Analyseur syntaxique ascendant

Arthur Garnier

February 5, 2015

1 Principe

- Il lit le texte de gauche à droite
- On part du mot à analyser
- On remplace itérativement les fragments du mot courant qui correspondent à des **nombres droits** d'une production par le membre gauche de la production en question.
- L'analyse réussit si le mot courant final est l'axiome.

Les analyseurs syntaxiques ascendant sont appelés **LR(k)** (Left to Right scanning).

k → On utilise k unités lexicales de la phrase d'entrée pour faire la prédiction.

Right most dérivation → On constuit une dérivation droite en partant du mot à analyser.

Familles d'analyseurs ASC :

- LR(0)
- SLR(1)
- LR(1)

ASA se base sur deux opérations :

- Décalage + lecture
- Réduction

1.1 Exemple

$$\begin{cases} s' \to s \\ s \to Ac \\ A \to AaAb \\ A \to d \end{cases}$$

Mot à analyser : dadbc$

- "Shift" : Lire la $1^{\text{ère}}$ unité lexicale et empiler ⇒ $\boxed{\text{d}}$
- Réduction car d dans la pile correspond à un membre dt d'une production ⇒ $\boxed{\text{A}}$

PAS de lecture (on n'avance pas dans le texte)

$\boxed{\text{Aad}}$

$\boxed{\text{AaA}}$

- Lire et empiler

$\boxed{\text{AaAb}} \Rightarrow \boxed{\text{A}}$

$\boxed{\text{Ac}} \Rightarrow \boxed{\text{s}} \Rightarrow \boxed{\text{s'}}$

Comment, par algorithme, décider à chaque étape d'une réduction ou d'une lecture ?

→ construire un automate déterministe LR(0) qui dira quoi faire en fonction :

- Du contenu de la pile
- De l'unité lexicale courante

2 Automate LR(0)

2.1 Définition : item

Un **item** d'une grammaire G est une production de G avec un "point" repérant une position dans sa partie droite.

A → xy donne lien à 3 items :

- A → .xy
- A → x.y
- A → xy.

Rq : A → ε donne 1 item : A → .

$\boxed{\text{Le mot vide } \varepsilon \text{ n'est **pas** une unité lexicale}}$

Intuitivement c'est la quantité de motif reconnu dans une partie droite de production.

2.2 Définition : fermeture

Soit I un ensemble d'items de G

Fermeture (I) = Ensemble d'items construits à partir de I, tel que :

- Placer chaque item de I dans Ferm(I)
- Si $[A \to \alpha.B\beta] \in Ferm(I)$ et $B \to \gamma$
- Alors ajouter B → .γ à Ferm(I) (sauf si déjà dedans)

itérer

2.3 Définition états de l'automate LR(0)

Les états de l'automate LR(0) seront des ensembles I d'items obtenus par fermeture.

2.4 Transisions de l'automate LR(0)

Posons :

- I = Un ensemble d'items
- X = Symbole de G

On déduit Transition (I,X) comme la fermeture de l'ensemble des items A $\rightarrow \alpha X.\beta$ tel que A $\rightarrow \alpha.X\beta \in B$

$$\begin{cases} s' \rightarrow s \\ s \rightarrow Ac \\ A \rightarrow AaAb \\ A \rightarrow d \end{cases}$$

Construire l'automate LR(0)

0 $\rightarrow$ Pas de symbole de prévision

I_0

- S'$\rightarrow$ .S
  - Fermeture de S car . devant un Non terminal
- S $\rightarrow$ .Ac
  - Fermeture de A
- A $\rightarrow$ .AaAb
- A $\rightarrow$ .d

I_1 = AV(I_0, S)

- S' $\rightarrow$ S.

I_2 = AV(I_0, A)

- S $\rightarrow$ A.c
- A $\rightarrow$ A.aAb

I_3 = AV(I_0, d) = AV(I_5, d)

- A $\rightarrow$ d.

I_4 = AV(I_2, c)

- S $\rightarrow$ Ac.

I_5 = AV(I_2, a) = AV(I_6,a)

- A$\rightarrow$Aa.Ab
- Ferm(A)

- A → .AaAb
- .d

I_6 = AV(I_5, A)

- A → AaA.b
- A → A.aAb

I_7 = AV(I_6, b)

- A→AaAb.

2.5 Table d'analyse LR(0)

Utilisation de l'automate LR(0) pour construire la table d'ASA LR(0)

	Term	¬ Term
I_0		
.		
.	Action	Transition
.		
I_n		

4 actions possibles :

- Décalage + lecture
- Réduction
- Accepter
- Erreur

L'automate LR(0) + 1 pile vont permettre de dire ce qu'il faut faire, c'est à dire :

- Lire
- Réduire

2.6 Remplissage de la table LR(0)

- Table **Action** :
  - Si [S' → S.] s' = axiome appartient à l'état I_i, alors remplire ACTION(i,$)="Accepter texte" (état de satisfaction). Analyse terminée et OK
  - Si [X → β.] ∈ I_i, X ≠ S' alors ACTION(I_i, a) = "réduction de X → β"
  - Si [X → $\alpha.a\beta$] ∈ I_i (avec A∈T) alors ACTION(I_i, a)="Lire et aller à I_j"
  - Mettre "erreur" dans toute les autres cases de ACTION
- Table TRANSISTION :

– Si dans l'automate LR(0) on a TRANSISTION(I_i,X)=I_j alors TRANSISTION(I_i, X)=I_0

- S'→ S = r_0
- S → Ac = r_1
- A → AaAb = r_2
- A → d = r_3
- d_j = Lire et aller à

	a	b	c	d	$	S'	S	A
I_0	err	err	err	d_3	err		1	2
1					Accepter			
2	d_5		d_4					
3	r_3	r_3	r_3	r_3	r_3			
4	r_1	r_1	r_1	r_1	r_1			
5				d_3				6
6	d_5	d_7						
7	r_2	r_2	r_2	r_2	r_2			

Conclusion : Aucun Conflit ⇒ Analyseur est LR(0)

Exemple de fonctionnement

Etat	Pile	Bulle	Texte source
0	d_3 Lecture et aller à 3		**d**adbc$
0d3	r_3		**a**dbc$
0A2a5d3			**d**bc$
0A2a5A6	r_3		**b**c$
012aSA6b7	r_2		**c**$
0A2c4	r_1		**$**
A0S1			$

3 Analyseurs SLR(1)

3.1 Exemple - Introduction

Soit G :

- E' → E
- E → E+T
- E → T
- T → T*F
- T → F

- F → idf

Automate LR(0)

I_0

- E'→.E
- E → E+T
- E → T
- T → T*F
- T → F
- F → idf

I_1 = AV(I_0, E)

- E' → E.
- E → E.+T

I_2 = AV(I_0, T)

- E → T.
- T → T.*F

I_3 = AV(I_0, F)

- T→F.

I_4 = AV(I_0, idf)

- F → idf.

Dérouler sur i+i*i$

Etat	Pile	Bulle	Texte source
0			**i**+i*i$
0i4	r_5		**+**i*i$
0F3	r_4		
0T2	r_2		
0E1	Conflit		

Construction de table SLR(1) à partir de l'automate LR(0)

Table transition : idem que pour LR(0)

Table ACTION :

- Si [S' → S.] … Idem LR(0)

- Si [A → $\alpha.a\beta$ idem
- Si [A → $\beta.$ ∈ I_i (A≠S') remplir ACTION (I_i, a) = "Réduire par A → B"

Etat	Pile	Bulle	Texte source
0			((1.2),3)$
0(**3**			(1.2),3)$
0(**3**(**3**			1.2),3)$
0(**3**(**3**1**4**			.2),3)$
0(**3**(**3**V**2**			.2),3)$
0(**3**(**3**A**6**			.2),3)$
0(**3**(**3**A**6**.**7**			2),3)$
0(**3**(**3**A**6**.**7**2**4**			),3)$
0(**3**(**3**A**6**.**7**V**2**			),3)$
0(**3**(**3**A**6**.**7**A**10**			),3)$
0(**3**(**3**A**6**.**7**A**10**)**13**			,3)$
0(**3**(**3**A**6**			,3)$
0(**3**(**3**A**6**,**9**			3)$
0(**3**(**3**A**6**,**9**3**4**			)$
0(**3**(**3**A**6**,**9**V**2**			)$
0(**3**(**3**A**6**,**9**A**12**			)$
0(**3**(**3**A**6**,**9**A**12**S**14**			)$
0(**3**(**3**A**6**			)$
0(**3**(**3**A**6**S8			)$
0(**3**(**3**A**6**S8)**11**			$
0A**1**			$

4 Analyseurs LR(1) (et LALR(1))

4.1 Rappel

On a vu LR(0) puis SLR(1), la différence au niveau des symboles (unités lexicales) de réduction.

Les symboles de réduction pour SLR(1) est inclu dans l'ensemble des symboles de réduction pour LR(0)=T

Outils : YACC/BISON

4.2 Pourquoi LR(1) ?

Soit I_i =

- A → B.aDa

- D → B.
- . . .

Si on lit **a** ACTION(I_i,a)=d_k

Si on lit **a**, on réduit par D → B pour tout symbole appartenant au suivant de D donc, entre-autre pour **a**.

La méthide SLR(1) n'est pas assez puissante pour se souvenir de suffisamment de contexte pour décider de l'action à effectuer (lecture ou réduction) sur une entrée.

⇒ On va ajouter une informations supplémentaire dans les items.

Un item devient : $\boxed{[\mathrm{A} \rightarrow \alpha.\beta, a]}$ où $A \rightarrow \alpha.\beta \in$ Grammaire et $a \in T \cup \{\}\$$

La pré-vision n'aura d'effet que sur les items de la forme $[A \rightarrow \alpha., a]$ où l'action devient maintenant : $\boxed{\text{Réduire par } A \rightarrow \alpha}$. Uniquement lorsque le symbole d'entrée est **a**.

5 Définition et construction des items LR(1)

On travaillera sur cet exemple :

- S' → S
- S → CC
- C → aC
- C → d

5.1 Un item LR(1) :

Une production avec un . à droite, dans un contexte particulier :

S → .CC,$

5.2 Un état de l'automate LR(1)

C'est un ensemble d'items LR(1) obtenues par fermeture

5.3 Calcul des fermetures

Fermeture(I_i) =

- Placer tous les items de I_i dans fermeture (I_i)
  - Si $[A \rightarrow \alpha.B\beta, a] \in I\ i$
  - Si $[B \rightarrow \gamma] \in G$
  - Si $b \in$ Premier(βa)
- Alors ajouter

- $[B \to .\gamma, b]$ dans fermeture (I_i)
- Sauf si déjà dedans

On construit un automate LR(1) avec des états comprenant des items LR(1) (c'est à dire avec des contextes).

Automate :

- Etat initial
- Etat de satisfaction
- Transistions
- (état d'erreur)

5.4 Construction de la table LR(1)

	T	N
état $\updownarrow$	Actions	Transistions

- Si $[A \to \alpha.a\beta, b] \in I_i$: ACION[I_i, a]= "dj" où j est tel que Transistion(I_i, a) = I_j
- Si $[A \to \alpha., a]$ = "réduire par A→ α"
- Si $[S' \to S., \$] \in I_i$ alors ACTION(I_i, \$) = "OK"
- Autre : Erreur

I_0 =

- S' → .S, \$
- S → .CC, \$ (Premier() =
- C → .aC,[ad](Premier(C)={ad}
- C → .d, [ad]

I_1 AV(I_0, S)

- S' → S.,\$

I_2 = AV(I_0, C)

- S → C.C,\$
- C → .aC,\$
- C\$ → .d,\$

I_3 = AV(I_0, a) = AV(I_3, a)

- C → a.C, [ad]
- C → .aC, [ad]
- C → .d, [ad]

I_4 = AV(I_0, d) = AV(I_3, d)

- C $\rightarrow$ d.,[ad]

I_5 = AV(I_2 C)

- S $\rightarrow$ CC., \$

I_6 = AV(I_2, a) = AV(I_6, a)

- C $\rightarrow$ a.C, \$
- C $\rightarrow$ .aC, \$
- C $\rightarrow$ .d, \$

I_7 = AV(I_2, d) = AV(I_6, d)

- C $\rightarrow$ d., \$

I_8 = AV(I_3, C)

- C $\rightarrow$ aC., [ad]

I_9 = AV(I_6, C)

- C $\rightarrow$ aC.,\$

Table LR(1)

S' $\rightarrow$ S (r_0) S $\rightarrow$ CC (r_1) C $\rightarrow$ aC (r_2) C $\rightarrow$ d (r_3)

	a	d	\$	S	C	S'
I_0	d_3	d_4		1	2	
I_1			OK			
I_2	d_6	d_7			5	
I_3	d_3	d_4			8	
I_4	r_3	r_3				
I_5			r_1			
$\cancel{I_6}$	d_6	d_7			9	
$\cancel{I_7}$			r_3			
I_8	r_2	r_2				
$\cancel{I_9}$			r_2			
I_{47}	r_3	r_3	r_3			
I_{89}	r_2	r_2	r_2			
I_{36}	r_{36}	r_{47}			89	

Pas de conflit $\Rightarrow$ LR(1)

Analyseur LALR(1) : Nb état LR(1) > nb état SLR(1)

Essayer de réduire la taille de la table et tendre vers la taille d'une table d'un automate LR(0)

Etats ayant les mêmes noyau, et des contextes différents :

- I_4-$I_7 = I_{47}$
- I_8-$I_9 = I_{89}$
- I_3-I_6