Université de Montpellier – Master Informatique
Fondements de l'IA symbolique (HAI710I)

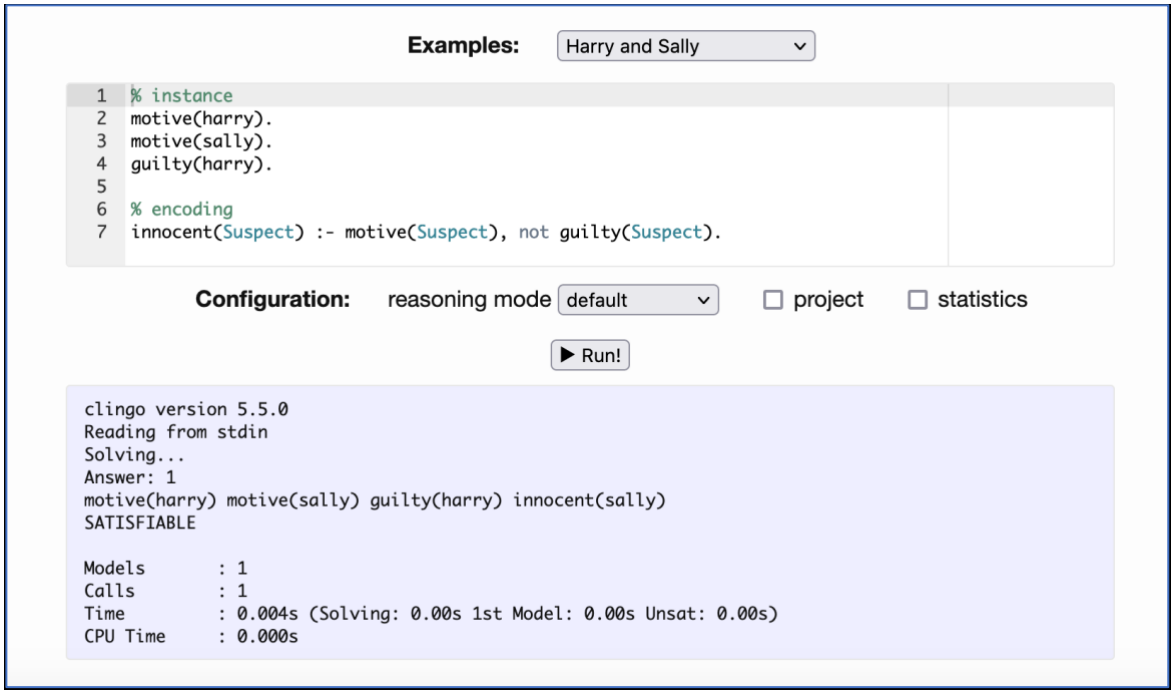
TP Moteur de règles

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 : ce premier TP n'en considère qu'une partie, vous verrez véritablement ce que sont les « answer sets » au prochain TP.

Clingo peut être installé sur votre machine, mais il est aussi disponible sous la forme d'une page web, permettant d'entrer 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).

Ici, **not** désigne la négation du *monde clos*, ou négation *par défaut* : pour pouvoir appliquer une règle dont le corps (hypothèse) comporte un littéral de la forme **not A**, il faut que **A ne soit pas présent** dans la base de faits courante. Voir le prochain cours pour plus de précisions.

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), mais dans ce TP nous n'aurons que des cas avec une seule base de faits saturée.

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

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

Pour commencer (en logique des propositions)

Commençons par un exemple simple en logique des propositions (tous les symboles devront commencer par une minuscule puisqu'il n'y a pas de variable) : Benoît et Cloé organisent une réunion d'amis ; Cloé veut absolument inviter Djamel ; si Félix vient, il faut inviter Amandine ; si Emma vient, il faut inviter Xéna ; Benoît voudrait inviter Félix et Xéna, mais Xéna ne supporte pas Amandine, donc Benoît n'invite Xéna que si Amandine ne vient pas.
Modéliser cette situation avec deux faits (b et c) et 5 règles :

- si c alors d (ou plutôt "d si c", ce qu'on note d :- c.)
- si f alors a
- si e alors x
- si b alors f
- si b et not a alors x.

Qui viendra à la réunion ? La base de connaissances est satisfiable (ouf !) et Clingo fait remarquer au passage qu'Emma ne sera de toutes façons pas invitée, car e n'apparait dans aucune tête de règle, c'est à dire ni dans un fait - vu comme une règle à corps vide - ni dans une conclusion de "vraie" règle.

Pour commencer (en logique du premier ordre)

Passons à la logique du premier ordre. En réalité, Clingo instancie chaque règle par toutes les constantes apparaissant dans la base de connaissances (mais avec plus de discernement que la méthode brutale vue en TD), il se ramène donc à la logique des propositions, même si l'utilisateur ne le voit pas.

1. Modéliser les connaissances suivantes (en 3 faits et 8 règles) en utilisant les prédicats unaires animal, plante, carnivore, herbivore, omnivore et humain, et le prédicat binaire mange (« X mange Y »).

1. La chèvre de Mr Seguin est un herbivore
2. Le loup de Mr Seguin est un carnivore.
3. Le petit chaperon rouge est un humain
4. Les carnivores et les herbivores sont des animaux
5. Les omnivores sont à la fois des carnivores et des herbivores (« si on est un omnivore alors on est un carnivore et un herbivore »)
6. Les humains sont des omnivores
7. Tout animal carnivore ne mange que des animaux
8. Tout animal herbivore ne mange que des plantes
9. Tout animal carnivore mange n'importe quel animal herbivore

Vous constaterez qu'on obtient des faits étranges : le petit chaperon rouge se mange lui-même, la chèvre est une plante, ainsi que le petit chaperon rouge.

2. On modifie la phrase 9 : Tout animal carnivore mange n'importe quel animal herbivore *qui n'est pas lui-même*. Prendre en compte cette modification : le symbole de différence est `!=`, et c'est une macro pour `not ==`, autrement dit (X != Y) si rien n'indique que (X == Y). Le petit chaperon rouge devrait aller mieux.

3. Nous savons que les animaux ne sont pas des plantes (ou l'inverse), autrement dit les deux ensembles d'entités sont disjoints. Ceci peut s'exprimer sous la forme d'une règle appelée "contrainte négative" :

$\forall X.\ animal(X) \wedge plante(X) \rightarrow \bot$

ce qui est équivalent à : $\forall X.\ animal(X) \rightarrow \neg plante(X)$ ou encore $\forall X.\ plante(X) \rightarrow \neg animal(X)$

Avec la syntaxe de Clingo, ceci s'écrit comme une règle à tête vide (un corps vide est considéré comme toujours vrai, une tête vide comme toujours fausse) :

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

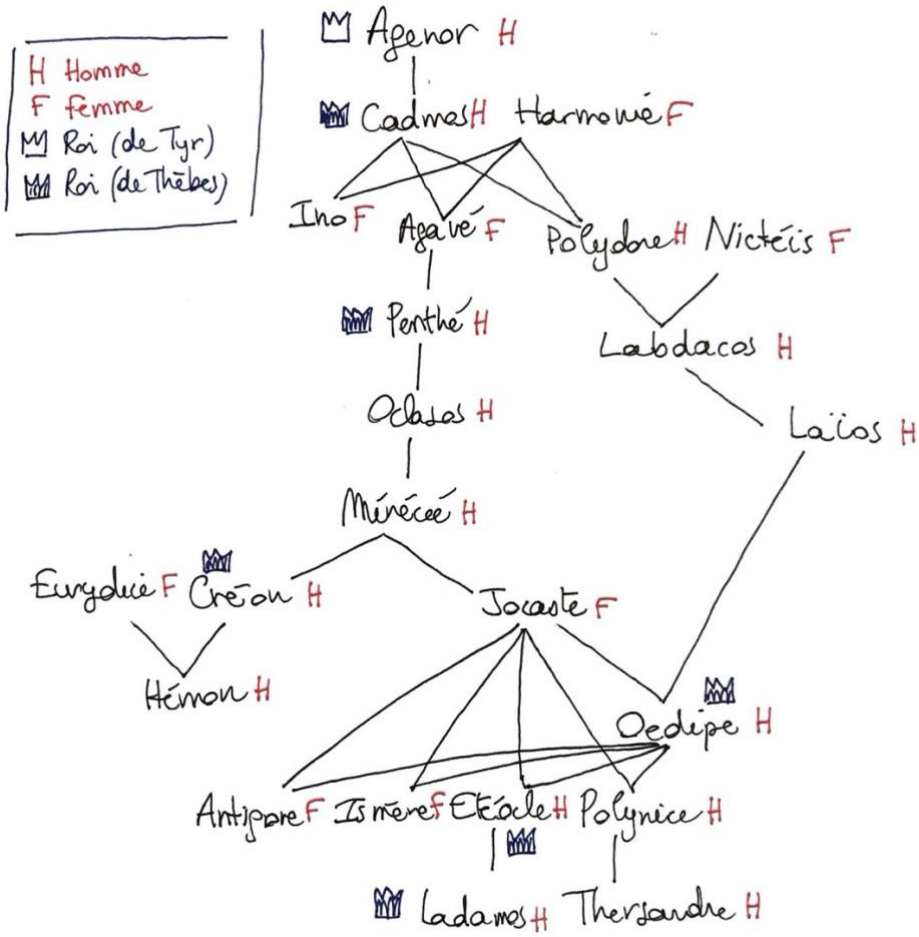
Ajouter cette contrainte. Que répond Clingo ?

4. Bien sûr, c'est la définition d'omnivore qui ne convient pas : un omnivore n'est ni un carnivore ni un herbivore. Commenter les règles qui définissent omnivore, et indiquer simplement qu'un omnivore est un animal. Exécuter à nouveau Clingo et analyser le résultat.

La famille d'Oedipe

Le fichier oedipe-family-facts.lp fournit une base de faits décrivant la famille d'Œdipe, personnage de la mythologie grecque. Vous pouvez copier-coller ce fichier dans la fenêtre de Clingo.

Les prédicats utilisés sont les suivants (où / indique l'arité du prédicat) : **personnage**/1, **homme**/1, **femme**/1, **aEnfant**/2 (qui lie un parent à l'un de ses enfants), **roi**/2 (qui lie un roi à la ville dont il est roi). La figure ci-dessous donne une vue d'ensemble de cette base de faits :



Important : 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 femme/1.  % montre l'ensemble des faits sur le prédicat unaire femme.
```

Vous pouvez ajouter plusieurs commandes #show à la fin de votre base de connaissances.

Questions

N'oubliez pas de sauvegarder la version définitive de vos règles dans un fichier texte.

1. Définir les prédicats binaires **père** (X est père de Y), **mère** (X est mère de Y), **parent** (X est parent de Y). À chaque fois, vérifier que les faits déduits sont ceux attendus. Bien sûr, parent et aEnfant sont des synonymes.
2. Écrire une règle permettant de répondre à la question : qui sont les rois dont le père était déjà roi ? On n'a pas la notion de requête proprement dite, on la remplace par une règle dont le prédicat de tête correspond à la notion de réponse :

   ```
   answer(X) :- requête sur X et autres variables
   ```

   En affichant (`#show`) les atomes qui ont ce prédicat, on obtient les réponses.
3. Écrire une règle permettant de répondre à la question : qui sont les rois dont le père était déjà roi du même lieu ?
4. Définir le prédicat **grand-parent** (X est grand-parent de Y), puis écrire une règle permettant de répondre à la question : qui sont les grands-parents d'Œdipe ?
5. Définir le prédicat **ancêtre** (X est ancêtre de Y) puis écrire une règle permettant de répondre à la question : qui sont les ancêtres d'Œdipe ? On considère que tout ascendant est un ancêtre.
6. Qui sont les personnages de sexe inconnu ? Vous aurez besoin ici de négation (`not`).

   On définit d'abord le prédicat `sexe_connu` :

   ```
   sexe_connu(X) :- femme(X).
   sexe_connu(X) :- homme(X).
   ```

   Ensuite on voudrait écrire :

   ```
   sexe_inconnu(X) :- not sexe_connu(X). % unsafe rule
   ```

   Cependant, Clingo n'admet que des requêtes dites sûres (*safe*) : toute variable apparaissant dans un littéral négatif doit aussi apparaitre dans un littéral positif du corps de la règle. Ici, X n'apparait pas dans un littéral positif du corps de la règle, Clingo ne sait donc pas quelles sont les valeurs possibles pour X.

   On va lui dire que X est un personnage :

   ```
   sexe_inconnu(X) :- personnage(X), not sexe_connu(X). % safe
   ```

   Tous nos personnages ont un sexe connu, mais vous pouvez commenter cette information pour certains personnages et vérifier que vos règles les retrouvent.
7. Qui sont les personnages dont les **deux parents** sont **inconnus** ? Pour cela, définir d'abord la notion de **père inconnu** et de **mère inconnue**. Pour la première notion, on a envie d'écrire ceci :

   "X a un père inconnu si X est un personnage qui n'a pas de père".

   - Première tentative :

   ```
   pere_inconnu(X) :- not pere(Y,X), personnage(X).   % unsafe à cause de Y
   ```

   - Deuxième tentative (on rend la règle sûre) :

   ```
   pere_inconnu(X) :- not pere(Y,X), personnage(X), personnage(Y).
   ```

   mais cette règle dit que "pour tous personnages X et Y, si Y n'est pas le père de X, alors X est de père inconnu", autrement dit elle sélectionne les X dont au moins un personnage n'est pas le père !

   - Une solution :

```
pere_connu(X) :- pere(Y,X). %
pere_inconnu(X) :- personnage(X), not pere_connu(X).
```

- Une solution plus commode : Clingo permet que des variables soient "muettes" dans le corps d'une règle : si une variable apparaît dans *un seul* littéral de *toute* la règle, on peut la rendre "muette", c'est-à-dire ne pas la nommer ; on la remplace par "_". Par exemple, `not pere(_,X)` est vrai pour une valeur `c` de X si on n'a aucun atome de la forme `pere(_,c)`.

```
pere_inconnu(X) :- not pere(_,X), personnage(X).
```

8. Qui sont les personnages dont l'un des deux parents est inconnu mais **pas les deux** ?
9. Définir la notion de **frère ou sœur** (en anglais : sibling) : deux siblings ont leurs deux parents en commun, puis définir la notion de **demi-frère ou demi-sœur** (en anglais half-sibling) : deux half-siblings ont un parent en commun mais pas les deux.
10. Définir la notion de **relation incestueuse** (selon la norme en vigueur dans la mythologie grecque) : deux personnages sont en relation incestueuse si l'un est parent de l'autre et qu'ils ont un enfant en commun. Qui sont les personnages en relation incestueuse ?
11. Un peu plus difficile. Définir :
    - le prédicat `ancetreCommun(Z,X,Y)` : Z est un ancêtre commun à X et Y.
    - le prédicat `plusProcheAncetreCommun(Z,X,Y)` : Z est un plus proche ancêtre commun à X et Y, c'est-à-dire qu'aucun descendant de Z n'est un ancêtre commun à X et Y.
      Vous pouvez si besoin définir des prédicats intermédiaires.
    - un prédicat qui donne la réponse à la question « qui sont les plus proches ancêtres communs à Oclasos et Laios ?
    - un prédicat qui donne la réponse à la question « qui sont les plus proches ancêtres communs à Oclasos et Oedipe ?

      Vérifiez bien que les résultats obtenus sont corrects.