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.
`Mace4` essaye de montrer que ϕ n'est pas une conséquence logique de Γ en construisant une modèle de Γ (une structure qui valide toute formule dans Γ) qui rend ϕ fausse.
On peut se servir des deux programmes via un interface graphique jointe. Depuis un terminal[1], tapez la commande suivante :

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

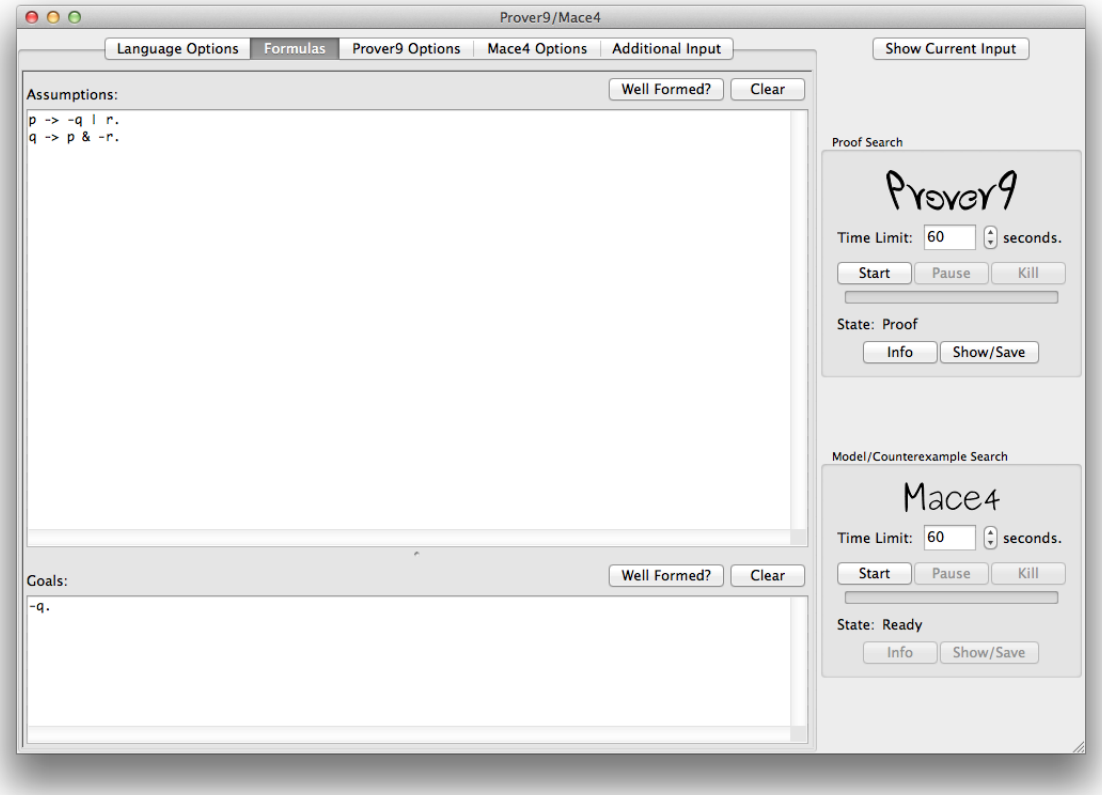
ou bien cette autre :

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

Exemple. On souhaite utiliser la méthode de la coupure via l'outil `Prover9` pour montrer que

$$\{\, p \Rightarrow (\neg q \vee r), q \Rightarrow (p \wedge \neg r) \,\} \models \neg q \,,$$

c'est-à-dire que $\neg q$ est une consequence logique de l'ensemble $\{\, p \Rightarrow (\neg q \vee r), q \Rightarrow (p \wedge \neg r) \,\}$.
Pour ce faire, on recopie (en utilisant a syntaxe de `Prover9`) les formules de Γ dans la fenêtre des assomptions, et la formulas $\neg p$ dans la fenêtre des buts (goals) :

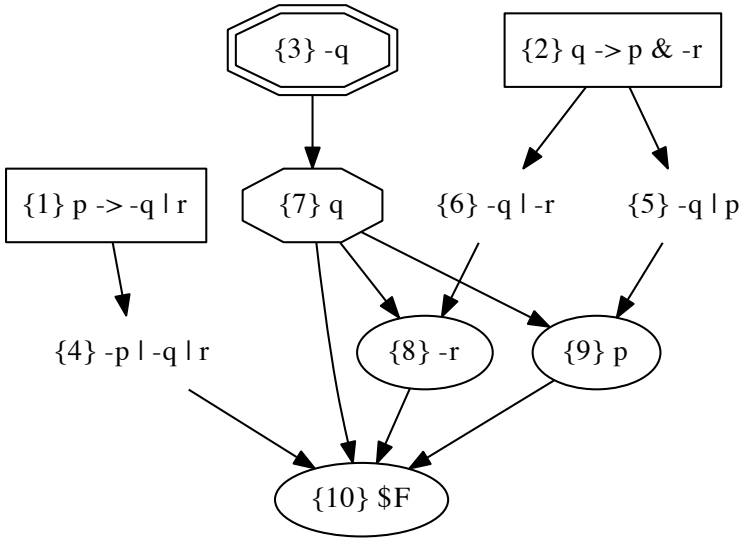


1. Sous Linux, ouvrez l'application `terminal` en faisant une recherche sur le mot clèf « `terminal` ».

Voici la preuve construite par `Prover9` :

```
1 p -> -q | r # label(non_clause).  [assumption].
2 q -> p & -r # label(non_clause).  [assumption].
3 -q # label(non_clause) # label(goal).  [goal].
4 -p | -q | r.  [clausify(1)].
5 -q | p.  [clausify(2)].
6 -q | -r.  [clausify(2)].
7 q.  [deny(3)].
8 -r.  [back_unit_del(6),unit_del(a,7)].
9 p.  [back_unit_del(5),unit_del(a,7)].
10 $F.  [back_unit_del(4),unit_del(a,9),unit_del(b,7),unit_del(c,8)].
```

En utilisant l'outil `gvzify`, on peut transformer cette preuve de façon graphique comme suit :



Exercice 1. Via le manuel de `prover9`, découvrez comment on peut représenter les opérateurs logiques et les formules avec `Prover9` et `Mace4`.

Exercice 2. Utilisez `Prover9` (et/ou `Mace4`) pour résoudre cet exercice du TD 5 : *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. ajoutez dans la fenêtre des assomptions les formules dans Γ ;
2. ajoutez la formule ϕ dans la fenêtre des buts ;
3. démarrez `Prover9` (et/ou `Mace4`) ;
4. inspectez ensuite la preuve produite (ou le modèle produit).

Exercice 3. Utilisez `Mace4` pour compter combien de relations d'ordre il y a sur un ensemble de 3 éléments. Pouse ce faire, on utilisera un langage du premier ordre sans symboles de fonctions, et avec les deux symboles de prédicat $(less, 2)$ et $(=, 2)$.
Étapes à suivre :

1. écrivez, sur papier, ce que veut dire qu'une relation binaire *less* est une relation d'ordre (réflexivité, transitivité, antisymétricité) ; écrivez ces conditions comme des formules de la logique du premier ordre sur le langage donné ;
2. sélectionnez, dans la fenêtre des options de `Mace4`, les options indiquant que l'on cherche tous les modèles sur un ensemble de taille 3 : `max_models=-1` et `domain_size=3` ;
3. ajoutez dans la fenetre des assomptions les formules logiques (du premier ordre) qui décrivent l'« être un relation d'ordre » ;
4. démarrez `Mace4` et inspectez ensuite le résultat ; dessinez le diagramme de Hasse de chacun des ordres ; sauvez ce résultat dans un fichier nommé `ordre.out` ;
5. affichez les résultats via l'outil `viewmodels.py` qu'on peut télécharger de la page du cours ; pour ce faire, posez vous dans un répertoire conteant les fichier les fichiers `viewmodels.py` et `ordre.out` et, tapez depuis un terminal,

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

   À defaut, tapez

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

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