IADM : Création d'un moteur d'inférences

Juin 2015

Projet Arstin

Manuel de l'utilisateur

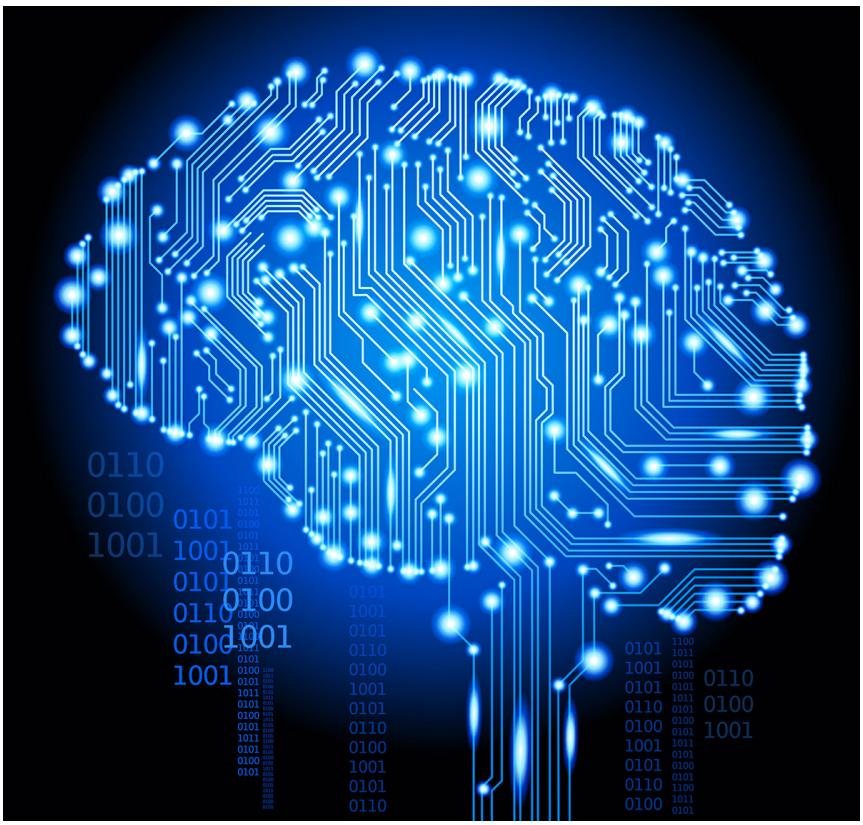


Illustration : Socurious.com

***

Par :

Armand Sibille

Justin Cano

***

Destinataire :

Catherine Jazzar

ÉCOLE CENTRALE MARSEILLE 1A

Table des Matières :

I - Présentation

Arstin est un moteur d'inférences d'ordre 0 permettant d'effectuer des chaînages avant et arrière sur une base de faits et une de données. Les règles et les faits sont exprimés en chaînes de caractère. Notre programme fonctionne à l'aide du langage Java et est exécutable grâce à la console de commande Windows.

Les tailles des bases de règles et de faits n'ont *a priori* pas de limite (si ce n'est les capacités de l'ordinateur de l'usager) et les commandes ne sont pas tenues de respecter la casse pour des raisons de commodité.

Ce programme est disponible librement à l'adresse URL suivante :
http://jcano.perso.ec-m.fr/visible/docs/S6/arstin

II - Lancement du programme

Il convient d'exécuter le programme dans l'invite de commande de Windows.
Pour cela, l'utilisateur devra ouvrir l'invite, et réaliser les opérations suivantes :

A. Se placer dans le répertoire contenant le programme :

Si "C:\Users\Justin\Desktop\Programmation\Arstinv1" est le chemin absolu qui mène au répertoire contenant le fichier.jar de Arstin, taper alors :

```
> cd  C:\Users\Justin\Desktop\Programmation\Arstinv1
```
puis, appuyer sur entrée

B. Exécuter Arstin

Il suffit alors de taper dans l'invite de commande (si l'exécutable .jar se nomme arstin) :
```
> java -jar arstin.jar
```
puis, appuyer sur entrée, le programme doit normalement s'exécuter.

C. Environnement Java

Si un message d'erreur apparaît dans le terminal disant que la version de Java n'est pas reconnue ou bien qu'il manque des fonctionnalités, c'est que l'environnement Java n'est pas suffisant pour supporter notre application.
Il faut donc le mettre à jour à l'adresse suivante :
https://www.java.com/fr/download/

III - Architecture des menus

Les menus sont agencés de la façon suivante : (entre parenthèses, les raccourcis pour accéder aux sous-menus)

- ***MENU PRINCIPAL***
  - ***chainageArriere (car)***
  - ***chainageAvant (cav)***
  - ***editBaseDeRegles (r)***
    - ***resetLocal (rl)***
    - ***resetGlobal (rg)***
    - ***saveBaseDeRegles (s)***
    - ***chargerBaseDeRegles (c)***
    - ***ajouterRegle (a)***
    - ***supprimerRegle (d)***
    - ***printLocal (pl)***
    - ***printGlobal (pg)***
    - ***Exit (q)***
  - ***editBaseDeFaits (f)***
    - ***resetLocal (rl)***
    - ***resetGlobal (rg)***
    - ***saveBaseDeFaits (s)***
    - ***chargerBaseDeFaits (c)***
    - ***ajouterFait (a)***
    - ***supprimerFait (d)***
    - ***printLocal (pl)***
    - ***printGlobal (pg)***
    - ***Exit (q)***
  - ***Exit (****aucun* ***: pour éviter les fausses manipulations)***

On désigne par “exit” les fonctions permettant de faire un “retour” dans le menu d’ordre supérieur, un “retour” dans le menu principal entraîne la **fermeture du programme.**

À noter, le chaînage avant et le chaînage arrière implémentent un module explicatif. N’hésitez pas à suivre tout le raisonnement afin de mieux comprendre le résultat obtenu.

IV- Recommandations syntaxiques

Il est fortement **déconseillé** d'utiliser des faits, conditions, conclusions faisant appel aux caractères: **,** ou **=** ou **[** ou **]**.
De même, il est **déconseillé** de tenter d'utiliser les mots **conclusion**, **exit** pour un fait, une condition, une conclusion.

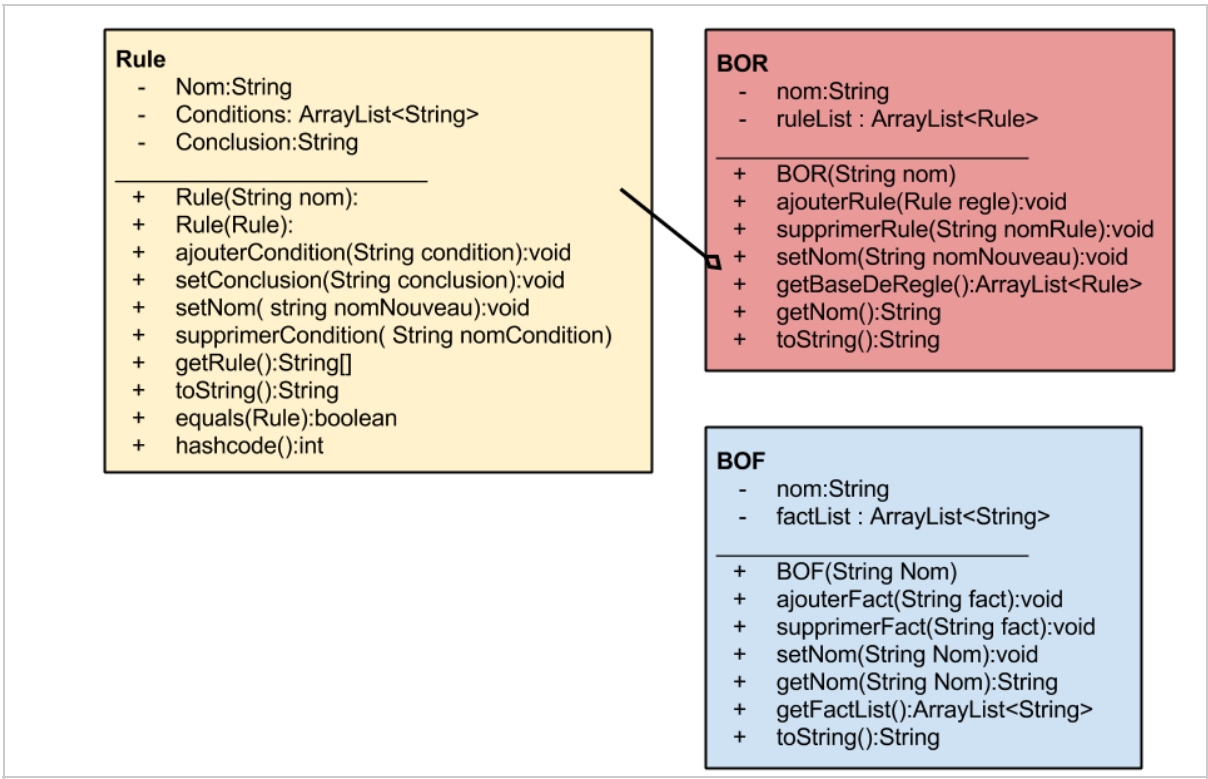
Néanmoins, vous pouvez faire appel à des espaces, utiliser la casse pour vos faits, conditions, conclusions. Éviter cependant les accents en cas de sauvegarde ou de chargement de fichiers.

NB: Les commandes de menus ne tiennent pas en compte la casse

V - Fonctions Principales

A. Avant-propos

Afin de mieux comprendre le fonctionnement de notre programme, voici ci-dessous les diagrammes UML des diverses classes d'objets utilisées par la classe "Main" :



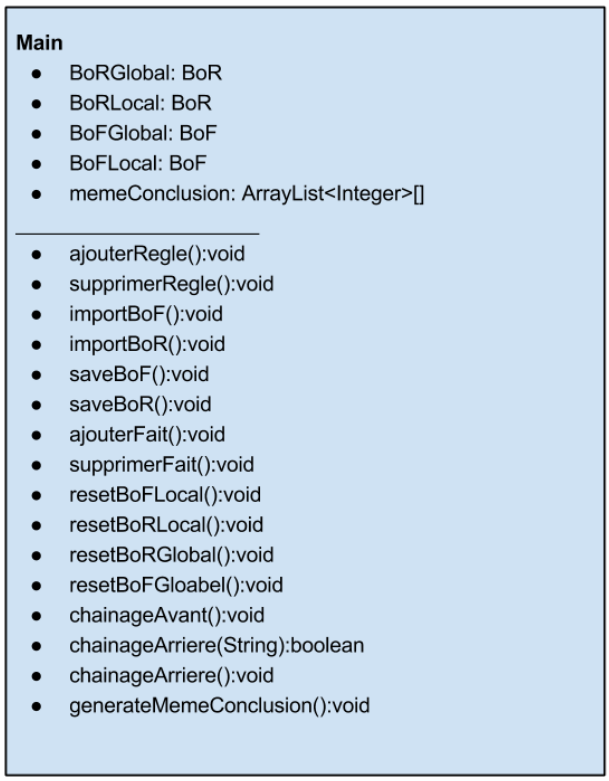
Nous avons crée trois classes permettant de gérer les deux types de bases du programme (faits & règles) :

- ➢ **Rule :** qui apporte leur structure aux règles, c'est à dire un tableau de chaînes de caractères (conditions) et deux chaînes de caractères (nom & conclusion)
- ➢ **BOR :** qui définit ce qu'est une base de règle base de règles, c'est à dire un tableau de règles et une chaîne de caractère (nom)
- ➢ **BOF:** idem pour les faits, sauf que le tableau ici est directement écrit en caractères.

Remarque : Deux bases de chaque catégorie (une locale et une globale) sont générées automatiquement par le programme, ceci permettant à l'utilisateur de se servir des différents chaînages en supprimant ou en ajoutant des règles dans la base locale à partir de la base globale. Ceci évite une recopie fastidieuse des différent(e)s règles/faits.

Ces trois classes sont faites pour assurer le bon fonctionnement du menu de dialogue (Main) qui comporte les méthodes permettant d'assurer les fonctionnalités évoquées au paragraphe III.

Fig 2 : Diagramme UML du menu de dialogue principal



On distingue quatre divisions au sein de ce menu, que nous présenterons ci-dessous. On remarquera que pour exécuter les différentes instances du menu on peut utiliser leur nom complet ou leur raccourcis qui sont rappelés entre parenthèses.

B. Éditer les bases de règles

Lorsqu'il est sur le menu principal, l'utilisateur doit entrer "***EditBaseDeRegles***" ou bien "***r***" afin d'accéder à cette fonction.
Une fois rentré à l'intérieur du menu, plusieurs choix s'offrent à l'utilisateur :

- Réinitialiser la base Globale (rg)
  - Vide la base Globale de toute règle de manière immédiate
- Réinitialiser la base Locale (rl)
  - Vide la base Locale de toute règle de manière immédiate
    - Rq : ces deux méthodes sont irréversibles !!
- Ajouter une Règle (a)
  - Ajoute une nouvelle règle dans la base locale selon la procédure qui suit, si elle est inédite, elle est copiée dans la base globale :
    - Vous pouvez reprendre une règle de la base globale afin de la copier dans la base locale, il suffit d'entrer son nom (qui est par défaut un nombre)
    - Vous pouvez ajouter des conditions successives afin de créer une nouvelle règle
      - Une fois celles-ci entrées, tapez "conclusion"
      - Puis, écrire la conclusion
      - La règle doit normalement être crée
- Supprimer une règle (d)
  - Supprime la règle dont on renseigne le nom
- Enregistrer la base de règles (s)
  - Génère un fichier contenant ladite base, il faut suivre la procédure affichée:
    - Indiquer le chemin du répertoire choisi pour l'enregistrement
      - Ex : C:\users\toto\documents\arstin
    - Indiquer le nom du fichier
      - Ex: maBaseDeRegles
    - La procédure se termine ici, un message de succès doit s'afficher. On peut ouvrir le fichier avec une application de type NotePad, on remarquera que le caractère de séparation est la virgule "," d'où les mises en gardes du paragraphe IV.
- Charger une base de règles (c)
  - Charge une base de règles déjà pré-enregistrée en respectant la procédure suivante :
    - Indiquer l'emplacement du répertoire où votre base se trouve
    - Indiquer le nom du fichier de base de règles (sans extension !)
    - Un message de confirmation doit s'afficher, répondre "oui" à la question

        - Le cas contraire, Arstin peut retourner qu'il n'a pas trouvé le fichier, vérifiez le chemin **absolu** qui mène au fichier.
        - Arstin peut aussi dire qu'il ne s'agit pas d'une base de règles, vérifier que vous ne vous êtes pas confondu(e) avec une base de faits (ou réciproquement).
    - Un message de succès devrait normalement clore l'utilisation de cette fonction.
- Afficher la Base Locale (pl)
- Afficher la Base Globale (pg)

C. Éditer les Bases de Faits

Lorsqu'il est sur le menu principal, l'utilisateur doit entrer "***EditBaseDeFaits***" ou bien "***f***" afin d'accéder à cette fonction.
Une fois rentré à l'intérieur du menu, plusieurs choix s'offrent à l'utilisateur :

Ici, les procédures sont presque analogues aux dernières (pour la base de règles), nous préciserons les différences en marron :

- Réinitialiser la base Globale (rg)
    - Vide la base Globale de toute règle de manière immédiate
- Réinitialiser la base Locale (rl)
    - Vide la base Locale de toute règle de manière immédiate
        - Rq : ces deux méthodes sont irréversibles !!
- Ajouter un Fait (a)
    - Ajoute un nouveau fait dans la base locale selon la procédure qui suit, si il est inédit, il est copié dans la base globale :
        - Vous pouvez ajouter des faits successifs en entrant des chaines de caractères entrecoupées d'appui sur la touche "entrée"
- Supprimer un fait (d)
    - Supprime le fait dont on renseigne la chaîne de caractères.
- Enregistrer la base de faits (s)
- Charger une base de faits (c)
- Afficher la Base Locale (pl)
- Afficher la Base Globale (pg)

D. Procéder au Chaînage Avant :

Dans le menu principal, l'utilisateur doit entrer "***cav***" ou bien "***chainageAvant***", ceci exécutera automatiquement le chaînage avant entre les bases de faits et de règles **locales**, l'instance globale ne servant que "d'entrepôt" temporaire aux divers objets.
Le chaînage avant s'exécute **en explicitant** point par point les étapes qu'il est amené à conduire.

L'**algorithme** utilisé afin de procéder à ce chaînage est le suivant :

```
CompteurA=0;
while CompteurA<Taille(BDR)
	i=0;
	Nombre_de_cond=0;
	Cond_fausse=False;
	if(règle(a) non marquée)
		while( i=<Taille(Regle(a))&&Cond_fausse==FALSE)
			j=0;
			cond_trouvee=FALSE;
			while(j<Taille(BDF)&&cond_trouvee==FALSE)
				if cond(i)==fait(j)
					cond_trouvee == TRUE;
				j++;
			if cond_trouvee == TRUE
				Nombre_de_cond++;
			else
				cond_fausse=TRUE
		if( nombre_de_cond == taille(règle(a))
			ajouter conclusion(règle(a)) à la BDF;
			marquer règle(a);
			CompteurA=0;
		i++;
	CompteurA++;
	a++;
```

A la fin de l'exécution, on peut observer l'évolution dans la base de faits, le programme y ayant ajouté les conclusions des règles.

Remarque : Le marquage se fait ici par l'utilisation d'un tableau de booléens de taille égale à celle de la base.

E. Procéder au Chaînage Arrière

On peut activer le chaînage Arrière depuis le menu principal en exécutant la commande "***chainageArriere***" ou bien son raccourci "***car***".

Lorsqu'on l'exécute, ARSTIN demande le **but** du chaînage, qui devra être renseigné. Puis, il balaiera la base de faits locale et renverra "***true***" si il y trouve le but. Le cas échéant, il cherchera à appliquer une à une les règles activées par la base de faits, si la conclusion de l'une des règles actives est égale au but, il retournera "***true***". Si le but n'est pas la conclusion d'une des règles ou bien que celle-ci reste inactive, il renverra "***false***".

Comme pour le chaînage précédent, **un module d'explication** décrit ce que fait l'algorithme en temps réel.

L'algorithme utilisé est le suivant :

```
aRetourné==False;
fact= premier fait de BdF;
while(reste dans la BdF)&&(aRetourné==False)
if (But==Fact)
       aRetourné==True
       Passer au fait suivant
if ( aRetourné==False)
rule=1ère règle de BdR
       while (aRetourné==False)&&( rule dans BdR)
       if (But==conclusion de rule)
              condV==True
                     cond=1ère cond de rule
                     while ( condV==True) && (cond dans rule)
                     condV=ChaînageArrière(cond)
                            passer à la cond suivante
                     if (CondV==True)
aRetourné==True
                            BdF.add(conclusion)
              passer à la règle suivante
return aRetourné
```

Le but recherché sera visible dans la base de fait au terme de l'exécution si et seulement si elle est un succès.

VI - Exemples d'utilisation

A. Déductions successives (Exemple Centralien)

Dans l'archive .zip contenant le programme ARSTIN vous trouverez un fichier nommé "TestBaseDeFaits" et un autre nommé "TestBaseDeRegles". Chargez ces deux derniers.

La base de règle chargée contient les règles suivantes :

```
[Rule [nom=0, conditions=[eleve, TP de datamining],
conclusion=IADM],
Rule [nom=1, conditions=[IADM, Centrale Marseille],
conclusion=Responsable de l'UE : Mme Jazzar],
Rule [nom=2, conditions=[marseille, ecole, ingenieurs generalistes],
conclusion=Centrale Marseille],
Rule [nom=3, conditions=[tanagra, Centrale Marseille], conclusion=TP
de datamining]]]
```

De même, la base de faits contient :

```
BOF [nom=Local, factList=[Musicien, ecole, eleve, ingenieurs
generalistes, marseille, vient de filiere MP, originaire de Lille,
tanagra]]
```

Chaînage avant :

1. <u>Sans toucher aux faits</u>

Lors du premier passage, les règles 0 et 1 ne sont pas respectées.
On remarque que la règle 2 est immédiatement activée ce qui ajoute le fait "Centrale Marseille", ce qui active la règle 3, qui active la 0 et la 1. Le programme s'arrête ici et a donc déduit que notre professeur d'IADM est Mme Jazzar.

2. <u>En supprimant un fait</u>

Si l'on retire le fait "eleve", les règles 0 et 1 resteront fausse, la seule déduction faisable est que l'on est à Centrale Marseille et que le TP.

3. <u>Rajouter une règle</u>

En faisant attention à la casse, on peut rajouter les règles suivantes :
"Centrale Marseille" + "eleve" + "vient de filiere MP" -> "c'est Armand ?"
"c'est Armand?" + "originaire de Lille" -> "Ah non Armand vient de Pau"
"vient de filiere MP" + "eleve" + "Centrale Marseille" -> "Ah non, Justin est un ex-PSI"
"Ah non, Justin est un ex-PSI" + "Ah, non Armand vient de Pau" -> "mysterieux inconnu"

Chaînage Arrière :
Faire une réinitialisation de la base locale de faits et réimporter “TestBaseDeFaits”.
On peut essayer de voir si il est possible de déduire les différents faits :

Rechercher “tanagra” -> renvoie ***true***, il est initialement dans la base
Rechercher “Centrale Marseille” -> renvoie ***true*** et ajoute Centrale Marseille à la base de faits, il est déductible.
Rechercher “Armand Sibille” -> renvoie ***false***

B. Grand Nombre de conditions (Exemple S.Holmes vs Pr. Moriarty)

Grand ennemi du professeur Moriarty, le célèbre détective vivant au 221B de Baker Street à Londres veut être testé : saura t-il confondre son adversaire, qui vient juste de dérober les joyaux de Sa Majesté ?

Charger la base de règles 221BR :

```
BOR [nom=Local, baseDeRegle=[Rule [nom=12, conditions=[Watson,
Sherlock Holmes, Sebastian Moran, Journal Decoupe, Taille, Megot,
Joyaux Royaux, Empreintes], conclusion=James Moriarty est coupable],
Rule [nom=13, conditions=[Watson, Hotel Whitehall],
conclusion=Megot],
Rule [nom=14, conditions=[Megot], conclusion=Sebastian Moran],
Rule [nom=15, conditions=[Journal Decoupe, Perkins],
conclusion=Hotel Whitehall], Rule [nom=16, conditions=[Hotel
Whitehall, Sherlock Holmes, Watson], conclusion=Joyaux Royaux],
Rule [nom=18, conditions=[Empreintes, Sherlock Holmes],
conclusion=Taille]]]
```

“Pour confondre Moriarty, Sherlock Holmes doit être sur le terrain avec le Dr Watson. Il doit au préalable démasquer le vil complice de Moriarty, le colonel Sebastian Moran, qui détient les joyaux de la couronne. Avec l’aide de l'anthropométrie, il pourra confondre Moriarty d’une empreinte de chaussure qui lui donnera sa taille. Le journal découpé, laissé par Moriarty par défi, permettra d’aller arrêter le professeur retranché à l'hôtel Whitehall grâce à Perkins et sa bande de chiffonniers qui écument les poubelles des hôtels luxueux londoniens. Le mégot, repéré par Watson et identifié par Holmes, au Whitehall, permettra de déduire que Moran est dans sa résidence de Holborn avec les joyaux de la Reine, prouvant le crime.”

Testez le génie du plus célèbre détective britannique en entrant les faits suivants :

```
Sherlock Holmes, Watson, Perkins, Journal Decoupe, Empreintes
```

Réaliser le chaînage arrière puis le chaînage avant de ces bases, n'est-il pas fort ce Sherlock ?