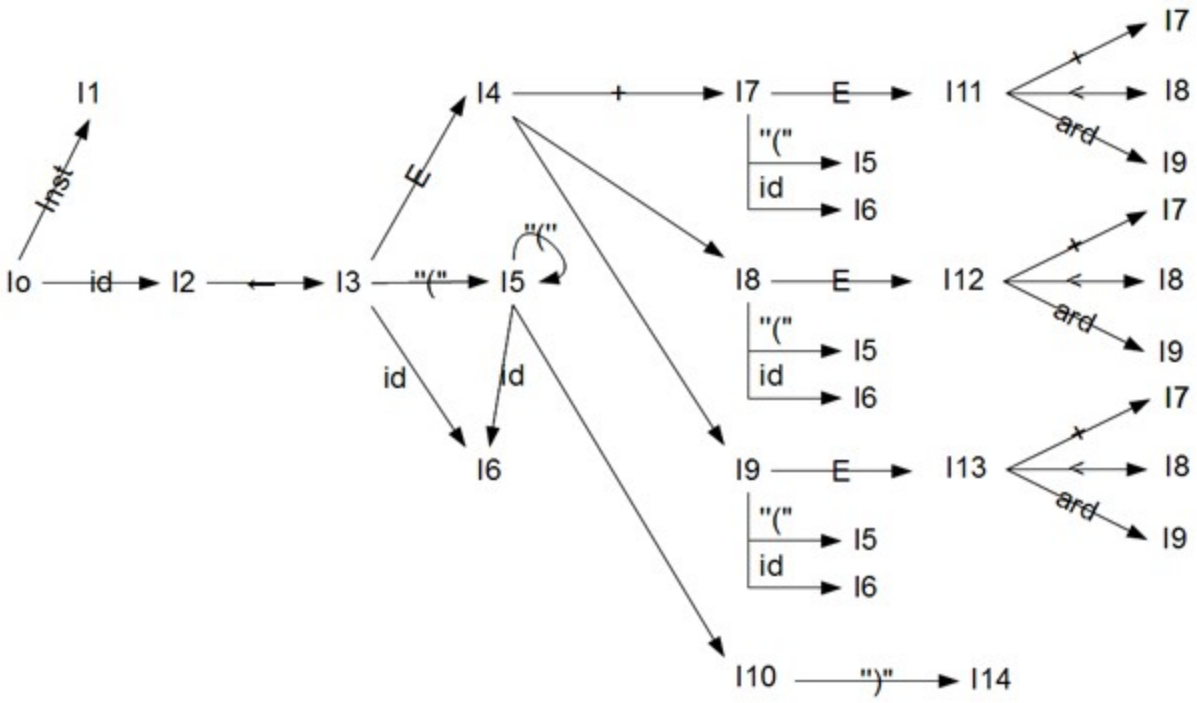
Compte Rendu - Chapitre 3
Analyse lexicale et syntaxique ascendante, ocamllex et ocamlyacc

Graignic Guillaume
Olivier Cadoret

I/ Table SLR calculée en préparation de TP

I0 => {Inst' → .Inst, Inst → .id ← E}

I1 = tcons(I0,Inst) = {Inst' → Inst.}
I2 = tcons{I2,id) = {Inst → id. ← E}
I3 = tcons(I2,E) = {Inst → id ← .E,E → .E+E}
{E → .E<E, E → .EAndE, E → .(E) , E → .id}
I4 = tcons(I3,E) = {Inst → id<E., E → E.+E , E → E.<E, E → E.AndE}
I5 = tcons (I3,'(') = {E → (.E), E → .E + E , E → .E<E , E → .EAndE , E → .(E) , E → .id}
I6 = tcons(I3,id) = {E → id.}
I7 = tcons(I4,+) = {E → E + .E , E → .E + E , E → .E<E , E → .EAndE , E → .(E) , E → .id}
I8 = tcons(I4,<) = {E → E<.E , E → .E + E , E → .E<E , E → .EAndE , E → .(E) , E → .id}
I9 = tcons(I4,And) = {E → EAnd .E,E → .E + E,E → .E<E , E → .EAndE , E → .(E) , E → .id}
I10 = tcons(I5,E) = {E → (E.),E → E.+E , E → E.<E , E → E.AndE}
tcons(I5, '(') = I5
tcons(I5,id) = I6
I12 = tcons(I8,E) = {E → E<E. , E → E,+E , E → E.<E , E → E,AndE}
I13 = tcons(I9,E) = {E → EAndE, , E → E,+E , E → E,<E , E → E,AndE}
I14 = tcons(I10,'(') = {E → (E),}



II/ Résolutions pour les conflits et leurs justifications

suivant(c) = {+ , < , And ,) , *$}*

Inst' → Inst
1 Inst → Id "' ← " E
2 E → E "' + "E
3 E → E "' < "E
4 E → E "' And "E
5 E → "(" E ")"
6 E → id

	Id	←	+	<	And	(	)	*$*	Inst	E
I0	d2								1	
I1								Acc		
I2		d3								
I3	d6					d5				4
I4			d7/r(1)	d8/r(1)	d9/r(1)		r(1)	r(1)		
I5										10
I6			r(6)	r(6)	r(6)					
I7	d6					d5				11
I8	d6					d5				12
I9	d6					d5				13
I10							d14			
I11			d7/r(2)	d8/r(2)	d9/r(2)		r(2)	r(2)		
I12			d7/r(3)	d8/r(3)	d9/r(3)		r(3)	r(3)		
I13			d7/r(4)	d8/r(4)	d9/r(4)		r(4)	r(4)		
I14			d7/r(5)	d8/r(5)	d9/r(5)		r(5)	r(5)		

Pour la ligne 4 voir question 3.
Pour les lignes 11,12,13 et 14 cela s'explique par le fait que le And est prioritaire sur le < qui est lui même prioritaire sur le +.

III/ Réponses aux questions

Question 1

L'utilisation d'un crible avec ocamllex permet de diminuer le nombre d'états et de transitions.

Question 2

Il est beaucoup plus facile d'écrire une grammaire LR qu'une grammaire LL. Dans une grammaire LL, il est nécessaire de lever toute ambiguité tandis que dans une grammaire LR, cela n'est pas nécessaire.
Par contre, une grammaire LL nécessite une analyse descendante qui est plus facile à mettre en place qu'une analyse ascendante (pour les grammaires LR). En effet, une

analyse LR demande la réalisation de tables SLR ce qui peut se révéler fastidieux.

Question 3

La première ligne intéressante est la ligne I4. On y arrive par :
I0,Id -> d2
I2,"<-" -> d3
I3,Expr -> d4
On a alors des conflits shift/reduce pour les léxèmes "+", "<" et "and". On effectue un décalage car on est toujours dans la règle (Inst -> Id "<-" Expr).

Question 4

Une colonne vide dans une table SLR signifie que ce terminal ne sera jamais rencontré.

IV/ Code et jeux de tests commentés