TP $\mathbb{K}$ - Sémantique des langages de programmation

Le but de ce TP est d'apprendre à se servir du *semantical framework* $\mathbb{K}$ (https://kframework.org/), qui permet de spécifier des sémantiques de façon formelle et de pouvoir les exécuter. Dans ce TP, nous définissons la sémantique d'un petit langage impératif nommé IMP.

De manière générale, pour formaliser la sémantique d'un langage de programmation avec $\mathbb{K}$, il faut définir :

— la syntaxe, par l'écriture d'une grammaire BNF
— la sémantique, à l'aide d'une configuration et de règles de réécriture

1 Avant de commencer

Avant de commencer ce TP, vous devez accomplir les missions suivantes, dans l'ordre :

1. Télécharger $\mathbb{K}$ via https://github.com/runtimeverification/k/releases.
2. Télécharger les fichiers `imp.k`, `42.imp`, `simple.imp` et `euclide.imp` dans un même dossier.
3. Compiler la sémantique du langage IMP à l'aide de la commande `kompile imp.k`.

   Appelez-moi si vous obtenez une erreur ou un warning.
4. Exécuter le programme `42.imp` à l'aide de la commande `krun 42.imp`.

   Appelez-moi si vous obtenez autre chose que la configuration `<k> 21 + 21 ~> . </k>`, qui correspond à l'état initial de votre programme car vous n'avez pas encore défini la sémantique du langage IMP : exécuter un programme est donc équivalent à ne rien faire.

2 Travail pour cette séance

L'objectif de cette séance est de formaliser la sémantique d'un petit langage impératif composé d'expressions arithmétiques et booléennes, ainsi que d'instructions simples.

2.1 Expressions arithmétiques

1. Ajouter la règle de réécriture suivante dans votre fichier : `rule I1 + I2 => I1 +Int I2`.

   Voici quelques informations complémentaires :
   — L'opération `+Int` correspond à l'addition sur les entiers de la bibliothèque standard de $\mathbb{K}$ importée grâce à la commande `imports DOMAINS`.
   — Les variables utilisées dans les règles de réécriture doivent toujours commencer par une lettre en majuscule.
2. Constater que vous pouvez exécuter le programme `42.imp`.
3. Constater que vous ne pouvez pas parser le programme `2 * 3 + 5`.
4. Compléter la syntaxe des expressions arithmétiques dans votre fichier $\mathbb{K}$.
5. Compléter la sémantique[1] des expressions arithmétiques dans votre fichier $\mathbb{K}$.

1. Nous pouvons presque dire "à petits pas".

6. Que se passe t-il avec l'expression `2 + 3 + 5` ? Comment résoudre le problème ?
7. Écrire des expressions arithmétiques valides dans des fichiers séparés afin de tester votre sémantique.

2.2 Expressions booléennes

1. Compléter votre sémantique pour ajouter les expressions booléennes, en respectant le comportement coupe-circuit des opérateurs `and` et `or`.
2. Tester votre sémantique, par exemple avec l'expression booléenne `2 <= 1 and 0 / 0 <= 3 or true`.

2.3 IMP

1. Constater que vous ne pouvez pas *parser* le fichier `simple.imp`.
2. Étendre la syntaxe de IMP en ajoutant l'affectation, la séquence, la construction `if-then-else`, la boucle `while` et la notion de bloc.
3. Étendre la configuration pour définir la sémantique des instructions.
   La structure des configurations est donnée à l'aide d'une syntaxe à la XML.
4. Compléter votre sémantique avec des règles de réécriture.

   Par exemple, la règle de réécriture suivante signifie que, comme la variable `X` est associée à la valeur `N`, on peut évaluer cette variable en la remplaçant par `N` :
   ```
   rule <k> X:Id ~> S </k> <state> (X |-> N) M </state>
   =>   <k> N    ~> S </k> <state> (X |-> N) M </state>.
   ```
5. Tester votre sémantique avec les fichiers `simple.imp` et `euclide.imp`.
   Vous serez certainement amené.e à adapter la syntaxe de ces fichiers.
6. Amusez-vous à écrire des programmes dans le langage que vous venez de formaliser, puis exécutez-les.

3 Travail bonus

Vous avez la possibilité de me rendre l'un des exercices suivants pour bénéficier de points bonus.

3.1 MiniML

1. Ouvrir un nouveau fichier et définir la syntaxe et la sémantique d'un petit langage fonctionnel, nommé ici MiniML.

3.2 MyLanguage

1. Ouvrir un nouveau fichier et définir la syntaxe et la sémantique d'un langage de votre choix.

4 Travail pour la semaine prochaine en guise de DM

Afin d'approfondir votre compréhension de ce que vous avez appris durant cette séance, je vous demande de me rendre, via eCampus avant le prochain cours, un unique exercice dans l'une des catégories ci-dessous :

- **Définir un interpréteur**
  - pour une petite calculatrice en notation polonaise inversée
  - pour une petite calculatrice en Scheme ou MiniML ou λ-calcul.

- **Définir un type-checker**
  - pour la logique propositionnelle.
  - pour une petite calculatrice.
  - pour un λ-calcul simplement typé avec annotations (à la Church).

- **Définir un compilateur/traducteur**
  - Traduction d'une expression arithmétique/booléenne en notation infixe en une expression en notation préfixe.
  - Traduction d'une mini calculatrice vers assembleur.
  - Traduction IMP vers Arduino.

- **Résoudre un petite problème mathématique**
  - Prendre en entrée une équation du second degré et renvoyer la solution ou dire qu'il n'y a pas de solution.
  - Prendre en entrée deux nombres et renvoyer le PGCD.
  - Renvoyer si un triangle est rectangle à l'aide du théorème de Pythagore.
  - Implémenter le théorème de Thalès.

Vous pouvez également me proposer un exercice de votre choix.

Consigne :
A) Écrire une BNF pour un langage modélisant votre problème.
B) Écrire la configuration adéquate.
C) Écrire les règles de réécriture pour résoudre le problème.
D) Écrire aux moins 10 tests.
E) Écrire un petit rapport (1 ou 2 pages) expliquant ce que vous avez fait et justifiant vos choix. Votre rapport doit être structuré ainsi :
   1. Quelques lignes pour rappeler le sujet que vous avez choisi.
   2. Une première partie présentant la syntaxe et la sémantique de ce que vous avez formalisé en $\mathbb{K}$.

      Vous présenterez la grammaire dans une syntaxe standard et justifierez la configuration que vous avez considérée.
   3. Une seconde partie présentant les tests que vous avez réalisés, ainsi que justifiant leur pertinence.
   4. Vous pouvez également faire une partie dédiée aux problèmes rencontrés ou bien les incorporés dans les deux parties précédentes.
   5. Une conclusion récapitulative.