Analyse Syntaxique et Compilation, TP n° 3 : Étude d'un compilateur jouet

On se propose de travailler sur un langage très simple d'expressions logiques et arithmétiques.

Les types qu'on rencontrera sont `int` et `bool`. Les spécifications de la syntaxe du langage découlent des analyseurs lexicaux et syntaxiques décrits dans les fichiers que vous trouverez dans le répertoire `/ens/meyera/Public/Compil/tp3` :

- `expsyntax.ml` : la syntaxe abstraite du langage ;
- `explexer.mll` : le code `ocamllex` d'un analyseur lexical du langage ;
- `expparser.mly` : le code `menhir` d'un analyseur syntaxique (parser) du langage ;
- `exptyper.ml` : le code de l'analyseur de type (**à compléter**) ;
- `expeval.ml` : le code de l'évaluateur (**à compléter**) ;
- `main.ml` : le programme principal (lecture des paramètres, appels, affichage des résultats) ;
- `ex_00.exp` : une expression du langage.

Exercice :

1. Recopier le répertoire `tp3` entier dans votre espace de travail.
2. Observer l'analyseur et comprendre son fonctionnement.
3. Compiler et utiliser l'analyseur sous `ocaml`. Dans l'état actuel, l'analyseur de type renvoie toujours la valeur `bool`, et l'évaluateur toujours la valeur `true`.
4. Ecrire les règles de typage et compléter le fichier `exptyper.ml`. On disposera alors d'une fonction

   ```
   typer_exp: (string * texp) list -> exp -> texp
   ```

   qui étant donné un environnement de typage et une expression retourne le type de cette expression.
5. Écrire les règles d'évaluation et les programmer dans un fichier `expeval.ml`. On disposera alors d'une fonction

   ```
   eval_exp: (string * vexp) list -> exp -> vexp
   ```

   qui étant donné un environnement de valeurs et une expression close retourne la valeur de cette expression.