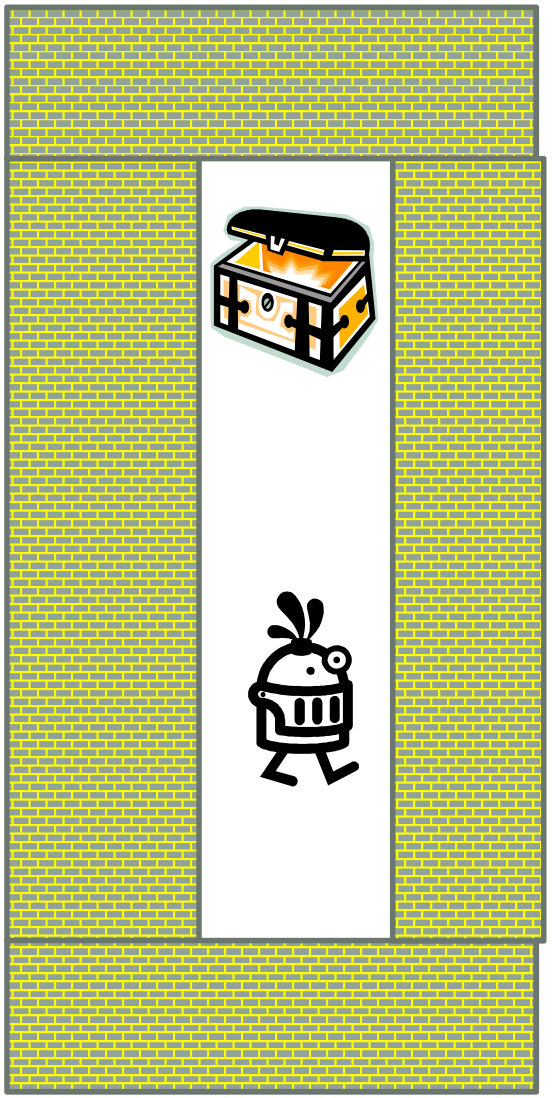
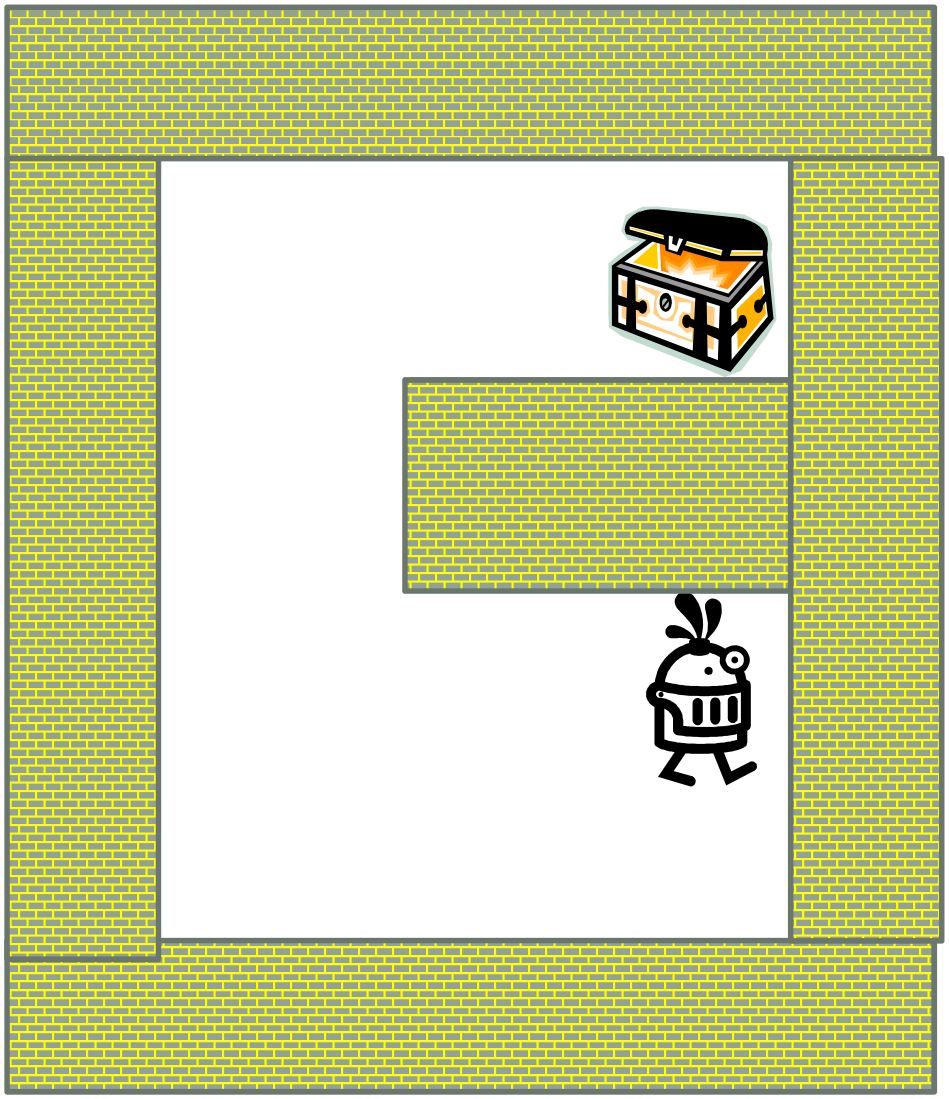
MODÈLES D'AGENT

Cours 2

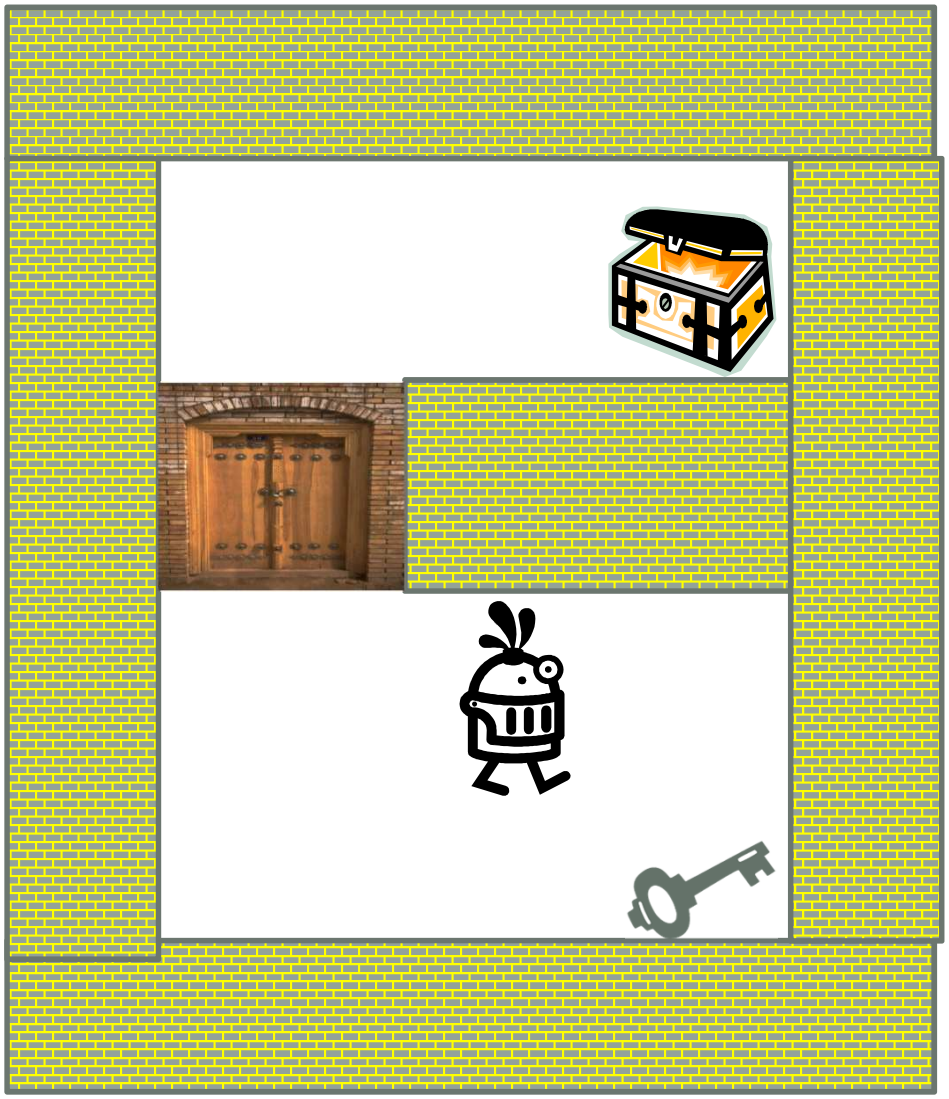
Comment programmer l'agent?



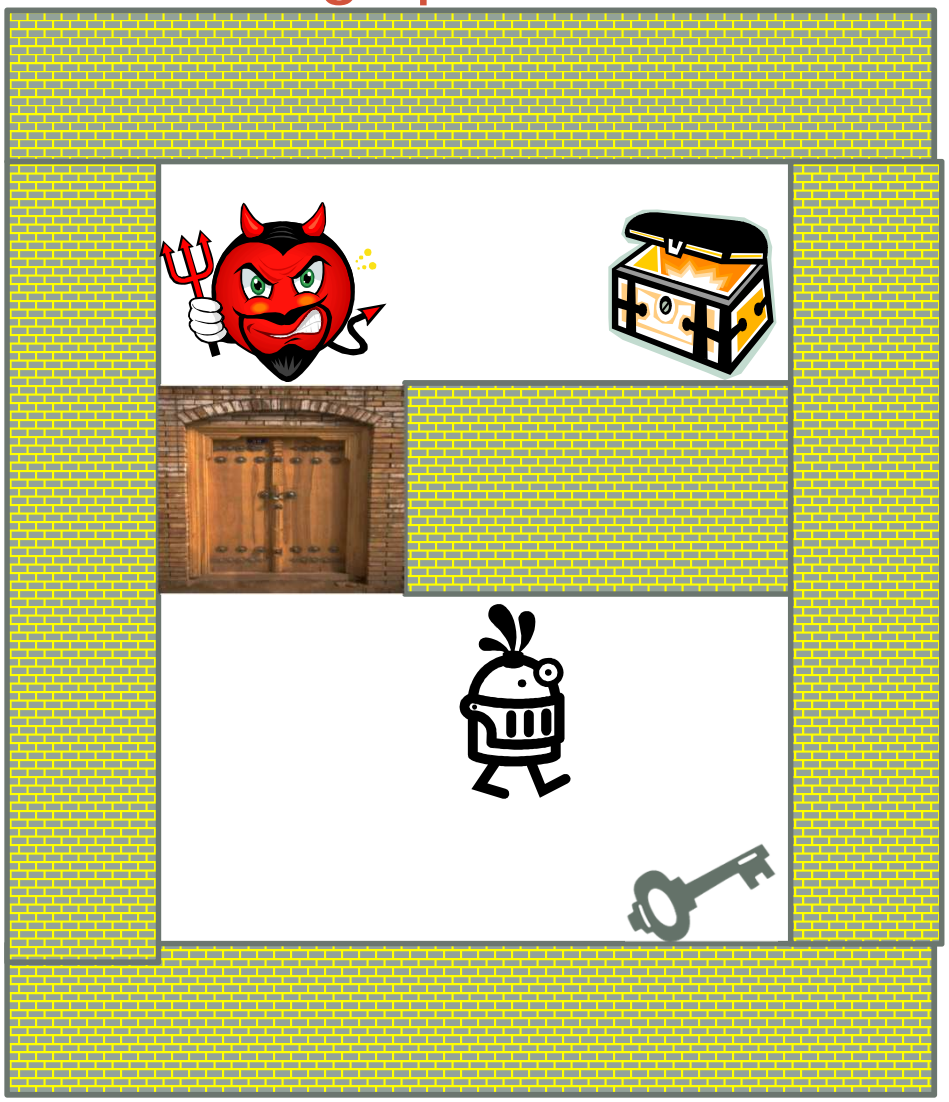
Première solution: fait n'importe quoi, ça finira bien par marcher…



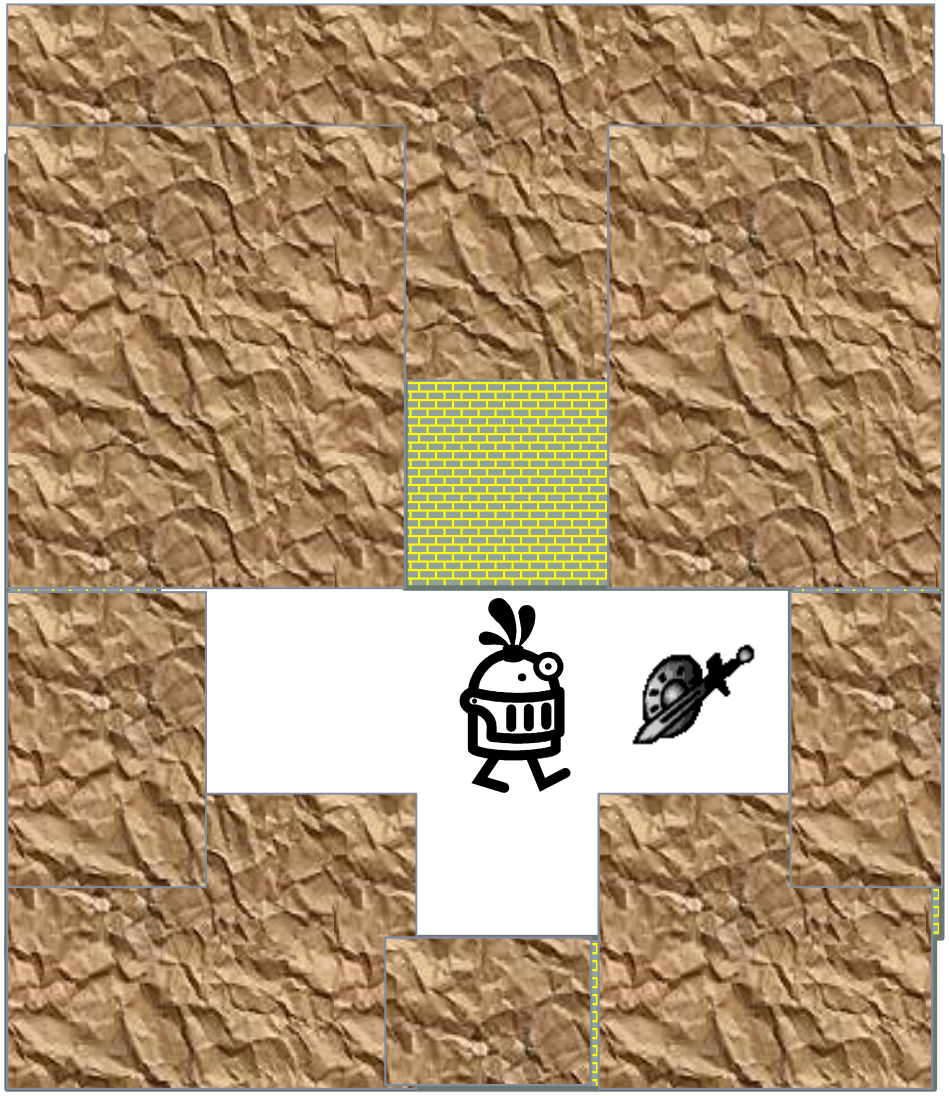
Deuxième solution: commence par longer les murs, prendre a chaque fois que c'est possible, et essaye de tout ouvrir



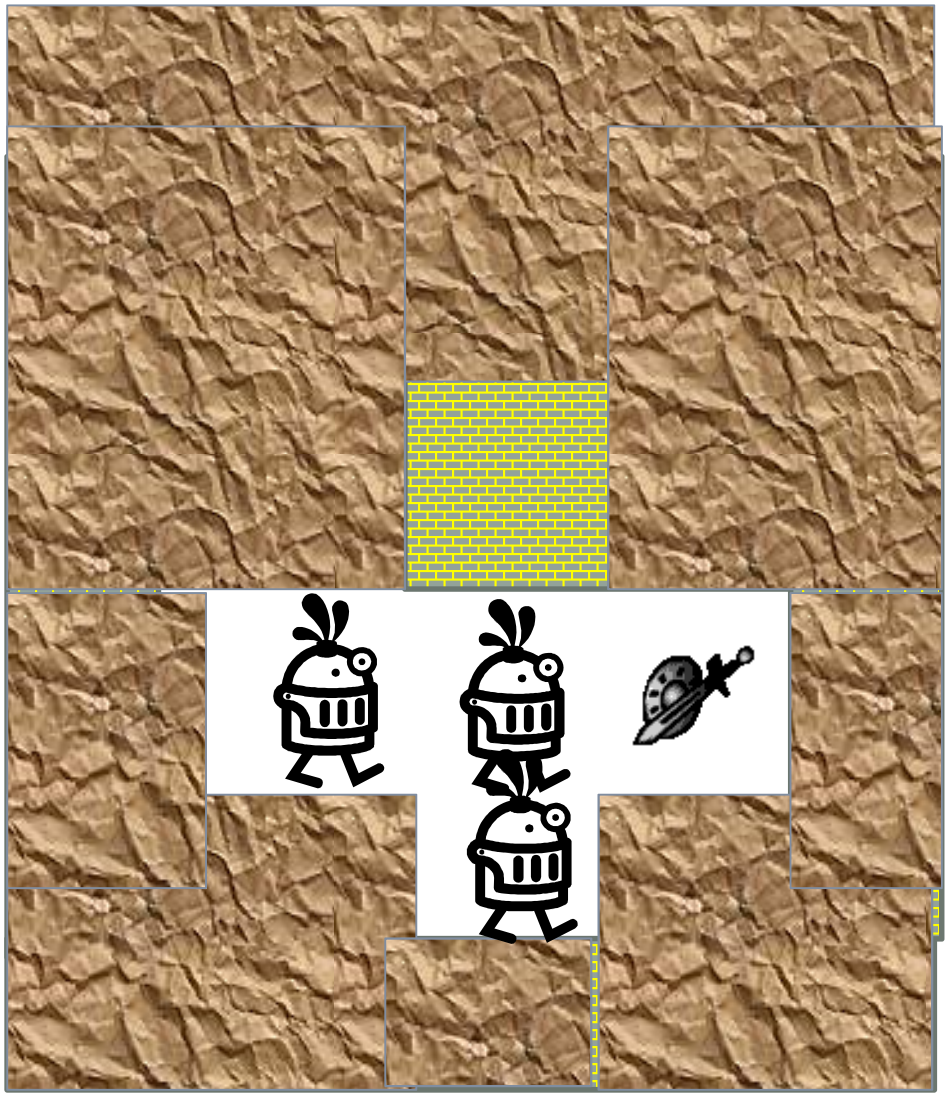
Troisième solution: lorsqu'un héros rencontre un mur, le héros tourne. Lorsque c'est une clé, il la prend. Lorsque c'est un monstre, dommage pour lui…



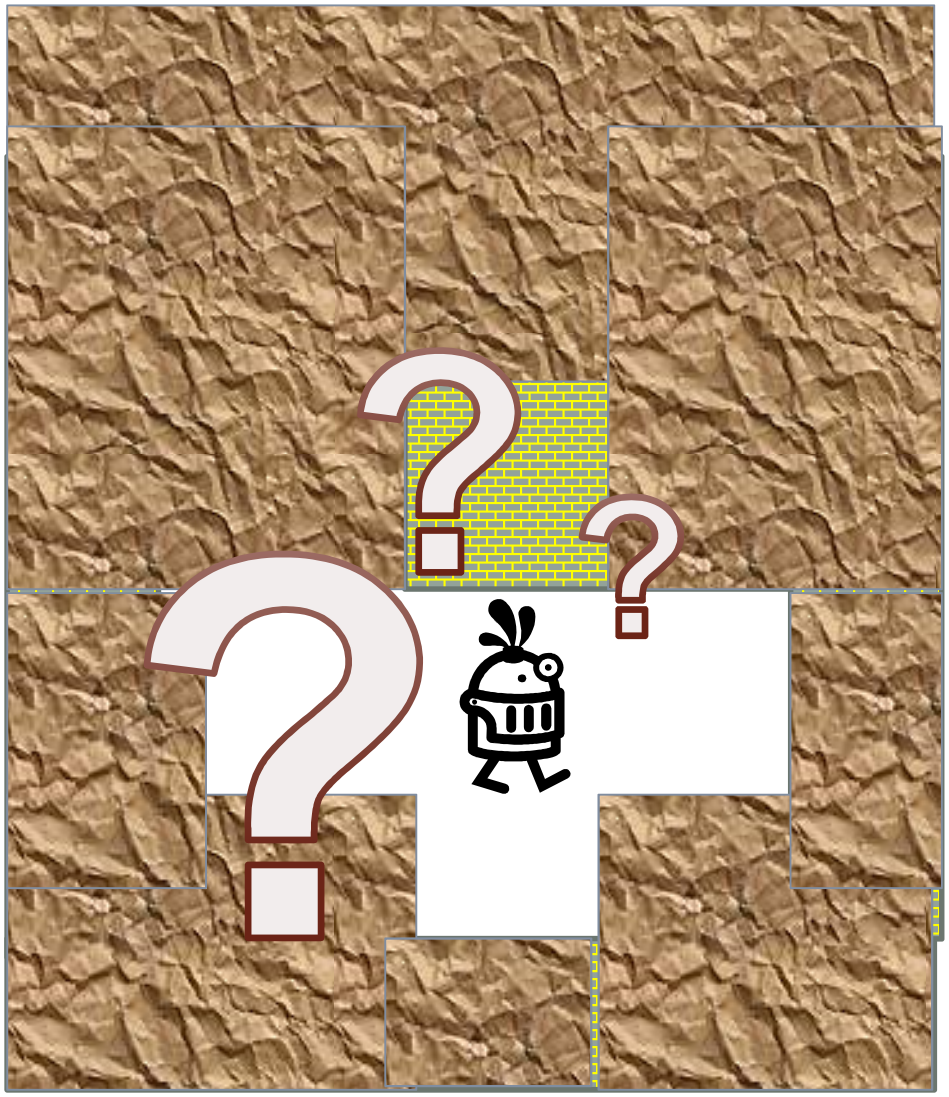
Quatrième solution: débrouille toi pour trouver le trésor!



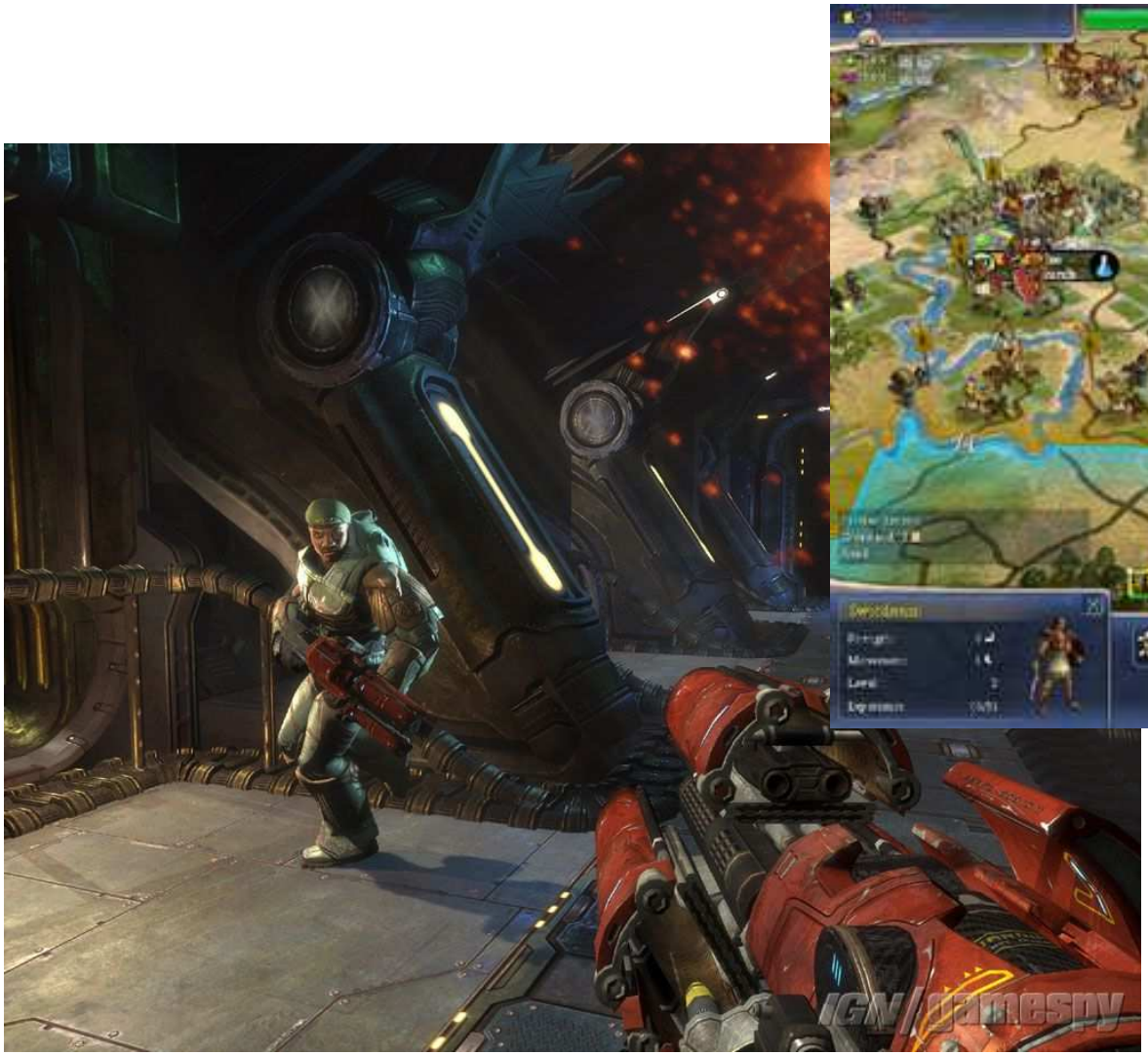
Cinquième solution: appelle tes copains, je coordonne d'en haut pour gérer les morts et vous dire où ne pas aller.



Sixième solution: Qui suis-je? Pourquoi suis-je dans ce jeu idiot?

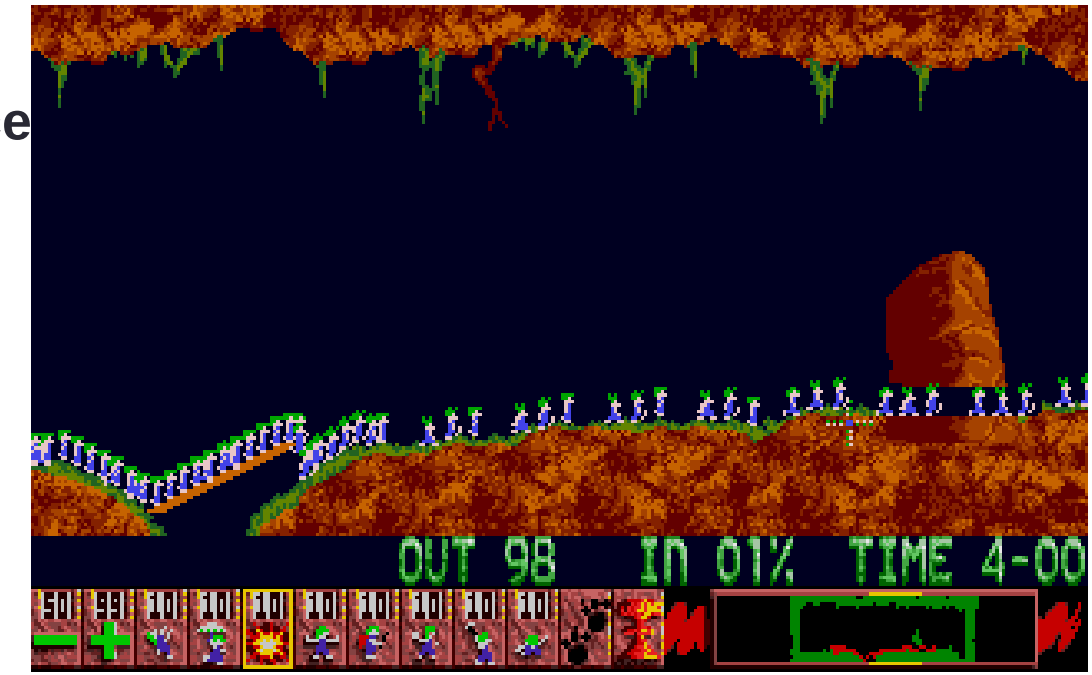


Le problème est très différent selon les applications

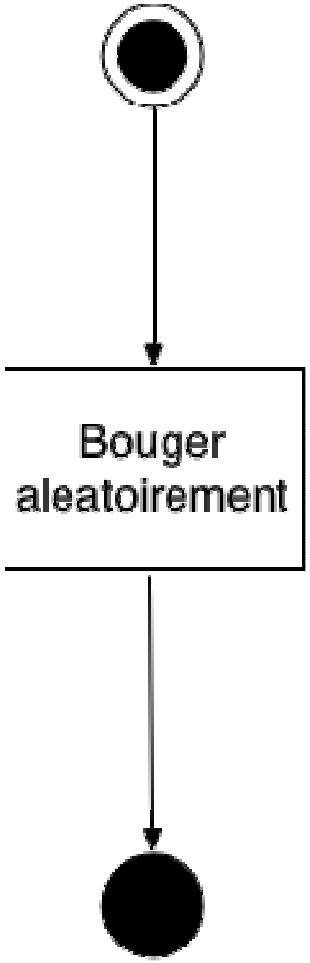


Modèles d'agent

- Introduction
- **Modèles Zero-Intelligence**
- Modèles réactifs
- Modèles interactionnel
- Agents cognitifs
- Modèles multi-niveaux
- Modèles reflexifs



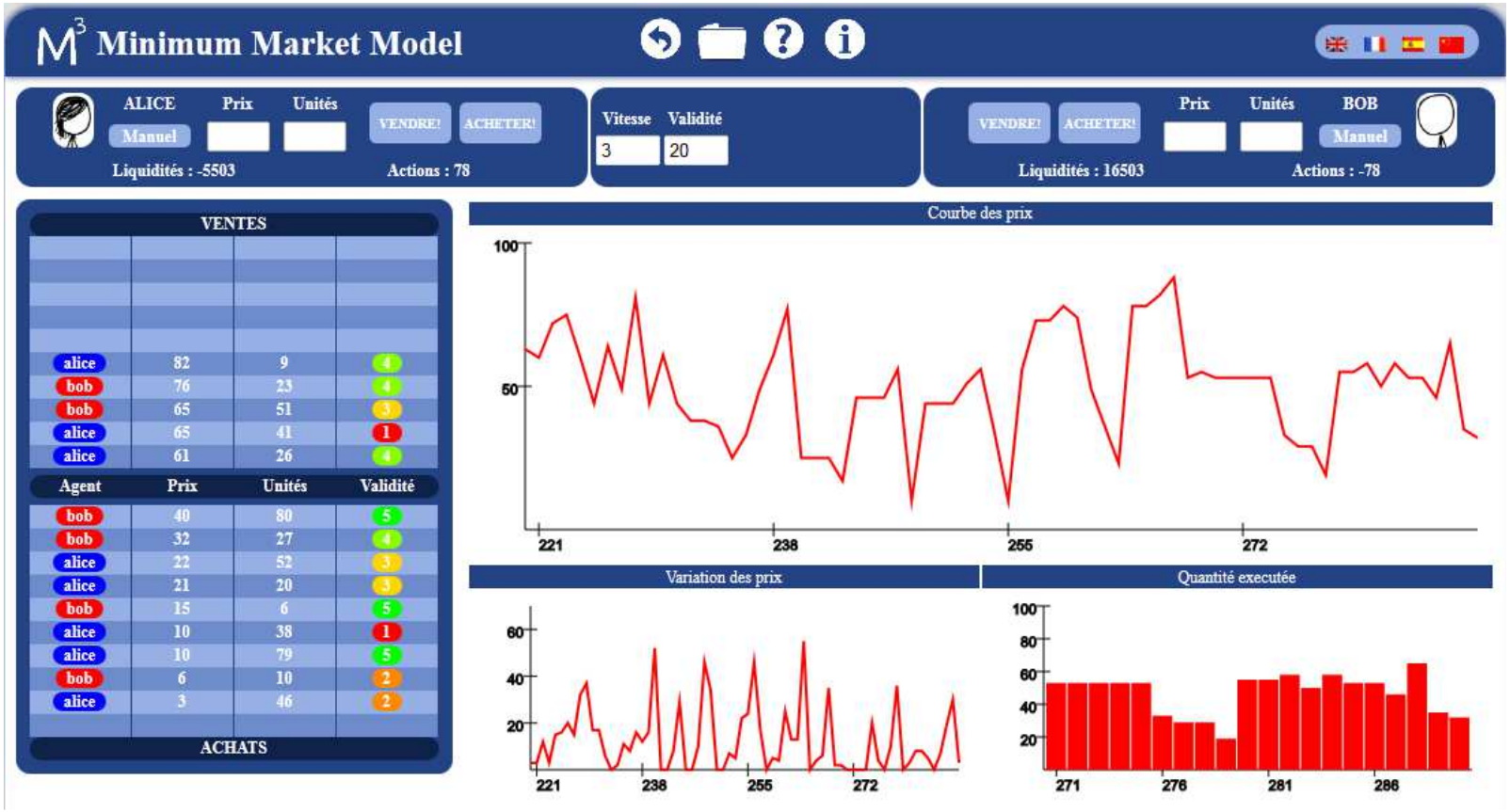
Agent Reflexe: Zero Intelligence (ZI)




```
species prey {
float size <- 1.0 ;
rgb color <- #blue;
vegetation_cell myCell <- one_of
(vegetation_cell) ;

init {
location <- myCell.location;
}

reflex basic_move {
myCell <- one_of (myCell.neighbours) ;
location <- myCell.location ;
}

aspect base {
draw circle(size) color: color ;
}
}
```

Une des applications principales des ZI: avoir un élément de comparaison. Ou simuler des comportements réalistes dans certains contextes…

- http://atom.univ-lille1.fr/js/

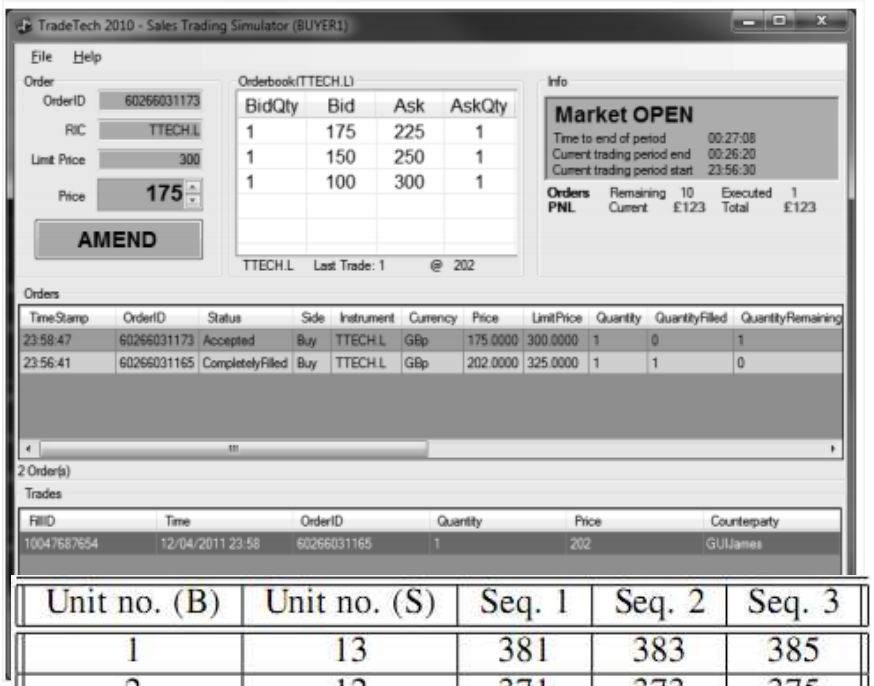


Quand le ZI fait mieux que l'agent humain…

- [Das et al. 2001] IJCAI
- [Vytelingum et al. 2008] AI
- [De Luca et Cliff 2011] IJCAI

- Principe:
  - 10 rounds de 3mn
  - 13 unités a vendre/acheter
  - 12 vendeurs, 12 acheteurs
  - 50% humain, 50% AI

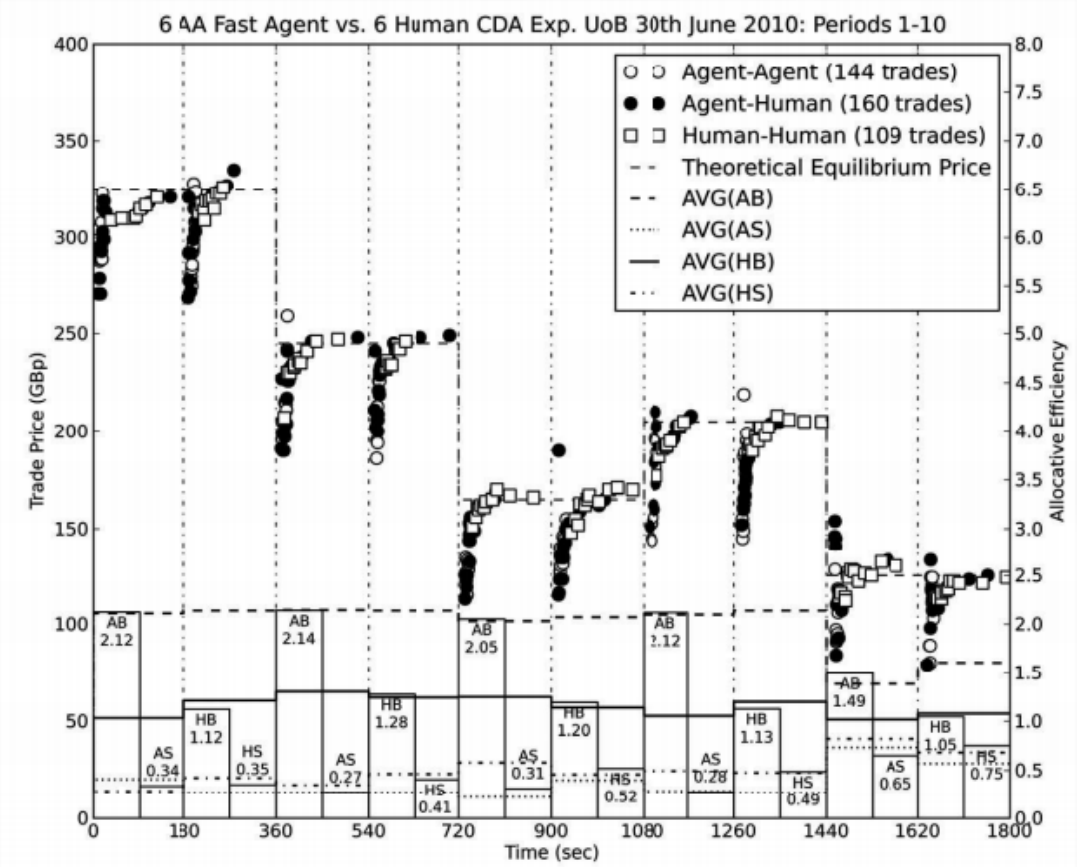
- Stratégies d'IA
  - Zero Intelligence Plus (ZIP)
    - Hausse de p si transaction supérieure
    - Baisse de p si transaction inférieure
  - GDX
    - Maximisation du profit espéré en considérant la probabilité que l'ordre passe à ce prix (en considérant les derniers M trades)
  - AA
    - Agressivité dynamique en fonction des trades acceptés
    - Agressivité: tendance à préférer des moins bons trades à des trades profitables



Unit no. (B)	Unit no. (S)	Seq. 1	Seq. 2	Seq. 3
1	13	381	383	385
2	12	371	373	375
3	11	361	363	365
4	10	351	353	355
5	9	341	343	345
6	8	331	333	335
7	7	321	323	325
8	6	311	313	315
9	5	301	303	305
10	4	291	293	295
11	3	281	283	285
12	2	271	273	275
13	1	261	263	265

- [Vytelingum 2008] AA>GDX>ZIP
- 2001/2011 AI>Humain
- ZI n'est pas suffisant pour battre l'humain, ZIP l'est

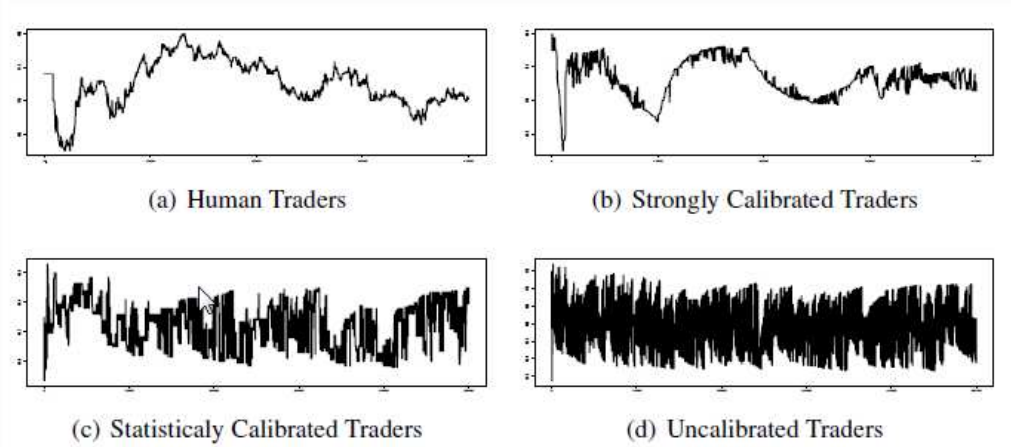
- Effet psychologique dans l'évolution du prix (forme de la courbe) s'il y a des joueurs humains
  - Courbe asymétrique et acheteurs favorisés
  - Courbe symétrique avec des agents artificiels
  - Les IA s'adaptent bien



Experiment	Trades			Performance			Market	
Strategy	A-A	A-H	H-H	Eff(A)	Eff(H)	Δ Profit(A-H)	Eff	Profit Disp
AA	41%	32%	27%	1.078	0.867	27%	0.978	793
GDX	33%	42%	25%	1.040	0.866	23%	0.954	568
ZIP	39%	30%	31%	1.014	0.941	9%	0.978	418

Le ZI ne reproduit pas la complexité de l'humain

- [Derveeuw et al 2007] [Veryzhenko et al. 2010] Artificial Economics
- Analyse du réalisme intraday à l'aide d'ATOM
- Agent envoyant des ordres aléatoires à des prix aléatoires
- 3 calibrations possibles comparées aux données réelles
  - Fonction log-normale
  - Normale calibrée en fonction du min-max des ordres d'achat/vente
  - Normale calibrée avec une fluctuation variant dans le temps
- Un ZIT n'est pas suffisant pour obtenir un comportement intraday réaliste
- Un agent simple bien calibré est suffisant

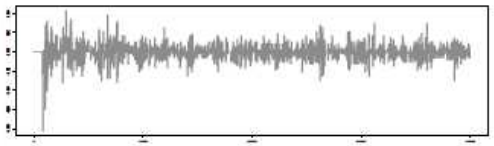

(a) Human Traders

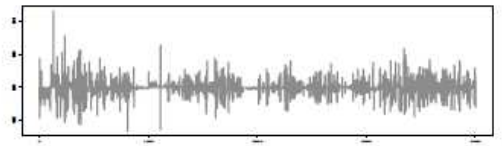
(b) Strongly Calibrated Traders

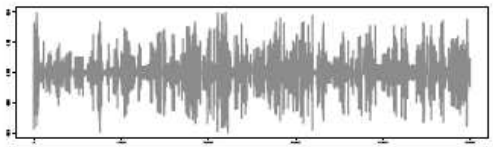
(c) Statisticaly Calibrated Traders

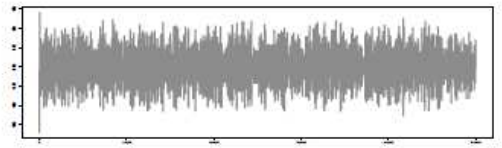
(d) Uncalibrated Traders

Fig. 1: Intraday price series


(a) Human Traders


(b) Strongly Calibrated Traders


(c) Statisticaly Calibrated Traders


(d) Uncalibrated Traders

$$\gamma(0) = 1 \qquad \delta(t) \sim N(0,1)$$

$$\gamma(t) = \gamma(t-1) - 0.001 \times t$$

$$P(t) = P_{min} + (P_{max} - P_{min} + 1) \times \gamma(t) \times \delta(t)$$

Comme les humains, les agents n'arrivent pas à battre le marché…

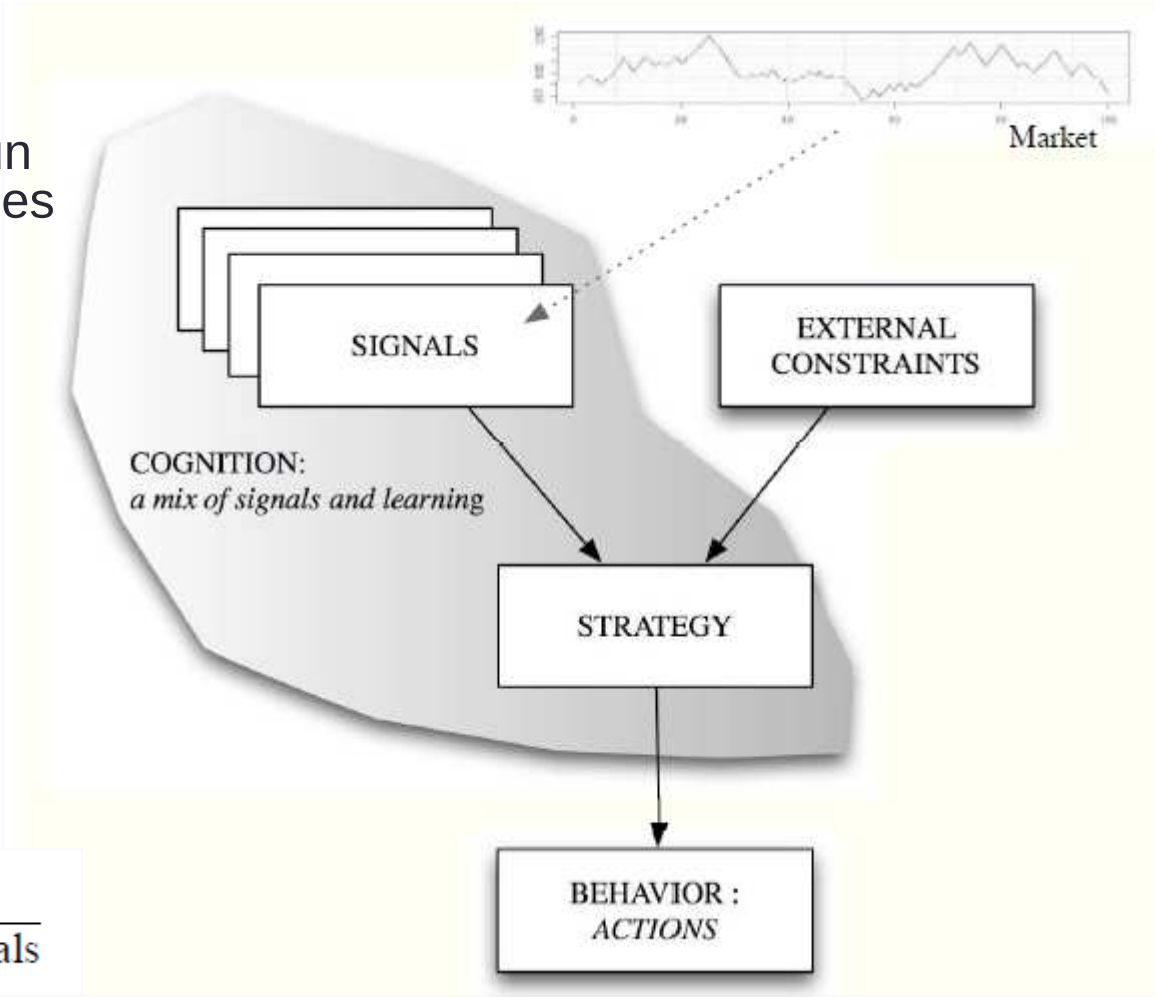
- [Braudouy et Matthieu 2007] Artificial Economics

- Objectif: tester l'efficacité de stratégies chartistes à l'aide d'un simulateurs et de données réelles

- +100000 signaux,
- Dont 6640 HitRate>50%
- Dont 97 avec une activité suffisante

- Plusieurs types de stratégies d'utilisation des signaux
  - Basic: suivi du signal
  - Inverse
  - ZIT

$$HR_{MMA_{90-10}} = \frac{\text{correct signals}}{\text{total number of signals}}$$



…sauf en trichant

- Résultat:
  - Impossible de battre le marché, en particulier avec des frais de transaction
  - Il est parfois plus intéressant de faire l'inverse de ce qu'indique le signal
  - Un agent « cheater » fait beaucoup mieux

	0% rate				0.5% rate			
Signature	Mean (10^{-4})	σ	Sharpe Ratio	Rank /97	Mean (10^{-4})	σ	Sharpe Ratio	Rank /97
B&H	3.0879	0.0109	0.0283	–	3.0879	0.0109	0.0283	–
MMA-1-4	3.9428	0.0074	0.0529 *	13	-0.010	0.00870	-0.1151	88
MMA-1-6	4.4715	0.0074	0.0599 *	2	-6.006	0.00852	-0.07043	69
MMA-1-7	4.1877	0.0074	0.0562 *	7	-5.6647	0.00845	-0.06701	67
Mom.-2-1	3.2710	0.0081	0.0402 *	51	-9.3296	0.00915	-0.10194	83
Mom.-5-0	4.1035	0.0074	0.0552 *	8	-6.3746	0.00835	-0.07630	70
Var.-1-1-4	3.9915	0.0074	0.0535 *	12	-6.5312	0.0083	-0.0778	72
Var.-1-5-1	3.1547	0.0107	0.0294 *	67	1.5685	0.01081	0.01450	23
Var.-1-7-1	3.0960	0.0108	0.0285 *	70	2.8503	0.01087	0.02622	5
Var.-1-8-1	3.0403	0.0108	0.0279	71	2.8839	0.01088	0.02650	4
Var.-1-9-1	2.9761	0.0108	0.0273	73	2.9091	0.01088	0.0267	3

MMA : mixed moving average, Mom. : momentum, Var.: Variation
** stands for "actually outperform the Market"*

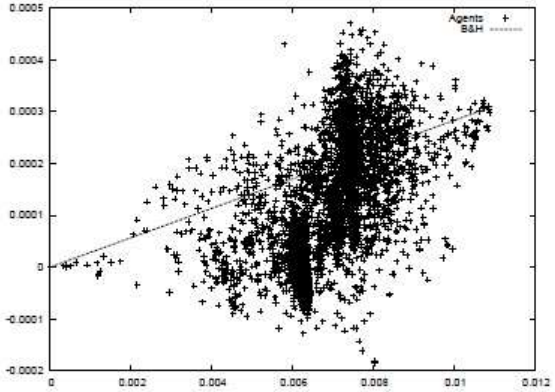


Fig. 6. Without trans. costs

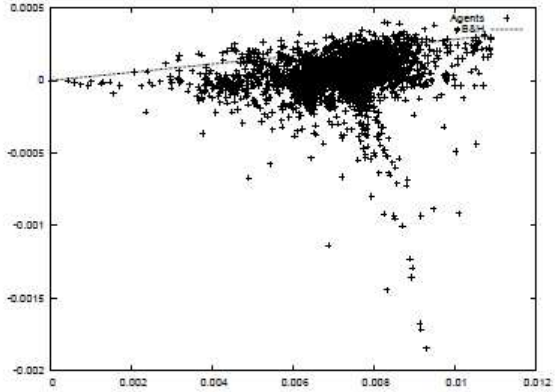


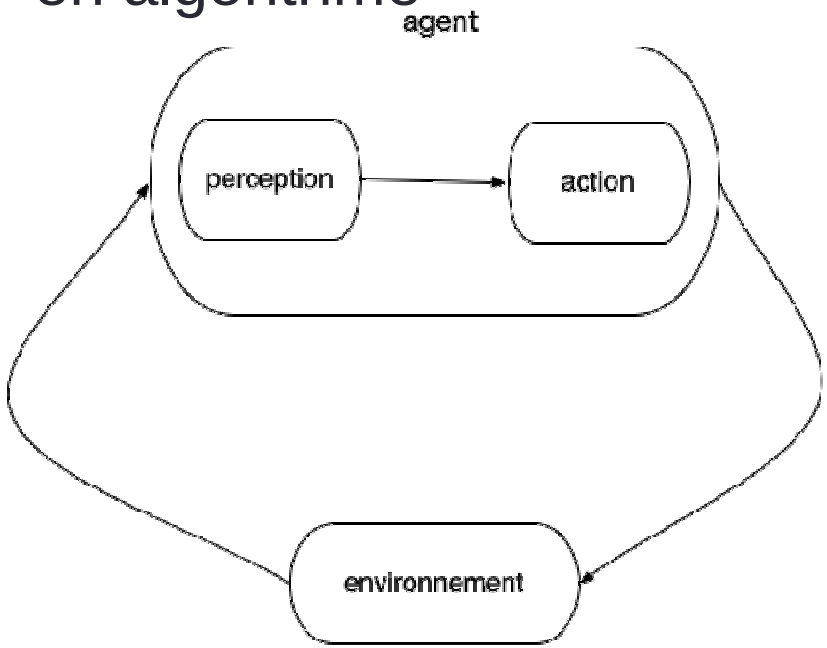
Fig. 7. With 0.5% trans. costs

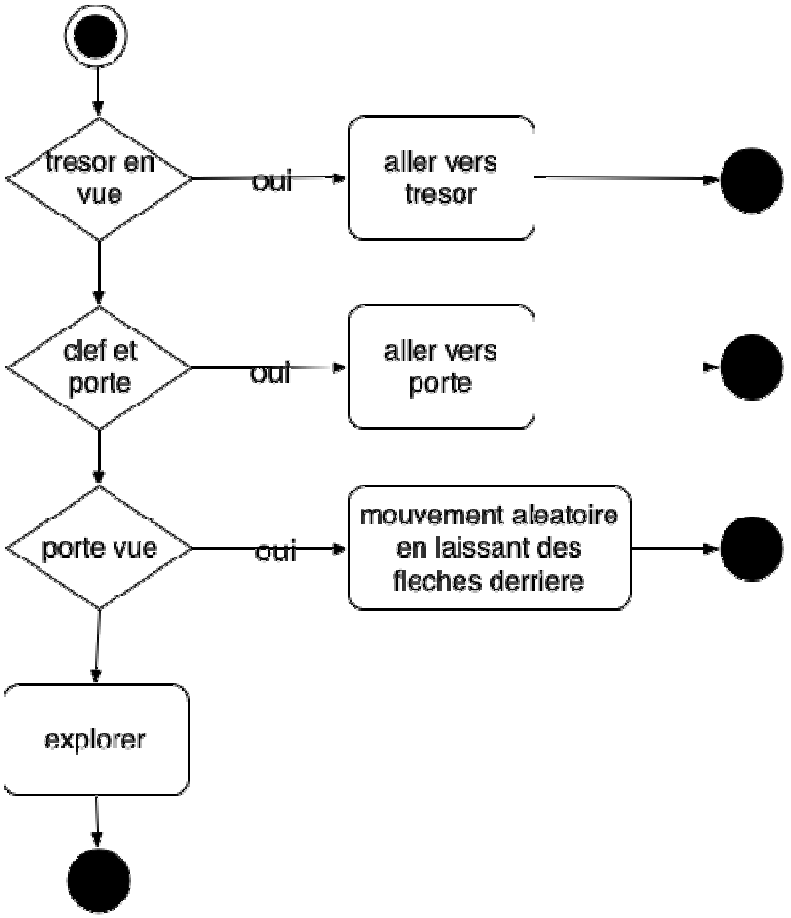
Modèles d'agent

- Introduction
- Modèle Zero-Intelligence
- **Modèles réactifs**
- Modèles interactionnel
- Agents cognitifs
- Modèles multi-niveaux
- Modèles reflexifs

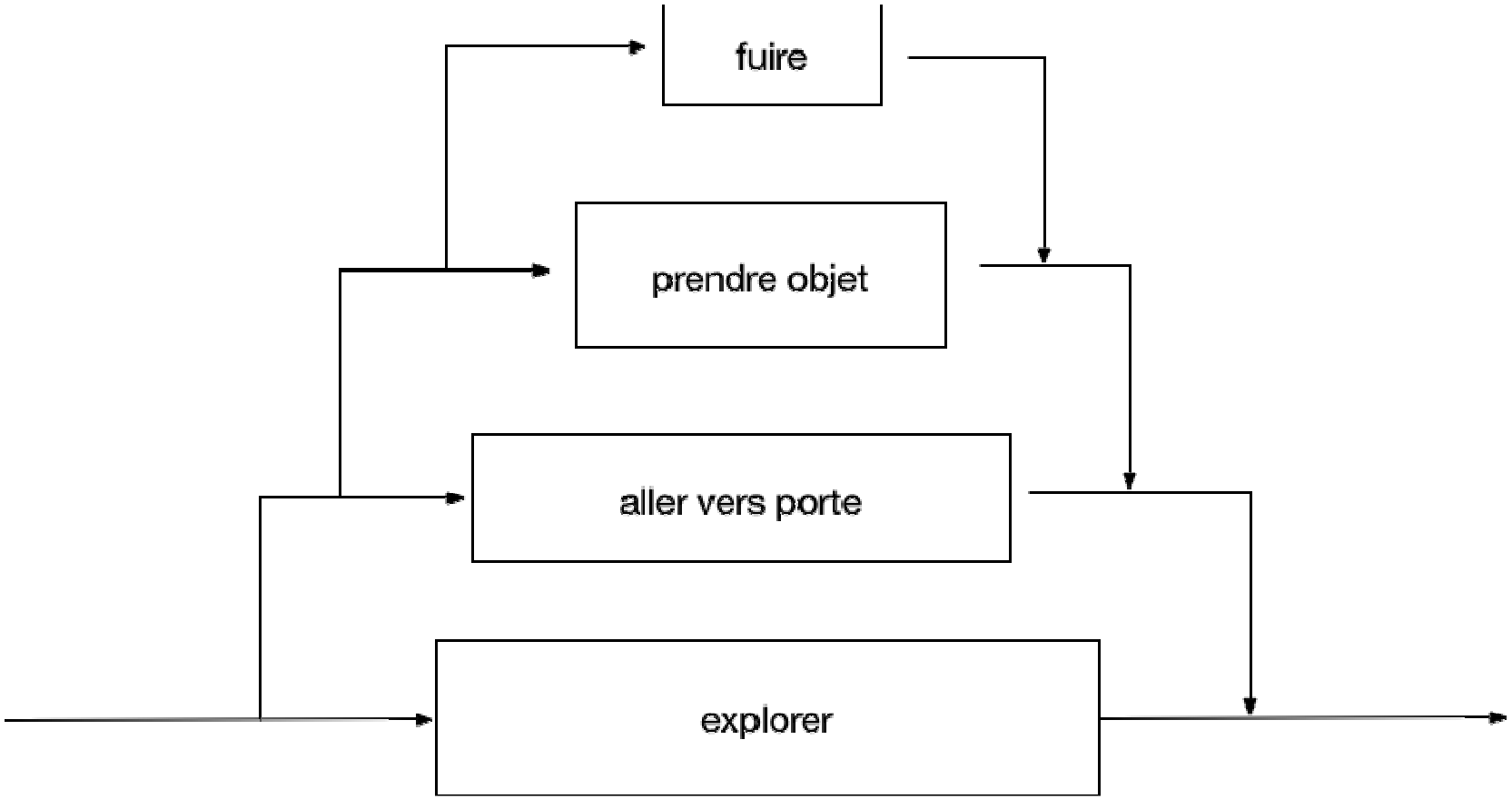
Agent Reflexe: Ensemble de règles

- Logique réactive: Perception -> Action
- Description intuitive
- Traduction instantanée
  en algorithme



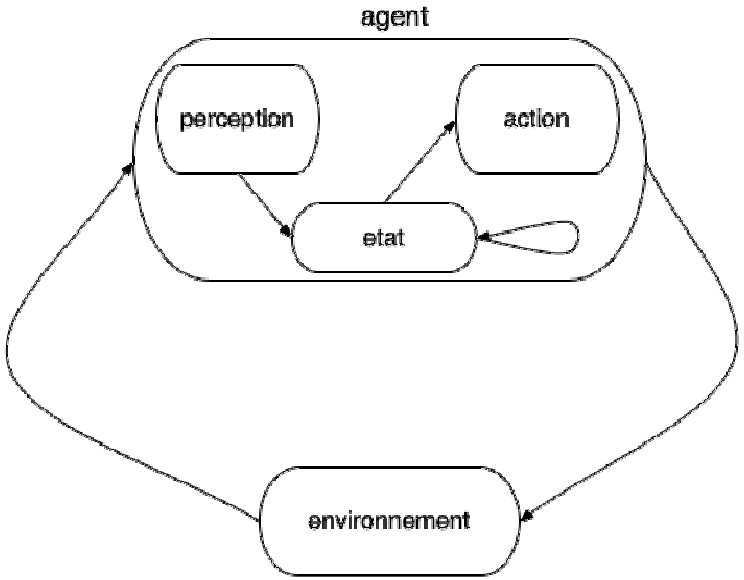


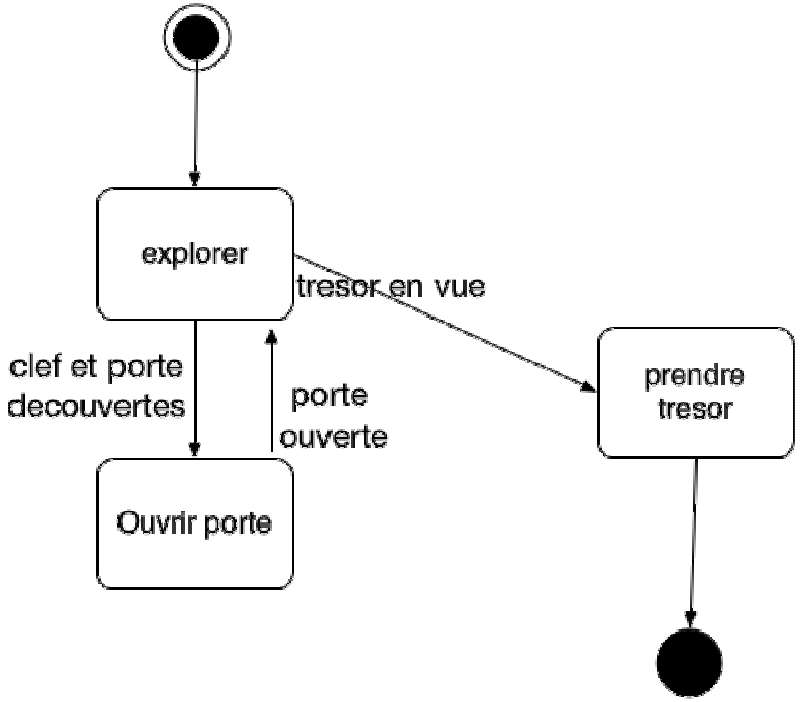
Le raisonnement par exception permet un comportement plus complexe

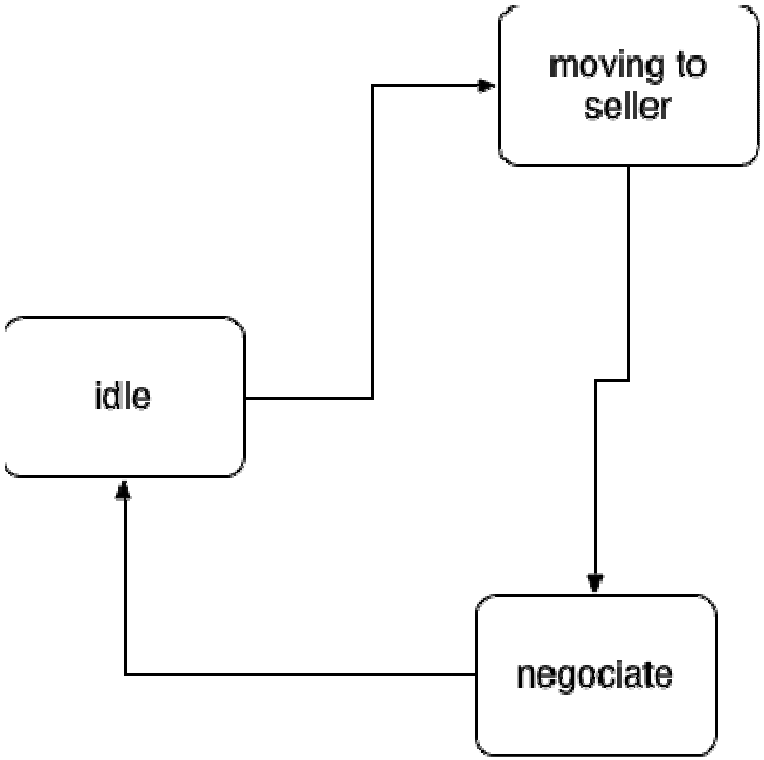


Les états permettent de structurer le comportements

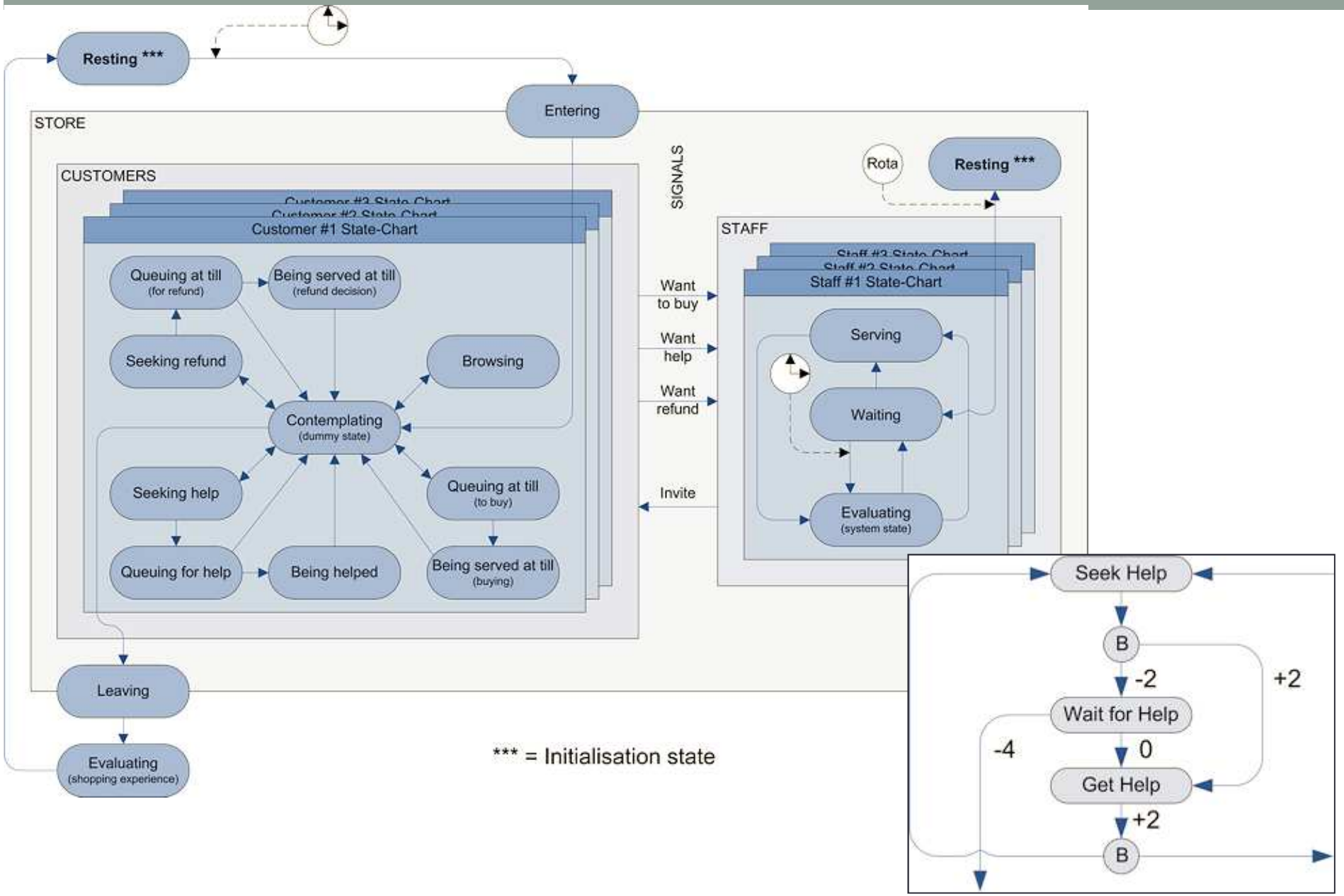
- Introduction possible de la notion d'Etat
- Permet de modéliser simplement des comportements complexes
- Traduction en fonctions





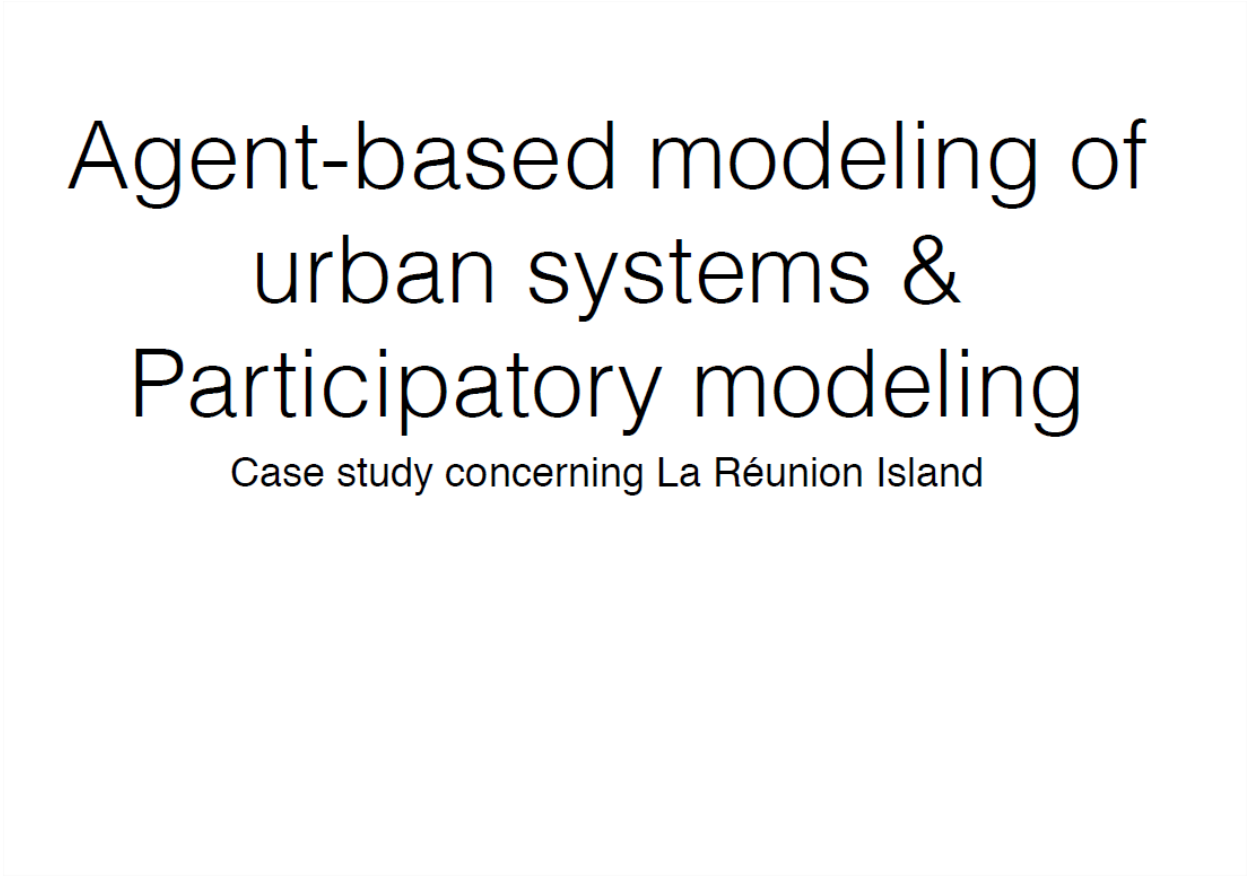



```
species buyer parent: acteur control:fsm
{


state idle initial: true {
do action: choosenextseller {
}
transition to: MovingToSeller  when:
((nextseller!=nil)and(nextseller!=currentse
ller));
}
state MovingToSeller {
do action: goto {
arg target value: nextseller ;
}
transition to: Negociatingwhen:
((length(nearsellers)>0) and (nearsellers
index_of nextseller)!=-1);
}
state Negociating {
transition to: idle when: (true) {
do action: negociate ;
}
}
```



[Siebers et Aickelin 2011] JASSS

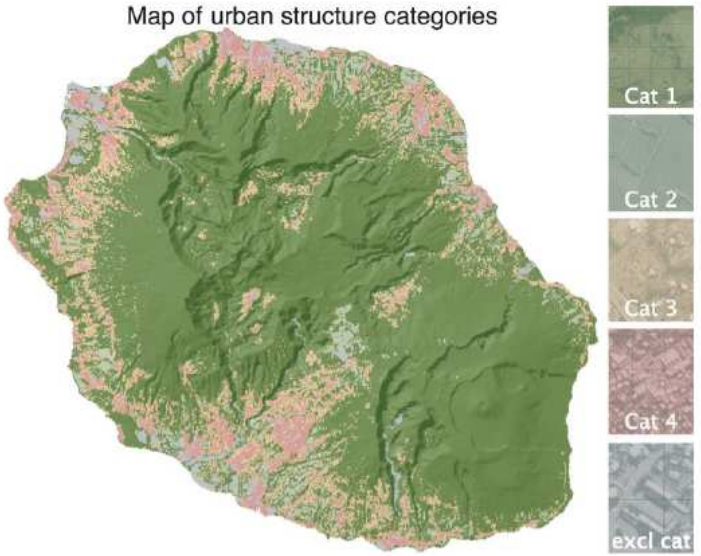
Un exemple de projet en cours…



- [ANR Acteur, Frederic Rousseaux & Nicolas Becu 2014]

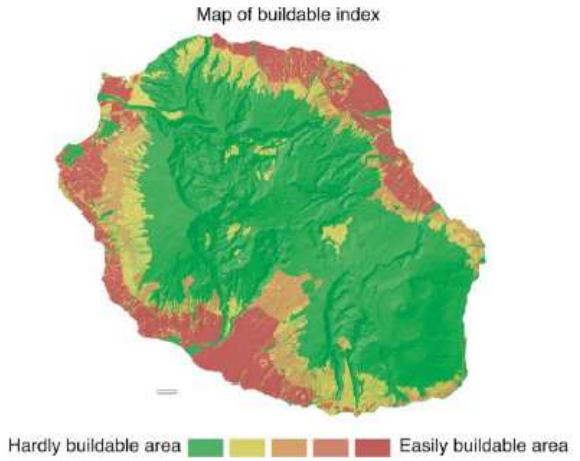
Introduction

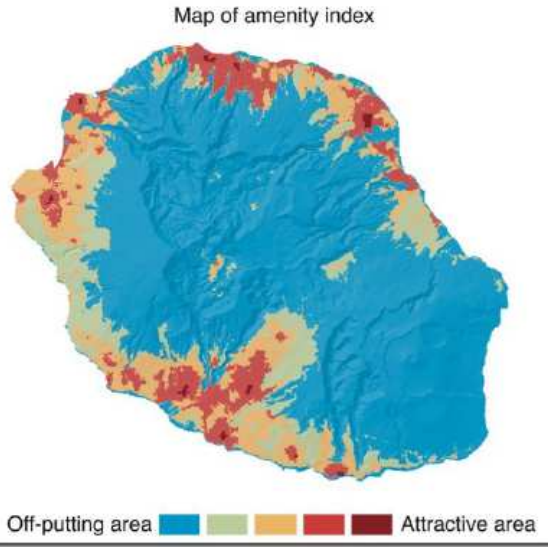
- Un modèle existant développé sous GAMA
- Modèle simple d'évolution de l'occupation du sol (description de structures urbaines et la population)
- Description assez fine de l'ensemble du territoire à grande résolution (mailles de 200m)
- Comportement des habitants basique (basé sur Schelling)



Le territoire décrit :

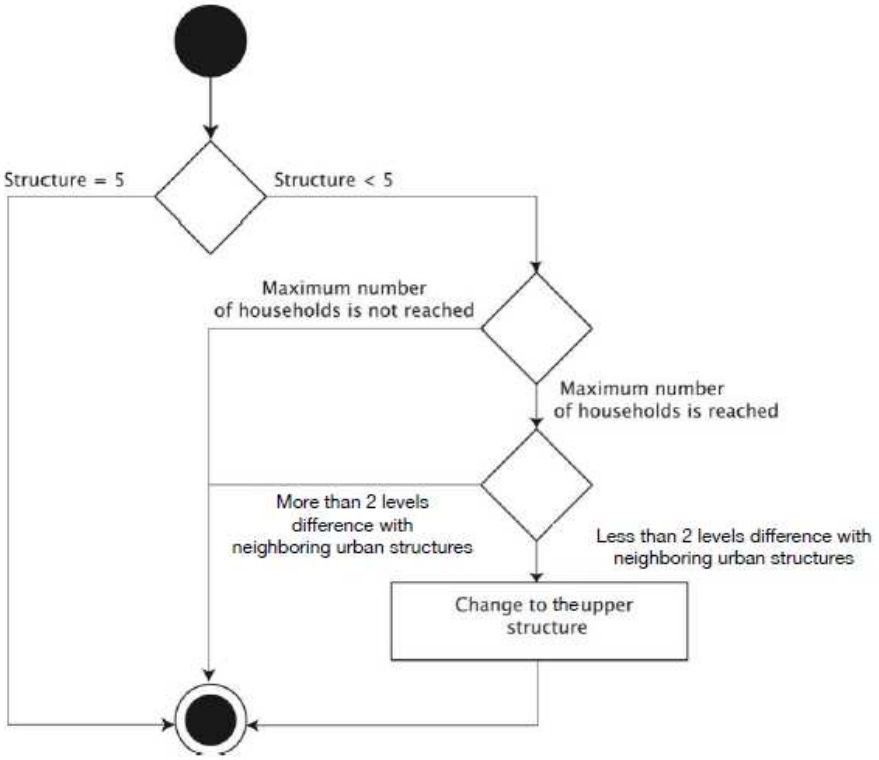
- Un degré de constructibilité (réglementations, limites physiques)
- Un niveau d'aménité (attractivité)
- Une structure urbaine (4 classes d'évolution possible : peu ou non bâti, périurbain bas, urbain moyennement dense, urbain dense. 1 classe figée : zones industrielles)
- Une population (données INSEE de 2010)





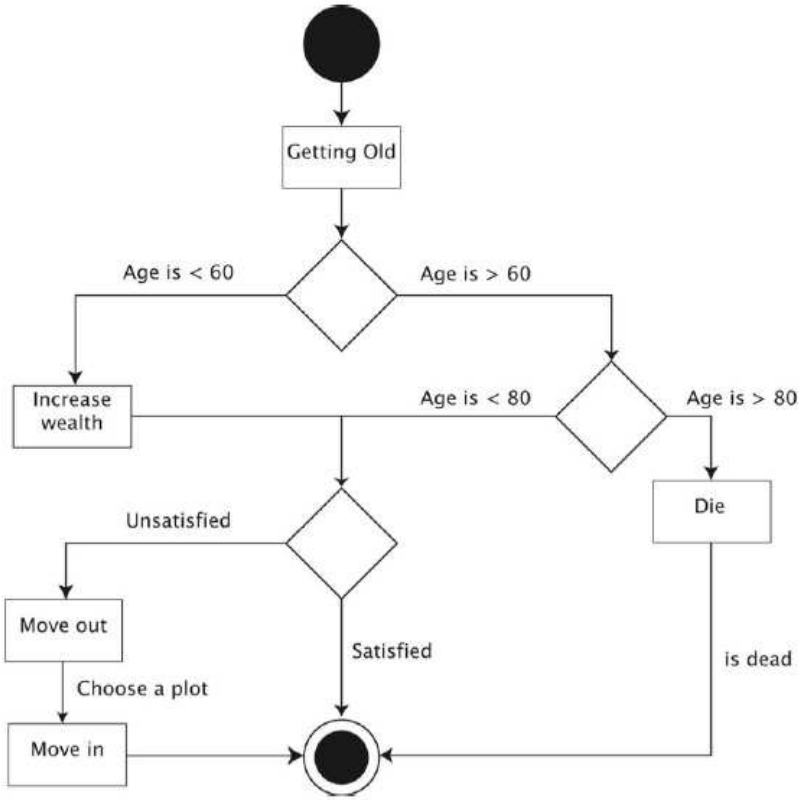
Deux agents

- Un agent « structure urbaine »
- 5 classes correspondants à 5 archétypes de structures urbaines
- évolution linéaire en fonction de règles simples (population, voisinage)
- Beaucoup de cellules (382520 : 655x384)

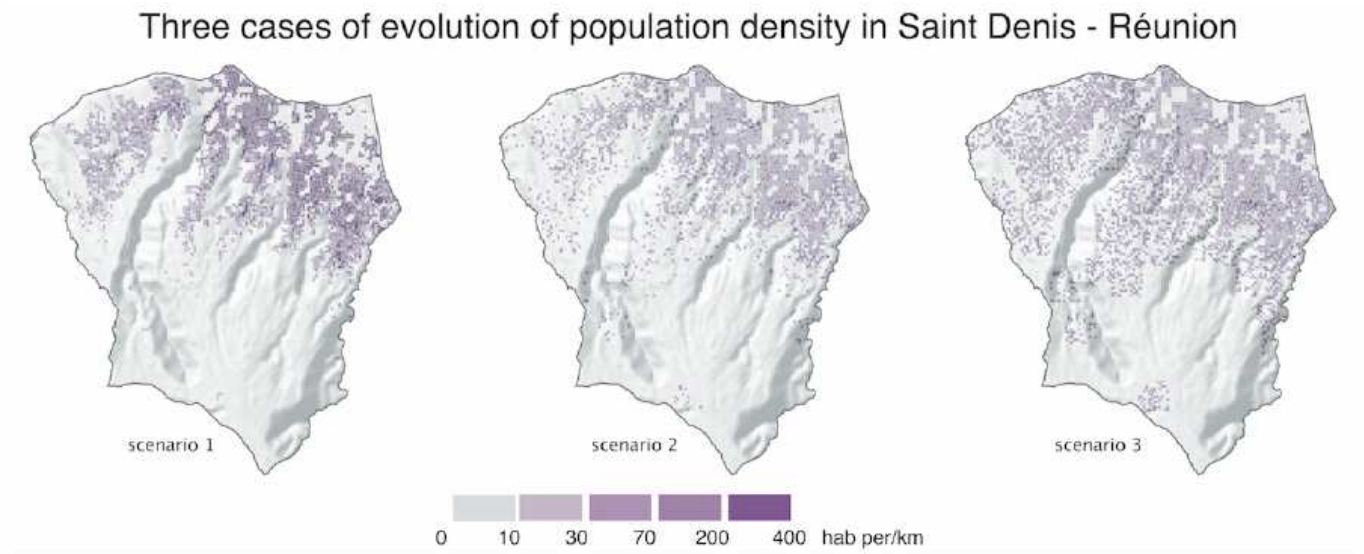


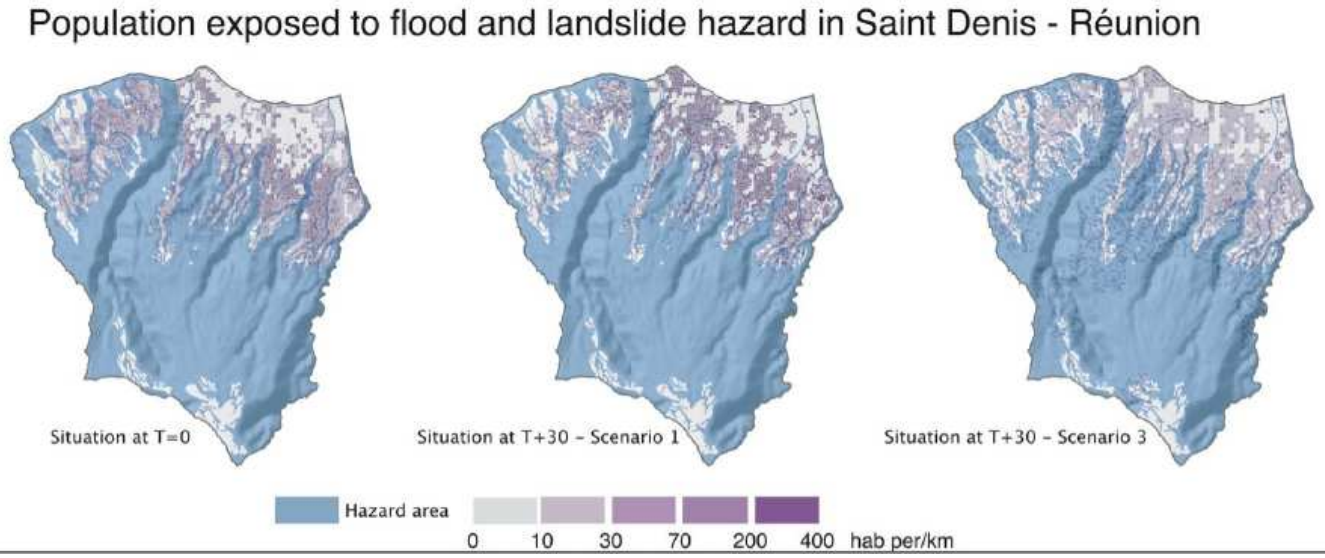
Deux agents

- Un agent « households »
- Déménage vers une cellule (n'importe laquelle dans l'ile) qui le satisfait
- Satisfaction liée à la structure du territoire (proximité d'aménité, ressemblance sociale, possibilité de construction)
- Beaucoup d'agents (+ de 300000)



Scénarios

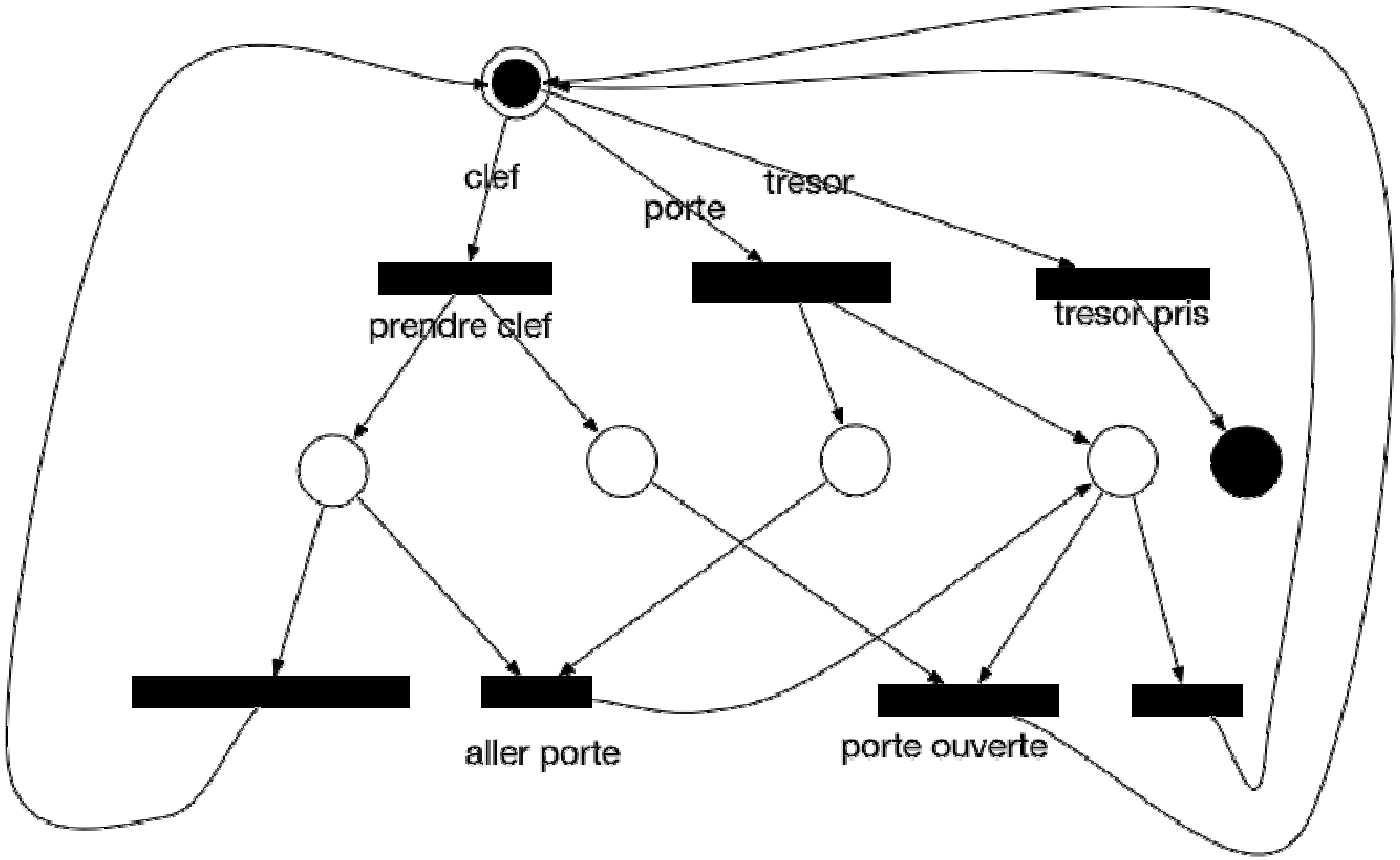

Three cases of evolution of population density in Saint Denis - Réunion
scenario 1
scenario 2
scenario 3
0
10
30
70
200
400
hab per/km


Population exposed to flood and landslide hazard in Saint Denis - Réunion
Situation at T=0
Situation at T+30 - Scenario 1
Situation at T+30 - Scenario 3
Hazard area
0
10
30
70
200
400
hab per/km

Automates

- Avantages
  - Intuitif: Programmation directe du comportement
  - Particulièrement adapté pour la simulation
  - Simple à programmer, rapide à exécuter
- Limites
  - Rigide
  - Pas de raisonnement sur les objectifs ou d'abstraction
  - Il faut prévoir la plupart des cas
- Extensions possibles:
  - Modularité adapté pour l'évolution
  - Paramètres simples à interpréter et donc à calibrer/optimiser

Les réseaux de Petri permettent une richesse plus grande et des calculs formels

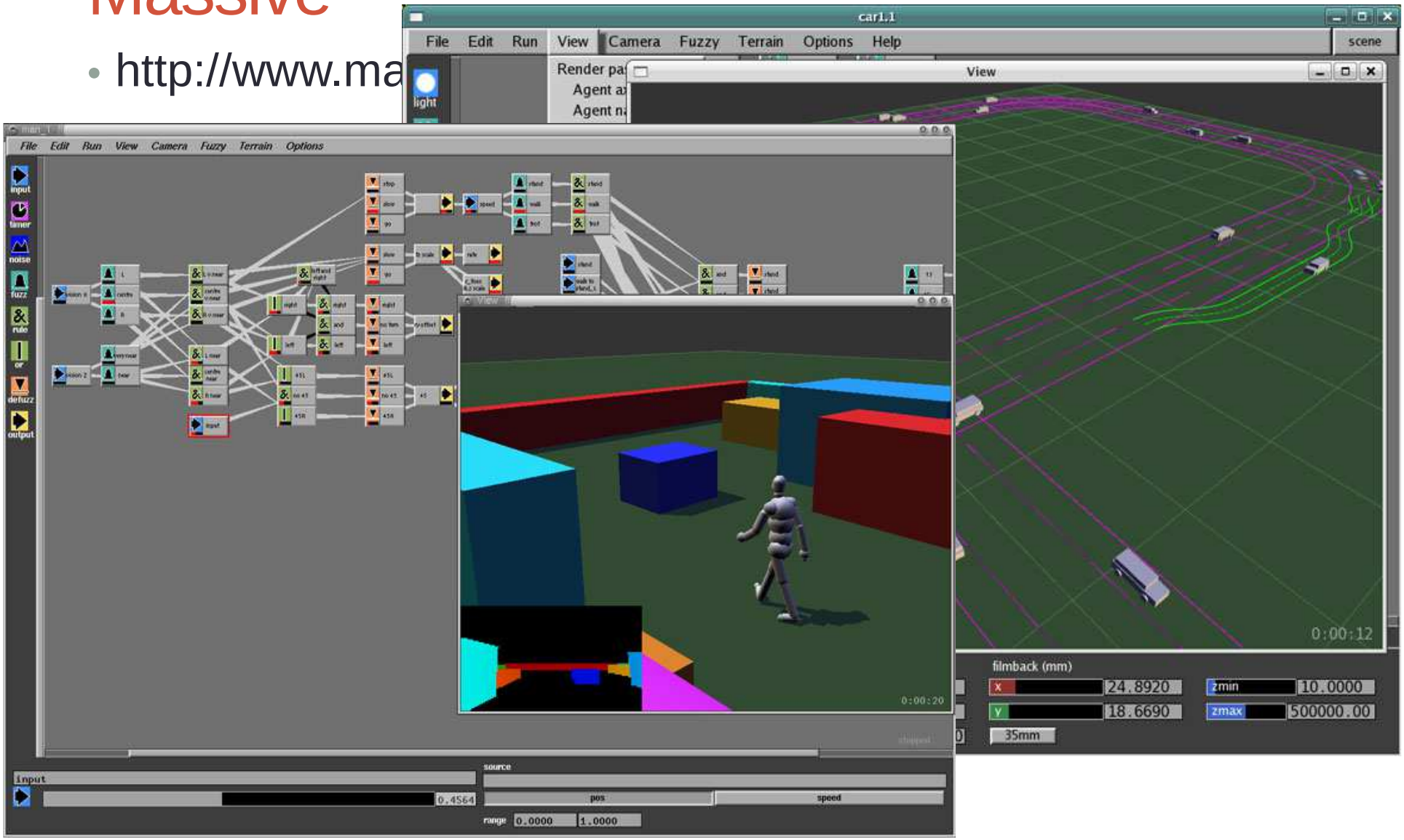


Et ils ont un grand nombre d'extensions

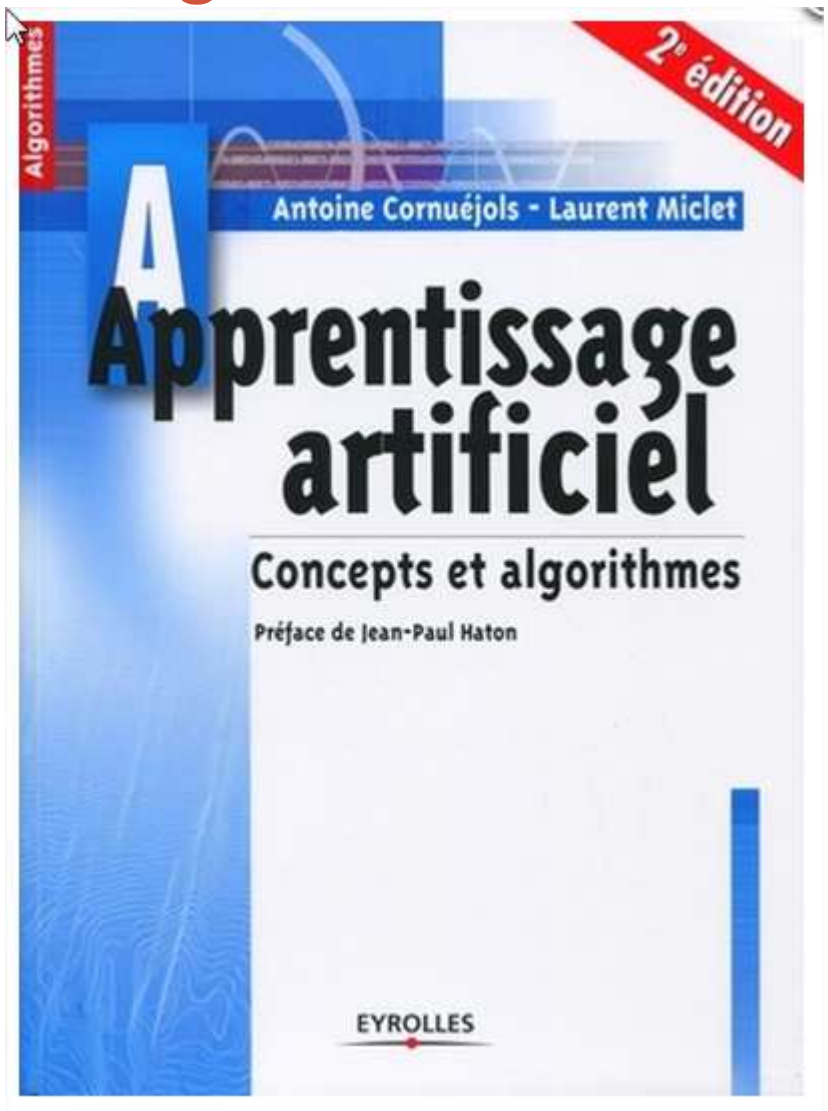
- Réseau de Petri colorés
  - Plusieurs couleurs de jetons
  - Permet des conditions plus complexes
- Réseau de Petri récursifs
  - Définition d'une transition par un autre réseau de pétri
  - Permet de représenter des structures complexes, éventuellement de calculer les réseaux au moment de leur développement
- Système distribué
  - Interaction de réseaux de Petri
  - Transitions partagées par plusieurs agents
  - Permet de formaliser du travail collaboratif

Je veux concurrence Peter Jackson: Massive

- http://www.ma



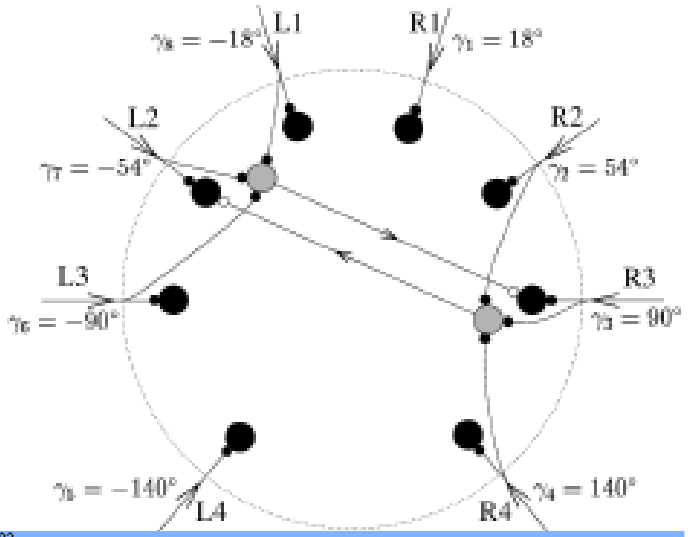
Le réactif permet une grande variété d'apprentissages



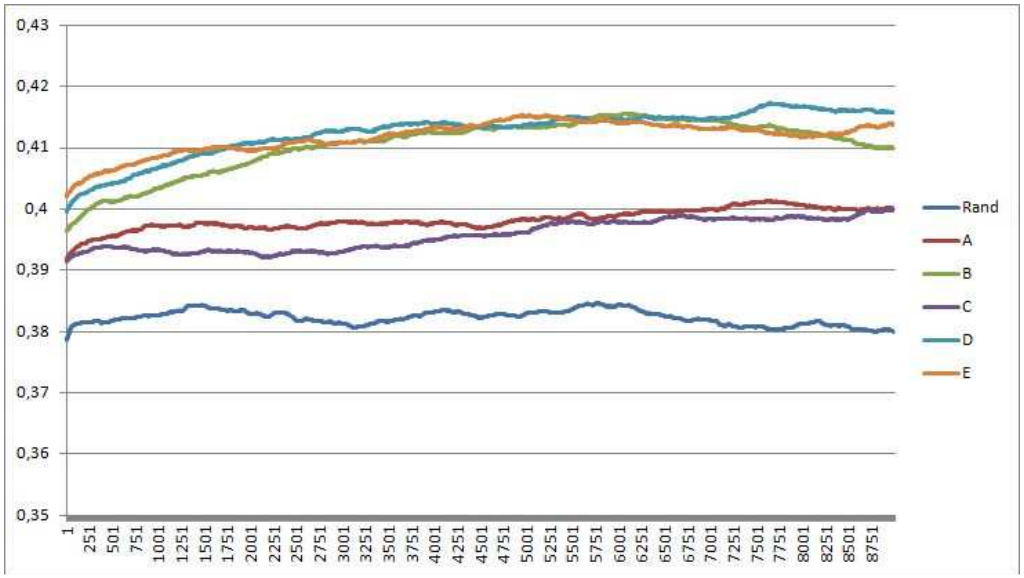
Réservoir de neurones pulsés comme contrôleur d'agent

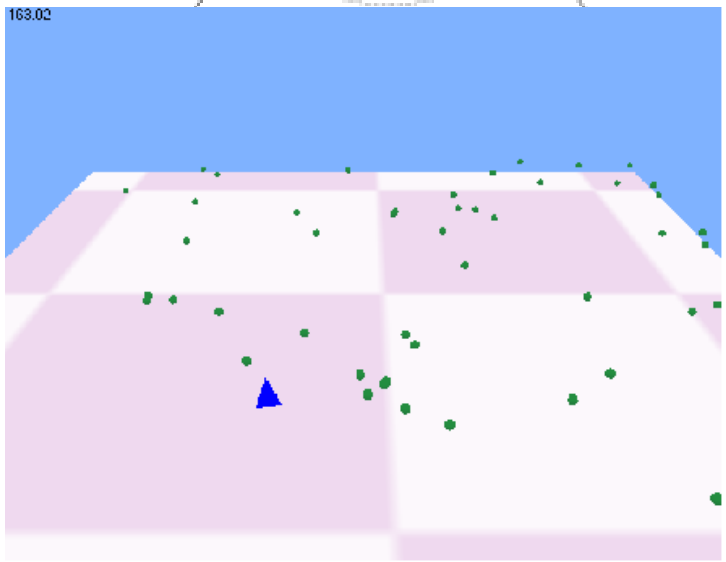
[Monteiro, Caillou and Netto 09]

- Réservoir de réseau pulsés reliés aux:
  - Perceptions (énergie et angle de nourriture)
  - Actions (angle de rotation)
- Apprentissage STDP [Song&Miller&Abbot, 2003]



- Travail en cours
- Résultats:

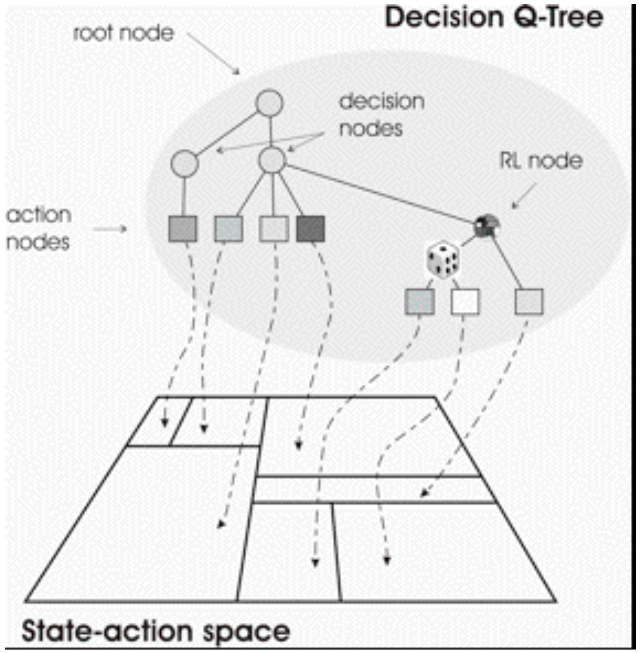


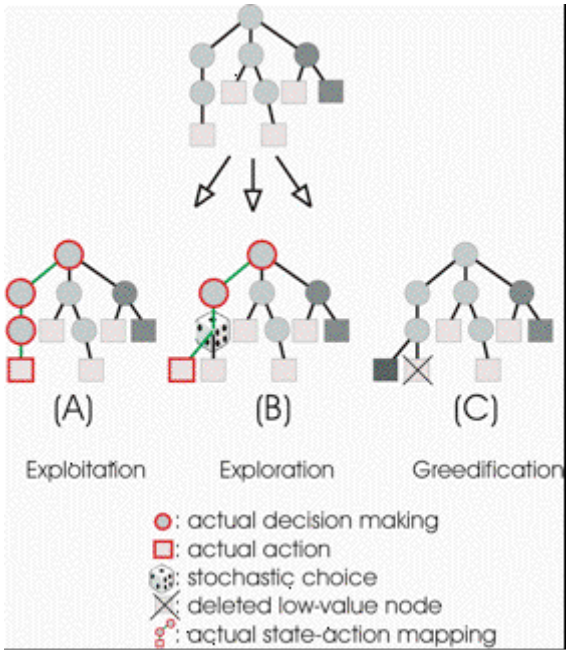


Pour s'adapter… vie artificielle

- New Ties [2004-2007]
- Combinaison de:
  - Apprentissage évolutionnaire (transmission de gènes)
  - Apprentissage social (Communication et apprentissage de langage)
  - Apprentissage individuel (Q Learning dans arbre de décisions)
- Plus de 1000 agents évoluant simultanément sur un cluster
- Problèmes:
  - Comprendre ce qu'il se passe…

[Gilbert et al. 2006]





Mais n’est pas utilisé dans la plupart des jeux

Counter Strike

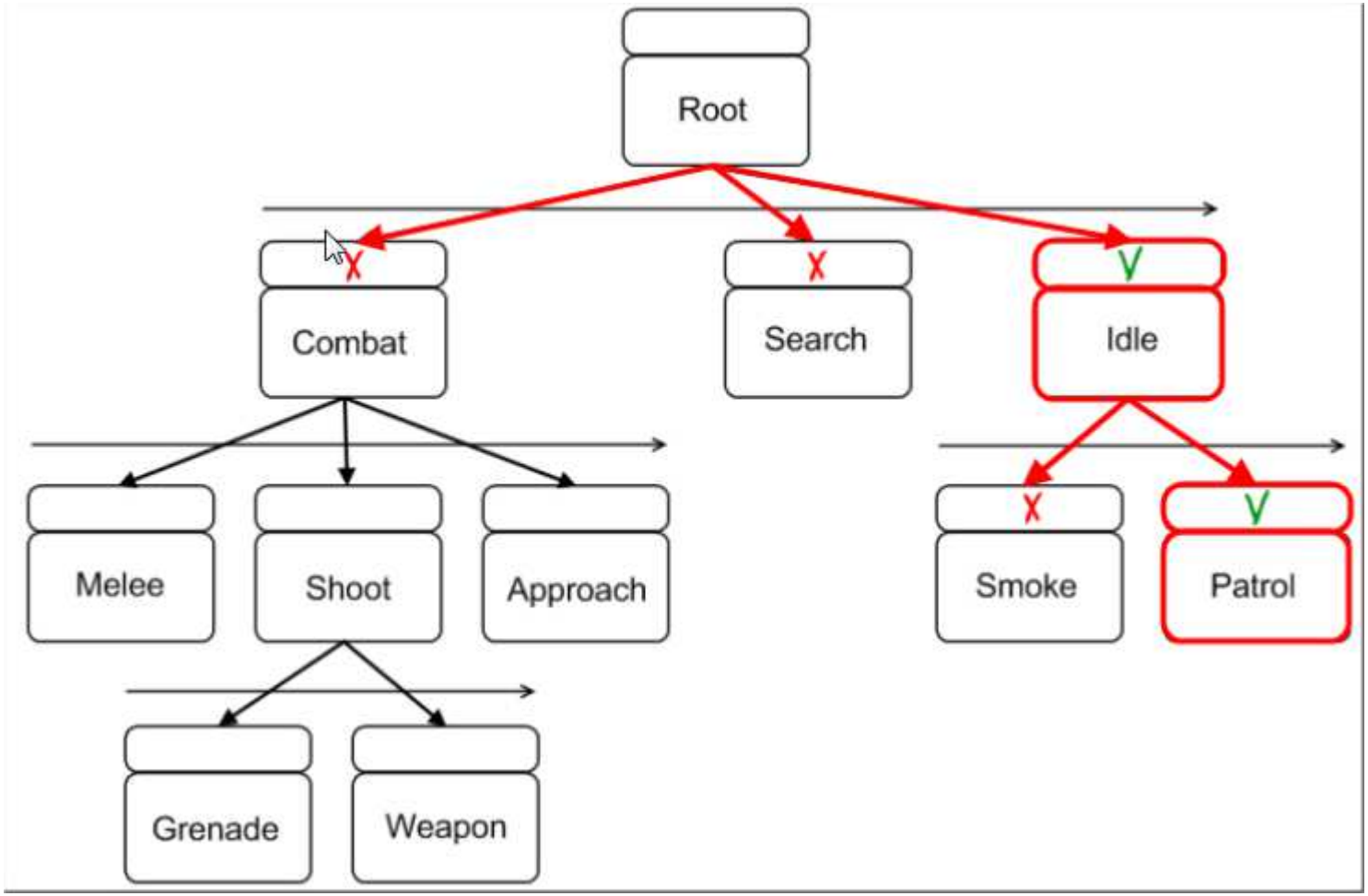
Rule-Based Scenario

Halo 2

FSM

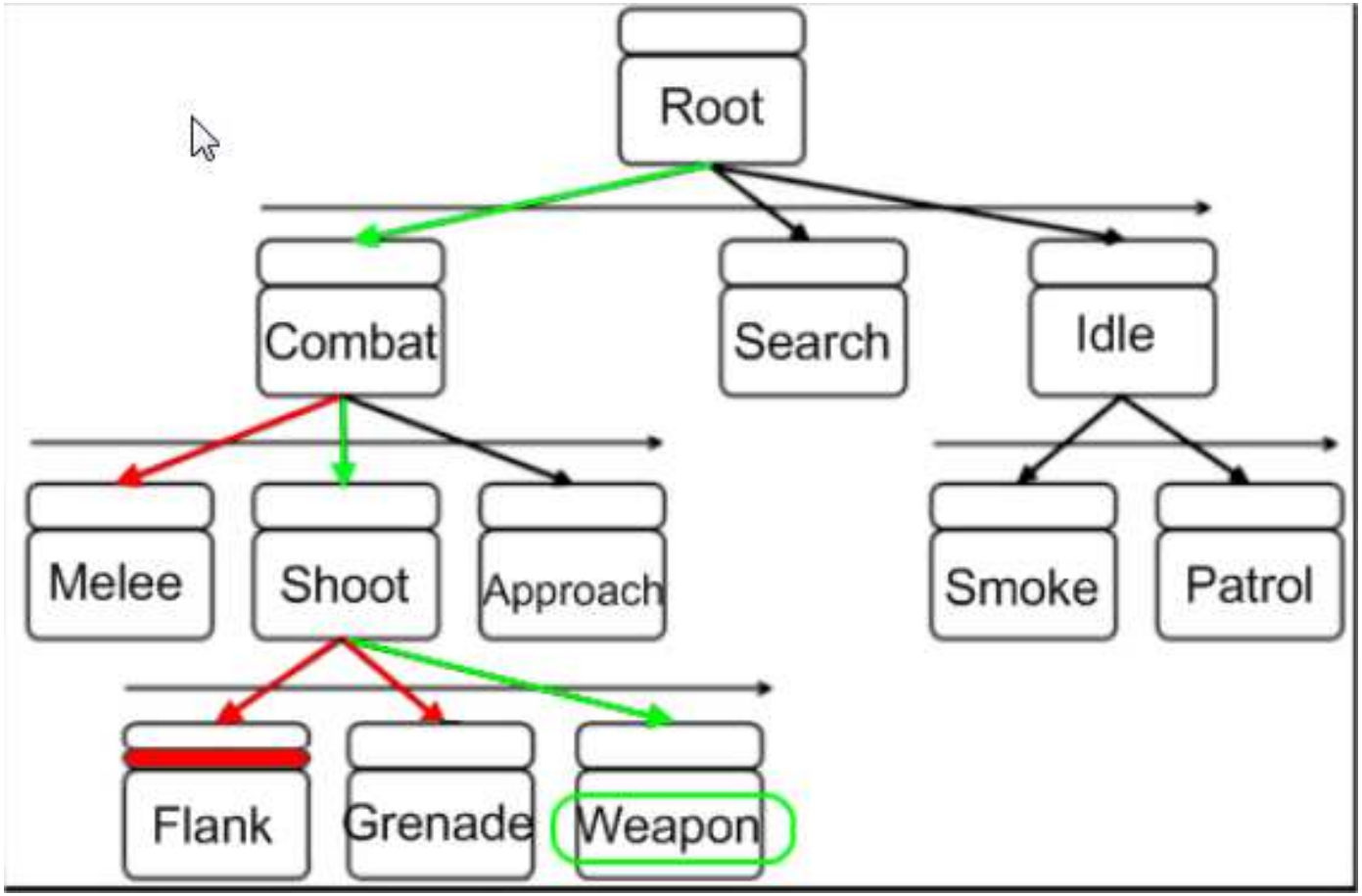
[Umarov et al. 2012] Int. Journal of Gaming and Computer mediated Simulation

Même les jeux modernes ont des structures simples, tels que Crysis avec des Behavior Trees

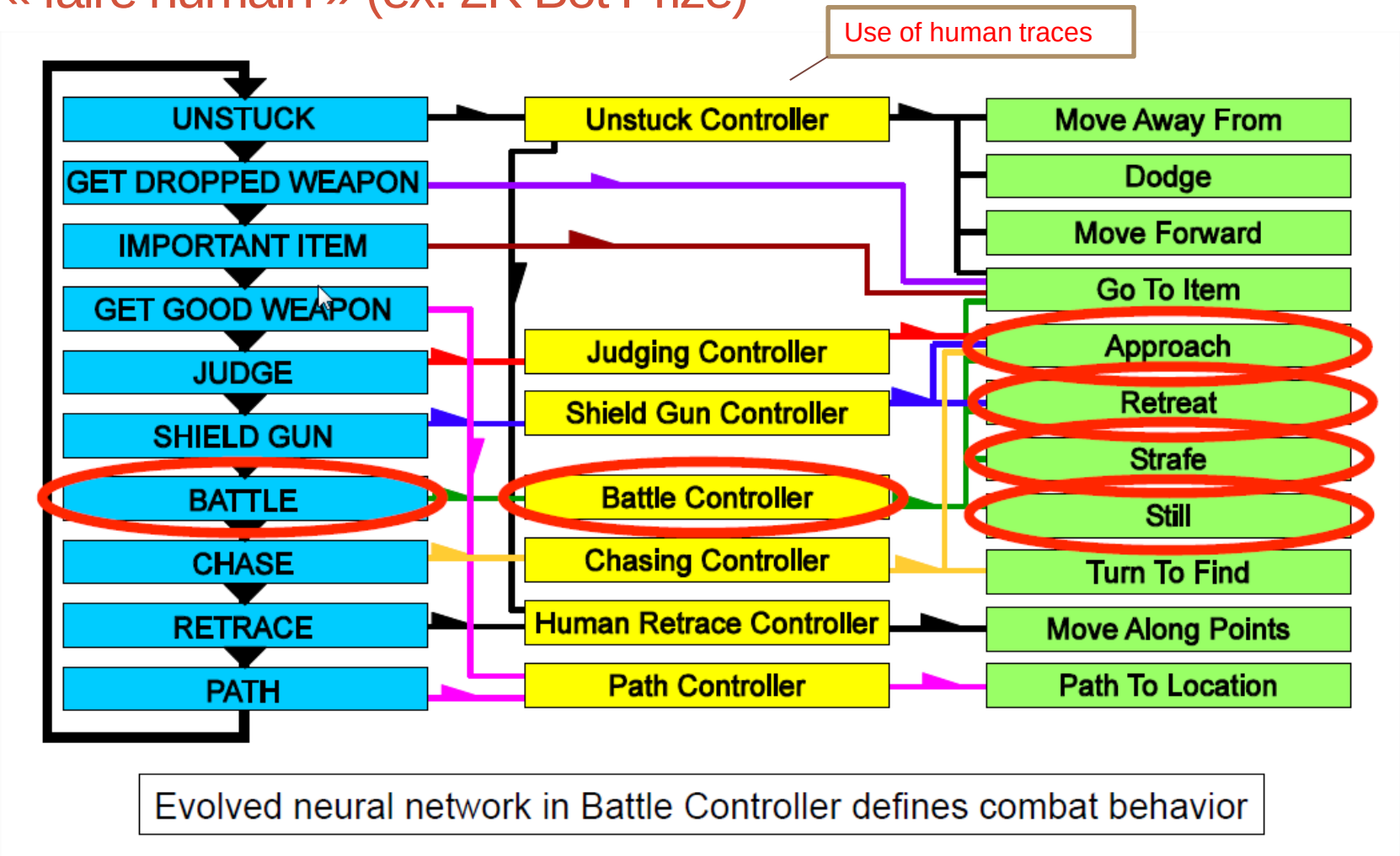


[Pillosu et al. 2009] AI Game Dev

La coordination de plusieurs agents se fait en conditionnant le nombre: quand deux agents sont disponible pour le Flanking, le comportement est ouvert pour eux



La recherche s'occupe de réaliser des bots plus réalistes à base d'apprentissage, avec un double objectif de performance et de « faire humain » (ex: 2K Bot Prize)



[Schrum et al. 2009] UT^2: Human-like Behavior via **Neuroevolution** of Combat Behavior and **Replay of Human Traces**

Humanness results

Most human bots

bot name	humanness %
UT^2	21.0526 %
HumanLikeBot	17.6471 %
ICE-WCCI2012	7.6923 %

Most human humans

player name	humanness %
Craig Speelman	40.0000 %
Chris Holme	21.4286 %
John Weise	11.1111 %
Samaneh Rastegari	9.0909 %

Most human epic bots

skill level	humanness %
1	33.3333 %
4	28.5714 %
2	11.1111 %
3	0.0000 %

Judging results

Best bot judges

bot name	accuracy %
UT^2	53.8462 %
HumanLikeBot	51.2195 %
ICE-WCCI2012	40.0000 %

Best human judges

human name	accuracy %
John Weise	63.4146 %
Chris Holme	61.9048 %
Samaneh Rastegari	56.8182 %
Craig Speelman	50.0000 %

[Schrum et al. 2009] UT^2: Human-like Behavior via **Neuroevolution** of Combat Behavior and **Replay of Human Traces**

On peut aussi utiliser de la méta-optimisation

[Amato et Shani AAMAS 2010]

- Contexte: Civilization IV
  - API libre pour développer des IA
  - Caractères existants
- Objectif:
  - choix dynamique entre 4 « caractères » au cours d'une partie
  - Dans tel état de la partie, vaut-il mieux que je joue comme Gandhi, Washington, Frederick II ou Gengis Kahn?

- Méthode:
  - Q-Learning tous les 10 tours sur la stratégie à choisir
  - 2 QL améliorés (construction dynamique d'un MDP du monde, avec ou sans features)
  - 4 états à 3 valeurs: diff. Population, Territoire, Millitaire, Terrain disponible
  - Récompense: diff. Score
- Résultat:
  - Apprentissage plus efficace que le fixe et l'aléatoire après 50/100 parties d'apprentissage
  - Apprentissage de modèle efficace

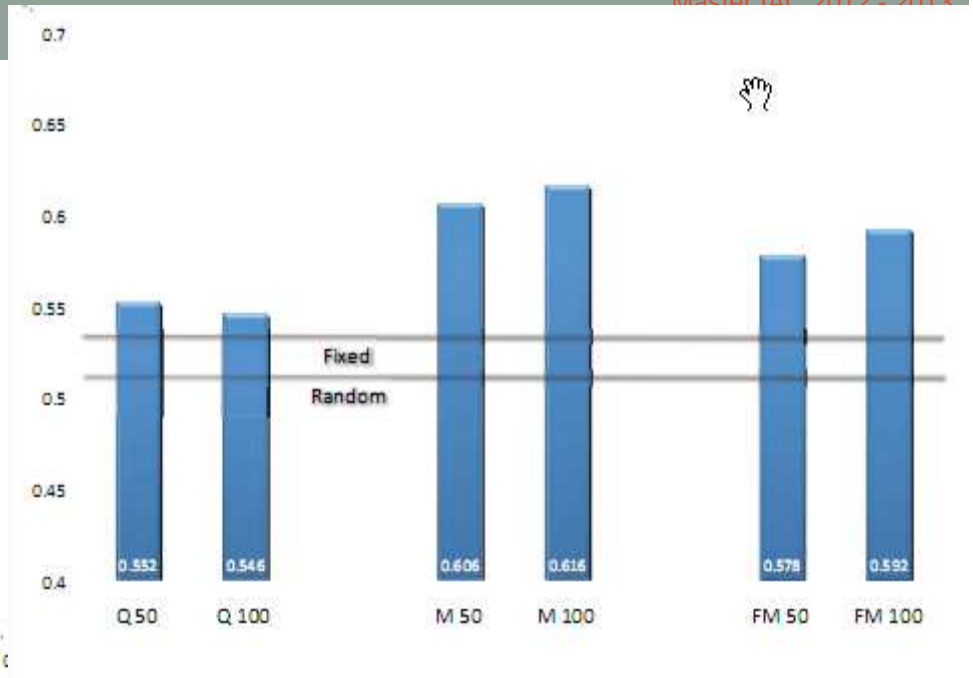


Figure 4: Results of Frederick learning to play against Washington

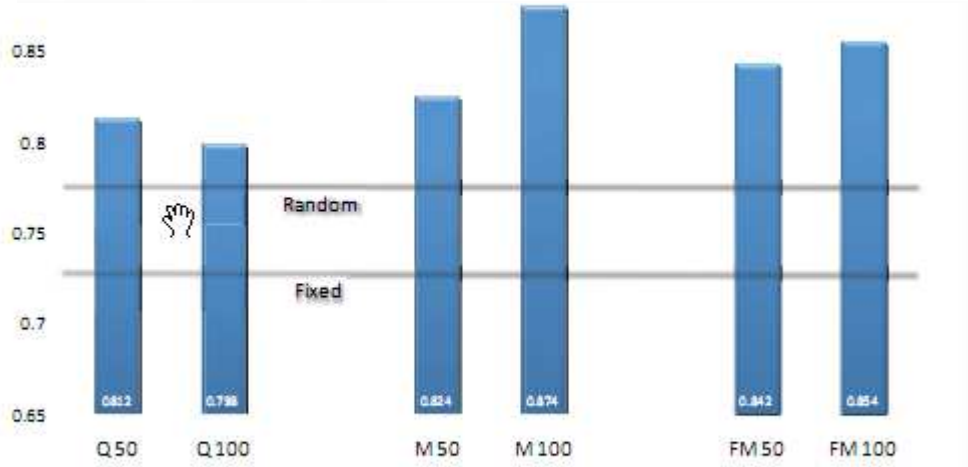
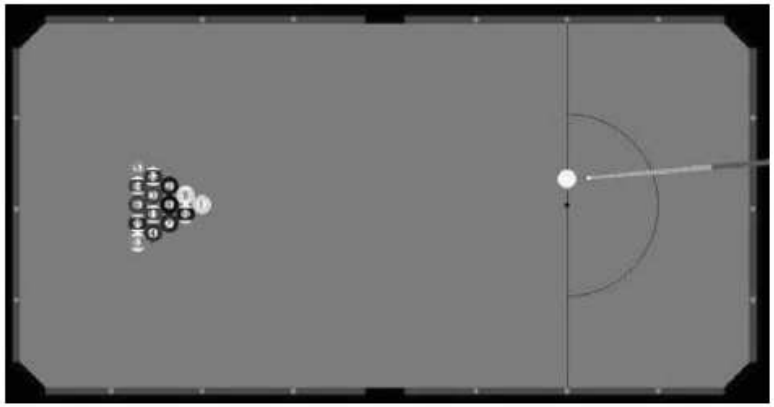


Figure 5: Results of Gandhi learning to play against Genghis Kahn

Problème inverse: quel jeux proposé pour proposer un challenge interessant?

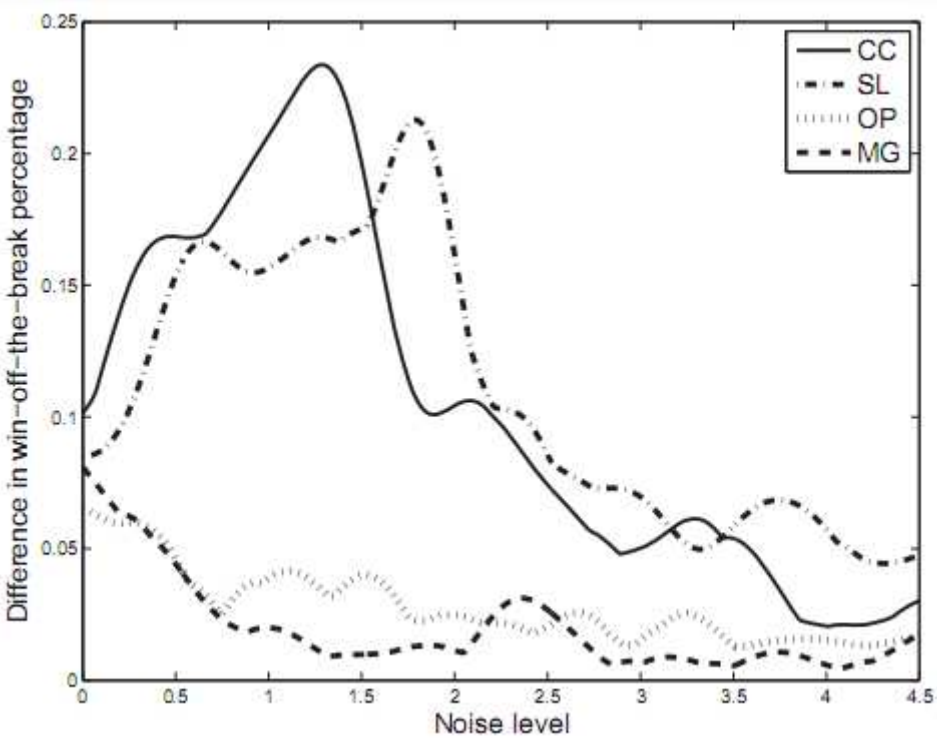


- [Archibald et al. 2010] AAMAS

- 5 paramètres par coup (vitesse, orientation, angle, position)
- Un bruit gaussien est ajouté sur chaque paramètre
  - Quel doit être sa valeur pour forcer les agent à jouer stratégique?

- 4 types d'agents
  - CC: 25 à 100 simulation par coup, recherche plus ou moins profonde dans l'arbre selon l'intérêt du coup. Les 20 meilleurs sont approfondis
  - SL: identique sans l'approfondissement
  - OP: planificateur optimiste
  - MG: MachineGunner sans planification

- Résultat:
  - Pour identifier les agents « intelligent » un faible niveau de bruit est utile et nécessaire

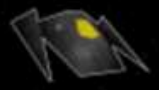
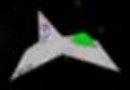
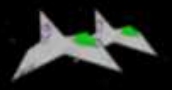
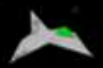


Modèles d'agent

- Introduction
- Modèle Zero-Intelligence
- Modèles réactifs
- **Modèles interactionnel**
- Agents cognitifs
- Modèles multi-niveaux
- Modèles reflexifs

Modèle interactionnel: les interactions determinent le comportement

- http://www.lifl.fr/SMAC/projects/galaxian/

target / source	Ø						
	SeekTarget(0) Fire(8) Explode(9)		Confront(1000,2) Intercept(1000,3) Engage(1000,4) Escape(1000,5)				
	SeekTarget(0) Fire(8) Explode(9)	Confront(1000,2) Intercept(1000,3) Engage(1000,4) Escape(1000,5)	CreateSquad(30,2)				Join(100,6) Follow(1000,7)
							FightBack(200,0)
	LaunchFighter(0)						
	LaunchFighter(0)						
	Disband(0) Fire(3)			Engage(5000,1)			Merge(50,2)

Modèle interactionnel: IODA

- http://www.lifl.fr/SMAC/projects/ioda/index.php
- [Kubera et al. 2008]…

- Principe:
  - Le comportement et la dynamique du système sont définis au niveau des interactions entre les agents

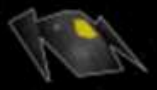
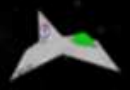
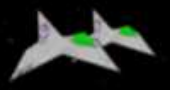
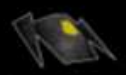
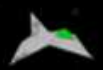
- Nombreuses applications
  - Modèles simples (fourmi, …)
  - Supermarché
  - Combat spatial

- et extensions
  - Module pour NetLogo
  - Extension multi-niveau Padawan

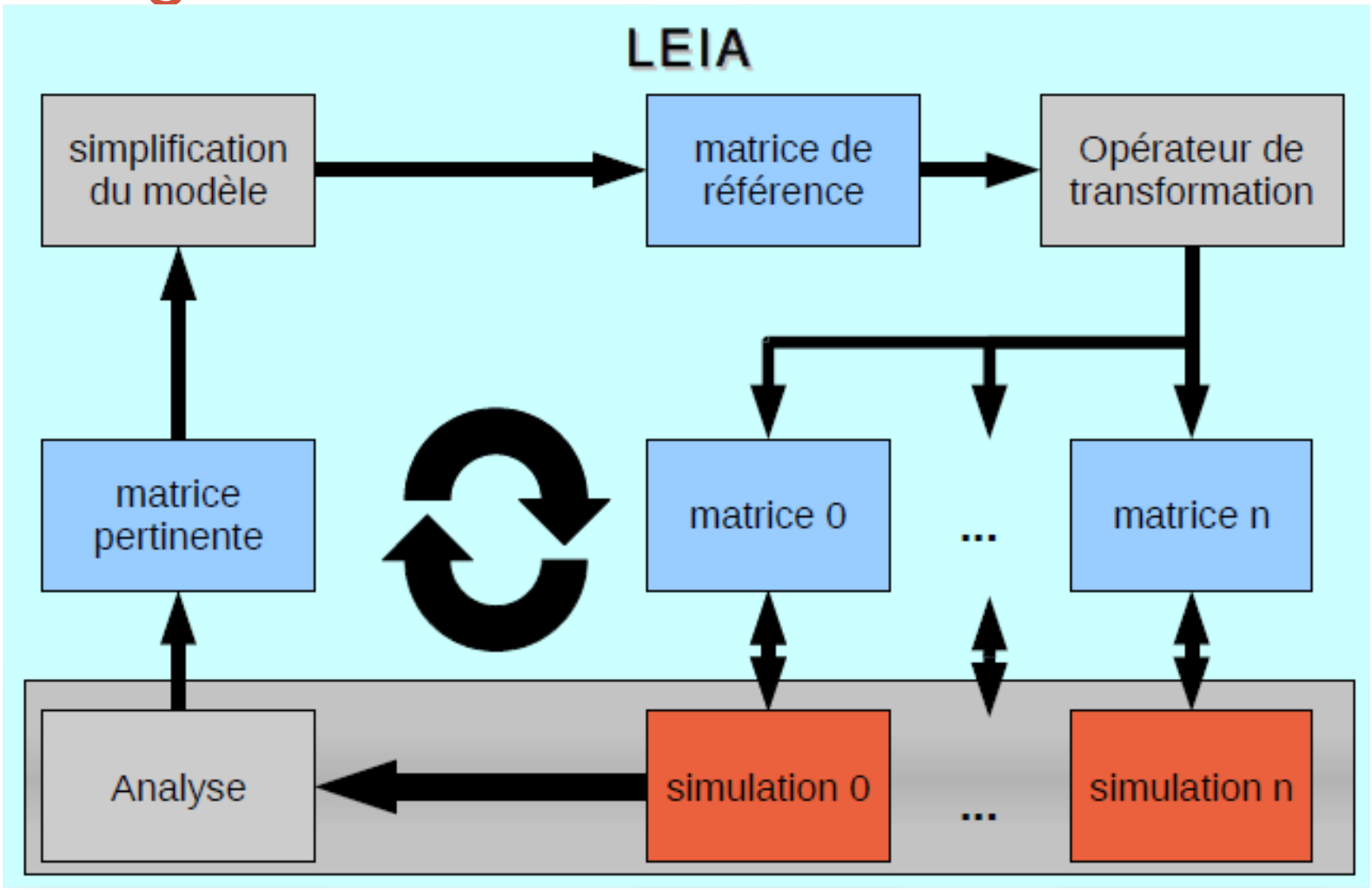
Source \ Target	∅	Forum	Building	Person	Resource	Farm
Forum : Building	(*Form Peon*) (*Form Warrior*) (*Place new Farm*)					
Peon : Person	(*Wander*)	(*Get Forum Goal*) (*Go Towards*) (*Release Goal*) (*Store Resource*) (*Report Farm Built*)			(*Get Harvest Goal*) (*Go Towards*) (*Release Goal*) (*Harvest*)	(*Get Build Goal*) (*Go Towards*) (*Release Goal*) (*Build*)
Warrior : Person	(*Wander*)		(*Get Attack Goal*) (*Go Towards*) (*Release Goal*) (*Attack*)	(*Get Attack Goal*) (*Go Towards*) (*Release Goal*) (*Attack*)		
Tree : Resource	(*Expand*)					

Modèle interactionnel

- http://www.lifl.fr/SMAC/projects/galaxian/

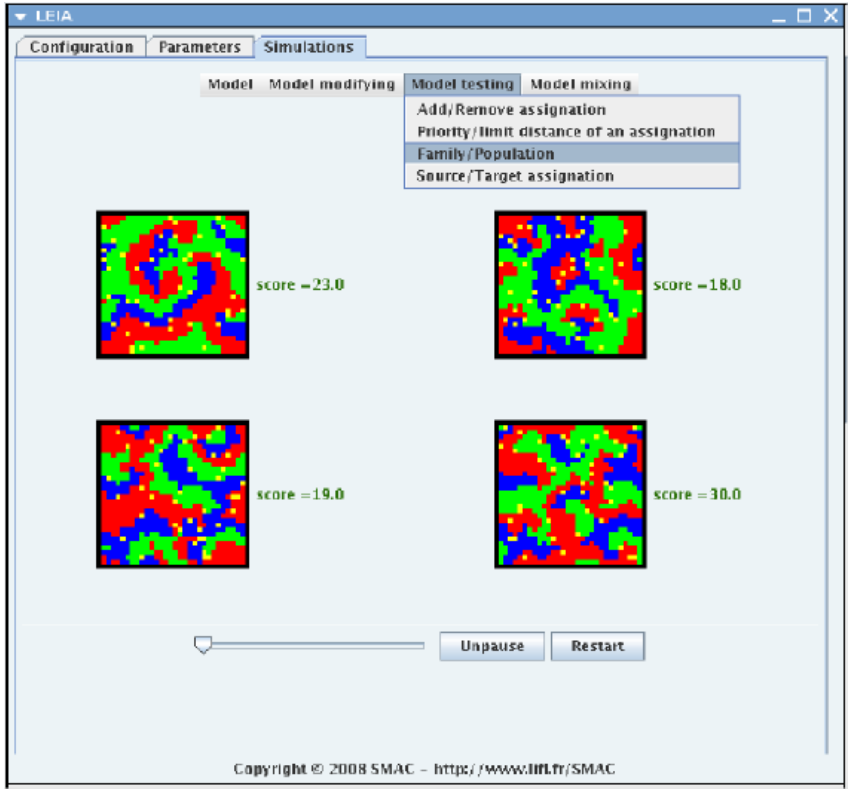
source \ target	Ø						
	SeekTarget(0) Fire(8) Explode(9)		Confront(1000,2) Intercept(1000,3) Engage(1000,4) Escape(1000,5)				
	SeekTarget(0) Fire(8) Explode(9)	Confront(1000,2) Intercept(1000,3) Engage(1000,4) Escape(1000,5)	CreateSquad(30,2)				Join(100,6) Follow(1000,7)
							FightBack(200,0)
	LaunchFighter(0)						
	LaunchFighter(0)						
	Disband(0) Fire(3)			Engage(5000,1)			Merge(50,2)

LEIA: analyse automatique de configuration



LEIA: analyse automatique de configuration

- Évaluation automatique de l'intérêt d'un ensemble de paramètres à partir de:
  - Activité des agents
  - Évolution de l'environnement
  - Stabilité et cohésion
  - Densité
  - Fréquence d'utilisation des interactions
- Identification de phénomènes cyclique
- Exploration visuelle de l'espace des paramètres



Modèles d'agent

- Introduction
- Modèle Zero-Intelligence
- Modèles réactifs
- Modèles interactionnel
- **Agents cognitifs**
- Modèles multi-niveaux
- Modèles reflexifs

1ère solution extrême: L'agent rationnel (Computo Economicus)

- Action=ArgMax(Utilité/Etat Actuel)

- Comportement réactif (état perçu) mais notion d'objectif
- Très utile pour les méthodes de négociation automatique

- Principale limite:
  - Pas de prise en compte de la complexité algorithmique
  - Le ArgMax nécessite dans le cas général une évaluation de toutes les possibilités
  - Les problèmes d'information incomplete sont reportés dans la fonction d'utilité

2ème solution extrême: le planificateur

- Données:
  - État complet du monde
  - Objectif final

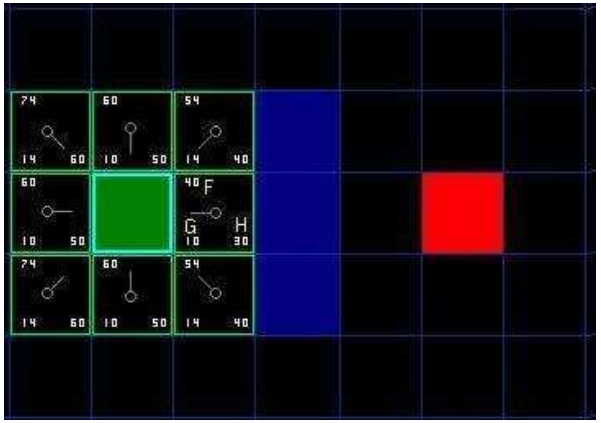
- Un planificateur (STRIPS, …) déduit l'ensemble des actions à effectuer pour y arriver.
- En théorie, résultat optimal pour l'agent

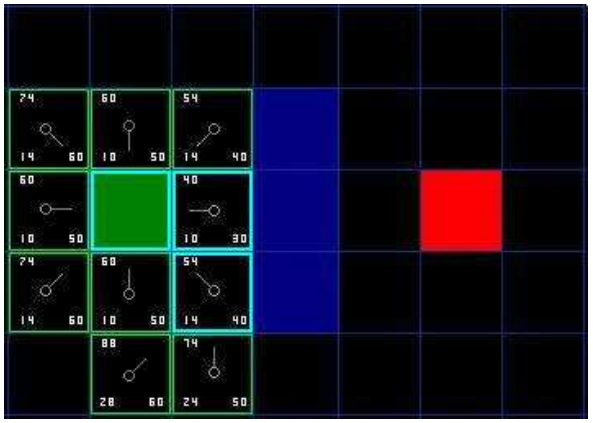
- Irréaliste comme solution unique dans la plupart des cas:
  - Ne prend pas en compte l'information imparfaite
  - Nécessite un recalcul complet à chaque changement non prévu de l'état du monde, et en particulier après chaque action non anticipée d'un autre agent éventuel.

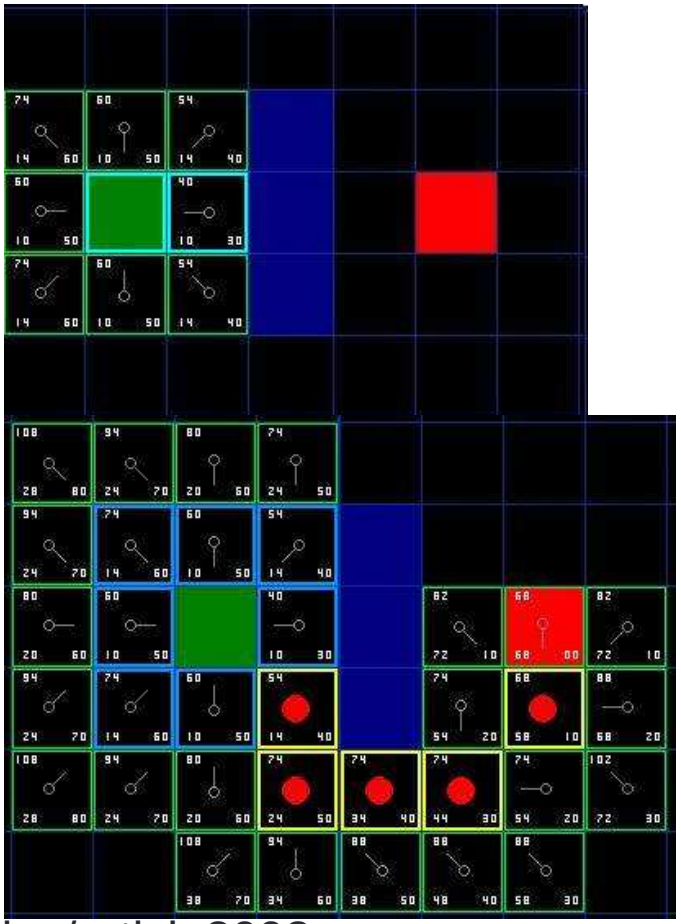
- Utile pour la partie « raisonnement sur les plans »

A*: optimisation

- Même plus de représentation symbolique
- Juste une fonction de cout et une heuristique d'évaluation

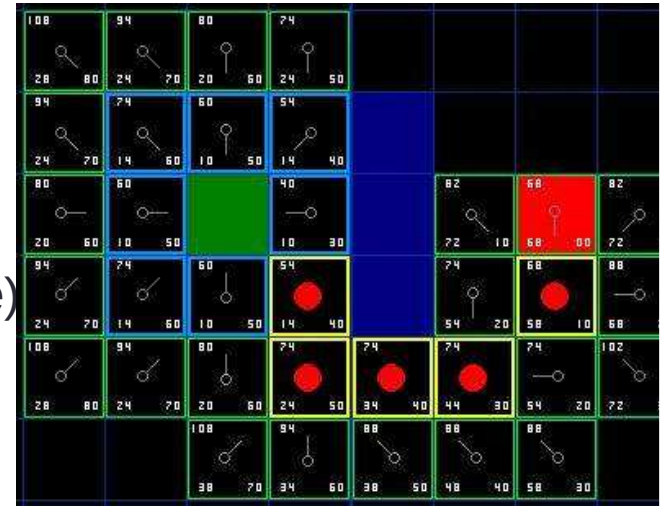






http://www.gamedev.net/reference/articles/article2003.asp

A*: principe



- F=G (chemin depuis l'origine)+H(estimation restante)

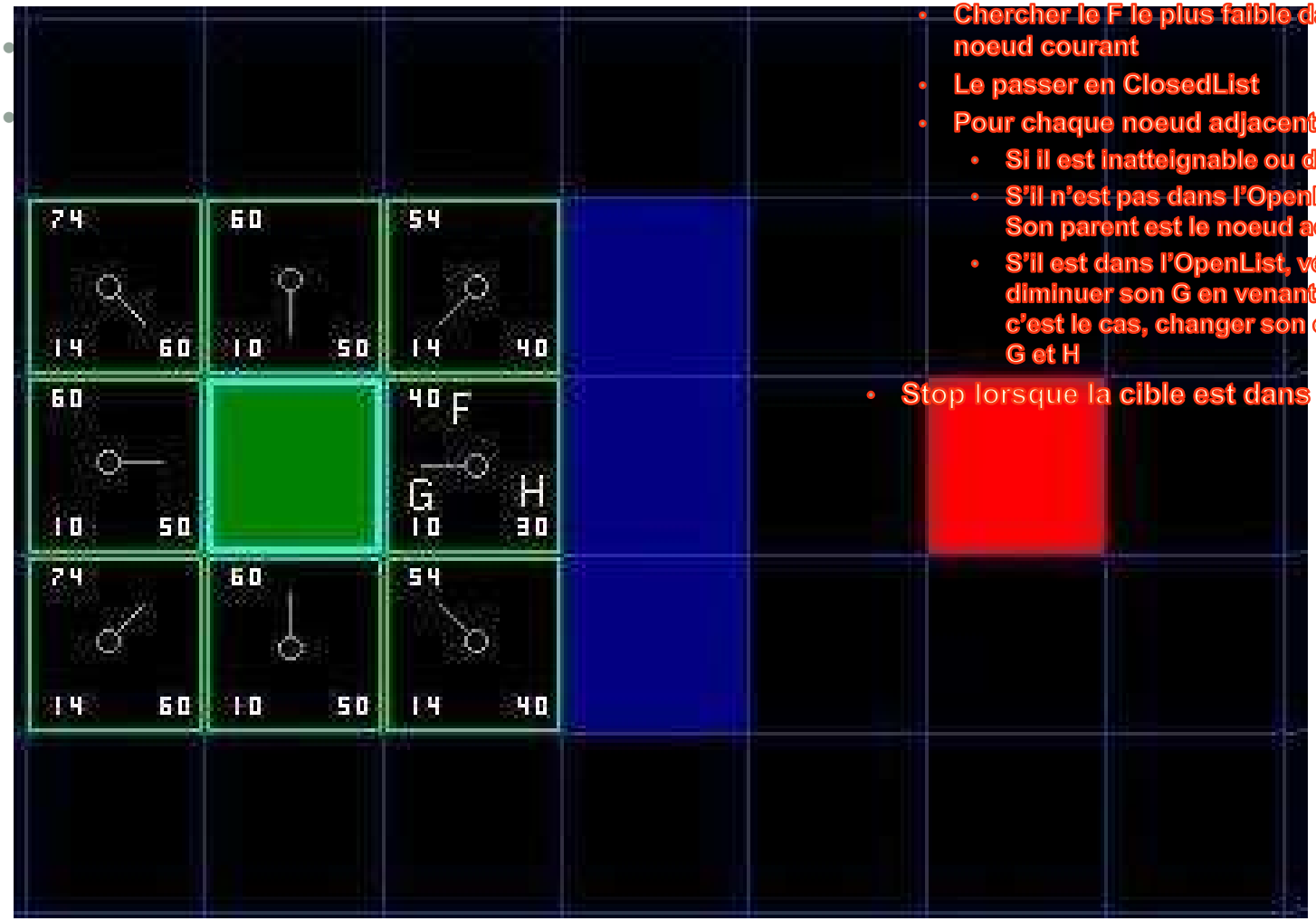
- Ajouter le noeud initial a l'OpenList
- Repeter:
  - Chercher le F le plus faible dans l'OpenList, c'est le noeud courant
  - Le passer en ClosedList
  - Pour chaque noeud adjacent au noeud courant:
    - Si il est inatteignable ou dans la ClosedList, ignorer
    - S'il n'est pas dans l'OpenList, l'ajouter a l'OpenList. Son parent est le noeud actuel. Calculer F, G et H.
    - S'il est dans l'OpenList, vérifier si il est possible de diminuer son G en venant du noeud courant. Si c'est le cas, changer son origine et mettre a jour F, G et H
- Stop lorsque la cible est dans la ClosedList

- Nombreuses publications sur des améliorations

A*: optimisation

- F=G (chemin depuis l'origine)+H(estimation restante)

- Ajouter le noeud initial a l'OpenList
- Repeter:
  - Chercher le F le plus faible dans l'OpenList, c'est le noeud courant
  - Le passer en ClosedList
  - Pour chaque noeud adjacent au noeud courant:
    - Si il est inatteignable ou dans la ClosedList, ignorer
    - S'il n'est pas dans l'OpenList, l'ajouter a l'OpenList. Son parent est le noeud actuel. Calculer F, G et H.
    - S'il est dans l'OpenList, vérifier si il est possible de diminuer son G en venant du noeud courant. Si c'est le cas, changer son origine et mettre a jour F, G et H
  - Stop lorsque la cible est dans la ClosedList

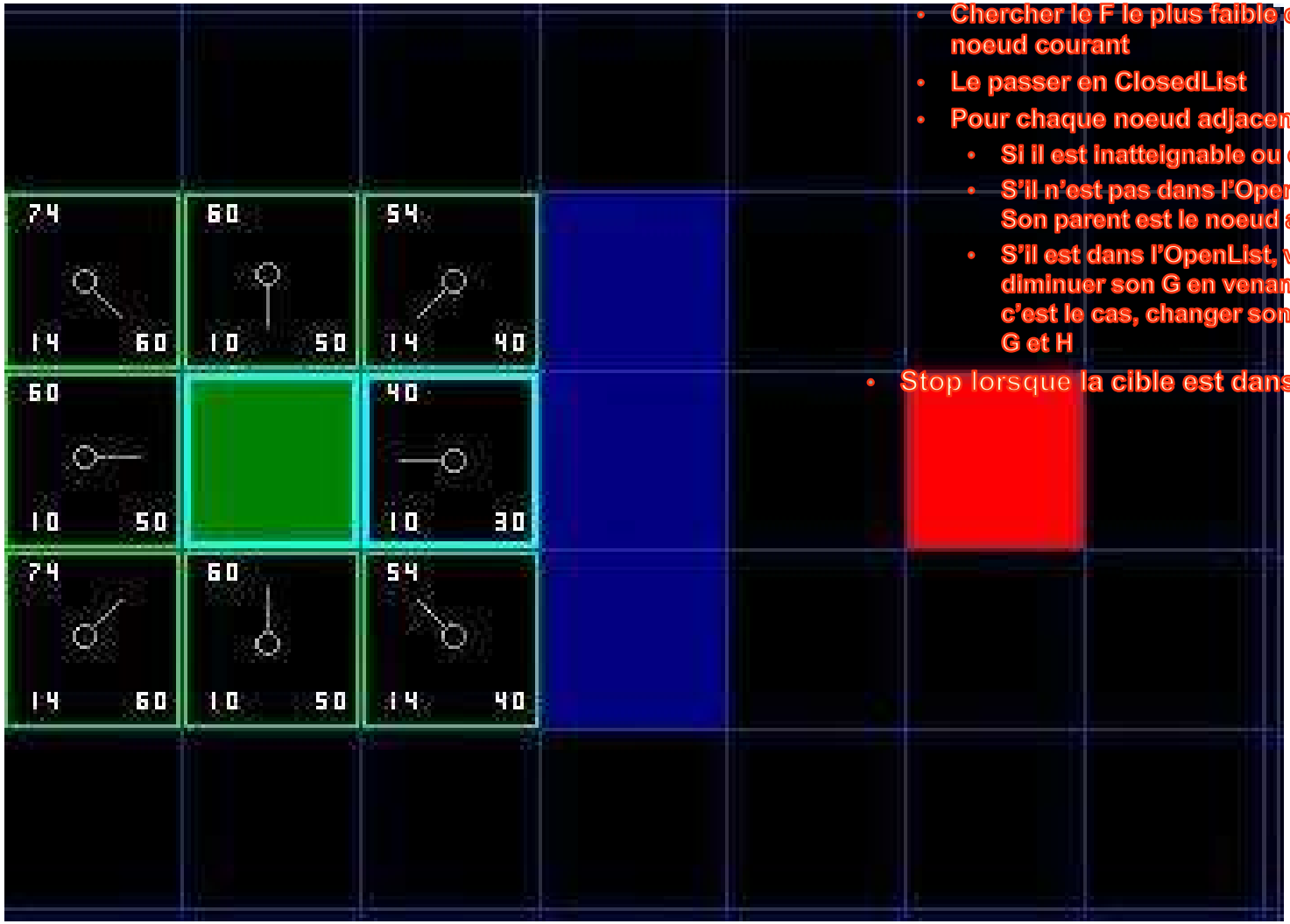


http://www.gamedev.net/reference/articles/article2003.asp

A*: optimisation

- F=G (chemin depuis l'origine)+H(estimation restante)

- Ajouter le noeud initial a l'OpenList
- Repeter:
  - Chercher le F le plus faible dans l'OpenList, c'est le noeud courant
  - Le passer en ClosedList
  - Pour chaque noeud adjacent au noeud courant:
    - Si il est inatteignable ou dans la ClosedList, ignorer
    - S'il n'est pas dans l'OpenList, l'ajouter a l'OpenList. Son parent est le noeud actuel. Calculer F, G et H.
    - S'il est dans l'OpenList, vérifier si il est possible de diminuer son G en venant du noeud courant. Si c'est le cas, changer son origine et mettre a jour F, G et H
- Stop lorsque la cible est dans la ClosedList

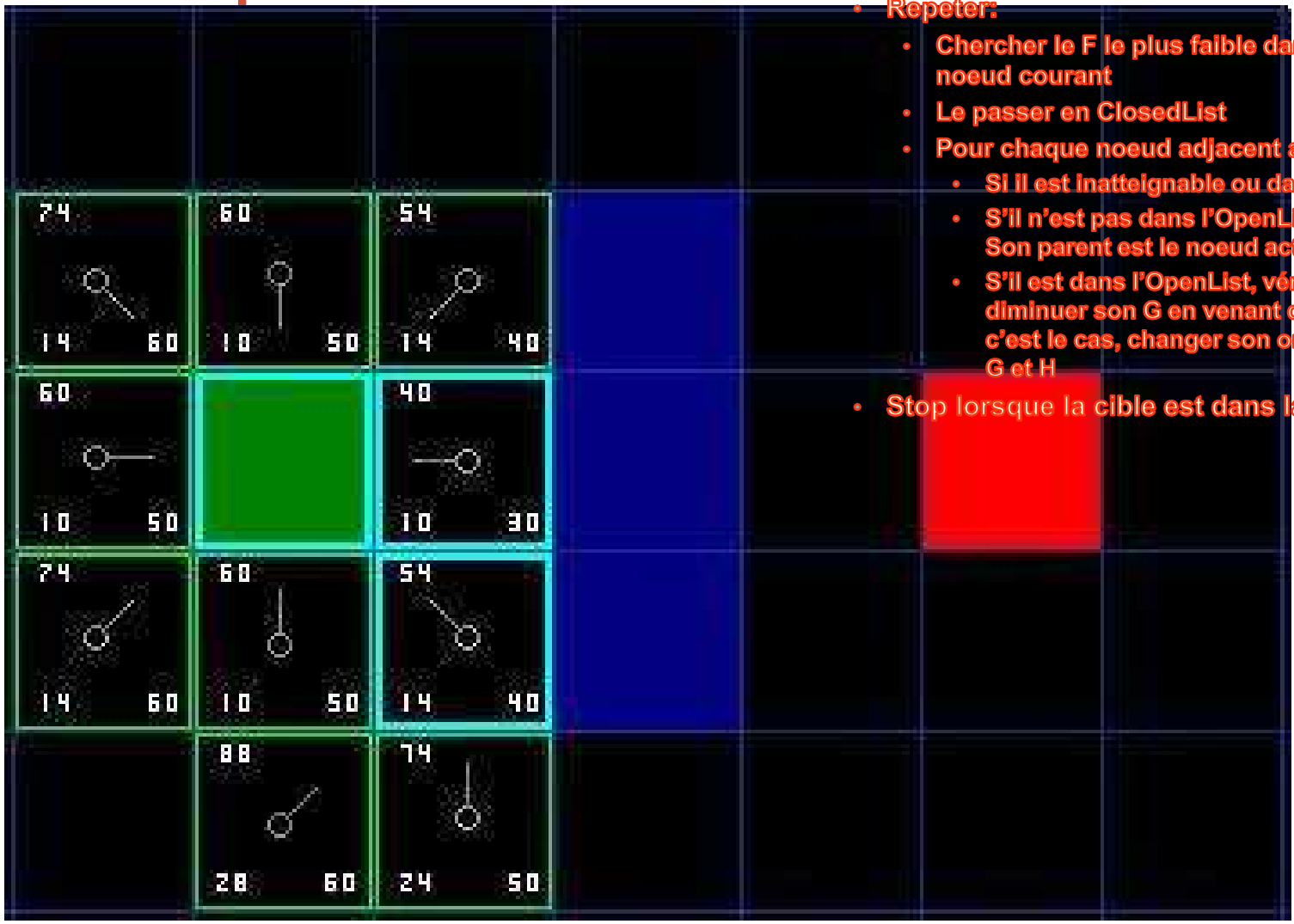


http://www.gamedev.net/reference/articles/article2003.asp

A*: optimisation

- F=G (chemin depuis l'origine)+H(estimation restante)

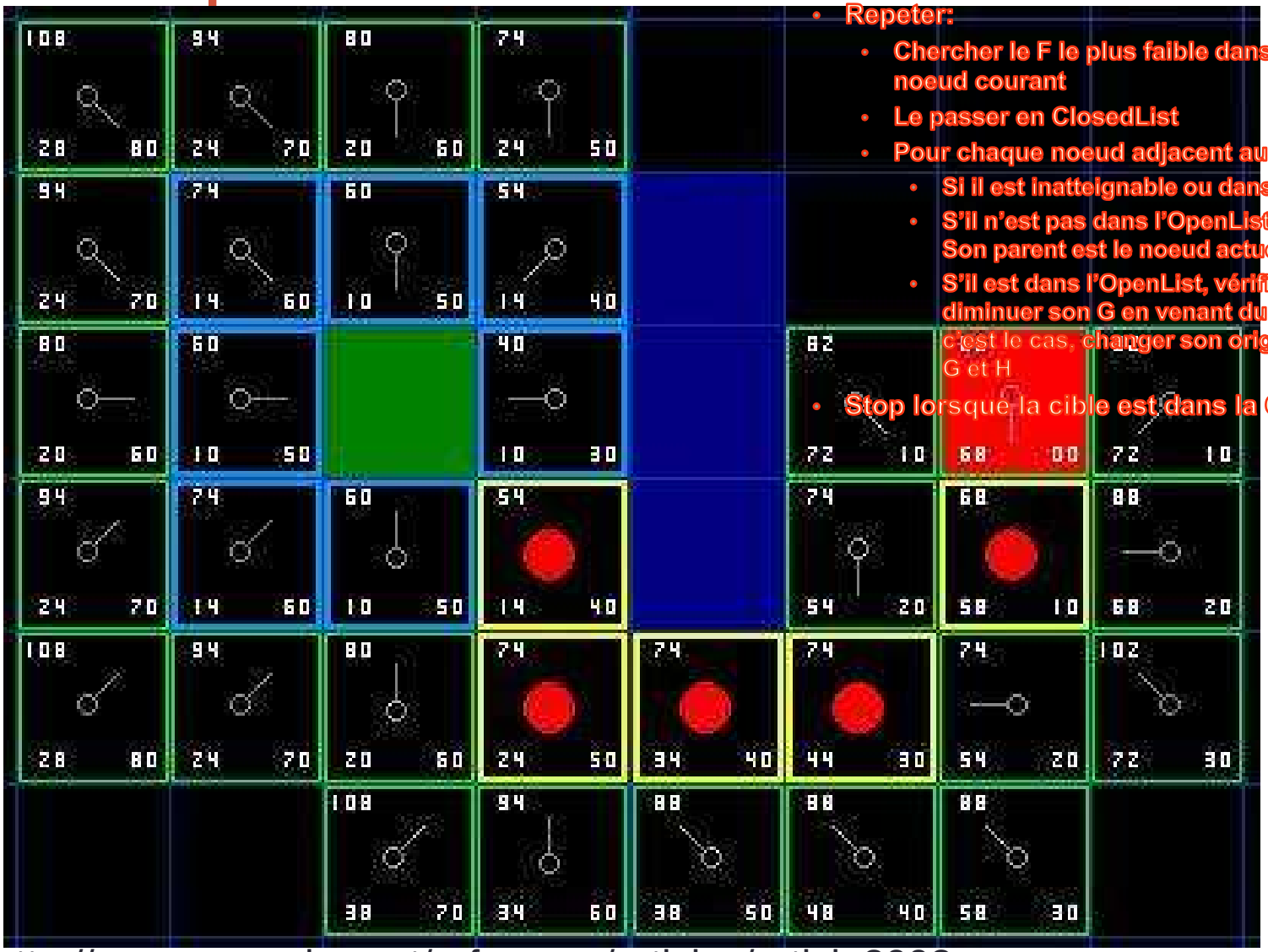
- Ajouter le noeud initial a l'OpenList
- Repeter:
  - Chercher le F le plus faible dans l'OpenList, c'est le noeud courant
  - Le passer en ClosedList
  - Pour chaque noeud adjacent au noeud courant:
    - Si il est inatteignable ou dans la ClosedList, ignorer
    - S'il n'est pas dans l'OpenList, l'ajouter a l'OpenList. Son parent est le noeud actuel. Calculer F, G et H.
    - S'il est dans l'OpenList, vérifier si il est possible de diminuer son G en venant du noeud courant. Si c'est le cas, changer son origine et mettre a jour F, G et H
- Stop lorsque la cible est dans la ClosedList



http://www.gamedev.net/reference/articles/article2003.asp

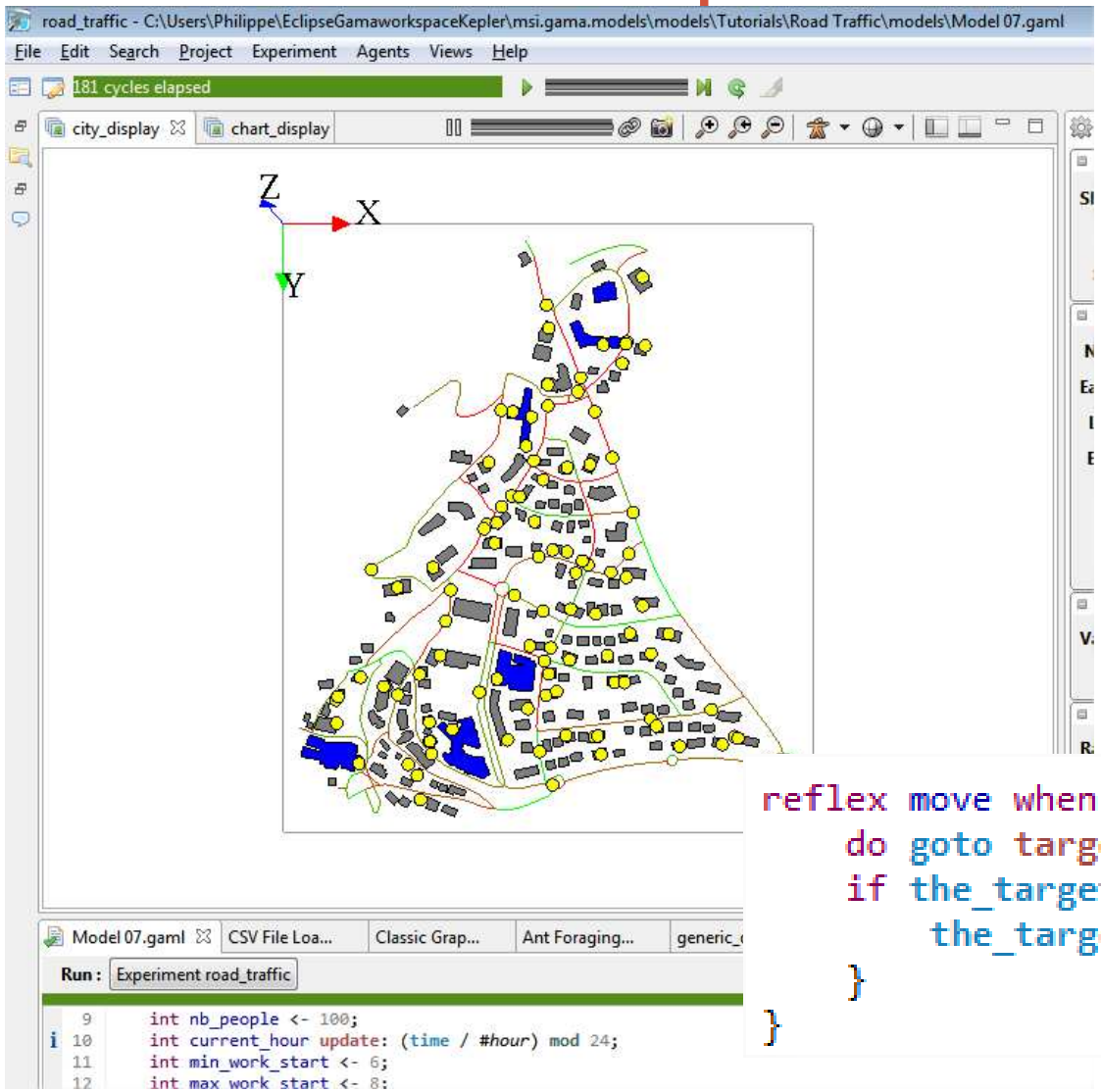
A*: optimisation

- F=G (chemin depuis l'origine)+H(estimation restante)
- Ajouter le noeud initial a l'OpenList
- Repeter:
  - Chercher le F le plus faible dans l'OpenList, c'est le noeud courant
  - Le passer en ClosedList
  - Pour chaque noeud adjacent au noeud courant:
    - Si il est inatteignable ou dans la ClosedList, ignorer
    - S'il n'est pas dans l'OpenList, l'ajouter a l'OpenList. Son parent est le noeud actuel. Calculer F, G et H.
    - S'il est dans l'OpenList, vérifier si il est possible de diminuer son G en venant du noeud courant. Si c'est le cas, changer son origine et mettre a jour F, G et H
- Stop lorsque la cible est dans la ClosedList



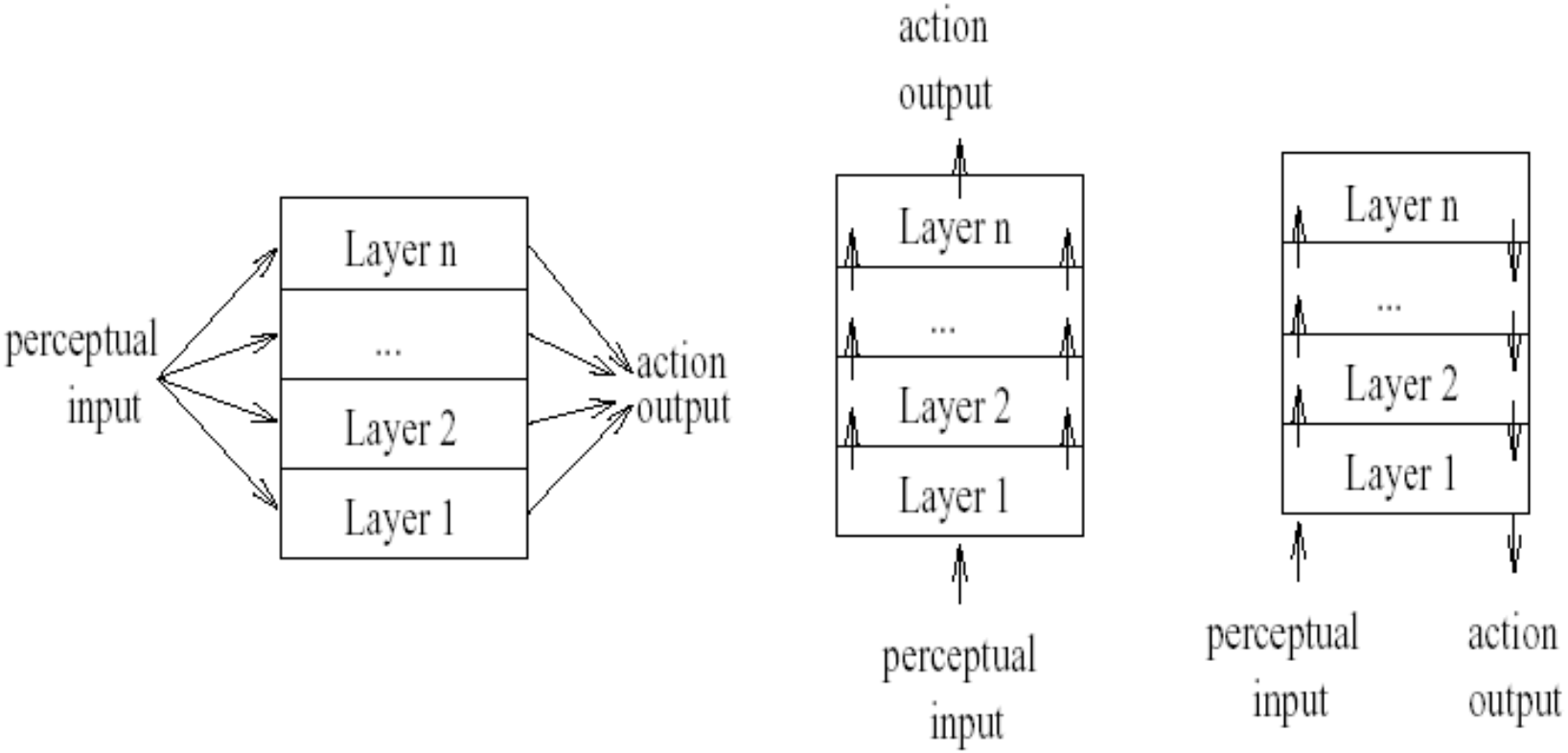
http://www.gamedev.net/reference/articles/article2003.asp

La planification, cela fonctionne bien, mais c'est lent et c'est proactif…




```
reflex move when: the_target != nil {
    do goto target: the_target on: the_graph ;
    if the_target = location {
        the_target <- nil ;
    }
}
```

Pour pouvoir utiliser la planification de manière efficace, un contrôle mixte est nécessaire, par exemple en multi niveaux



(a) Horizontal layering

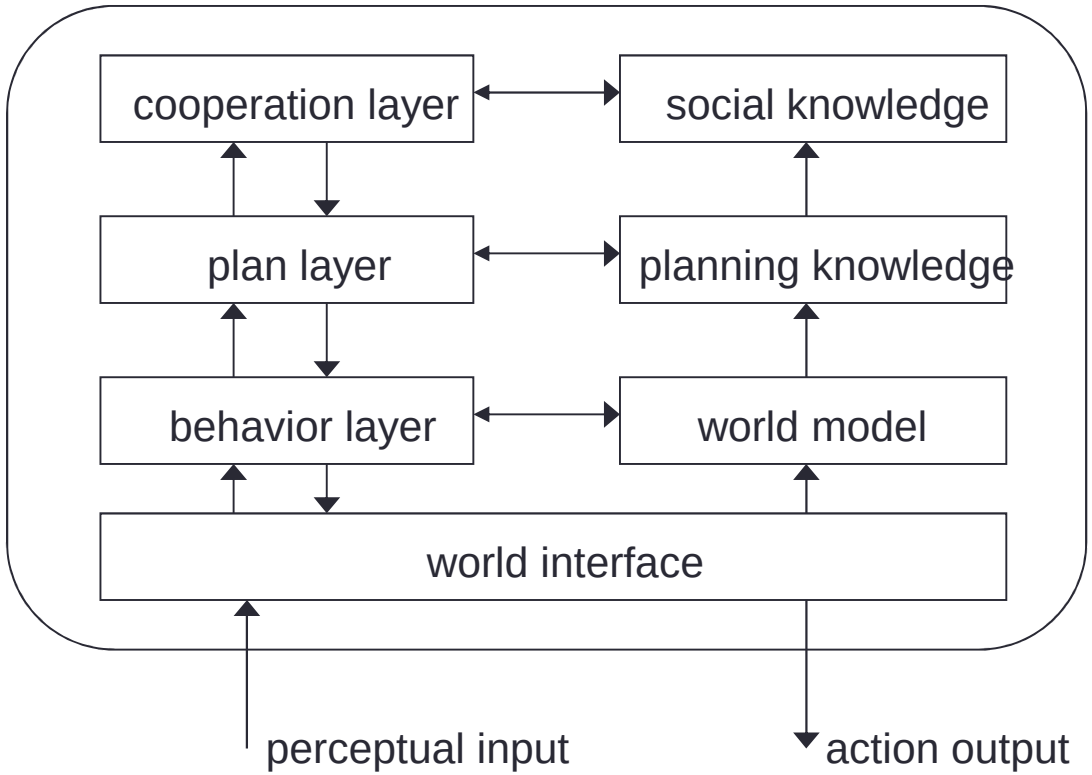
(b) Vertical layering (One pass control)

(c) Vertical layering (Two pass control)

[Wooldridge 09]

Avec Interrap, chaque niveau a un module de planification et l'infirmation remonte si le niveau inférieur est insuffisant

- [Muller et Pischel 93]

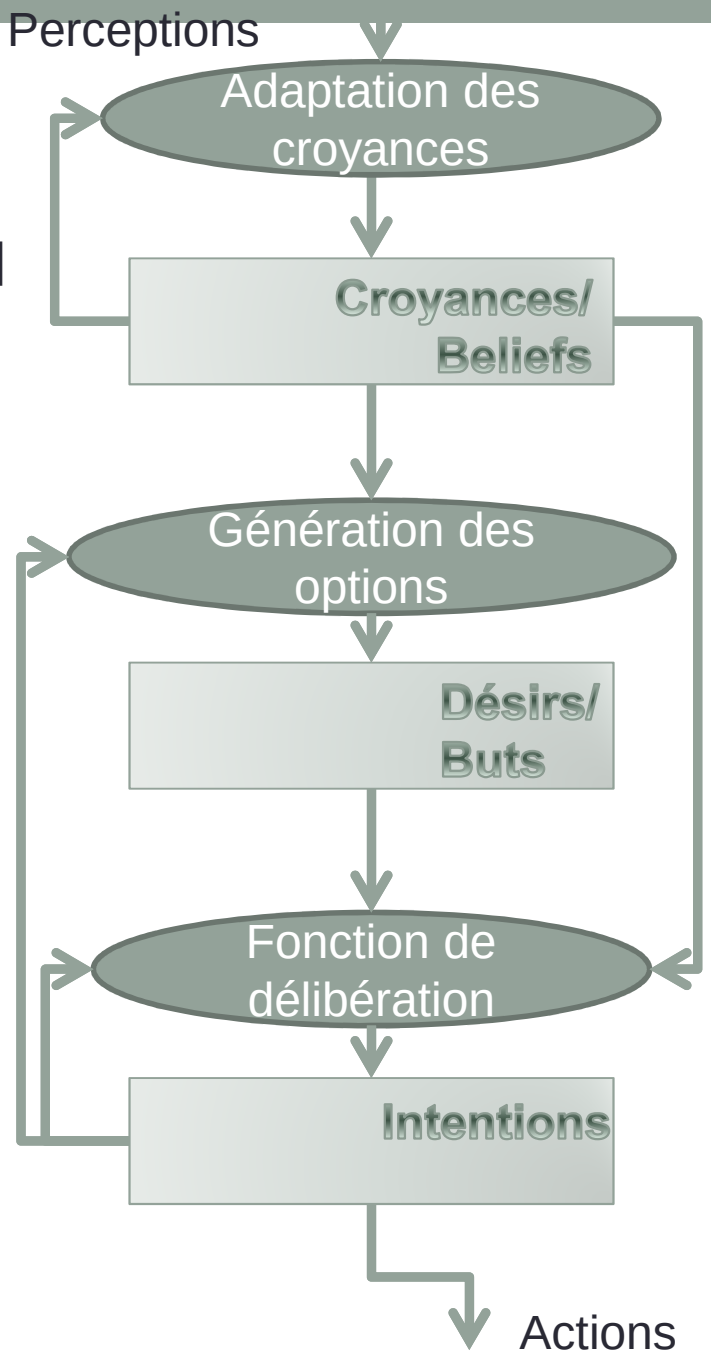


Un des modèles les plus développé: le modèle BDI

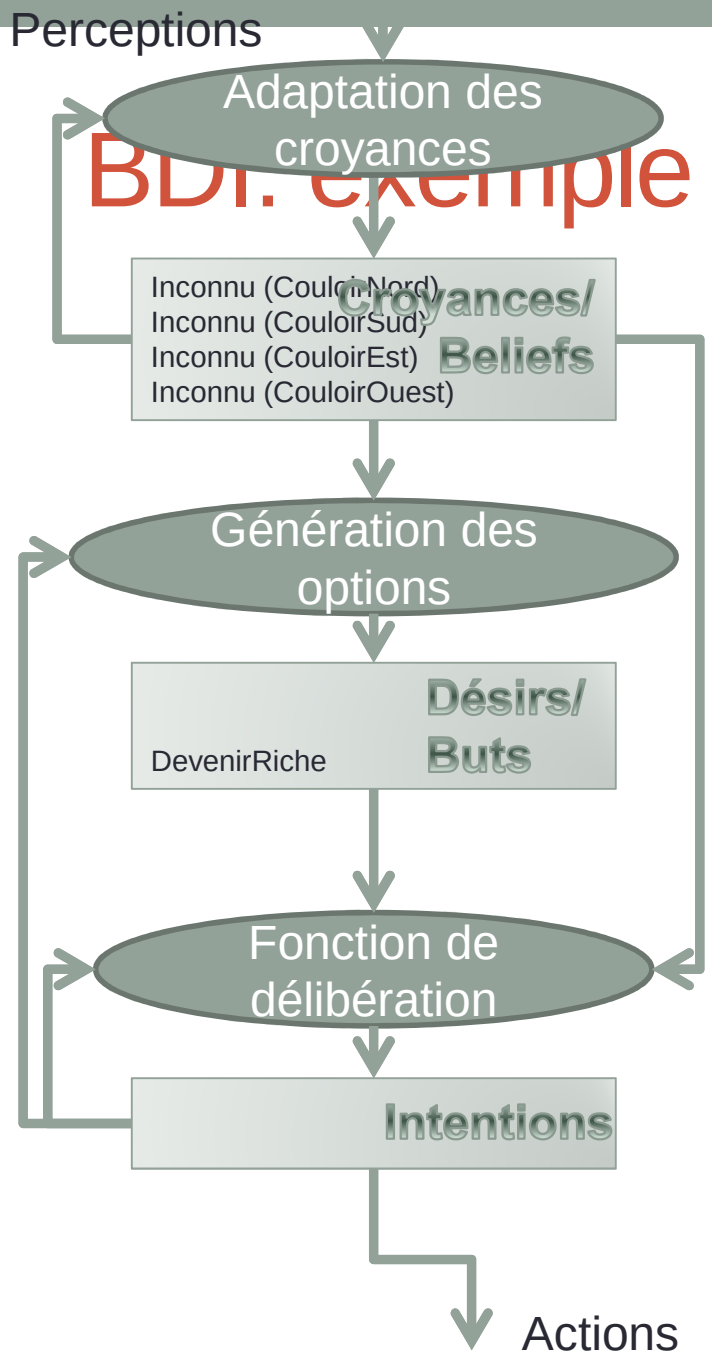
[Bratman 87,88]

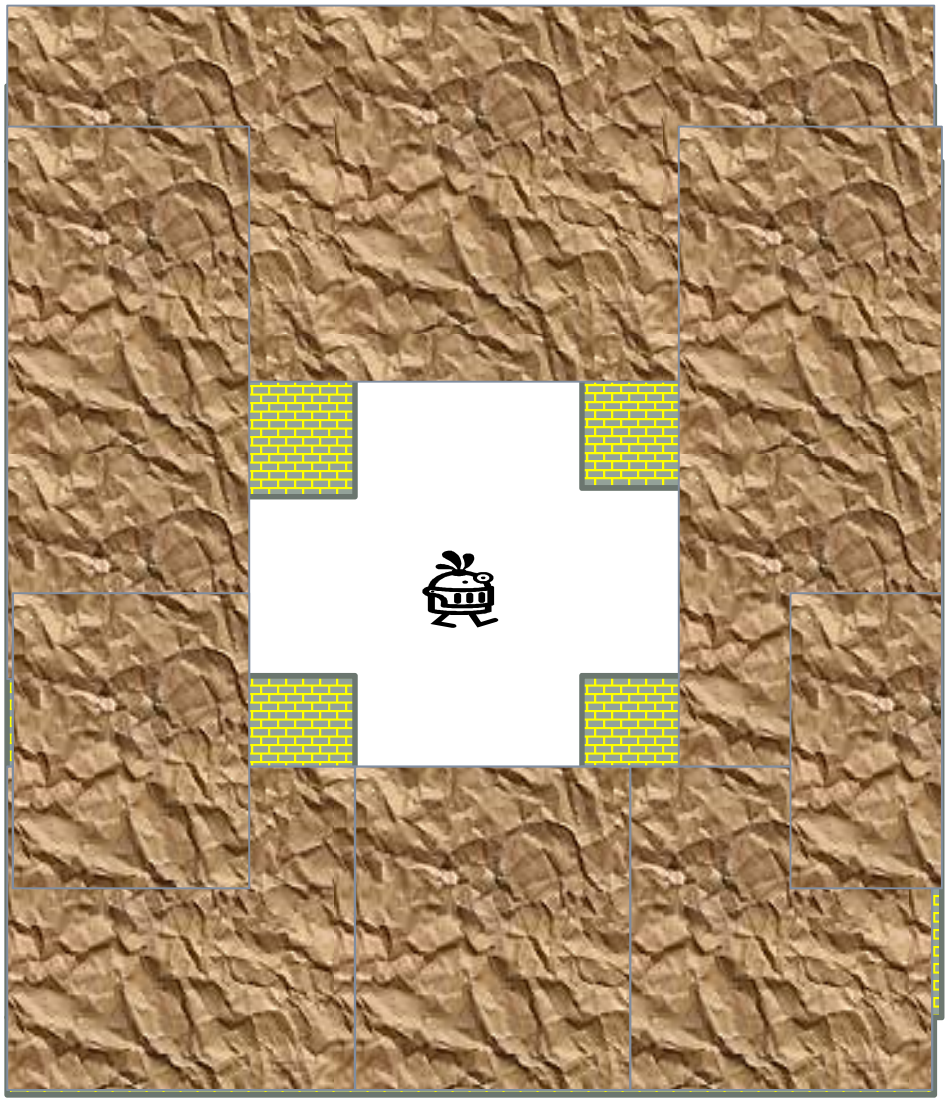
- Objectif:
  - Raisonner sur les objectif
  - Donner une solution pratique et robuste

- Beliefs: ce que je sais du monde
- Desires: ce que je souhaite
  - Goals/Buts: sous-ensemble cohérent de Desires
- Intentions: ce que je fais (plans en cours)

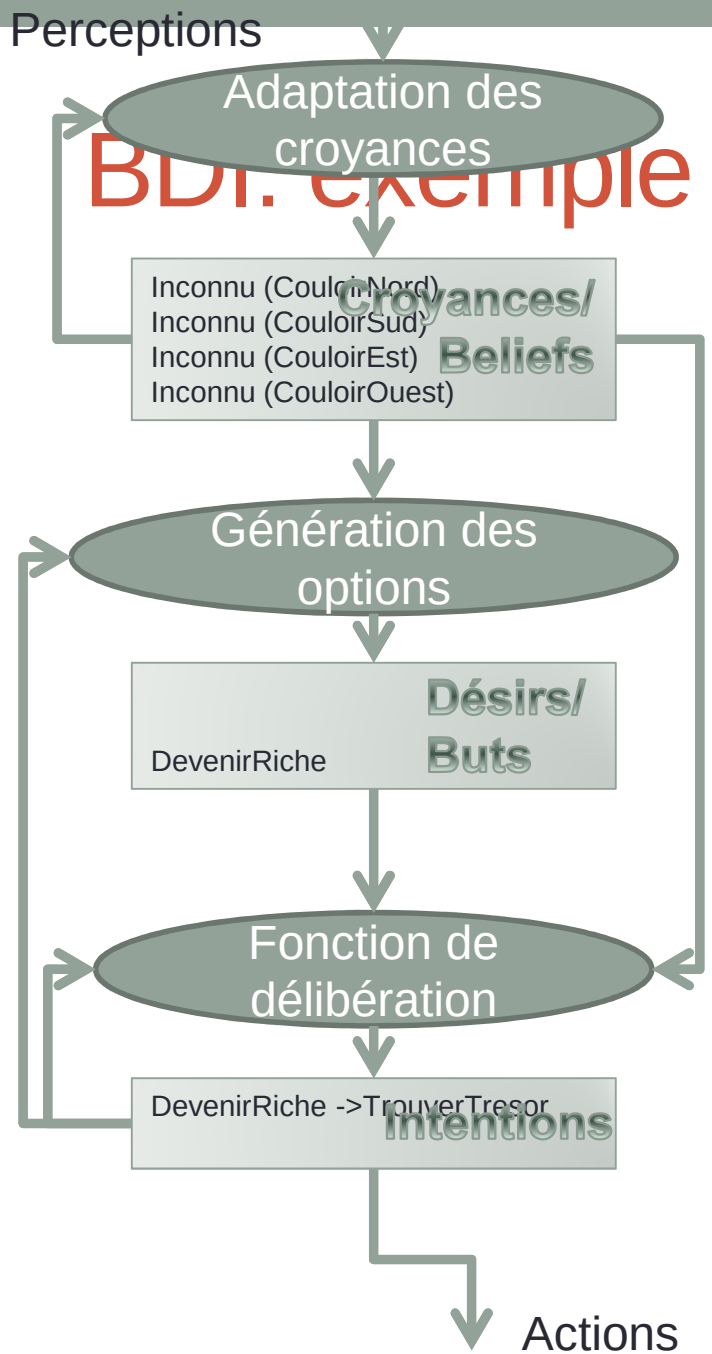


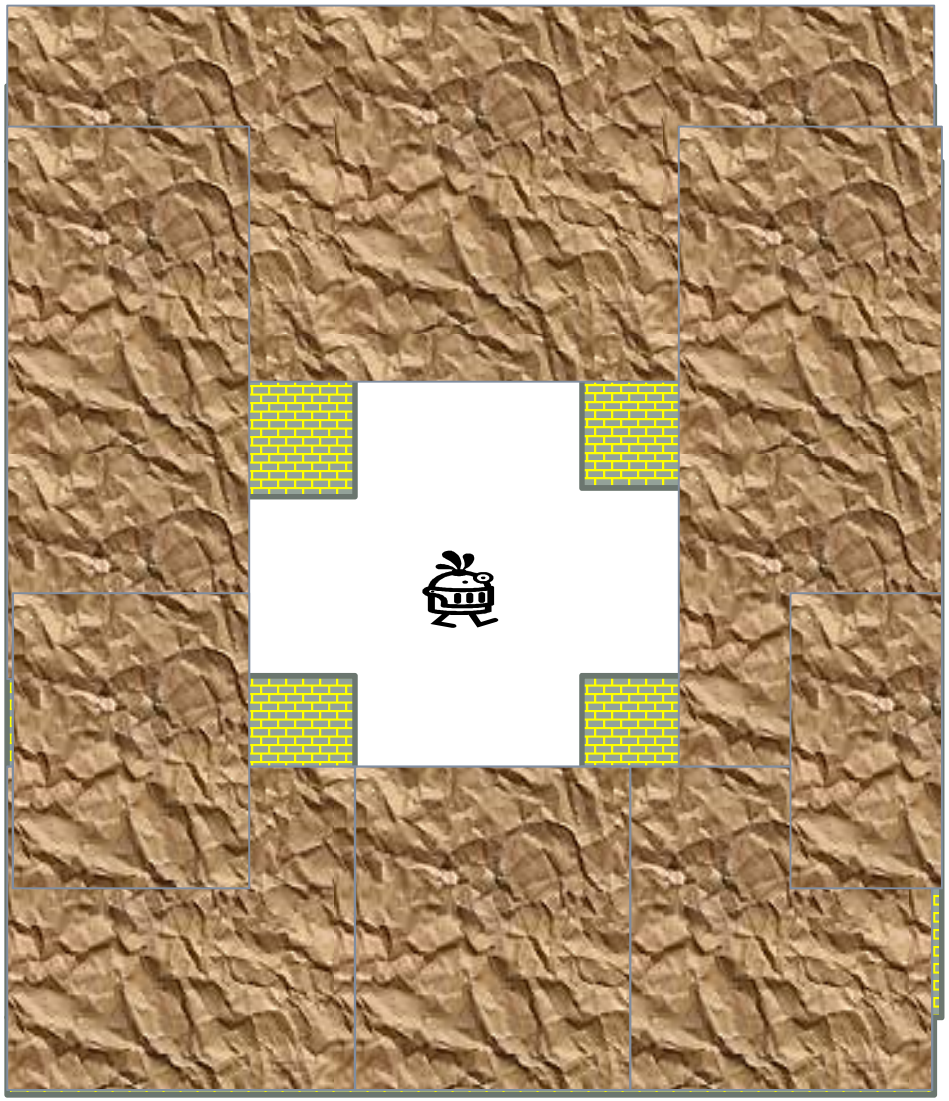
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Désirs/ Buts
Fonction de délibération
Intentions
Actions

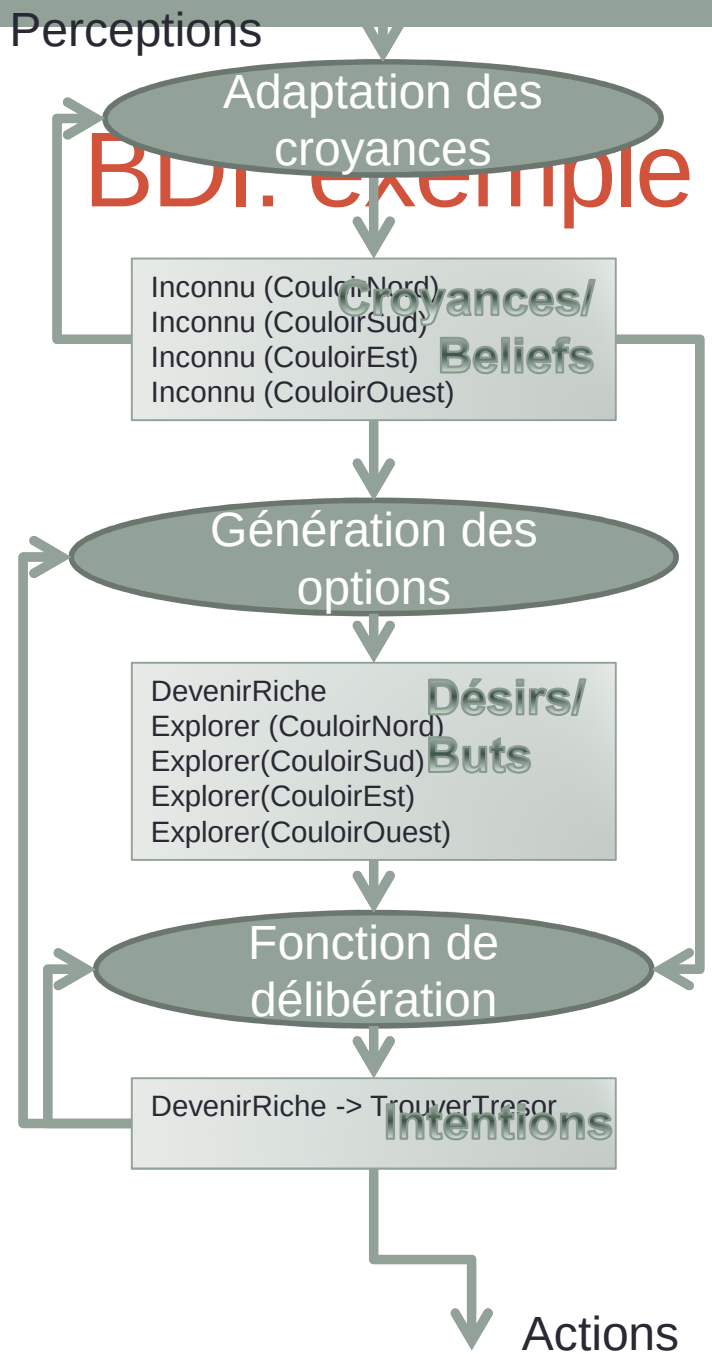


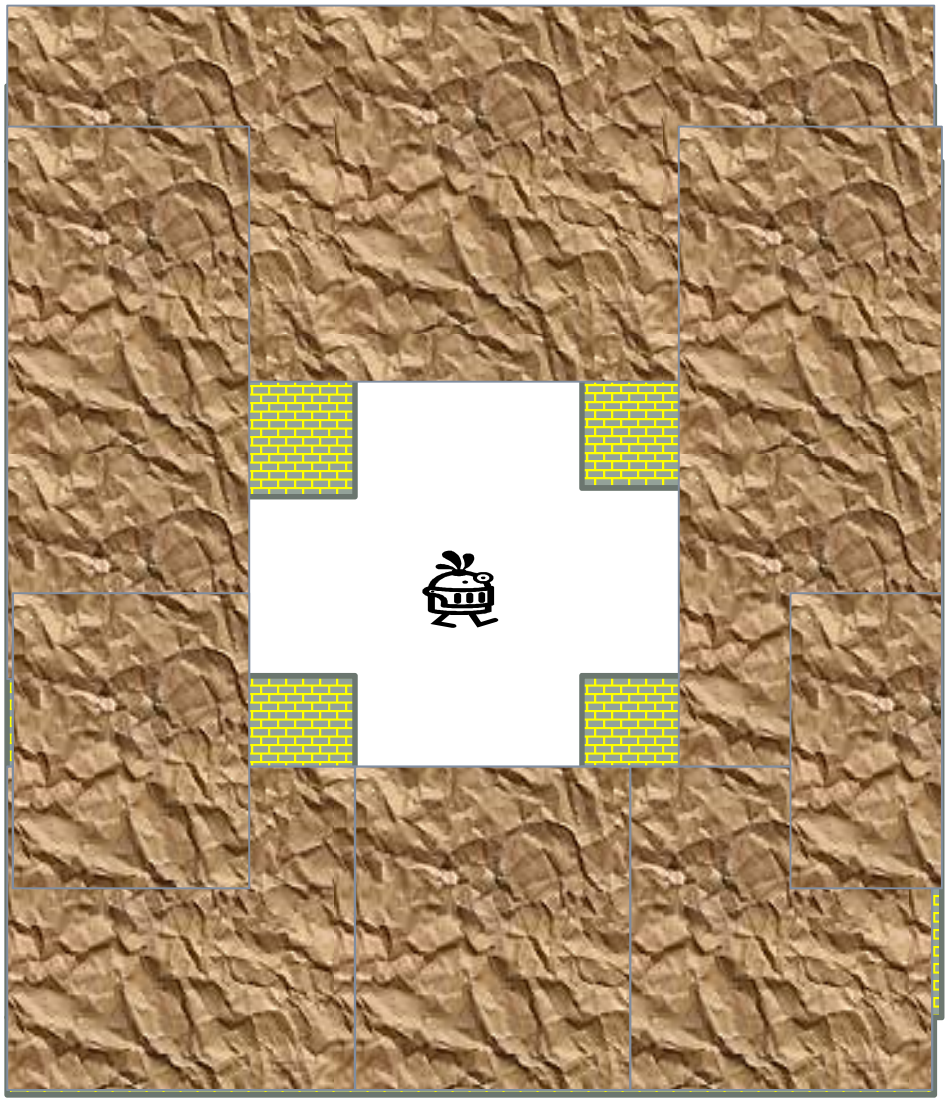
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Désirs/ Buts
Fonction de délibération
DevenirRiche ->TrouverTresor
Intentions
Actions

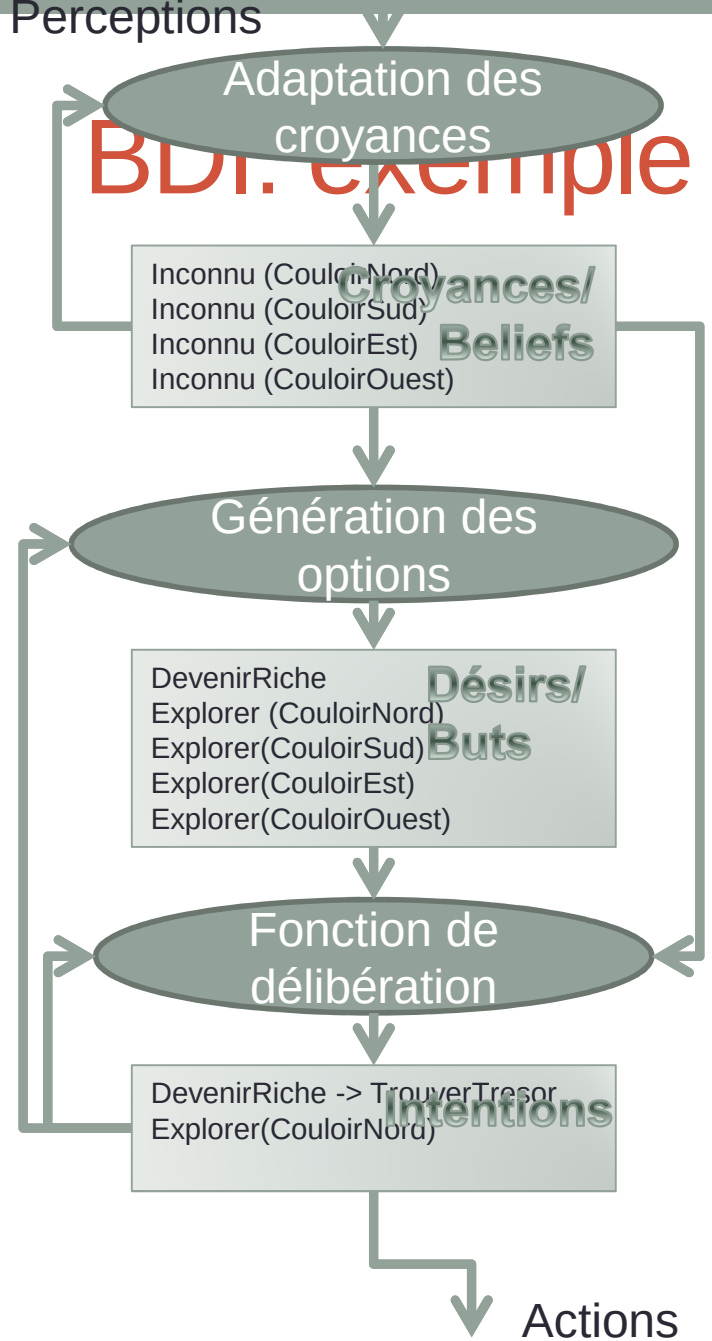


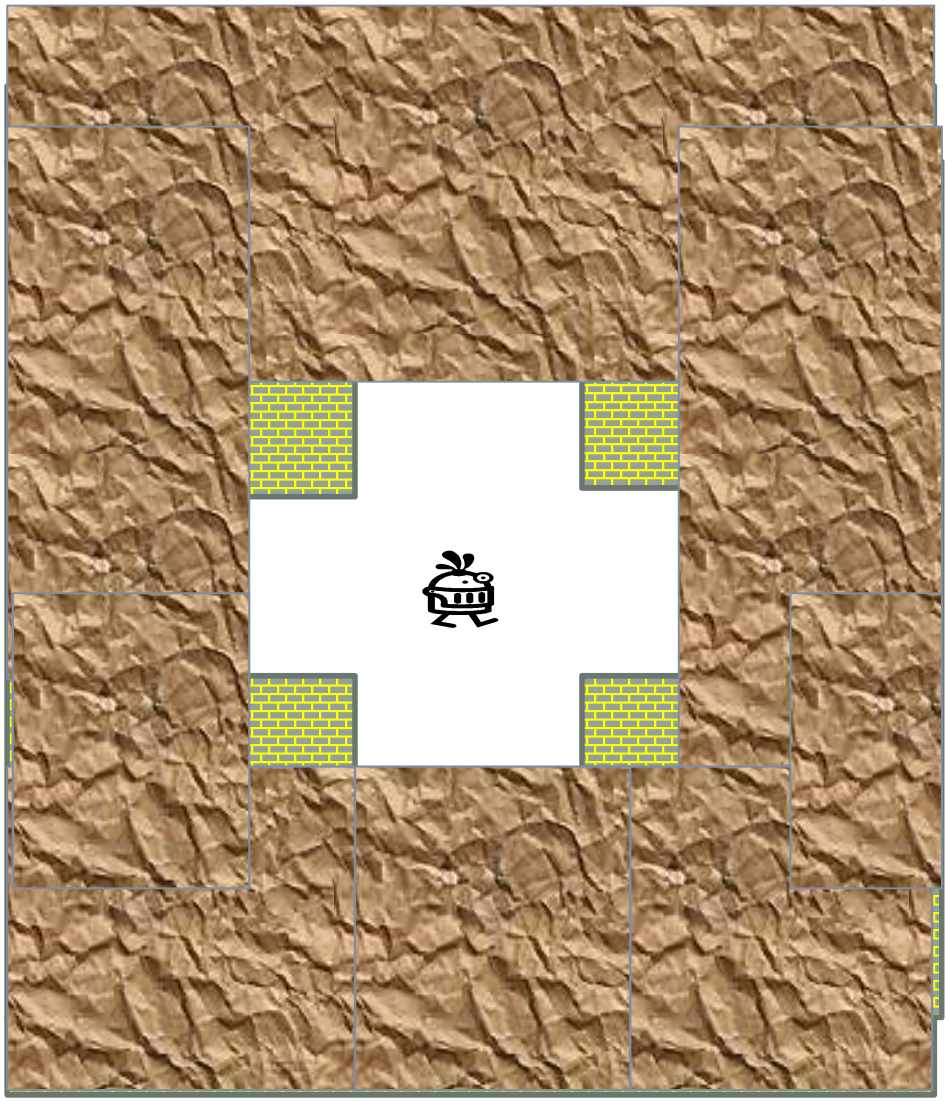
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord)
Explorer(CouloirSud)
Explorer(CouloirEst)
Explorer(CouloirOuest)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Intentions
Actions

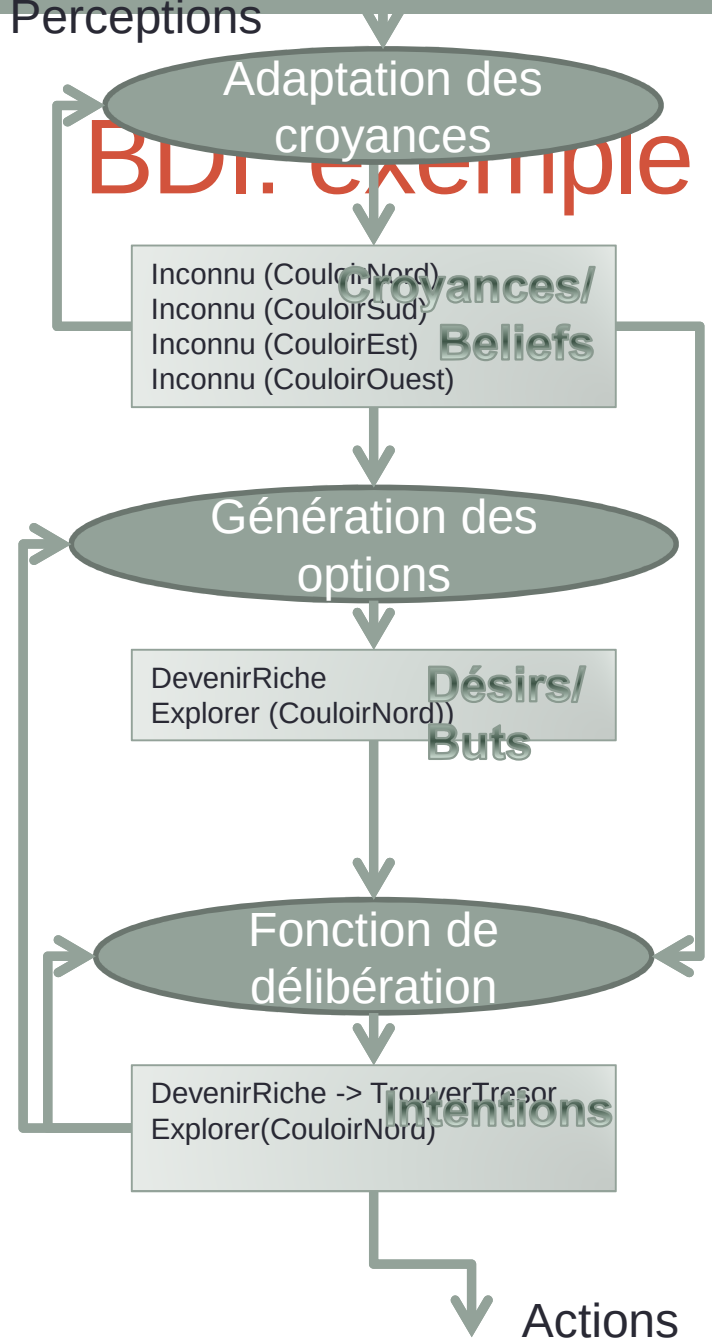


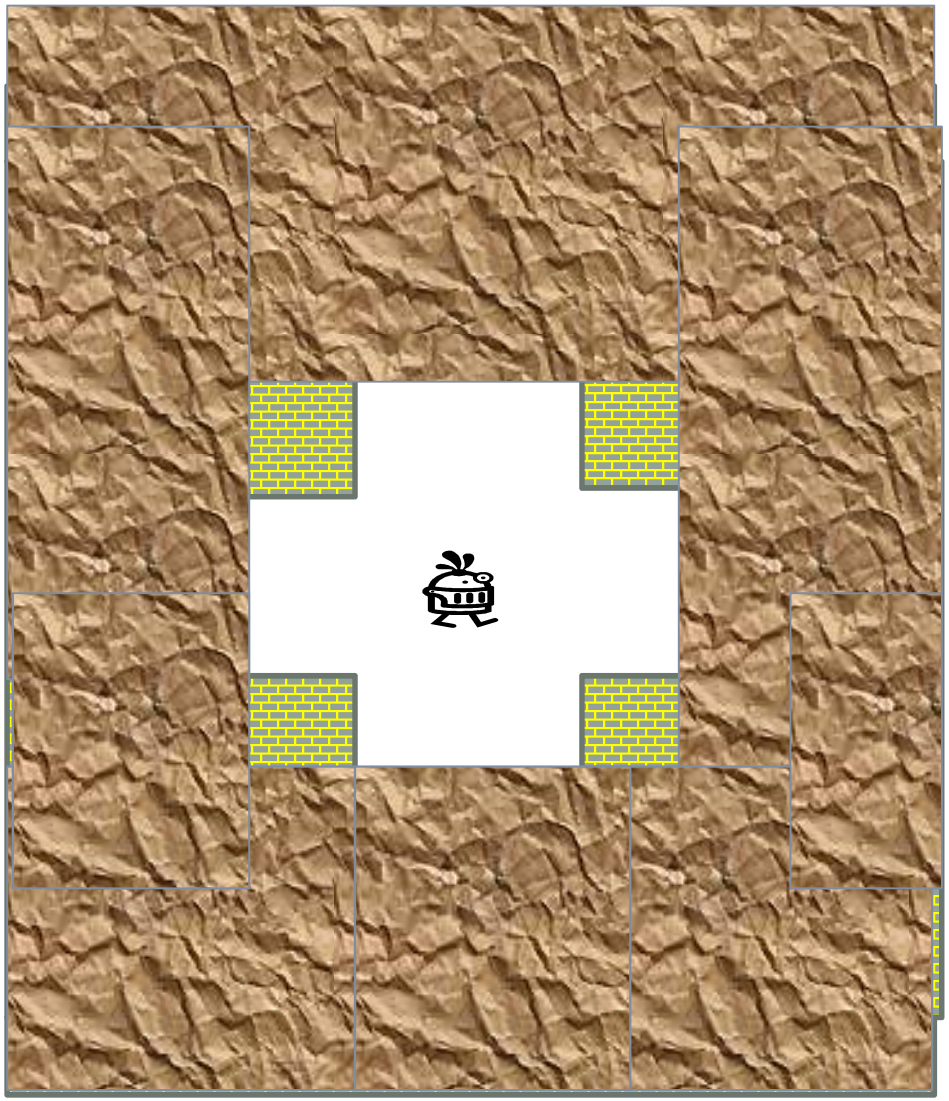
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord)
Explorer(CouloirSud)
Explorer(CouloirEst)
Explorer(CouloirOuest)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirNord)
Intentions
Actions

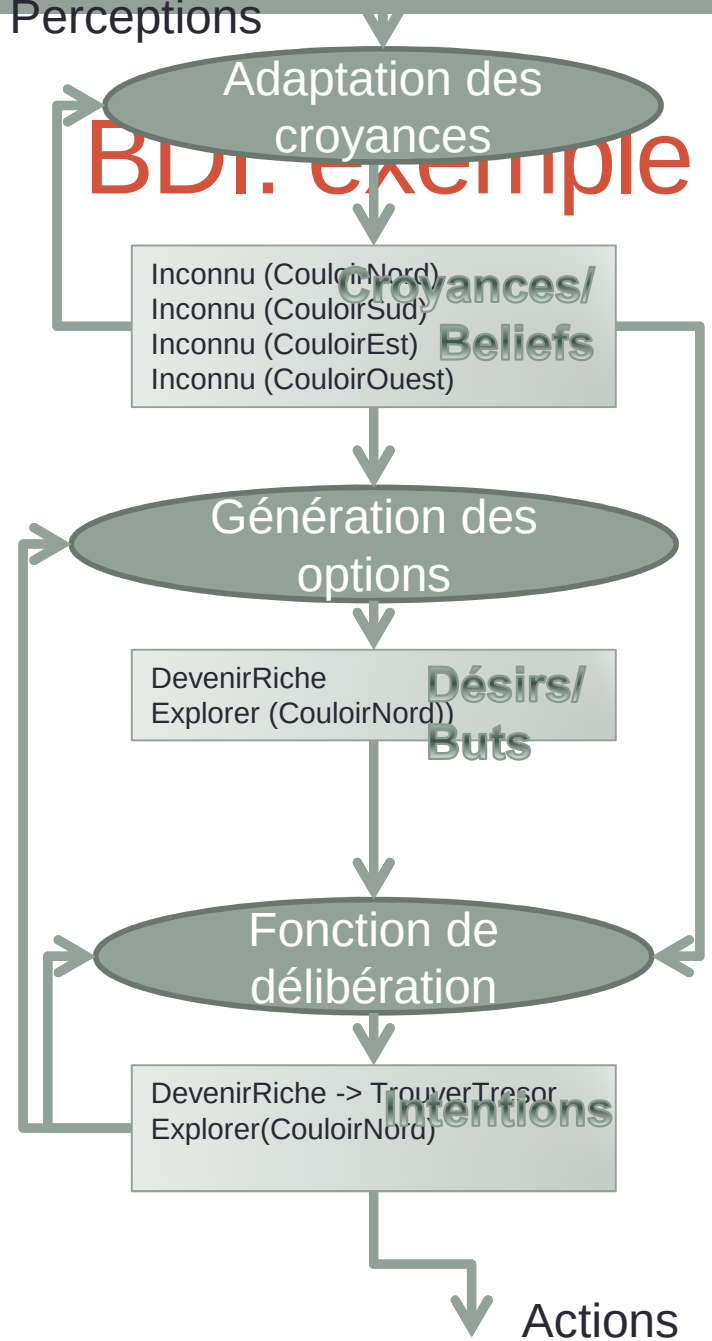


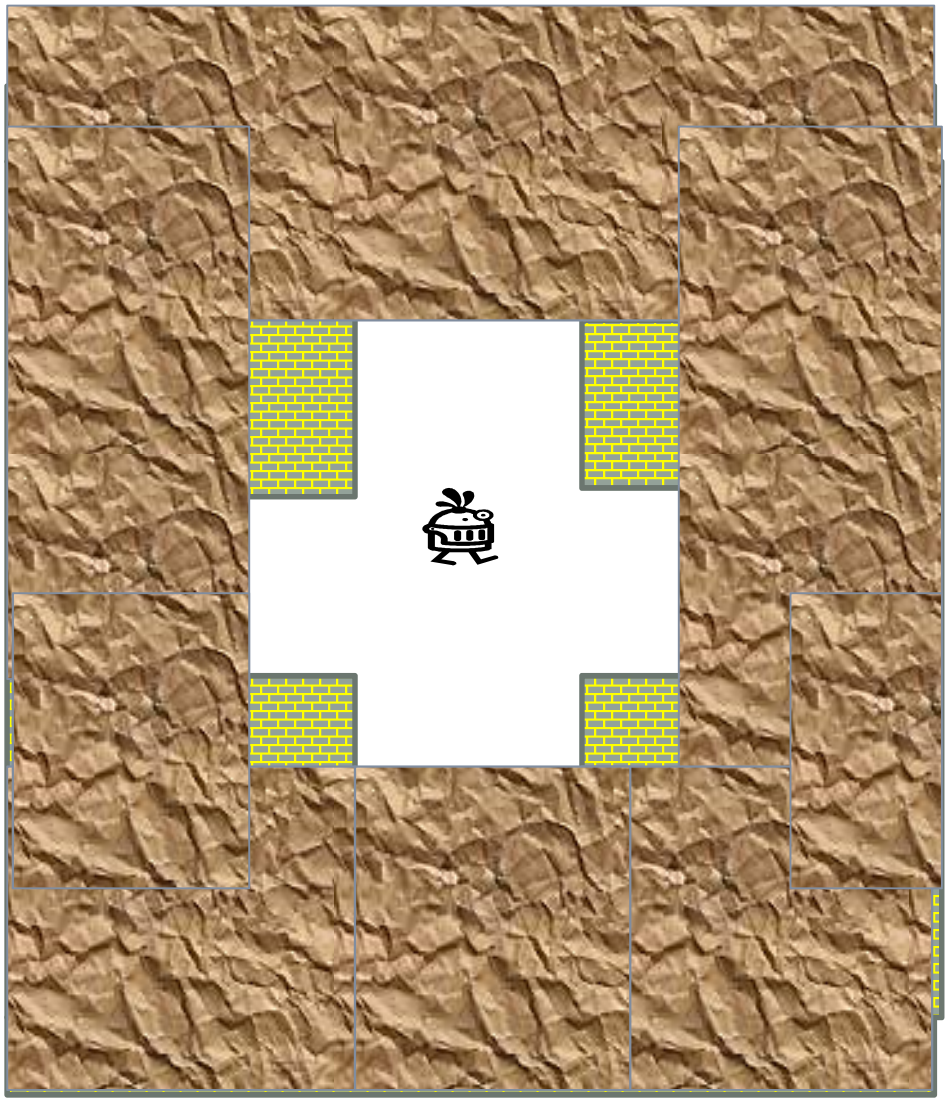
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord))
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirNord)
Intentions
Actions

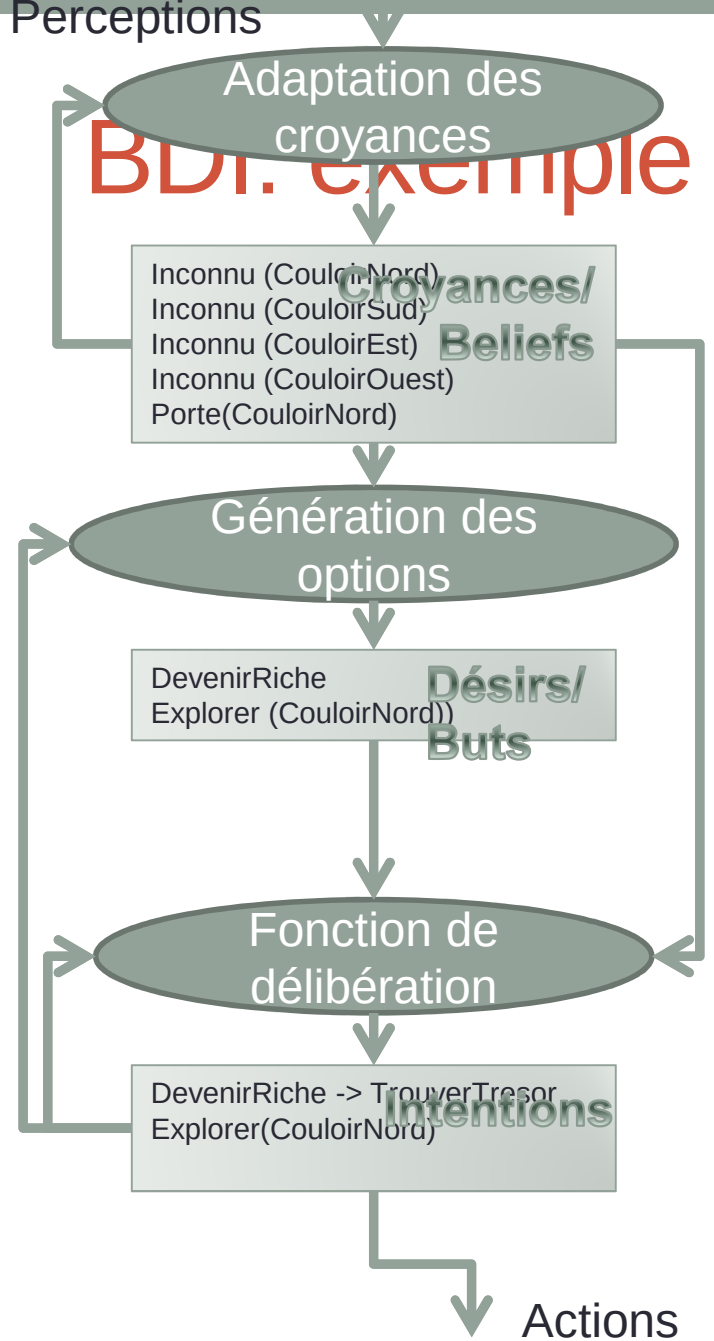


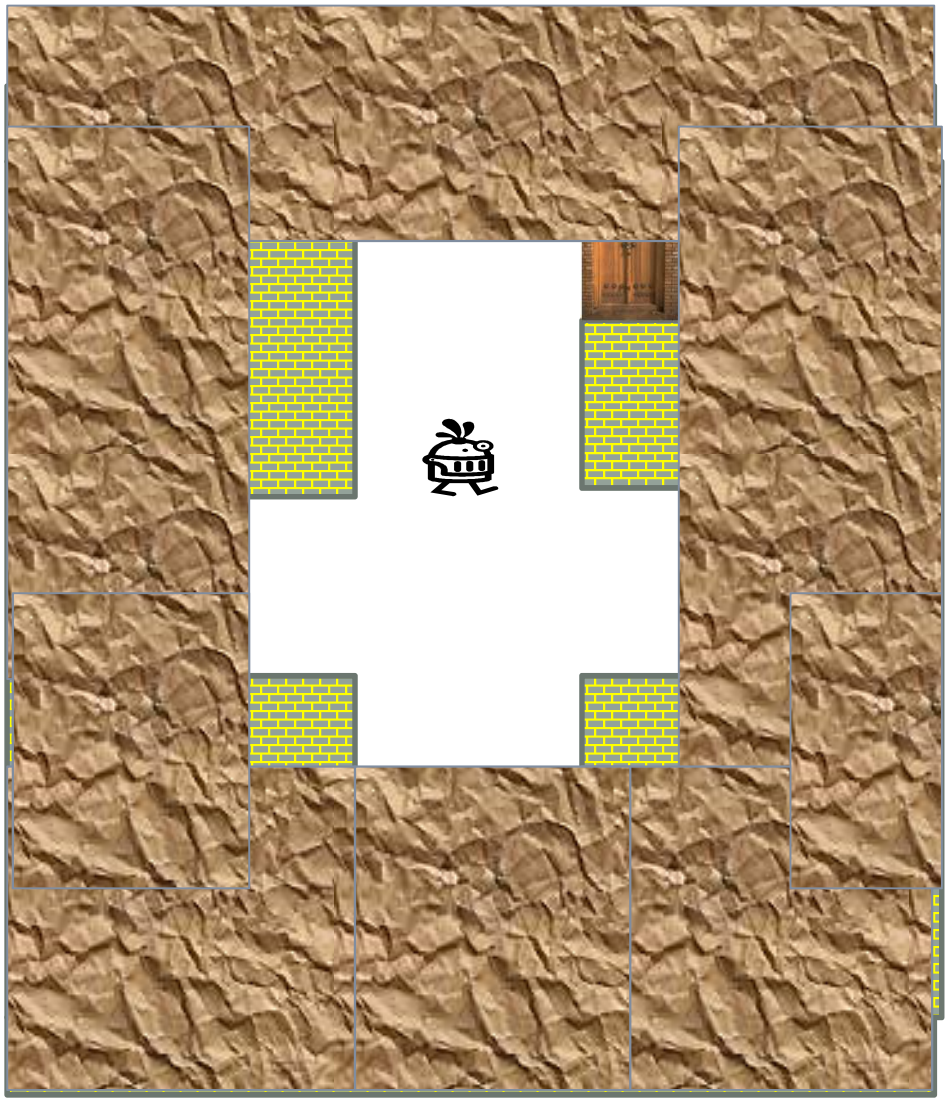
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord))
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirNord)
Intentions
Actions

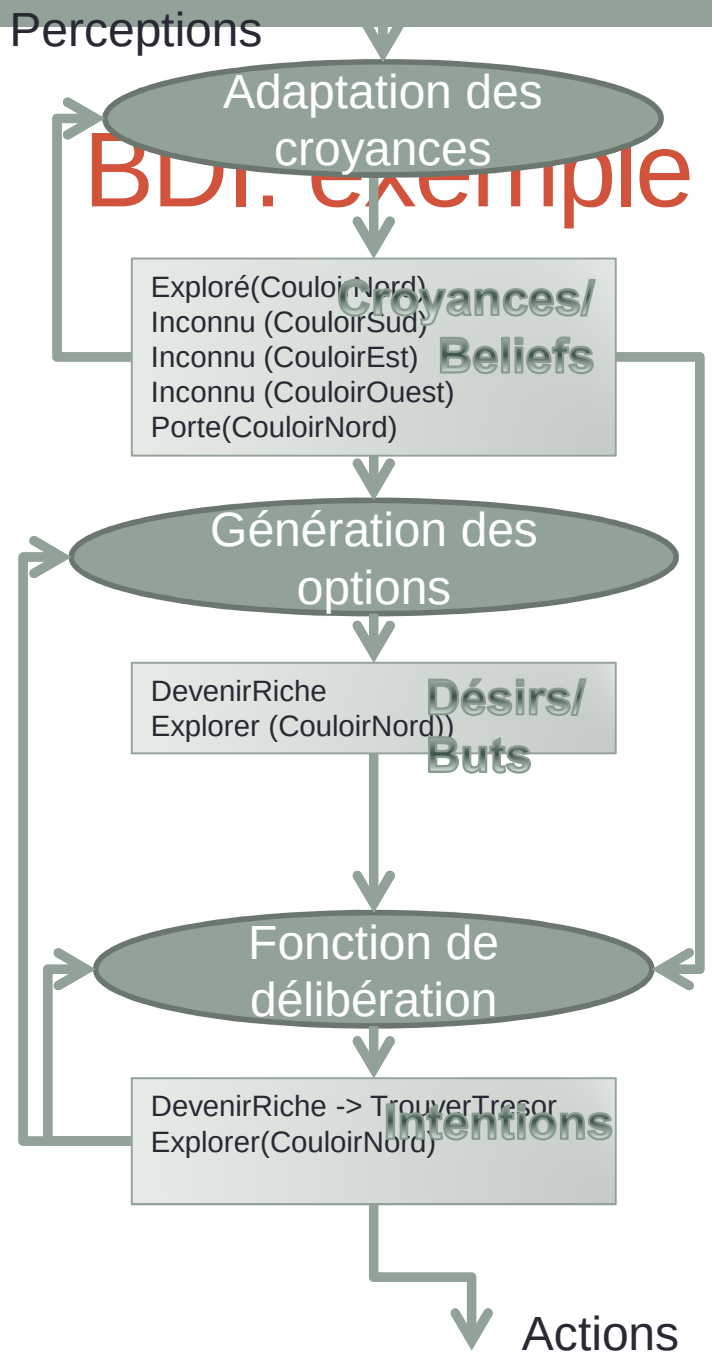


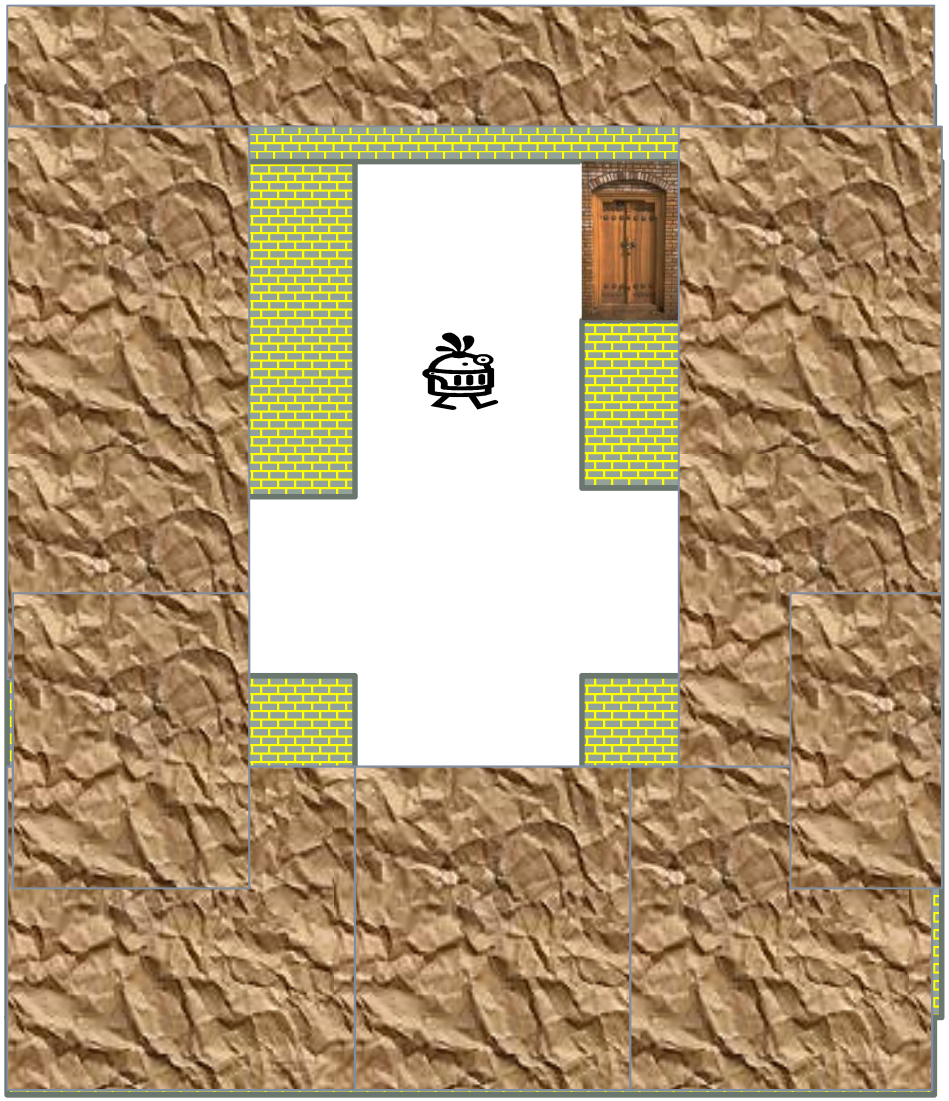
BDI: exemple


Perceptions
Adaptation des croyances
Inconnu (CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord))
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirNord)
Intentions
Actions

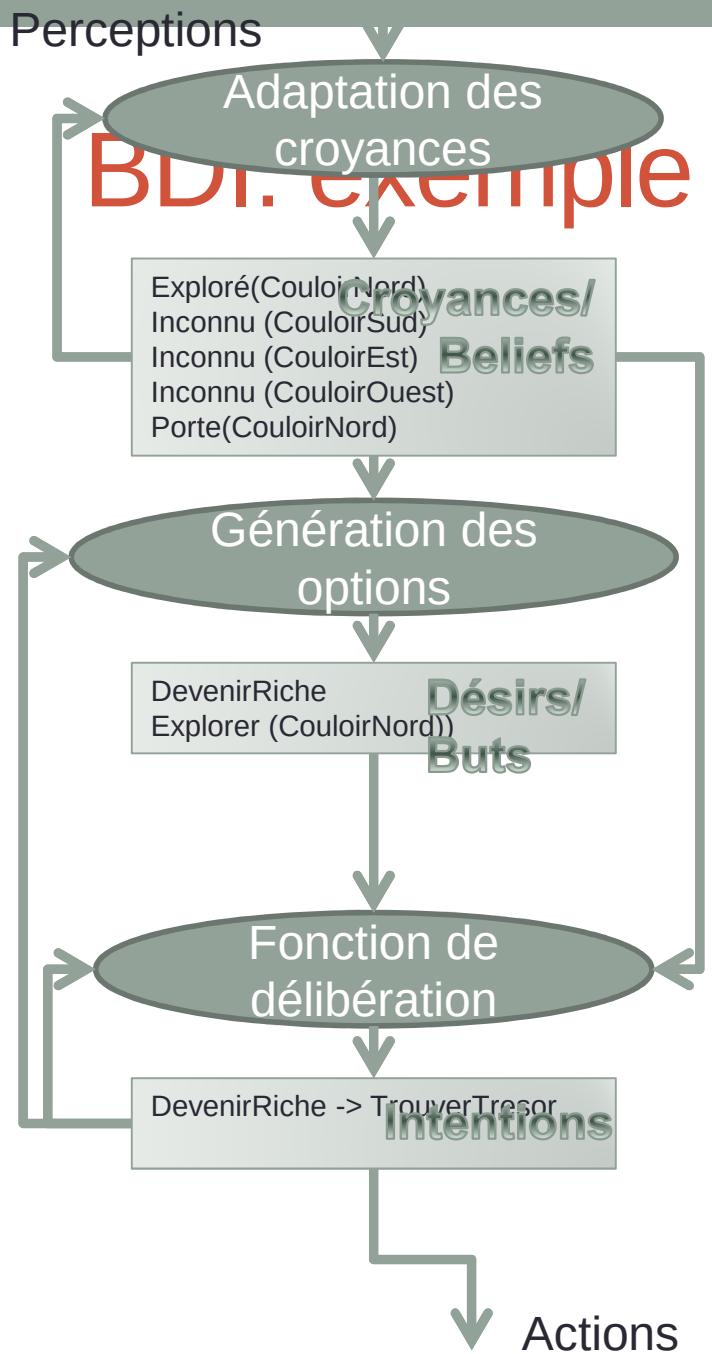


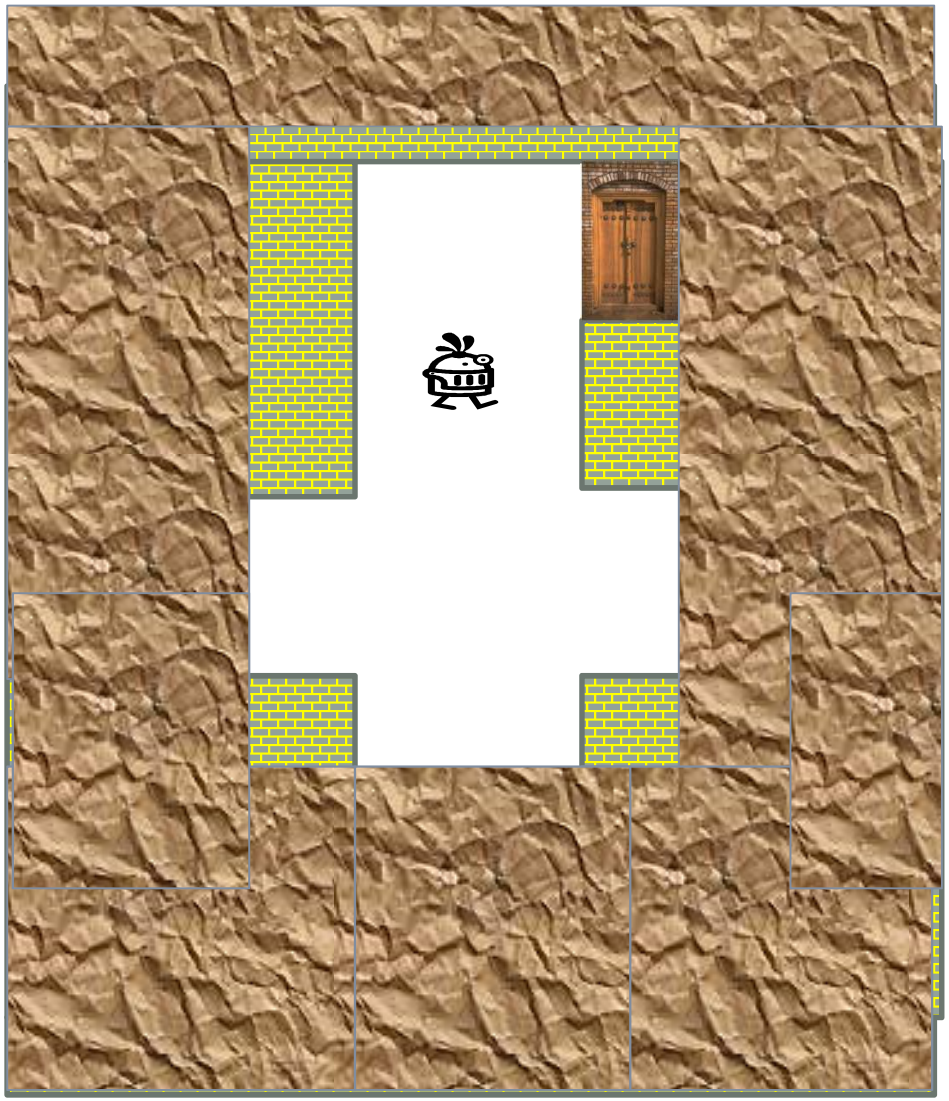
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord))
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirNord)
Intentions
Actions

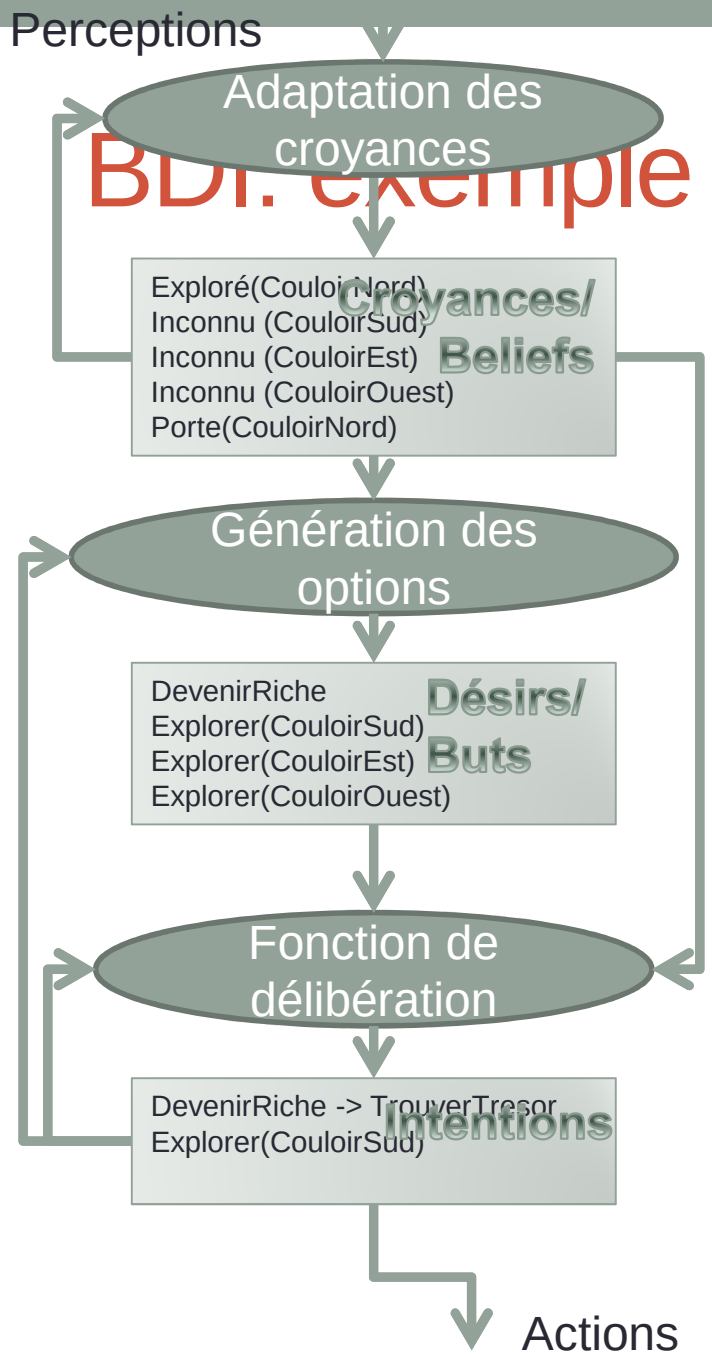


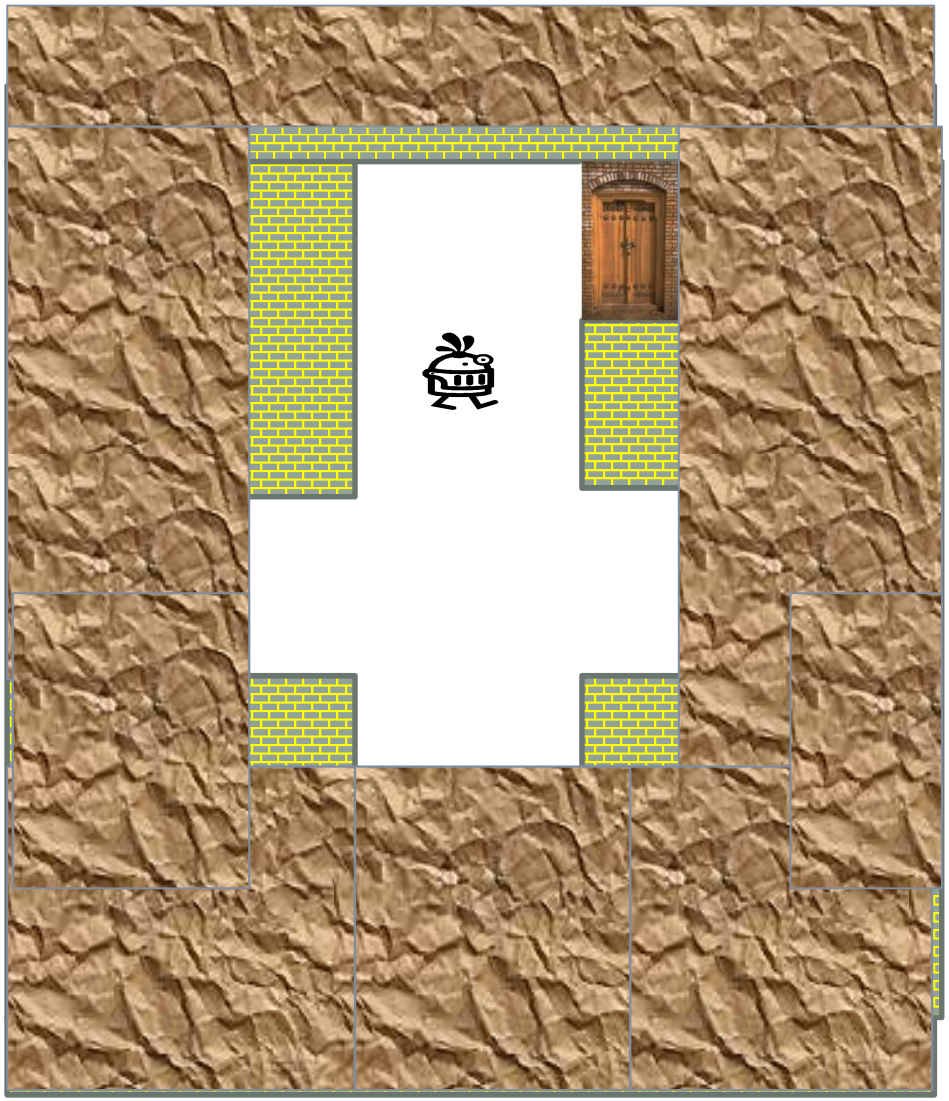
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirNord))
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Intentions
Actions

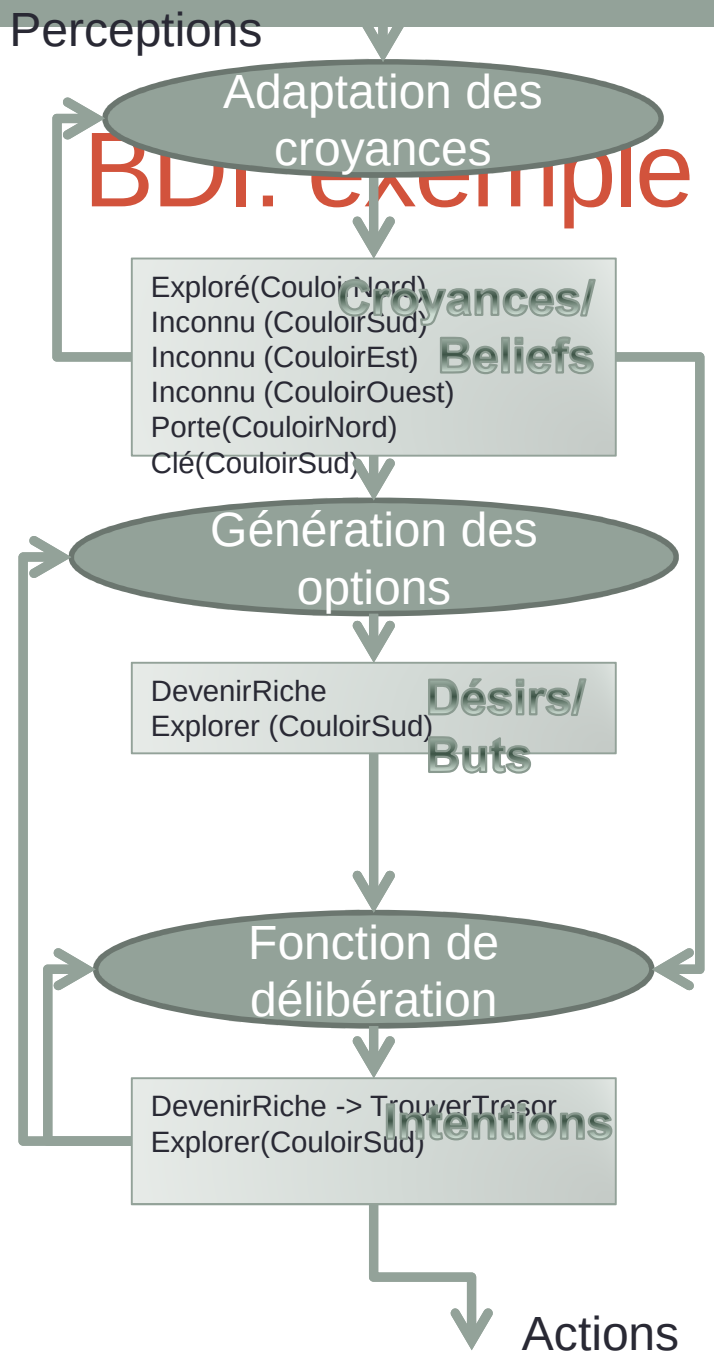


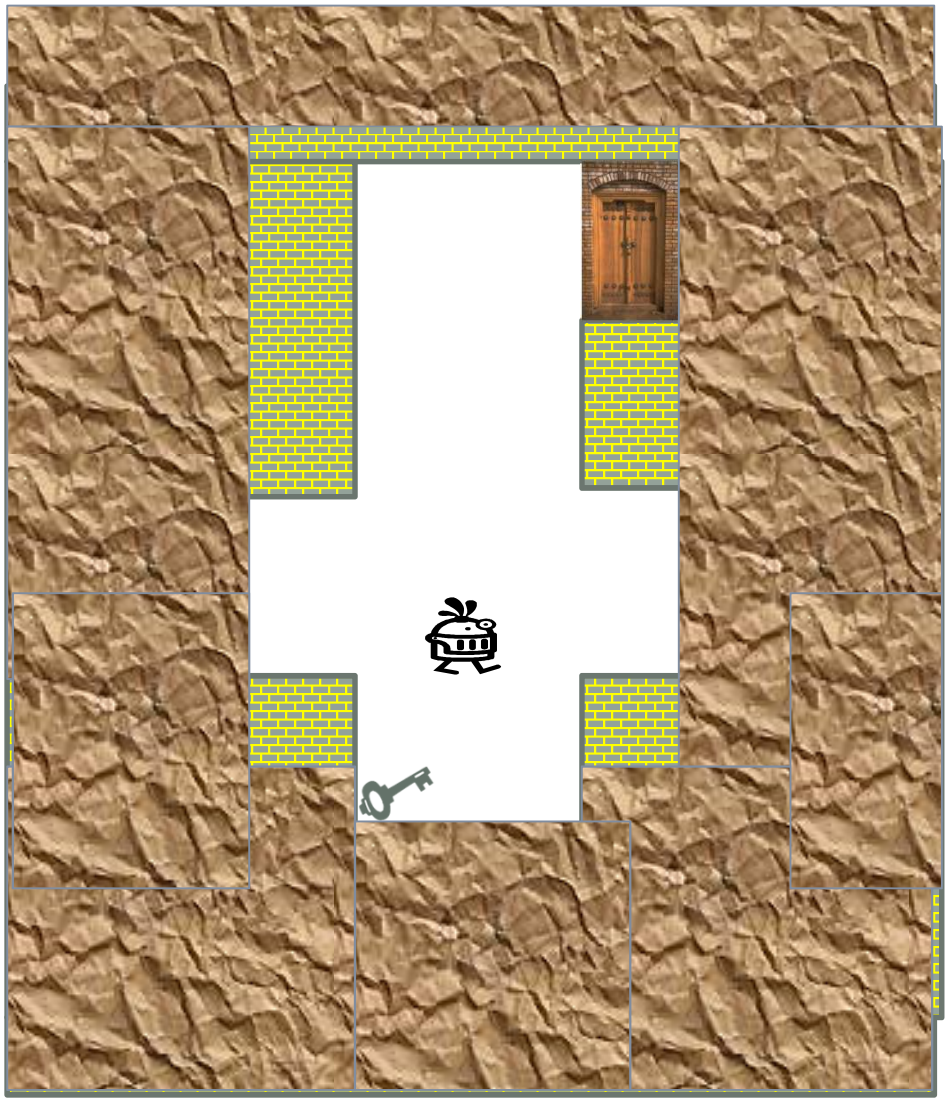
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer(CouloirSud)
Explorer(CouloirEst)
Explorer(CouloirOuest)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Intentions
Actions

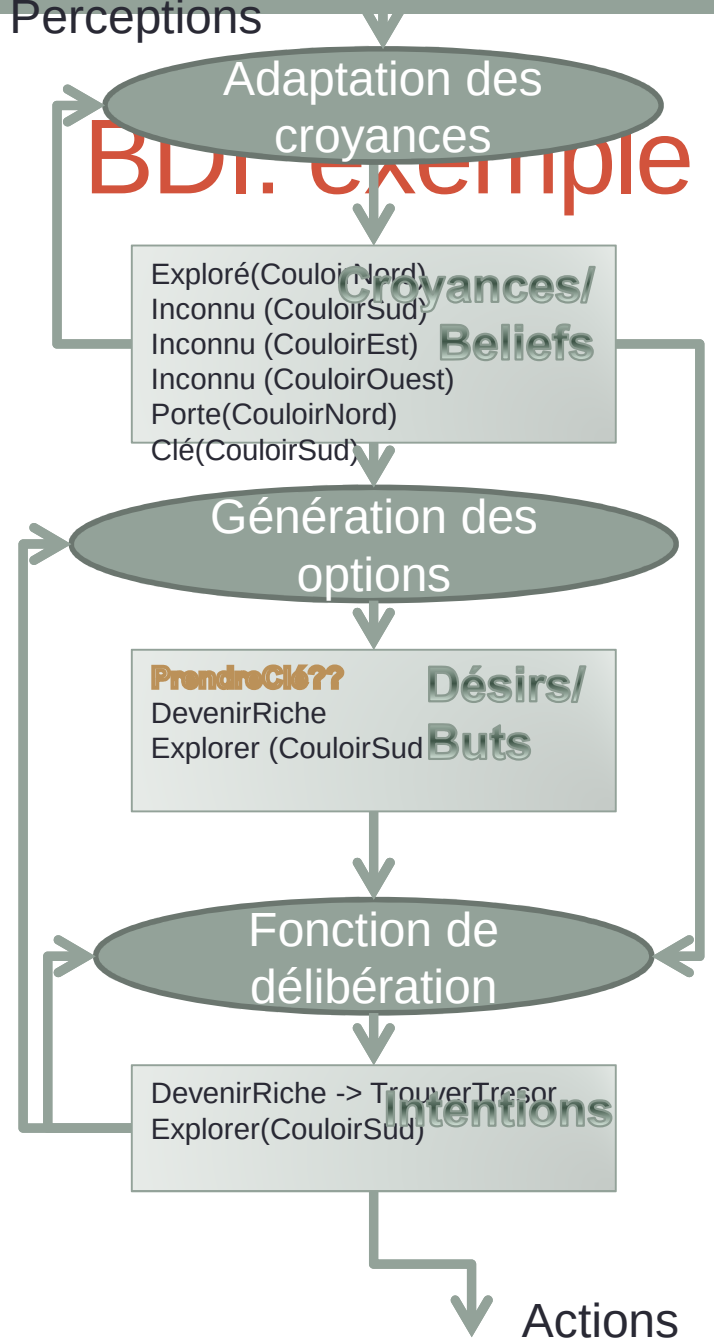


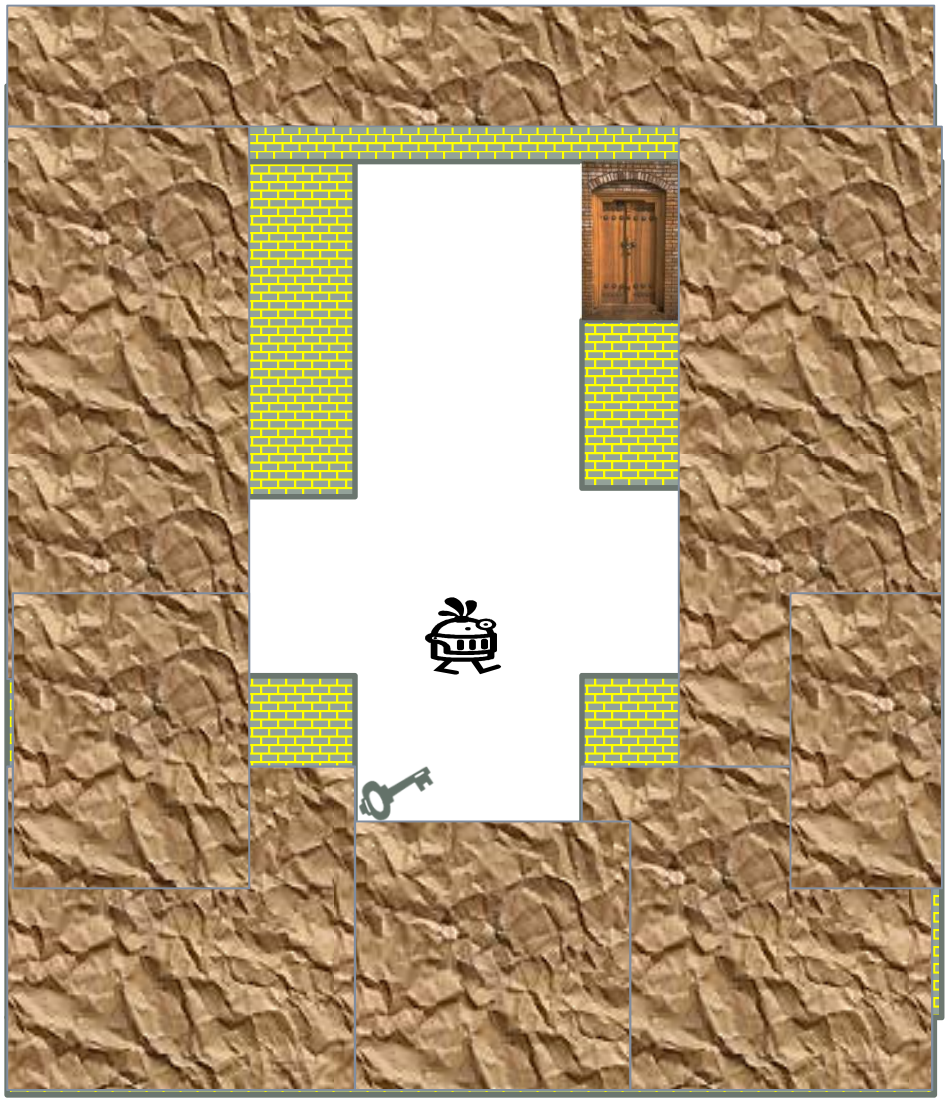
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirSud)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Intentions
Actions

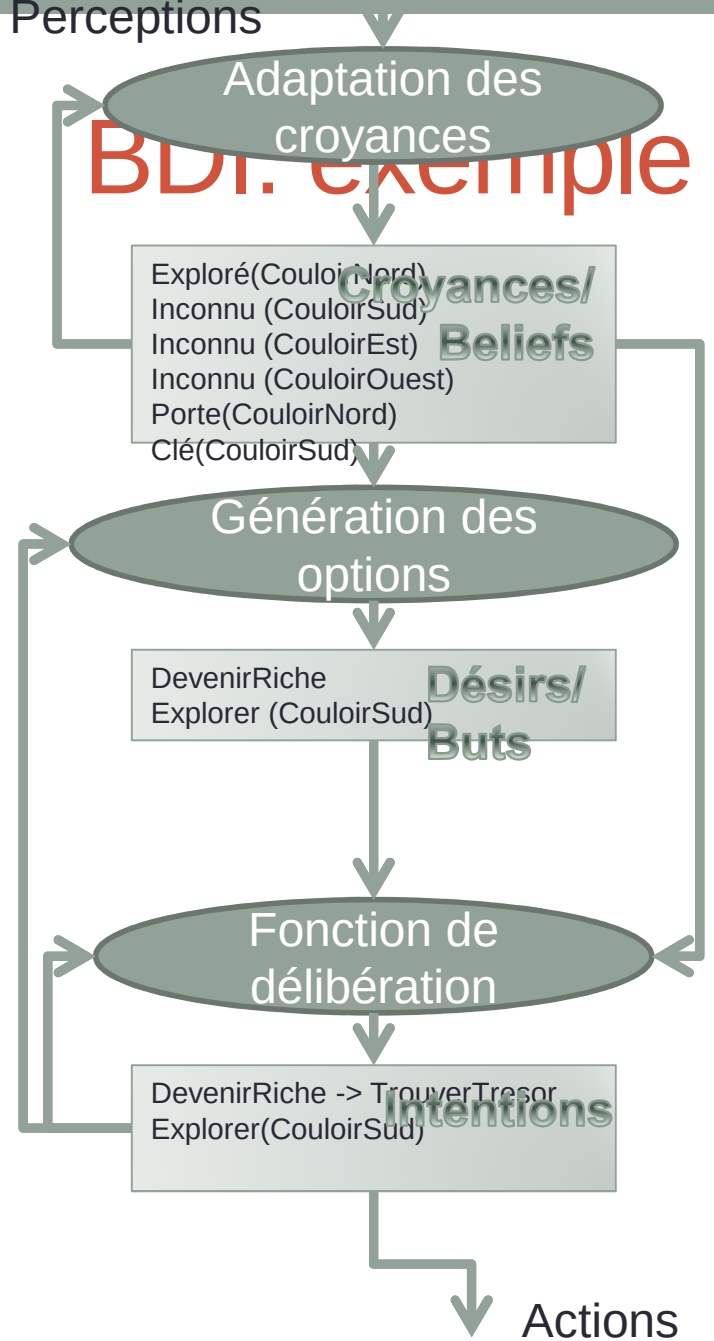


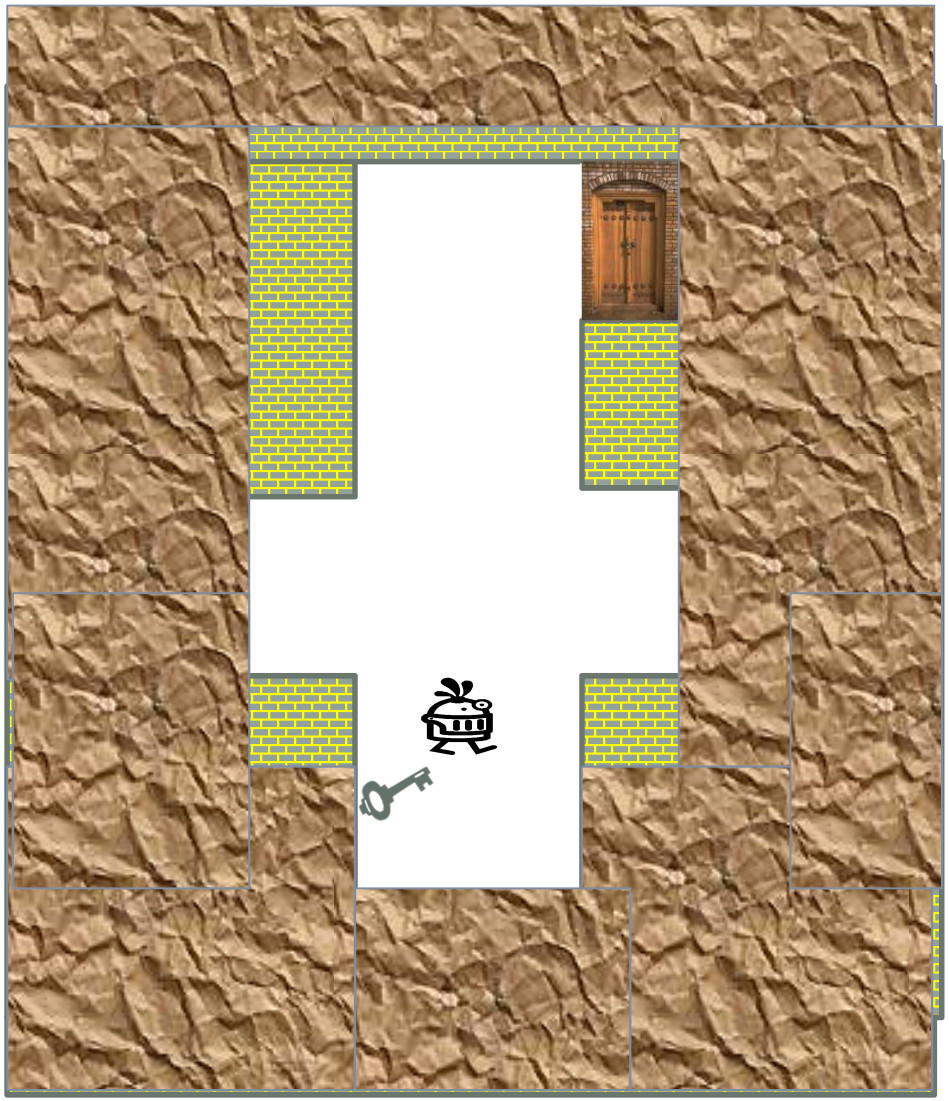
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Croyances/ Beliefs
Génération des options
PrendreClé??
DevenirRiche
Explorer (CouloirSud
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Intentions
Actions

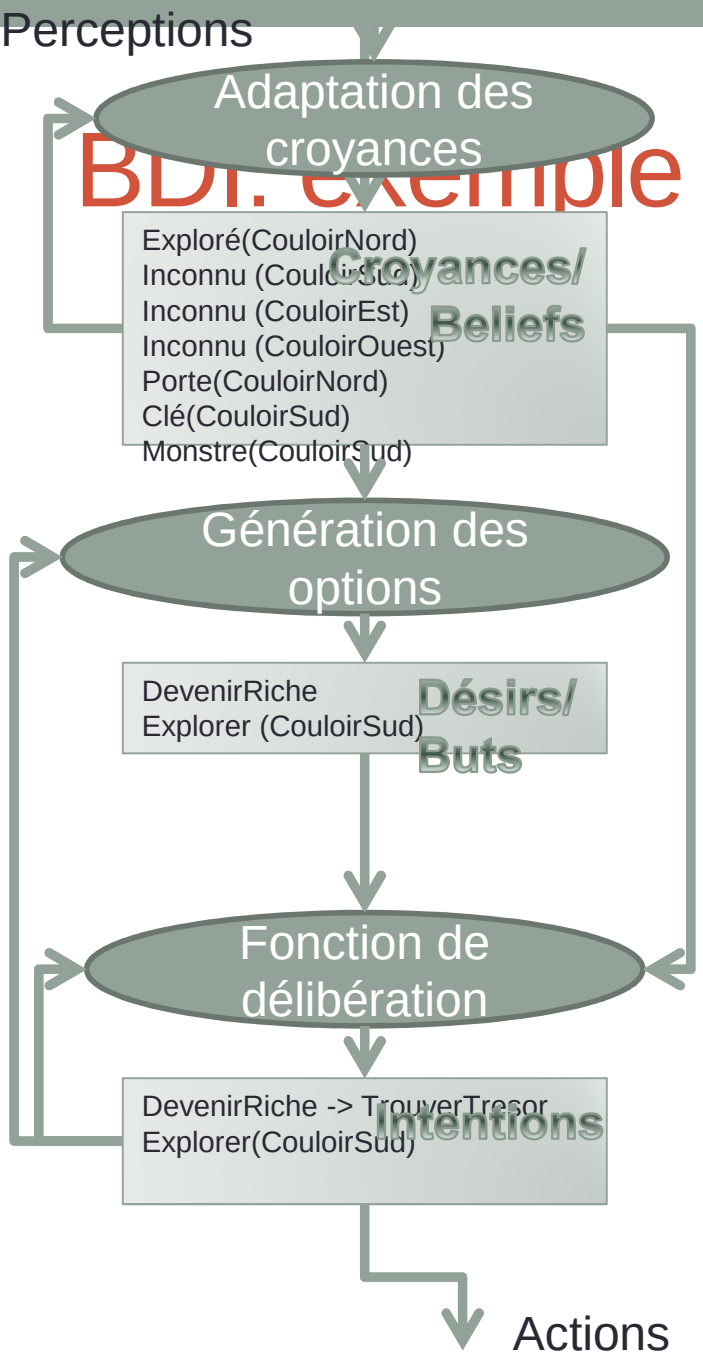


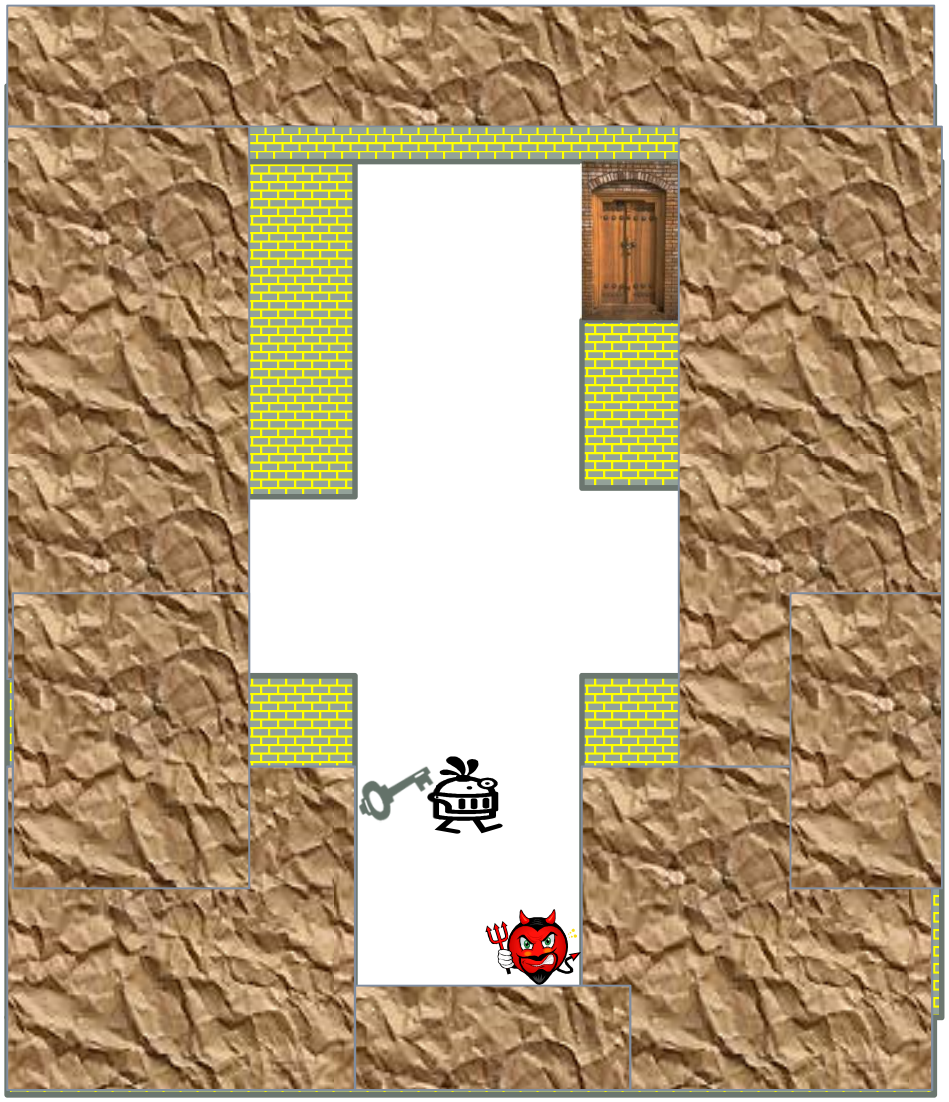
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirSud)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Intentions
Actions

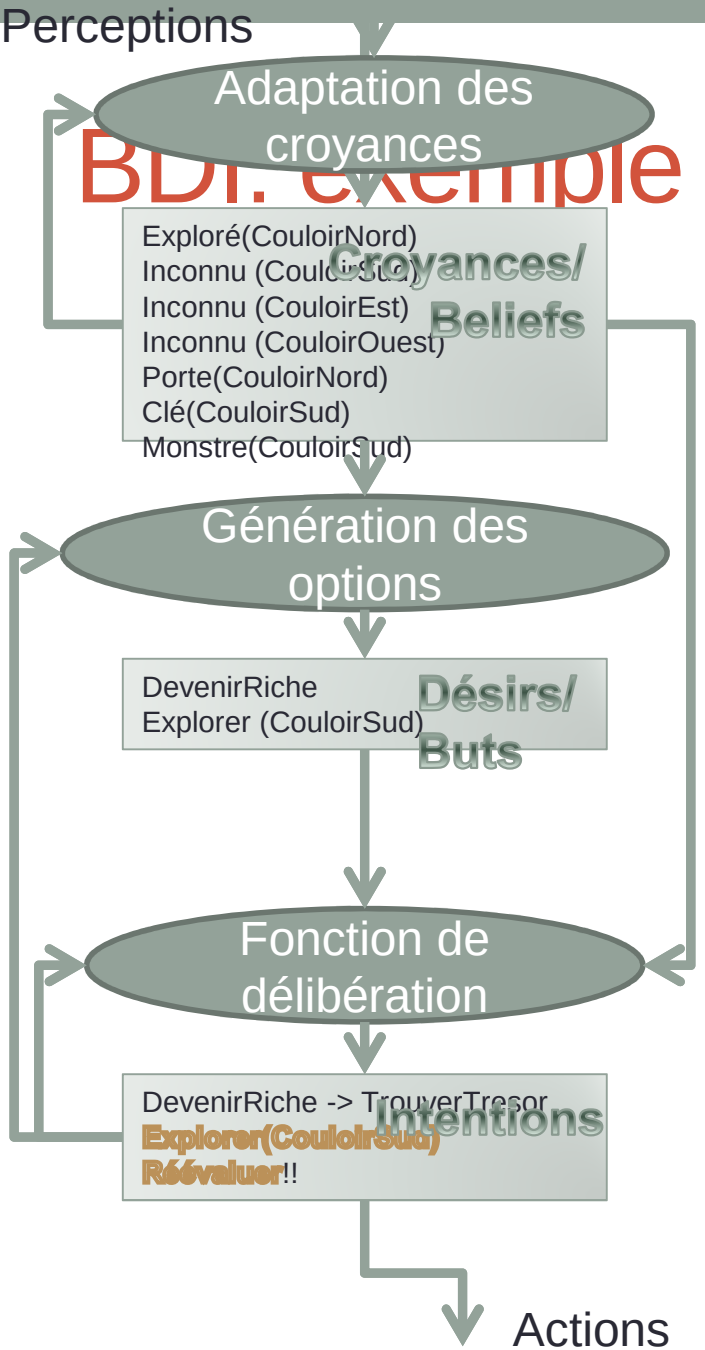


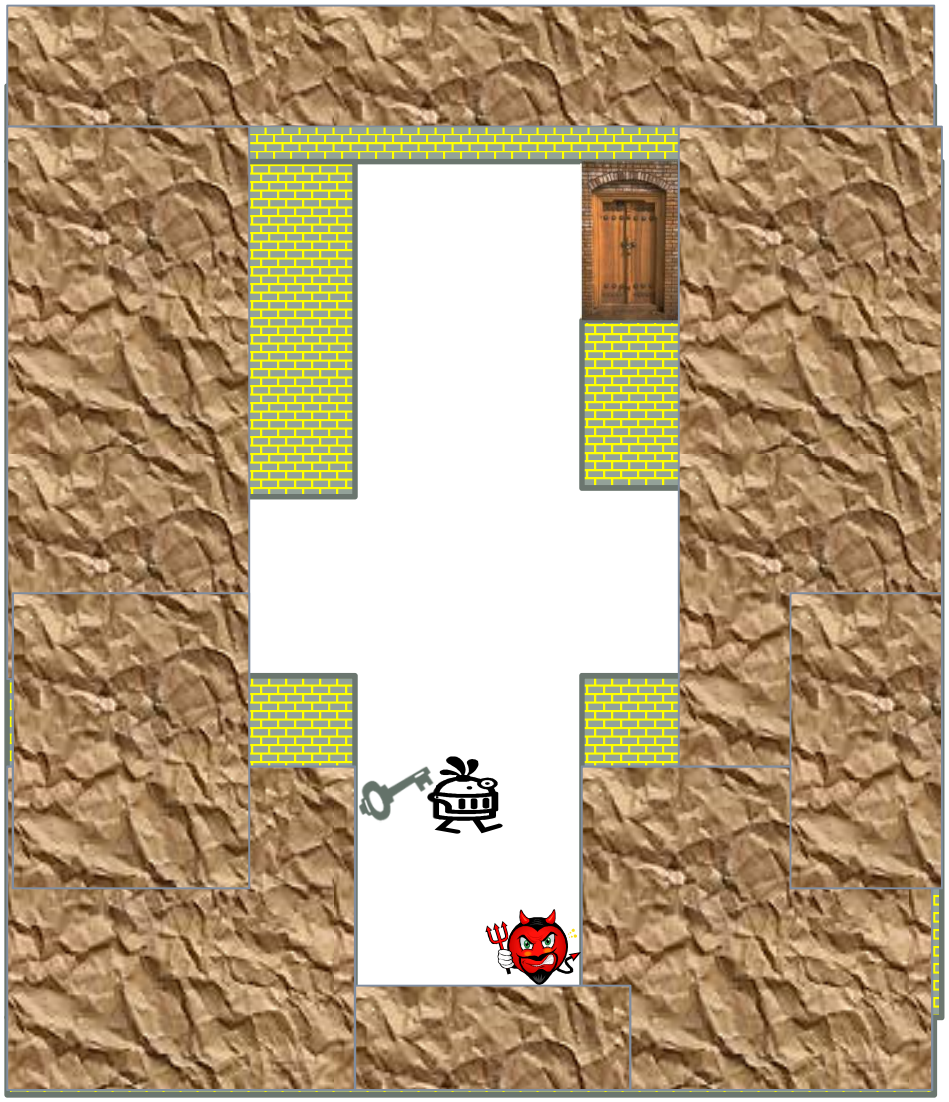
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Monstre(CouloirSud)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirSud)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Intentions
Actions

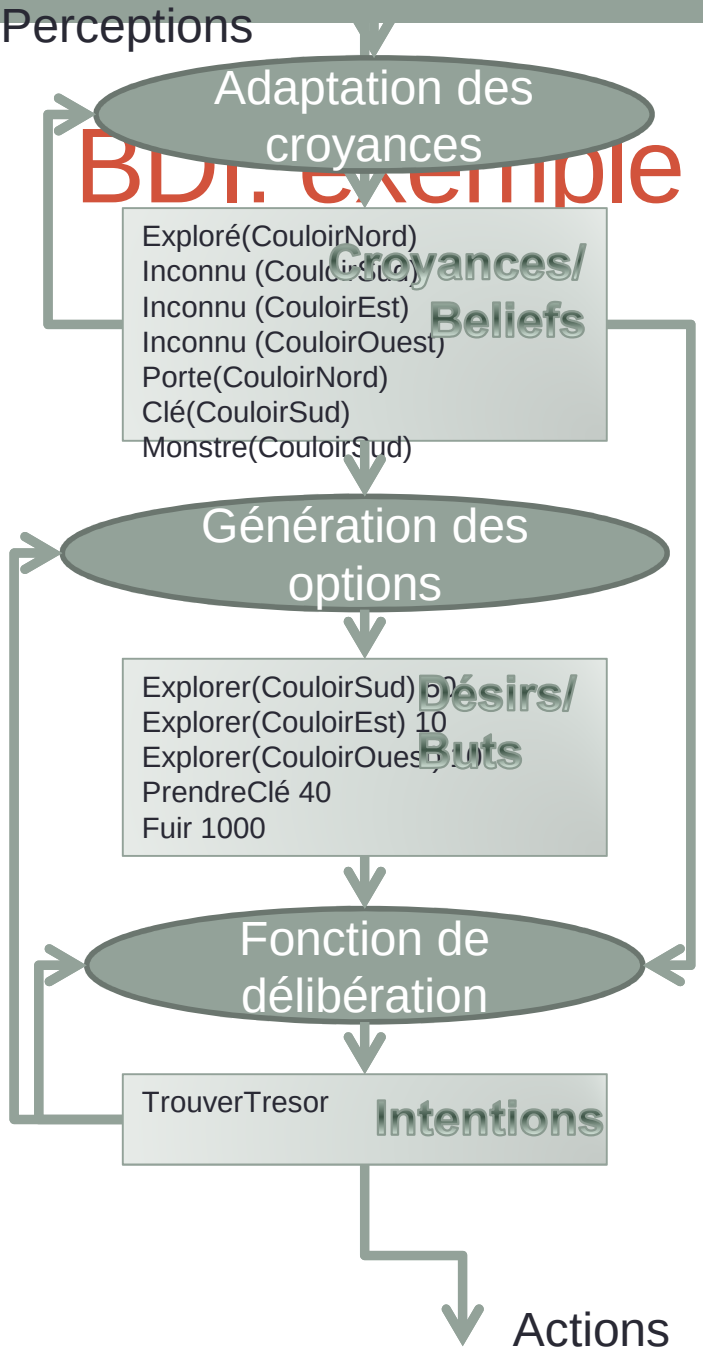


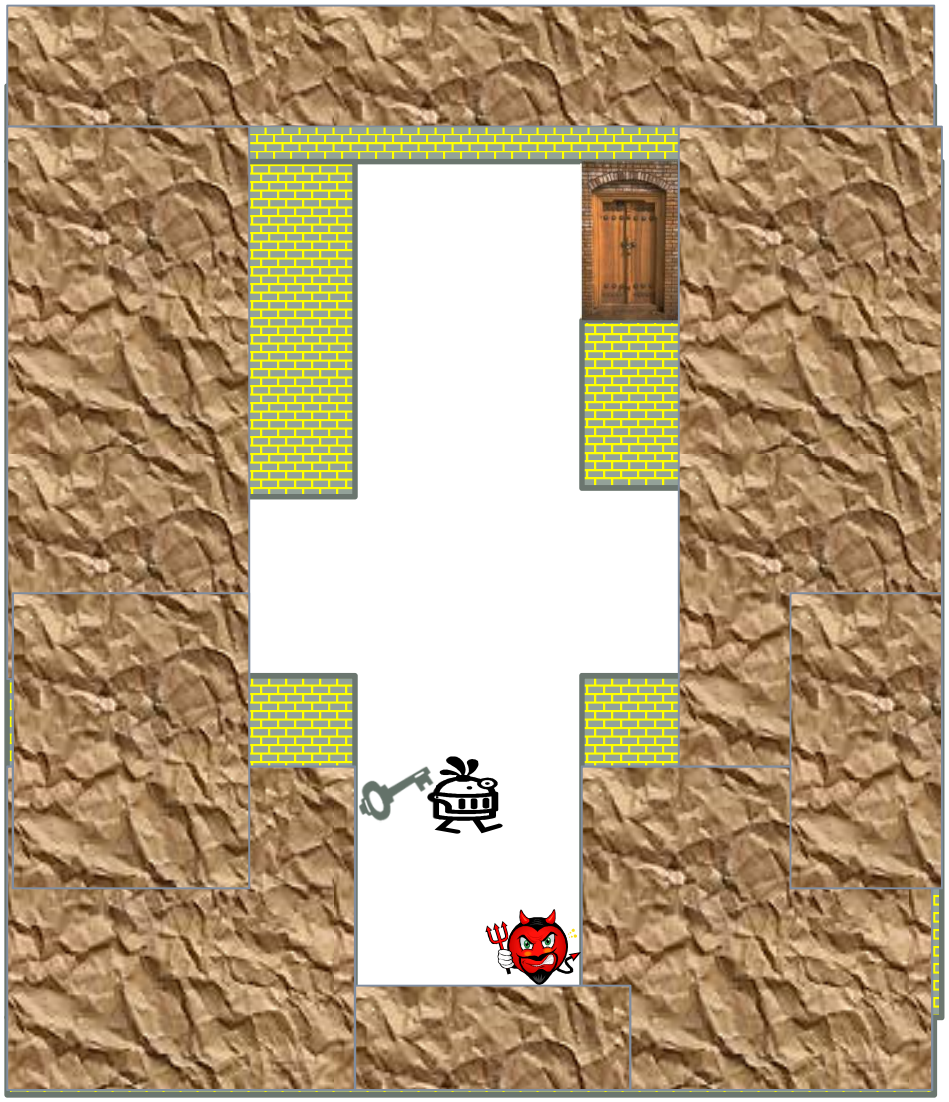
BDI: exemple


Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Monstre(CouloirSud)
Croyances/ Beliefs
Génération des options
DevenirRiche
Explorer (CouloirSud)
Désirs/ Buts
Fonction de délibération
DevenirRiche -> TrouverTresor
Explorer(CouloirSud)
Réévaluer!!
Intentions
Actions



BDI: exemple

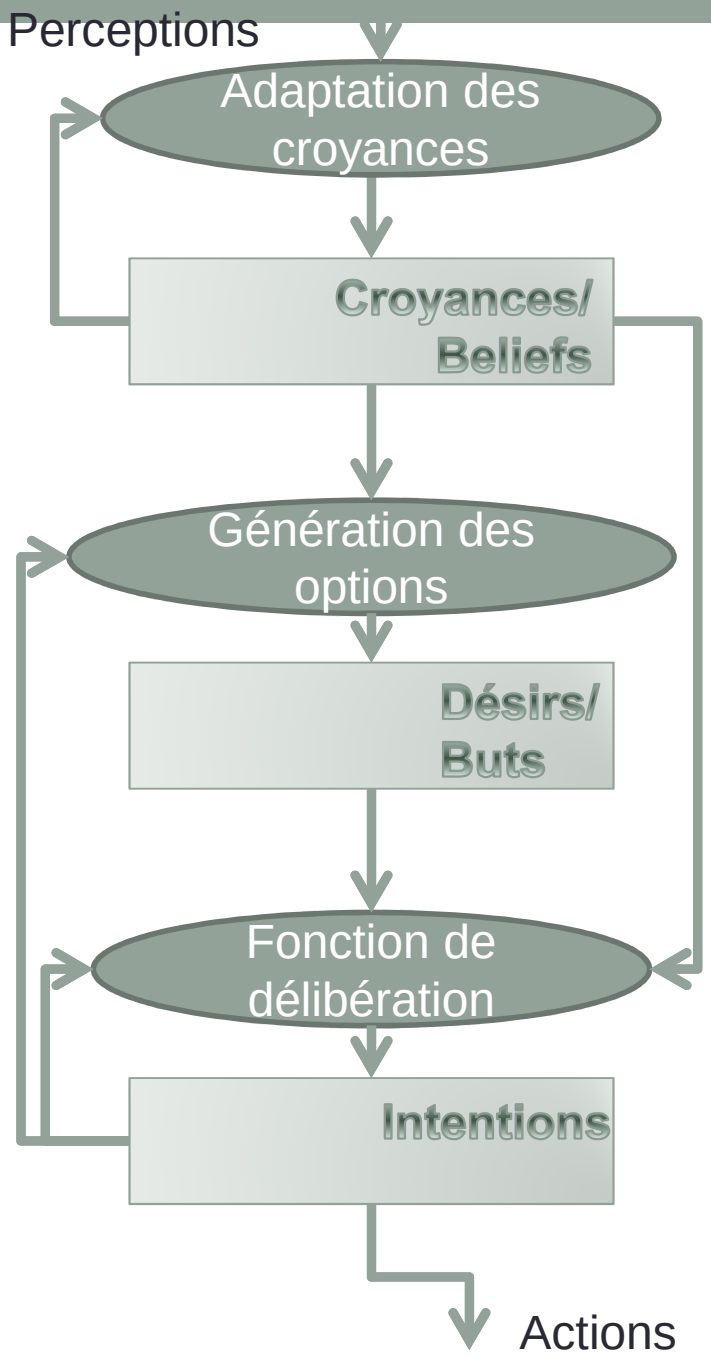

Perceptions
Adaptation des croyances
Exploré(CouloirNord)
Inconnu (CouloirSud)
Inconnu (CouloirEst)
Inconnu (CouloirOuest)
Porte(CouloirNord)
Clé(CouloirSud)
Monstre(CouloirSud)
Croyances/ Beliefs
Génération des options
Explorer(CouloirSud) 10
Explorer(CouloirEst) 10
Explorer(CouloirOuest) 10
PrendreClé 40
Fuir 1000
Désirs/ Buts
Fonction de délibération
TrouverTresor
Intentions
Actions



BDI: principes

- Beliefs: ce que je sais du monde
- Desires: ce que je souhaite
  - Goals/Buts: sous-ensemble cohérent de Desires
- Intentions: ce que je fais (plans en cours)

- Fonctions importante:
  - Génération d'options pour générer de nouveaux desirs/buts à partir des Beliefs et des Intentions
  - Délibération pour choisir les meilleurs plans
  - Reconsidération pour savoir quand se remettre en cause (doit être moins couteux que la délibération)



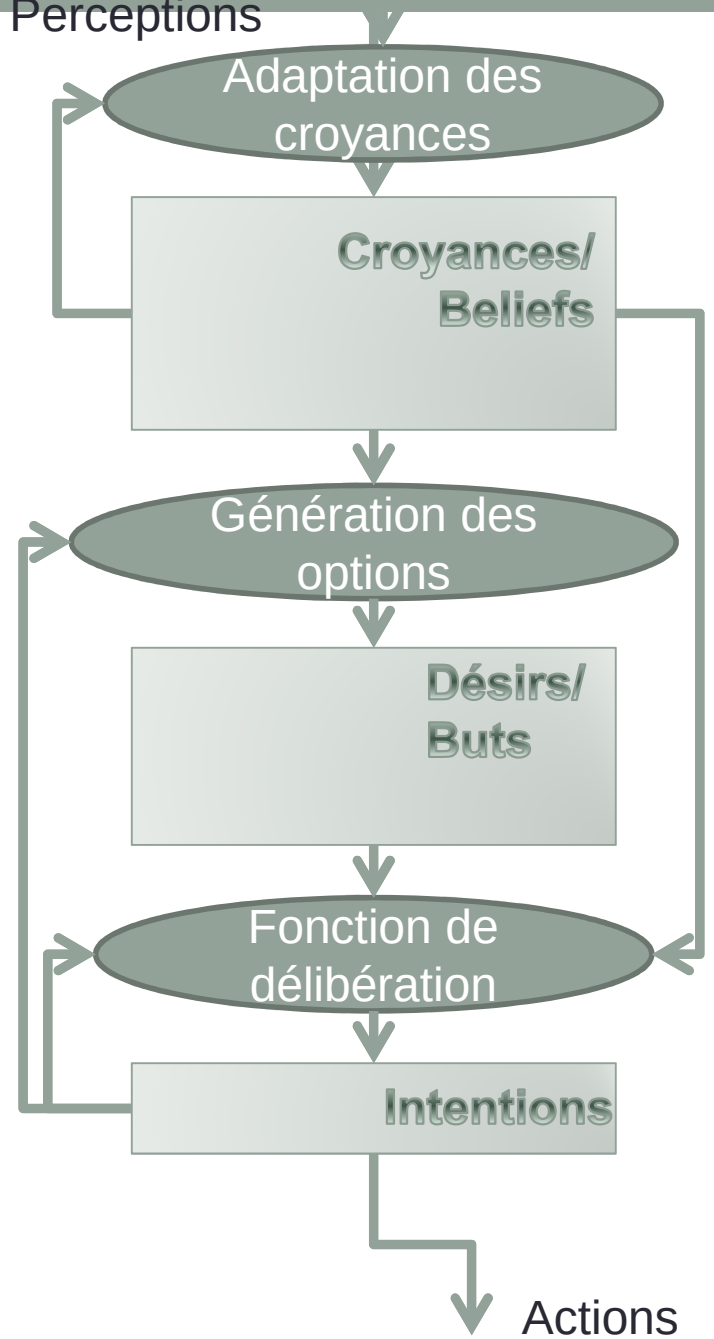
BDI: réévaluation des intentions

- Deux cas types:
  - Agent prudent: réévalue après chaque action
  - Agent déterminé: ne réévalue jamais (une intention est abandonnée lorsqu'elle est atteinte ou impossible)
- Dans le cas d'environnement stable, les agents déterminés se comportent mieux.
- Dans le cas d'environnement variables, les agents prudents arrivent à repérer plus vite quand leurs intentions ne sont plus les plus efficaces

BDI: exemple2

- Syntaxe JAM
- [Wooldridge 09]

```
GOALS:
  ACHIEVE blocks_stacked;

FACTS:
  FACT ON "Block5" "Block4";      FACT ON "Block4" "Block3";
  FACT ON "Block1" "Block2";      FACT ON "Block2" "Table";
  FACT ON "Block3" "Table";       FACT CLEAR "Block1";
  FACT CLEAR "Block5";            FACT CLEAR "Table";

Plan: {
  NAME: "Top-level plan"
  GOAL: ACHIEVE blocks_stacked;
  CONTEXT:
  BODY:     ACHIEVE ON "Block3" "Table";
            ACHIEVE ON "Block2" "Block3";
            ACHIEVE ON "Block1" "Block2";
}

Plan: {
  NAME: "Stack blocks that are already clear"
  GOAL: ACHIEVE ON $OBJ1 $OBJ2;
  CONTEXT:
  BODY:     ACHIEVE CLEAR $OBJ1;
            ACHIEVE CLEAR $OBJ2;
            PERFORM move $OBJ1 $OBJ2;
  UTILITY: 10;
  FAILURE: EXECUTE print "\n\nStack blocks failed!\n\n";
}

Plan: {
  NAME: "Clear a block"
  GOAL: ACHIEVE CLEAR $OBJ;
  CONTEXT: FACT ON $OBJ2 $OBJ;
  BODY:     ACHIEVE ON $OBJ2 "Table";
  EFFECTS: RETRACT ON $OBJ1 $OBJ;
  FAILURE: EXECUTE print "\n\nClearing block failed!\n\n";
}
```




```
GOALS:
  ACHIEVE blocks_stacked;

FACTS:
  FACT ON "Block5" "Block4";      FACT ON "Block4" "Block3";
  FACT ON "Block1" "Block2";      FACT ON "Block2" "Table";
  FACT ON "Block3" "Table";       FACT CLEAR "Block1";
  FACT CLEAR "Block5";            FACT CLEAR "Table";

Plan: {
  NAME: "Top-level plan"
  GOAL: ACHIEVE blocks_stacked;
  CONTEXT:
  BODY:      ACHIEVE ON "Block3" "Table";
             ACHIEVE ON "Block2" "Block3";
             ACHIEVE ON "Block1" "Block2";
}

Plan: {
  NAME: "Stack blocks that are already clear"
  GOAL: ACHIEVE ON $OBJ1 $OBJ2;
  CONTEXT:
  BODY:      ACHIEVE CLEAR $OBJ1;
             ACHIEVE CLEAR $OBJ2;
             PERFORM move $OBJ1 $OBJ2;
  UTILITY: 10;
  FAILURE: EXECUTE print "\n\nStack blocks failed!\n\n";
}

Plan: {
  NAME: "Clear a block"
  GOAL: ACHIEVE CLEAR $OBJ;
  CONTEXT: FACT ON $OBJ2 $OBJ;
  BODY:     ACHIEVE ON $OBJ2 "Table";
  EFFECTS: RETRACT ON $OBJ1 $OBJ;
  FAILURE: EXECUTE print "\n\nClearing block failed!\n\n";
}
```

BDI: propriétés

- Avantage
  - Choix de différents comportements d'agents possibles (prudent, déterminé, …)
  - Un vocabulaire adapté à la réflexion sur les objectifs dans un environnement dynamique
  - Un modèle générique de niveau élevé
  - Permet de conserver une grande liberté de choix sur la représentation des connaissances, la méthode de génération des actions, la planification, …

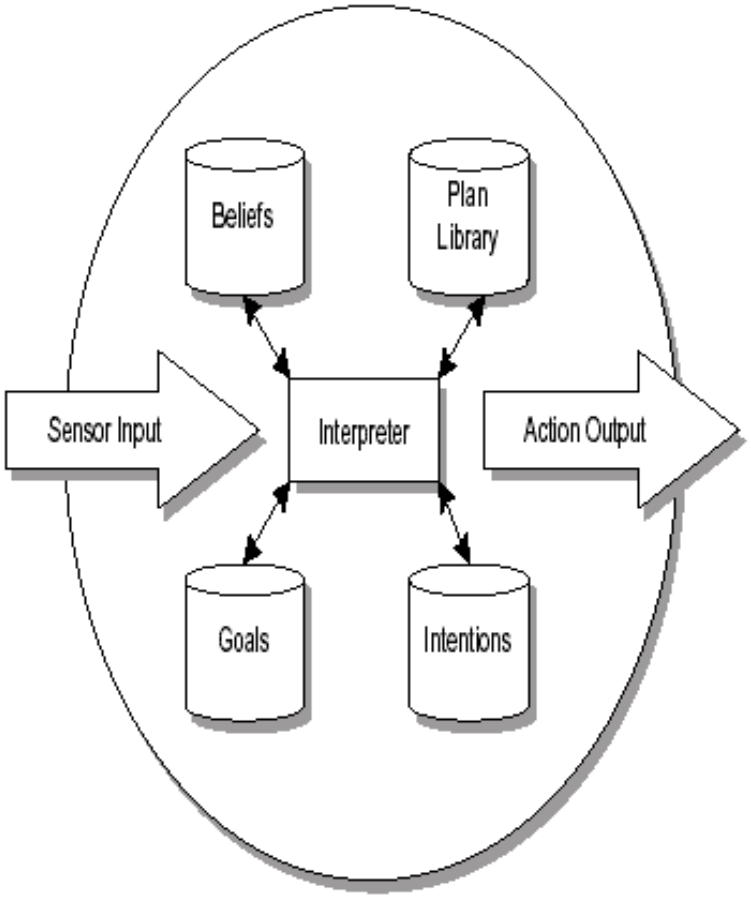
- Limites
  - Aucune précision sur les fonctions de sélection
  - Comment l'implémenter efficacement?
  - Comportement proactif et reactif, mais pas d'anticipation

BDI appliqué

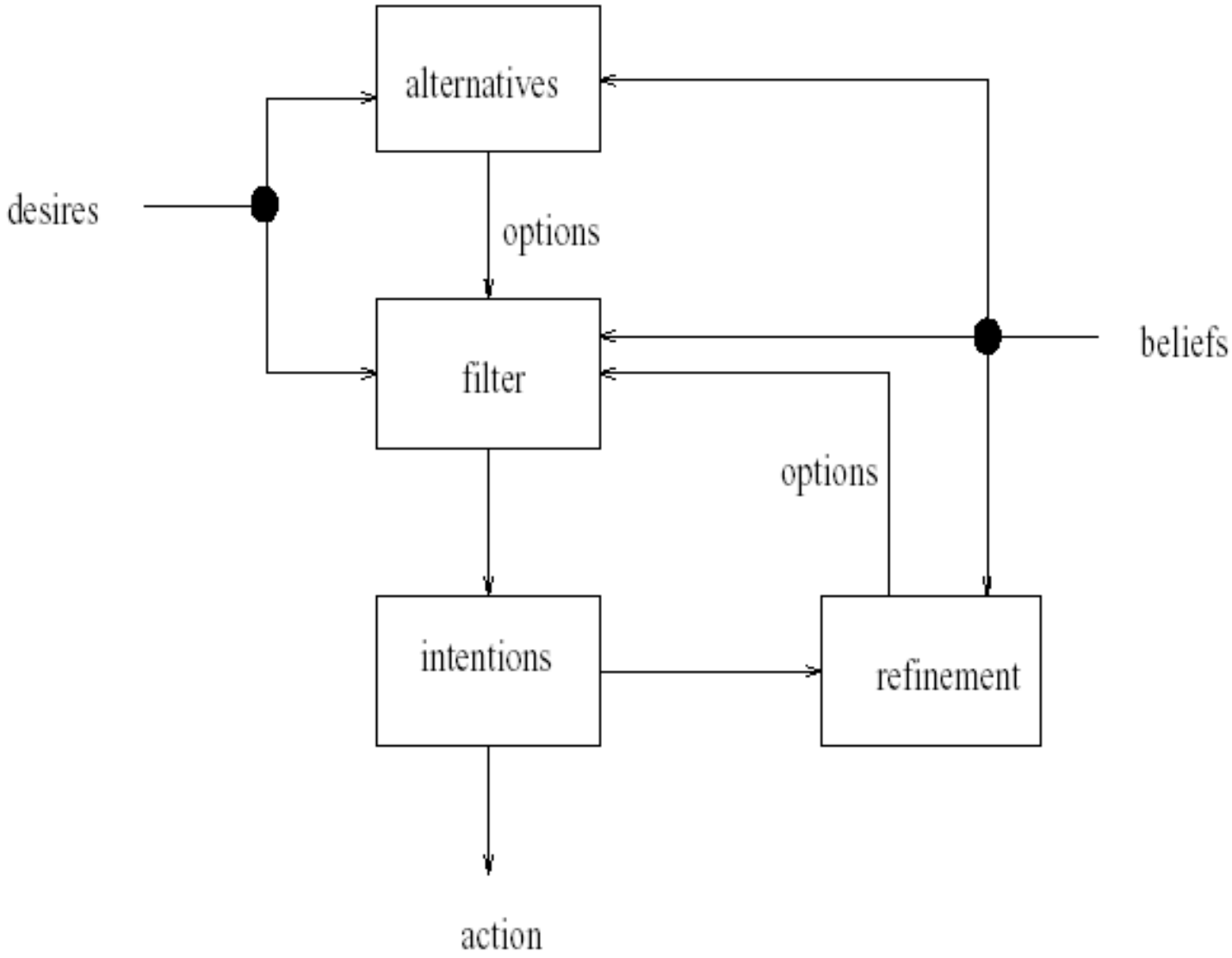
- PRS [Georgeff et al 87]
- Version java: Jam [Huber 99]
- Version Jade:
  - http://jason.sourceforge.net/faq/
  - Langage specifique: AgentSpeak

- Application
  - Contrôle du traffic aérien (tests a Sidney) OASIS
  - Management de process (SPOC)

- Ensemble de plans dans la base de plans
- Les désirs sont les plans dont les post-conditions sont remplies
- Plusieurs évaluations possibles des désirs
  - Meta-plans
  - Utilité

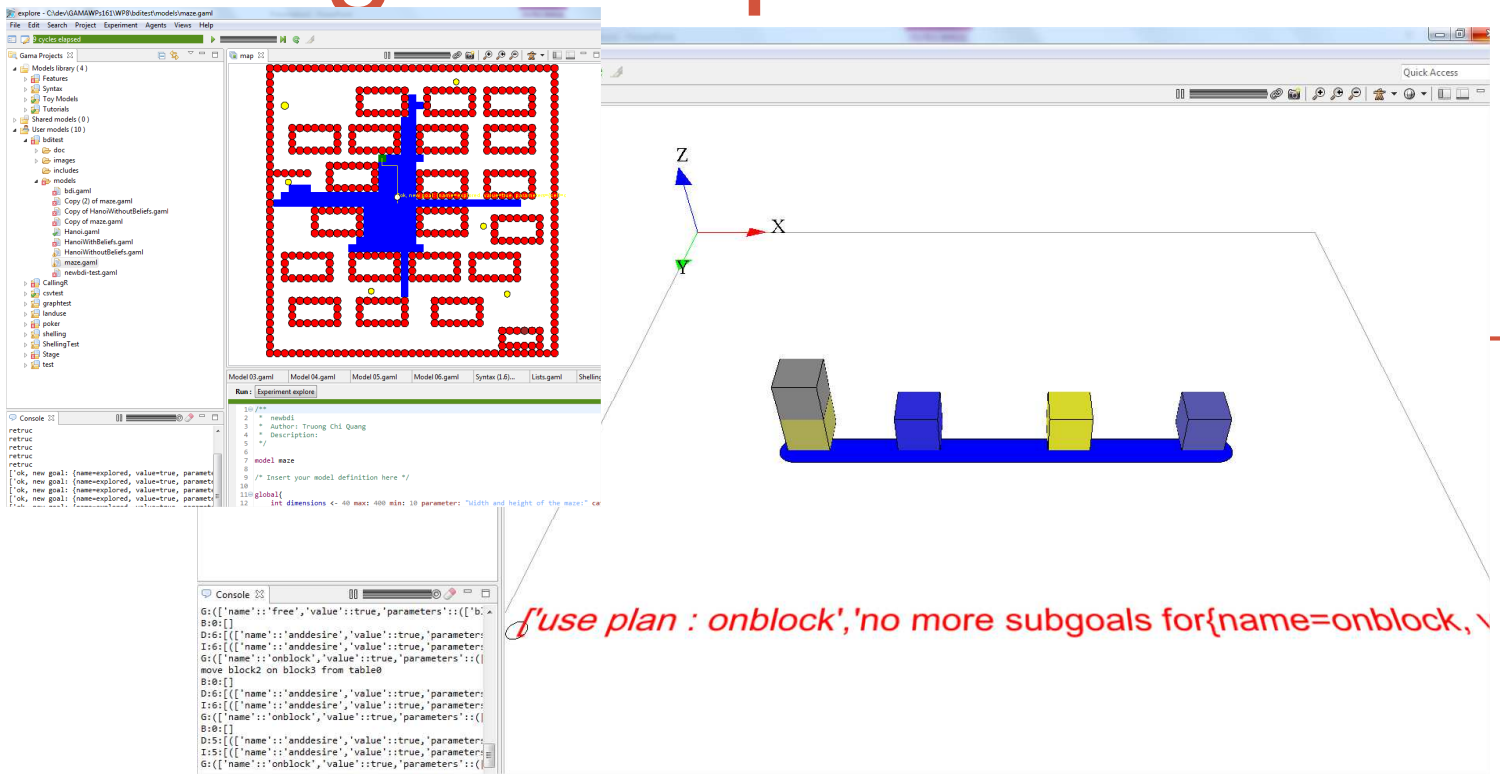


BDI appliqué: IRMA



Le BDI dans gama…

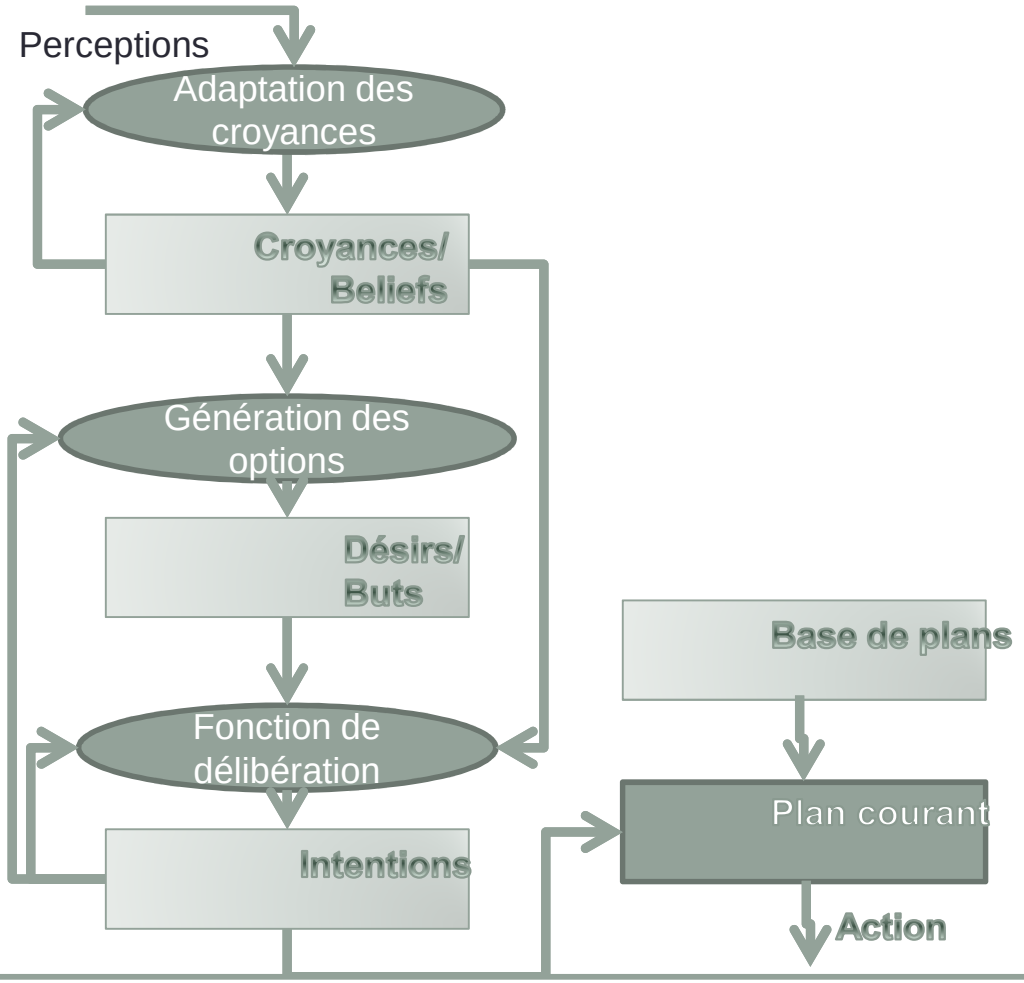
Plugin SimpleBDI



Architecture

- 3 Bases de « prédicats »
- - Beliefs
  - (ce que je croit être vrai sur le monde)
- - Desires
  - (ce que je veut)
- - Intentions
  - (ce que je suis en train d'essayer d'obtenir)

- Un prédicat? Une map…




```
map d1<-new_predicate(name:"onblock",value:true,parameters:["over"::block[0],"under"::block[1]],priority:1
```

Concrètement, dans GAMA

- Des objectifs (avec éventuellement des sous-objectifs), représentés avec des prédicats

```
create hanoiphilosopher
{
    set location<-{2,75};
    map d1<-new_predicate(name:"onblock",value:true,parameters:["over"::block[0],"under"::block[1]],priority:100);
    map d2<-new_predicate(name:"onblock",value:true,parameters:["over"::block[1],"under"::block[2]],priority:100);
    map d3<-new_predicate(name:"onblock",value:true,parameters:["over"::block[2],"under"::block[3]],priority:100);
    map d4<-new_predicate(name:"onblock",value:true,parameters:["over"::block[3],"under"::block[4]],priority:100);
    map da12<-new_predicate(name:"anddesire",value:true,parameters:["first"::d1,"second"::d2],priority:100);
    map da34<-new_predicate(name:"anddesire",value:true,parameters:["first"::d3,"second"::d4],priority:100);
    map masterplan<-new_predicate(name:"anddesire",value:true,parameters:["first"::da12,"second"::da34],priority:100);
    do add_desire(predicate:masterplan);
}
```

- Des plans, qui ont comme condition d'activation un type d'objectif

```
plan anddesire when:is_current_goal(new_predicate(name::"anddesire")) priority:3 executed_when:true
{
        map currentgoal<-get_current_goal();
    write("and "+currentgoal);
    map goalparams<-((currentgoal at "parameters") as map);
    map subdesire1<-goalparams at "first";
    map subdesire2<-goalparams at "second";
    list subdesire1<-eval_gaml((goalparams at "first") as string);
```

Des « perceptions » qui ne sont rien d'autres que des reflexes pour percevoir le monde

```
perceive myperceive
{
    my_cell<-first(cell overlapping self);
    do add_belief(new_predicate("explored","true",["cell"::my_cell]));
    add my_cell to:zone_exploree;
    if (!(zone_libre contains my_cell))
```

Des mots clés pour ajouter/supprimer des différentes bases et manipuler les objectifs

```
is_current_goal(new_predicate(name::"free"))
        map masterplan<-new_predicate(name:"anddesire",value:true,parameters:["first"::da12,"second"::da34],priority:100);
currentgoal_on_hold();
        map currentgoal<-get_current_goal();
add_desire(new_predicate(name::"free",value::true,priority::100,parameters::["block"::overblock]),currentgoal);
        do add_desire(predicate:masterplan);
 if (!testgoal(subdesire1))
 {
     do add_subgoal(currentgoal,subdesire1);
     do add_desire(subdesire1);
 }
        do add_belief(new_predicate("explored","true",["cell"::my_cell]));
```

La boucle de raisonnement derrière

- Si il existe des perceptions, perceive…
- Retirer l'intension courante avec proba PersistentCoef
- Si l'intension courante n'a pas de plan disponible ou est OnHold (avec des sous-objectifs non atteints), choisir un nouveau désir
- Si un plan persistant est en cours, le changer avec proba PersistentTaskCoef
- Si il n'y a pas d'intension courante, choisir un nouveau désir
- Si pas de plan en cours, choisir un plan pour l'intension courante
- Exécuter le plan en cours
- Si le plan est considéré comme fini, pas de plan courant
- Si l'intension courante est dans la base des faits, retirer de la base d'intension et désirs, supprimer le plan courant

Exemple
- D=(Et(On(A,B),On(B,C),
  - On(A,B)
  - On(B,C)
- I=Et(On(A,B),On(B,C)) – OnHold

- P= Pour (Et(.)), PlanA
  - Pour (On(.), PlanB

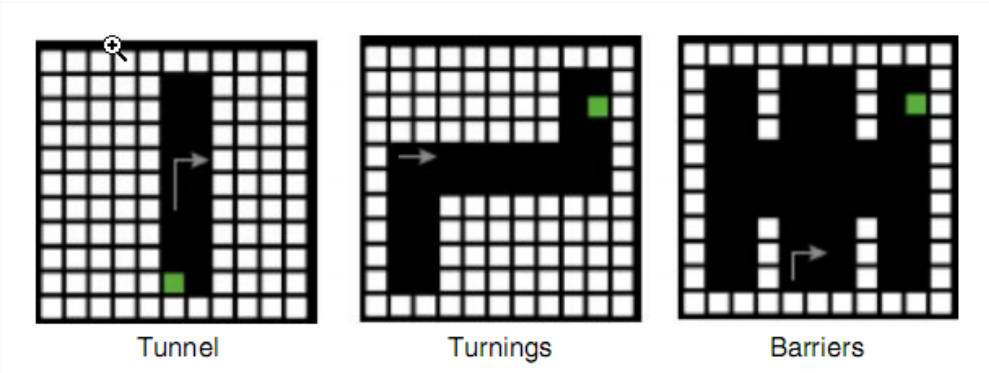
- PlanEnCours=-

Un exemple appliqué: Black & White [2001]

- Beliefs:
  - Valeurs d'attributs (objets uniques)
  - Arbre de décisions (ensembles)
  - La créature ne forme des croyances que sur des objets qu'elle a perçu
- Désires:
  - Perceptron
- Exemples d'apprentissages:
  - Faits (emplacement de villages)
  - Méthodes (comment réaliser des actions)
  - Priorités (une action est plus importantes qu'une autre)
  - Utilité d'une action/objet pour un désir particulier

BDI: meta-raisonnement

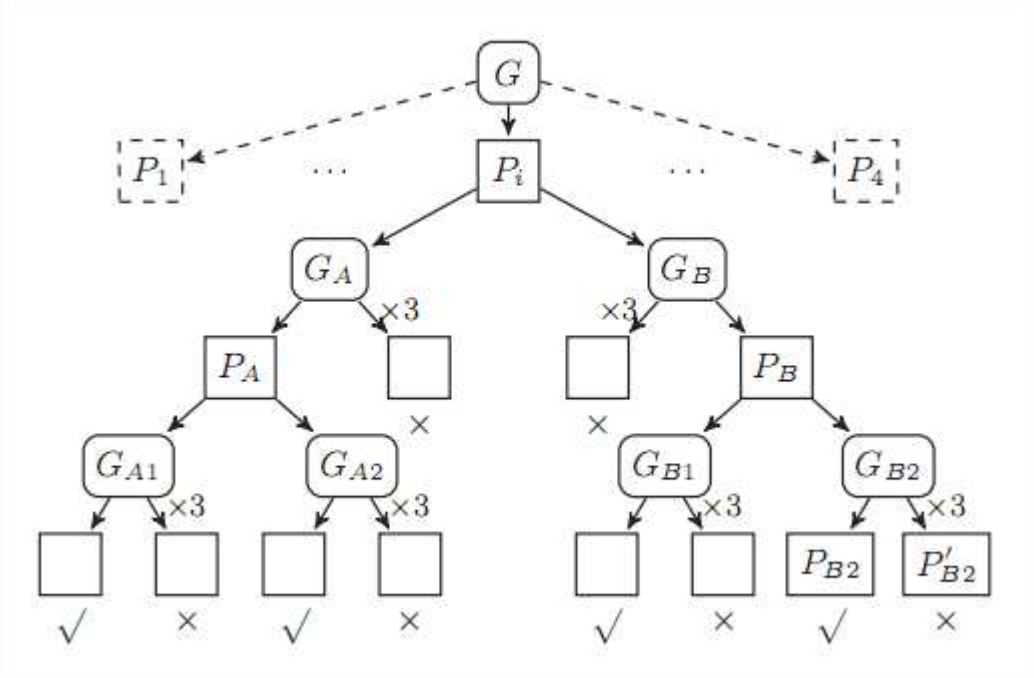
- JAAMAS [Subugdja et al. 2009]
- Objectif:
  - utiliser le modèle BDI pour apprendre et évaluer les plans
- Méthode:
  - Un objectif d'evaluation de plan et un ensemble de primitives de création/modification de plans



Action	Input
[create-plan:]	[schema:]
[update-plan:]	[old:]; [new:]
[remove-plan:]	[old:]
[make-plan-schema:]	[in-plan:]; [goal:]; [precond:]
[append-plan-acts:]	[in-plan:]; [acts:]; [cond:]
[insert-plan-acts:]	[in-plan:]; [acts:]; [cond:]; [p
[remove-plan-acts:]	[in-plan:]; [pos:];
[inc-plan-acts-times:]	[in-plan:]; [inc-fact:]
[dec-plan-acts-times:]	[in-plan:]; [dec-fact:]
[set-plan-acts-times:]	[in-plan:]; [times:]
[set-plan-rep-cond:]	[in-plan:]; [cond:]
[set-plan-acts-else:]	[in-plan:]; [acts:]
[inc-plan-util:]	[in-plan:]; [inc-fact:]
[dec-plan-util:]	[in-plan:]; [dec-fact:]
[get-action-plan:]	[in-plan:]; [index:]

BDI: apprentissage sur les plans

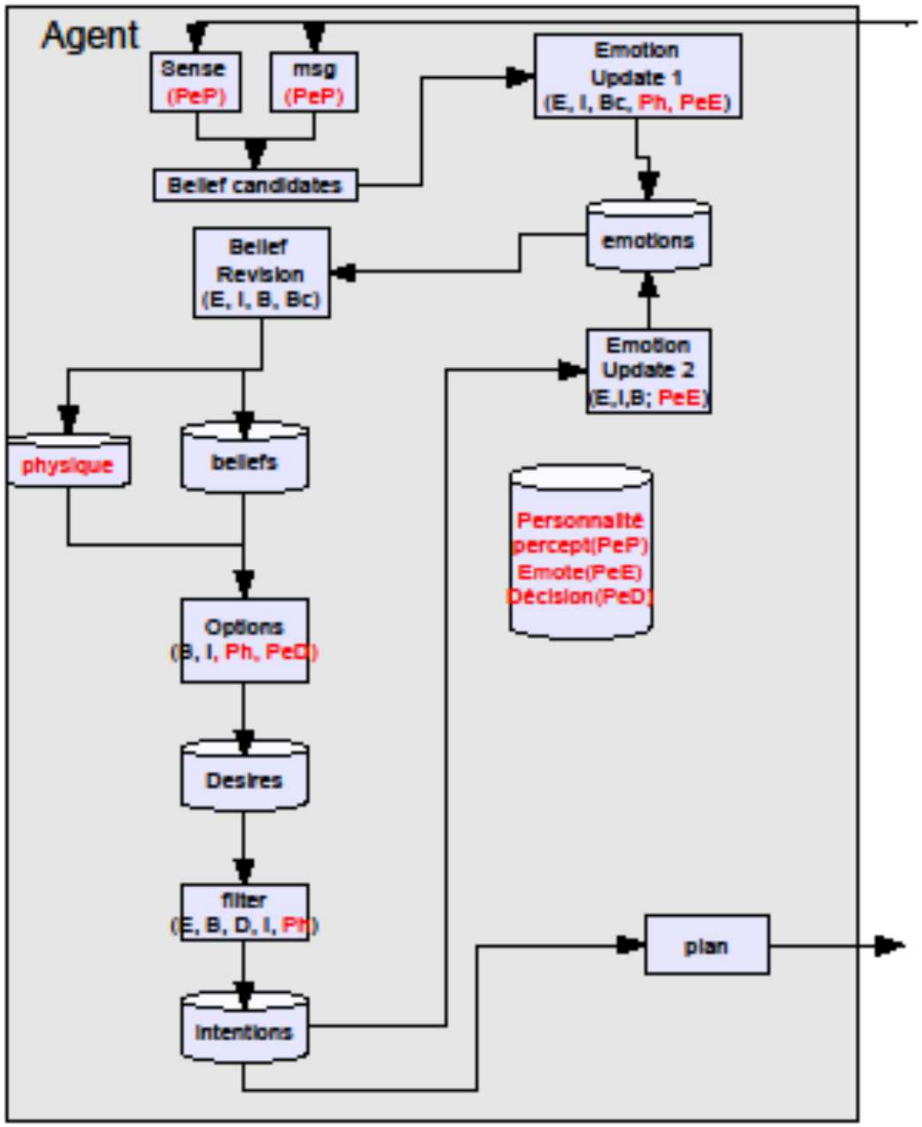
- AAMAS [Singh et al. 2010]
- Objectif:
  - Construire les meilleurs arbres de plans pour sélectionner les bonnes intentions dans un contexte donné.
- Principes:
  - Base de plans vue comme un arb de décision
  - Apprentissage par renforcement après chaque exécution
  - Choix de configuration d'hyperparametres pour le type d'exploration/exploitation:
    - Utilisation du taux d'exploration de sous-plans
    - Remontée des échecs uniquement lorsque le taux de succès est stabilisés
- Résultats:
  - Apprentissage des meilleurs plans/sous plans avec des arbres simulés
  - Bons résultats de paramètres prenant en compte la couverture mais sans stabilisation



BDI: émotions

[Jiang et al 2007]
[Jones et al 2009]

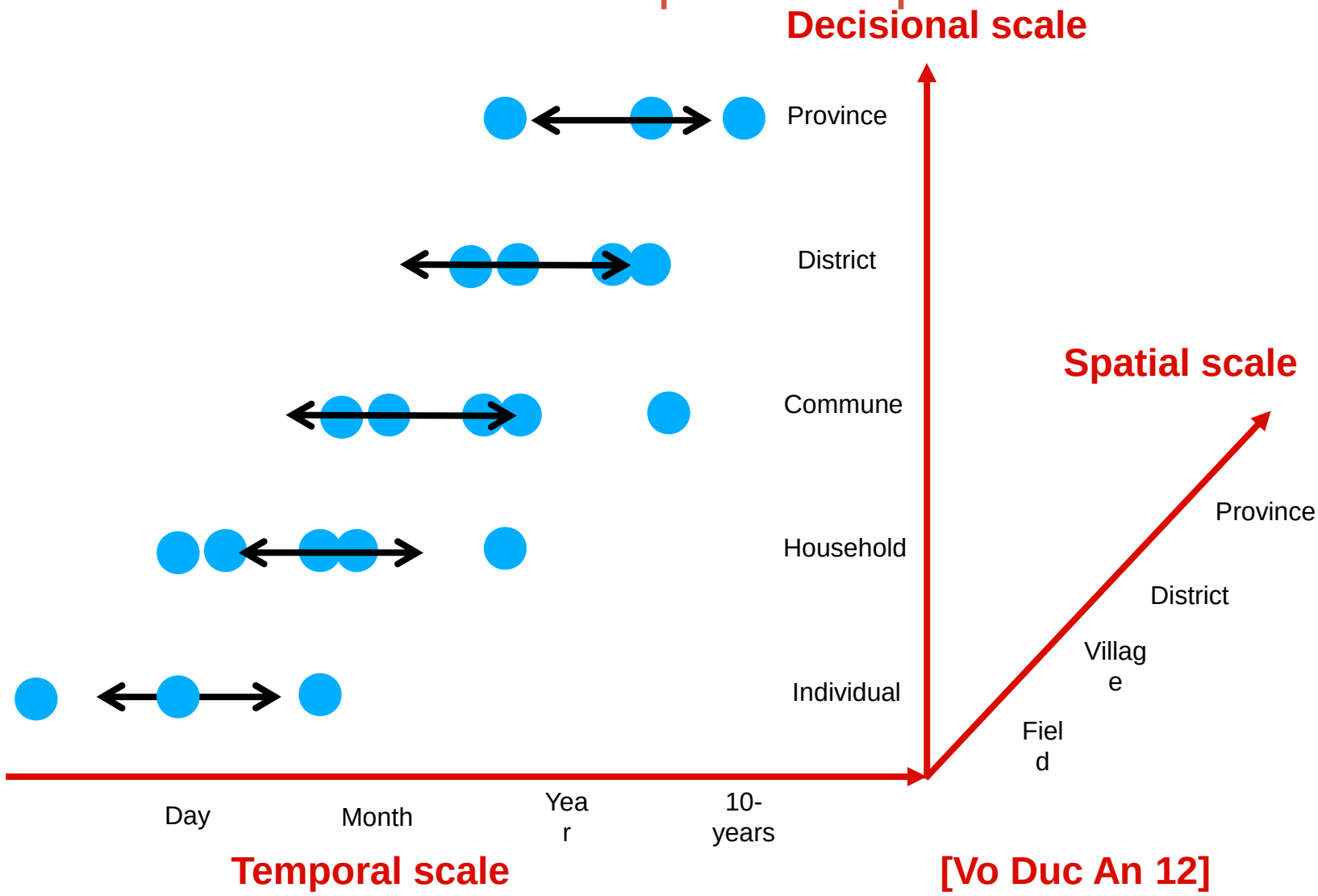
- Objectif:
  - Simuler un comportement réaliste
- Ex:
  - peur/espoir : la peur joue un rôle important dans une situation de crise. La présence ou la perspective d'un danger agit sur la peur des agents, pouvant aller jusqu'à la peur panique. Cette émotion affecte les réactions et peut, par exemple, entraîner la fuite de l'agent.
  - Empathie: ressent les émotions des autres



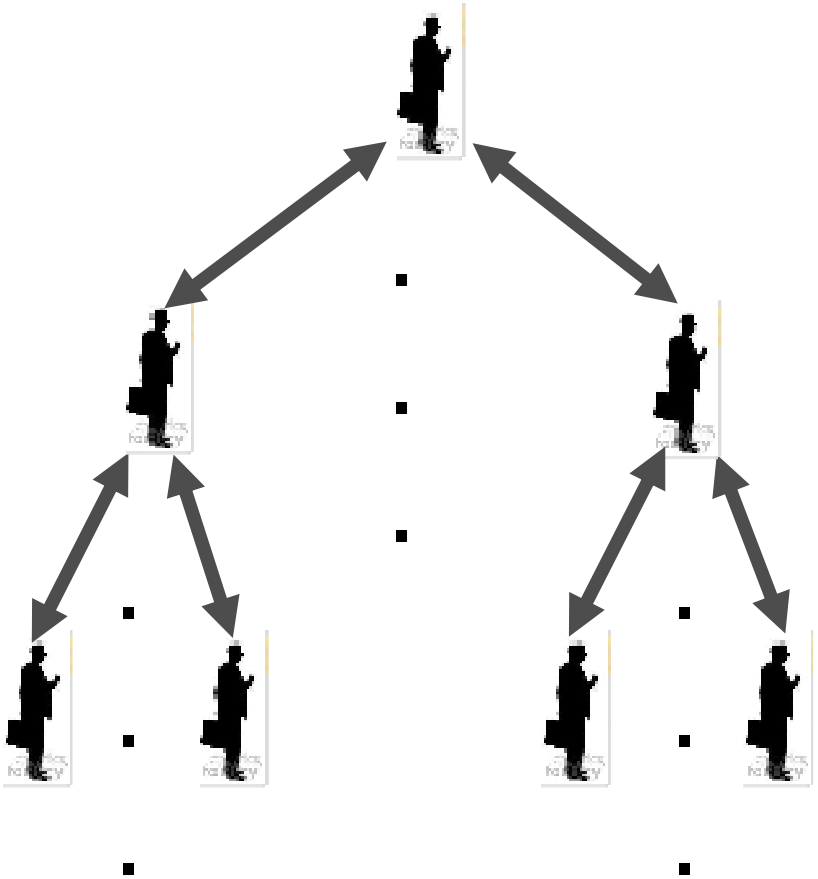
Modèles d'agent

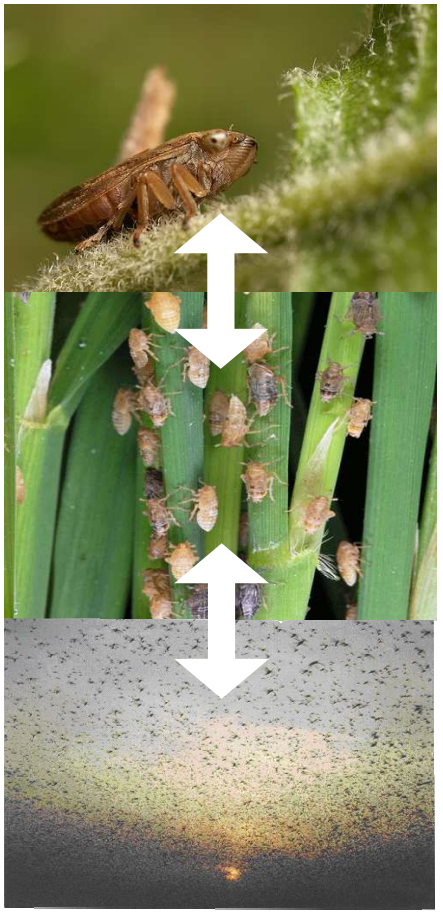
- Introduction
- Modèle Zero-Intelligence
- Modèles réactifs
- Modèles interactionnel
- Agents cognitifs
- **Modèles multi-niveaux**
- Modèles reflexifs

Certains modèles possèdent des agents agissant à des échellent différentes de temps et d'espace



Certains niveaux peuvent interagir entre eux, généralement de façon hiérarchique





[Vo Duc An 12]

Certains agents peuvent être représentés à plusieurs niveaux

Entities

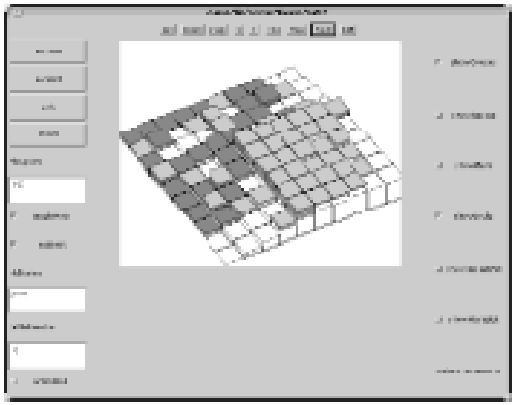
Insectes

Group
Insectes (dans un groupe)

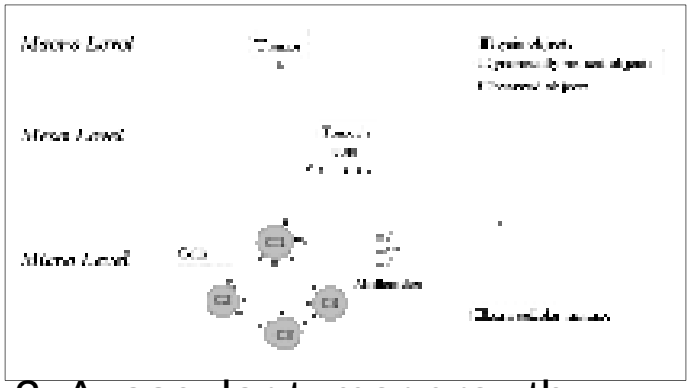
Cloud
Insectes (dans un nuage)

[Vo Duc An 12]

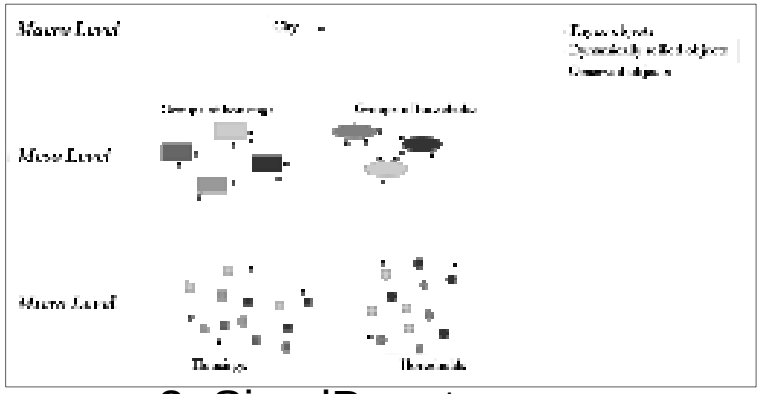
Les modèles multi-niveaux permettent de représenter des entités interagissant à des échelles différentes



1. RIVAGE [Servat and al., 1998]



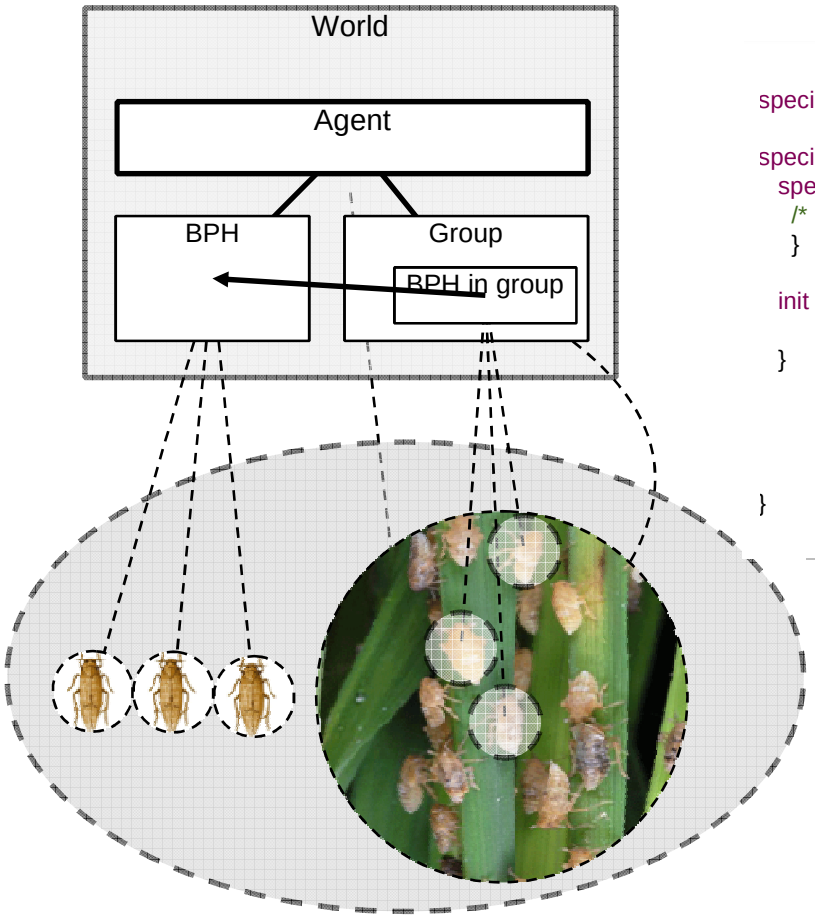
2. Avascular tumor growth [Lepagnot and Hutzler, 2009]



3. SimulBogota [Quijano and al., 2010]

[Vo Duc An 12]

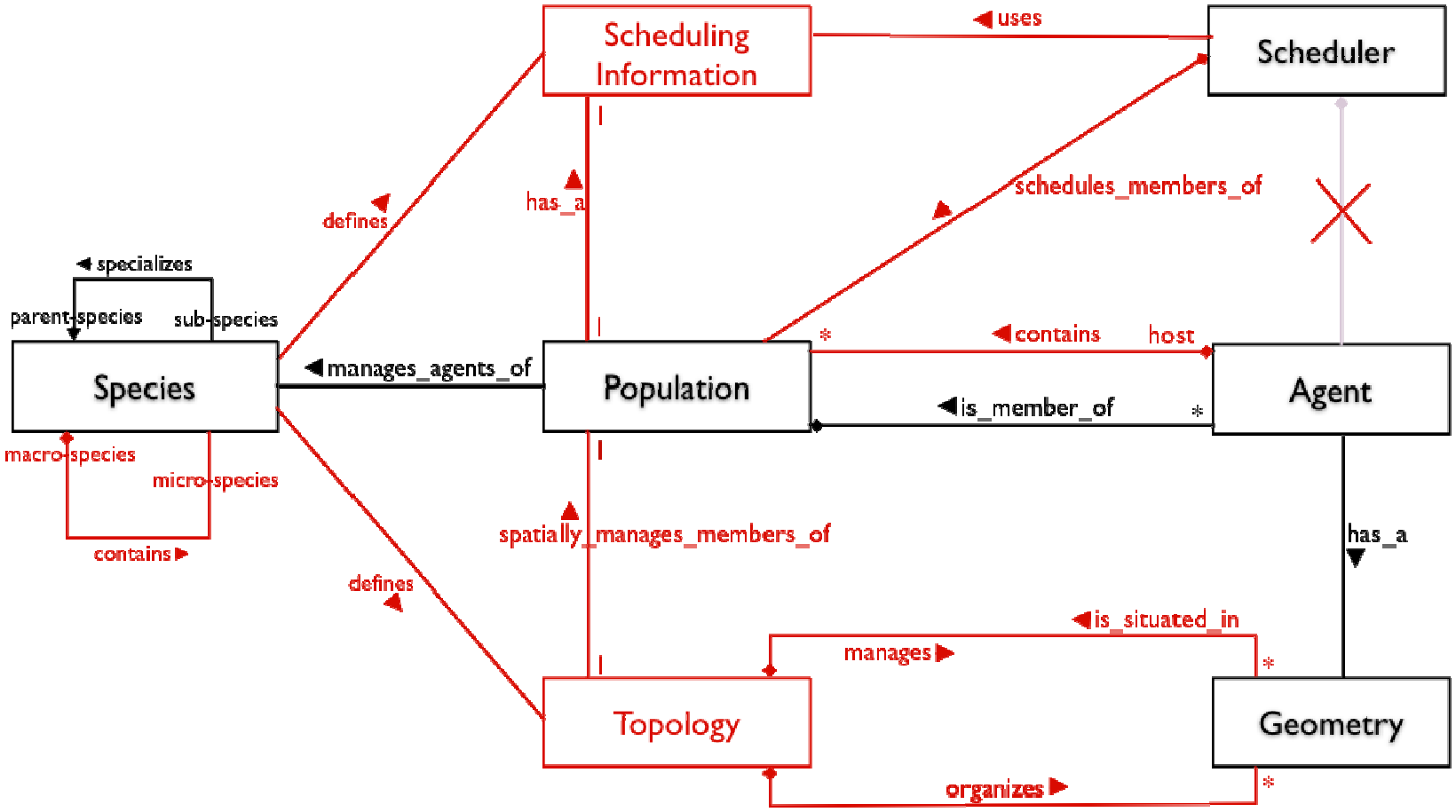
Dans Gama, on défini des sous-especes créant une hiérarchie d'espèce et une hiérarchie d'agents




```
species BPH { … }

species Group {
  species BPH_in_group parent: BPH {
    /* New or redefined attributes, behaviors */
    }

  init {
                capture (BPH inside shape) as: BPH_in_group;
  }

        reflex when : dying {
                release list (BPH_in_group) as : BPH ;
        }
}
```

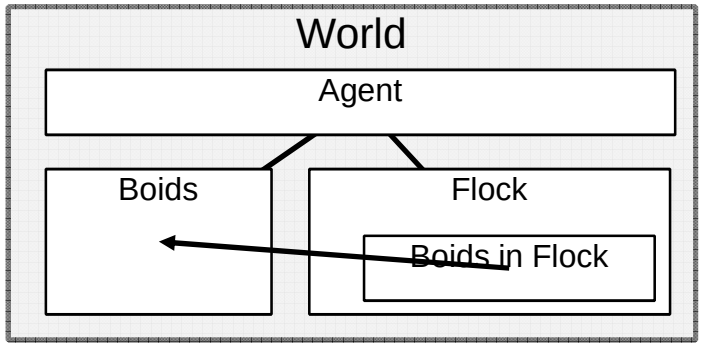
[Vo Duc An 12]

Chaque espèce peut avoir son scheduler, sa topologie et son comportement modifiée par sa macro-espèce



[Vo Duc An 12]

Les sous-espèces peuvent être définie sans modifier l'espèce parent

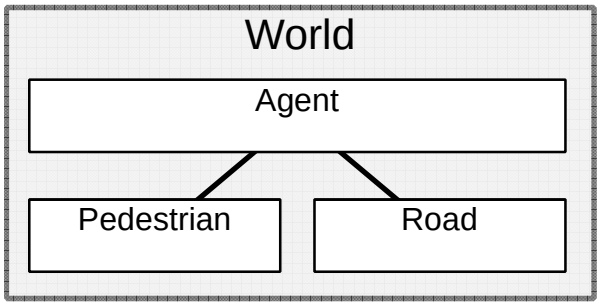


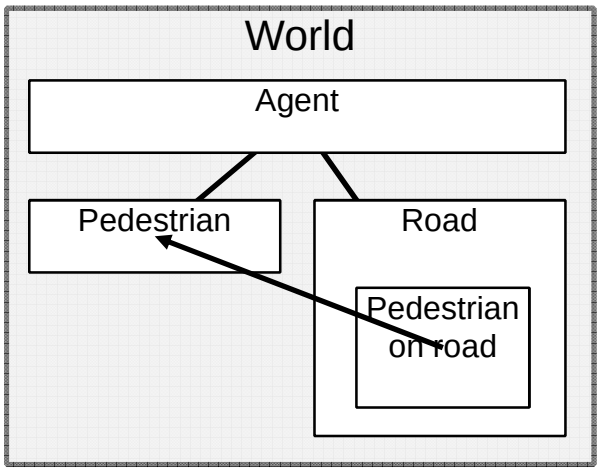

```
species boids  {
        /* No change */
        }

species flock {
        species boids_in_flock parent: boids {
                aspect image { draw geometry: circle(1); }
        }

        aspect default {
                draw geometry: polygon(list(members));
        }
}
```

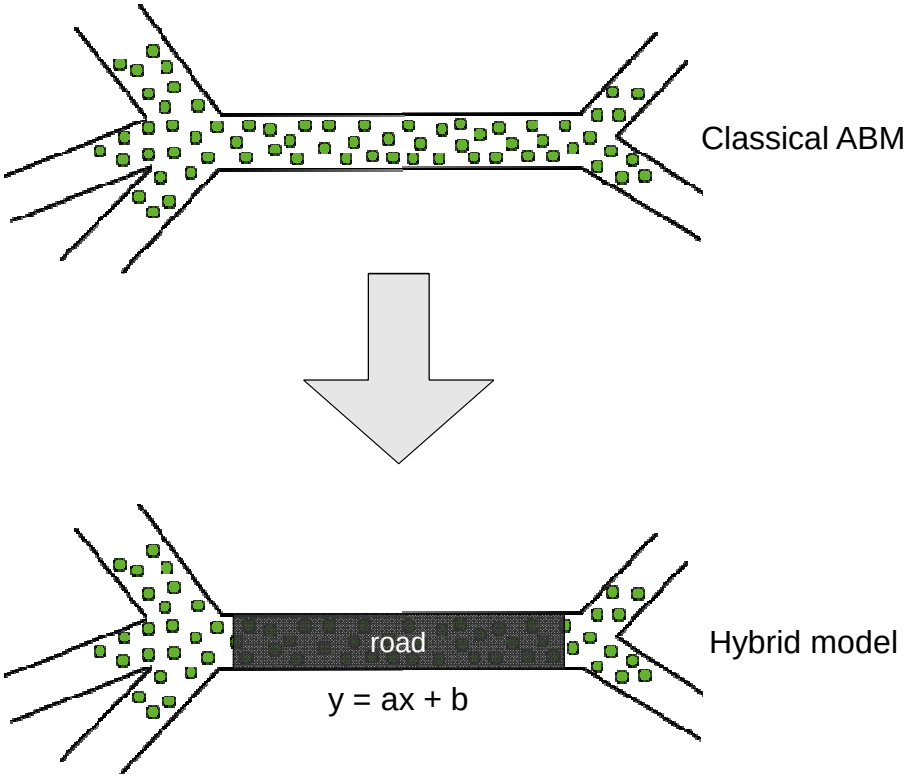
Boids: 91
Boids in flocks: 209
Flocks: 13

[Vo Duc An 12]

Un des avantages est de pouvoir agréger les comportements de micro-agents simples



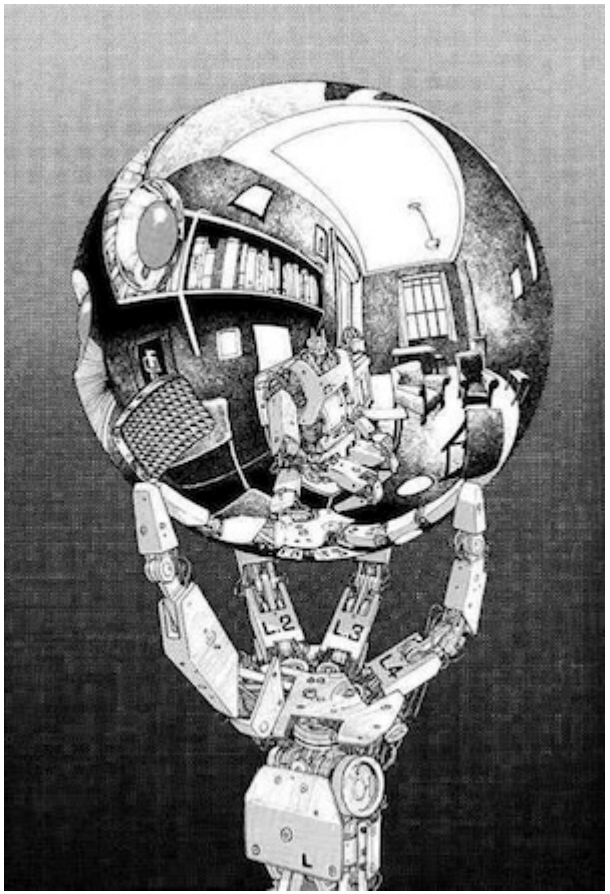




[Vo Duc An 12]

Modèles d'agent

- Introduction
- Modèle Zero-Intelligence
- Modèles réactifs
- Modèles interactionnel
- Agents cognitifs
- Modèles multi-niveaux
- **Modèles réflexifs**



AM

[Lenat 1978, Davis et Lenat 1982]

- Objectif:
  - Découvrir de nouveaux concepts mathématiques intéressants
- Méthode:
  - Représentation des concepts sous forme de Frames
  - Utilisation d'heuristiques pour créer et enrichir les concepts à partir des précédents
    - Suggérer une tache
    - Créer un concept
    - Ajouter/Supprimer une info à un slot
  - Contrôle: un agenda de taches avec un temps alloué par tache

AM

- État d'origine:
  - 115 concepts
  - 242 heuristiques
    - « Si f est une fonction A->B et B' un sous ensemble extrême de B, définir f-1(B')

- Résultats:
  - nouveaux concepts tels que les nombres premiers, la multiplication, nombre ayant plus de diviseurs que tous les nombres plus petits (inconnu de Lenat)

NAME: Primes

STATEMENT: Numbers with two divisors

SPECIALIZATIONS: Odd-primes, Small-primes,
: Pair-primes

GENERALIZATIONS: Positive numbers

IS-A: Class-of-numbers

EXAMPLES:
Extreme-exs: 2,3
Extreme-non-exs: 0,1
Typical-exs: 5,7,11,13,17,19
Typical-non-exs: 34,100

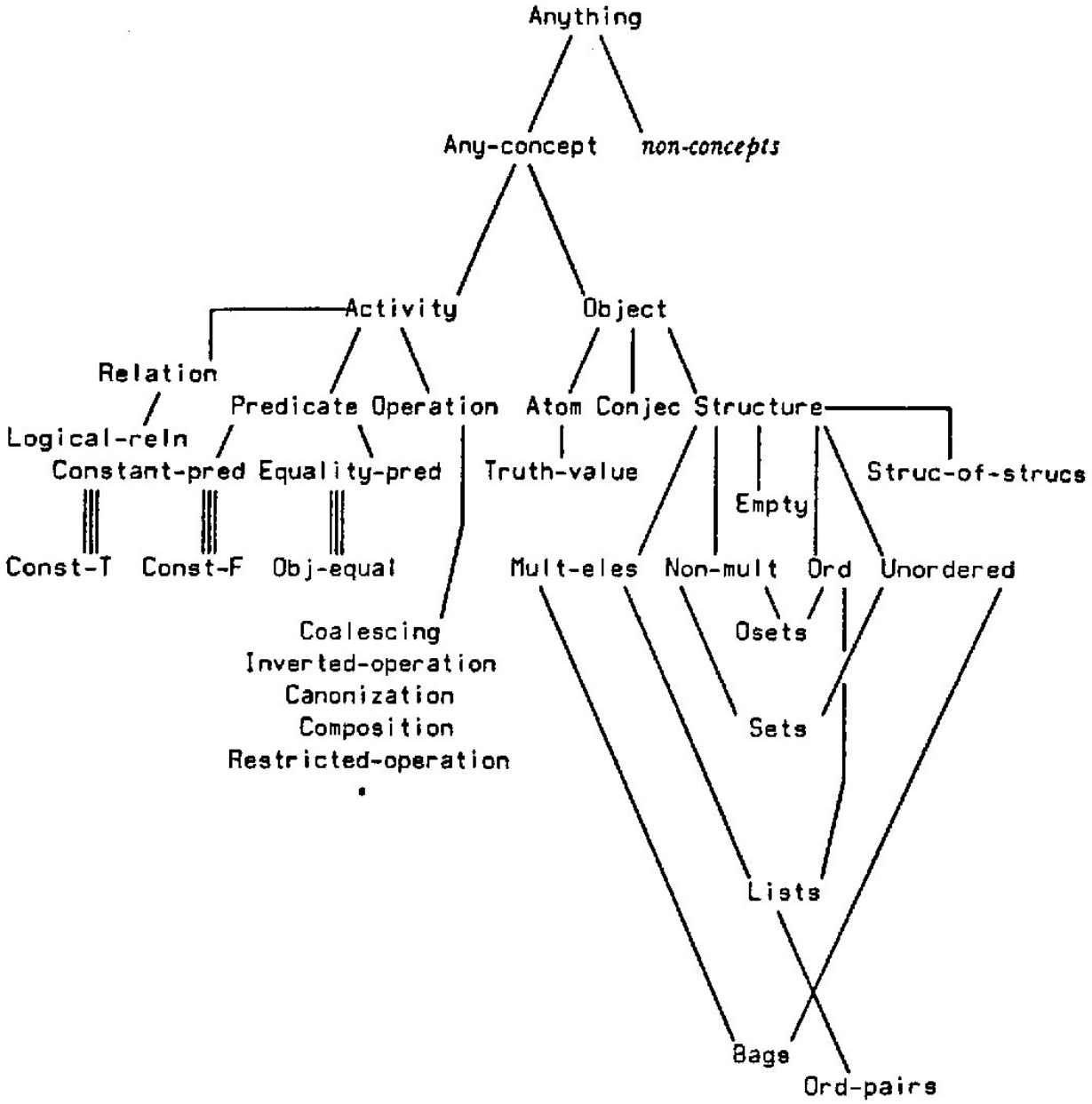
CONJECTURES:
Good-conjects: Unique-factorization
Good-conject-units: Times, Divers-of, Exponentiate,
Nos-with-3-divis, Squaring

ANALOGIES: Simple Groups

WORTH: 800

ORIGIN: Application of H2 to Divisors-of
Defined-using: Divisors-of Creation-date: 3-19-76

HISTORY:
Good Examples: 840 Bad Examples: 5000
Good Conjectures: 3 Bad Conjectures: 7


Anything
Any-concept
non-concepts
Activity
Object
Relation
Predicate
Operation
Atom
Conjec
Structure
Logical-reln
Constant-pred
Equality-pred
Truth-value
Empty
Struc-of-strucs
Const-T
Const-F
Obj-equal
Mult-eles
Non-mult
Ord
Unordered
Coalescing
Inverted-operation
Canonization
Composition
Restricted-operation
Osets
Sets
Lists
Bags
Ord-pairs

AM: limites

- Intervention de l’utilisateur pour guider le programme
- Les heuristiques de recherches sont fixes

Eurisko

[Lenat 1982, 1983a,b

- Objectif: chercher de nouveaux concepts, mais aussi de nouvelles heuristiques et de nouveaux types de slots
- Application: jeux de société de constitution de flotte spatiale
- Méthode:
  - Les heuristiques sont elles-mêmes représentées sous forme de Frames et donc modifiables
  - Introduction d'heuristiques de découverte d'heuristiques
    - « Si cette regle est souvent appliquée avec succès, généraliser la »
    - Si le résultat de f est souvent sans intérêt, baisser l'interet de f

- Ex du Frame EnergyGun:

Name: EnergyGun
Generalizations: (Anything Weapon)
AllIsA: (GameConcept GameObj Anything Category WeaponType
DefensiveWeaponType OffensiveWeaponType Obj
AbstractObj PhysGameObj PhysObj)
IsA: (DefensiveWeaponType OffensiveWeaponType PhysGameObj)
MyWorth: 400
MyInitialWorth: 500
Worth: 100
InitialWorth: 500
DamageInfo: (SmallWeaponDamage)
AttackInfo: (EnergyGunAttackInfo)
NumPresent: NEnergyGuns
UspPresent: EnergyGunUSP
DefendsAs: (BeamDefense)
Rarity: (0.11 1 9)
FocusTask: (FocusOnEnergyGun)
MyIsA: (EuriskoUnit)
MyCreator: DLenat
MyTimeOfCreation: "4-JUN-81 16:19:46"
MyModeOfCreation: (EDIT NucMissile)

- 3 niveaux de contrôle:
  - Supérieur: choix du concept
  - Intermédiaire: choix de la tache dans l'agenda du concept
  - Inférieure: choix de l'heuristique pour cette tache

- Eurisko joue contre lui-même pour s'améliorer progressivement:
  - Enrichir le concept « jouer à la bataille »
  - Favoriser les heuristiques gagnantes

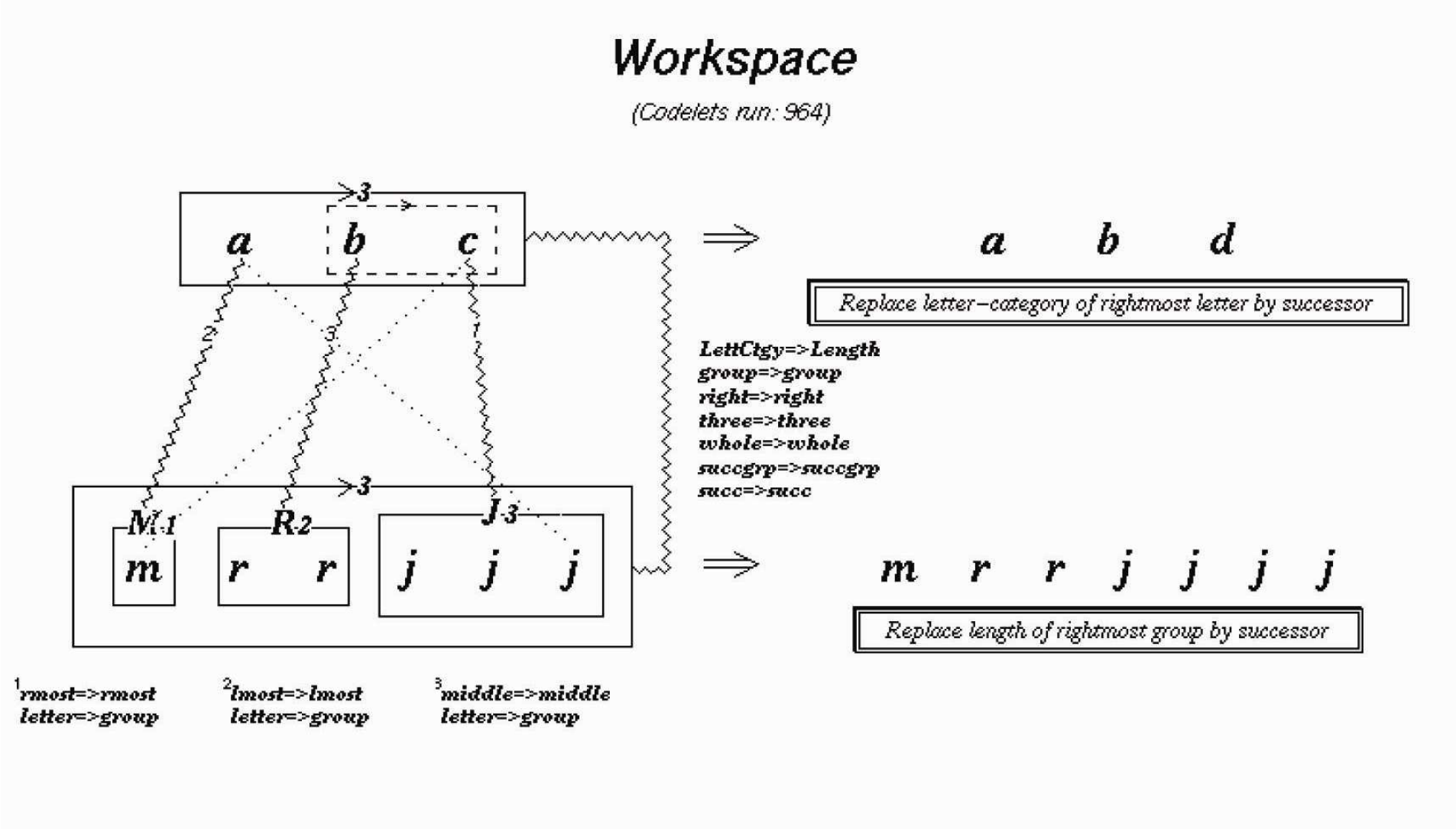
Eurisko

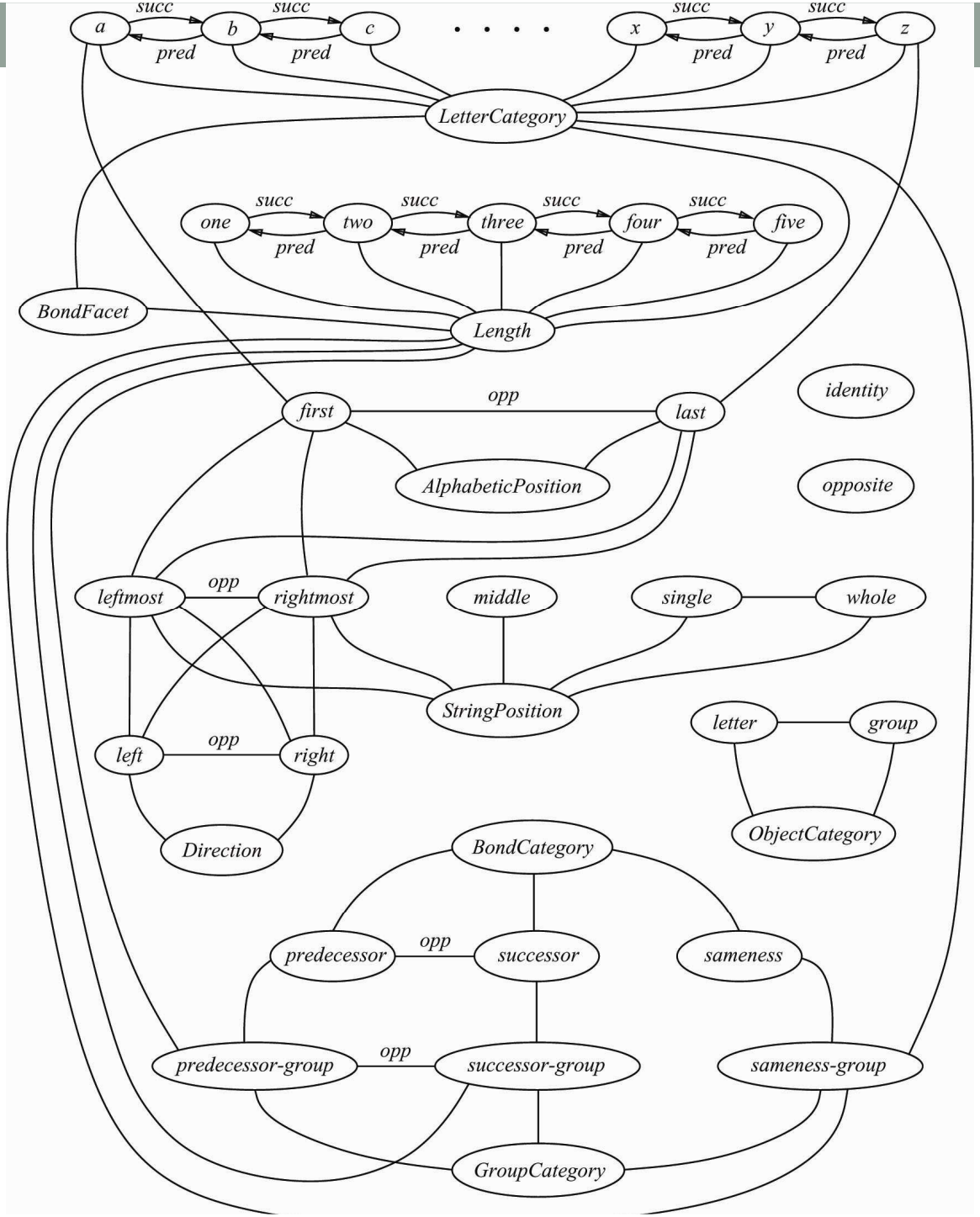
- Résultats:
  - Découverte d’un concept permettant de ne jamais perdre (un petit vaisseau inatteignable.
  - Découverte d’heuristiques efficace de construction de flottes
  - Vainqueur du concours 2 années de suite
- Limites:
  - Toujours une forte intervention de l’utilisateur

CopyCat, MetaCat, MusiCat

[hofstadter 90, 93] [Marshall 99,02,04] [Hofstadter 04+]

- Objectif:
  - Simuler un raisonnement par analogie.
  - Résoudre un problème d'analogie textuelle (CopyCat, MetaCat) ou musicale (MusiCat)
  - ABC -> ABD, ABBCCC -> ?
  - ABC->ABD, XYZ-> ?
- Méthode:
  - Ensemble d'heuristiques (coglets) qui vont s'activer et agir pour construire progressivement et collectivement le résultat final.
  - Les coglets ont une probabilité de choix dépendant de leur « degré d'activation ». L'activation est transmise par les concepts activés précédemment.
  - Le graphe des concepts est donné dans CopyCat, variable dans MetaCat.
- Version java: http://itee.uq.edu.au/~scottb/_Copycat/


Workspace
(Codelets run: 964)
a b c
3
M 1
R 2
J 3
m
r r
j j j
a b d
Replace letter–category of rightmost letter by successor
LettCtgy=>Length
group=>group
right=>right
three=>three
whole=>whole
succgrp=>succgrp
succ=>succ
m r r j j j j
Replace length of rightmost group by successor
1 rmost=>rmost
letter=>group
2 lmost=>lmost
letter=>group
3 middle=>middle
letter=>group


a
b
c
x
y
z
succ
pred
LetterCategory
one
two
three
four
five
BondFacet
Length
identity
first
last
opp
AlphabeticPosition
opposite
leftmost
rightmost
middle
single
whole
StringPosition
letter
group
left
right
Direction
ObjectCategory
BondCategory
predecessor
successor
sameness
predecessor-group
successor-group
sameness-group
GroupCategory

CopyCat et MetaCat

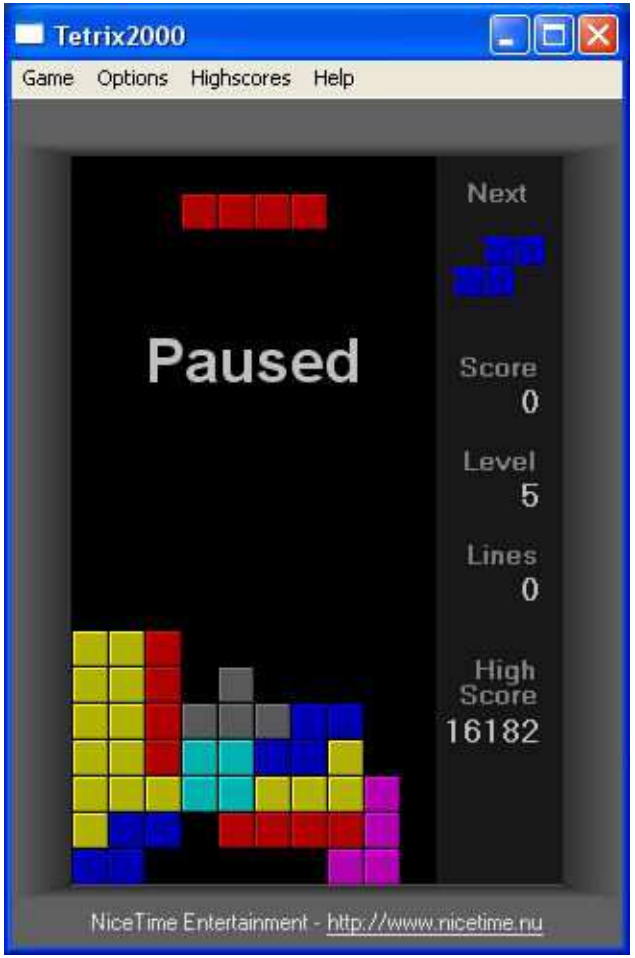
- Résultat
  - Analogies semblables aux résultats humains
  - Pas de problème d'intervention de l'utilisateur car le but n'est pas d'obtenir un seul résultat
  - Sur MusiCat, construction de nocturnes « à la Chopin »

- Limites
  - Pas d'apprentissage de règles ou de concepts

Arco

[Caillou 03, 04, 05]

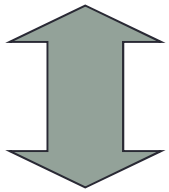
- Tetris: jeu temps réel
  - Importance de l'apprentissage de nouveaux concepts et heuristiques: modification du comportement et de la sémantique ([Kirsh et Maglio 94])
    - Perception
    - Action
    - Stratégie
- Ex : Si il y a un carré rouge en haut, la pièce actuelle est une barre



Objectif

Adaptation

- Comment permettre à l'agent de s'adapter à son environnement ?

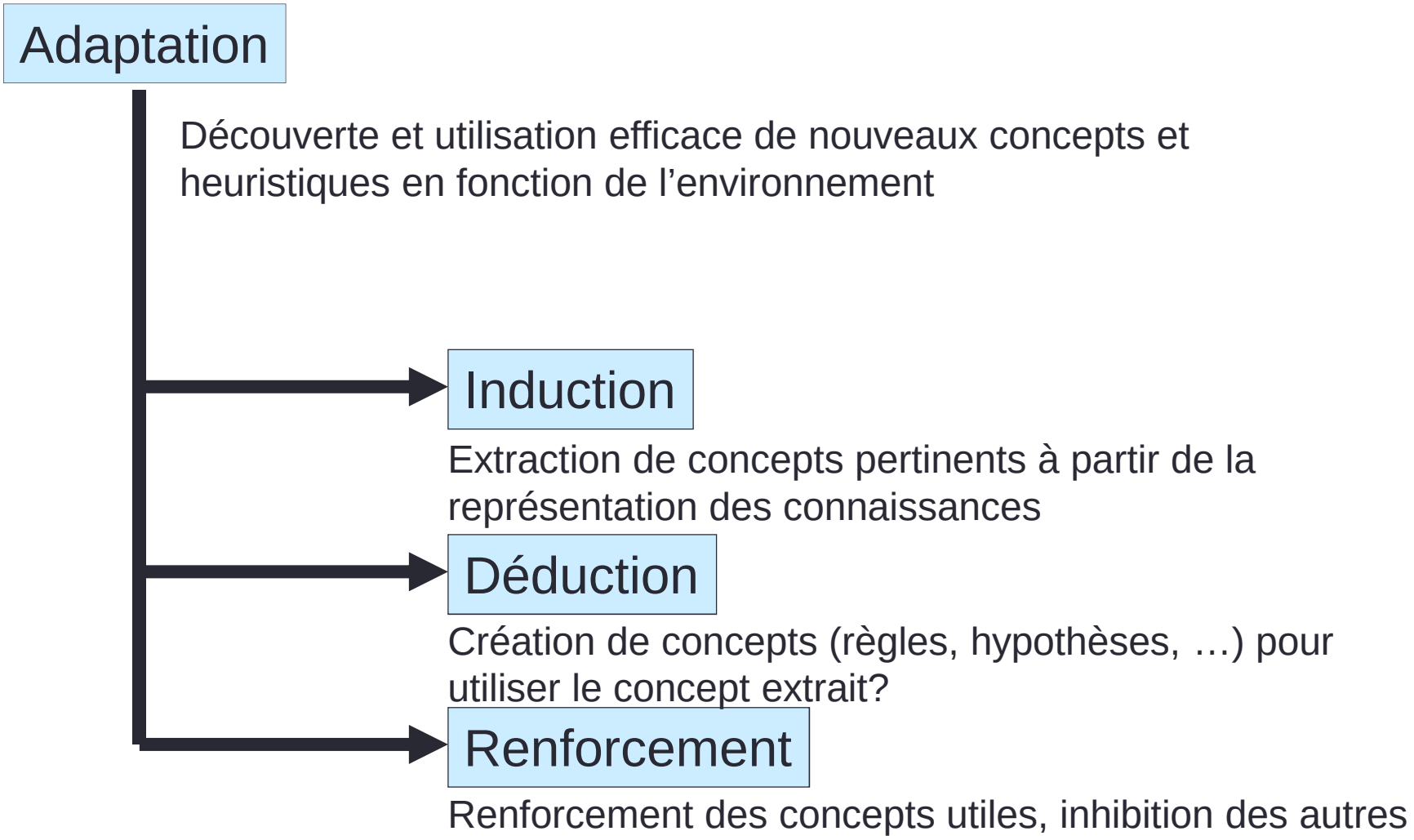


- Comment découvrir et utiliser efficacement de nouveaux concepts et heuristiques en fonction de l'environnement ?

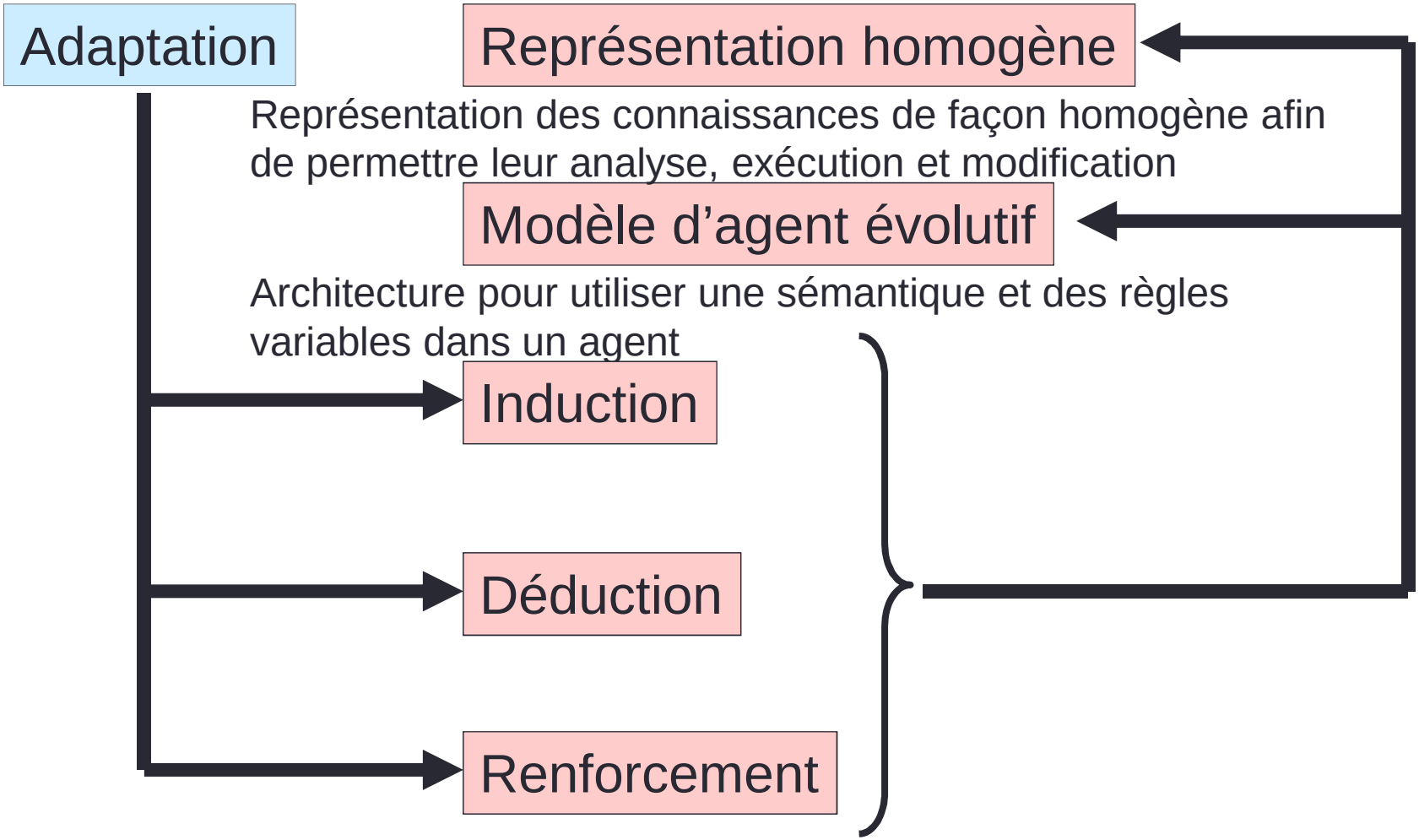
Caractéristiques du domaine

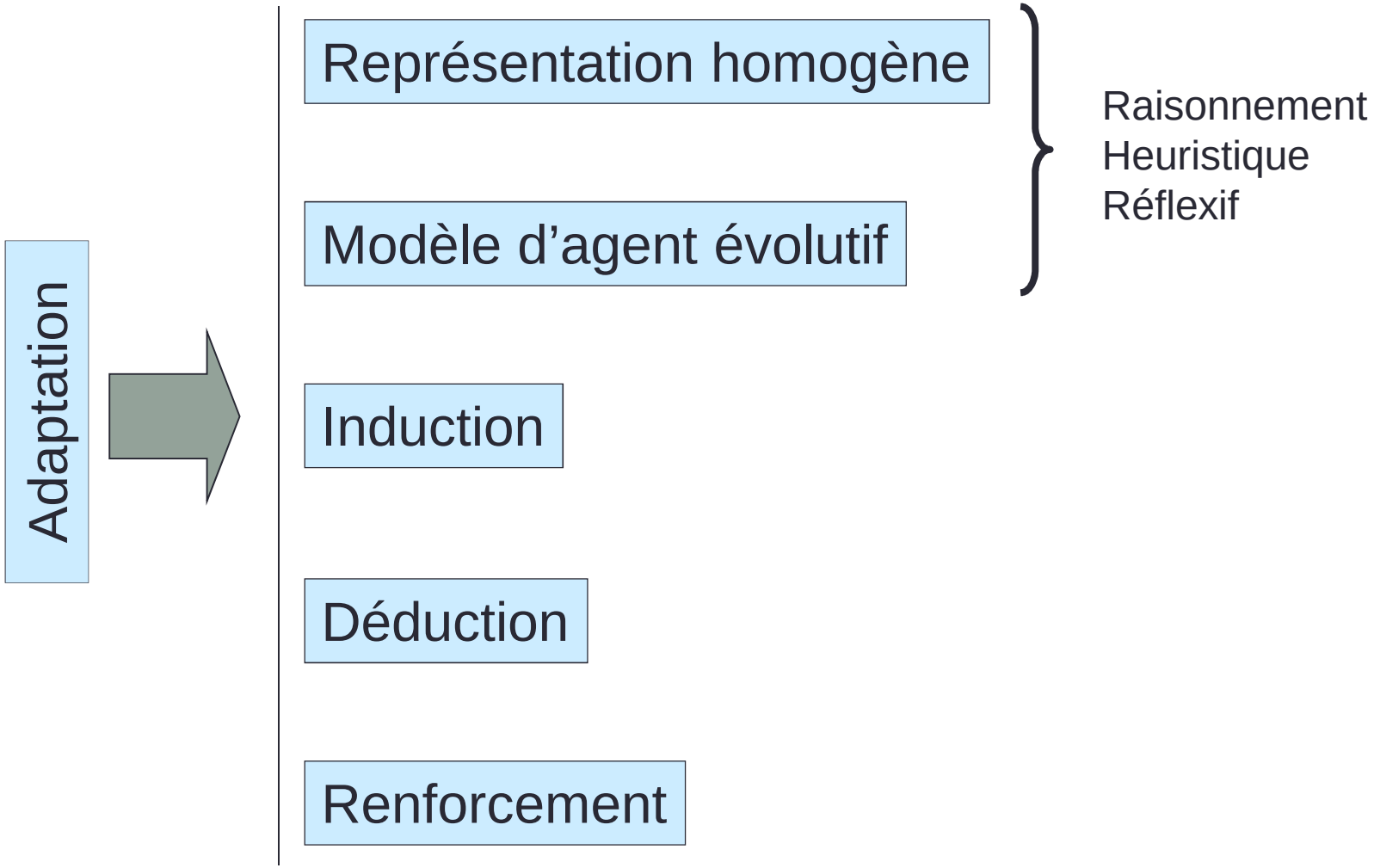
- Adaptabilité à l'environnement
- Environnement complexe, a priori inconnu et relativement stable
- Impossibilité de calculer la meilleur solution (d'où intérêt de l'utilisation d'heuristiques)
- Par exemple:
  - Agent d'aide à la conception
  - Robot
  - Agent négociateur

Sous-objectifs d'apprentissage

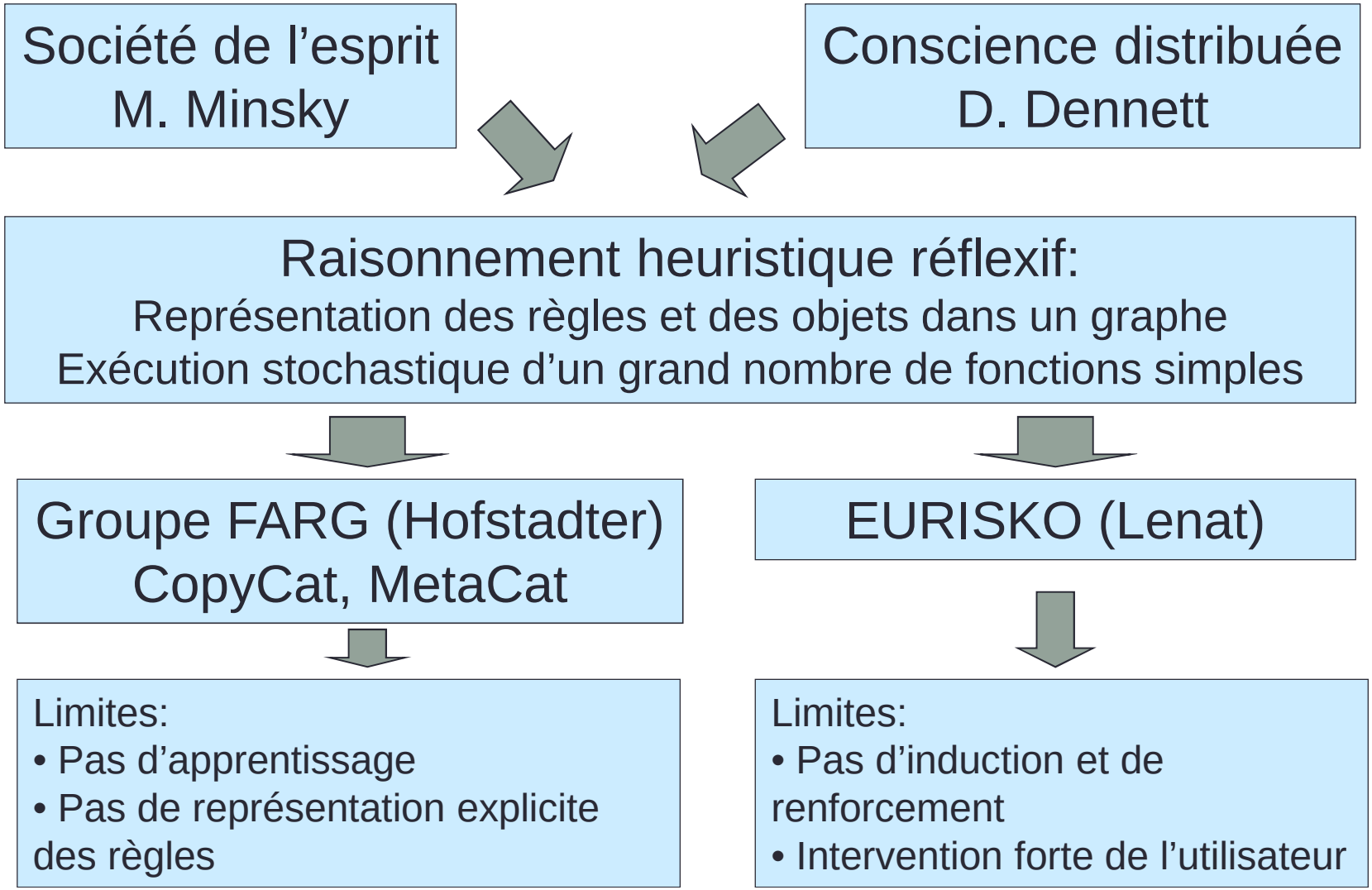


Sous-objectifs fonctionnels

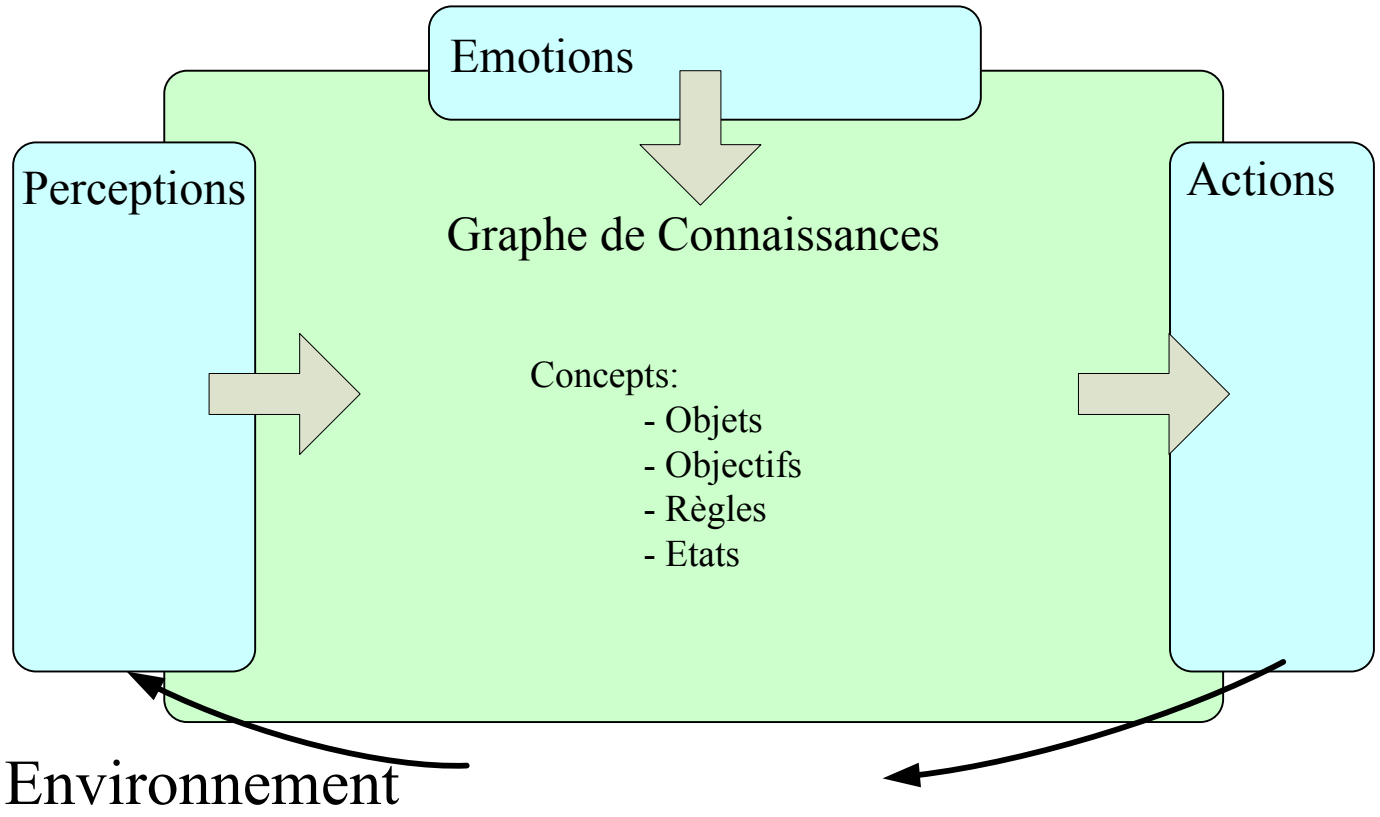



Adaptation
Représentation homogène
Modèle d'agent évolutif
Induction
Déduction
Renforcement
Raisonnement
Heuristique
Réflexif

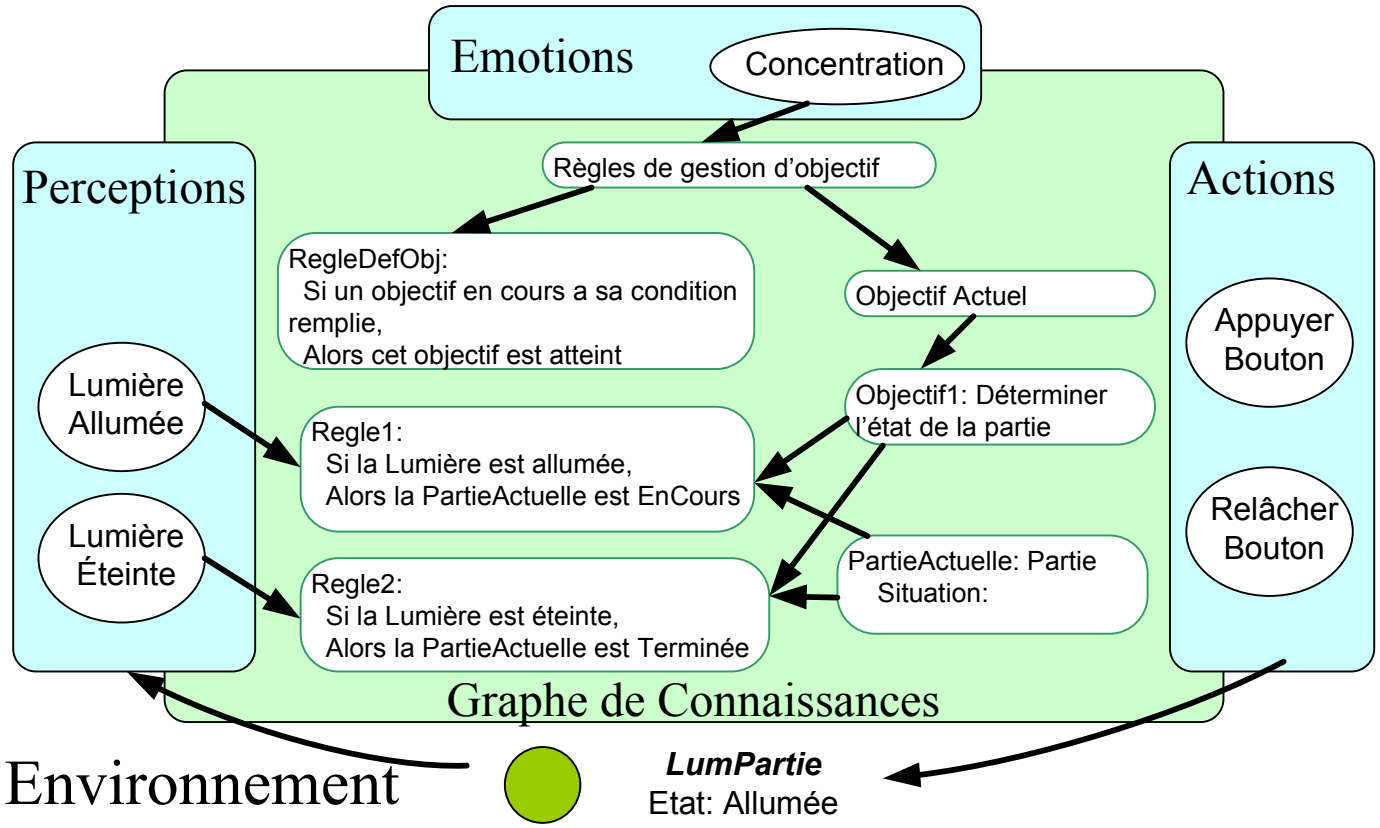
Le raisonnement heuristique réflexif



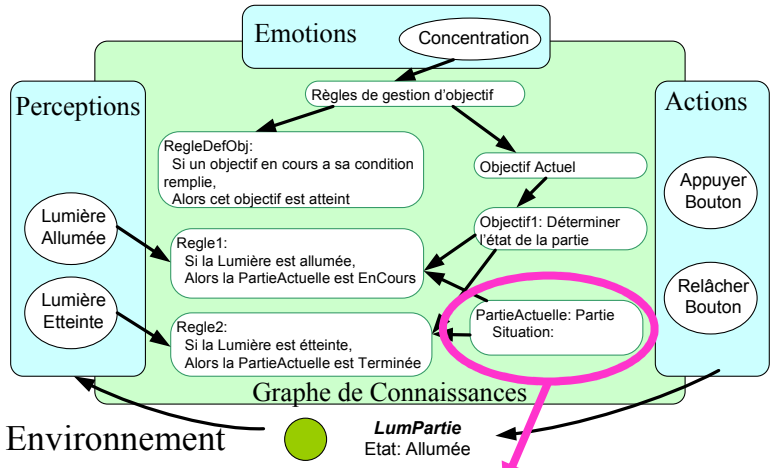
Modèle d'agent

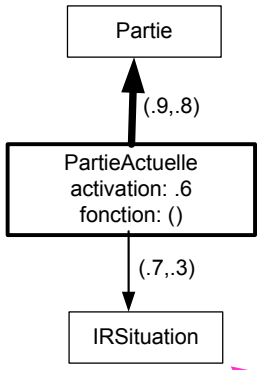


Modèle d'agent

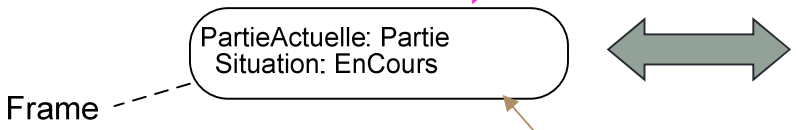


Représentation des connaissances

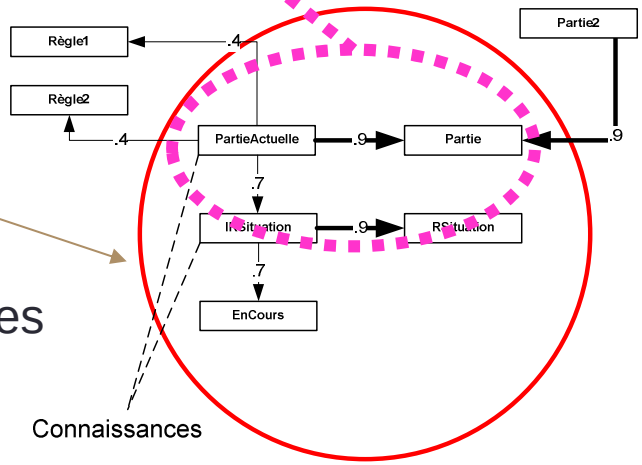




Connaissance: Nœud du graphe
- Degré d'**activation** [0;1]
- Fonction d'interprétations
- **Liens** orientés non typés
  - valeur d'**Intensité** [-1;1]
  - valeur de **stabilité** [0;1]

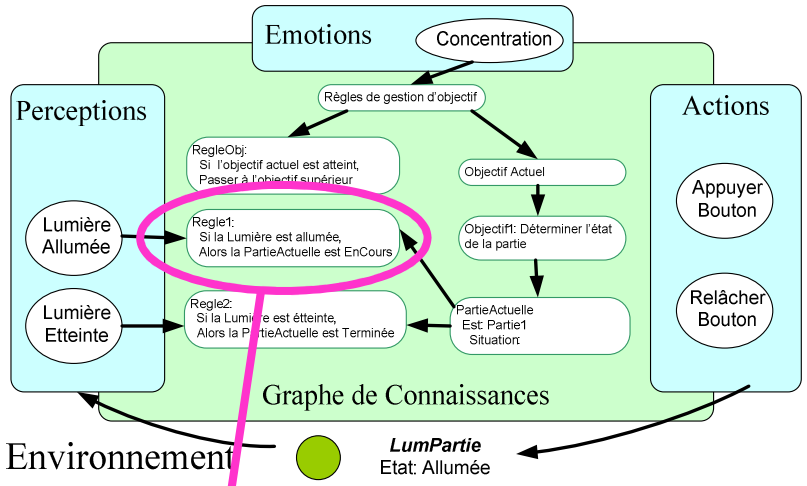


Concept
PartieActuelle

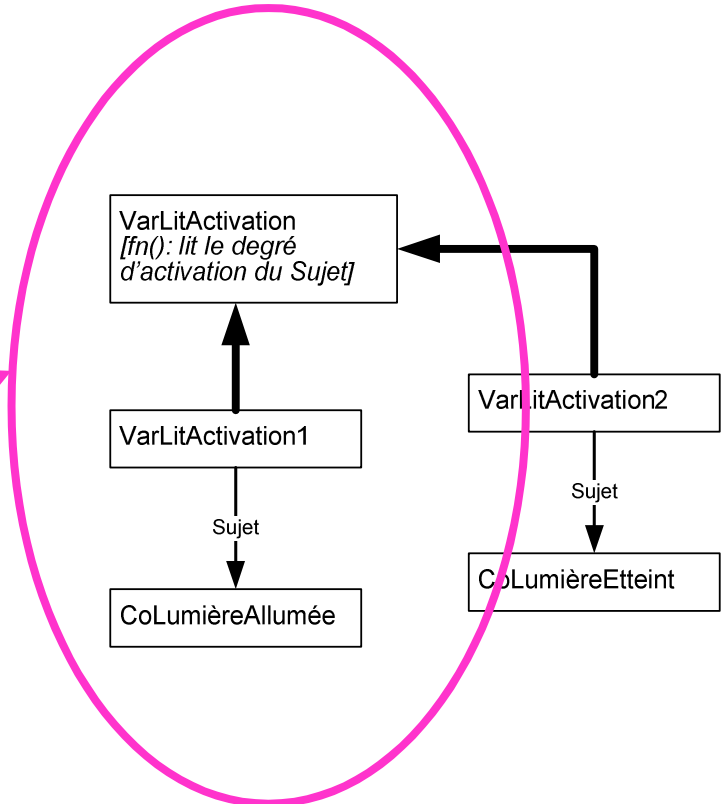


Concept: Ensemble de connaissances activables à partir d'une connaissance racine
Frame: Représentation simplifiée d'un concept sous forme de relations attributs-instanciation

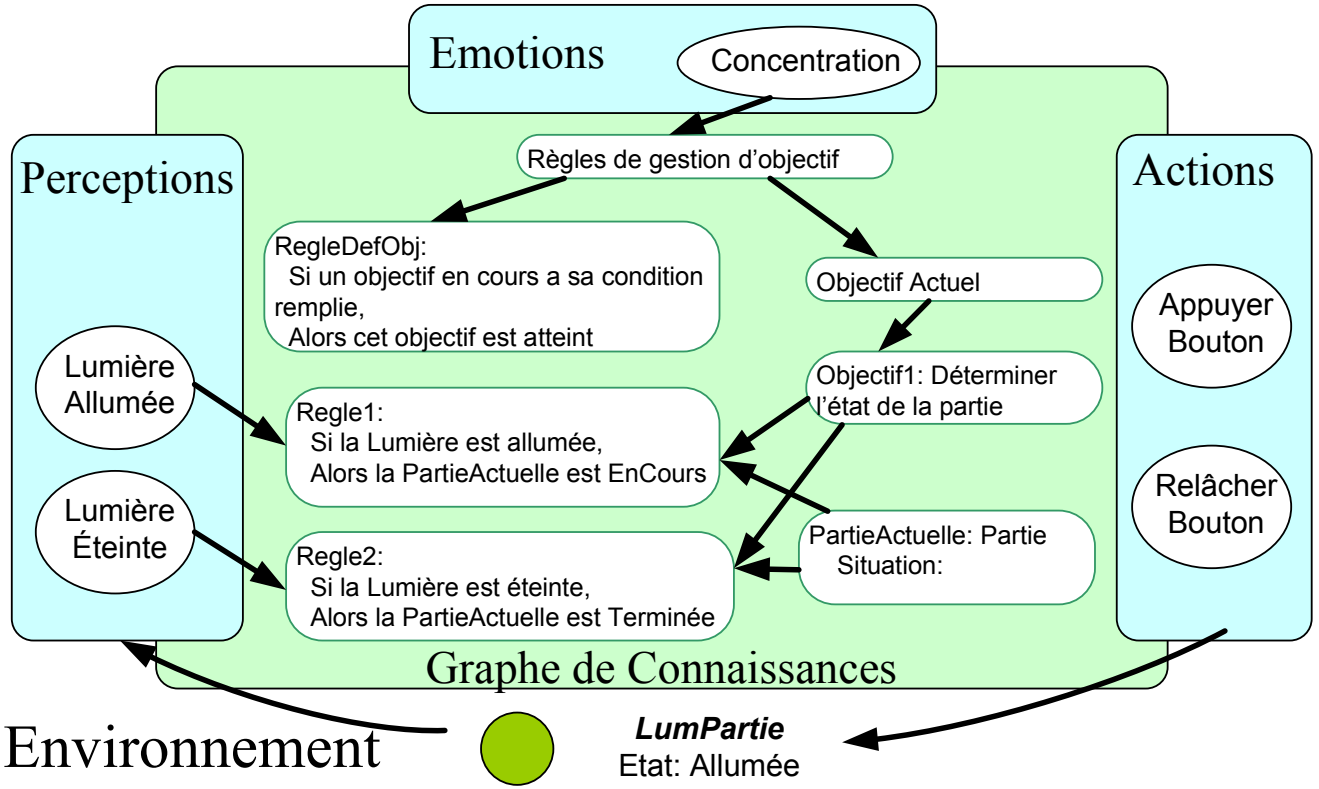
Représentation des règles

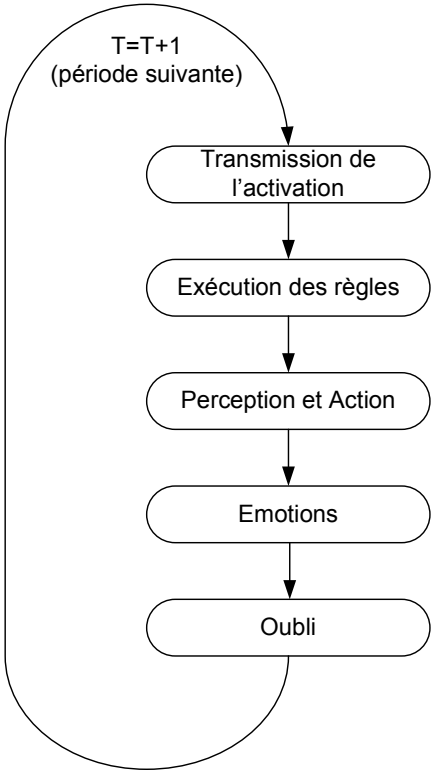



```
Regle1: RegleSiAlors
  Si: Test1: TestSup
    Sujet: VarLitActivation1: VarLitActivation
      Sujet: CoLumièreAllumée
    Quoi: VarLitActivation2: VarLitActivation
      Sujet: CoLumièreEtteinte
  Alors: defParam1: defParam
    Sujet: Lumière
    Param: Situation
    Quoi: Allumée
```



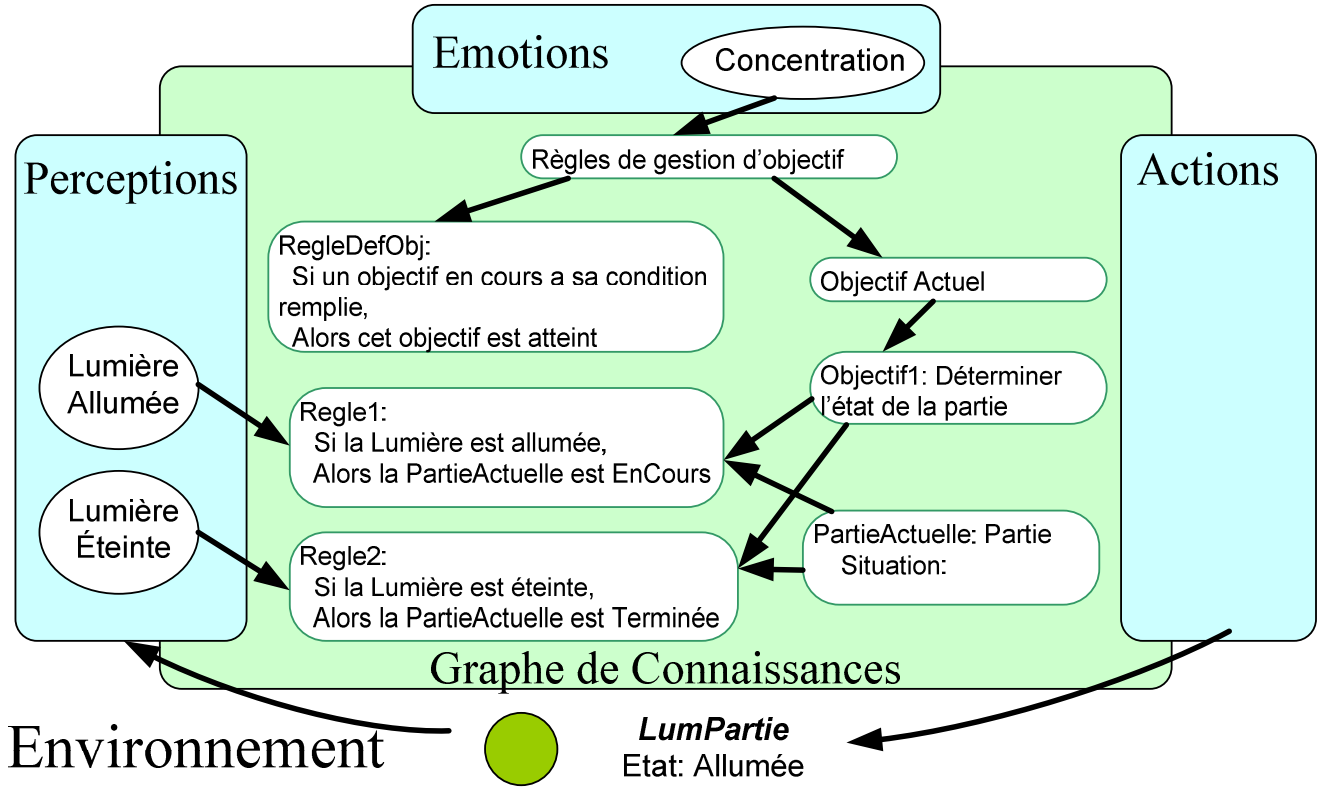
Fonctionnement global



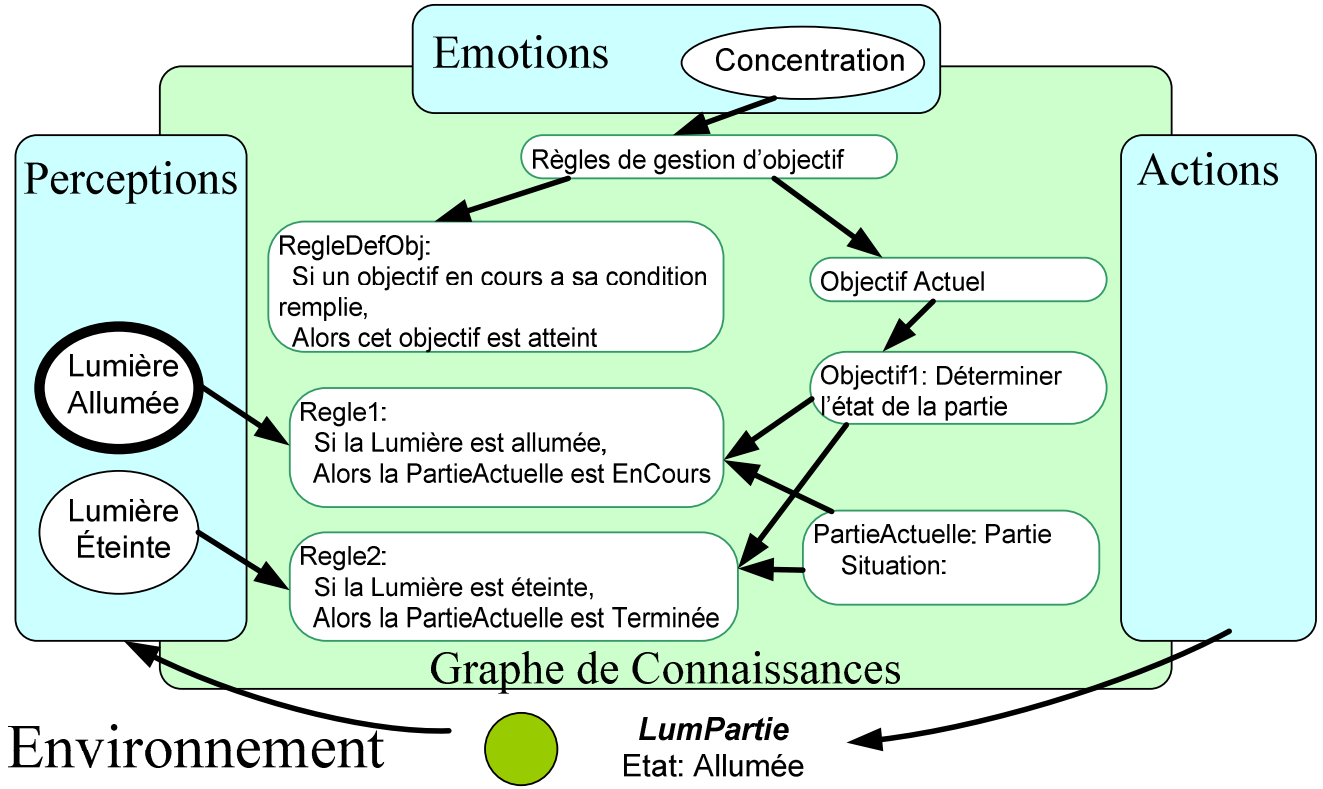


- Exécution successive d'un grand nombre de règles
- Sélection des concepts utilisés (règles pour interprétations, concepts de façon général comme paramètres des règles) de façon probabiliste en fonction du degré d'activation

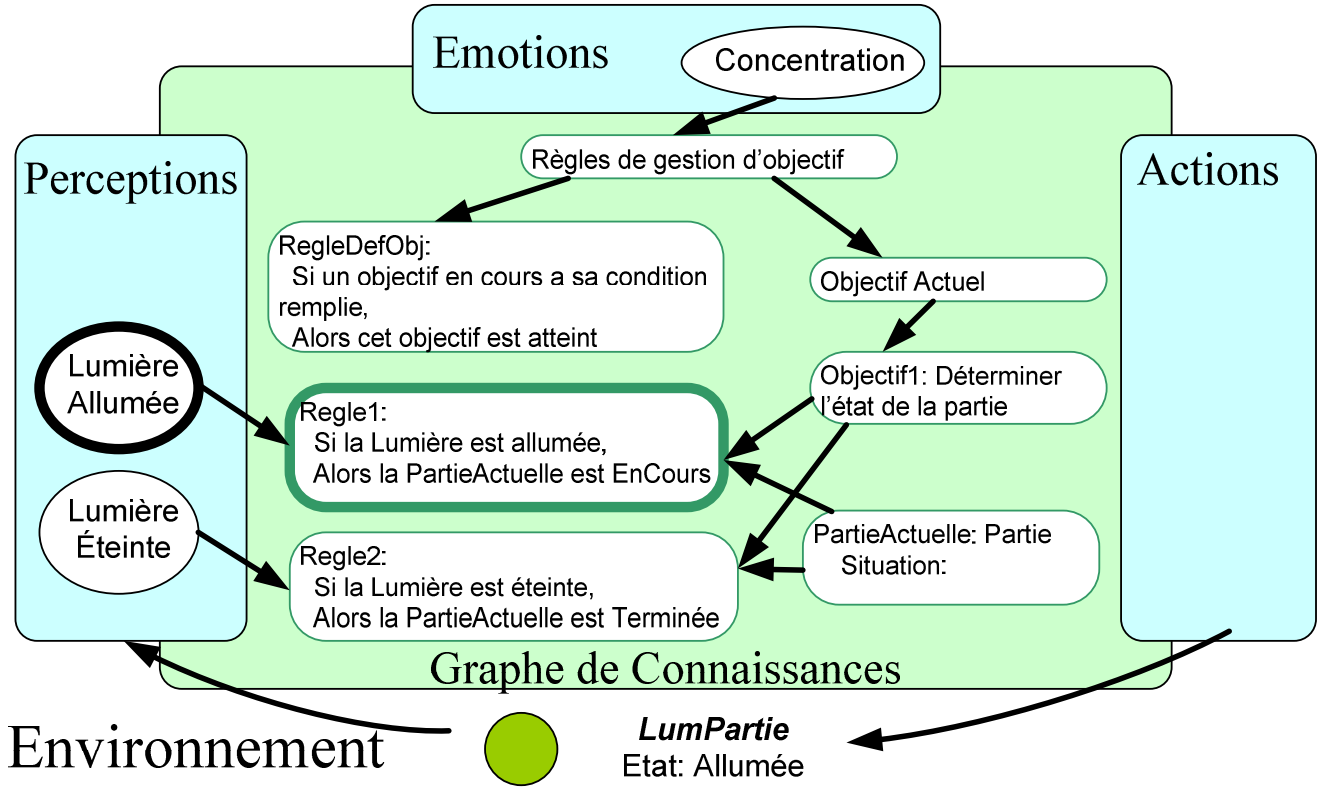
Transmission de l'activation



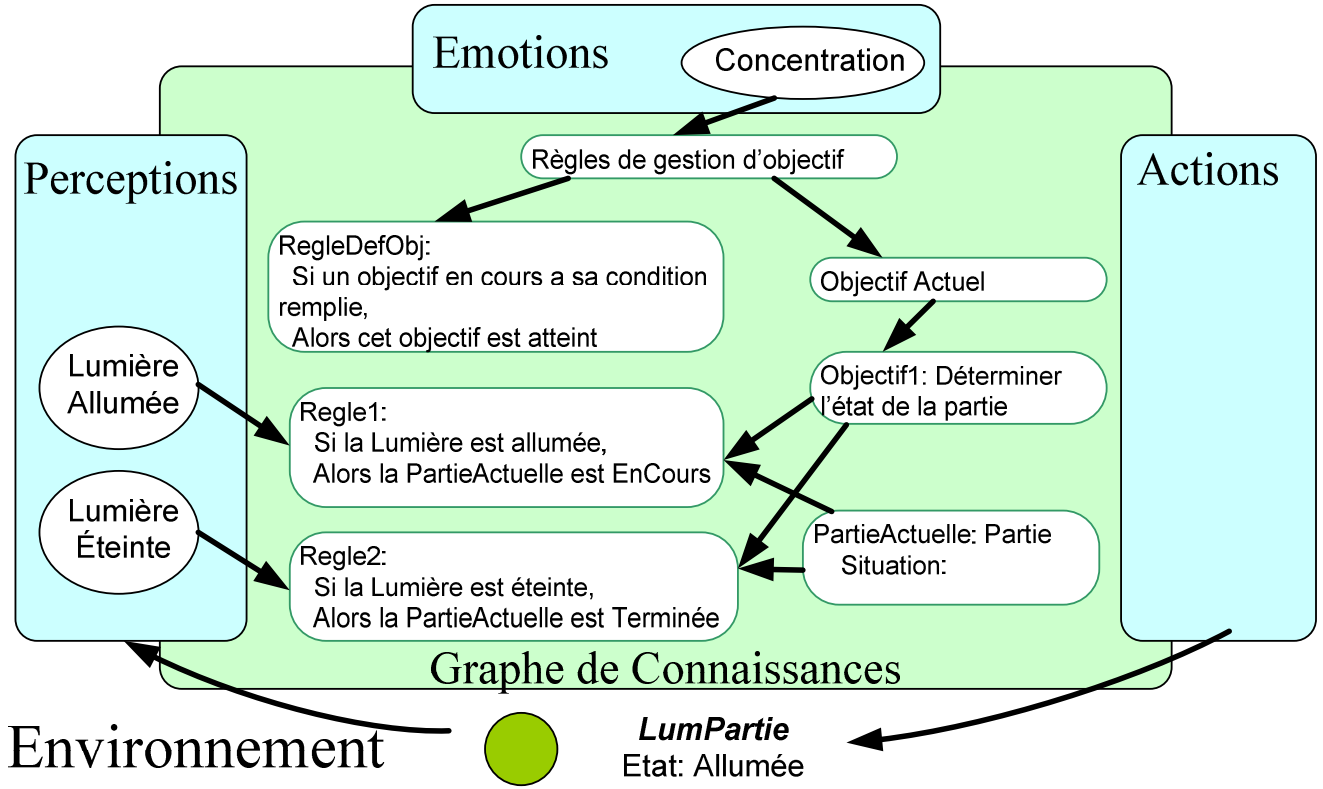
Transmission de l'activation



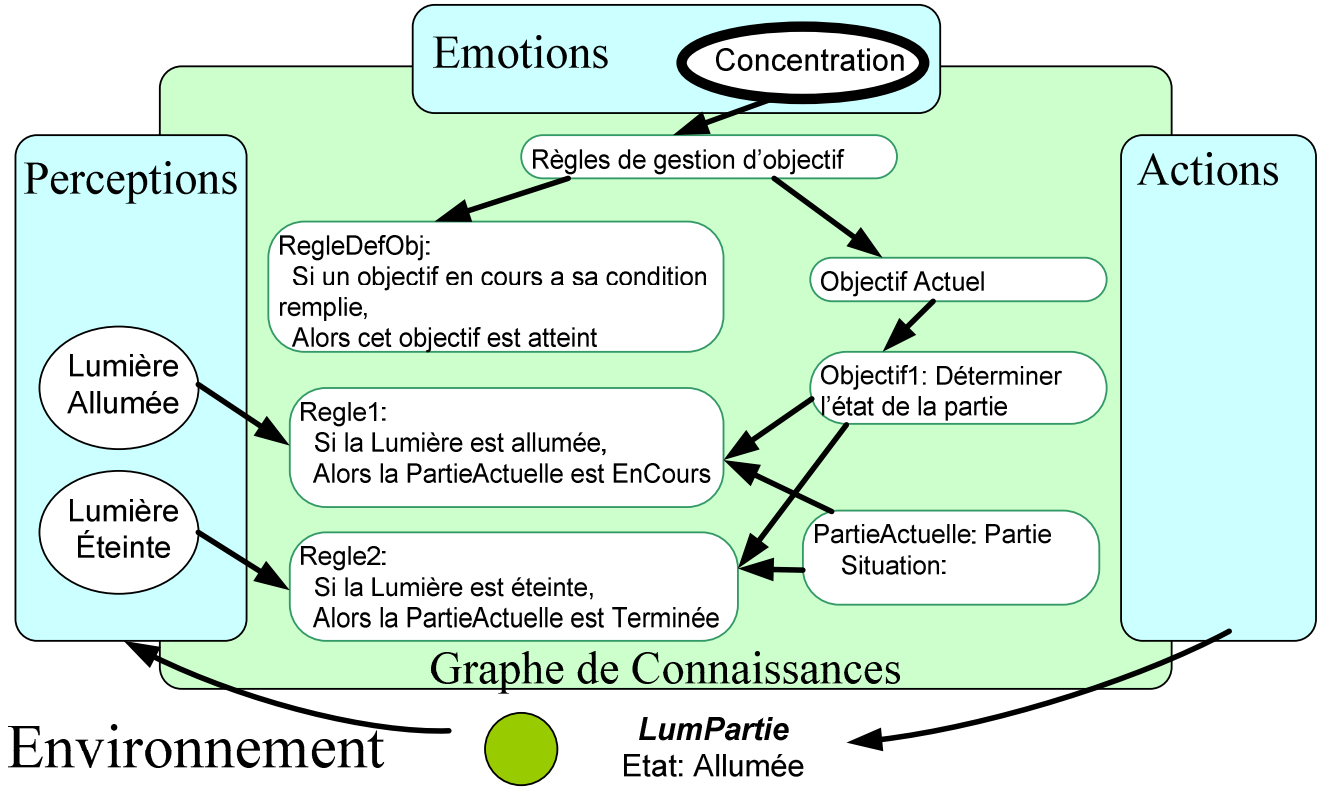
Transmission de l'activation



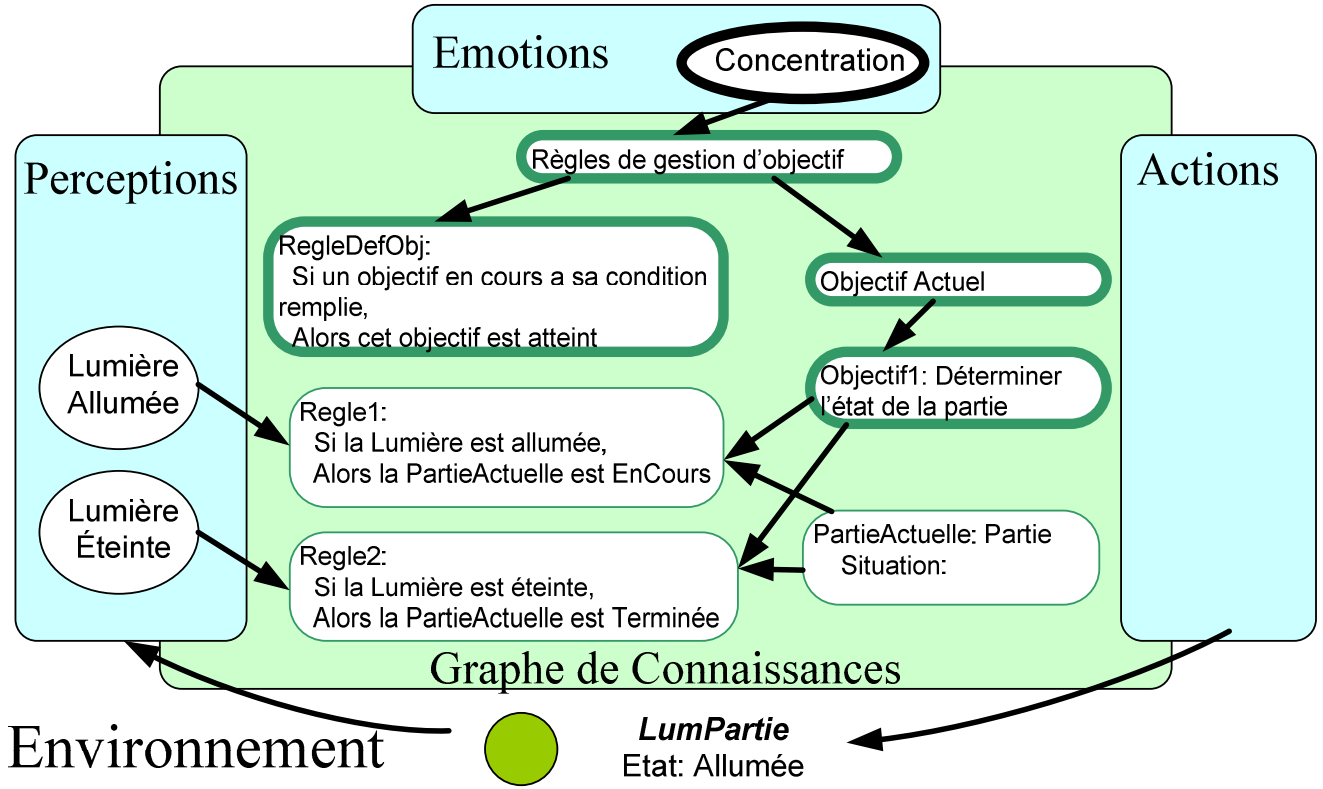
Transmission de l'activation



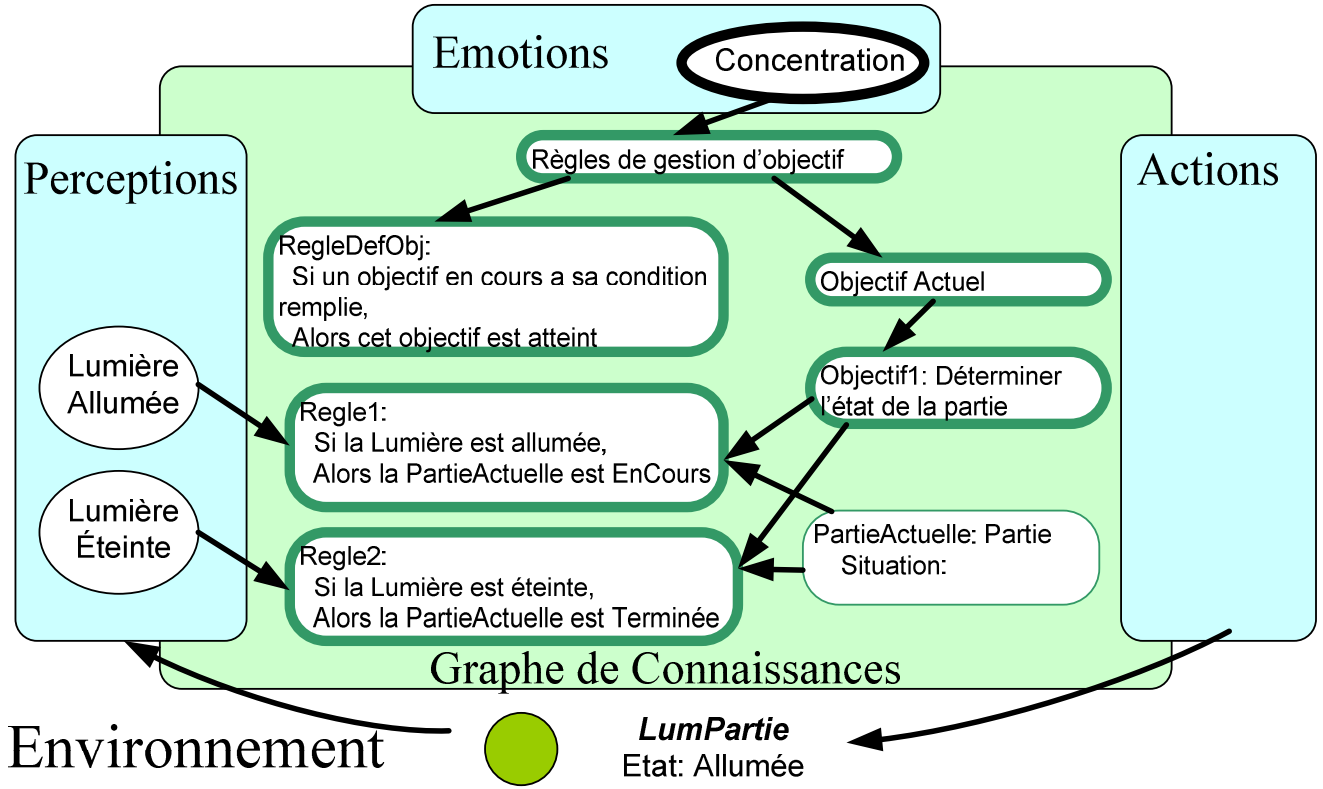
Transmission de l'activation



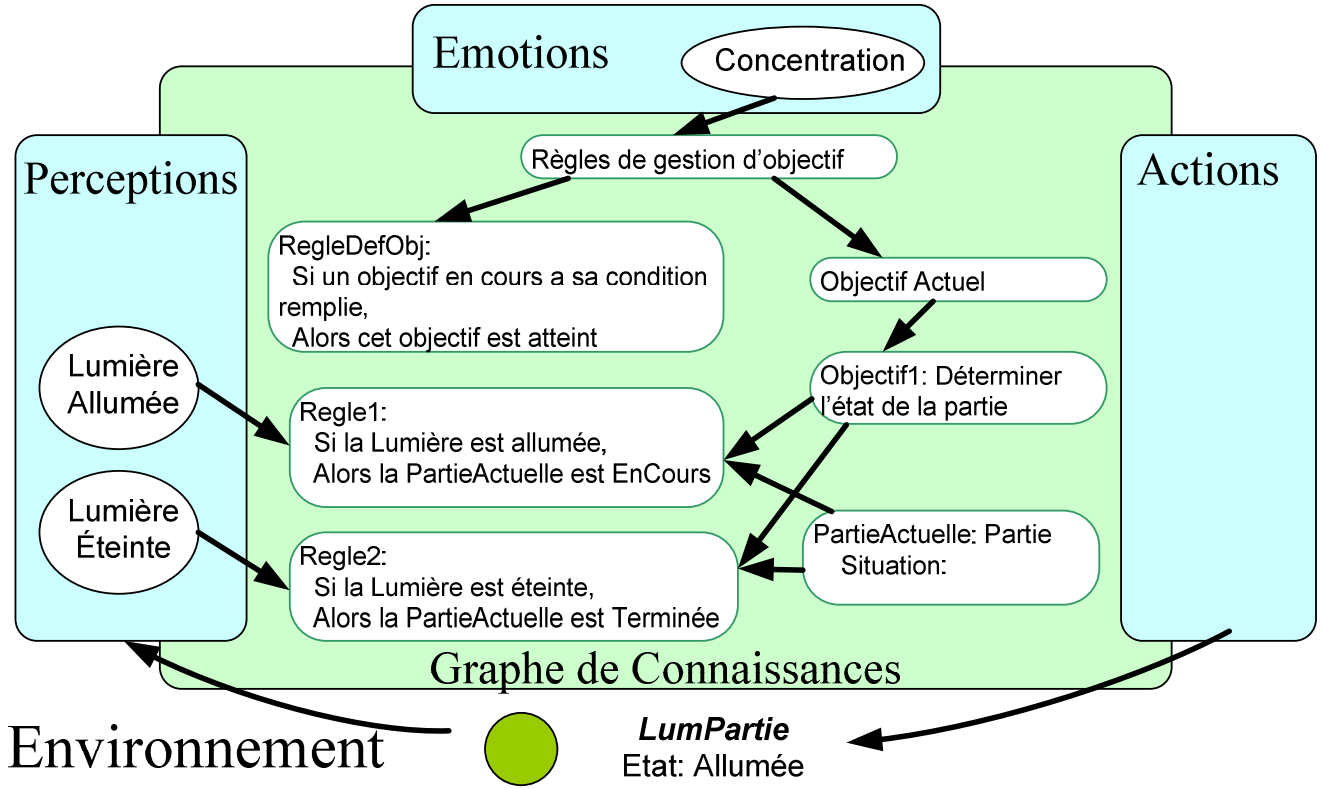
Transmission de l'activation



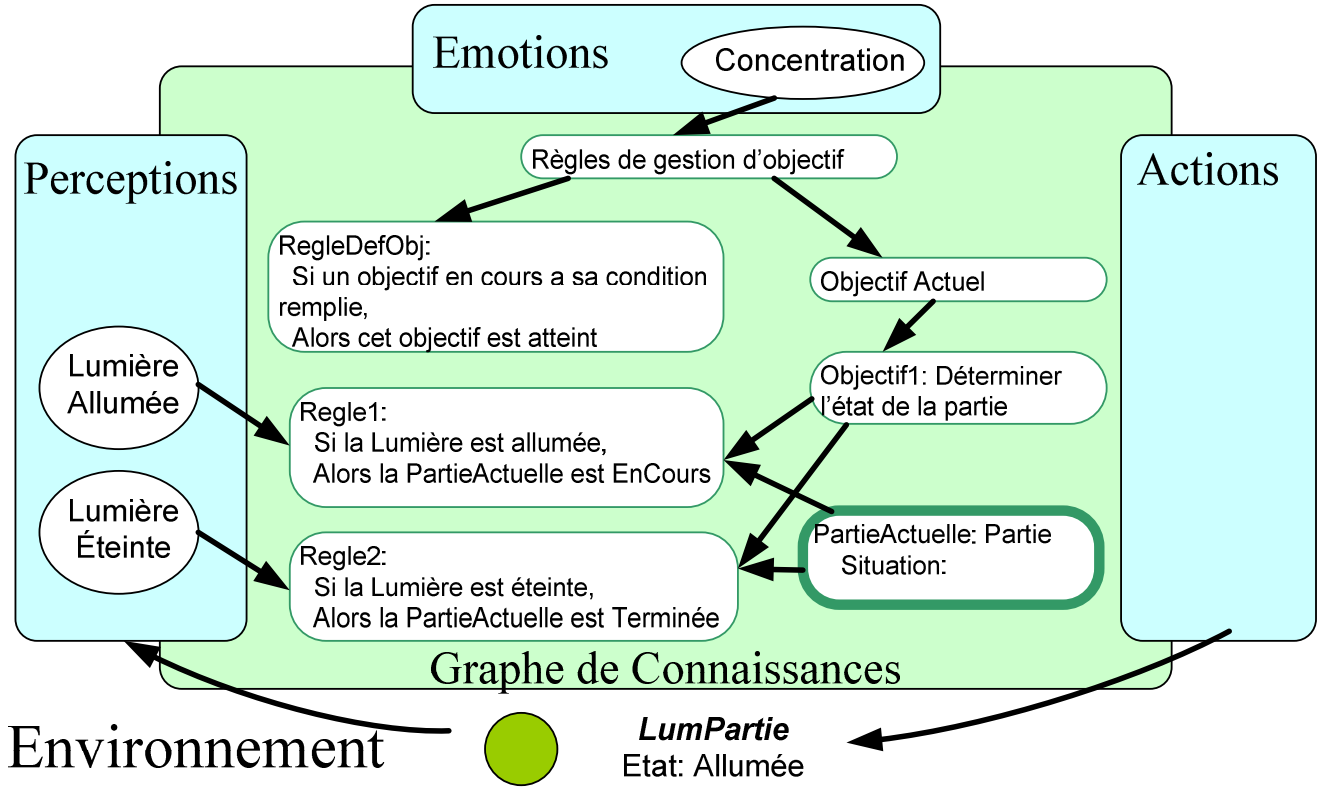
Transmission de l'activation



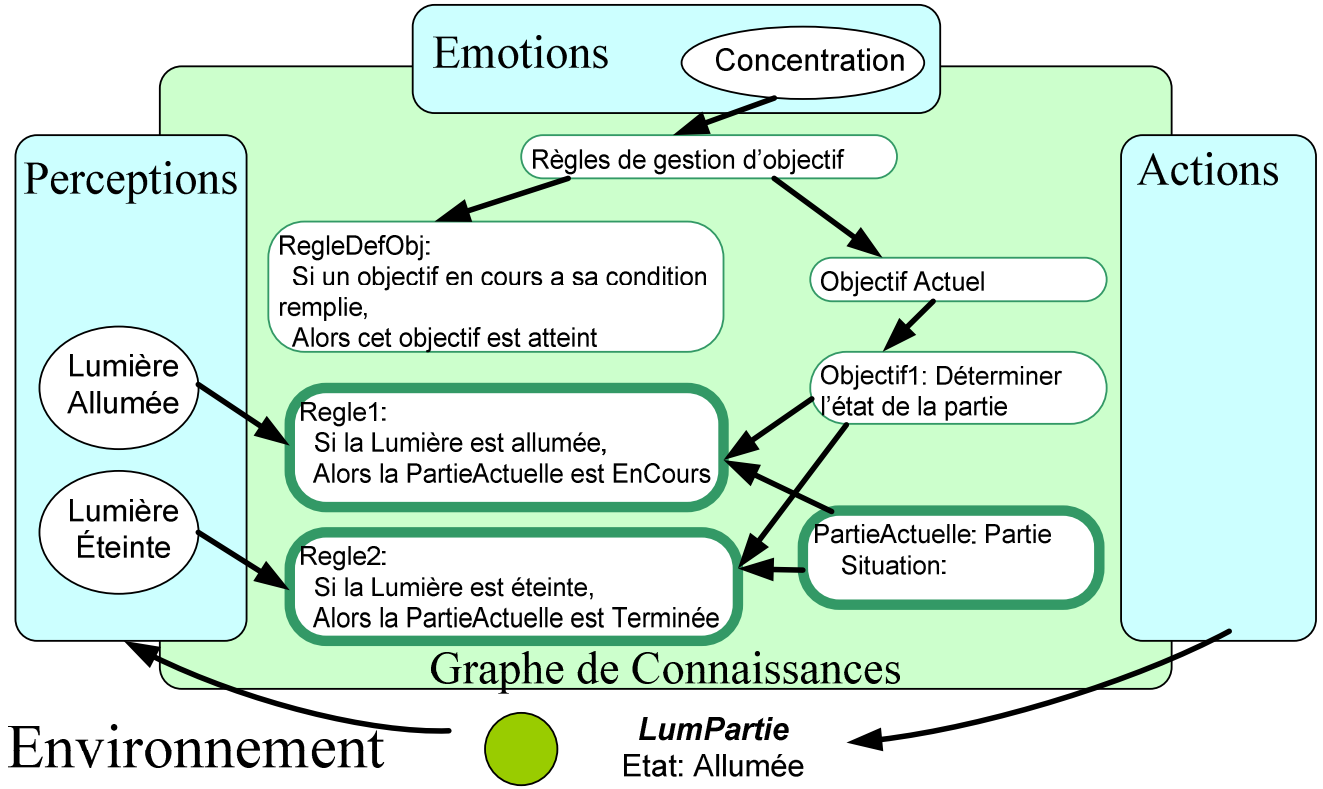
Transmission de l'activation



Transmission de l'activation



Transmission de l'activation



Contrôle de l'agent

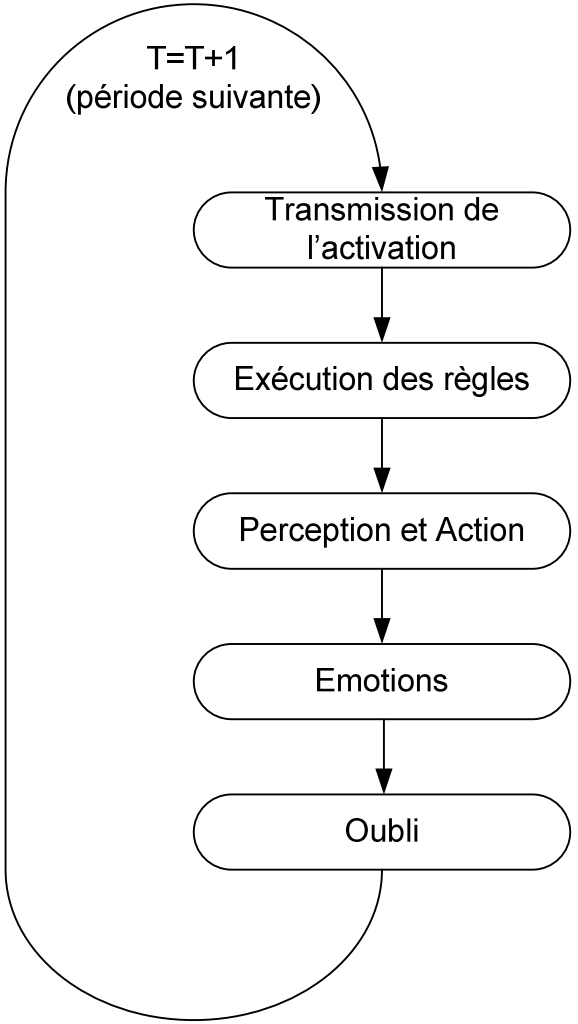
- Contrôle explicite: gestion des buts
  - Court terme
  - Raisonnement par planification
  - Apprentissage par création et déduction
  - Limites:
    - Si l'objectif est fixé au départ, il dépend de la sémantique de départ. Sinon, il doit y avoir intervention de l'utilisateur.
    - Si l'objectif est modifiable par l'agent, risque de modification dans une direction non souhaitée
- Contrôle implicite : émotions
  - Long terme
  - Influence le raisonnement par activation
  - Apprentissage par renforcement des liens

Emotions

- Nécessité d'un contrôle non modifiable par l'agent et le plus indépendant possible de la sémantique.
- Par exemple: température de CopyCat ou gain d'un système par renforcement
- Emotion: Méthode qui influence le raisonnement en fonction de l'état de l'environnement ou de l'état interne de l'agent [Damasio 99]
- Influences:
  - Introduction d'activation dans le graphe
  - Renforcement des liens entre les connaissances dont l'activation a le plus augmenté
  - Perception explicite des fortes variations pour chercher à les reproduire ou à les éviter

Synthèse

- Représentation homogène des concepts
- Fonctionnement homogène à l'aide de la sélection probabiliste des concepts utilisés
- Sélection des concepts grâce à la transmission de l'activation dans le graphe



Apprentissage

- Apprentissage combiné
  - Induction à l'aide de l'analyse de données symboliques
  - Déduction
  - Renforcement et contrôle

Apprentissage combiné