Chaînage arrière :

Exercice 1 du TD sur le chaînage arrière (West est-il un criminel?)

K = (F,R),
Prédicats:
Americain/1
Vend/3 : Vend(x,y,z) : x vend y à z
PaysHostile/1
Criminel/1
Missile/1
Possede/2 : Possede(x,y) : x possede y
Ennemi/2
Arme/1

Faits (F)

- Possède(**Nono**,**M1**)
- Missile(**M1**)
- Ennemi(**Nono**, **Amérique**)
- Américain(**West**)

Règles (R)

- **R1 : Américain(x), Vend(x, y, z), Arme(y), PaysHostile(z) --> Criminel(x)**
- R2 : Ennemi(x,**Amérique**) --> PaysHostile(x)
- R3 : Possede(**Nono**, y), Missile(y) -->Vend(**West**, y, **Nono**)
- R4 : Missile(x) --> Arme(x)

F* :

- Possède(**Nono**,**M1**)
- Missile(**M1**)
- Ennemi(**Nono**, **Amérique**)
- Américain(**West**)
- PaysHostile(**Nono**) par R2
- Vend(**West**,**M1**,**Nono**) par R3
- Arme(**M1**) par R4
- Criminel(**West**) par R1
-

Suite d'application de Règles : R2, R3 et R4 (sans ordre imposé), puis R1

Chaînage arrière : on part d'une requête et on réécrit la requête avec les règles et les faits jusqu'à produire un ensemble d'atome vide. On ne va jamais applique les règles, on va les "remonter"

Exemple :
q0 = Criminel(**West**) ?
On réécrit q0 avec R1 : on unifie Criminel(**West**) avec Criminel(x) cad la tête de R1
q1 = ~~Americain(**West**)~~, Vend(**West**, y, z), Arme(y), PaysHostile(z)

On réécrit q1 avec le fait Americain(**West**)
q2 = Vend**(West, y, z)**, Arme(y), PaysHostile(z)
On renomme les variables de R3 : Possede(**Nono**, y**'**), Missile(y**'**) -->Vend(**West**, y**'**, **Nono**)
On réécrit q2 avec R3 en unifiant Vend**(West, y, z) avec** Vend(**West**, y**'**, **Nono**) : y --> y**'** et z --> **Nono**
q3 = Possede(**Nono**, y**'**), Missile(y**'**), Arme(y'), PaysHostile(**Nono**)
On réécrit q3 le fait Possède(**Nono**,**M1**) en unifiant y' --> **M1**
q4 : ~~Missile(**M1**),~~ Arme(**M1**), PaysHostile(**Nono**)
On réécrit q4 avec le fait Missile(**M1**)
q5 : Arme(**M1**), PaysHostile(**Nono**)
On réécrit q5 avec R4 en unifiant x et **M1** (x --> **M1**)
q6 : ~~Missile(**M1**)~~, PaysHostile(**Nono**)
On réécrit q6 avec le fait Missile(**M1**)
q7 : PaysHostile(**Nono**)
On réécrit q7 avec la règle R2 en unifiant x et **Nono**
q8 : ~~Ennemi(**Nono,Amérique**)~~
On réécrit q8 avec le fait Ennemi(**Nono,Amérique**)
q9 = ø
On a pu prouver q (q0) en utilisant les faits et les règles.

Chaînage avant :

Exercice 2

Graphe de dépendance des règles : graphe orienté dont les sommets sont les règles, et on a un arc (R1,R2) si et seulement si R2 dépend de R1. ("R1 peut éventuellement déclencher R2")

1) **Comment exploiter ce graphe pour éviter de tester TOUTES les règles à chaque étape ?**
A l'étape i > 1, on teste les règles qui sont successeur d'au moins une règle appliquée de façon utile à l'étape (i-1)

2) Critère concret pour dire si R2 dépend de R1 : le prédicat de tête de R1 (q dans l'exemple ci-dessous) apparait dans un atome du corps de R2

R1: p(x,y), p(y,z) --> q(x,z)
R2 : q(u,v), p(v,w) --> r(w)

Sur l'exemple : comment prouver que R2 dépend de R1 selon la définition du début de l'exercice :
Constantes = { a, b ,c, d }
F = {p(a,b) p(b,c) p(c,d) }
R1 applicable sur F ce qui produit q(a,c). [ici aussi on peut avoir q(b,d)]
R2 applicable sur F U {q(a,c)} par un homomorphisme nouveau, ce qui produit r(d).

Petit bémol :
R1 : **p(x,y), p(y,z) --> p(x,y)** // cette règle ne produit jamais de nouvel atome (n'a jamais d'application utile)
C'est juste un cas pathologique : on va supposer qu'on élimine les règles instrinsèquement redondantes comme celle-ci

Graphe de dépendance :
R1 --> R2 , R3
R2 --> R2, R3 [la dépendance de R2 envers R2 est une vraie dépendance - elle peut bien conduire à un nouvel homomorphisme - mais l'application correspondant à ce nouvel homomorphisme ne sera jamais utile].
R3 --> {}
R4 --> R1, R2, R4
[R4 fait une fermeture transitive]

Sur cet exemple, comment exploiter le graphe de dépendance des règles pour éviter de tester toutes les règles à chaque étape ?
On considère la base de faits F = {F1 ... F7}
Etape 1 : on teste toutes les règles

- R1 applicable et produit PermisValable(Ingrid,Danemark)

R4 applicable et produit FaitPartie(Copenhague,UnionE)
Etape 2 : on teste les successeurs de R1 et R4 : ici pas chance, ça nous donne toutes les règles à tester
R1 pas applicable par un nouvel homomorphisme
R2 applicable par un nouvel homomorphisme et produit **PermisValable(Ingrid,France)**
R3 applicable par un nouvel homomorphisme et produit **PeutConduire(Ingrid, Danemark)**
Etape 3 : on teste les successeurs de R2 et R3 : donc R2 et R3 qui sont successeurs

de R2, R3 n'a pas de successeurs.
R2 applicable par un nouvel homomorphisme et produit PermisValable(Ingrid,France) : on l'a déjà --> application inutile
R3 applicable par un nouvel homomorphisme et produit **PeutConduire(Ingrid, France)**
Etape 4 : on teste les successeurs de R3. Rien donc
Fin du chaînage avant.

4) Un fait peut être vu comme une règle à corps vide. Par exemple, F1 peut être vu comme la règle : T --> Ville(Copenhague) ici, on voit T comme "vide"
On insère les faits dans le graphe. Par exemple, F1 a pour successeur R1, F2 a pour sucesseurs R1, R2, etc.
Ce n'est pas la peine d'insérer les faits comme successeurs de règle car ça ne nous apporterait rien.
=> les faits sont insérés comme des sources du graphe (= sans précédesseur)
Qu'est-ce qu'on y gagne?
Etape 1 : on considère seulement les règles qui sont successeurs d'un fait.

Peut-on affiner encore le graphe de dépendance ?

- Cas où les règles ont des constantes dans leur tête (ici R2 par exemple, mais on en tire rien)
- Ex :
- R1 : B(x,y) --> p(x,**a**) où **a** est une constante
- R2 : p(z,**b**), q(z) --> H(z)
- A-t-on R2 dépend de R1 ? Oui selon le critère "concret". Mais selon le critère "abstrait" ? Non
- Critère plus fin : R2 dépend de R1 s'il est possible de rendre identiques la tête de R1 et un atome du corps de R2 en remplaçant des variables
- Plus formellement : R2 dépend de R1 si la tête de R1 est **unifiable** avec un atome du corps de R2
- R3 : p(**b,**z), q(z) --> H(z)
- Ici, R3 dépend de R1 selon le critère affiné car on peut unifier p(x,**a**) et p(**b**,z)
- Unificateur : x --> **b** et z --> **a**

Exercice 1

intuitivement sg = "same generation" (= au même niveau")

1- saturation

Etape **1** : F0 = F
R1 x1 --> a y1--> b sg(a,b) utile autre notation pour l'homomorphisme : {(x1,a),(y1,b) }
R1 x1 --> b y1--> c sg(b,c) utile
R1 x1 --> a y1--> c sg(a,c) utile

Etape **2** : **F1 = F U {sg(a,b), sg(b,c), sg(a,c)}**
R2 (x2,y2,z2,t2) = (d,a,b,d) produit le fait sg(d,d) utile {(x2,d),(y2,a),(z2,b),(t2,d)}
R2 (x2,y2,z2,t2) = (d,a,c,e) produit le fait sg(d,e) utile
~~R2 (x2,y2,z2,t2) = (g,c,b,d) produit le fait sg(g,d) utile~~
R2 (x2,y2,z2,t2) = (d,b,c,e) produit le fait sg(d,e) non utile

Etape **3** : **F2 = F1 U {sg(d,d), sg(d,e) }**

R2 (x2,y2,z2,t2) = (f,d,d,f) produit le fait sg(f,f) utile
R2 (x2,y2,z2,t2) = (f,d,e,g) produit le fait sg(f,g) utile

Etape **4** : **F3 = F2 U {sg(f,f) , sg(f,g)}**
Fin (pas de nouveaux homomorphismes)
F* = F3

2 -
a - Un homomorphisme h est nouveau à l'étape i s'il envoie le corps de la règle dans au moins un atome ajouté à l'étape i-1
(autrement dit, h(corps(R)) n'est pas inclus dans Fi-2

Ici, le corps de chaque règle contient au plus un atome avec un prédicat intentionnel (un seul atome de prédicat sg) : règle linéaire

b- Pour tester si un homomorphisme est nouveau à l'étape i : on a un seul atome qui peut s'envoyer dans un atome produit à l'étape (i-1) et tous les autres s'envoient dans la base de faits de départ. On construit les homomorphismes partiels qui envoient l'atome intentionnel dans un atome produit à l'étape précédente :
    ici pour i = 4 par exemple :
        y2 --> f et z2 --> f
        y2 --> f et z2 --> g
et on complète en regardant dans la base de faits initiale

3 - q() = {sg(x,y), up(y,z), flat(z,**c**)}

Pour la recherche : on part de flat(z,**c**) qu'on cherche dans F (car prédicat extensionnel), forcément on a z -->**a** ou z --> **b**
On procède similairement avec up(y,z) qui prend la forme up(y,**a**) ou up(y,**b**) => forcément y --> **d**
Reste sg(x,y) qui devient forcément sg(x,**d**). On a seulement sg(**d**,**d**) dans F*.

Finalement :
h1 : q --> F*
    x --> d
    y --> d
    z --> a

h2 : q --> F*
    x --> d
    y --> d
    z --> b
Il suffit d'exhiber un de ces 2 homomorphismes pour prouver que la base de connaissances répond positivement à q.