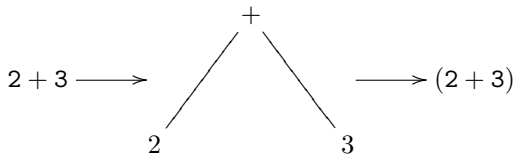
Compilation — TD 3

J. Chroboczek et P. Letouzey

18 Octobre 2006

Vous savez déjà qu'un *parseur* est un module qui traduit une syntaxe concrète, représentée comme une suite de caractères, en une syntaxe abstraite, représentée par un arbre. Un *pretty-printer* est l'inverse d'un parseur : c'est un module qui, étant donné un arbre de syntaxe abstraite, produit une représentation syntaxique concrète de cet arbre.

Le pretty-printer est un *inverse à droite* du parseur : en partant d'un arbre de syntaxe abstraite, appliquer le pretty-printer puis le parseur nous ramène à l'arbre de départ. Il n'est cependant normalement pas un *inverse à gauche* du parseur : appliquer le parseur à un programme en syntaxe concrète puis le pretty-printer à l'arbre résultant donne un programme certes équivalent mais pas nécessairement identique au programme de départ.



Le but de ce TP est d'écrire un pretty-printer pour le langage CTigre utilisé dans le projet. Cet exercice vous permettra de vous familiariser avec la syntaxe abstraite utilisée ; de plus, vous pourrez vous servir du pretty-printer que vous aurez écrit dans votre projet.

0. Dans un module `PrettyPrint`, écrivez une fonction `pprint` de type

$$\texttt{Absyntax.exp} \rightarrow \texttt{unit}$$

qui affiche « `...` » sur le terminal. Modifiez le `Makefile` fourni pour qu'il compile votre module, et modifiez la fonction `main` du fichier `main.ml` pour que le programme final lance votre fonction lorsqu'il est invoqué avec l'option `-ppast`.

Compilez et testez votre programme sur un programme CTigre.

1. Modifiez la fonction `pprint` pour qu'elle imprime correctement les entiers, les variables, les chaînes, les expressions séquentielles et les applications. Testez-la sur le programme suivant, en vérifiant que votre programme peut être appliqué à son résultat.

```
print("Je vais afficher 2.");
printint(2)
```

2. Étendez votre fonction aux opérandes en paranthésant systématiquement les expressions. Testez le programme résultant sur le programme suivant :

```
print("1 + 2 font ");
printint(1 + 2)
```

Étendez ensuite votre programme aux conditionnelles, et testez-le sur un exemple que vous choisirez.

3. Écrivez une fonction `pprint_dec` qui « pretty-printe » une déclaration. Étendez votre programme aux instructions *let*, et aux affectations. Testez-le résultat sur la factorielle récursive donnée dans le projet ainsi que sur un programme impératif élémentaire.

4. Étendez votre pretty-printer à la totalité du langage CTigre.

5. Votre pretty-printer est peu économe en parenthèses. Par exemple, il affiche « `2 + 3 + 4` » comme « `((2 + 3) + 4)` » et « `2 + 3 * 5` » comme « `((2 + 3) * 5)` ». Modifiez-le pour qu'il évite d'afficher des parenthèses lorsque ce n'est pas nécessaire. Il faudra bien-sûr prendre en compte les règles d'associativité à gauche ainsi que les priorités relatives des opérateurs CTigre.