```
%{(*
     Chapitre 3
     Graignic Guillaume
     Cadoret olivier
     grammar.mly
*)%}

%{
(* On ouvre arbre.ml qui dÃ©finit le type arbre_abstrait*)
open Arbre
%}

%{(* On dÃ©finit les token possible *)%}
%token <string> IDENT
%token BEGIN VIRG PTVIRG PLUS INT BOOL AFFEC INF AND PAROUV PARFERM END EOF
ERROR

%{(* On gÃ¨re les prioritÃ©s entre les tokens AND INF et PLUS *)%}
%left AND
%left INF
%left PLUS

%{(* On associe le type rendu par la rÃ©gle s Ã  un arbre abstrait *)%}
%type <Arbre.arbre_abstrait> s

%{(* On commence par la rÃ©gle de grammaire s*)%}
%start s

%{(* On dÃ©finit les rÃ©gles de grammaire une Ã  une et
     on construit l'arbre_abstrait au fur et Ã  mesure *)%}
%%
s:
bloc EOF {$1}
;

bloc:
BEGIN sdecl PTVIRG sinst END {Bloc($2,$4)}
;

sdecl:
decl {$1::[]}
| decl VIRG sdecl {$1::$3}
;

decl:
typ IDENT {Declaration ($1,$2)}
;

typ:
INT {Int}
| BOOL {Bool}
;

sinst:
inst {$1::[]}
| inst PTVIRG sinst {$1::$3}
;
```

```
57
58 inst:
59 bloc {$1}
60 | IDENT AFFEC expr {Affectation($1,$3)}
61 ;
62
63 expr:
64 expr PLUS expr {Plus($1,$3)}
65 | expr INF expr {Inf($1,$3)}
66 | expr AND expr {Et($1,$3)}
67 | PAROUV expr PARFERM {$2}
68 | IDENT {Ident($1)}
69 ;
```