TP2 Compilation
Analyse syntaxique descendante

Damien Crémilleux, Tom Demulier–Chevret

INSA de Rennes
4INFO, LSR-B

1 Compte-rendu

1.1 Grammaire

1.1.1 Nouvelle grammaire utilisée

La grammaire 1.2 a été réécrite sous une forme équivalente qui ne comporte plus d'expression régulière.

```
1. <Expr>           —> <Termb> <SuiteExpr>
2. <SuiteExpr>      —> ''ou'' <Expr> | ϵ
3. <Termb>          —> <Facteurb> <SuiteTermb>
4. <SuiteTermb>     —> ''et'' <Termb> | ϵ
5. <Facteurb>       —> <Relation> | ''('' <Expr> '')'' |
                       ''si'' <Expr> ''alors'' <Expr> ''
                          sinon'' <Expr> ''fsi''
6. <Relation>       —> <Ident> <Op> <Ident>
7. <Op>             —> ''='' | ''<>'' | ''<'' |
                       ''>'' | ''>='' | '<=''
8. <Ident>          —> <Lettre> <SuiteIdentEt>
9. <SuiteIndentEt>  —> <SuiteIndent> <SuiteIndentEt> | ϵ
10.<SuiteIdent>     —> <Lettre> | <Chiffre>
```

1.1.2 Preuve de la propriété LL(1)

Les règles 1, 3, 6 et 8 de la grammaire de comporte pas de ou, il n'y a donc pas de vérification à faire les concernant.

Ce n'est pas le cas des règles 2, 4, 5, 7. Ident est considéré comme un terminal, il n'y a donc pas de vérification à effectuer pour les règles 9 et 10.

Vérification de la règle 2

$$premier(\text{``ou''}\ \langle Expr\rangle) \cap premier(\epsilon) = \{\text{``ou''}\} \cap \{\epsilon\} = \emptyset$$

$$null(\langle SuiteExpr\rangle) = null\ (\text{``ou''}\ \langle Expr\rangle) \vee null(\epsilon) = vrai$$
$$premier(\langle SuiteExpr\rangle) = premier(\text{``ou''}\ \langle Expr\rangle) \cup premier(\epsilon) = \{\text{``ou''}\}$$

suivant(<SuiteExpr>) = suivant(<Expr>)
suivant(<Expr>) = suivant(<SuiteExpr>)

suiv_Expr	{}	∅	∅
suiv_SuiteExpr	{}	∅	∅

Par Kleene, premier(<SuiteExpr>) ∩ suivant(<SuiteExpr>) = ∅

null(``ou'' <Expr>) = faux
null(ϵ) = vrai

La règle 2 vérifie donc bien les propriétés d'une grammaire LL(1).

Vérification de la règle 4

premier(``et'' <Termb>) ∩ premier($\epsilon$) = {``et''} ∩ {ϵ} = ∅

null(<SuiteTermb>) = null (``et'' <Termb>) ∨ null(ϵ) = vrai
premier(<SuiteTermb) = premier(``et'' <Termb>) ∪ premier($\epsilon$) = {``et''}
suivant(<SuiteTermb>) = suivant(<Termb>)
suivant(<Termb>) = suivant(<SuiteTermb>)

suiv_Termb	{}	∅	∅
suiv_SuiteTermb	{}	∅	∅

Par Kleene, premier(<SuiteTermb>) ∩ suivant(<SuiteTermb>) = ∅

null(``et'' <Termb>) = faux
null(ϵ) = vrai

La règle 4 vérifie donc bien les propriétés d'une grammaire LL(1).

Vérification de la règle 5

premier(<Relation>) = premier(<Ident>) = {``ident''}
premier(``('' <Expr> '')'') = {``(''}
premier(``si'' <Expr> ``alors'' <Expr> ``sinon'' <Expr> ``fsi'') = {``si''}
{``ident''} ∩ {``(''} ∩ {``si''} = ∅

null(<Facteurb>) = faux

La règle 5 vérifie donc bien les propriétés d'une grammaire LL(1).

Vérification de la règle 7

premier(``='') ∩ premier(``<>'') ∩ premier(``<'') ∩ premier(``>'') ∩ premier(``<'') ∩ premier(``>='') ∩ premier(``<='') = ∅

null(<Op>) = faux

La règle 7 vérifie donc bien les propriétés d'une grammaire LL(1).

1.2 Questions de compréhension du TP

Question 2.1 Les commentaires sont éliminés lors de l'analyse lexicale, ils n'ont donc pas besoin d'être inclus dans la grammaire définissant les constructeurs du langage.

Question 2.2 L'implémentation réalisée repose sur la construction d'un arbre. Cette structure de donnée permet de mémoriser les éléments à compter, à l'image d'un automate à pile.

Question 2.3 L'intérêt d'utiliser une grammaire LL(1) est de pouvoir effectuer une analyse descendante complètement déterministe. En effet la pré-lecture de la prochaine unité lexicale suffit à garantir l'unicité de la règle de dérivation.

Question 2.4 L'intérêt d'utiliser un arbre abstrait réside dans le fait de manipuler un objet beaucoup plus léger, qui ne contient plus que la structure profonde indispensable.

2 Sources

Les sources OCaml sont disponibles dans le dossier Sources. Le fichier *ulex.ml* contient la déclaration de type des unités lexicales, tandis que *lexer.mll* décrit l'analyseur lexical. L'analyse syntaxique descendante est réalisée dans le fichier *anasynt.ml*.

3 Tests