

## Semaine 3 : Les systèmes experts à base de règles

Lecture principale : Luger (2009)

### Système de production

Source : [Young - Production Systems in Cognitive Psychology](#)

- Un système de production est une collection de règles **si ... alors ...** qui ensemble forment un modèle de simulation et traitement de l'information d'une tâche cognitive ou d'un ensemble de tâches.
- Le fonctionnement est un cycle de reconnaissance (conditions) et d'action
- Caractéristiques :
  - Traitement en série et en parallèle
  - Indépendance des règles
  - Contrôle flexible
- Utilisés dans des domaines avec une forte concentration de connaissances, par exemple le domaine médical et le jeu d'échecs.
- Deux mémoires :
  1. Mémoire de production (règles)
  2. Mémoire de données (informations sur la tâche en cours)
- Résolution de conflits
  - Un conflit se produit si plusieurs règles sont satisfaites simultanément
  - Différentes méthodes de résolution : ordre de précedence des règles, âge des données, complexité des conditions de la règle, ...
- Young and O'Shea (1981) : Résolution de soustractions chez les enfants : une douzaine de règles. Les erreurs les plus fréquentes sont répliquées facilement en enlevant ou ajoutant une règle au système.
- Depuis les années 1990 : Architectures cognitives intégrées :
  - Soar (Newell 1990) :
  - ACT (Anderson and Lebiere 1998) :

### Définition

- Les systèmes experts représentent les connaissances servant à résoudre les problèmes sous la forme de règles **si ... alors ...**
- Plus ancienne forme de représentation des connaissances
- Un système expert à base de règles est construit à partir d'un **système de production** tel que défini par Newell and Simon :
  - Modéliser la performance humaine dans la résolution de problèmes
  - Base de connaissances spécifique au domaine : ensemble de règles de production

- Expérimentation sur la cryptarithmétique : résolution de problèmes arithmétiques où les nombres sont encodés. Observation du processus de résolution et établissement de règles.
- Un système expert doit respecter ces conditions :
  - Ouvert à l'inspection
  - Facilement modifiable
  - Nature heuristique

## Recherche

### Recherche vers l'avant

- La recherche vers l'avant est utilisée lorsqu'il est difficile de formuler une hypothèse ou un but.
- Par exemple, dans un problème d'interprétation.
- Recherche avec parcours en largeur.

### Recherche vers l'arrière

- La recherche par l'arrière dans un système expert revient au test d'une hypothèse.
- L'objectif est identifié et placé en mémoire.
- Des objectifs intermédiaires peuvent être résolus en demandant à l'utilisateur des informations supplémentaires.
- Pas de mécanisme de propagation par l'arrière.
- Recherche avec parcours en profondeur.

## Explication et transparence

Les deux questions auxquelles un système expert à base de règles peut répondre sont : pourquoi ? et comment ?

- Pourquoi demandes-tu cette information ?
  - Retourne la règle courante pour laquelle il manque une information.
- Comment es-tu arrivé à cette conclusion ?
  - Retourne la séquence de règles qui ont permis d'arriver à la conclusion.

Caractéristiques :

- Le programme peut être arrêté à chaque cycle et inspecté.
- Chaque règle représente un élément de résolution du problème, une étape logique.
- Contrairement à un programme traditionnel compilé où l'instruction courante fera peu de sens.

Avantages et inconvénients :

- Le mécanisme est simple, mais le pouvoir d'explication est remarquable.
- Si les règles sont mal divisées par contre, les réponses seront adéquates, mais les explications seront vagues, arbitraires ou illogiques.
- Ceci réduira la confiance de l'utilisateur envers le système.

## Raisonnement à partir des données

Ce mode de raisonnement est une recherche avec parcours en largeur : les conditions des règles sont vérifiées une par une dans l'ordre où les règles sont enregistrées dans la base de connaissances.

**Recherche opportuniste** : Dès qu'une règle est déclenchée et génère une nouvelle information, cette information est ajoutée comme une prémisse et sert à déterminer en priorité les prochaines règles à vérifier.

- La recherche est beaucoup moins orientée.
- L'atteinte d'un objectif n'existe pas vraiment.
- Les explications disponibles pour l'utilisateur sont plutôt limitées. On peut seulement partager la liste des règles qui ont été déclenchées jusqu'à maintenant.

## Heuristiques et contrôle

### Système avec objectif

- Il est possible de contrôler la recherche par la structuration et l'ordonnement des règles.
- Tri des règles :
  - Interprétation des règles de façon procédurale.
  - Doit suivre un plan comme le ferait un expert humain.
- Ordonner les prémisses dans une règle :
  - Ce qui a le plus de chances d'échouer est le plus facile à tester en premier.
  - Tenir compte du coût de calcul de chacun des tests.

### Système à base de données

Trois heuristiques à haut niveau pour la résolution de conflits (Luger, section 6.2.3.) :

- Réfraction :

- Si une règle a été déclenchée, elle ne peut être déclenchée à nouveau tant que les éléments qui correspondent à ses conditions n'ont pas été modifiés. Ceci prévient les boucles.
- Récence :
  - Préférer les règles dont les conditions correspondent à des éléments récemment ajoutés en mémoire. Donne une seule ligne de raisonnement.
- Spécificité :
  - Il est approprié d'utiliser une règle plus spécifique qu'une règle plus générale. On définit la spécificité par le nombre de conditions.

Algorithme RETE :

- Structure en réseau.
- Graphes acycliques directionnels qui représentent des ensembles de règles à plus haut niveau.
- Plus efficace que l'approche naïve qui revient à tester les règles une par une en boucle.
- Utilisé par FICO Advisor

## Langages et outils utilisés

(Source : pages Wikipédia correspondantes)

- Prolog, Scheme, Java
- Langage de systèmes de production OPS5 [Page Wikipédia](#)
- CLIPS : C Language Integrated Production System (1985)
  - [Site officiel](#)
- Système Soar (John Laird, 1983), toujours très utilisé aujourd'hui
  - [Site web officiel](#)
- Paquetage Rulu pour Python
  - [Rulu Documentation](#)

## Jeux et démonstrateurs

- [Clips the Game](#)