TP n°2-3
Parsing

Florent Bouchez, Christoph Lauter
prenom.nom@ens-lyon.fr

14-28 février 2007

1 Parsing

Maintenant que vous avez bien terminé l'analyse lexicale (n'est-ce pas ?), il est temps de passer à l'analyse grammaticale.

Récupérez les nouveaux fichiers du TD, après avoir fait une sauvegarde du travail déjà fait pour éviter les mauvaises surprises :

`/home/fbouchez/compilation/TP2_parser`

Il y a un certain nombre de fichiers nouveaux, d'autres modifiés, et d'autres à supprimer...

- Nouveaux fichiers :
  - `ast.{ml,mli}` : fixent la structure de l'arbre de syntaxe abstraite que vous allez devoir produire. Vous n'avez qu'à vous préoccuper du fichier d'interface. Oubliez la fonction `ast_to_proc` qui est inutile pour ce TP ;
  - `parser.mly` : le fichier que vous devrez compléter.
- Fichiers modifiés :
  - `error_handler.{ml,mli}` : ajout d'une structure de position ;
  - `Makefile` : et oui puisque notamment les fichiers de parsing doivent être générés à partir de `parser.mly` ;
  - `main.ml` : qui permet de tester à la fois le *lexing* et le *parsing* maintenant.
- Fichier disparu (!) : `parser.mli` puisqu'il va maintenant être généré par `ocamlyacc`.

2 Au boulot (encore)

Vous allez à nouveau remplir les endroits où les TD-men ont, malencontreusement, effacé les parties les plus importantes du code. Vous avez de la chance, on a trouvé où c'était et on les a indiquées par des "...".

1. Finissez de remplir la liste des *tokens* en rajoutant `INT`, `STR` et `WORD`.
2. Définissez les priorités des *token* qui sont des opérateurs.

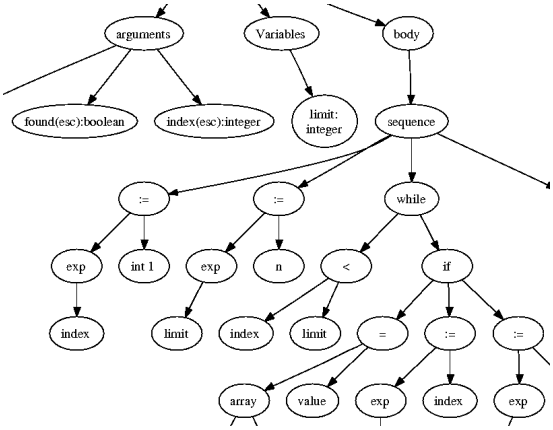

FIG. 1 – Partie de l'AST de `example.pas`

3. Réécrivez toutes les règles grammaticales.[1]

Pour l'instant, ne vous préoccupez pas des champs de position dans les nœuds de l'AST, utilisez `No_pos` pour les remplir.

Une fois que c'est fini, vous devrier avoir un programme capable de vous faire un bel AST. Avec l'option `-dot` et les outils `graphviz`, vous devriez obtenir quelque chose dans le goût de la figure 1.

3 Un peu de boulot en plus

Facultatif, mais tellement indispensable...

1. Maintenant utilisez les champs de position pour localiser les structures grammaticales dans le code initial (créez vous les fonctions auxiliaires nécessaires dans `error_handler`).
2. Ajoutez une gestion des erreurs par l'utilisation du *token* spécial `error`. Faites alors des (chouettes) fonctions dans `error_handler` pour renvoyer des (cool) messages d'erreurs avec les positions ; pensez au petit programmeur de pascal- qui vous bénira si vous le faites – ou vous maudira si vous ne le faites pas.

[1] Attention, bien que la plus courte, cette question est la plus longue.