TP3 Compilation
Analyse syntaxique ascendante, ocamllex et ocamlyacc

Damien Crémilleux, Tom Demulier–Chevret

INSA de Rennes
4INFO, LSR-B

1 Compte-rendu

1.1 Table SLR

État	+	<	and	(	)	<-	Ident	Inst	Expr
0							d1		
1						d2			
2				d4			d5		3
3	d6	d7	d8						
4				d4			d5		9
5	r14	r14	r14		r14				
6				d4			d5		10
7				d4			d5		11
8				d4			d5		12
9	d6	d7	d8		d13				
10	d6/r10	d7/r10	d8/r10		r10				
11	d6/r11	d7/r11	d8/r11		r11				
12	d6/r12	d7/r12	d8/r12		r12				
13	r13	r13	r13		r13				

1.1.1 Résolutions pour les éventuels conflits

On trouve 9 conflits dans la table SLR générés par 3 tokens : 6 pour les conflits de prédominance entre deux tokens différents (ex : a and b inf c) et 3 pour les conflits de réductions entre même tokens (ex : a + b + c). L'ordre de préfominance gardé correspond à celui donné par la grammaire : AND > INF > PLUS. De plus nous avons choisis un décalage à droite pour les conflits entre même tokens.

1.2 Questions de compréhension du TP

Question 3.1 Un crible filtre la suite des lexèmes et génère les identifiants. En utilisant ocamllex, on peut générer une hashtable pour stocker les identifiants

et limiter ainsi le nombre de transitions (en effet ocamllex est limité à 32767 transitions).

Question 3.2 Écrire une grammaire sous forme LR plutôt qur LL comporte plusieurs avantage :
- La classe de grammaire couverte par LR est plus large.
- La détection d'erreur est faire au plus tôt.
- On peut traiter des grammaires ambiguës.

Question 3.3 Une colone vide dans une table SLR correspond à un token inutilisé.

2 Sources

Cf dossier source.

3 Tests

3.1 Tests Positifs

```
begin int a,bool b, int c; b <- a < c end

Anasynt.arbre_bloc =
Anasynt.Node_bloc
 ([Anasynt.Node_decl_int "a"; Anasynt.Node_decl_bool "b";
   Anasynt.Node_decl_int "c"],
 [Anasynt.Node_inst ("b",
   Anasynt.Node_inf (Anasynt.Leaf_ident "a", Anasynt.
       Leaf_ident "c"))])
```

```
begin
int a,
int c;
a <- c;
c <- a + c
end

Anasynt.arbre_bloc =
Anasynt.Node_bloc ([Anasynt.Node_decl_int "a"; Anasynt.
    Node_decl_int "c"],
 [Anasynt.Node_inst ("a", Anasynt.Leaf_ident "c");
  Anasynt.Node_inst ("c",
   Anasynt.Node_plus (Anasynt.Leaf_ident "a", Anasynt.
       Leaf_ident "c"))])

begin
```

```
int a,
int c;
a <- c;
c <- a + c;
begin
bool b;
b <- b and b
end
end

Anasynt.arbre_bloc =
Anasynt.Node_bloc ([Anasynt.Node_decl_int "a"; Anasynt.
    Node_decl_int "c"],
 [Anasynt.Node_inst ("a", Anasynt.Leaf_ident "c");
  Anasynt.Node_inst ("c",
   Anasynt.Node_plus (Anasynt.Leaf_ident "a", Anasynt.
       Leaf_ident "c"));
  Anasynt.Node_inst_bloc
   (Anasynt.Node_bloc ([Anasynt.Node_decl_bool "b"],
     [Anasynt.Node_inst ("b",
       Anasynt.Node_and (Anasynt.Leaf_ident "b", Anasynt.
           Leaf_ident "b"))]))])
```

3.2 Tests Negatifs

```
begin int a ;
 a <- 654b + c
 end

Fatal error: exception Failure("lexing: empty token")


begin ;
 a <- b + c
 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 1, caractere 6")


begin
        int a,
        bool b;
        int c;

        a <- a + c;
        b and (a < c)
 end
```

```
Fatal error: exception Failure("Erreur dans le fichier , a la ligne 4, caractere 1")


begin
        int a,
        bool b;

        a <- a <- b
 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 5, caractere 8")


begin
        bool b;
        b and and b and
 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 3, caractere 3")


begin
        int a,
        bool b;
        int c;

        a <- a + c,
        b and (a < c)
 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 4, caractere 1")


begin
        int a,
        b,
        int c;

        a <- a + c;
        b and (a < c)
 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 3, caractere 1")
```

```
begin
        int a;
        a <- a + c

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 4, caractere 0")


begin
        int a,
        int c;

        a <- c;
        begin
                int b,
                int c;
                b <- c

 end

Fatal error: exception Failure("Erreur dans le fichier , a la ligne 12, caractere 0")
```