Fiche de TP no. 1

Dans ces TPs nous allons apprendre comment utiliser les programmes `Prover9` et `Mace4`. La page web de ces programmes, avec les manuels, se trouve à l'url

http://www.cs.unm.edu/~mccune/mace4/
http://www.cs.unm.edu/~mccune/mace4/manual/2009-11A/

`Prover9` essaye de montrer que ϕ est une conséquence logique de Γ, en utilisant le calcul de la résolution, et un autre calcul, la paramodulation, qui est plus adapté à traiter le raisonnement par égalités.
Pour se servir e cette programme, écrivez sur un terminale

```
prover9 -f nomfichier.in > nomfichier.out
```

où `nomfichier.in` est le fichier qui contient une description de Γ et ϕ, et `nomfichier.out` est le nom du fichier qui contiendra les réponses données par `Prover9`.
`Mace4` essaye de construire un modèle de Γ qui rend ϕ fausse. L'utilisation est similaire :

```
mace4 -f nomfichier.in > nomfichier.out
```

On peut aussi utiliser les deux programmes via un interface graphique, en donnant, depuis un terminal, la commande suivante :

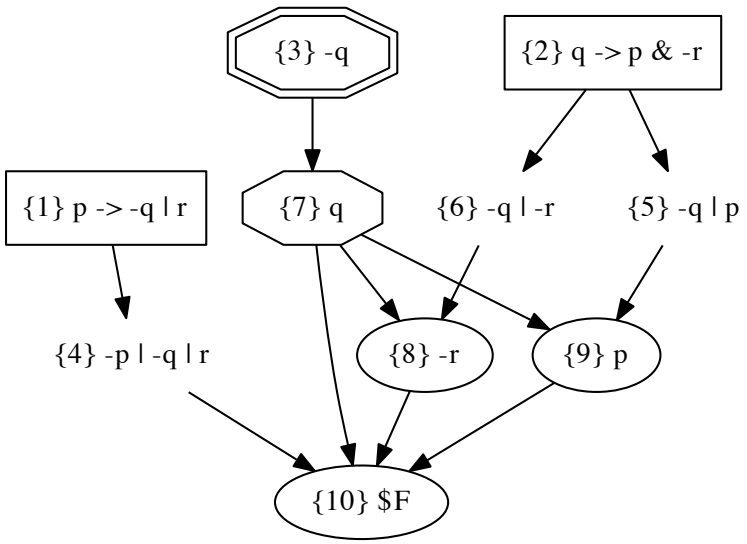
```
/opt/p9m4-v05/prover9-mace4.py
```

Exemple. On souhaite utiliser Prover9 (et la méthode de la coupure) pour montrer que $\{\, p \Rightarrow (\neg q \vee r), q \Rightarrow (p \wedge \neg r) \,\} \models \neg q$. Voici le fichier input pour `Prover9` :

```
formulas(assumptions).
p -> (-q | r).
q -> (p & -r).
end_of_list.

formulas(goals).
-q.
end_of_list.
```

Voici la preuve construite par `Prover9` :



Exercice 1. Via le manuel de `prover9`, découvrez comment on peut représenter les opérateurs logiques et les formules dans un fichier `fic.in`.

Exercice 2. Utilisez `Prover9` (et/ou `Mace4`) pour résoudre l'exercice 4.4 du TD 4 : *prouvez ou infirmez les affirmations suivantes* :

1. $\models p \Rightarrow p$
2. $\models ((p \Rightarrow q) \wedge (q \Rightarrow r)) \Rightarrow (p \Rightarrow r)$
3. $\models ((s \Rightarrow r) \wedge p \wedge \neg r) \Rightarrow \neg r \wedge \neg s \wedge p$
4. $\models [(p \wedge q) \vee (r \wedge q)] \Rightarrow (p \vee r)$
5. $\{q \Rightarrow (\neg q \vee r), q \Rightarrow (p \wedge \neg r)\} \models q \Rightarrow r$
6. $\{q \Rightarrow (\neg q \vee r), q \Rightarrow (p \wedge \neg r)\} \models q \wedge r$
7. $\models (p \wedge (q \vee r)) \Leftrightarrow ((\neg p \Rightarrow r) \wedge (p \wedge q))$.
8. $\models (p \vee (q \wedge r)) \Leftrightarrow ((p \Rightarrow r) \vee (p \wedge q))$.
9. $\{p \Rightarrow q, q \Rightarrow r, p \vee \neg r\} \models p \wedge q \wedge r$.
10. $\{p \Rightarrow q, q \Rightarrow r, p \vee \neg r\} \models (p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$.

Étapes à suivre, pour chaque problème à résoudre de l'exercice qui demande de montrer que $\Gamma \models \phi$:

1. dans fichier `fic.in`, ajoutez entre les mots clés `formulas(assumptions).` et `end_of_list.` les formules dans Γ ;
2. dans le même fichier, ajoutez entre les mots clés `formulas(goals).` et `end_of_list.` la formule ϕ ;
3. démarrez `Prover9` (et/ou `Mace4`) avec le fichier `fic.in`
4. inspectez ensuite le fichier `fic.out`.

Exercice 3. Utilisez `Mace4` pour compter combien de relations d'ordre is y a sur un ensemble de 3 éléments. Étapes à suivre :

1. écrivez, sur papier, ce que veut dire qu'une relation binaire R est une relation d'ordre (réflexivité, transitivité, antisymétricité) ; reformulez ces conditions comme des formules de la logique du premier ordre ;
2. ajoutez au début du fichier `ordre.in` les instructions qui spécifient que l'on souhaite trouver toutes les structures de taille 3 qui satisfont ces formules : `assign(max_models, -1).` et `assign(domain_size,3).` ;
3. dans le fichier `ordre.in`, ajoutez entre les mots clés `formulas(sos).` et `end_of_list.` les formules logiques (du premier ordre) qui décrivent 'être un relation d'ordre' ;
4. démarrez `Mace4` avec le fichier `ordre.in` et inspectez ensuite le fichier `ordre.out` ;
5. affichez les résultats via l'outil `viewmodels.py` qu'on peut télécharger de la page du cours : depuis un terminal, tapez

```
./viewmodels.py ordre.out
```

Repetez tout pour compter combien de relations d'ordre il y a sur un ensemble de 4 éléments.