Université de Montpellier - Master informatique Janvier 2026

Théorie des bases de données et de connaissances (HAI933I)

Correction Examen

Durée totale : 2 heures
Document autorisé : 1 feuille A4 manuscrite recto-verso

Cette correction concerne la partie règles existentielles et la partie modèles stables. Au moment où j'écris cette correction, je n'ai corrigé que la partie modèles stables des 45 copies (sur 8 points). Le résultat n'est pas vraiment glorieux: 28 copies ont 2 pts ou moins sur cette partie, et seulement 7 copies ont la moyenne.

Bref, une correction qui m'a déprimé, heureusement qu'un étudiant a pris le temps de me faire une blague (*Et oui, nous y croyons encore*), ce qui me montre que la mienne n'est pas passée inapperçue.

1 First-order queries (3 pts)

Consider the following database:

Films

Title	Director	Actor
The Imitation Game	Tyldum	Cumberbatch
The Imitation Game	Tyldum	Knightley
...	...	...
Internet's Own Boy	Knappenberger	Swartz
Internet's Own Boy	Knappenberger	Lessig
Internet's Own Boy	Knappenberger	Berners-Lee
...	...	...
Dogma	Smith	Damon
Dogma	Smith	Affleck

Venues

Cinema	Address	Phone
UFA	St. Petersburger Str. 24	4825825
Schauburg	Königsbrücker Str. 55	8032185
...	...	...

Program

Cinema	Title	Time
Schauburg	The Imitation Game	19:30
Schauburg	Dogma	20:45
UFA	The Imitation Game	22:45
...	...	...

Question Write the following queries using the relational algebra OR a first-order query.

1. Return all actors who appear in a movie that is being shown at the cinema "Schauburg".
2. Return all domain elements of the database that are not actors.
3. Return the titles of all movies that are featured in exactly two cinemas.

2 Acyclicity and Width (3.5 pts)

2.1 Analyzing a specific query (1.5 pts)

Consider the conjunctive query Q given by

$$\exists w.R_1(x,y) \wedge R_2(y,z) \wedge R_3(z,w) \wedge R_4(z,x),$$

and the database D given by

$$\{R_1(a,b), R_2(b,c), R_3(c,d), R_3(c,e), R_4(c,a)\}.$$

Question 1 It Q acyclic? Explain using one of the ways we learned in class.

Question 2 What is the treewidth of Q? Give a tree decomposition with this width.

Question 3 How many answers does Q have over D? Specify all answers.

2.2 General questions (2 pts)

For each of the following items, does there exist a conjunctive query satisfying the requirements? If so, give such an example and explain why it satisfies the requirements. Otherwise, explain why it cannot exist.

Question 1 An acyclic query with treewidth 2.

Question 2 A cyclic query with generalized hypertree width 2.

3 Règles existentielles (5 pts)

Je vous propose *mon* corrigé de cette partie de l'examen, proposée par Marie-Laure Mugnier. J'avais fait cette correction pour vérifier que les questions correspondaient bien à ce que je vous avais raconté en cours.

Par contre, je n'ai pas corrigé vos copies sur cette partie, aussi les commentaires seront réduits au minimum (les trucs qui m'ont sauté aux yeux quand je parcourais vos copies).

Dans ce qui suit, les bases de connaissances sont de la forme $(F, \mathcal{R})$, où F est une base de faits et $\mathcal{R}$ un ensemble (fini) de règles existentielles (positives).

Question 1 On considère un modèle universel de $(F, \mathcal{R})$. Que signifie *universel* ici ?

Une interprétation *universelle* U d'une base de connaissances $\mathcal{K}$ est une interprétation plus générale que tout modèle de $\mathcal{K}$: pour tout modèle de $\mathcal{K}$, il existe un homomorphisme de U dans $\mathcal{K}$.

La question de cours la plus simple à laquelle on pouvait penser. Vous aviez droit à une feuille de notes manuscrites. Et pourtant j'ai vu de nombreuses erreurs (principalement dans le sens des homomorphismes).

Si vous ne travaillez pas un minimum, on ne peut rien faire.

Question 2 Quel est l'intérêt de la notion de modèle universel pour répondre à des (unions de) requêtes conjonctives ?

Dans le cours, nous avons appelé *représentant* d'une base de connaissances $\mathcal{K}$ une interprétation $\mathcal{I}$ telle que: pour toute (union de) requête conjonctive Q, $\mathcal{K} \models Q$ si et seulement si $\mathcal{I} \models Q$. Disposer d'un représentant permet ainsi de répondre à toute (union de) requête conjonctive par un simple calcul d'homomorphisme dans $\mathcal{I}$, en ignorant les règles.

Nous avons également prouvé en cours que tout modèle universel est un représentant.

Attention, la réciproque n'est pas vraie. Nous avons prouvé en cours que le résultat d'un core chase infini n'était pas un modèle (et donc pas un modèle universel), et avons affirmé (sans le démontrer) que c'était bien un représentant.

Mais je ne pense pas que cette subtilité était exigée par Marie-Laure.

Question 3 Toute base de connaissances $(F, \mathcal{R})$ admet-elle un modèle universel ? Justifiez votre réponse.

Nous avons vu en cours que le résultat d'un chase monotone (quand aucune règle ne peut générer $\bot$) était un modèle universel. Donc toute base de connaissances admet un modèle universel.

Attention, la justification par un chase quelconque ne marche pas, comme on vient de le voir pour le core chase. La preuve n'est valide qu'en exhibant un chase monotone comme l'oblivious, le semi-oblivious, ou le restricted chase.

MAs là encore, il est bien possible que Marie-Laure accepte un chase quelconque.

Question 4 Soit la règle $R_1 : p(x, y) \rightarrow \exists z\ p(y, z)$.
a- Donnez une base de faits F_1 telle que $(F_1, \{R_1\})$ ait un modèle universel *fini*.
b- Donnez une base de faits F_2 telle que $(F_2, \{R_1\})$ n'ait aucun modèle universel *fini*.

a- Si on prend $F_1 = \emptyset$, alors un chase monotone (comme l'oblivious chase) produira $F_1^* = \emptyset$: c'est un modèle universel (voir question 3), et il est bien fini.
b- Si on prend $F_2 = \{p(a, b)\}$, alors le core chase produira $F_2^* = \{p(a, b), p(b, Z_0), p(Z_0, Z_1), \cdots\}$. Comme le core chase ne s'arrête pas, la base de connaissances n'admet pas de modèle universel fini (voir question 6).

Question 5 On considère la base de faits $F = \{p(a, b)\}$ et l'ensemble de règles $\mathcal{R} = \{R_1, R_2\}$, où R_1 est la règle vue plus haut :

$R_1 : p(x, y) \rightarrow \exists z\ p(y, z)$
$R_2 : p(x, y) \wedge p(y, z) \rightarrow p(z, y)$.

a- En utilisant une variante de chase dont vous donnerez le nom, montrez que la base de connaissances $(F, \mathcal{R})$ a à la fois un modèle universel infini et un modèle universel fini.
b- Quelle est la relation entre les différents modèles universels d'une base de connaissances ? Soyez précis dans votre formulation.

Nous avons vu dans un exercice que le *restricted chase* était le seul qui avait un comportement différent suivant l'ordre d'application des règles. La réponse doit tourner autour de ça.

Beuacoup d'étudiants ont assez bien réussi cette partie, qui avait fait l'objet d'un exercice que j'avais corrigé en cours.

a- Si on utilise le restricted chase avec une stratégie "Datalog-first", c'est à dire en appliquant en priorité la règle R_2, alors, après application de R_1 puis de R_2, on obtient $I_2 = \{p(a, b), p(b, Z_0), p(Z_0, b)\}$ et le restricted chase s'arrête ici (toute nouvelle application de règle se replie sur I_2). I_2 est donc un modèle universel fini.

Maintenant, utilisons une autre stratégie: nous allons toujours appliquer R_1 une fois de plus avant d'appliquer R_2. Nous obtenons alors $I^* = \{p(a, b), p(b, Z_0), p(Z_0, Z_1), p(Z_0, b), p(Z_1, Z_2), p(Z_1, Z_0), \cdots\}$ et l'atome introduit par application de R_2 arrive toujours "trop tard" pour pouvoir replier celui introduit par l'application de R_1. Avec cette stratégie, le restricted chase produit ici un modèle universel infini.
b- Un modèle universel s'envoie par homomorphisme dans tous les modèles (donc en particulier dans tous les modèles universels), les modèles universels sont donc tous équivalents.

Pour la partie **b.**, j'ai vu passer un *isomorphe* à la place de *équivalent* ! Or $\{p(a)\}$ et $\{p(a), p(X)\}$ sont équivalents et sont tous deux des modèles universels de $(\{p(a)\}, \emptyset)$, mais ils ne sont pas isomorphes.

Question 6 Existe-t-il une variante de chase dont l'arrêt est garanti sur toute base de connaissances possédant un modèle universel fini ? Justifiez votre réponse.

Par une propriété vue en cours, le core chase est la seule variante du chase qui garantit l'arrêt sur toute base de connaissances possédant un modèle universel fini.

Question 7 Revenons à la question 2 : la notion de modèle universel garde-t-elle son intérêt pour répondre à des requêtes du premier ordre quelconques (donc pas forcément des (unions de) requêtes conjonctives) ? Expliquez.

La démonstration du théorème "les modèles universels sont des représentants" repose sur une étape importante: si Q s'envoie par homomorphisme dans un modèle universel MU, alors pour tout modèle M, comme MU s'envoie par homomorphisme dans M, alors Q s'envoie par homomorphisme dans M (par composition).

Si on veut généraliser à d'autres requêtes que les (union de) requêtes conjonctives, la composition d'homomorphismes que nous avons utilisée devient "si il y a une réponse à Q dans MU et MU s'envoie par homomorphisme dans M, alors il y a une réponse à Q dans M". En d'autres termes, les réponses aux requêtes doivent être stables par homomorphisme.

Si on a des requêtes qui n'ont pas cette propritété de stabilité par homomorphisme, la preuve devient fausse et il y a de bonnes chances que le théorème devienne faux aussi. On va donc tenter un contrexemple avec des requêtes utilisant la négation par l'échec.

Soit la base de connaissance $\mathcal{K} = (\{p(a,a)\}, \emptyset)$. Parmi ses modèles universels, considérons $M = \{p(a,a)\}$ et $M' = \{p(a,a), p(a,Y)\}$.

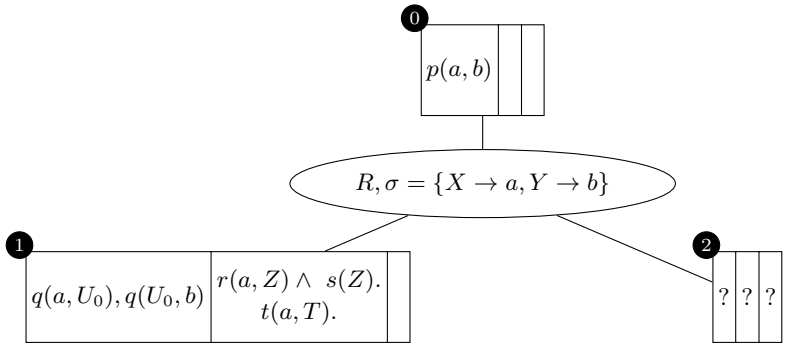
Considérons maintenant la requête `?() :- p(X, Y), not p(Y, Y).`

La réponse à cette reqête est FAUX dans M et VRAI dans M'. Avec une telle requête (qui n'est pas stable par homomorphisme), deux modèles universels ne donnent pas les mêmes réponses: il ne peuvent donc pas être utilisés comme représentants.

La seule question un peu compliquée de cette partie.

4 ASPeRIX: questions de cours (4 pts)

Soit un arbre de recherche ASPeRIX dont la racine est le sommet d'étiquette 0 dans le dessin ci-dessous, où les trois champs des sommets sont, dans l'ordre, IN, OUT et MBT. L'évaluation d'une règle R a produit deux successeurs : le sommet 1 (fils gauche) représente son application.



Question 1 Donnez la règle R (qui ne contient aucune constante) dont l'application a produit le sommet 1.

La règle R qui a généré le sommet 1 est celle-ci:

```
[R] q(X, U), q(U, Y) :- p(X, Y), not(r(X, Z), s(Z)), not t(X, T).
```

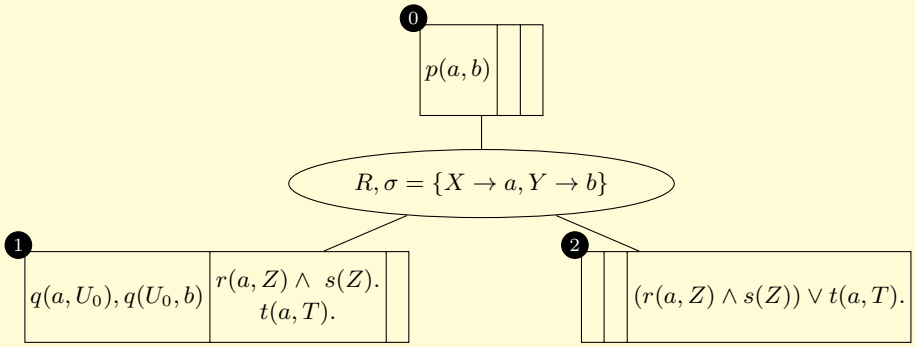
Dans l'ensemble cette question a été bien faite.

Dans les horreurs, j'ai vu trop de règles avec des parties négatives qui se retrouvaient dans la tête (de la négation par l'échec en tête de règle, ça n'a aucun sens).

Plus surprenant, certains étudiants n'ont pas souhaité mettre 2 atomes en tête de règle. Pourtant, la définition le prévoit (H, B^+ et les B_i^- sont des ensembles d'atomes).

Alors il y a eu la solution "pas trop moche" qui consiste à créer deux règles (ce qui donne un arbre équivalent, mais pas égal à celui que j'avais imposé), et les solutions "vraiment fausses" qui consistent à oublier un des 2 atomes de la tête, voire à l'intégrer dans un corps positif ou négatif.

Question 2 Remplissez les champs IN, OUT et MBT du sommet 2 représentant la non application de la règle R.



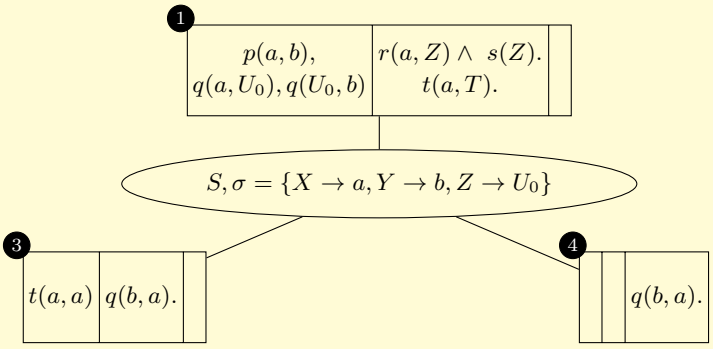
Seuls 2 étudiants ont correctement répondu à cette question ! Presque tous ont rajouté 2 conjonctions dans le MBT ($r(a,Z), s(Z)$ et $t(a,T)$), alors qu'il faut la disjonction de ces conjonctions pour indiquer que l'une d'elles doit être prouvée. Je pensais avoir assez insisté là-dessus (par oral et par écrit) lors de la correction du contrôle continu.

Plus surprenant, un étudiant a rajouté $r(X,Z), s(Z)$ et $t(X,T)$ dans le MBT, tout en se demandant si il ne vaudrait pas mieux spécialiser X par a. Et ce qui m'a le plus surpris, c'est qu'à la question suivante, il spécialise correctement les corps négatifs. Je suppose qu'il avait répondu à sa question, mais trop tard.

Question 3 Evaluez, si c'est possible, la règle S suivante sur les feuilles de l'arbre ASPeRIX :

```
[S] t(X, X) :- p(X, Y), q(X, Z), not q(Y, X).
```

La règle S n'est pas évaluable sur le sommet 2 (il n'y a pas d'atome de prédicat q dans son champ IN), mais nous pouvons l'évaluer sur le sommet 1.



Dans l'ensemble, une question bien traitée (quand il n'y a qu'un corps négatif, tout se passe mieux), mais trop d'étudiants ont essayé d'évaluer la règle sur le sommet 2 (elle n'était pas évaluable), et certains n'ont indique que le fils gauche lors de l'évaluation sur le sommet 1. Pourtant, j'avais fait exprès de vous fournir un exemple d'évaluation dès la première question !

Rappel *(pour les questions suivantes) Dans un arbre ASPeRIX complet et fini, la détection des modèles stables se fait en vérifiant, sur les feuilles, l'absence de violation de* OUT *et la satisfaction de tous les* MBT*.*

Ceux qui lisent l'énoncé ont bien compris qu'on doit vérifier *sur les feuilles.*

Question 4 Expliquez pourquoi la violation du OUT dans le fils gauche du sommet 1 permet de couper l'arbre de recherche sur ce sommet, sans attendre d'avoir construit une branche complète.

La contrainte $t(a, T)$ dans le champ IN du sommet 1 est encore présente dans tous les descendants du sommet 1, en particulier dans le sommet 3 dont le champ IN viole cette contrainte (il y a un homomorphisme de $t(a, T)$ dans $t(a, a)$).

Dans tout successeur de 3 (et en particulier les feuilles d'un arbre complet), $t(a, a)$ sera encore dans le champ IN (car l'application de règle ne fait qu'ajouter des atomes) et $t(a, T)$ sera encore dans le champ OUT.

Aucun successeur de 3 ne peut donc mener à un modèle stable, et on peut immédiatement couper l'arbre de recherche.

D'accord, c'est un peu verbeux, mais je voulais bien expliquer. J'ai accepté (même si ça empiète un peu sur la question 6) la réponse: aucune dérivation commençant par cette branche ne sera persistante, donc la branche ne mènera pas à un modèle stable. J'avoue ne pas avoir pensé à cette méthode. Plus court, mais fait intervenir de l'artillerie lourde à la place d'un argument très simple. Mais comme l'élégance est un facteur subjectif, j'ai compté tous les points.

Beaucoup d'étudiants ont juste dit: "il y a violation du OUT, donc on coupe l'arbre de recherche." C'est bien, c'est l'algorithme vu en cours, mais je voulais une *explication.*

Question 5 Expliquez pourquoi la non satisfaction du MBT dans le fils droit du sommet 1 ne permet pas de couper l'arbre de recherche sur ce sommet, sans attendre d'avoir construit une branche complète.

La contrainte $(r(a, Z) \wedge s(Z)) \vee t(a, T)$. n'est pas satisfaite dans le champ IN du sommet 1.

Pourtant, elle pourrait être satisfaite dans un successeur du sommet 1, si, par exemple, l'application d'une règle rajoutait $t(a, d)$ dans le champ IN.

On ne peut donc pas couper l'arbre de recherche en cas de non satisfaction du MBT.

En général mieux fait que la question précédente. Encore une fois, trop d'étudiants répondent juste "la satisfaction du MBT ne peut se vérifier que sur les feuilles", sans aucune explication justifiant pourquoi le comportement est différent du cas précédent.

Remarque *Le but de deux prochaines questions est de prouver, de façon un peu différente de ce qui a été fait en cours, que l'arbre de recherche ASPeRIX énumère effectivement les résultats possibles des dérivations (finies) persistantes et complètes. Afin de simplifier vos démonstrations, vous pourrez considérer que la base de faits ne contient pas de variables et que les règles utilisées ne comportent pas de variables existentielles (ni de symboles de fonctions de Skolem).*

Pour ceux qui ont le tort de ne pas lire les remarques: si je veux une preuve différente de l'équivalence entre les résultats des bonnes branches et ceux des dérivations persistantes et complètes, c'est à dire une nouvelle preuve qu'ASPeRIX produit des modèles stables, ce n'est pas pour que vous utilisiez l'argument "comme le résultat d'ASPeRIX est un modèle stable" dans votre démonstration.

Les deux questions suivantes ont été peu traitées.

Question 6 Soit $\mathcal{D} = (F_0, \ldots, F_k)$ une dérivation (finie) persistante et complète telle que chaque F_i est obtenu par application de R_i sur F_{i-1} suivant h_i.

On considère maintenant la branche $\mathcal{B}$ d'un arbre de recherche ASPeRIX quelconque construite de la façon suivante:

- la racine a pour champs $(F_0, \emptyset, \emptyset)$;
- nous évaluons à chaque étape la règle R_i suivant l'homomorphisme h_i (dans l'ordre de la dérivation $\mathcal{D}$) et considérons *uniquement* dans la branche $\mathcal{B}$ les fils gauches de cette évaluation (c'est à dire ceux qui correspondent à l'application de la règle).

Vous montrerez que la branche $\mathcal{B}$ est une branche complète, qu'elle ne viole aucun OUT et satisfait tous les MBT (c'est donc un résultat d'ASPeRIX, qui est exactement F_k).

Nous prouvons successivement que la branche $\mathcal{B}$ est complète, qu'elle ne viole aucun OUT, et qu'elle satisfait tous les MBT.

1. **complétude** voir que le champ IN de la feuille f de $\mathcal{B}$ contient exactement F_k. Supposons de $\mathcal{B}$ n'est pas complète. Alors il existe une règle R évaluable sur f qui n'a pas encore été évaluée dans $\mathcal{B}$. Ceci veut dire, par construction, qu'il y a une régle applicable sur F_k qui n'a pas encore été appliquée dans $\mathcal{D}$. Ceci est absurde, puisque $\mathcal{D}$ est complète.
2. **non violation des OUT** supposons qu'une contrainte du OUT introduite par l'évaluation d'un trigger (R, σ) soit violée dans la feuille f de $\mathcal{B}$. Ceci voudrait dire que la règle R n'est plus applicable suivant σ dans F_k. Ceci est absurde, puisque $\mathcal{D}$ est persistante.
3. **satisfaction des MBT** puisque la branche $\mathcal{B}$ ne contient que des fils gauches, alors tous ses champs MBT sont vides. Par vacuité, la feuille b de $\mathcal{B}$ satisfait tous les MBT.

Il y avait trois arguments principaux à exposer:

1. puisque $\mathcal{D}$ est complète, alors $\mathcal{B}$ est complète;
2. puisque $\mathcal{D}$ est persistante, alors il n'y a aucune violation du OUT dans $\mathcal{B}$;
3. puisque $\mathcal{B}$ ne contient que des fils gauches, alors ses MBT sont tous vides et donc la feuille de $\mathcal{B}$ les satisfait tous.

Je voulais *absolument* voir quels arguments sur $\mathcal{D}$ étaient utilisés pour chaque conclusion sur $\mathcal{B}$.

Beaucoup ont commencé en disant que $\mathcal{D}$ était persistante et complète, ont continué par un gloubi-boulga (ceux qui on la ref n'ont plus l'âge d'être en M2) incompréhensible où il est impossible de reconnaitre ce qui sert à quoi, puis ont conclu. Je ne sais pas si vous avez conscience que vous essayez de m'arnaquer, mais ça ne marche pas.

Enfin, il y a ceux qui n'ont pas compris l'esprit de l'exercice et on dit que F_k était un modèle stable, donc le résultat de la branche était un modèle stable, et donc la branche ne violait aucun OUT et satisfaisait tous les MBT. Alors tout d'abord, vous utilisez le résultat qu'on cherche à prouver. Et ensuite, la démonstration est quand même fausse. En effet, on peut avoir des arbres ASPeRIX qui produisent deux sommets ayant même champ IN, mais l'un est une bonne branche et pas l'autre. Pour vous en persuader, vous pouvez construire l'arbre ASPeRIX du programme suivant, en évaluant R_1 avant R_2.

```
a.
[R1] c :- a, not b.
[R2] c :- a.
```

Question 7 Réciproquement, considérons maintenant une branche complète finie $\mathcal{B}$ de l'arbre de recherche ASPeRIX qui ve viole aucun OUT et satisfait tous les MBT. Notez que, contrairement à celle que nous avons construite à la question précédente, cette branche peut contenir aussi bien des fils gauches (application) que des fils droits.

Vous montrerez comment extraire de cette branche une dérivation dont le résultat (le dernier F_k) est le même que celui de la branche $\mathcal{B}$ (l'union de tous ses IN). Vous démontrerez que cette dérivation est persistante et complète.

Nous construisons la dérivation $\mathcal{D}$ qui correspond à toutes les applications de règles. Nommons $S_0, \cdots, S_k$ une partie des sommets de $\mathcal{B}$, avec S_0 sa racine et le reste des S_i correspondant aux

sommets gauches de la branche (ordonnés de la racine jusquà la feuille).

- F_0 est le champ IN de F_0;
- Pour $i \geq 1$, si S_i a été créé par évaluation de (R, σ) sur $p(S_i)$, alors on pouvait déjà évaluer (R, σ) sur S_{i-1} (car les fils droits entre S_{i-1} et S_i n'ont rien rajouté au champ IN). On peut donc construire F_i par application de (R, σ) sur F_{i-1}, et on vérifie que F_i contient les mêmes atomes que le champ IN de S_i.

Reste à vérifier que la dérivation $\mathcal{D}$ que nous avons construite est bien persistante est complète.

1. **persistance** si on a appliqué (R, σ) sur F_i, alors (R, σ) est encore applicable sur σ sinon un élément du champ OUT de S_i serait violé dans S_k.
2. **complétude** supposons (R, σ) applicable sur F_k qui n'a pas encore été appliqué dans la dérivation. Alors, puisque $\mathcal{B}$ est complète, (R, σ) a été utilisée pour générer un fils droit de $\mathcal{B}$. La feuille f de $\mathcal{B}$ contient donc dans son champ MBT la disjonction D des $\sigma(B_i^-)$ (où les B_i^- sont les corps négatifs de R). Puisque $\mathcal{B}$ satisfait tous les MBT, alors le champ IN de f satisfait D. Voir que S_k a le même champ IN que f, et donc S_k satisfait D. Donc F_k satisfait D, ce qui veut dire que (R, σ) est bloqué dans F_k, ce qui est absurde.

Parmi les rares étudiants qui ont tenté cette question, beaucoup ne m'ont pas explicité comment ils construisaient leur dérivation: c'était donc une démonstration dans le vide.

Seule une étudiante a correctement identifié que c'est la satisfaction des MBT qui assurait la complétude de la dérivation (bravo Sonia).

5 Modélisation : un problème d'emploi du temps (4,5 pts)

Dans cet exercice, nous considérons une base de faits qui liste des groupes d'étudiants qui doivent présenter leur travail, ainsi que des dates et des salles disponibles pour ces soutenances. Le code ci-dessous présente une partie de cette base de faits. L'objectif de l'exercice est d'affecter chaque groupe à une salle et à une date, tout en respectant certaines contraintes. Les questions 2 et 3 peuvent être traitées de façon indépendante.

Vous utiliserez des règles **conjonctives** : autrement dit vous avez droit à la négation stable, mais pas à la disjonction en tête de règle. Vous avez également droit aux contraintes négatives (règles avec $\perp$ en tête).

```
groupe(g1).  groupe(g2).  % liste de tous les groupes
salle(s1).  salle(s2).  % liste de toutes les salles
date(d1).  date(d2).  % liste de toutes les dates/créneaux horaires
```

Question 1 Une *solution possible* du problème est un ensemble d'atomes (sans variable) de la forme `affectation(G, S, D)` (signifiant *"le groupe G soutiendra dans la salle S à la date D"*) tel que :

1. pour tout groupe g, il existe un et un seul atome de la forme `affectation(g, S, D)`.
2. pour tout couple (s, d), il existe au plus un atome de la forme `affectation(G, s, d)`.

Vous écrirez un programme dont les modèles stables correspondent aux solutions possibles.

```
affectation(G, S, D) :- groupe(G), salle(S), date(D), not autre_affectation(G, S, D).
% attention:  deux cas d'autrre_affectation !
autre_affectation(G, S, D) :- salle(S), date(D), affectation(G, S1, D1), S != S1.
autre_affectation(G, S, D) :- salle(S), date(D), affectation(G, S1, D1), D != D1.
% ne pas oblier que 2 groupes ne peuvent passer au même endroit au même moment
!  :- affectation(G, S, D), affectation(G1, S, D), G != G1.
```

Ici, je demandais une *légère* modification du pattern du **choix multiple** qui a été vu de nombreuses fois en cours, et sur lequel j'avais insisté: il peut servir à toutes les modélisations que je propose. Et bien seule une étudiante l'a fait (presque[a]) correctement (bravo Sonia).

[a]Dans le corps de la règle qui génère `autreAffectation`, vous avez mis `S != S2` et `D != D2`. Le problème est que si un seul des 2 est différent, ça ne génèrera pas `autreAffectation`, alors qu'il le faudrait.

Le pattern du choix multiple on connait tous les éléments d'un domaine et tous les éléments d'un codomaine, et on veut générer les modèles stables correspondant à chaque *fonction* de domaine dans codomaine. Et bien ça s'écrit comme ça:

```
dom(d1).dom(d2).
cod(c1).  cod(c2).
image(D, C) :- dom(D), cod(C), not autre_image(D, C).
autre_image(D, C) :- image(D, C1), cod(C), C != C1.
```

Intuitivement, la première règle veut dire "si D est un domaine et C un codomaine, et que je n'ai pas autre chose que C comme image de D, alors C est image de D". La deuxième règle explicite ce que veut dire "avoir autre chose que C comme image de D": il faut avoir trouvé une image $C1$ de D, *et il faut* $C1$ *distinct de* C.

Le problème est que ce pattern ne résiste pas à des modifications, oublis ou improvisations. Oublier la différence dans la deuxième règle fait que plus aucune application de la première ne sera persistante (vous pouvez dérouler l'arbre ASPeRIX pour le vérifier), et vouloir éviter le prédicat auxiliaire `autre_image` ne peut mener qu'à une catastrophe (vous ne pourrez pas mettre la différence dans le `not`, car ce serait une imbrication de `not`).

En résumé, ce pattern aurait dû être sur votre fiche.

Erreur très fréquente que j'ai vue sur cet exercice: une règle positive qui dit "si j'ai un groupe, une salle et une date, alors je les affecte" puis des contraintes dont vous espérez qu'elles vont virer ce qui ne marche pas. Mais votre règle sans corps négatif ne génère qu'une unique branche, que vos contraintes vont invalider, et il n'y aura pas de modèle stable.

Enfin, il ne fallait pas oublier la dernière contrainte pour éviter que 2 groupes se retrouvent au même endroit au même moment. A ce sujet, beaucoup d'étudiants ont écrit leurs contraintes, comme en clingo, avec une tête vide. Attention, ceci vient du fait que, en clingo, la tête est une disjonction et donc que la tête vide est absurde. Or nous généralisons ici les règles existentielles, la tête est une conjonction et donc la tête vide est valide. Il aurait donc fallu le symbole $\bot$ (ou `!`) pour la tête des contraintes. N'étant pas très pointilleux sur la syntaxe, je ne vous ai pas enlevé de points.

Question 2 Afin de gérer la disponibilité des différentes personnes , nous ajoutons à la base de faits les informations suivantes (là encore, il ne s'agit que d'une partie des informations disponibles). Ces atomes listent les membres, tuteurs et jurys de chaque groupe, ainsi que les indisponibilités des personnes (l'atome `indisponible(P, D)` signifiant que la personne P ne pourra pas assister à une soutenance à la date D). Pour gérer les indisponibilités de ces personnes, on ajoute à la base de faits les informations suivantes (là encore, il ne s'agit que d'une partie des informations données).

```
membre(john, g1), membre(paul, g1), membre(george, g1), membre(ringo, g1).
tuteur(brian, g1), jury(yoko, g1).
indisponible(yoko, d1), indisponible(yoko, d3), indisponible(yoko, d4).
```

Ecrivez les règles permettant de gérer les contraintes découlant de ces nouvelles connaissances. En particulier :

1. si une personne est liée (membre, tuteur ou jury) à deux groupes, alors ces deux groupes ne pourront pas soutenir à la même date;

2. si une personne est liée à un groupe, alors ce groupe ne peut pas soutenir à une date déclarée indisponible pour cette personne.

```
dans_groupe(T, G) :- tuteur(T, G).
dans_groupe(J, G) :- jury(J, G).
dans_groupe(M, G) :- membre(M, G).
!  :- affectation(G, S, D), affectation(G1, S1, D), G != G1,
   dans_groupe(P, G), dans_groupe(P, G1).
!  :- affectation(G, S, D), dans_groupe(P, G), indisponible(P, D).
```

Question mieux réussie dans l'ensemble. Attention, beaucoup d'étudiants ont ici écrit des règles et pas des contraintes, ce qui ne permettait pas de filtrer les modèles stables. L'oubli des trois premières règles "utilitaires" a souvent mené à de trop nombreuses contraintes, ou à utiliser de la disjonction (membre ou jury ou tuteur) dans les corps, ce qui n'est pas autorisé par la syntaxe vue en cours.

Question 3 De la même façon, les salles ne sont pas toutes équipées de la même manière, et certains groupes ont besoin d'un équipement particulier. Ceci est exprimé dans la base de faits de la façon suivante.

```
nécessite(g2, vr).  nécessite(g3, visio).
equipement(s1, vr).  equipement(s1,visio).  equipement(s2, visio).
```

Ecrivez la ou les règles exprimant que si un groupe a besoin (nécessite) d'un certain équipement, alors il doit passer dans une salle qui fournit cet équipement particulier.

```
!  :- necessite(G, E), affectation(G, S, D), not equipement(S, E).
```

Certainement la question la plus simple, et pourtant j'ai vu du grand n'importe quoi. Il fallait juste une contrainte disant qu'il est impossible d'avoir un groupe qui a besoin d'un équipement qui n'est pas dans la salle dans laquelle ce groupe passe.

Question 4 Malheureusement, la première fois que l'on fait tourner le programme sur le cas d'étude, aucun modèle stable n'est généré car le programme est surcontraint. Réécrivez les règles de la question 2 de façon à ce qu'elles ne génèrent plus *absurde*, mais un atome `impossible(P, G)` voulant dire que P ne pourra pas assister à la soutenance du groupe G. Attention, il faudra dans vos règles veiller à ce que, si une personne est liée à deux groupes qui soutiennent à la même date, un seul atome "impossible" soit généré.

```
impossible(P, G) :- affectation(G, S, D), in(P, G), indisponible(P, D).
impossible(P, G1) :- affectation(G, S, D), affectation(G1, S1, D), G < G1,
   dans_groupe(P, G), dans_groupe(P, G1).
```

La première règle a été pas mal faite. Dans la seconde règle, personne n'a eu l'idée d'utiliser le $<$ pour la rupture de symétrie. Ca vous a souvent obligé à faire des règles du type "si $(P, G2)$ n'est pas impossible, alors $(P, G1)$ est impossible." mais là encore, si vous ne rompez pas la symétrie à un moment, ça ne mènera à aucun modèle stable.

Epilogue *On pourrait ensuite (ce qui n'est pas demandé ici) utiliser un opérateur d'aggrégation pour compter le nombre d'impossibilités par personne, et borner ce nombre pour éliminer les modèles stables ayant "trop" d'impossibilités, ce qui pourait être fait de la façon suivante en Clingo :*

```
% Pas plus d'une impossibilité par personne
:- lieA(P, G), N = #count{ :  impossible(P, G) }, N > 1.
```

Dans une future version d'Integraal, ceci s'écrira plutôt comme ça, mais ça nécessite tout d'abord l'aboutissement d'un stage de M2.

```
!  :- dans_groupe(P, G), C(P, G) := impossible(P, G), count(C) > 2.
```