

Programmation orientée émotion

Méthode de résolution semi-automatique de problèmes algorithmiques par modélisation du mécanisme émotionnel humain

Nicolas Sabouret



Kévin Darty



[Plan]

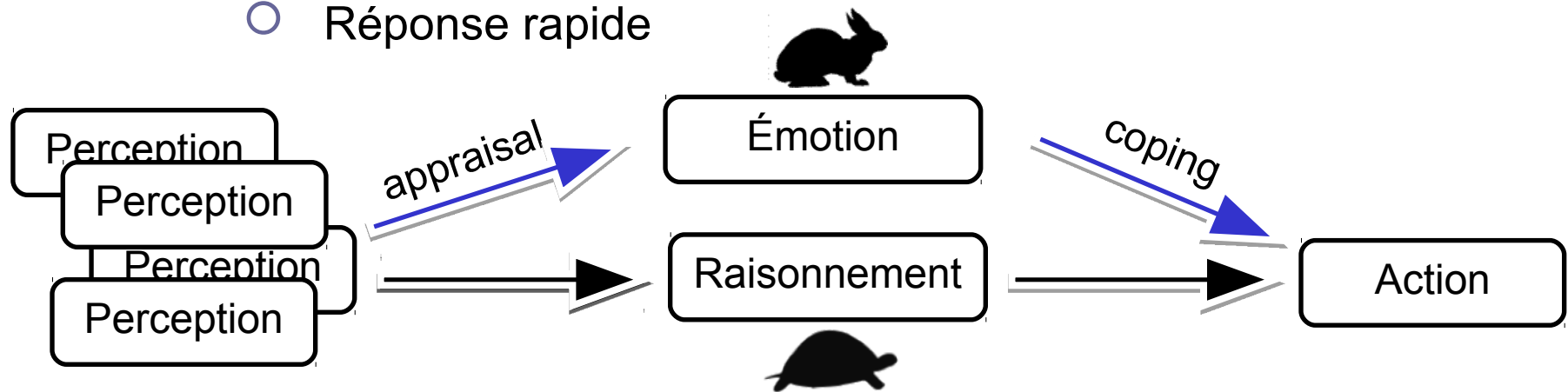


- Contexte
 - Informatique affective
 - Résolution de problèmes en IA
- Objectifs
- Modèle *Programmation Orientée Émotions* (EOP)
 - Architecture
 - Description
- Implémentation
 - Moteur de résolution
 - Exemple du Wumpus
 - Évaluation

Contexte : Informatique affective

■ Émotion

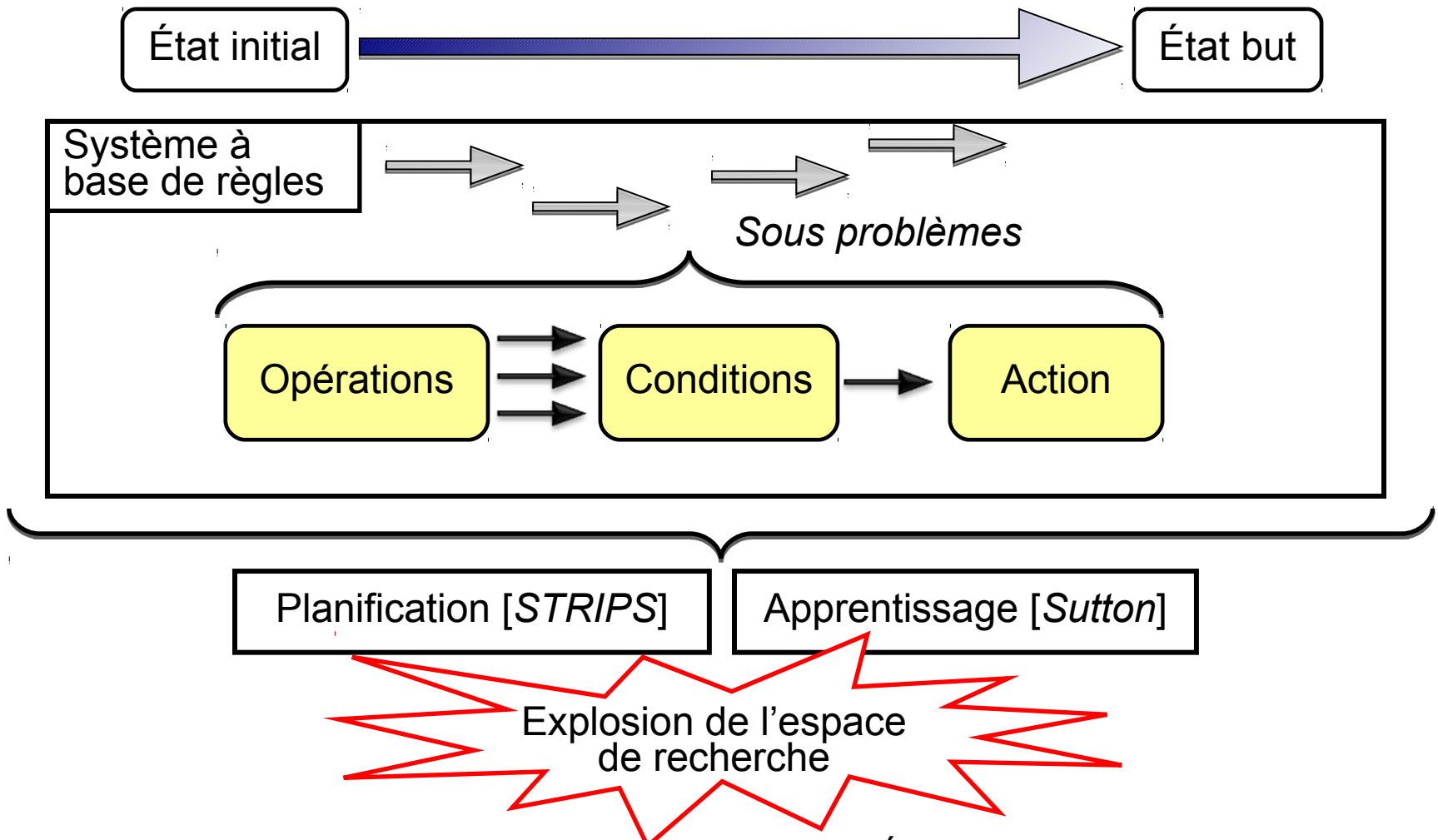
- Mécanisme d'adaptation [*Darwin*]
- Réponse rapide



■ Modèles formels

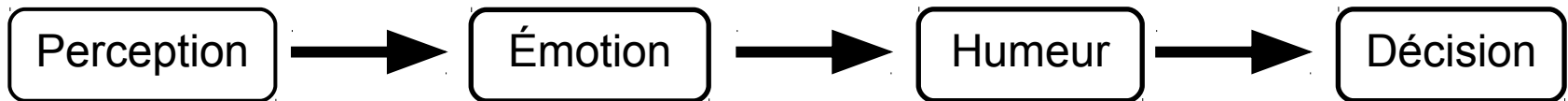
- Dimensionnel : PAD [*Mehrabian*]
- Catégorisation : OCC [*Ortony, Clore & Collins*]
- Évaluation par catégorie et évolution par l'humeur [*Gebhard*]

Résolution de Problème en IA

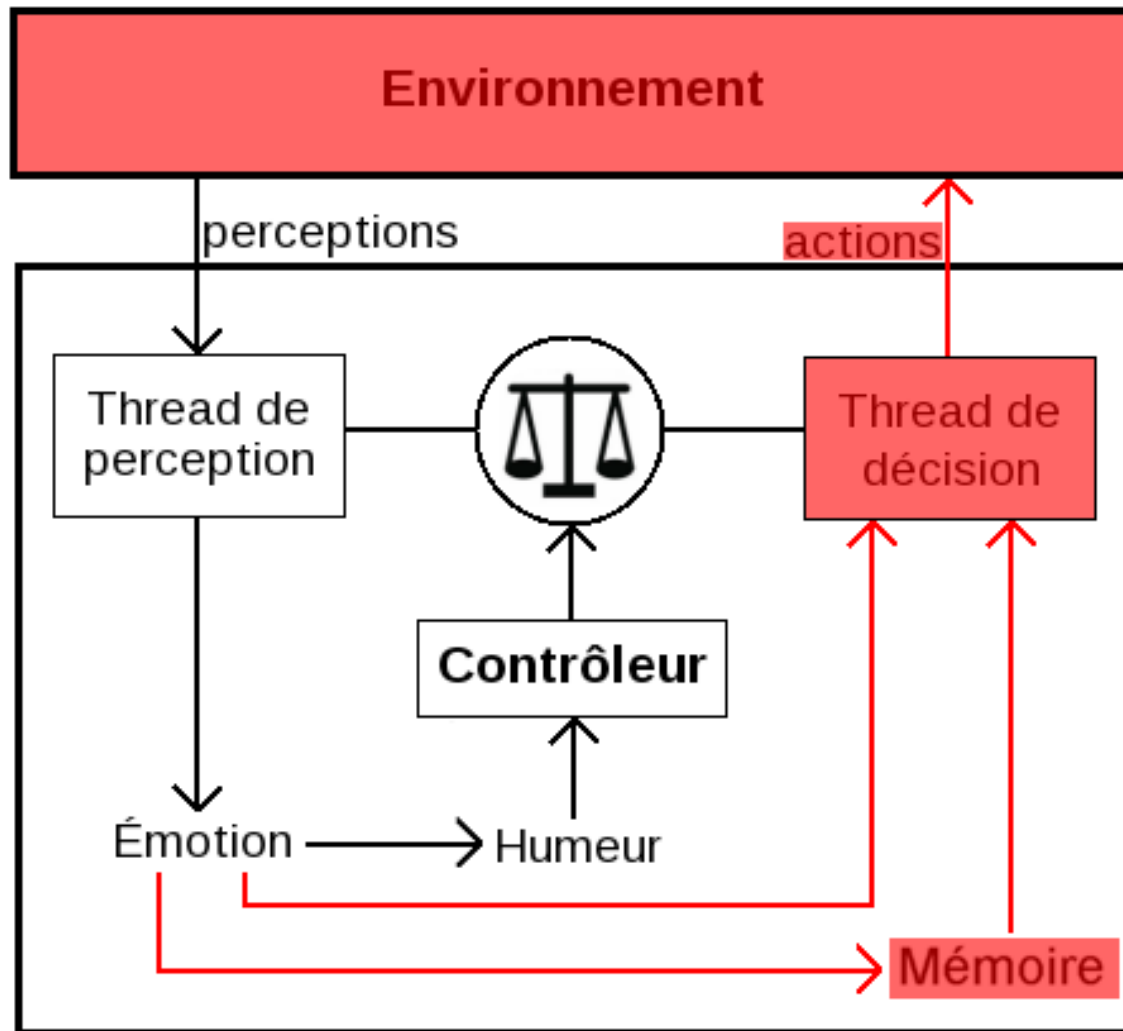


[Objectifs]

- Résolution de problèmes
- Environnement dynamique
- Heuristique : mécanisme émotionnel
(modèle type Lazarus)



Modèle EOP : Architecture



Modèle Émotionnel

Humeur

- PAD

Émotion

- PAD
- Durée de vie
- Sources

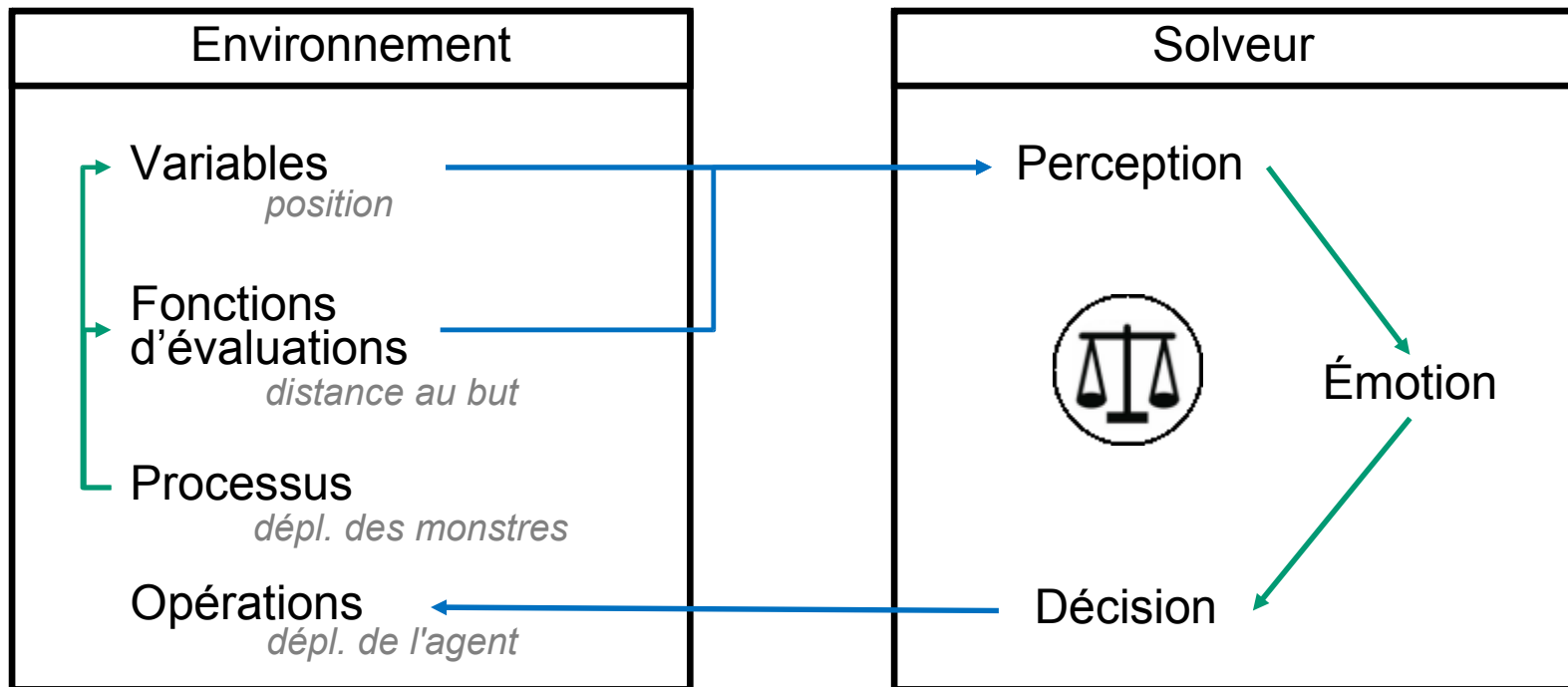
PAD $\vec{PAD}^{(e)} = (P^{(e)}, A^{(e)}, D^{(e)}) \in [-1, 1]^3$

- **Pleasure**
représente le sentiment de proximité du but. Un fort plaisir équivaut à un état "proche" de l'état but.
 $P^{(e)} = f(goal(v) - goal(v'))$
- **Arousal**
exprime le sentiment de nouveauté dans les perceptions. Un faible arousal correspond à un parcours où les états restent similaires.
 $A^{(e)} = h(g(distance(v, v')))$
- **Dominance**
est une indication du sentiment de contrôle. Une forte dominance équivaut à une plus grande capacité d'action sur l'environnement.
 $D^{(e)} = h\left(\frac{card(t \in \mathcal{D}.c_t = \top)}{card(\mathcal{D})}\right)$

$$f(x) = \begin{cases} 1 - e^{-x} & \text{si } x \geq 0 \\ e^x - 1 & \text{sinon} \end{cases} \quad h : [0, 1] \rightarrow [-1, 1] \quad g : \mathbb{R}^+ \rightarrow [0, 1[$$

$$h(x) = 2x - 1 \quad g(x) = 1 - e^{-x}$$

Modèle EOP : Description



Implémentation : Solveur

Perceptions et Décisions

Variables

Mood factor: 2.0
Priority smoothness: 0.5

Mood

Pleasure: 0,004
Arousal: -0,11
Dominance: 0,123

All threads

- Under fear decision 1.0
- Under Arousal decision 1.0
- ListenToDistanceDecision : 1.0
- Patch memory decision 5.0
- trap fear perception 20.0
- Treasure perception 10.0
- Distance trustfully perception 10.0
- Monster stink perception 20.0

Valid threads

- Under fear decision 1.0
- trap fear perception 20.0

Current thread

trap fear perception 20.0

Emotional memory

Emotional state : (-0.5, 0.0, 0.0), trap fear perce|
Emotional state : (-0.5, 0.0, 0.0), trap fear perce|

Current emotions

Emotional state : (-0.5, 0.0, 0.0), trap fear perce|

Speed

Play Step

0 10 20 30 40 50 60 70 80 90 100 99

Tic 813

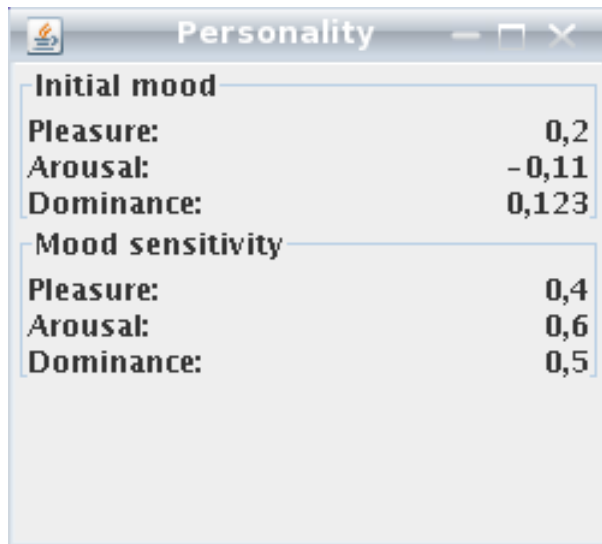
State Solvable Unsolved

Humeur

Mémoire

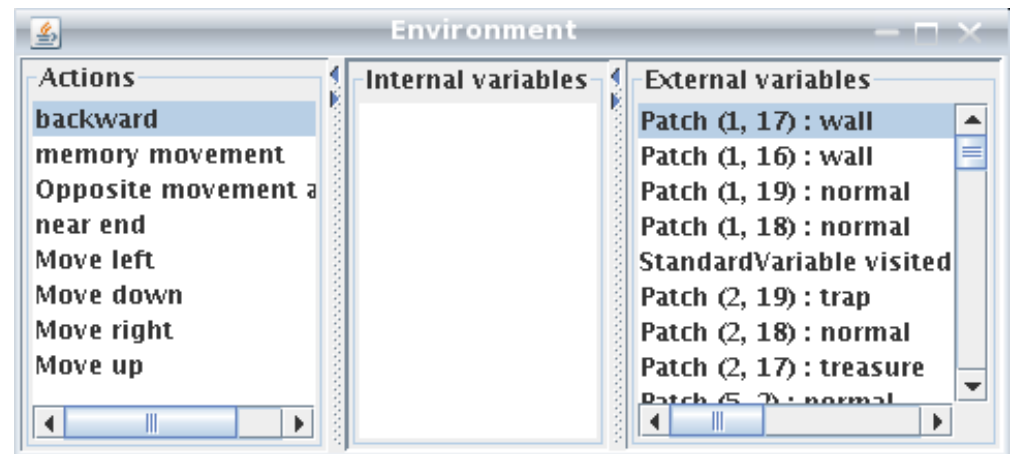
Implémentation : Solveur

Personnalité



Initial mood	
Pleasure:	0,2
Arousal:	-0,11
Dominance:	0,123
Mood sensitivity	
Pleasure:	0,4
Arousal:	0,6
Dominance:	0,5

Environnement



Actions	Internal variables	External variables
backward		Patch (1, 17) : wall
memory movement		Patch (1, 16) : wall
Opposite movement a		Patch (1, 19) : normal
near end		Patch (1, 18) : normal
Move left		StandardVariable visited
Move down		Patch (2, 19) : trap
Move right		Patch (2, 18) : normal
Move up		Patch (2, 17) : treasure
		Patch (5, 2) : normal

Implémentation

- Exemple du Wumpus



Départ



Mur



Wumpus



Arrivée



Trésor



Piège



Agent



[Implémentation]

- Évaluation
 - Comparaison avec le parcours en largeur Floyd
 - Répartition des cases :
 - 5% de pièges
 - 5% de trésors
 - 40% de murs
- 27% d'atteinte de la case d'arrivée
- 23% de trésors récoltés

[Conclusions]



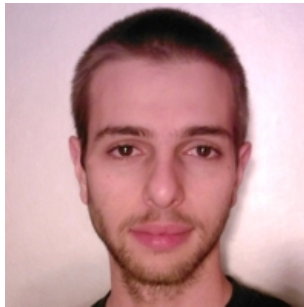
- Architecture de résolution de problème
 - Boucle perception/action
 - Evaluation cognitive d'émotions et humeur
 - Adaptation du comportement (perceptions et actions) aux émotions et à l'humeur
- Propriétés de résolution de problème
 - Adaptation en environnement dynamique (Souplesse, Réactivité)
 - Programmation facilitée et plus modulaire
- Première implémentation et évaluation

[Perspectives]



- Définition précise de la classe de problème
 - Environnement dynamicité ?
 - Environnement inconnu ?
 - Temps contraint ?
 - Un peu tout ça ?
- Réduire la partie laissée au programmeur
 - Définition des fonctions de distances
 - Gestion automatique des priorités
 - (Heuristiques ? Méthodes d'apprentissage ?)
- Vers un nouveau paradigme de programmation ?

[Merci !]



Et aussi :

- Cindy Mason
- Guillaume Carré