TP n° 2

Paradigmes et interprétation
Licence Informatique
Université Côte d'Azur

Dans ce TP, on modifie l'interpréteur avec fonctions et environnement : repartez du fichier `fun.rkt`.

Soustraction

Ajoutez la soustraction au langage. Comme vous ajoutez une nouvelle expression au langage, il faut modifier le type `Exp` et ajouter une variante. Il faudra ensuite modifier les fonctions `parse`, `interp` et `subst` pour prendre en compte cette modification.

```
(test (interp-expr `{- 1 2} empty) -1)
```

Addition généralisée

Généralisez l'addition pour qu'elle accepte un nombre quelconque d'opérandes. Il vous faut adapter la variante associée du type `Exp` et modifier les fonctions `parse`, `interp` et `subst`.

```
(test (interp-expr `{+} empty) 0)
(test (interp-expr `{+ 1} empty) 1)
(test (interp-expr `{+ 1 2} empty) 3)
(test (interp-expr `{+ 1 2 3 4 5} empty) 15)
```

Paramètres des fonctions

Dans la plupart des langages, on peut définir des fonctions qui acceptent un nombre quelconque de paramètres. Pour l'instant, l'interpréteur ne supporte que les fonctions à un paramètre. Vous allez apporter les modifications nécessaires à l'interpréteur pour pouvoir définir et utiliser des fonctions à nombre de paramètres quelconque (zéro ou plus).

Par exemple, `{define {aire l h} {* l h}}` définit une fonction à deux paramètres. De même, `{define {zero} 0}` définit une fonction sans paramètre. Des appels associés à ces fonctions sont `{aire 2 3}` et `{zero}`.

Pour implémenter ces changements, vous allez devoir modifier les types `Exp` et `FunDef` ainsi que les fonctions `parse`, `parse-fundef`, `interp` et `subst`.

Un nouveau type de comportement anormal peut se produire : l'application d'une fonction avec le mauvais nombre d'arguments. Vous devez détecter ces cas et lancer une erreur contenant la chaine `"wrong arity"`. Vous pouvez utiliser les tests suivants pour vérifier que vos modifications sont correctes.

```
(test (interp-expr `{f 1 2}
                   (list `{define {f x y} {+ x y}}))
      3)
(test (interp-expr `{+ {f} {f}}
                   (list `{define {f} 5}))
      10)
(test/exn (interp-expr `{f 1}
                       (list `{define {f x y} {+ x y}}))
          "wrong arity")
```

Bonus. Une fonction est mal définie quand deux noms de paramètres sont identiques. Modifiez la fonction `parse-fundef` pour qu'elle détecte ce problème et renvoie alors une erreur `"bad syntax"`.