

TELECOM Nancy

Projet de compilation

Compilation

Rapport de projet intermédiaire

Réalisé par :
HANTZ Simon
KESLICK Malaury
LASSAGNE Guillaume
RACOUCHOT Maïwenn

Professeurs :
COLLIN Suzanne
DA SILVA Sébastien
OSTER Gérald

2019

Table des matières

1 Introduction

L'objectif de ce projet est d'écrire un compilateur du langage Algol 60 à partir de la spécification fournie sur le site Algol60.org ainsi que celle fournie par les responsables du tutorat. Ce projet est à réaliser par groupe de 4 étudiants. Il est séparé en deux modules : PCL1 et PCL2.

Ce rapport traite de la partie PCL1 pour laquelle il faut créer une grammaire LL(1) grâce à l'outil Antlr, la structure de l'arbre abstrait ainsi que des jeux d'essais permettant de vérifier le bon fonctionnement de ces deux productions. Il est aussi question d'expliciter la gestion de projet mise en place afin de gérer les ressources humaines et temporelles que nous avions à notre disposition.

2 Gestion de projet

2.1 Matrice SWOT

Voici la matrice SWOT (présentant les forces ($STRENGHTS$), les faiblesses ($WEAKNESSES$), les opportunités ($OPPORTUNITIES$) et les menaces ($THREATS$)) de notre groupe.

STRENGTHS

1. Bonne entente au sein du groupe
2. Rigueur et envie d'apprendre des membres du groupe

WEAKNESSES

1. Peu de connaissances initiales sur l'approche du problème
2. Certaines machines personnelles ne pouvant pas utiliser Antlrwork

OPPORTUNITIES

1. Permet d'apprendre à réaliser un travail vers lequel nous ne nous serions probablement pas tournés initialement
2. Permet de mieux comprendre le cours de Traduction
3. Tutorats proposés par les 3A très utiles à l'avancée du projet

THREATS

1. Possibilité de partir dans la mauvaise direction dès le départ et de rendre le travail très chronophage
2. Emplois du temps des différents membres coïncidant très peu : compliqué pour travailler ensemble

2.2 Matrice RACI

Dans cette sous-partie nous allons vous présenter la répartition du travail lors de cette première partie du projet. Il est important de noter que toutes les validations de décisions et de productions se faisaient ensemble, ce qui explique l'absence d'**A**cteurs dans cette matrice RACI. Pour rappel, R signifie **R**esponsable, C est pour **C**onsulté et I est l'initiale d'**I**nformé.

	Simon HANTZ	Malaury KESLICK	Guillaume LASSAGNE	Maïwenn RACOUCHOT
Grammaire LL(1)	R	C	C	C
AST	I	C	R	R
Gestion de projet	C	R	C	C
Rapport	R	R	R	R

FIGURE 1 – Matrice RACI

Simon Hantz	Malaury Keslick	Guillaume Lassagne	Maïwenn Racouchot
Grammaire : 80h AST : 3h Rapport 2h :	Grammaire : 6h AST : 20h Rapport : 8h	Grammaire : 3h AST : 20h Rapport : 10h	Grammaire : 3h AST : 30h Rapport : 7h
Total : 85h	Total : 34h	Total : 33h	Total : 40h

2.3 Diagramme de Gantt

Nous allons ici vous parler de la gestion de notre temps. Puisque les dates que nous avions initialement prévues pour chaque étape n'ont finalement pas été respectées, nous avons décidé de vous présenter nos diagrammes de Gantt prévisionnel et effectif. La différence entre ces deux diagrammes s'explique par la difficulté que nous avons eu à rendre notre grammaire LL(1). Finalement, nous avons réussi à rattraper ce retard lors de la phase suivante, puisque le retard dans la première étape a permis aux responsables de l'AST de se familiariser au maximum avec ses concepts.

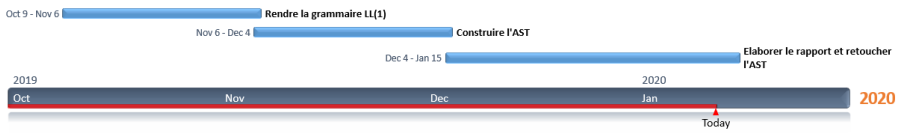

FIGURE 2 – Diagramme de Gantt prévisionnel

FIGURE 3 – Diagramme de Gantt effectif

2.4 Les réunions

Lors de cette première étape du projet, nous nous retrouvions lors des mercredi après-midi afin de faire un bilan sur l'avancé, et de travailler ensemble. Vous pourrez trouver les comptes-rendus de ces réunions en annexe. De plus, nous discutions souvent des problèmes que nous pouvions rencontrer en dehors de ces journées, que cela soit grâce à un groupe de discussion en ligne, ou via des stand-up meetings improvisés à l'école.

3 La grammaire du langage

Notation utilisée pour la grammaire :
— Règles en italique
— Terminaux (mots-clés) entre guillemets
— Expressions régulières d'ANTLR (parenthèses, étoile, plus, point d'interrogation, pipeline) : caractères ni entre guillemets ni en italique
— **IDENTIFIER** : identificateur (de variable, fonction et type)
— **UNSIGNED NUMBER** : nombre pouvant être entier, décimal ou négatif
— **STRING** : chaine littérale
— **TYPE** : différents typage, réel, booléen ou entier
— **PARAMETER_DELIMITER** : permet de délimiter deux paramètres

---

root → *program*

program → 'begin' *unlabelled*
| 'label' ' :' *program*
| *declaration* ';' *program*
| 'inline' '(' **STRING** ')' ';' *program*

program_bis → 'begin' *unlabelled*

program_ter → 'begin' *unlabelled*
| 'label' (' :' (*program_bis* | *for_statement* | *conditional_statement*) | *identifier_list* ';' (*specifier identifier_list* ';')* (*program_bis* | *for_statement* | *conditional_statement*))

unlabelled → *unlabelled_block*
| *compound_tail*

unlabelled_block → *block_head compound_tail*

block_head → (*declaration* ';')+

compound_tail → ('label :')* *statement* (';' ('label :')**statement*)* 'end'

statement → *program_bis*
| *conditional_statement*
| *unlabelled_basic_statement*
| *for_statement*
|

statement_bis → *specifier identifier_list* ';' (*specifier identifier_list* ';')* *after_statement_bis*

| *after_statement_bis*

after_statement_bis → *program_ter*
| *for_statement*
| *conditional_statement*

for_statement → *for_clause statement*

for_clause → 'for' IDENTIFIER ' :=' *for_list_element for_list* 'do'

for_list → (',' *for_list_element*)*

for_list_element → *boolean_expr* ('step' *expression* 'until' *expression*
| 'while' *expression* | (',' *expression*)*)

conditional_statement → *if_clause statement* (options{greedy=true;} : 'else' *statement*)

unlabelled_basic_statement → *go_to_statement*
| 'inline' *actual_parameter_list*
| IDENTIFIER ('[' *boolean_expr* (','*boolean_expr*)*']')?
(((' :";' ?| ' :='|'=') *expression*?)* | (*actual_parameter_list*))

go_to_statement → ('goto' | 'go to') *expression*

expression → *boolean_expr*
| *if_clause expression* ('else' *expression*)?
| *go_to_statement*
| *program*

if_clause → 'if' *boolean_expr* 'then'

boolean_expr → *logic_expr* ('<=>' *logic_expr*)*

logic_expr → *logic_expr_bis* ('=>' *logic_expr_bis*)*

logic_expr_bis → *logic_expr_ter* (('OR' | 'or' | '||' | 'AND' | 'and' |
'&&' | '&' | '|') *logic_expr_ter*)*

logic_expr_ter → *add_expr* (('<' | '>' | '<=' | '>=' | '=' | '<>')
add_expr)*

add_expr → ('+')? *mult_expr* (('+' | '-') *mult_expr*)*

mult_expr → *factor_expr* (('*' | '/' | '%') *factor_expr*)*

factor_expr → ('~')? *value* ('**' ('~')? *value*)*

value → IDENTIFIER (*after_identifier*)?
| '-' UNSIGNED_NUMBER
| UNSIGNED_NUMBER
| '(' *boolean_expr* ')'
| 'FALSE'
| 'TRUE'

after_identifier → *actual_parameter_list*
| '[' *boolean_expr* (',' *boolean_expr*)* ']'

if_statement → *if_clause program_bis*

actual_parameter → STRING
| *expression*

actual_parameter_list → '(' *actual_parameter* (',' *actual_parameter*)* ')'(IDENTIFIER ' :' '(' *actual_parameter* (',' *actual_parameter*)* ')')*

declaration → *type_array_proc_declaration*
| *array_declaration*
| *switch_declaration*
| *procedure_declaration*

type_array_proc_declaration → 'own' *type_array_proc_declaration_bis*
| TYPE *type_array_proc_declaration_ter*

type_array_proc_declaration_bis → *array_declaration*
| TYPE (*type_list* | *array_declaration*)

type_array_proc_declaration_ter → *procedure_declaration*
| *array_declaration*
| *type_list*

type_list → IDENTIFIER (',' IDENTIFIER)*

bound_pair → *expression* ' :' *expression*

bound_pair_list → *bound_pair* (',' *bound_pair*)*

array_declaration → 'array' *array_list*

array_list → *array_segment* (',' *array_segment*)*

array_segment → IDENTIFIER ('[' *bound_pair_list* ']'

	| *array_segment*
switch_declaration	→ 'switch' IDENTIFIER ' :=' *switch_list*
switch_list	→ *expression* (',' *expression*)*
procedure_declaration	→ 'procedure' *procedure_heading* (*statement_bis*) ?
procedure_heading	→ IDENTIFIER (*formal_parameter_list*) ? ' ;' (*value_part*) ?
specifier	→ 'string' | TYPE *fact_type_array_procedure* | 'array' | 'switch' | 'procedure'
fact_type_array_procedure	→ 'procedure' |'array' |
value_part	→ 'value' *identifier_list* ' ;'
identifier_list	→ IDENTIFIER (',' IDENTIFIER)*
formal_parameter_list	→ '(' IDENTIFIER (',' IDENTIFIER)* ')'(IDENTIFIER ' :' '(' IDENTIFIER (',' IDENTIFIER)* ')')*

Figure 1 : Grammaire Algol60

4 Création de l'AST

La principale difficulté qui nous est apparue lors de la construction de l'AST a été de couvrir tous les cas. Dans ce but, nous avons choisi de commencer par gérer des cas très simples (cf fig 4.1), puis de rajouter des cas plus complexes. Pour ce faire, nous nous sommes appuyés sur les listes de "lego pieces" que nous avons trouvées sur le site http ://algol60.org/. Nous avons commencé par prendre les programmes de la catégorie HelloAlgol puis Lego Pieces trouvables à cette adresse.

```
begin
        integer n;
        n := 1
end
```

Figure 4.1 : Un programme Algol60 simple

Pour chaque programme, nous commencions par générer sur ANTLR son arbre syntaxique (parse tree sur ANTLR) afin de déterminer à quelles règles de la grammaire il faisait appel. A partir de cet arbre, nous décidions des noeuds qui devraient ou non apparaître dans l'AST. Enfin nous ré-écrivions les règles de la grammaire correspondant à l'arbre en suivant les règles vues en TP. Nous avons ainsi renommé et enraciné les noeuds que nous avons jugés utiles, fais disparaître les terminaux et non terminaux inutiles. Nous avons de plus fait apparaître certains noeuds qui seront utiles lors de la suite. Par exemple le noeud SI a trois fils qui correspondent à la condition, au sinon et au alors, ce qui permettra d'avoir une vision claire de cette structure lors du parcours de l'arbre.
En procédant ainsi par itérations successives nous sommes parvenus à génerer des arbres corrects pour des cas complexes (cf fig 4.2). Il est toutefois à noter que ces programmes présentaient parfois des erreurs que nous avons corrigées au fur et à mesure de leur utilisation afin de générer un AST correct.

```
begin
        comment vector generation;

        real array a[1:100];
        integer i, dim;
        real b, c;

        procedure vectGen( v, ndim, start, stp);
        real array v; integer ndim; real start, stp;
        begin
                integer i;
                for i := 1 step 1 until ndim do v[i] := start + stp*i
        end;

        dim := 6;
        b :=0; c := 0.1;

        vectGen(a,dim,b,c);

        outstring(1,"result = \n");
        for i := 1 step 1 until dim do outreal (1, a[i]);
        outstring(1,"\n")

end
```

Figure 4.2 : Un programme Algol60 plus complexe

5 Jeux d'essais

1. begin integer n ;
   ininteger (0,n) ;
   if n ¡ 0 then outstring (1,"negative")
   else if n=0 then outstring (1, "0")
   else outstring (1, "positive") ;
   outstring (1,"")

   end

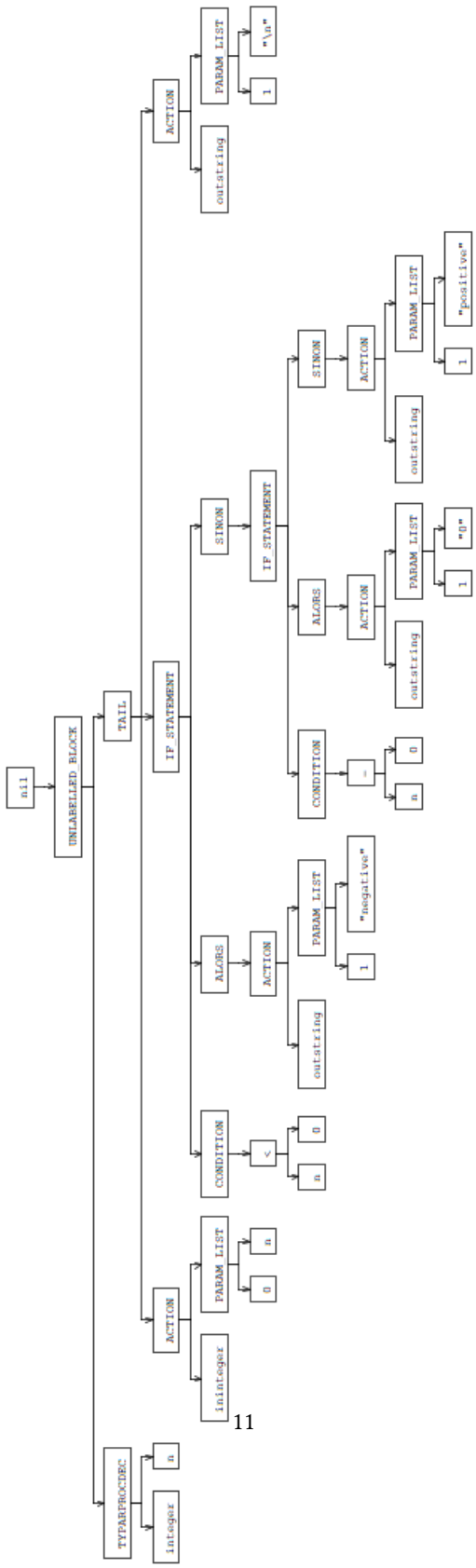

Figure 5.1 : AST programme if

2. begin
   integer i;
   for i := 1 step 1 until 20 do outinteger(1,i)
   end

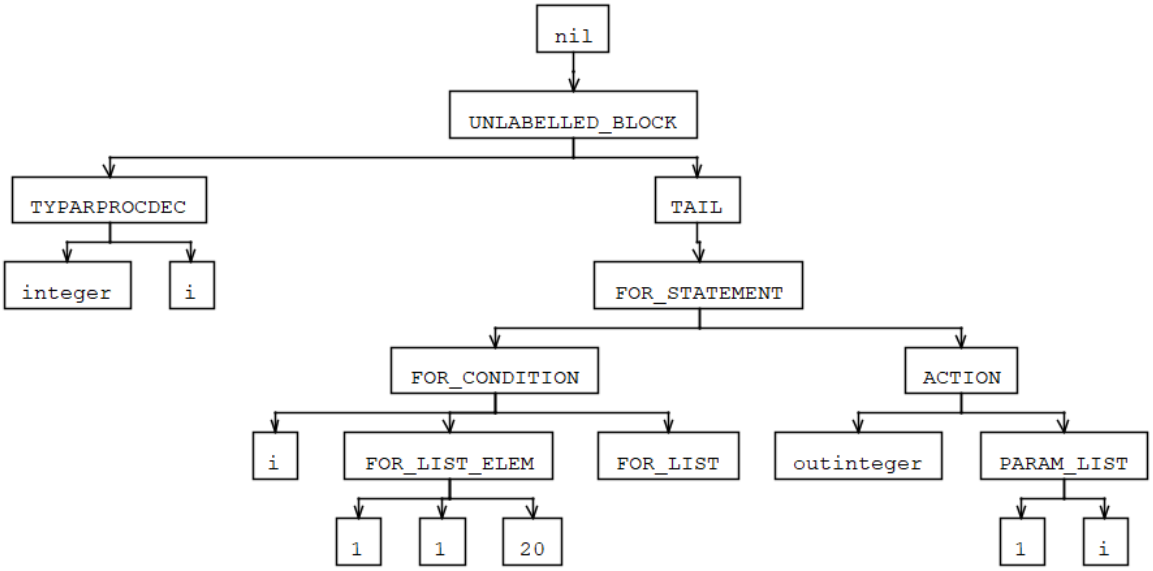

Figure 5.2 : AST programme for

3. begin
   comment dynamic memory allocation in inner block;
   integer n;

   n := 100;
   begin
   integer array a[1 :n];
   a[n] := n;
   outinteger (1, a[n])
   end

   end

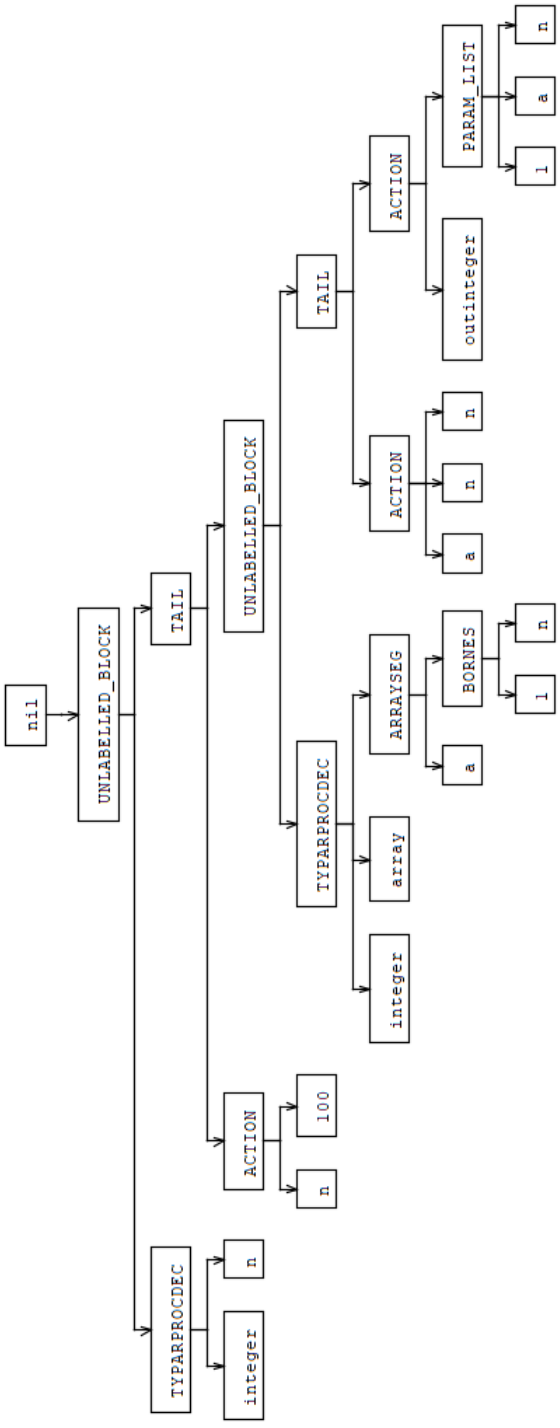

Figure 5.3 : AST programme block

4.
```
begin
comment abs function heviside parabola;
real x, y;

for x := -3.0 step 0.2 until 3.0 do
begin
y := ( abs (x-1) + abs (x+1) - 2.0 ) / 2.0;
outreal (1, x); outreal (1, y); outstring (1,"")
end

end
```

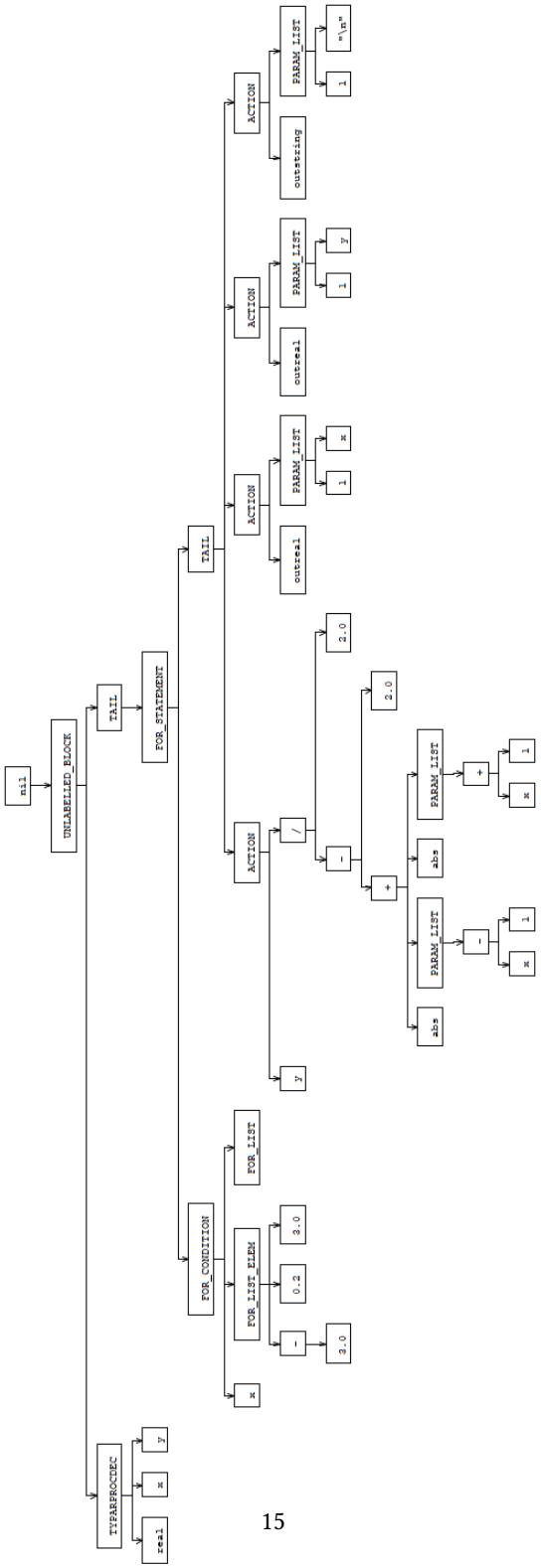

Figure 5.4 : AST programme mathlib

5. real procedure cputime ;
   inline("my_dsa.retval.u.real_val = (double)clock() / (double)CLOCKS_PER_SEC ;") ;

   begin
   outreal(1, cputime)
   end

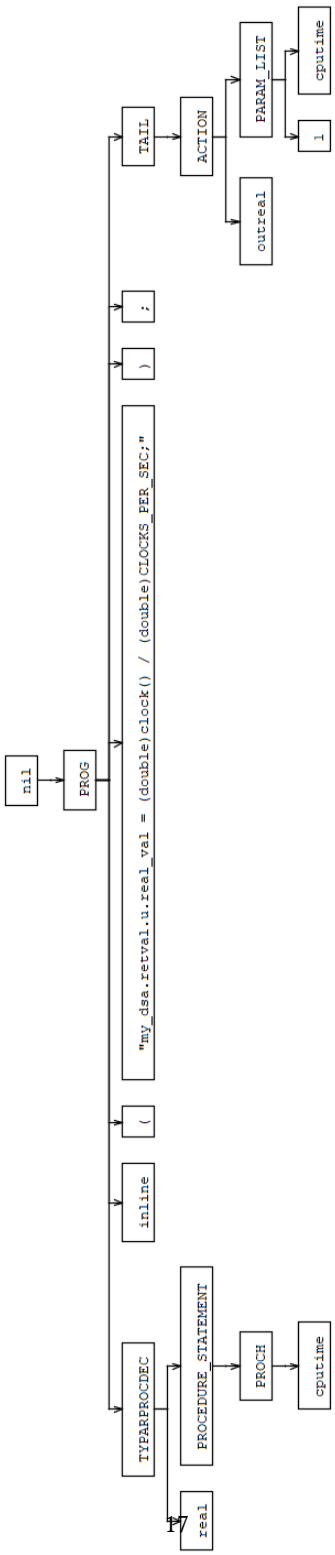

Figure 5.4 : AST programme system

6 Annexe

6.1 Compte rendu de réunion numéro 1.

Motif de la réunion : Réunion de lancement de projet	Lieu : 1.16 (Telecom Nancy)
Présent(s) : Simon Hantz Malaury Keslick François Pichot Maïwenn Racouchot	Date, heure et durée : Mercredi 09/10/2019, 14h (environ 2 heures)

Prise de connaissance du sujet et dicussions autour de celui-ci

La réunion débute par un retour sur le sujet que nous avons reçu par mail. Nous avons tous les quatre conscience que ce projet est un travail long, et qu'il faut bien s'organiser pour ne pas se faire surprendre par des deadline approchant rapidement. Une longue discussion s'entame quant aux attendus du projet, et à la manière dont il faudra procéder pour créer une grammaire. Nous reprenons quelques TD du module de Traduction 1 pour nous aiguiller sur la direction à prendre. En parrallèle, nous nous intéressons aux ressources proposées dans le sujet pour mieux comprendre le langage Algol60.

Nous discutons de la liste de mots réservés trouvée sur la page wikipédia anglaise d'Algol60, et finissons par acter le fait que ces mots doivent se placer dans les tokens. Malgré tout quelques questions se soulèvent, et nous les aborderons dans la dernière partie de ce compte rendu.

Grâce aux exemples de programmes fournis dans les ressources du sujet, nous pensons alors créer la grammaire en commençant avec les programmes les plus simples, puis en complexifiant au fur et à mesure. Nous tenterons de vérifier qu'aucune erreur, qu'aucun abus ne peut être fait avec notre grammaire, et ce à chaque étape de sa création.

Distribution du travail à effectuer sur la première partie

Nous nous mettons rapidement d'accord sur le fait que le travail de création de la grammaire est difficile à partager. Ainsi, nous y travaillerons essentiellement ensemble au cours des prochaines semaines, essentiellement lors des mercredi après-midi, dédiés aux différents projets de deuxième année.

Questions quant au sujet

La question majeure ayant émergée lors de cette réunion concerne les types de données que l'on peut manipuler avec Algol60. En effet, on se demande comment faire entrer en compte dans la grammaire le fait que - par exemple - un *int* doit être codé sur 32 bit, et un *long* sur 64.

Par rapport à cette question, et quant à toutes les autres que nous pouvons nous poser ou auxquelles nous n'avons pas encore pensé, nous décidons d'assister au tutorat de PCL proposé par des étudiants de troisième année le mercredi suivant.

Todo list

Se renseigner sur les différentes sources citées dans le sujet et noter toutes les questions qui peuvent survenir afin de pouvoir les poser en tutorat.

Date de la prochaine réunion : mercredi 16/10 après le tutorat.

6.2 Compte rendu de réunion numéro 2.

Motif de la réunion :	Lieu :
Travail sur la grammaire	1.12 (Telecom Nancy)
Présent(s) :	Date, heure et durée :
Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Mercredi 16/10/2019, 14h (environ 1 heure)

Prise de connaissance d'une grammaire et exploitation de celle-ci

Par rapport à la semaine précédente, nous avons changé de manière de faire en démarrant sur une grammaire existante afin de l'adapter à Antlr.
La réunion débute donc sur des discussions autour d'une grammaire connue d'Algol 60. Nous avons commencé par étudier celle-ci et nous nous sommes rapidement rendu compte qu'elle était récursive gauche immédiate, non factorisée et donc pas LL(1).
Nous reprenons quelques TDs de Mathématiques pour l'informatique afin de nous aider à éliminer la récursivité immédiate gauche. En parrallèle, nous nous intéressons aux premiers et suivants de cette grammaire.

Distribution du travail à effectuer sur la grammaire

Nous nous mettons rapidement d'accord sur une répartition équitable des productions de la grammaire et chacun calcule les premiers et suivants des différentes règles pour statuer sur l'état de celle-ci et, la rendre par la suite, LL(1).

Questions quant au sujet

L'une des principales interrogation est sur la manière de démarer le projet. Nous avons eu des échos comme quoi certains groupes reprenaient la grammaire de 0 et d'autres utilisaient une existante... Nous pensons qu'il sera plus aisé d'adapter une grammaire existante que d'en recréer une de toute pièce. Par rapport à cette question, et quant à toutes les autres que nous pouvons nous poser ou auxquelles nous n'avons pas encore pensé, nous décidons d'assister au tutorat de PCL proposé par des étudiants de troisième année ce vendredi.

Todo list

Utiliser Antlr pour modifier les règles de grammaire.
Continuer les premiers et les suivants des productions. Factoriser la grammaire.

Date de la prochaine réunion : mercredi 23/10.

6.3 Compte rendu de réunion numéro 3.

Motif de la réunion :	Lieu :
Travail sur la grammaire	Salle de travail (Telecom Nancy)
Présent(s) :	Date, heure et durée :
Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Mercredi 23/10/2019, 14h (environ trente minutes)

Discussions et travail effectué

Nous avons, depuis la dernière réunion, continué à travailler sur la grammaire dans le but de la rendre LL(1). Celle-ci a été dérécursivée et factorisée, mais il nous reste les problèmes d'ambiguïté à gérer. Ceux-ci nous compliquent beaucoup la tâche puisque les dépendances entre les règles de la grammaire sont nombreuses. Nous nous fixons donc comme objectif de réussir à rendre cette grammaire LL(1) pour la fin des vacances afin de pouvoir nous concentrer sur l'AST avant le rendu de la première partie.
En parallèle, nous avons commencé à transcrire la grammaire sous format ANTLR pour nous avancer et nous familiariser le plus rapidement possible avec ses notations.

Todo list

Continuer le travail sur les ambiguités et la transcription en format ANTLR.

6.4 Compte rendu de réunion numéro 4.

Motif de la réunion : Réunion d'avancé de travail	Lieu : Audio-conférence
Présent(s) : Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Date, heure et durée : Mercredi 07/11/2019, 14h (environ trente minutes)

Discussions sur le travail en cours et restant avant le premier rendu

Nous avons, depuis la dernière réunion, continué à travailler sur la grammaire dans le but de la rendre LL(1). Simon a pratiquement terminé ce travail. En attendant, Malaury, Maïwenn et Guillaume travaillent sur l'AST afin de se familiariser au plus possible avec ses subtilités, dans le but de rendre ce travail le plus efficace possible lorsque la grammaire sera terminée.

Todo list

Simon : Terminer de rendre la grammaire LL(1).
Malaury, Maïwenn et Guillaume : travailler sur l'AST.

6.5 Compte rendu de la réunion numéro 5.

Motif de la réunion :	Lieu :
Réunion d'avancé de travail	Médialab (Telecom Nancy)
Présent(s) :	Date, heure et durée :
Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Mercredi 20/11/2019, 14h (environ quarante minutes)

Discussions autour de la grammaire et décisions prises

Simon nous annonce qu'il se heurte encore à des problèmes complexes dans son travail sur la grammaire, qui n'est donc pas totalement LL(1) pour le moment. Nous prenons donc la décision de demander la grammaire de Mme. Collin et de commencer à travailler dessus, pour ne pas prendre trop de retard au cas où Simon ne parviendrait pas à la terminer. Simon s'est entretenu avec Mme Collin qui lui a dit que demander la grammaire ne serait pas pénalisant si nous arrivions à finir la nôtre et que nous l'utilisions pour la suite du projet.

Todo list

Simon : Tenter de rendre la grammaire LL(1).
Malaury, Maïwenn et Guillaume : commencer l'AST sur la grammaire récupérée.

6.6 Compte rendu de la réunion numéro 6.

Motif de la réunion :	Lieu :
Réunion d'avancé de travail	Médialab (Telecom Nancy)
Présent(s) :	Date, heure et durée :
Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Mardi 26/11/2019, 13h (environ quarante minutes)

Grammaire LL(1) et AST

Simon nous annonce qu'il est enfin parvenu à rendre la grammaire LL(1). Les trois autres membres du groupe abandonnent donc le travail entamé sur la grammaire fournie et passent sur cette nouvelle version. On remarque que ce travail peut-être un peu compliqué pour Maïwenn, dont l'ordinateur refuse de travailler avec AntlrWork, mais Malaury propose des échanges d'ordinateurs lorsqu'elle souhaite travailler dessus, puisque le groupe s'est déjà rendu compte que travailler en même temps sur cette tâche était assez contre-productif.

Todo list

Malaury, Maïwenn et Guillaume : élaborer l'AST à partir de la grammaire LL(1) finalisée et sélectionner les programmes à utiliser lors de la démonstration du 04/12.

6.7 Compte rendu de la réunion numéro 7.

Motif de la réunion :	Lieu :
Réunion d'avancé de travail	Salle P.I. (Telecom Nancy)
Présent(s) :	Date, heure et durée :
Simon Hantz Malaury Keslick Guillaume Lassagne Maïwenn Racouchot	Mercredi 04/12/2019, 15h (environ vingt minutes)

Retour sur la démonstration

Nous revenons sur la démonstration de notre grammaire et de notre AST que nous venons de faire à Mme. Collin. Après un petit problème au démarrage, nous avons finalement su nous rattraper et exposer notre travail. Nous avons pris en note toutes les remarques que notre professeur a pu nous apporter, afin de retravailler sur ces points afin le rendu de cette partie.

Todo list

Revoir les quelques points posant problèmes (cf feuille dans la conversation messenger) + rédiger le rapport.