* **Ontologies**
  + **Description**
    - Spécification explicite rendant partiellement compte d’une conceptualisation
    - Spécification formelle d’une conceptualisation partagée
    - Distinction entre les entités du monde réel et les catégories telles que les concepts, propriétés et relations pour les décrire
    - Synthèse :

Modélisation partagée d’un domaine pour améliorer la communication entre personnes, entre personnes et applications logicielles, ou entre applications

Caractérisation des objets du domaine : leurs propriétés, leurs relations avec d’autres objets

* + **But** 
    - Se mettre d’accord sur le sens des termes employés dans une organisation, une communauté, un métier
    - Faire en sorte que les personnes et les logiciels se comprennent
  + **OWL**
    - Web Ontology Language
    - Trois sous langages :

OWL lite

Owl DL

OWL full

* + - Objectifs :

Standardiser les moyens de définir des ontologies qui peuvent être utilisées sur le Web

Etendre les constructions de RDF Schema pour permettre des relations plus complexes entre entités

* **SMA : Systèmes Multi-Agents**
  + **Un agent (définition)** 
    - Entité qui peut percevoir son environnement grâce à des **capteurs** et qui peut agir sur celui-ci grâce à des **effecteurs**.
    - Système informatique qui agit d’une façon **autonome et flexible** pour atteindre les objectifs pour lesquels il a été conçu.
* **AG : Algorithmes Génétiques**
  + **Mécanismes d’inspiration**
    - Sélection naturelle et génétique :

Survie des structures mieux adaptées

Propagation génétique

* + - Echanges d’informations pseudo aléatoires
    - Hasard, mais pas purement aléatoire
    - Synthèse intuitive :

Problème = environnement

Solutions = individus évoluant dans cet environnement

On ne retient que les individus les mieux adaptés à l’environnement à chaque génération

Au bout d’un certain nombre de générations, individus bien adaptés (=> solutions robustes)

* + **Fonctionnement simplifié**
    - On dispose d'une population de chaînes de caractères sur un alphabet que nous supposerons binaire.
    - À chaque individu est associé un score (fitness), on en profite pour calculer la somme de ces scores ainsi que les valeurs maximale, minimale et moyenne
    - Trois opérateurs vont être utilisés :

Un opérateur de sélection (select) dont le rôle est de choisir les individus les plus adaptés (ceux possédant un score élevé).

Un opérateur de croisement (cross-over ou hybridation)

Un opérateur de mutation (mutation).

* + - Attention : Avant de pouvoir utiliser un AG, il faut définir :

Un codage adapté au problème (les chromosomes)

Une fonction (fitness) qui va caractériser l'adéquation de la solution (représentée par son chromosome) au problème.

* + **Différences AG // approches plus classiques**
    - Utilisation d’un codage des paramètres // paramètres eux-mêmes
    - Travail sur une population de points // un point particulier
    - Utilisation des fonctions étudiées // leurs propriétés
    - Utilisation de règles de transitions probabilistes // déterministes

**RNA : Réseaux de Neurones Artificiels**

* + **Définition**
    - Réseaux fortement connectés de processeurs élémentaires tournant en parallèle
    - Un processeur : calcul d’une sortie unique en fonction des infos reçues
    - Grand nombre de cellules de base interconnectées
    - Différentes variantes selon le type de cellule, d’architecture du réseau, de dynamique du réseau, et du mode d’apprentissage
  + **Processus connexionniste**
    - 1. Base d’exemples
    - 2. Initialisation du RNA
    - 3. Apprentissage
    - 4. Test
    - 5. Utilisation
  + **Propriétés**
    - Capacité de généralisation :
      * Résout plus qu’il n’a appris
      * Généralise à partir d’exemples si suffisamment nombreux
      * Réponses sensées face à des problèmes nouveaux
    - Rapidité  :
      * Puissance informatique requise peu importante
      * L’essentiel a déjà été fait dans la phase d’apprentissage
    - Mémoire distribuée :
      * Une forme n’est pas mémorisée à un endroit précis
      * Mémoire d’un fait délocalisée ; correspond à l’état d’activation du réseau
    - Résistance au bruit :
      * Traitement de données dégradées
      * Entrées pas trop dégradées acceptables
    - Adaptabilité
      * Apprentissage en cours d’utilisation
      * Evolution et adaptation en constatant les changements de problèmes évolutifs
    - Résistance à la dégradation

**Modèle du neurone formel**

* + - Créé par Mac Culloch et Pitts
    - Modèle mathématique très simple dérivé d'une analyse de la réalité biologique
    - MAIS le modèle biologique fait intervenir une notion temporelle qui est difficile à intégrer dans un modèle simple.
    - **Méthode :** Remplacer l'intégration temporelle par une sommation des signaux arrivant au neurone (appelés les entrées du neurone).
    - Somme obtenue comparée à un seuil, on en déduit de la comparaison la sortie du neurone.
    - Sortie = 1 si somme supérieure au seuil, 0 sinon

**Méthode plus formelle :** Soustraire le seuil à la somme des entrées, et faire passer le résultat par la fonction de Heaviside (cette façon de faire repose sur les propriétés « physiques » du neurone)

**Perceptron**

Premier RNA, mis au point par Rosenblatt

Deux couches :

* + - * Rétine
      * Couche de sortie qui donne la réponse à la stimulation présente en entrée

**Raisonnement à partir de cas**

* + **Etapes**
    - 1. Chercher en mémoire un cas de problème similaire

Difficile à automatiser, relève de l’analogie

Impose de savoir calculer une distance entre le problème actuel et les problèmes en mémoire -> Nécessité de discerner les paramètres pertinents

Impose aussi une représentation hiérarchique (sorte-de, etc)

Problème doit avoir été correctement énoncé

* + - 2. Adaptation de la solution au problème actuel

Impose de posséder des opérateurs de transformation

On peut transposer la solution ou transposer la résolution (passif ou actif)

* + - 3. Mémorisation du nouveau cas

Ne pas simplement stocker en mémoire plate parfois, fusionner les cas en un cas plus abstrait : mémoire hiérarchique

* + **Familles de modèles de RàPC**
    - Modèle Structurel

Toutes les caractéristiques importantes pour décrire un cas sont déterminées à l’avance par le concepteur du système

Cas complètement structurés et représentés par des paires <attribut,valeur> (~frames ou objets)

Calcul de la similarité = distance entre les valeurs de mêmes attributs, puis somme pondérée de ces similarités

* + - Modèle Conversationnel

Mise sur l’interaction entre l’usager et le système pour définir progressivement le problème à résoudre et sélectionner les solutions appropriées

Cas comprend 3 parties : problème P, série de questions (index pondéré) et de réponses Qa, action A

Questions représentées en priorité si elles n’ont pas reçu de réponse, et par ordre décroissant de priorité (pondération)

Quand un cas a un degré de similarité suffisant, on considère que l’on a une réponse, si aucun et plus de questions -> Problème non résolu

* + - Modèle Textuel

Résolution de problème à partir d’expériences dont la description est obtenue dans des documents textuels, peuvent être non structurés ou semi-structurés

**Pattern matching/Unification (pas fait ça en cours je crois… infos qui viennent du net)**

* + **Pattern matching =**

Filtrage par motifs,

Vérification de la présence de constituants d'un [motif](http://fr.wikipedia.org/wiki/Motif) par un [programme informatique](http://fr.wikipedia.org/wiki/Programme_informatique), ou parfois par un matériel spécialisé.

Par contraste avec la [reconnaissance de forme](http://fr.wikipedia.org/wiki/Reconnaissance_de_forme), les motifs sont complètement spécifiés.

* + **Unification**

L’unification est utilisée lors d’une comparaison entre deux éléments, lorsque l’on trouve une propriété qui est variable des deux côtés.

Le pattern matching ne peut pas comparer dans ces cas précis.

**Les différentes approches de l'IA**

**L'IA symbolique** s'attache à élucider et imiter le raisonnement logique.

**But :** Etude du raisonnement

**Origine :** la logique

Manipule des expressions et met en œuvre des processus de raisonnement

Nécessite une représentation explicite des connaissances du problème et des méthodes de résolution

Demande un effort important de programmation

**Exemple** : Systèmes experts, planification

**L'IA numérique** vise la perception et le fonctionnement réflexe.

**But :** Etude de la perception (immédiate, implicite) et du réflexe (idem)

**Origine :** la physiologie

Construit des modèles informatiques

Demande peu ou pas de programmation explicite

Exploite des processus d'apprentissage

Permet de développer des systèmes auto adaptatifs et évolutifs

**Exemple :** Réseaux de neurones, algorithmes génétiques

**L'IA hybride** est l'association de l'IA symbolique et numérique.

**L'IA distribuée/les SMA**

**Recherche dans un espace d’états**

* + - **Espace d’états** = Ensemble des situations qu’un système peut rencontrer par un ensemble d'états accessibles S
    - **Recherche** = Partir d’un état initial pour arriver à un état final, et déterminer les états intermédiaires
    - Le problème est de savoir quelle action effectuer dans tel ou tel état pour atteindre l’objectif donné

**Représentation des connaissances**

* + **Utilité**
    - Sert à leur traitement ultérieur (chercher, organiser, classer, extraire, déduire, établir des contradictions, réviser)
    - On vise la recherche de connaissances implicites mais inhérentes aux faits de la base
  + **Formalismes**
    - **Approches non logiques :** réseaux sémantiques + graphes conceptuels

-> Sens intrinsèque attribué aux structures, existence de procédures adaptées souvent efficaces, facilité de révision

**Approches logiques :** logiques classiques (propositionnelle 1er ordre et 2eme ordre) + logiques de description (famille des logiques modales)

-> Sens externe (interprétation, modèle), procédures génériques plus ou moins adaptées, possibilité de caractériser des propriétés du formalisme et des procédures (expressivité, complétude)

**Le LISP : Programmation fonctionnelle, interactive, et symbolique**

**Fonctionnelle :** Fonctions et expressions peuvent être compilées et évaluées individuellement depuis un environnement interactif, programmation incrémentale

**Interactive :** Pas de structure fixe, un ensemble de fonctions qui s'appellent entre elles. Elle favorise le découpage d'un programme en fonctions (modularité) et la récursivité

**Symbolique :** Gamme de données étendue par rapport aux langages classiques, en Lisp les symboles sont crées et détruits dynamiquement : arguments de fonctions, symboles locaux, symboles globaux… Les valeurs sont également des symboles

-> L’exécution est souvent beaucoup plus compliquée que le programme.

**Frames**

* + - Créés par Minsky
    - Un frame est une structure de données qui représente une situation caractéristique ou prototypique.
    - Modèle conçu pour représenter les connaissances et permettre la réalisation de systèmes règles-faits intimement liés, sur un principe similaire aux systèmes experts mais aussi selon le principe opposé (données déclenchent les règles).
    - Une structure de frame se compose de plusieurs éléments, appelés slots ou éléments terminaux (sur différents niveaux : trois niveaux sont emboîtés : le triplet (frame, slot, facette).