Université de Montpellier – Master Informatique
Traitement sémantique des données (HAI824)

# Une couche sémantique pour l'accès à une base de données de cinéma

# (TP Datalog)

Dans ce TP, nous utiliserons **Clingo**, un moteur de règles du type "Answer Set Programming" (ASP). ASP est un type de programmation logique très puissant, nous n'utiliserons que la partie correspondant à Datalog.

L'intérêt de Clingo pour ce TP est qu'il est disponible en ligne, sans installation sur votre machine. Comme Clingo ne permet pas d'accéder directement à une base de données, nous allons simuler cet accès en traduisant la base de données en une base de faits.

Si vous n'avez jamais utilisé Clingo, lisez la petite présentation à la fin du sujet de TP puis lancez-vous :

https://potassco.org/clingo/run/

---

Considérons une **base de données relationnelle** sur les films qui contient 3 tables :

- Casting [tconst|nconst|category|character]
  - tconst : identifiant de film
  - nconst : identifiant de personne
  - category : rôle de la personne dans le film (actress, actor, director, composer)
  - character : personnage joué par la personne (si son rôle est actress ou actor, sinon la valeur null)
- Person [nconst|primaryname|birthyear|deathyear]
  - identifiant de personne
  - nom de la personne
  - année de naissance
  - année de mort (le cas échéant, sinon valeur null)
- Title [tconst|primarytitle|startyear|runtimeminutes]
  - identifiant de film
  - titre du film
  - année de sortie du film
  - durée en minutes du film

Ces tables ont été construites à partir de imdb : https://developer.imdb.com/non-commercial-datasets/. Nous allons utiliser une **base de faits** qui est la traduction logique d'un *extrait* de ces trois tables, consacré aux films Star Wars.

- ⇨ Télécharger le fichier texte `starwars-factbase.txt` depuis moodle. Pour se conformer à la syntaxe de Clingo (seules les variables commencent par une majuscule), les prédicats associés aux tables ont été renommés casting, person et title, et les constantes ont été également adaptées.
- ⇨ Lancer Clingo et copier-coller la base de faits dans la fenêtre.
- ⇨ Vous allez ensuite compléter votre base de faits en ajoutant des règles : pensez à sauvegarder régulièrement le contenu de la fenêtre Clingo dans un fichier texte.

**Étape 1. Hiérarchie de concepts (ou classes)**

Nous nous intéressons aux concepts suivants : film (**movie**), personne (**person**), personnage (**character**), acteur – quel que soit le genre de la personne - (**actor**) et directeur (**director**).

⇨ Écrire les règles organisant ces classes par spécialisation. Vous ajouterez une classe (**entity**) qui représente toutes les entités. Souvenez-vous que les variables commencent par une majuscule.

**Etape 2. Relations binaires obtenues à partir des relations de la base de données**

Dans un premier temps, nous nous intéressons aux relations suivantes (dont la signification devrait être assez intuitive) :

**hasTitle**(identifiant de film, titre)
**hasReleaseYear**(id film, année)
**hasName**(id personne, nom)
**actressIn**(id personne, id film)
**maleActorIn**(id personne, id film)
**isDirector**(id personne, id film)
**playedBy**(personnage, id personne, id film)

⇨ Écrire les règles définissant ces relations à partir de celles de la base de données (c'est-à-dire à partir des prédicats casting/4, person/4 et title/4).
⇨ Vérifier le bon fonctionnement de ces règles en exécutant Clingo (bouton Run !) et en sélectionnant l'affichage voulu en sortie grâce à la commande show.
Par exemple : `#show hasTitle/2.`

**Etape 3. Signature des relations**

⇨ Ajouter les règles qui définissent les signatures des relations des deux étapes précédentes. On n'ajoutera pas les règles qui correspondent à la classe entity (qui ne nous intéresse pas vraiment). Vos signatures doivent permettre de retrouver toutes les entités des classes qui nous intéressent (cf. question 1).

**Etape 4. Relation actorIn**

⇨ Ajouter les règles définissant la relation binaire **actorIn** qui généralise actressIn et maleActorIn.
⇨ Revoir l'ensemble des signatures de relation pour intégrer cette nouvelle relation (tenez compte des spécialisations entre relations pour supprimer les règles devenues redondantes).

**Etape 5. Premières requêtes**

⇨ Écrire quelques requêtes (sous forme de règles toujours) : pour chaque requête introduisez un prédicat answer, puis demandez à voir les faits obtenus ayant ce prédicat (commande #show).
⇨ Notamment, tester les requêtes suivantes :
- Quels sont tous les titres de film ?
- Quel personnage a été joué par plusieurs personnes ?

**Etape 6. Relations ternaires**

⇨ Ajouter les relations ternaires suivantes :
- **hasDirected**(id personne 1, id personne 2, id film), qui lie un réalisateur (id personne 1) d'un film (id film) à un acteur (id personne 2) qui joue dans ce film.
- **together**(id personne 1, id personne 2, id film) lie deux acteurs ayant joué dans le même film. Pour éviter d'avoir deux faits liant deux personnes pour un même film, vous pouvez utiliser le symbole < au lieu de la différence (not = ou !=).

⇨ Tester vos relations en posant notamment les requêtes suivantes :
- Quels acteurs Georges Lucas a-t-il dirigés ?
- Dans quels films Harrison Ford et Carrie Fisher ont-ils joué ensemble ?
- Qui sont les acteurs qui ont joué ensemble dans au moins 4 films ?

**Etape 7. Connections entre personnes**

- ⇨ Ajouter la relation binaire **connectedWith** telles que deux personnes IDP1 et IDP2 sont « connectées » si
  - IDP1 et IDP2 ont participé à un même film
  - ou IDP1 a participé à un film avec une personne connectée avec IDP2.

- ⇨ En utilisant la négation par défaut (not), poser la requête qui retrouve les couples de (noms de) personnes qui ne sont pas connectées. Remarque : on sort ici de Datalog positif.

**Etape 8 (facultative). On sort à nouveau de Datalog positif.**

Ajouter les règles permettant de répondre aux requêtes suivantes :

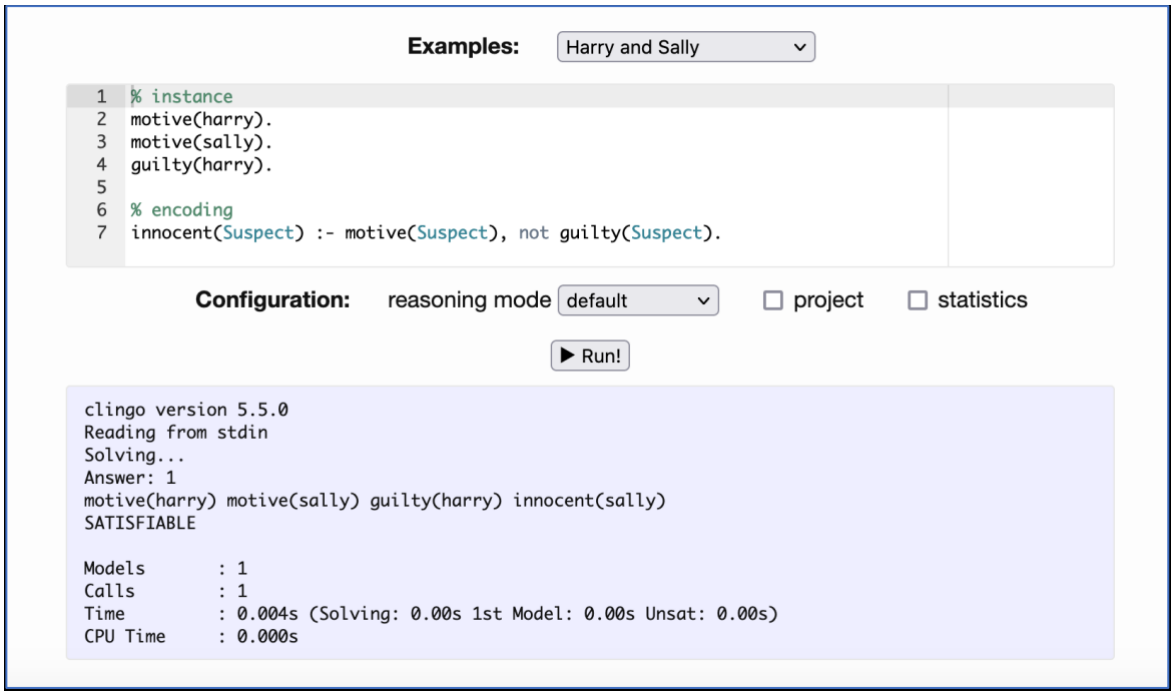
- Y a-t-il un personnage qui n'apparaît que dans 1 film ?
- Quel est le plus vieux film de ma base et quand est-il sorti ?
- Quels sont les films dont tous les acteurs sont vivants ?
- Trouver la distance minimum entre deux personnes connectées.

## Clingo : utilisation et syntaxe (rappels)

Clingo peut être installé sur votre machine, mais il est aussi disponible sous la forme d'une page web, permettant de saisir une base de connaissances dans une fenêtre texte puis de demander l'exécution du chaînage avant sur cette base. C'est cette version que nous allons utiliser.

https://potassco.org/clingo/run/
(ou bien chercher « Clingo online »)

Regardons de plus près le premier exemple fourni sur le site :




Cet exemple comprend une base de faits (appelée instance) et une seule règle. Chaque fait ou règle se termine par un point. Une règle s'écrit sous forme "conclusion **:-** hypothèse", en programmation logique on dit plutôt "tête :- corps" (**head :- body**). Les virgules dans le corps correspondent à des conjonctions et la

négation se note "**not**". Le signe **%** introduit un commentaire. De plus, les symboles commençant par des **majuscules** sont forcément des **variables** (ici, Suspect est une variable). Si on veut utiliser une constante commençant par une majuscule (ou comportant des caractères accentués), il faut l'entourer de guillemets.

La règle donnée en exemple n'est pas Datalog (positif) car elle comporte une négation, désignée par **not** : cette forme de négation est celle du *monde clos*, ou négation *par défaut* : intuitivement, un littéral **not A** est vrai si on ne peut pas prouver que A est vrai (on suppose donc par défaut que A est faux). Pour pouvoir appliquer une règle dont le corps comporte un littéral de la forme **not A**, il faut que **A ne soit pas présent** dans la base de faits courante (avec la négation classique **¬,** on demanderait plutôt que le fait **¬ A** soit présent dans la base de faits).

La situation représentée est la suivante : Harry et Sally ont tous deux un mobile d'avoir commis un certain crime et Harry est coupable. La règle dit que si quelqu'un a un mobile et qu'il n'est pas coupable ("rien ne permet de conclure qu'il est coupable") alors il est innocent.

Lorsque Clingo exécute l'exemple (bouton **Run!**), il détermine d'abord si la base de connaissance est satisfiable. Si c'est le cas, il fournit la (ou les) base(s) de faits saturée(s). Dans le cas général, il peut y avoir plusieurs bases de faits saturées (qu'on appelle des modèles stables - ou answer sets) à cause des négations.

Ici, la base de connaissances est satisfiable et il y a un seul résultat :

```
{ motive(harry), motive(sally), guilty(harry), innocent(sally) }
```

- **Affichage sélectif des résultats.** Pour éviter d'être submergé sous l'ensemble des faits produits, vous pouvez demander à Clingo de visualiser seulement les atomes ayant un certain prédicat, grâce à la commande #show, par exemple :

```
#show guilty/1.  % montre l'ensemble des faits sur le prédicat unaire guilty.
```

Remarquez qu'il vous faut forcément indiquer l'arité du prédicat voulu (en effet Clingo permet d'avoir plusieurs prédicats de même nom et d'arité différente). Vous pouvez ajouter plusieurs commandes #show à la fin de votre base de connaissances.

- **Pour aller plus loin**

  - **Contraintes négatives.** Pour exprimer une contrainte négative, on utilise une règle avec une tête vide qui correspond à ⊥ (un corps vide est toujours vrai, une tête vide est toujours fausse). Par exemple, la contrainte $\forall X.\ animal(X) \wedge plante(X) \rightarrow \bot$ s'écrit :

```
:- animal(X), plante(X).      % contrainte négative
```

  - **Négation par défaut :** `not` ne peut apparaître qu'en corps de règle (en tête `not` n'aurait aucun sens). En présence de négation, il est imposé que les règles soient sûres (safe) : toute variable apparaissant dans un littéral négatif doit apparaître aussi dans un littéral positif du corps de la règle.

  - **Disjonction (attention)** : la `virgule` désigne la conjonction dans un corps de règle et la disjonction dans une tête de règle. Si vous voulez écrire une règle dont la tête est une *conjonction* d'atomes, écrivez une règle pour chacun des atomes de la tête.