ATP partie II
Exercices

1. Nous allons tout d'abord appliquer manuellement les règles vues au cours pour vérifier qu'elles sont suffisantes, du moins sur un exemple. Pour considérer une cas simple, nous allons prendre une instance d'un problème combinatoire bien connu, le "pigeon hole" : 2 trous ne sont pas suffisants pour y placer 3 pigeons (un pigeon par trou au maximum !).
   (a) Ouvrez le fichier atp2_1.key et vérifiez qu'on y exprime bien qu'il n'est pas possible de placer 2 pigeons dans un unique trou, lorsqu'un trou ne peut contenir qu'un seul pigeon !
   (b) Chargez le fichier dans KeY et vérifiez **manuellement** que les règles vues au cours sont suffisantes pour justifier cette formule.
   Pour sélectionner une règle, il faut d'abord s'assurer que *Options -> Right click for macros* **n'est pas activé**. Il suffit alors d'utiliser le bouton de droite de la souris lorsque le curseur est placé sur un connecteur dans le panneau de droite de KeY pour choisir la règle.
   Notez qu'il faut bien choisir les règles pour ne pas obtenir une preuve trop compliquée. En général il est par exemple préférable d'utiliser les règles sans branchement en premier. Les règles vues au cours sont les suivantes.
   - *close* (en sélectionnant une formule présente des deux côtés)
   - *notRight*
   - *notLeft*
   - *orRight*
   - *andLeft*
   - *orLeft*
   - *andRight*
2. Rechargez le fichier précédent et cette fois-ci faites faire la preuve par l'outil. Est-ce que la preuve est plus courte ou plus longue ? Est-elle plus complexe ou plus simple ? Vous pouvez observez les règles utilisées, nous en reparlerons bientôt.
3. Pour montrer comment l'impossibilité de dériver une justification peut nous permettre de déterminer un contre-exemple, nous allons considérer le problème de placer deux pigeons dans deux trous. Ce problème a évidemment une solution et nous allons la trouver à l'aide de KeY !
   (a) Ouvrez le fichier atp2_3.key et vérifiez qu'on y exprime bien qu'il n'est pas possible de placer deux pigeons dans deux trous. Cette formule n'est pas valide (elle n'est pas toujours vraie), elle ne sera donc pas justifiable. De plus, un contre-exemple pour cette formule nous donnera une façon de placer deux pigeons dans deux trous !
   (b) Faites faire la recherche d'une justification par l'outil et vérifiez qu'il ne réussi pas à fermer les branches de la dérivation.
   (c) À partir d'une feuille ouverte de la dérivation précédente, déterminer les positions possibles pour les pigeons dans les trous. En principe il y a deux solutions et la dérivation devrait bien contenir deux feuilles y correspondant.

4. Comme KeY permet de traiter des programmes Java, il doit offrir des règles pour tous les types de base, comme par exemple les entiers. C'est ce que nous allons maintenant voir.
   (a) Ouvrez le fichier atp2_4.key et observez qu'il contient une formule logique affirmant une égalité fondamentale de l'arithmétique. Vérifiez si KeY peut justifier cette formule de façon automatique. Si c'est le cas, observez au moins les premières étapes pour tenter de comprendre les transformations effectuées par l'outil.
   (b) En sélectionnant vous-même les règles à appliquer, il devrait être possible d'établir cette égalité rapidement. Essayez de voir s'il ne serait pas possible d'utiliser la commutativité de l'addition pour arriver rapidement à une dérivation complète.
5. Modifiez le fichier précédent pour cette fois-ci vérifier l'égalité $x + y = y + x$. Répondez aux mêmes questions que pour le problème précédent.
6. Ne vérifier que des formules générales dont les variables peuvent prendre n'importe quelles valeurs, n'est pas suffisant pour justifier un programme. Un programme peut affecter des variables à des valeurs spécifiques, il est donc important de pouvoir représenter les affectations. En fait, KeY permet d'indiquer que certaines variables sont affectées à l'aide de *mises-à-jours* (*updates*). Une mise-à-jour est une suite d'affectations séparées par des `||` et placée entre accolades. La mise-à-jour apparaît toujours directement devant la formule à laquelle elle s'applique.
   (a) Ouvrez le fichier atp2_6.key et observez bien comment la variable est affectée dans une mise-à-jour. Faites faire la vérification automatiquement par l'outil et vérifiez que les règles utilisées correspondent bien à l'intuition qu'une mise-à-jour est une affectation.
   (b) Vérifiez que si vous remplacez la formule par quelque chose de faux, comme par exemple $(x = 1)$, l'outil ne trouvera évidemment pas de justification.
   (c) Il est possible d'affecter plusieurs variables simultanément tout simplement en les ajoutant dans la mise-à-jour (séparées par des `||`). Modifiez l'exemple précédent en vérifiant qu'on a bien $x + y = 0$ lorsque x et y sont toutes deux affectées à 0.
   (d) Finalement, ouvrez le fichier atp2_6_1.key et observez bien comment une mise-à-jour permet de remplacer des variables par n'importe quelles expressions. Quelle est la signification de cette formule (après le remplacement) ? Est-ce que l'outil peut justifier cet énoncé ? Est-ce conforme avec votre intuition ? S'il vous reste du temps, vous pouvez tenter d'utiliser la commutativité de l'addition manuellement. Est-ce que ceci permet de trouver une justification ? Si oui, pouvez-vous comprendre la signification des règles utilisées ?