Oraux Compilateur

Sujet 2006

Etienne Champetier
Diana Dragusin
Maxime Guyot
Ugo Meda
Valentin Viou

Question 1.1

Question 1.1

- Analyse lexicale
  chaînes de caractères $\rightarrow$ lexèmes
- Crible
  lexèmes $\rightarrow$ liste d'unités lexicales
- Analyse syntaxique
  listes d'unités lexicales $\rightarrow$ arbre syntaxique abstrait
- Analyse sémantique
  arbre syntaxique abstrait $\rightarrow$ arbre abstrait "décoré"
- Optimisations indépendantes de la machine cible
  arbre abstrait "décoré" $\rightarrow$?

Question 1.2- 1

X = fonc(T1, tuple(bool, liste(T1)))
Y = fonc(ent, tuple(T2, T2)

- T1/ent
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(T2, T2)
- T2/bool
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(bool, bool) *X* et Y ne peuvent pas s'unifier pour θ =[T1/ent, T2/bool]
- T2/liste(ent)
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(liste(ent), liste(ent)) *X* et Y ne peuvent pas s'unifier pour θ =[T1/ent, T2/liste(ent)]

X et Y ne peuvent pas s'unifier

Question 1.2- 2

X = fonc(T1, tuple(bool, liste(T1)))
Y = fonc(ent, tuple(T2, T3)

- T1/ent
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(T2, T3)
- T2/bool
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(bool, T3)
- T3/liste(ent)
  → fonc(ent, tuple(bool, liste(ent))) et fonc(ent, tuple(bool, liste(ent))

X et Y s'unifient pour θ = [T1/ent, T2/bool, T3/liste(ent)]

Question 2.1

Les groupes de caractères ne sont pas disjoints, alors que les classes de caractères sont disjointes. Or, il faut que l'intersection de 2 ensembles de caractères soit vide pour pouvoir l'utiliser dans un langage régulier.

Question 2.2

Si on a 2 ensembles A et B non disjoints, on va les remplacer par A-B B-A et $A \cap B$.

Question 2.3

Ici les unités lexicales sont exprimées en utilisant le formalisme des expressions régulières. Ceci n'est pas surprenant : on a procédé de la même manière lors des TP.

Question 2.4

Une « EOCL »est :

soit un caractère point, suivi d'un BS (blank_space), d'un NL (end_of_line) ou d'une fin de fichier

soit une fin de fichier

Question 2.5

Il faut expliquer les 2 cas de EOCL :
- Possibilité d'avoir un point dans le code - Arrêter l'analyse à la fin du fichier

Question 2.6

$V_T = \{EOCL, ':-', '|', ',', '.', '(', ')', '[', ']', VAR, STRING, INT, ATOM\}$
$V_N = \{prog, clS, clause, head, goals, term_g, termlist, list, listexpr, term\}$

Question 2.7

$I_0 = \{prog \Rightarrow \bullet clS$
$clS \Rightarrow \bullet clauseEOCL$
$cls \Rightarrow \bullet clauseEOCLcls$
$clause \Rightarrow \bullet head$
$clause \Rightarrow \bullet head :- goals$
$clause \Rightarrow \bullet :- goals$
$head \Rightarrow \bullet term_g$
$term_g \Rightarrow \bullet ATOM$
$term_g \Rightarrow \bullet ATOM(termlist)\}$

$I_1 = Transition(I_0, clause) = \{cls \Rightarrow clause \bullet EOCL$
$cls \Rightarrow clause \bullet EOCLcls\}$

$I_2 = Transition(I_0, head) = \{clause \Rightarrow head \bullet$
$clause \Rightarrow head \bullet :- goals\}$

$I_3 = Transition(I_0, term_g) = \{head \Rightarrow term_g \bullet\}$

$I_4 = Transition(I_0, ATOM) = \{term_g \Rightarrow ATOM \bullet$
$term_g \Rightarrow ATOM \bullet (termlist)\}$

Question 2.7

$I_5 = Transition(I_2, " :-")\{clause \Rightarrow head :- \bullet goals$
$goals \Rightarrow \bullet term_g$
$goals \Rightarrow \bullet goals, goals$
$term_g \Rightarrow \bullet ATOM$
$term_g \Rightarrow \bullet ATOM(termlist)\}$

$I_6 = Transition(I_5, goals) = \{clause \Rightarrow head :- goals \bullet$
$goals \Rightarrow goals\bullet, goals\}$

$I_7 = Transition(I_6, ",") = \{goals \Rightarrow goals \bullet goals$
$goals \Rightarrow \bullet term_g$
$goals \Rightarrow \bullet goals, goals$
$term_g \Rightarrow \bullet ATOM$
$term_g \Rightarrow \bullet ATOM(termlist)\}$

$I_8 = Transition(I_7, goals) = \{goals \Rightarrow goals, goals \bullet$
$goals \Rightarrow goals\bullet, goals\}$

Question 2.8

$$suivant(clause) = \{EOCL\}$$
$$suivant(head) = \{" :-"; EOCL\}$$
$$suivant(goals) = \{",", EOCL\}$$
$$suivant(term\_g) = \{" :-"; EOCL; ","; "|"; ")"; "]"\}$$

Question 2.9

Etat	EOCL	:-	,	(	|	]	)	ATOM	clause	head	goals	termg
0								d4	1	2		3
1	d											
2	r4	d5										
3	r7	r7										
4	r10	r10	r10	d	r10	r10	r10					
5								d			6	
6	r5		d7									
7								d			8	
8	r9		r9/d7									

Question 2.10

On a un conflit Shift/Reduce à l'état 8 dans la colonne $'$,$'$:
goals, *goals*, *goals*

soit d7 $\Rightarrow$ goals, (goals, goals)

soit r9 $\Rightarrow$ (goals, goals), goals

En Prolog la virgule correspond au « et »ça ne va donc pas changer le sens.

Question 2.11

I_1=transition(I_0,clause) est l'unique ensemble transition(I_0,clause) donc dans la case (I_0,clause) on peut uniquement décaler vers 1, on ne peut pas décaler vers un autre ensemble car il n'y en a pas.

Question 2.12

On explique qu'en prolog, on peut prend n'importe quel atôme et en faire un opérateur (infixe, ou préfixe). Faite pour analyser les termes, pas l'application des opérateurs.

Question 2.13

Elements de réponse :
Sans espace : foncteur.
Atome + blanc + parenthèse = opérateurs
On ne peut pas construire la table sans connaître les terminaux.

Question 2.14

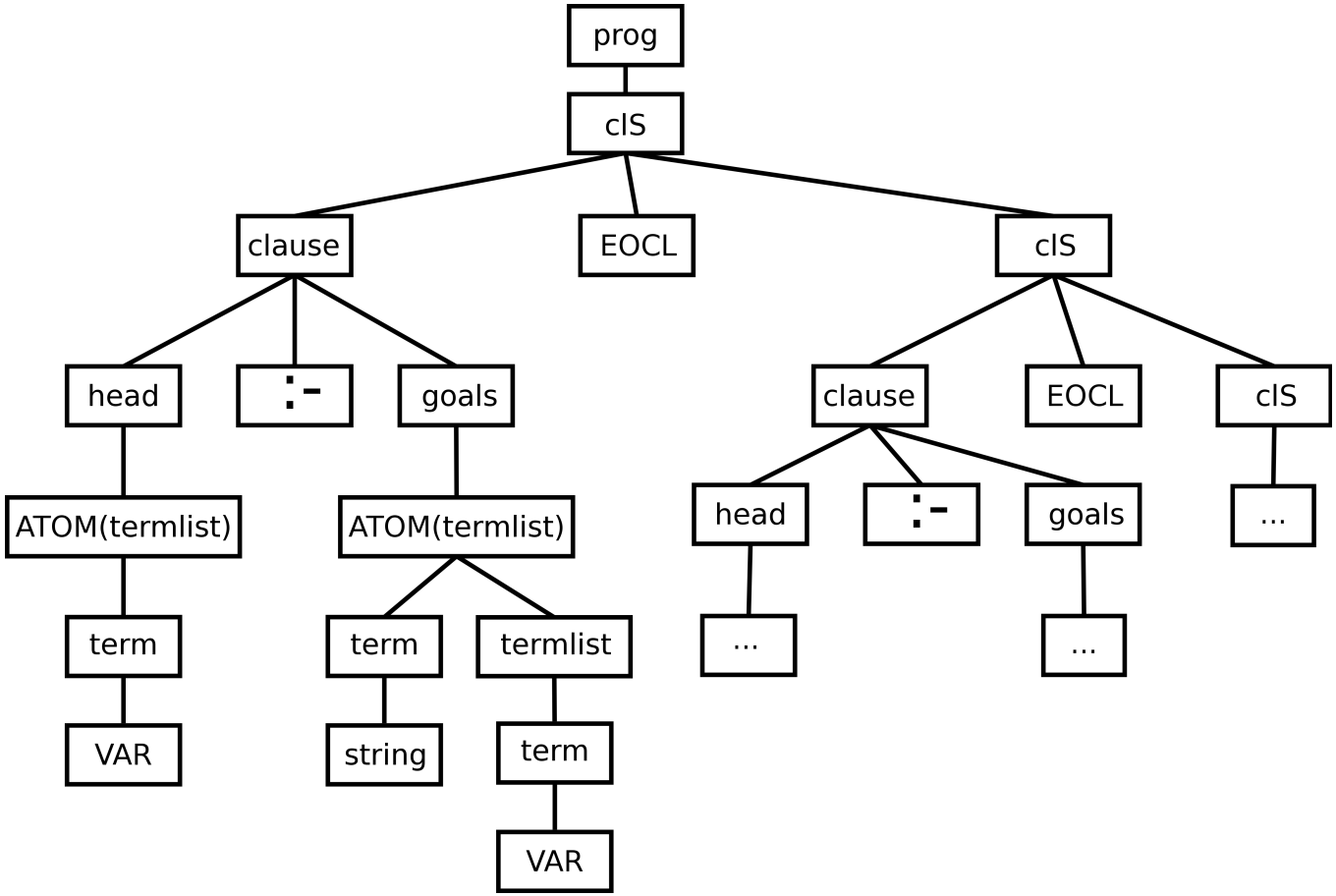
dededu35 est con !
Au lieu de bricoler, il aurait mieux fait de prendre un papier et résoudre lui-même.
(Foutage de gueule inside)

Question 2.15

Il n'avait qu'à définir ses opérateurs.

Question 3.1

Arbre syntaxique concret :



Question 3.2

1	prog ⇒ clS	prog.lstpred = clS.lstpred
2	clS ⇒ clause EOCL	clS.lstpred = clause.lstpred
3	clS ⇒ clause EOCL clS_1	clS.lstpred = clause.lstpred@clS_1.lstpred
4	clause ⇒ head	clause.lstpred = [head.pred]
5	clause ⇒ head :- goals	clause.lstpred = [head.pred]
6	clause ⇒ :- goals	clause.lstpred = []
7	head ⇒ term g	head.pred = term g.pred
10	term g ⇒ ATOM	term g.pred = ATOM.nom + "/0"
11	term g ⇒ ATOM(termlist)	term g.pred = ATOM.nom + "/" + termlist.ar
12	termlist ⇒ term	termlist.ar = 1
13	termlist ⇒ term , $termlist_1$	termlist.ar = 1 + $termlist_1$.ar

Question 3.3

1	prog ⇒ clS	prog.lstpreduse = clS.lstpreduse
2	clS ⇒ clause EOCL	clS.lstpreduse = clause.lstpreduse
3	clS ⇒ clause EOCL clS_1	clS.lstpred = clause.lstpreduse@clS_1.lstpreduse
4	clause ⇒ head	clause.lstpreduse = []
5	clause ⇒ head :- goals	clause.lstpreduse = goals.lstpreduse
6	clause ⇒ :- goals	clause.lstpreduse = goals.lstpreduse
7	head ⇒ term g	
8	goals ⇒ term g	goals.lstpreduse = [term g.pred]
9	goals ⇒ $goals_1$, $goals_2$	goals.lstpreduse = goals.lstpreduse@goals.lstpreduse
10	term g ⇒ ATOM	term g.pred = ATOM.nom + "/0"
11	term g ⇒ ATOM(termlist)	term g.pred = ATOM.nom + "/" + termlist.ar
12	termlist ⇒ term	termlist.ar = 1
13	termlist ⇒ term , $termlist_1$	termlist.ar = 1 + $termlist_1$.ar

Question 3.4

1	prog $\Rightarrow$ clS	prog.lstpred = clS.lstpred
2	clS $\Rightarrow$ clause EOCL	clS.lstpred = clause.lstpred
3	clS $\Rightarrow$ clause EOCL clS_1	clS.lstpred = (si clause.lstpred[0]==clS_1.lstpred[0] alors clS_1.lstpred sinon clause.lstpred@clS_1.lstpred
4	clause $\Rightarrow$ head	clause.lstpred = [head.pred]
5	clause $\Rightarrow$ head :- goals	clause.lstpred = [head.pred]
6	clause $\Rightarrow$:- goals	clause.lstpred = []
7	head $\Rightarrow$ term g	head.pred = term g.pred
10	term g $\Rightarrow$ ATOM	term g.pred = ATOM.nom + "/0"
11	term g $\Rightarrow$ ATOM(termlist)	term g.pred = ATOM.nom + "/" + termlist.ar
12	termlist $\Rightarrow$ term	termlist.ar = 1
13	termlist $\Rightarrow$ term , $termlist_1$	termlist.ar = 1 + $termlist_1$.ar

Question 3.5

Le fait de supprimer de suite est mieux niveaux performance