



Analyse et implémentation d'un modèle de conscience artificielle

Travail d'Étude et de Recherche

Master Informatique 1ère année (GMIN20B)

William Dyce Thibaut Marmin
Namrata Patel Clément Sipieter

Université Montpellier 2
Encadré par Violaine Prince et Guillaume Tisserant

7 mai 2012



Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Analyse Générale

Analyse & Implémentation

Outils de travail

Conclusion & Perspectives



Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Rappel historique

Projet de recherche

Un modèle de conscience
artificielle

Outils de travail

Conclusion & Perspectives

Analyse Générale

Analyse & Implémentation



Rappel historique

Deep Blue

Deep Blue

- Programme d'échecs développé par IBM.
- Victoire contre Garry Kasparov en 1997.
- Premier défaite d'un grand maître sous contraintes normales de temps.



Rappel historique

Deep Blue

Deep Blue

- Programme d'échecs développé par IBM.
- Victoire contre Garry Kasparov en 1997.
- Premier défaite d'un grand maître sous contraintes normales de temps.

Robert Levinson

- « *But doesn't know that it's playing chess.* »
- Est-ce donc vraiment de l'intelligence ?



Rappel historique

Théorie des Jeux

Théorème du Minimax

- Élaboré par John Von Neumann en 1928.
- Stratégie optimale pour jeux compétitives tels les échecs.

Rappel historique

Théorie des Jeux

Théorème du Minimax

- Élaboré par John Von Neumann en 1928.
- Stratégie optimale pour jeux compétitives tels les échecs.

Équilibre de Nash

- Définit par John Forbes Nash en 1950.
- Fondation de la Théorie des Jeux.

Rappel historique

Théorie des Jeux

Théorème du Minimax

- Élaboré par John Von Neumann en 1928.
- Stratégie optimale pour jeux compétitives tels les échecs.

Équilibre de Nash

- Définit par John Forbes Nash en 1950.
- Fondation de la Théorie des Jeux.

Élagage Alpha-beta

- Conçu par John McCarthy en 1958.
- Amélioration du Minimax.



Rappel historique

Limites

Domaine d'application

- Jeux compétitifs à somme nulle.
- Durée et nombre d'options finis.



Rappel historique

Limites

Domaine d'application

- Jeux compétitifs à somme nulle.
- Durée et nombre d'options finis.

Temps de calcul

- Complexité moyenne $O(b^{\frac{d}{2}})$ avec élagage.
 - d : profondeur de l'arbre de décision.
 - b : facteur de branchement.
- Utilisation d'heuristiques, donc perte d'optimalité.



Projet de recherche

Une approche alternative

- Apprentissage
- Reconnaissance de formes (concepts) ?
- Classification
- polyvalence du système ???



Un modèle de conscience artificielle

Origine du modèle

- « Conscience artificielle » par Guillaume Tisserant, Guillaume Maurin, Ndongo Wade et Anthony Willemot (Projet du module 'Cognition' en M2, 2010)
- Modèle de représentation de la Conscience proposé dans le Chapitre 4.

Version simplifiée du modèle

Le diagramme illustre l'architecture cognitive humaine, divisée en trois niveaux de conscience et cinq zones de traitement de l'information.

Niveaux de conscience (à gauche) :

- conscient** (jaune)
- préconscient** (jaune pâle)
- inconscient** (jaune très pâle)

Zones de traitement de l'information (à droite) :

- néocortex** (vert clair)
- cerveau de la mémoire** (vert moyen)
- cerveau reptilien** (vert foncé)

Processus et Mémoires (à l'intérieur) :

- Mémoire mnésique** (rouge) → **Mémoire épisodique** (rouge) → **Mémoire sémantique** (rouge) → **Mémoire associative SRP** (rouge) → **Mémoire procédurale** (rouge) → **Mémoire vive ci (état courant)** (orange) → **Base algorithmique** (bleu) → **Somme** (bleu) → **Sortie** (noir)
- Mémoire primaire ci** (rouge) → **Optimisateur** (bleu) → **Somme** (bleu)
- Choix conscient** (bleu) → **Choix d'action** (bleu) → **Somme** (bleu)
- Concepts primaires** (violet) → **Analysateur conceptuel** (vert) → **Analysateur conceptuel poussé** (vert) → **Mémoire sémantique** (rouge)
- Entrée** (noir) → **Analysateur conceptuel** (vert) → **Mémoire procédurale** (rouge)

Légende :

- Selon Freud (jaune)
- Selon Laborit (vert)
- Signaux de bas niveau (noir)
- Mémoire vive (orange)
- Base de concepts (violet)
- Concepts (vert)
- Concepts avancés (vert foncé)
- Choix d'action (bleu)
- Odronnement des choix (bleu foncé)
- Mémoires (rouge)
- Mémoire (rouge foncé)



Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Analyse Générale

Analyse générale

Restrictions appliquées au
modèle

Vision globale du modèle
opérationnel

Outils de travail

Conclusion & Perspectives

Analyse & Implémentation

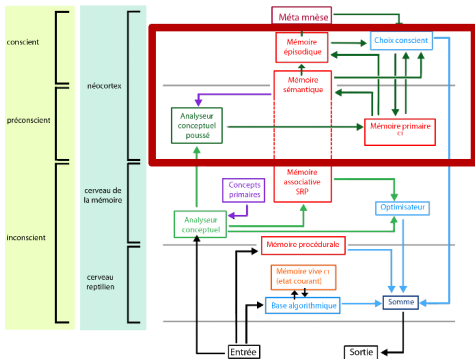
Contraintes de réalisation

Temporelle , d'effectifs et de compétences

- Temps : travail à réaliser en trois mois
- Effectif : quatre membres dans l'équipe
- Compétences :
 - Compétences requises à la **fin** du semestre
= Compétences nécessaires pour la réalisation du projet
 - Cours données souvent trop tard pour assurer leur bonne application au projet

Restrictions appliquées au modèle

Environnement , fonctionnement

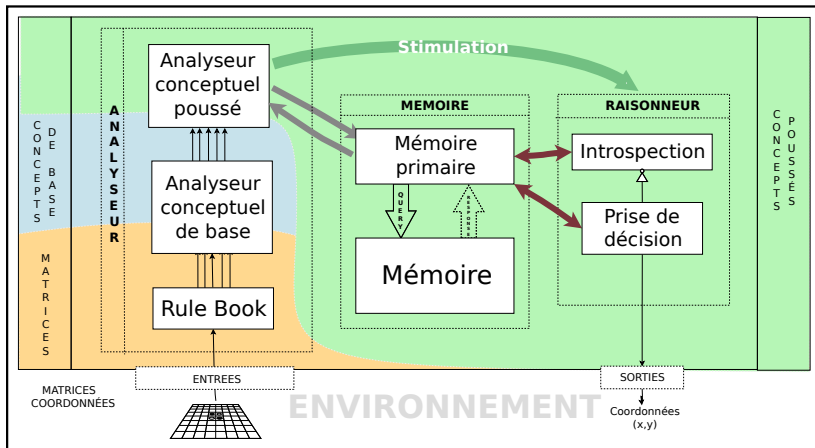


- **Environnement :**
Limitation à un type précis de jeu
- **Fonctionnement :**
Retrait et simulation des parties
 - Retrait de la métamnése
 - Simulation de la partie inconsciente



Vision globale du modèle opérationnel

Séquence





Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Outils de travail

Analyse Générale

Conclusion & Perspectives

Analyse & Implémentation

Environnement &
simulation

Analyseur conceptuel

Raisonneur

Mémoire

Conclusion partielle



Environnement & Simulation

Webservice `game_service`

Environnement "arbitre" du jeu (séparation des agents de l'environnement)

Representationnal State Transfer (REST)

- Technologie Java Servlet.
 - Requetes par HTTP.
 - Réponses en XML.
- Technologie HTML 5.
 - Asynchronous Javascript & XML (AJAX).
 - Bibliothèque jQuery.

Environnement & Simulation

Bibliothèque `game_logic`

Classe `BoardMatrix`

Plateau sous forme matricielle avec accesseurs adaptés.



Environnement & Simulation

Bibliothèque `game_logic`

Classe BoardMatrix

Plateau sous forme matricielle avec accesseurs adaptés.

Classe abstraite Rules

- Forme du plateau ? Configuration initiale ?
- Qui joue en premier ?
- Quand a-t-on gagné ? Perdu ? Un match nul ?
- Quelles sont les coups possibles ?



Environnement & Simulation

Bibliothèque `game_logic`

Classe BoardMatrix

Plateau sous forme matricielle avec accesseurs adaptés.

Classe abstraite Rules

- Forme du plateau ? Configuration initiale ?
- Qui joue en premier ?
- Quand a-t-on gagné ? Perdu ? Un match nul ?
- Quelles sont les coups possibles ?

Classe Game

Associe un Rules, un BoardMatrix, un état, un joueur courant...

Environnement & Simulation

Client Humain

- JQuery
- AJAX polling

Environnement & Simulation

Client machine frontière

Analyseur conceptuel

Généralités

- Représente les connaissances tirées de l'environnement
- Analyse ces connaissances afin d'en tirer des nouvelles



Analyseur conceptuel

Analyse détaillée

- Représentation des connaissances : **vocabulaire**
 - Graphes conceptuels de base ou
 - Formules de logique du premier ordre
- Analyse des connaissances : **mécanisme**
 - Interrogation avec la mémoire
 - Recherche d'homomorphismes



Analyseur conceptuel

Implémentation : Rôles du module

- Convertisseur :
 - Rend les données de l'environnement « lisibles » par l'IA
- Moteur d'inférence :
 - Applique les règles générés par l'IA afin d'en sortir des nouveaux concepts



Analyseur conceptuel

Implémentation : Classes principales

- **Choices (environnement)** : représente un plateau courant, un coup et l'ensemble des plateaux résultants de ce coup
- **BoardMatrix(environnement)** : représente un plateau en forme d'une matrice
- **Choices_FOL (IA)** : version logique du premier ordre de Choices (même structure, attributs décrits par des formules logiques)
- **CompleteBoardState (IA)** : version logique du premier ordre de BoardMatrix (classe qui décrit la configuration d'un plateau complet comme une liste de faits logiques)
- **RelevantPartialBoardState (IA)** : classe qui décrit la configuration d'une sous-partie pertinente d'un plateau comme une règle logique



Analyseur conceptuel

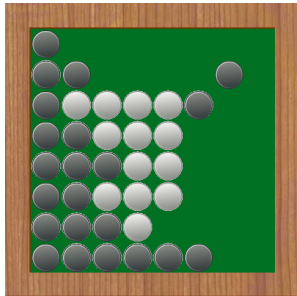
Implémentation détaillée

- **Convertisseur :**
 - Entrée (de l'environnement) : instance de « Choices »
 - Algorithme qui transforme un « BoardMatrix » en un « CompleteBoardState »
 - Sortie : instance de « Choices_FOL »
- **Moteur d'inférence :**
 - Entrée (de la mémoire) : instance de « RelevantPartialBoardState »
 - Algorithme de saturation de la base de faits des « CompleteBoardState » par la règle d'entrée
 - Ajout d'une liste de « RelevantPartialBoardState » présents dans chaque « CompleteBoardState » du paquet « Choices_FOL »
 - Sortie (passée à la mémoire) : instance de « Choices_FOL »



Raisonneur

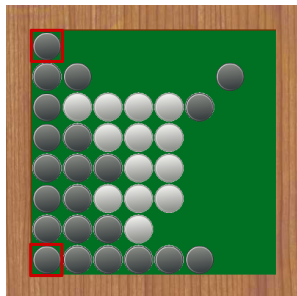
Moteur de choix





Raisonneur

Moteur de choix

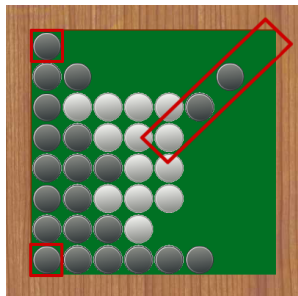


- $isCorner(x) \wedge isMine(x)$



Raisonneur

Moteur de choix



- $isCorner(x) \wedge isMine(x)$
- $isMine(w) \wedge isOpp(x) \wedge isOpp(y) \wedge aligned(w, x, y) \wedge isEmpty(z) \wedge aligned(x, y, z)$



Raisonneur

Valuation des formes

- Valuation d'une forme :

$$P(Gain|Forme) = \frac{P(Forme|Gain) \times P(Gain)}{P(Forme)}$$

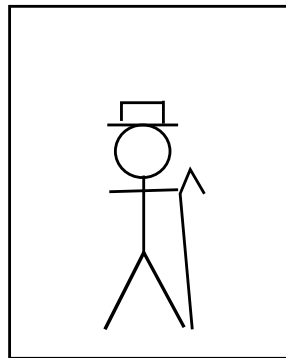
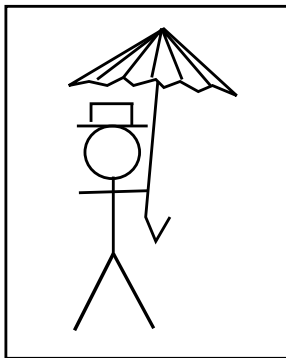
- Fin de partie → mise à jour des formes rencontrées



Raisonneur

Moteur d'introspection

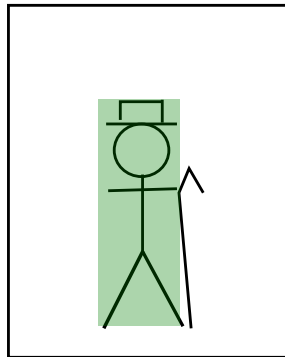
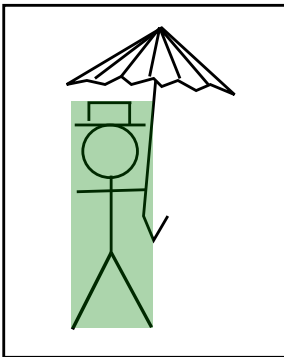
Extraction de formes



Raisonneur

Moteur d'introspection

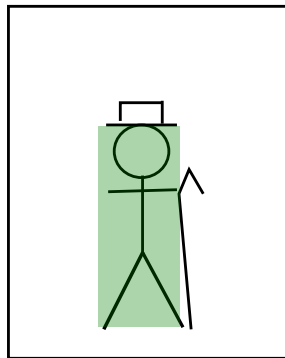
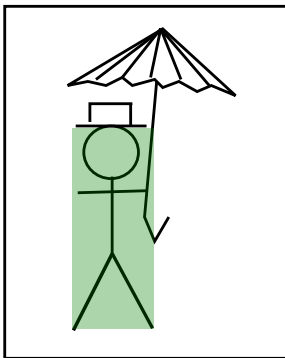
Extraction de formes



Raisonneur

Moteur d'introspection

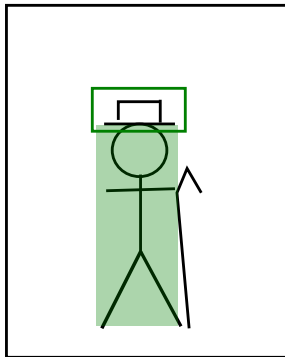
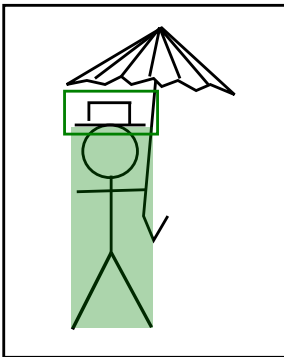
Extraction de formes



Raisonneur

Moteur d'introspection

Extraction de formes





Raisonneur

Moteur d'introspection

$\dots isMine(x0) \wedge$
 $isOpp(x1) \wedge$
 $isOpp(x2) \wedge$
 $aligned(x0, x1, x2) \wedge$
 $isEmpty(x3) \wedge$
 $aligned(x1, x2, x3) \wedge$
 $\dots isCorner(x3) \dots$

$\dots isMine(y0) \wedge$
 $isOpp(y1) \wedge$
 $isOpp(y2) \wedge$
 $aligned(y0, y1, y2) \wedge$
 $isEmpty(y3) \wedge$
 $aligned(y1, y2, y3) \wedge$
 $\dots isCorner(y3) \dots$



Mémoire

Interface Mémoire

Memory

Interface

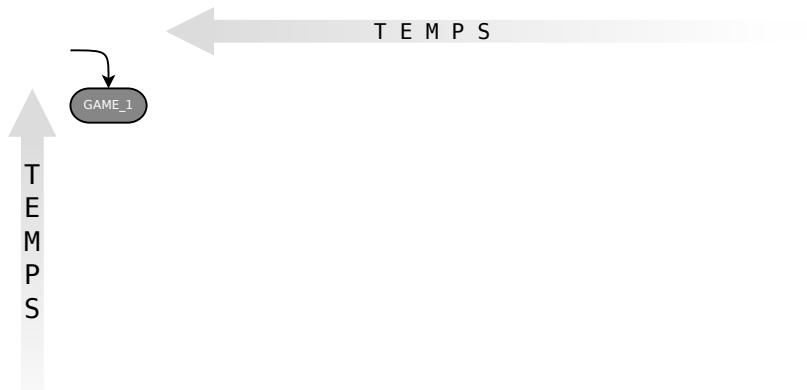
```
+getRelevantPartialBoardStates(): List<RelevantPartialBoardState>
+putOption(option:Option_FOL)

+getGradedOptions(): List<Pair<Option_FOL, Double>>
+OptionChosen(option:Option_FOL)
+BeginOfGame()
+EndOfGame(status:GameStatus,score:int)

+getLastWonGames(n:int): List<Game>
+getLastLostGames(n:int): List<Game>
+getAllRPBS(n:int): List<Pair<RelevantPartialBoardState, Double>>
+putRelevantStructure(rpbs:RelevantPartialBoardState): long
+addAssociation(cbs_id:long,rpbs:RelevantPartialBoardState)
```

Mémoire

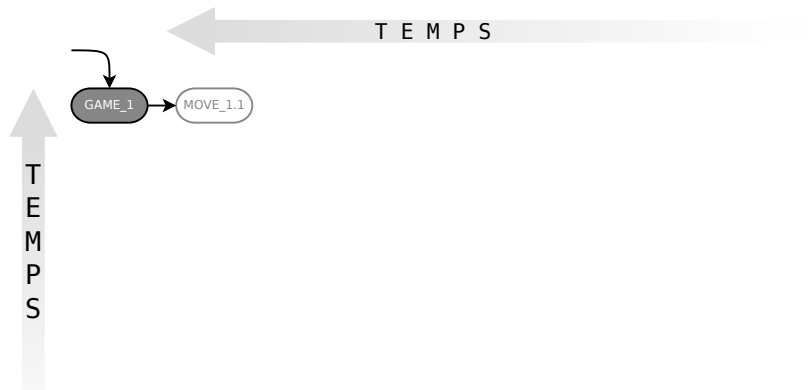
Mémoire épisodique





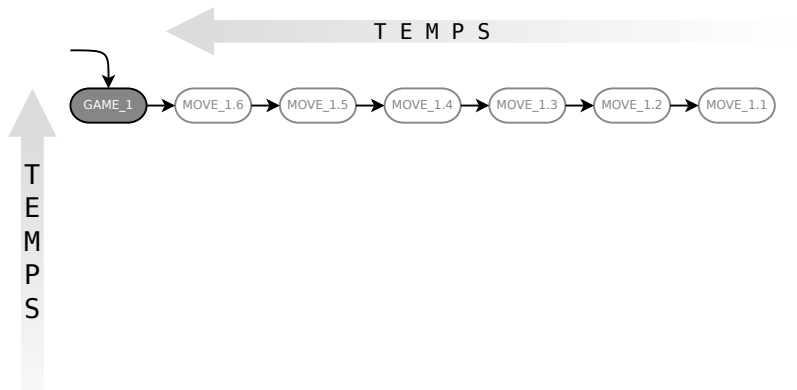
Mémoire

Mémoire épisodique



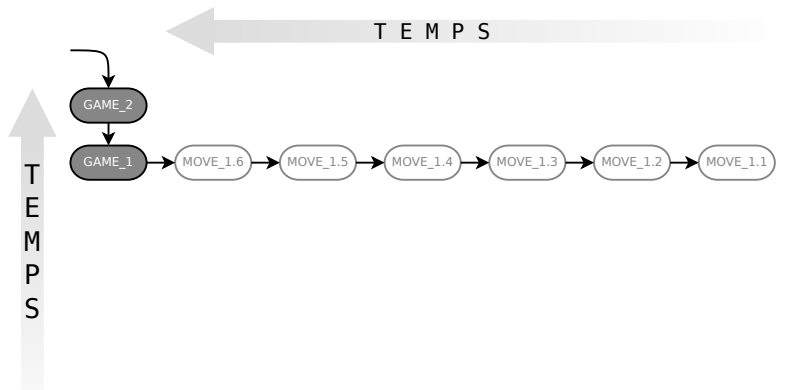
Mémoire

Mémoire épisodique



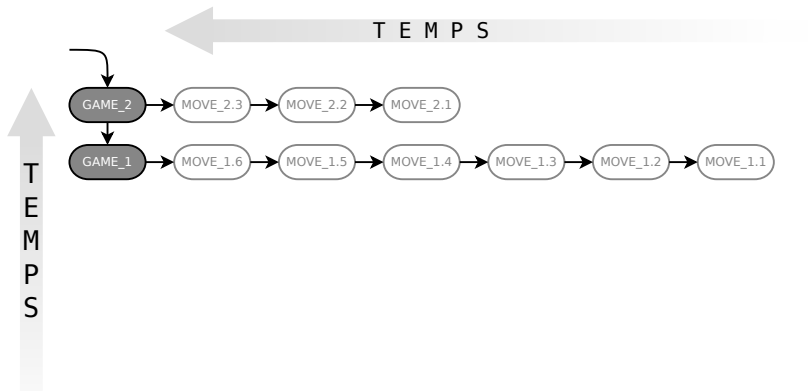
Mémoire

Mémoire épisodique



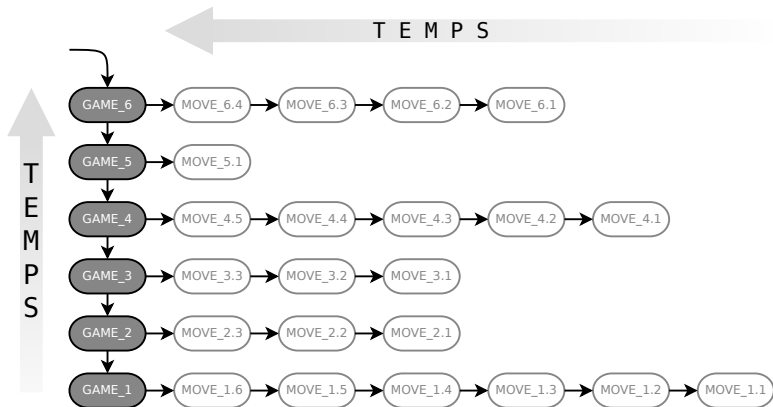
Mémoire

Mémoire épisodique



Mémoire

Mémoire épisodique





Mémoire

Mémoire sémantique

Vision en matrice

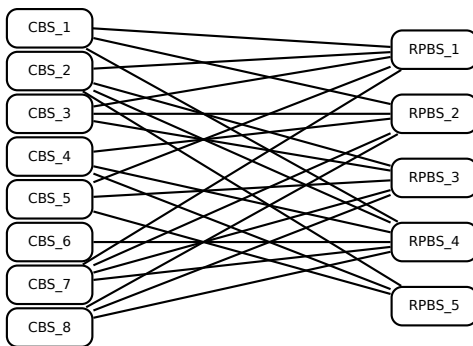
	RPBS_1	RPBS_2	RPBS_3	RPBS_4	RPBS_5
CBS_1	●	●		●	
CBS_2	●		●	●	●
CBS_3	●	●	●		
CBS_4		●		●	●
CBS_5	●		●		●
CBS_6				●	
CBS_7	●	●	●	●	
CBS_8		●	●	●	



Mémoire

Mémoire sémantique

Vision en graphe





Mémoire

Persistence



Neo4j

- Logiciel libre (GPLv3 / AGPLv3)
- SGBD NoSQL orienté graphe
- Respect des caractéristiques ACID
- Multiple versions (embedded in Java)



Mémoire

Persistence

Éléments

- Noeud racine
- Noeuds
- Relations (orientées)
- Types de relations
- Attributs (Noeuds & Relations)

Comment typer les noeuds ?



Mémoire

Persistence

Éléments

- Noeud racine
- Noeuds
- Relations (orientées)
- Types de relations
- Attributs (Noeuds & Relations)

Comment typer les noeuds ?

1. Créer un noeud Maitre
2. Créer une relation typée Racine \rightarrow Maitre
3. Créer des relations typées Maitre \rightarrow Noeuds



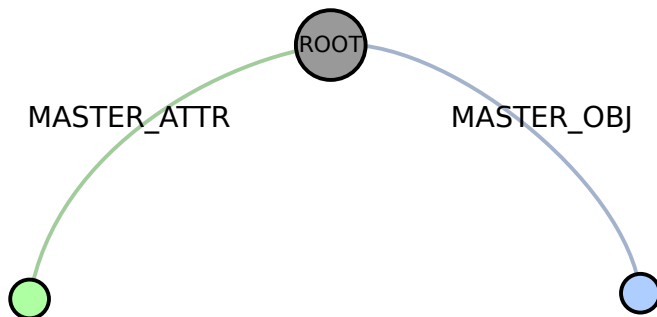
Mémoire

Graphe de la mémoire sémantique

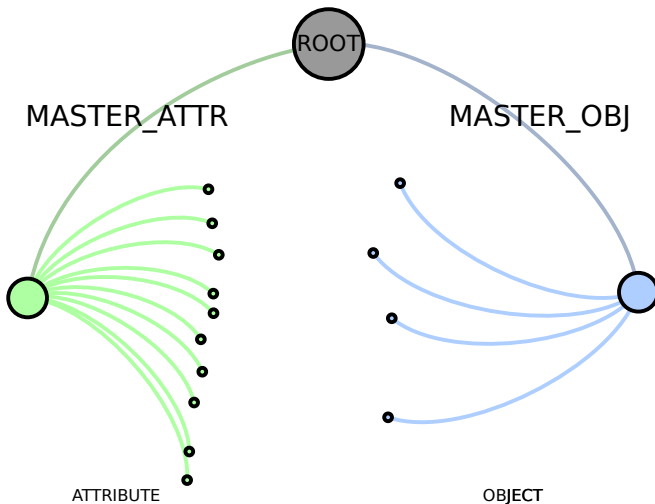


Mémoire

Graphe de la mémoire sémantique

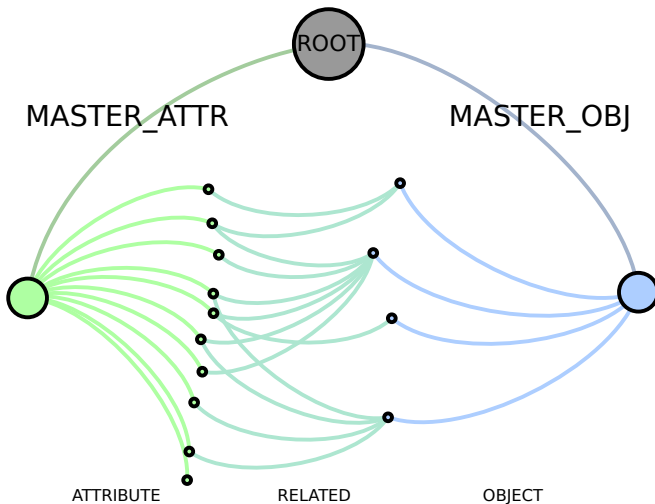


Graphe de la mémoire sémantique



Mémoire

Graphe de la mémoire sémantique





Mémoire

Graphe de la mémoire épisodique





Mémoire

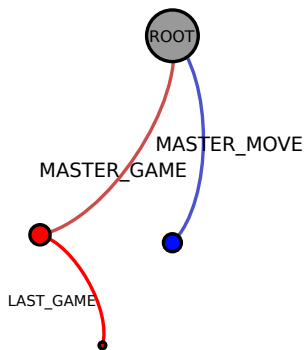
Graphe de la mémoire épisodique





Mémoire

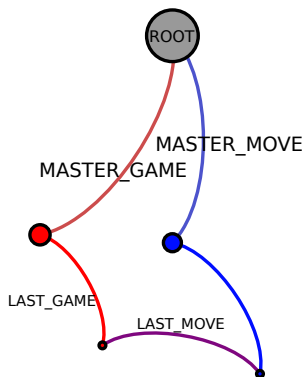
Graphe de la mémoire épisodique





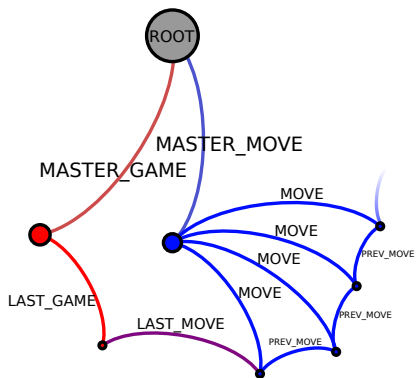
Mémoire

Graphe de la mémoire épisodique



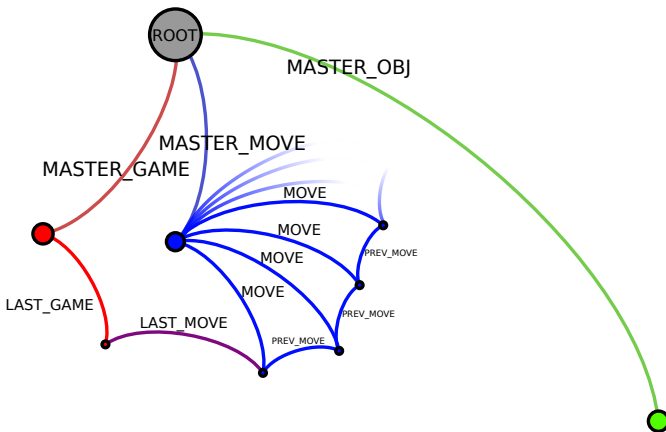
Mémoire

Graphe de la mémoire épisodique



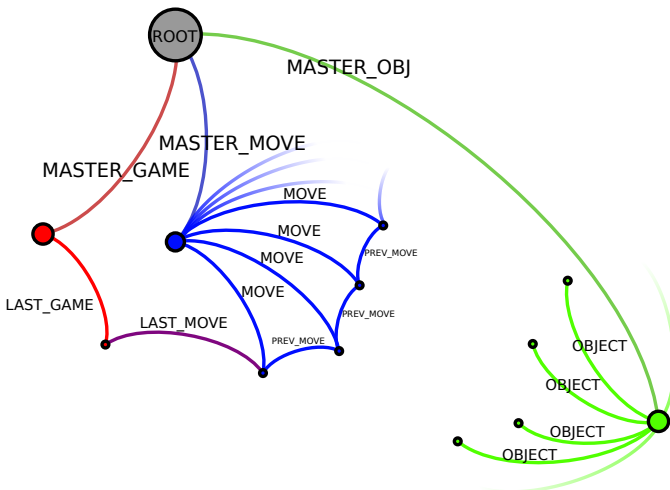
Mémoire

Graphe de la mémoire épisodique



Mémoire

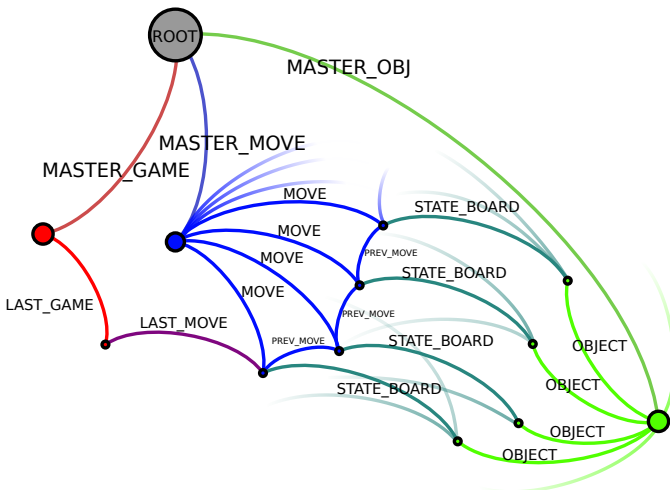
Graphe de la mémoire épisodique





Mémoire

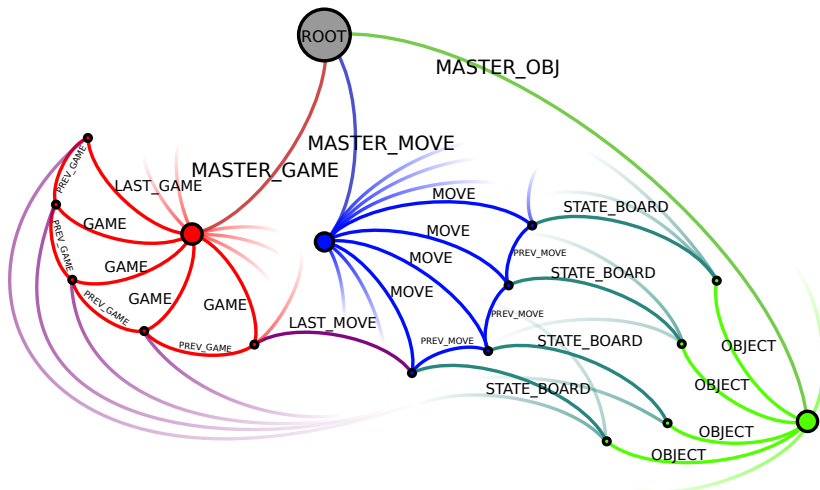
Graphe de la mémoire épisodique





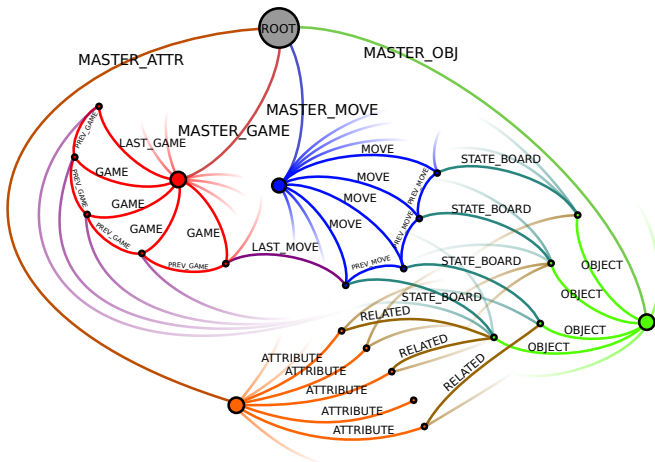
Mémoire

Graphe de la mémoire épisodique



Mémoire

Graphe complet de la mémoire

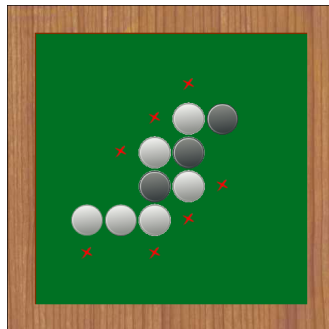


Optimisation

Structures de données

Optimisation

MultiThreading





Optimisation

Autre



Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Analyse Générale

Analyse & Implémentation

Outils de travail

Conclusion & Perspectives

Outils de travail

GIT - Un gestionnaire de version décentralisé



- Logiciel libre (GNUv2)
- Simple d'utilisation
- Hébergement via GitHub

`https ://github.com/marminthibaut/artificial_consciousness/`



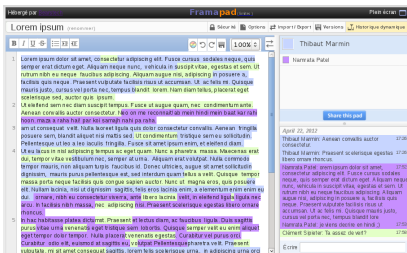


Outils de travail

Etherpad - Éditeur de texte collaboratif

EtherPad

EtherPad is the only web-based word processor that allows people to work together in *really real-time*.



- Logiciel libre
Licence Apache v2
- Collaboratif en temps réel
- Complet
Chat, couleurs, etc.
- Hébergement via Framapad
<http://framapad.org/>

Apêçu d'un pad hébergé sur framapad.org



Outils de travail

Développement



Logiciel Libre (EPL) / EDI renommé

Javadoc

- Standard industriel
- Réutilisation du code source

Log4j

- Outil libre (Apache v2)
- Journalisation



*Pas libre. . . / Plugin Eclipse &
autres EDI*



Analyse et implémentation d'un modèle de conscience artificielle

Introduction

Analyse Générale

Analyse & Implémentation

Outils de travail

Conclusion & Perspectives

Conclusion
Perspectives

Conclusion

Découverte de nouveaux outils / travail collaboratif en équipe

Perspectives

Ce qui (n') a (pas) été réussi
Sacrifices faits (système opérationnelle)
Problèmes rencontrés (NP-complétude)
Forces et faiblesses du système
Pistes à suivre pour une suite éventuelle
Évaluation du système à faire



Perspectives

Un treillis de concepts en mémoire ?

Le contexte

Un contexte est un triplet (G, M, I) avec

- G l'ensemble des objets
- M l'ensemble des attributs
- $I \in G * M$ l'ensemble des relations

On pourrait donc utiliser la mémoire sémantique comme contexte pour la création d'un treillis de concepts.



Perspectives

Un treillis de concepts en mémoire ?

Utile ?

- Abstraction supplémentaire
- Travail sur des ensembles de formes (RPBS)

Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)

Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)

Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)



Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)



Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)



Perspectives

Un treillis de concepts en mémoire ?

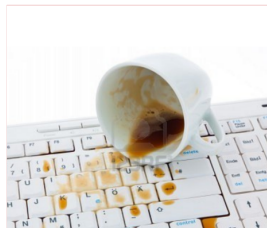
Exemple (concret)



Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)



Perspectives

Un treillis de concepts en mémoire ?

Exemple (concret)

