10)and(sarr[1]==10)):
            return(100)

        if (
```

Cf code sur site

Probleme du labyrinthe

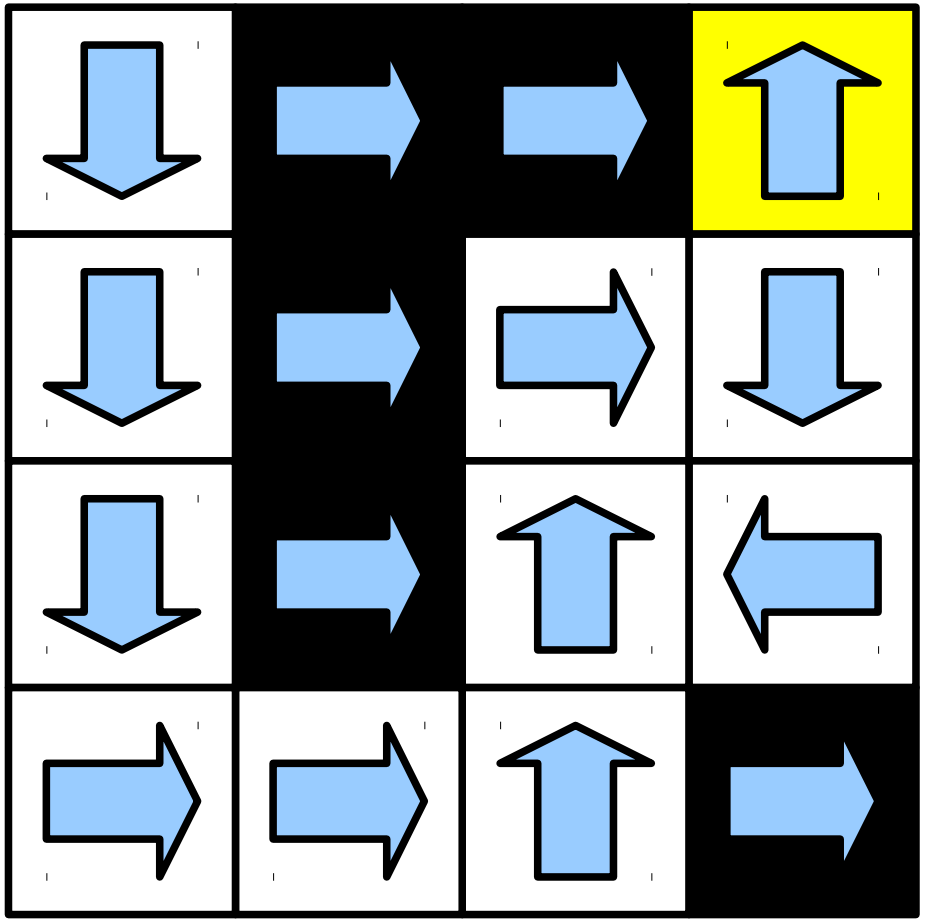
- Exemple de solution ?

- Politique : S -> A
  - Générateur de politiques



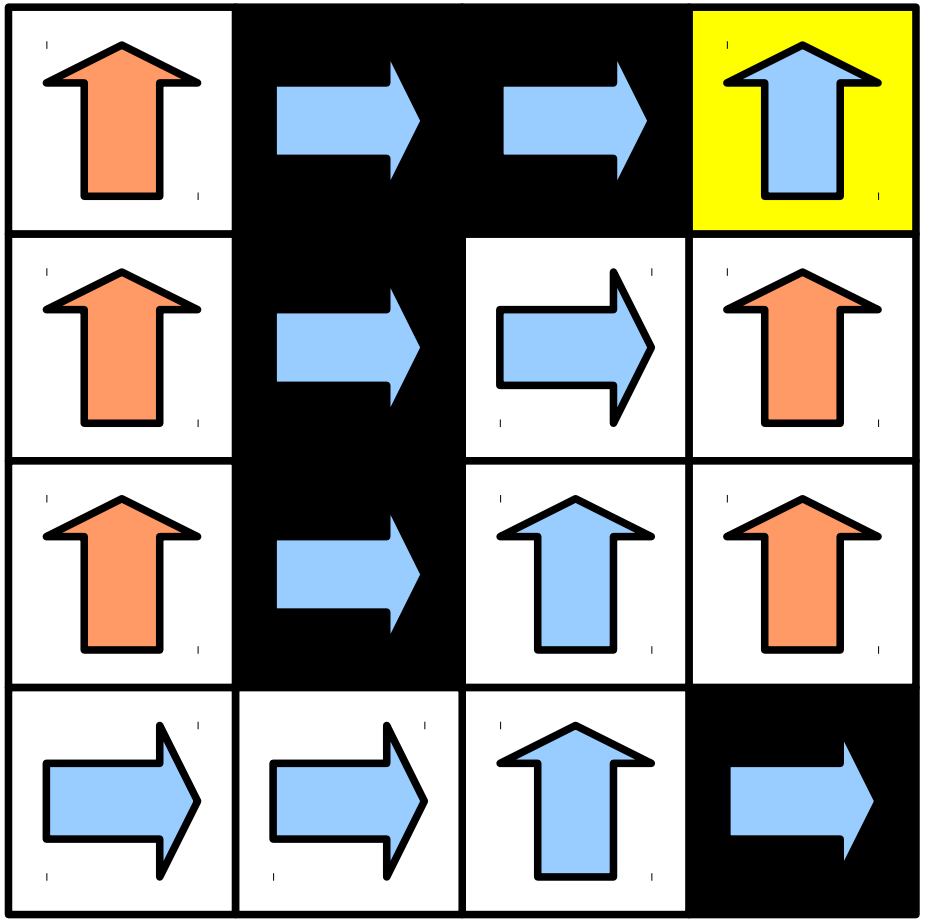
Probleme du labyrinthe

- Exemple 1 de solution ?



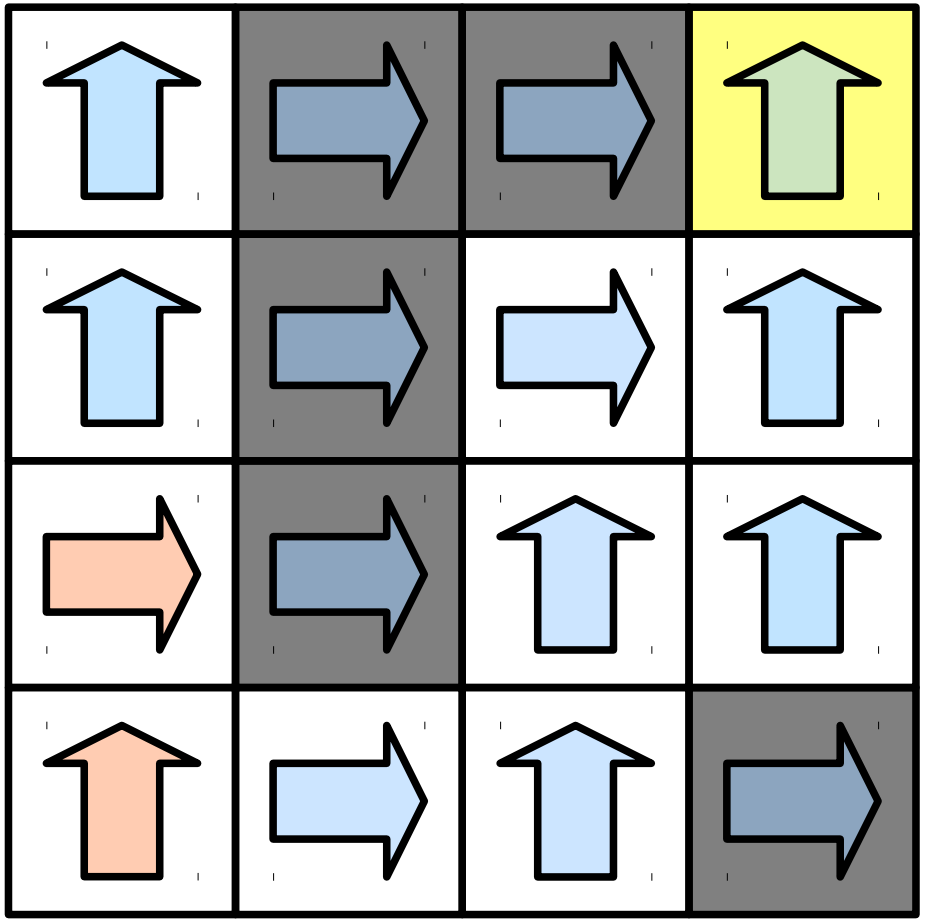
Probleme du labyrinthe

- Autre Exemple de solution ?

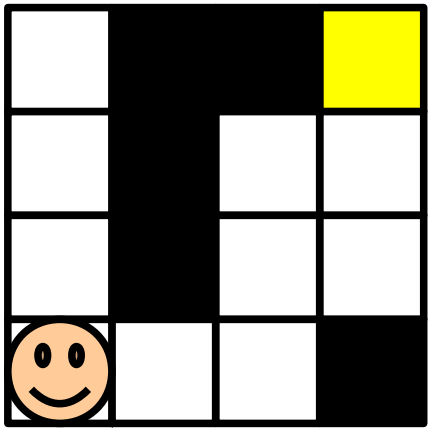


Probleme du labyrinthe

- Autre Exemple de solution ?

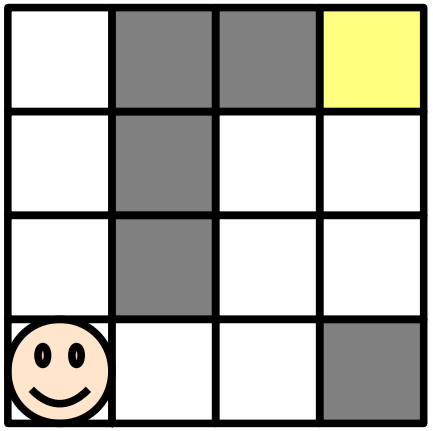


Probleme du labyrinthe



- Enumerer les solutions
- Combien de solutions différentes ?

Probleme du labyrinthe



- Enumerer les solutions
- Combien de solutions différentes ?

- Nombre de solutions $|A|^{|S|}$
  - Ici => 4^16 = 4.294.967.296
  - Énumération impossible

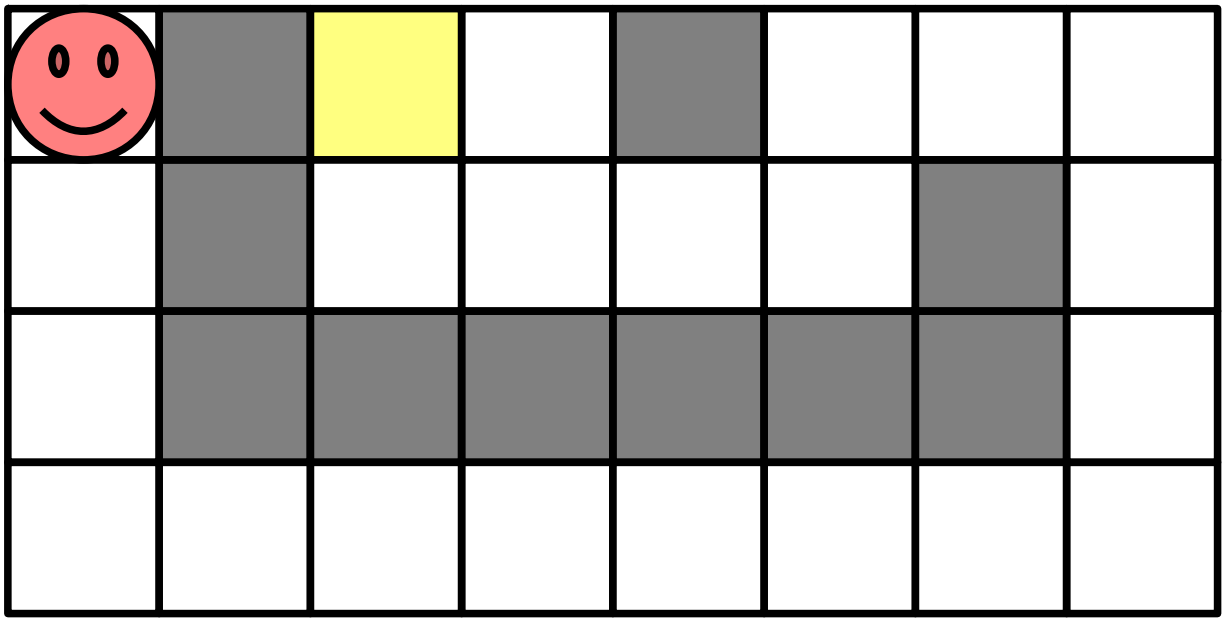
Probleme du labyrinthe

- Evaluer une politique
  - Exactement meme code qu'avant

- Meilleure politique dépend
  - Compromis entre récompense trou (-5)
  - Distance parcourue (-1 case)
  - Récompense objectif (100)

Probleme du labyrinthe

- Meilleure politique dépend
  - Compromis entre récompense trou (-5)
  - Distance parcourue (-1 case)
  - Récompense objectif (100)



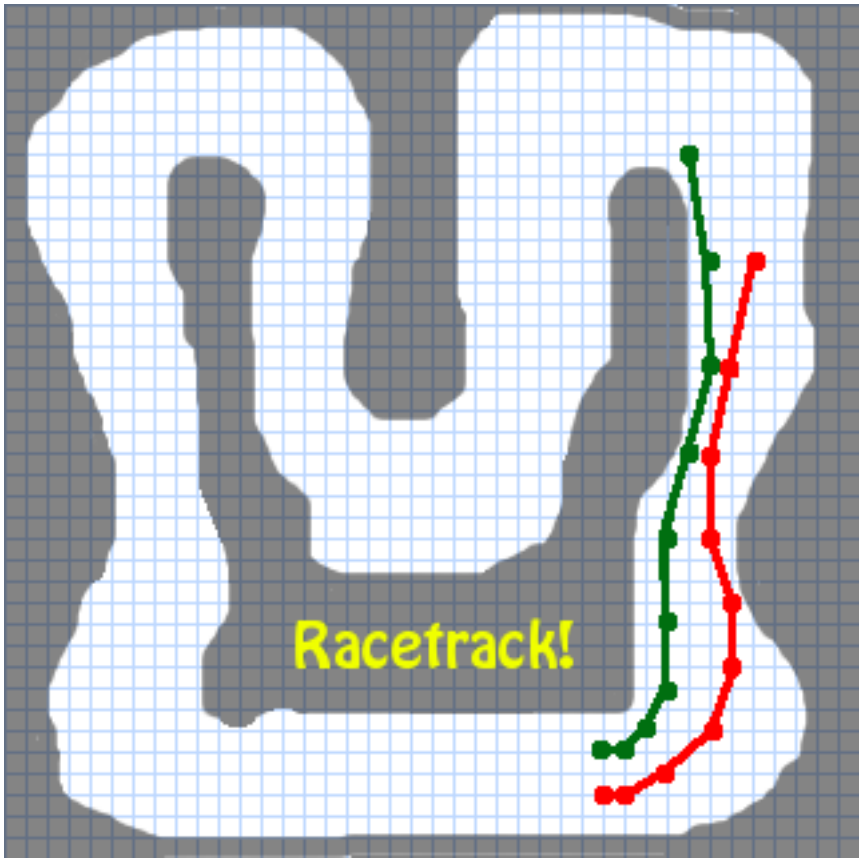
Plan slides Bonus

- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Observabilité partielle

Exemple - Racetracker

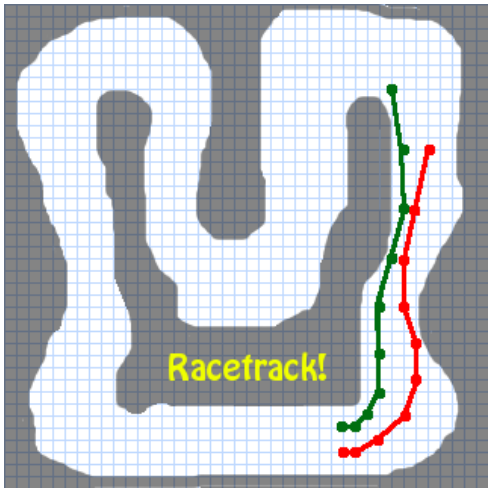
Modélisation de problemes

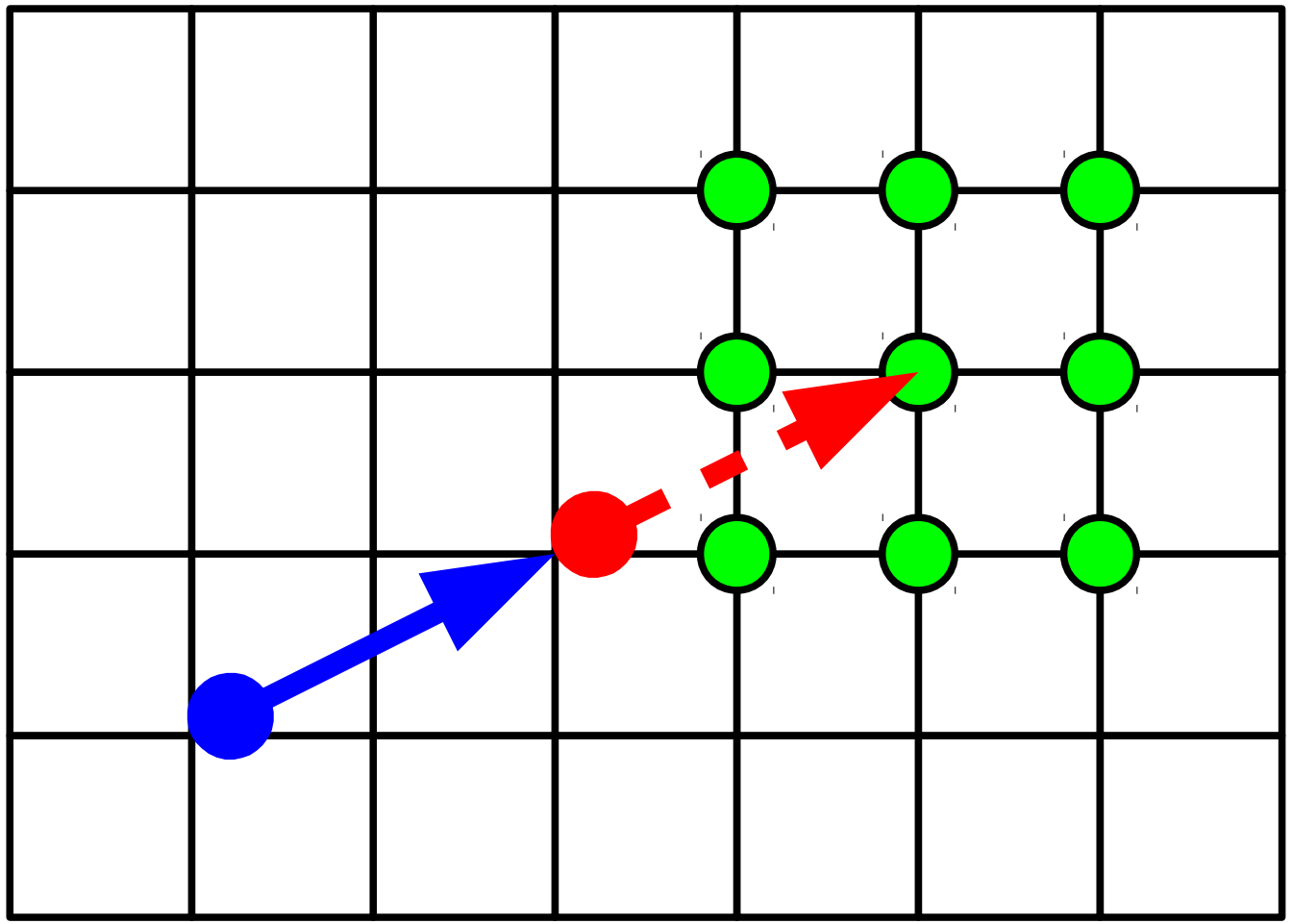
- Probleme RaceTrack (wikipedia)



Modélisation de problemes

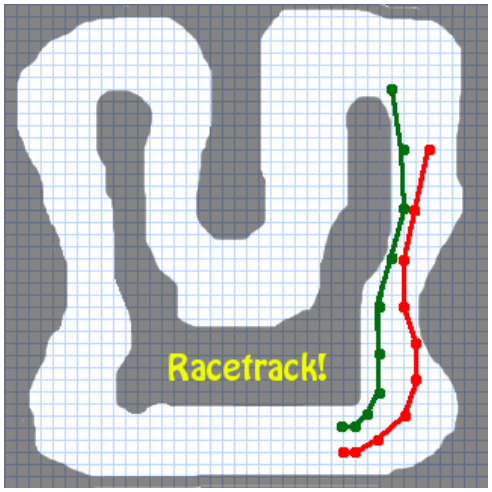
- Probleme RaceTrack

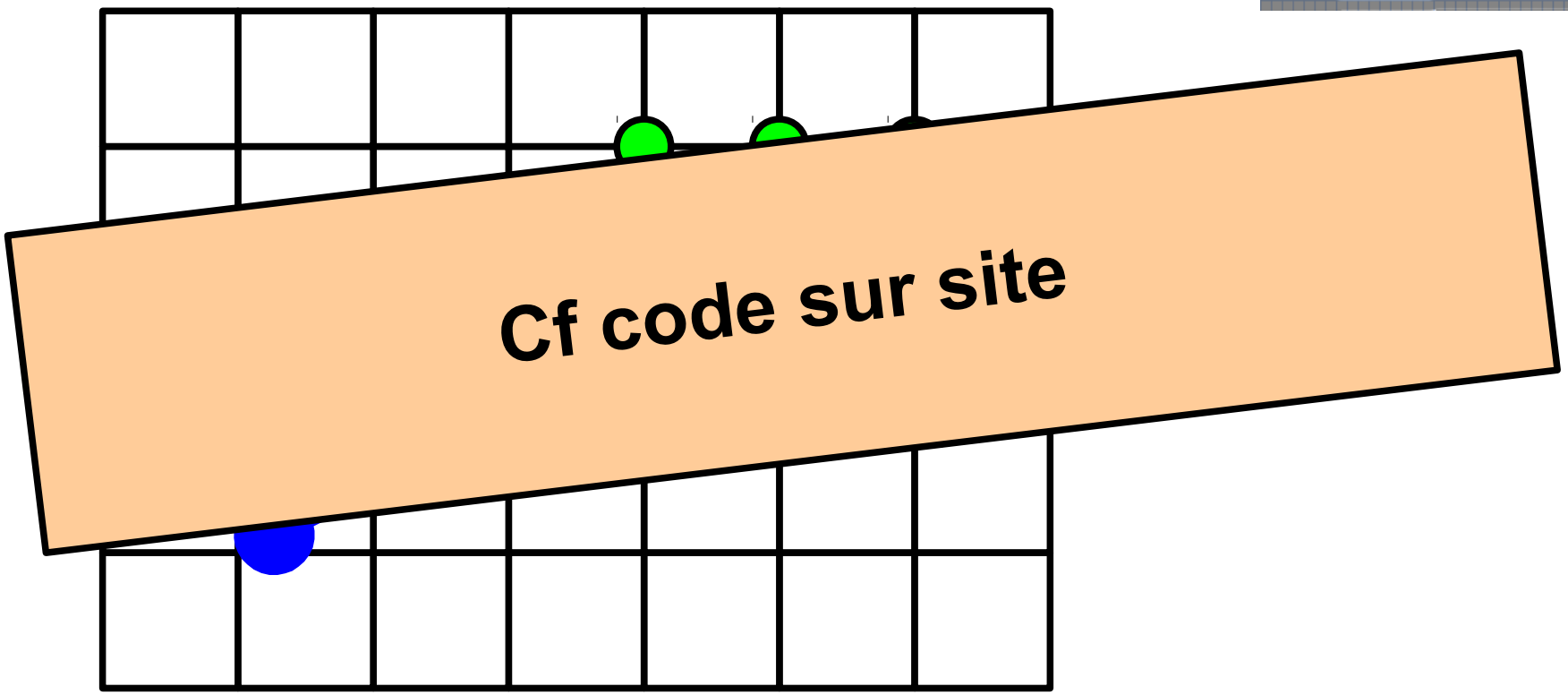




Modélisation de problemes

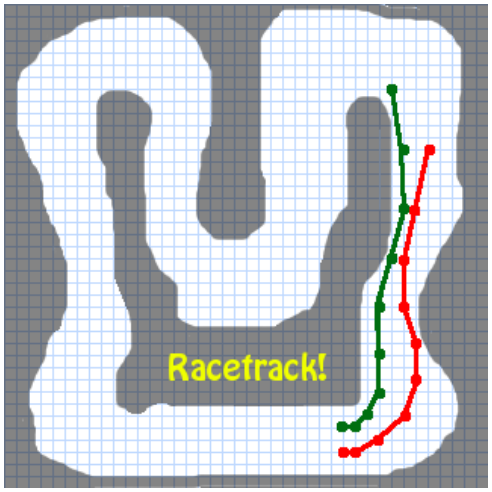
- Probleme RaceTrack
  - Etat ?
  - Action ?

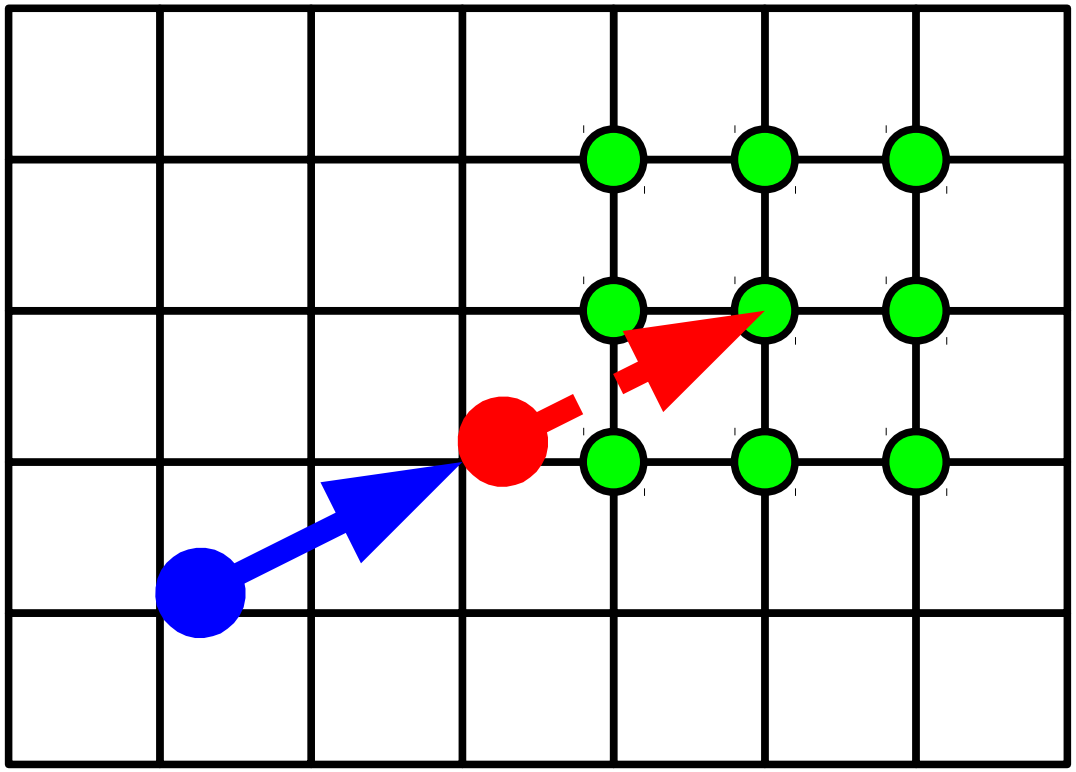




Modélisation de problemes

- Probleme RaceTrack
  - Etat ?
  - Action ?



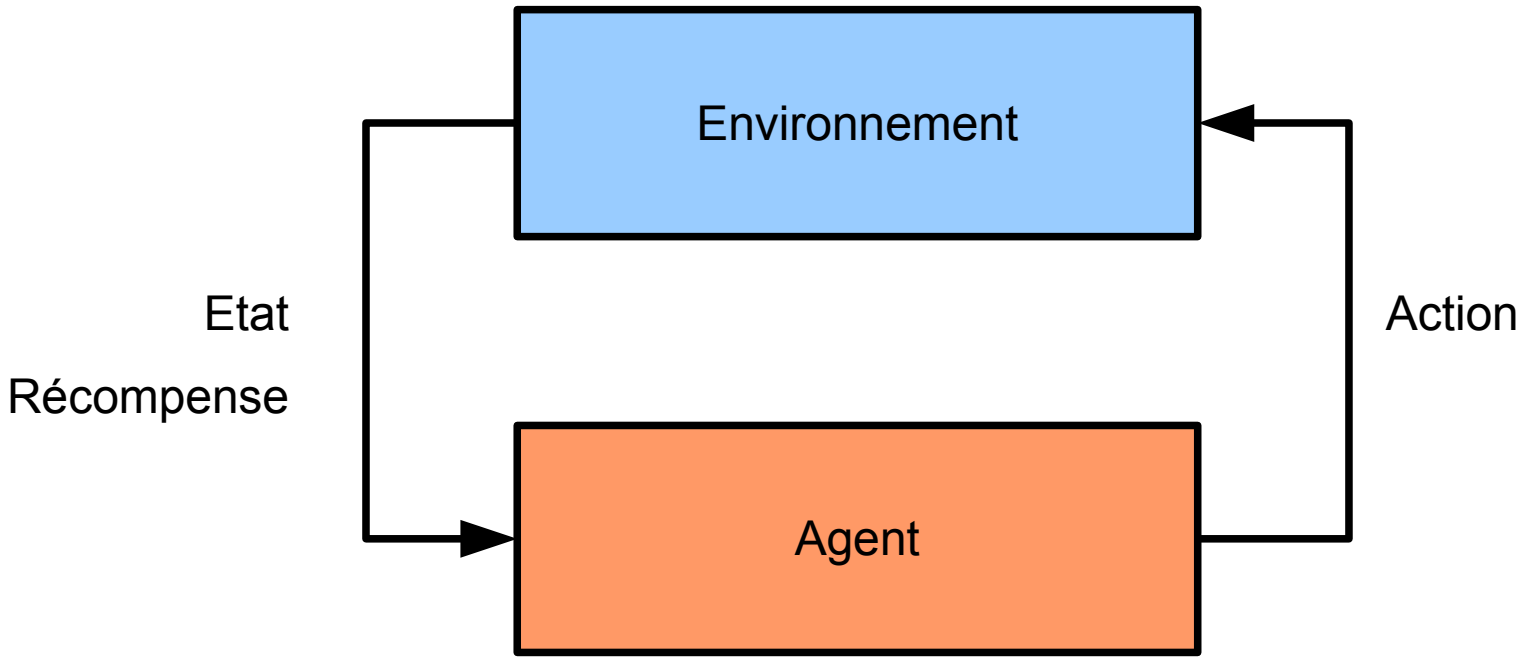


Plan slides Bonus

- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Observabilité partielle

Apprentissage par renforcement

- Apprentissage par renforcement
  - Probleme de décision
  - Monde inconnu (T, R), mais experiences possibles



Apprentissage par renforcement

- Apprentissage par renforcement
  - Probleme Monde inconnu
  - Connait pas T, R

- A chaque pas de temps
  - (s, a) -> s' et r
  - Mettre à jour Q(s,a) à partir informations

- Exemple
  - (C, gauche) => (MG, 0) ; (MG, D) => (C, 0) ; ...

Apprentissage par renforcement

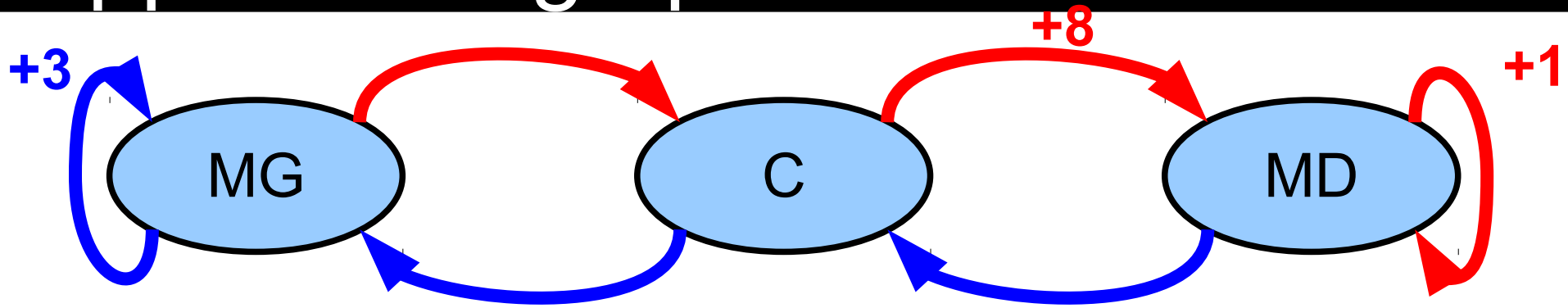
- Objectif:
  - Apprendre au fur et à mesure à partir Xp

- Principe Q-learning :
  - Apprendre directement politique (AR direct)
  - Pour tuple s, a => s', r
  - On applique équation Bellman

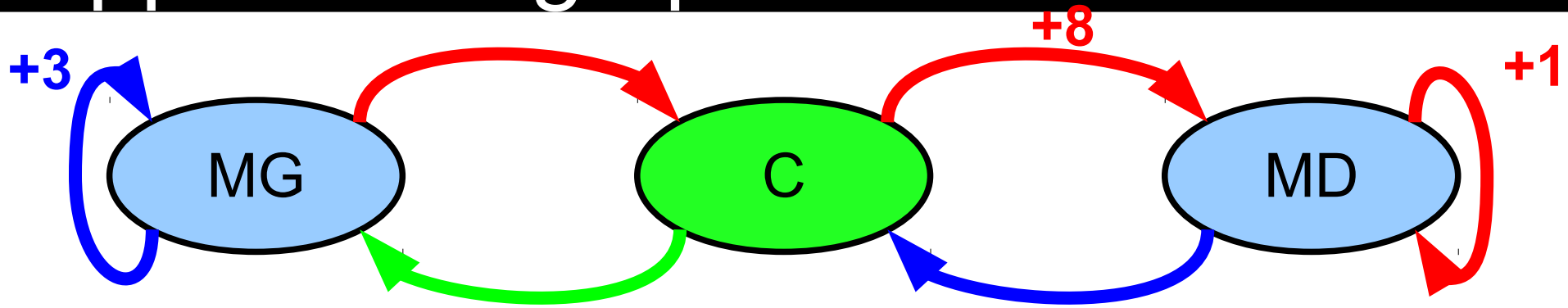
$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma \, max_{a'} Q^*(T(s,a),a')$$

$$Q^*(s,a) = r + \gamma \, max_{a'} Q^*(s',a')$$

Apprentissage par renforcement



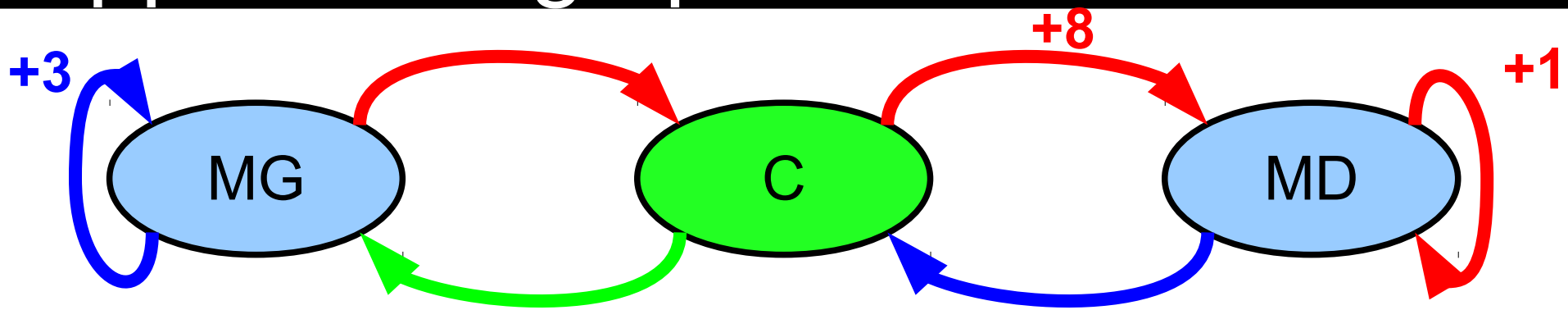
Apprentissage par renforcement



- C, Gauche ==> MG , +0

$$Q^*(C,G) = r + \gamma \, max_{a'} Q^*(MG, a')$$

Apprentissage par renforcement

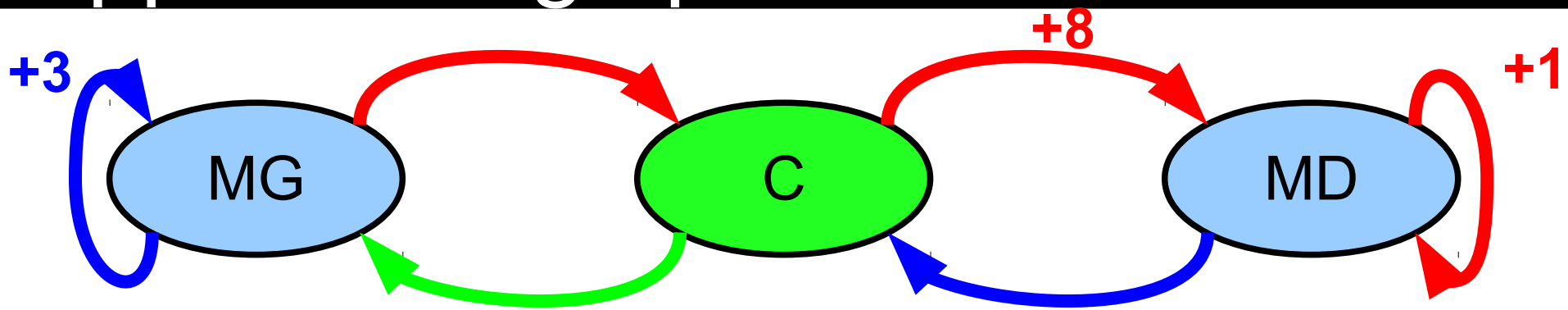


- C, Gauche ==> MG , +0

$$Q^*(C,G) = r + \gamma \, max_{a'} Q^*(MG,a')$$

	t=0	t=1
MG, G	0	0
MG, D	0	0
C, G	0	0
C, D	0	0
MD, G	0	0
MD, D	0	0

Apprentissage par renforcement

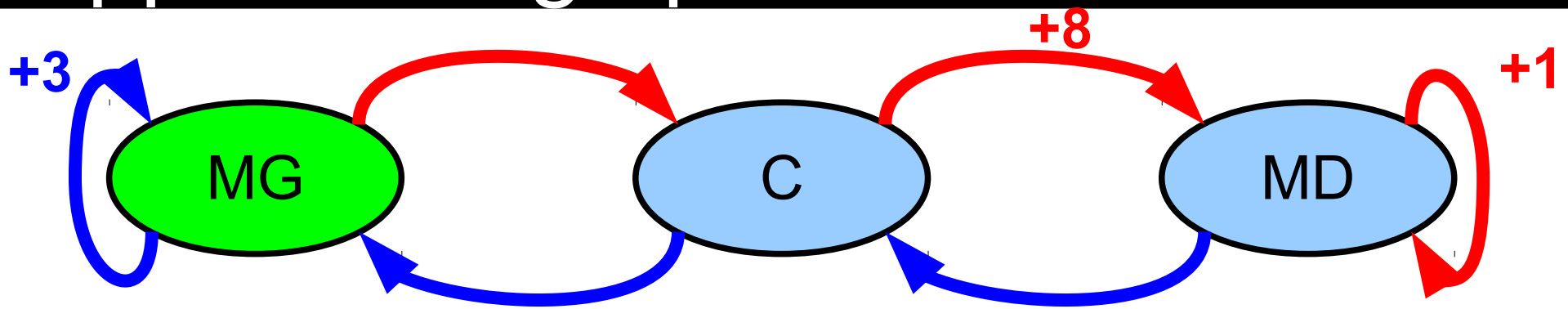


- C, Gauche ==> MG , +0

$$Q^*(C,G) = r + \gamma \, max_{a'} Q^*(MG,a')$$

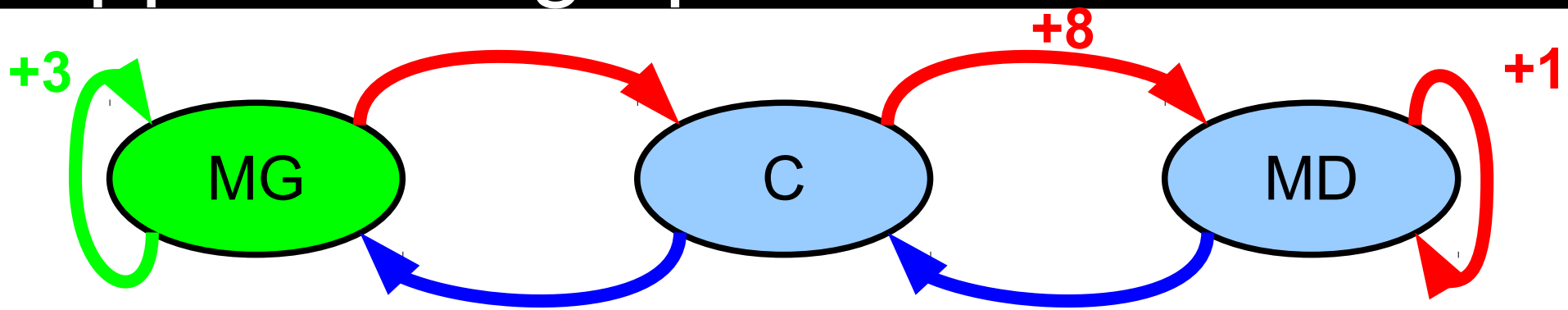
	t=0	t=1
MG, G	0	0
MG, D	0	0
C, G	0	0
C, D	0	0
MD, G	0	0
MD, D	0	0

Apprentissage par renforcement



	t=0	t=1
MG, G	0	0
MG, D	0	0
C, G	0	0
C, D	0	0
MD, G	0	0
MD, D	0	0

Apprentissage par renforcement

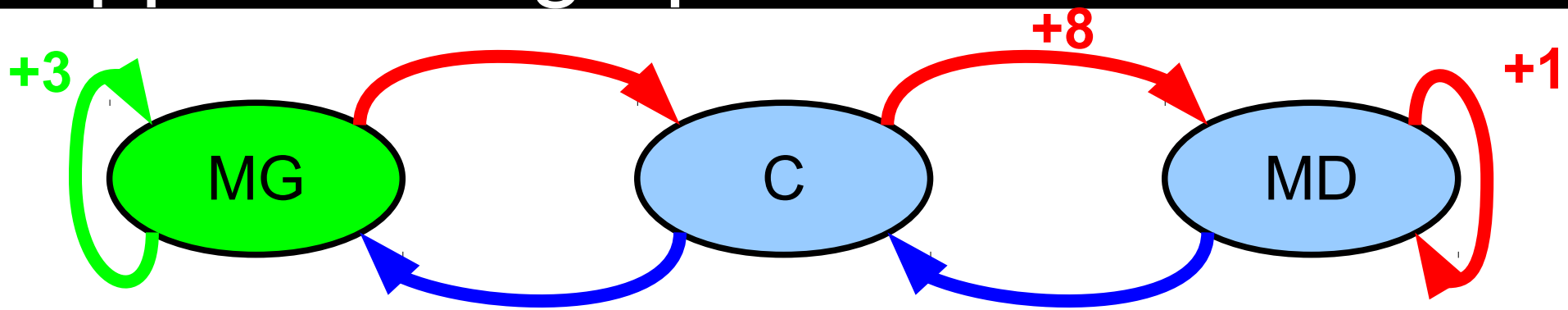


- MG, Gauche ==> MG , +3

$$Q^*(MG,G)=3+\gamma\ max_{a'}Q^*(MG,a')$$

	t=0	t=1	t=2
MG, G	0	0	XX
MG, D	0	0	0
C, G	0	0	0
C, D	0	0	0
MD, G	0	0	0
MD, D	0	0	0

Apprentissage par renforcement

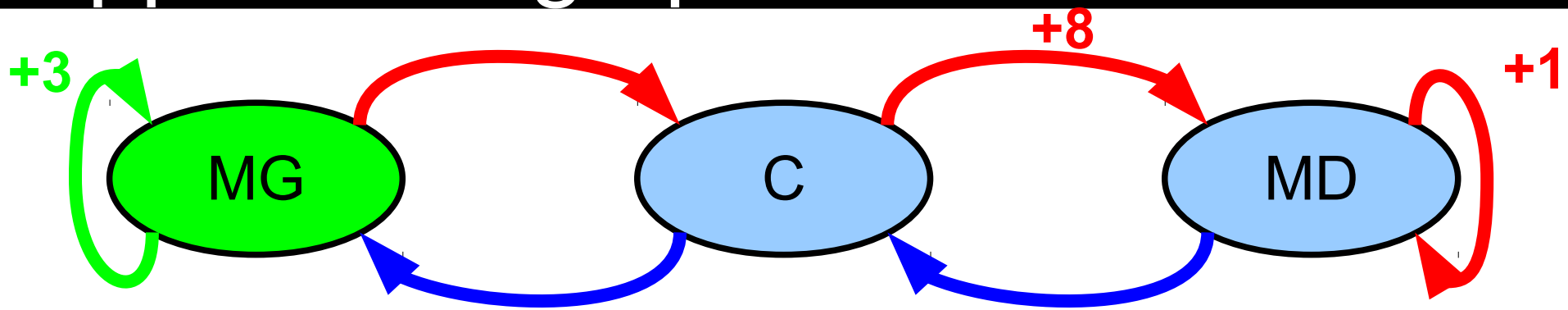


- MG, Gauche ==> MG , +3

$$Q^*(MG,G)=3+\gamma\ max_{a'}Q^*(MG,a')$$

	t=0	t=1	t=2
MG, G	0	0	3
MG, D	0	0	0
C, G	0	0	0
C, D	0	0	0
MD, G	0	0	0
MD, D	0	0	0

Apprentissage par renforcement

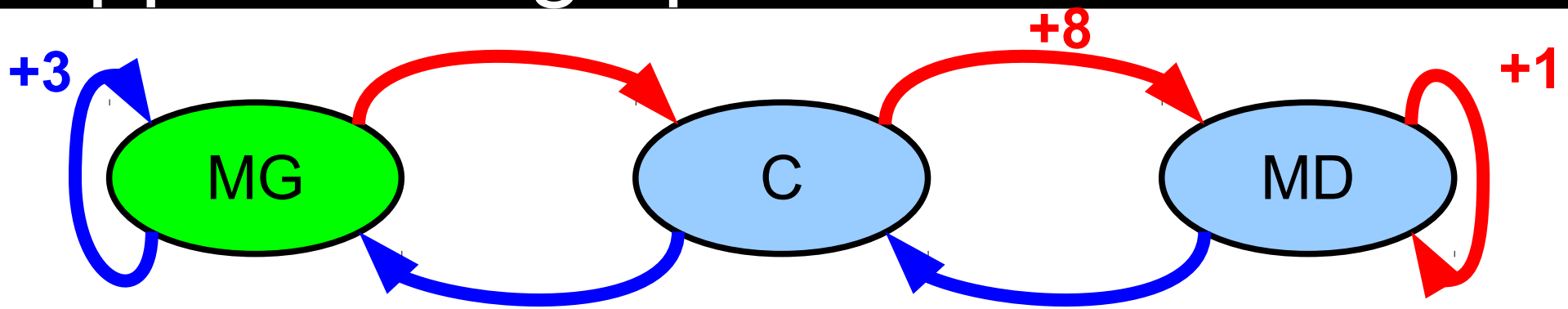


- MG, Gauche ==> MG , +3

$$Q^*(MG,G)=3+\gamma\ max_{a'}Q^*(MG,a')$$

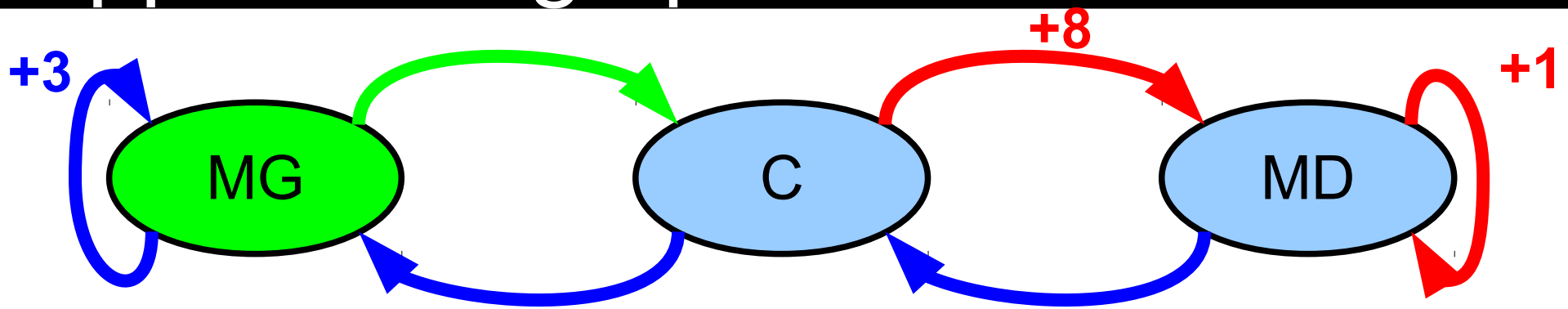
	t=0	t=1	t=2
MG, G	0	0	3
MG, D	0	0	0
C, G	0	0	0
C, D	0	0	0
MD, G	0	0	0
MD, D	0	0	0

Apprentissage par renforcement



	t=0	t=1	t=2
MG, G	0	0	3
MG, D	0	0	0
C, G	0	0	0
C, D	0	0	0
MD, G	0	0	0
MD, D	0	0	0

Apprentissage par renforcement

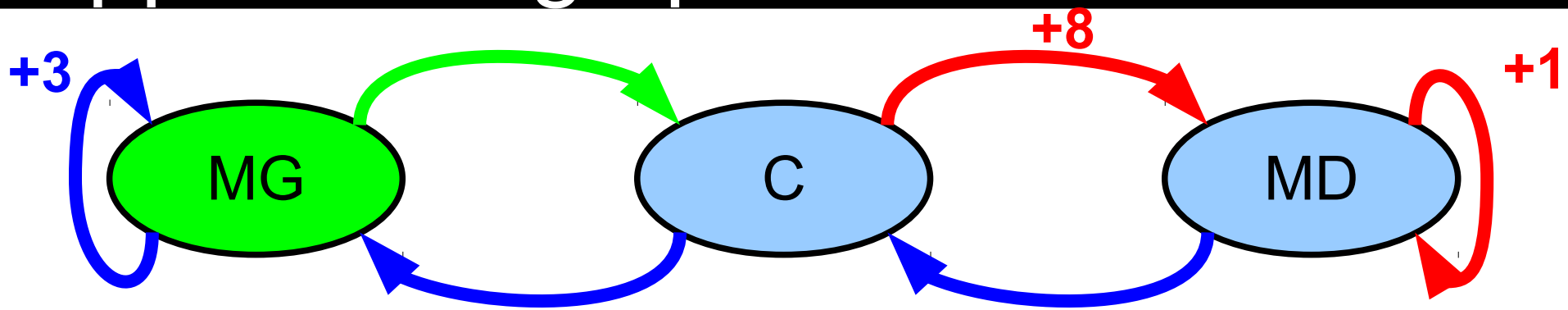


- MG, Droite ==> C , +0

$$Q^*(MG, D) = 0 + \gamma \, max_{a'} Q^*(C, a')$$

	t=0	t=1	t=2	t=3
MG, G	0	0	3	3
MG, D	0	0	0	XX
C, G	0	0	0	0
C, D	0	0	0	0
MD, G	0	0	0	0
MD, D	0	0	0	0

Apprentissage par renforcement

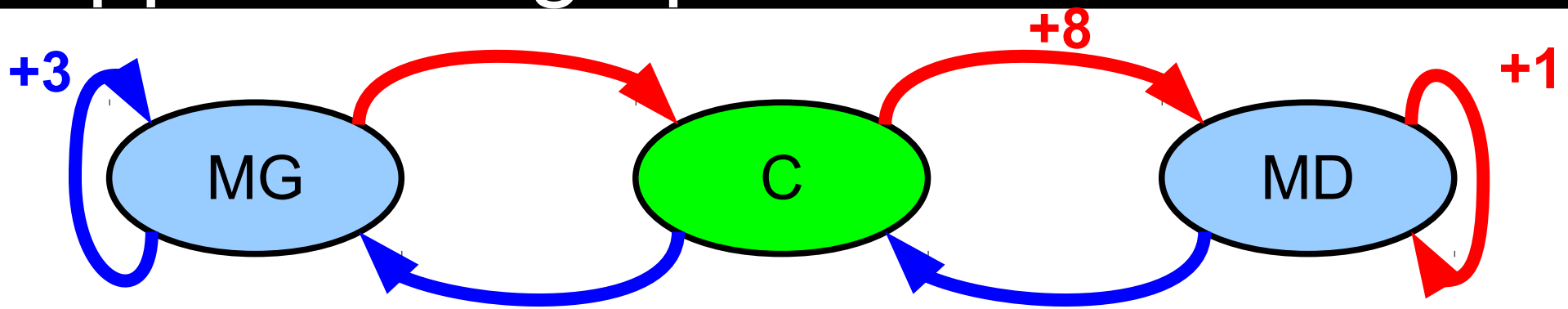


- MG, Droite ==> C , +0

$$Q^*(MG, D) = 0 + \gamma \, max_{a'} Q^*(C, a')$$

	t=0	t=1	t=2	t=3
MG, G	0	0	3	3
MG, D	0	0	0	0
C, G	0	0	0	0
C, D	0	0	0	0
MD, G	0	0	0	0
MD, D	0	0	0	0

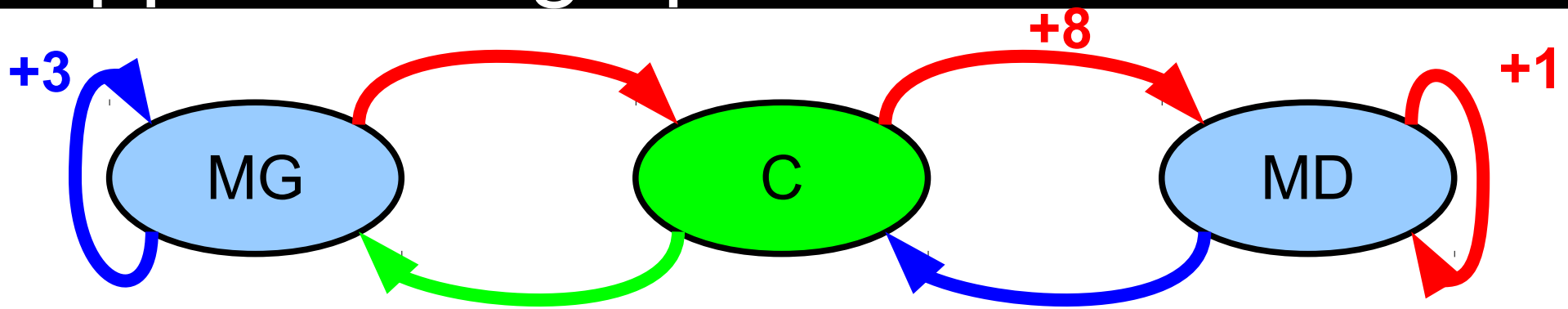
Apprentissage par renforcement



- C

	t=0	t=1	t=2	t=3
MG, G	0	0	3	3
MG, D	0	0	0	0
C, G	0	0	0	0
C, D	0	0	0	0
MD, G	0	0	0	0
MD, D	0	0	0	0

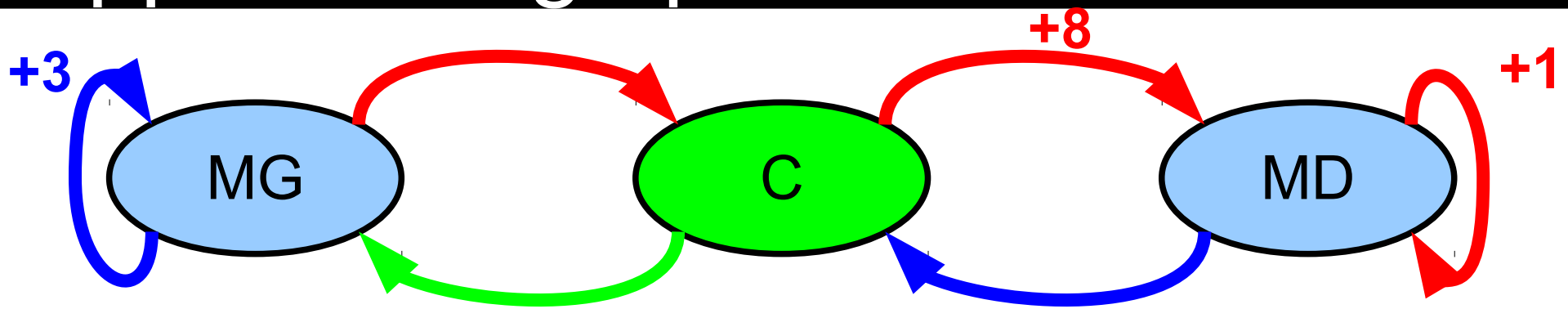
Apprentissage par renforcement



- C, Gauche => MG, 0

	t=0	t=1	t=2	t=3	t=4
MG, G	0	0	3	3	3
MG, D	0	0	0	0	0
C, G	0	0	0	0	XXX
C, D	0	0	0	0	0
MD, G	0	0	0	0	0
MD, D	0	0	0	0	0

Apprentissage par renforcement



- C, Gauche => MG, 0

$$Q^*(C,G) = 0 + \gamma \, max_{a'} Q^*(MG, a')$$

	t=0	t=1	t=2	t=3
MG, G	0	0	3	3
MG, D	0	0	0	0
C, G	0	0	0	0
C, D	0	0	0	0
MD, G	0	0	0	0
MD, D	0	0	0	0

t=4
3
0
2,7
0
0
0

Apprentissage par renforcement

- Dilemme exploration / exploitation

- Exploration
  - Je me déplace au hasard
  - Mais récompenses faibles

- Exploitation
  - Je me déplace au mieux de mes connaissances
  - MAIS ....

Apprentissage par renforcement

- Dilemme exploration / exploitation

	t=0	t=1	t=2	t=3	t=4
MG, G	0	0	3	3	3
MG, D	0	0	0	0	0
C, G	0	0	0	0	2,7
C, D	0	0	0	0	0
MD, G	0	0	0	0	0
MD, D	0	0	0	0	0

- Table me dit que mieux C ==> G
  - Mais je n'apprends pas à droite (qui est mieux)

Apprentissage par renforcement

- Ecrire code python

```
 def QLearning(self,Sdep,nPas,nEpisode,affiche):
        # repeter n episode
            # on reinitiliase au depart
            # pour chaque iteration de l'episode
                # miseAJour Qvaleur
                # part de l'etat d'arrivee
```

```
def majQLearning(self,s,Q):
        '''Permet de faire une mise a jour du Qlearning'''

        # on choisit une action au hasard
        # on execute l'action (s et rec)
        # cherche max arrivee

        # on met a jour Qvaleur
        Q[(s,a)]=rec+gamma*max

        #on retourne le nouvel etat
```

Plan slides Bonus

- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Stochastique
- Observabilité partielle

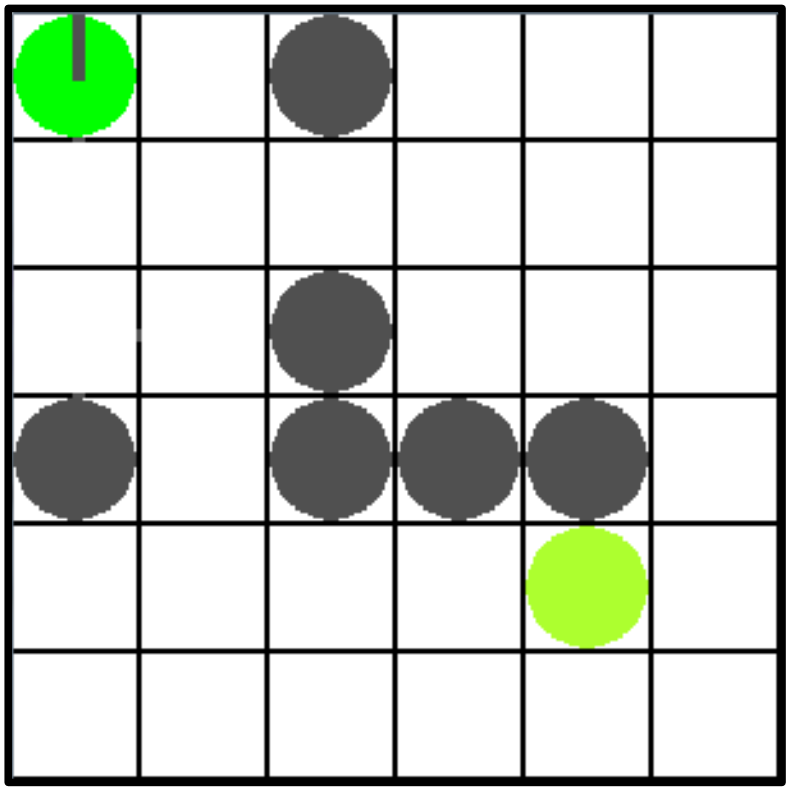
Monde stochatique

- MDP avec des probabilités
  - Transition S x A -> P(S)
- Pourquoi ?
  - Modélise phenomène continu / inconnu
  - Monde probabiliste
- Exemple

Déplacement avec glissements

- Labyrinthe identique
  - Mais quand on se déplace proba d'avancer 2 fois

- Gestion risque
  - Quelle politique ?



Equation bellman

- Equation de bellman légérement modifiée
  - Calcul d'esperance

- Monde deterministe

$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma \, max_{a'} Q^*(T(s,a),a')$$

Equation bellman

- Equation de bellman légérement modifiée
  - Calcul d'esperance

- Monde deterministe

$$Q^*(s,a) = R(s,a,T(s,a)) + \gamma \, max_{a'} Q^*(T(s,a),a')$$

- Monde stochastique

$$Q^*(s,a) = \sum_{s'} P(s,a,s')\left(R(s,a,s') + \gamma \, max_{a'} Q^*(s',a')\right)$$

Value itération

- Meme principe que précédent

- Repeter pour t
  - Chaque état et chaque action

$$Q^*(s,a)=\sum_{s'} P(s,a,s')\left(R(s,a,s')+\gamma\ max_{a'}Q^*(s',a')\right)$$

Apprentissage par renforcement

- Meme principe que précédent
  - MAIS ...

Apprentissage par renforcement

- Meme principe que précédent
  - MAIS ...
  - Une expérience ne suffit plus

Apprentissage par renforcement

- Meme principe que précédent
  - MAIS ...
  - Une expérience ne suffit plus

- Principe
  - Faire des statistiques sur les résultats obtenus

$$Q^*(s,a) = \alpha\,[r + \gamma\, max_{a'} Q^*(s',a')] + (1-\alpha)\,Q^*(s,a)$$

$$\alpha = 1/t$$

Plan slides Bonus

- Autres exemples
  - Chercheur d'or + deroulé Value iteration
  - Labyrinthe
  - RaceTracker
- Apprentissage par renforcement
- Stochastique
- Observabilité partielle

Problemes difficiles

- Observabilité partielle
- Mondes continus
- Taille espace d'états