Projet de compilation - Construction de l'arbre abstrait

1 Objectif

L'objectif de ce TP est de construire un arbre abstrait correspondant au programme en langage source. Cet arbre abstrait sera utilisé lors des étapes suivantes de la compilation, en particulier lors de la production de code.

2 Arbre abstrait

L'arbre de dérivation produit par l'analyse syntaxique possède de nombreux nœuds superflus, qui ne véhiculent pas d'information.
De plus, la mise au point d'une grammaire (élimination de l'ambiguïté, élimination de la récursivité à gauche, factorisation) nécessite souvent l'introduction de règles dont le seul but est de simplifier l'analyse syntaxique.
Un arbre abstrait constitue une interface plus naturelle entre l'analyse syntaxique et l'analyse sémantique, elle ne garde de la structure syntaxique que les parties nécessaires à l'analyse sémantique et à la production de code.
On trouvera dans la figure **??** un arbre de dérivation et l'arbre abstrait lui correspondant.

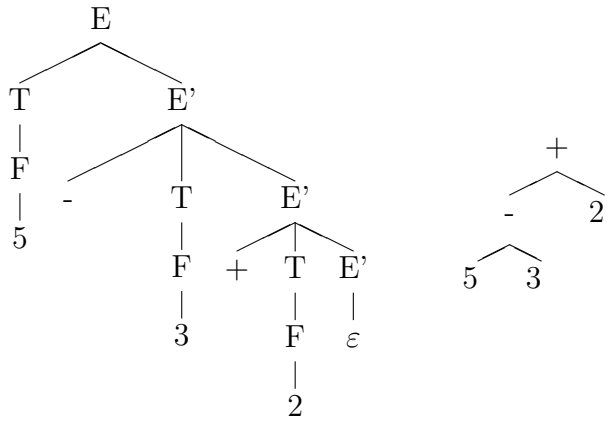

FIGURE 1 – Un arbre de dérivation et l'arbre abstrait correspondant

3 Construction de l'arbre abstrait

L'arbre abstrait est construit lors de l'analyse syntaxique, en ajoutant à toute fonction correspondant à un non terminal de votre grammaire du code destiné à la construction de l'arbre abstrait.
L'idée est de construire pour certains noœuds de l'arbre de dérivation un nœud de l'arbre abstrait (appelé nœud courant) et de le retourner sous forme d'attribut synthétisé. Etant donné la règle $A \rightarrow BC$, les fils du nœud courant, construit par la fonction `A()` sont les nœuds renvoyés par les appels aux fonctions `B()` et `C()`.
L'idée est illustrée ci-dessous sur la fonction correspondant à la règle de grammaire suivante :

```
instructionSi -> SI expression ALORS instructionBloc optSinon
```

```
n_instr *instructionSi(void)
{
  n_instr *$$ = NULL;
  n_exp *$2 = NULL;
  n_instr *$4 = NULL;
  n_instr *$5 = NULL;
  char *fct = "instructionSi";
  affiche_balise_ouvrante(fct, trace_xml);
  consommer( SI );
  $2 = expression();
  consommer( ALORS );
  $4 = instructionBloc();
  $5 = optSinon();
  $$ = cree_n_instr_si($2, $4, $5);
  affiche_balise_fermante(fct, trace_xml);
  return $$;
}
```

Les variables locales `$2`, `$4` et `$5` vont acceuillir les nœuds renvoyés par les appels aux fonctions `expression()`, `instructionBloc` et `optSinon()`. Ces nœuds vont constituer les fils du nœud courant, appelé `$$`. Cette opération est effectuée par l'appel `cree_n_instr_si($2, $4, $5)`. Le nœud ainsi créé (stocké dans la variable `$$`) constitue la valeur de retour de la fonction `instructionSi()`.
Les nœuds de l'arbre abstrait renvoyés par les fonctions correspondent à des attributs synthétisés. Ils doivent donc être construits à la fin de la fonction. Cependant, pour implémenter un arbre abstrait associatif à gauche pour les expressions, vous aurez besoin d'attributs hérités, qui prennent la forme de paramètres passés aux fonctions.
Les différents types de nœuds pouvant être renvoyés par les fonctions de votre analyseur ainsi que leurs constructeurs ont été définis dans les fichiers `syntabs.h` et `syntabs.c` disponibles sur le site du cours.
Vous pourrez les utiliser tels quels ou vous en inspirer pour écrire vos propres types et constructeurs.