Compilation

Analyse syntaxique : analyse ascendante

Rappels

- Symboles terminaux : lettre en minuscule (a,b,c,d,e,f,g,h)
- Symboles non-terminaux : lettre en majuscule
- Séquences de terminaux et de non-terminaux: lettres grecques (α,β,χ, ..)
- Séquences de terminaux: lettres en minuscule (u,v,w,x,g,z)

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR
  - Variantes de LR
    - SLR
    - LALR
  - Conflits LR

Analyse ascendante

- **Principe :**
  - Construire l'arbre de dérivation en démarrant des unités lexicales (feuilles) d'une chaîne jusqu'à arriver à la racine (axiome de départ)
    - Réductions successives jusqu'à retrouver l'axiome de départ de la grammaire
  - On procède à la réduction des symboles en allant de gauche à droite
  - Dés fois, plusieurs règles peuvent être utilisées pour réduire une suite de symboles

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

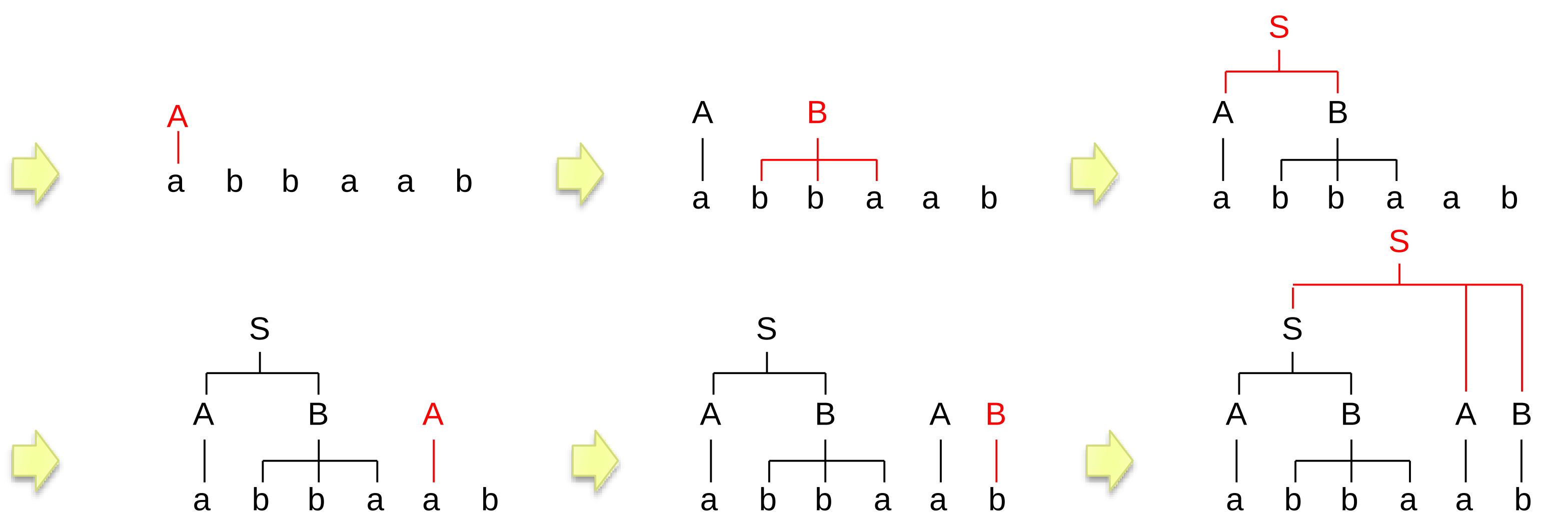
Réductions

- Une réduction c'est une dérivation prise dans le sens inverse:
  - On remplace une séquence de terminaux et/ou de non-terminaux par un non-terminal
  - La séquence doit être reconnue par la partie droite d'une production

Analyse ascendante

- **Exemple 1:**
  - S → AB | SAB
  - A → a | aab
  - B → b | bba

m=abbaab



On a réussi à retrouver la racine (axiome de départ) après la série de réduction. Donc, l'expression "abbaab" est reconnue par la grammaire

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Analyse ascendante par décalage-réduction

- C 'est une méthode d'analyse ascendante qui:
  - Utilise une pile pour sauvegarder les symboles de la grammaire (terminaux ou non-terminaux) déjà analysés
  - Se base sur deux types d'actions:
    - Empiler le symbole courant de la séquence à traiter au sommet de la pile (phase de décalage/lecture) tant qu'on n'a pas reconnu la partie droite d'une règle à appliquer
    - Réduite un membre droit d'une règle qui est dans la pile et empiler le membre gauche de cette règle (phase de réduction)

Analyse ascendante par décalage-réduction

- Conditions d'arrêt:
  - Si on termine avec le mot vide (toute la séquence est traitée) et que le sommet de la pile comporte l'état initial alors, la séquence est acceptée (reconnue par la grammaire)
  - Si on n'arrive pas à reconnaître la partie droite d'une règle de production ou si on termine les symboles de la séquence et que la pile contient d'autres symboles autres que l'axiome alors, la séquence est rejetée (non reconnue par la grammaire)

Analyse ascendante

- Exemple:
  - En utilisant une pile pour vérifier que la séquence "aabbac" est valide par rapport à la grammaire:
    - S → aaSSac | b

Pile	Entrée	Action
ε	aabbac	Décalage
a	abbac	Décalage
aa	bbac	Décalage
aab	bac	Réduction: S→b
aaS	bac	Décalage
aaSb	ac	Réduction: S→b
aaSS	ac	Décalage
aaSSa	c	Décalage
aaSSac	ε	Réduction: S→aaSSac
S	ε	Succès

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Analyse syntaxique LR

- C'est une méthode d'analyse ascendante qui se base sur les automates à pile

- Les symboles de la chaîne d'entrée sont lus de gauche à droite (d'où le L pour *Left*)

- On construit par réduction les parties droites des règles de la grammaire (d'où le R pour *Right*)

Tout comme les analyseurs LL, les analyseurs LR sont guidés par une table d'analyse

Analyse syntaxique LR

- Comment vérifier si une grammaire est LR?
- On parle plutôt de grammaires LR(K)
- Une grammaire est LR(k) si:
  - La connaissance de K terminaux après le symbole courant dans la chaîne à traiter (*look-ahead*) permet de décider de manière unique comment traiter le cas (réduction avec la partie gauche d'une règle de production ou décalage)
- Généralement, on travaille avec des grammaires LR(0) ou LR(1)

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Construire un automate à pile fini non déterministe: transformation

- Soit une grammaire G = (A,V, S, R) tel que:
  - A: alphabet (terminaux)
  - V : non-terminaux
  - S : axiome de départ
  - R: Règles de production
- On augmente la grammaire G par:
  - L'ajout d'un nouveau symbole terminal ($ par exemple)
  - L'ajout d'un nouvel axiome S'
  - L'ajout d'une nouvelle règle: S' → S$
- Toute entrée (chaîne) à analyser *m* est transformée en *m$*
- La réduction finale en S' est le seul cas d'acceptation

Construire un automate à pile fini non déterministe: transformation

- Exemple de grammaire :
  - $S \rightarrow AA$
  - $A \rightarrow aA \mid b$
- Grammaire transformée:
  - $S' \rightarrow S\$$
  - $S \rightarrow AA$
  - $A \rightarrow aA \mid b$

L'axiome de la grammaire était S. On a ajouté la règle $S' \rightarrow S\$$ et maintenant l'axiome c'est S' (la rencontre de cet axiome indique à l'analyseur syntaxique de terminer l'analyse)

Construire un automate à pile fini non déterministe: ce qu'on cherche à obtenir

- Pour une grammaire G, on cherche à construire une table semblable à la table ci-dessous:

Terminaux de G

Non-terminaux de G

États de l'automate LR(0)

	Actions			ALLER_À		
	Terminaux			Non-terminaux		
États		…			…	
		…			…	
…	…	…	…	…	…	…
		…			…	

États à empiler à la fin d'une réduction par le symbole non-terminal de la colonne correspondante (transitions de l'automate)

Action à effectuer (empiler/réduire) lorsqu'on est dans un état et qu'on a lu un symbole terminal

Comment calculer/déterminer les états?

Construire un automate à pile fini non déterministe: les items LR

- Détermination des états:
  - Chaque état est défini par un ou plusieurs items LR(0)
- Un item LR(0) appelé aussi item de G est une règle contenant un point (•) obtenue par la transformation d'une règle de G:
  - La première règle de G ($S' \rightarrow S\$$) est transformée en une règle:
    - $S \rightarrow \bullet\, S\$$
  - Chaque règle restante de G de la forme $A \rightarrow \alpha\, \beta$ est transformée en une règle:
    - $A \rightarrow \alpha \bullet \beta$
  - Une règle $A \rightarrow \varepsilon$ est transformée en:
    - $A \rightarrow \bullet$

Les transitions de l'automate sont étiquetées par des symboles terminaux (mot vide inclus) ou non-terminaux

Construire un automate à pile fini non déterministe: les items LR

- Intuition de la règle du type $S' \rightarrow \bullet\ S\$$:
  - On s'apprête à reconnaître un mot qui dérive de S (l'axiome)
- Intuition des règles $A \rightarrow \alpha \bullet \beta$:
  - On a reconnu une séquence qui dérive de α, il ne reste qu'à reconnaître une séquence dérivée de β pour pouvoir affirmer qu'on a reconnu un mot qui dérive de A (matérialisé par la réduction de $\alpha\beta$ en A)

Construire un automate à pile fini non déterministe: états de l'automate LR

- Exemple de génération des états de l'automate LR(0) :

Règle originale	Règle transformée	Déjà rencontré	S'attendre à
E → E+T	E → •E+T		Retrouver E+T
	E → E•+T	E	Retrouver +T
	E → E+•T	E+	Retrouver T
	E → E+T•	E+T	Réduite E+T par E

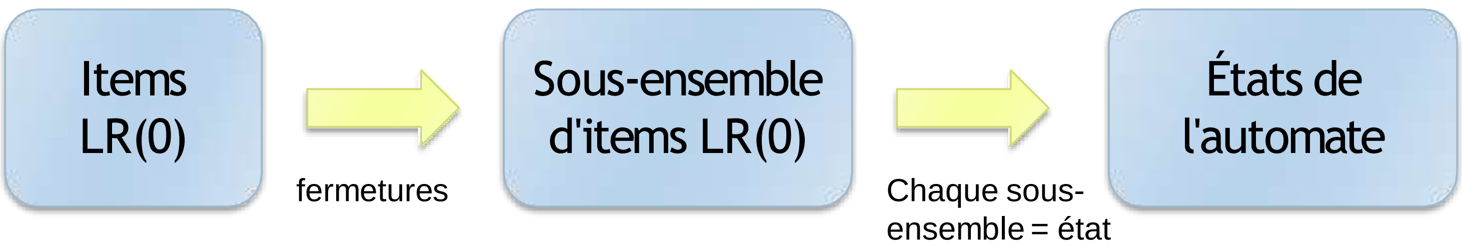
Construire un automate à pile fini non déterministe: états de l'automate LR

- État initial de l'automate c'est l'état qui:
  - Contient la règle : $S' \rightarrow \bullet\ S\$$
- États finaux de l'automate sont les états qui:
  - Contiennent au moins un item LR(0) de la forme : $A \rightarrow \alpha\ \bullet$

Comment calculer les items et les répartir en sous-ensembles qui formeront les états de l'automate LR(0)?

Fermeture d'un ensemble d'items LR(0)

- Les items LR(0), répartis en sous-ensembles, forment les états de l'automate, et sont donnés par des fermetures de sous-ensembles d'items LR(0)



Comment se calculent les fermetures des items LR(0)?

Calcul de la fermeture de l'ensemble d'items I

- Soit FERMETURE(I) la fermeture de I à calculer par rapport à une grammaire G

- FERMETURE(I) se calcule en appliquant les règles suivantes:
  1. Ajouter tout item de I à FERMETURE(I) :
     - FERMETURE(I) $\leftarrow$ I
  2. Si $A \rightarrow \alpha \bullet B\beta$ est dans FERMETURE(I) et que la production $B \rightarrow \gamma$ appartient à G alors ajouter $B \rightarrow \bullet\gamma$ à FERMETURE(I) :
     - FERMETURE(I) $\leftarrow$ FERMETURE(I) U $\{B \rightarrow \bullet\gamma\}$
  3. Appliquer la deuxième règle jusqu'à ce qu'aucun nouvel item ne peut être ajouté à FERMETURE(I)

Calcul de la fermeture de l'ensemble d'items I

- FERMETURE(I) se calcule en appliquant les règles suivantes:
  1. Ajouter tout item de I à FERMETURE(I) :
     - FERMETURE(I) ← I
  2. Si A → α•Bβ est dans FERMETURE(I) et que la production B→ γ appartient à G alors ajouter B→ •γ à FERMETURE(I) :
     - FERMETURE(I) ← FERMETURE(I) U {B→ •γ}
  3. Appliquer la deuxième règle jusqu'à ce qu'aucun nouvel item ne peut être ajouté à FERMETURE(I)

- Exemple:

  S' → S$

  S → AA

  A → aA | b

- Le point de départ : I={S'→•S$} }, alors le calcul de la fermeture de I se fait itérativement comme suit:
  - Application de la première règle:
    - FERMETURE(I) = {S' → •S$}
  - Application de la deuxième règle:
    - Pour la production S' → •S$ qui est de la forme A → α•Bβ, on y trouve une règle dans G qui est de la forme B→ γ et qui est: S → AA donc:
      - FERMETURE(I) = FERMETURE(I) U {S → •AA}

S'→ •S
S → •AA

Calcul de la fermeture de l'ensemble d'items I

- FERMETURE(I) se calcule en appliquant les règles suivantes:
  1. Ajouter tout item de I à FERMETURE(I) :
     - FERMETURE(I) ← I
  2. Si A → α•Bβ est dans FERMETURE(I) et que la production B→ γ appartient à G alors ajouter B→ •γ à FERMETURE(I) :
     - FERMETURE(I) ← FERMETURE(I) U {B→ •γ}
  3. Appliquer la deuxième règle jusqu'à ce qu'aucun nouvel item ne peut être ajouté à FERMETURE(I)

S'→ •S
S → •AA

- Application de la deuxième règle:
  - Pour la production S → •AA qui est de la forme A → α•Bβ, on y trouve deux règles dans G qui sont de la forme B→ γ :A → aA et A→ b , Donc:
    - FERMETURE(I) = FERMETURE(I) U {A → •aA, A → •b}

S'→ •S
S → •AA
A → •aA | •b

Calcul de la fermeture de l'ensemble d'items I

- Intuition derrière l'application de la deuxième règle
  - Si on à l'item$A \rightarrow \alpha \bullet B\beta$ cela veut dire qu'on a rencontré α et qu'on s'attende à retrouver $B\beta$
  - Comme B est un non-terminal, il sera certainement rencontré après avoir appliqué une réduction de la forme $B \rightarrow \bullet\gamma$
  - Ce qui justifie l'ajout de $B \rightarrow \bullet\gamma$ à la fermeture de I

Calcul de la fermeture de l'ensemble d'items I: algorithme

- Algorithme pour calculer la fermeture:
  - FermetureCalc(I)
    - J← I
    - Répéter
      - Pour(chaque item :A → α•Bβ de J) Faire
        - Pour(chaque production de G de la forme B→ γ)
          - Si(B→ •γ n'est pas dans J)Alors
            - J← J U {B→ •γ}
          - Fin
        - Fait
      - Fait
    - Jusqu'(à ce qu'aucun item n'est ajouté à J)
    - Retourner(J)
  - Fin

Calcul des états de l'automate LR(0)

- Les états (et les transitions) de l'automate LR(0) se calculent en utilisant la fonction ALLER_À() appelée encore GO_TO()

- ALLER_À() admet deux paramètres:
  - Un ensemble d'items I
  - Un symbole terminal ou non-terminal

- ALLER_À(I, X) retourne la fermeture de l'ensemble de tous les items de la forme $A \rightarrow \alpha X \bullet \beta$ tel que $A \rightarrow \alpha \bullet X \beta$ est dans I

Calcul des états de l'automate LR(0)

- **Intuition:**
  - La fonction ALLER_À(I, X) représente une transition par le symbole X de l'état représenté par l'ensemble des items I vers l'ensemble des items ALLER_À(I, X)

Calcul des états de l'automate LR(0)

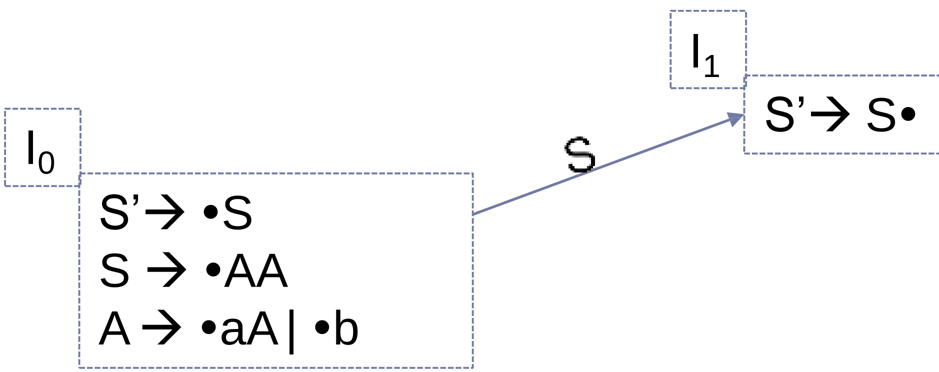
- Exemple:

L'état I_0 = FERMETURE($S' \rightarrow \bullet S$)

I_0

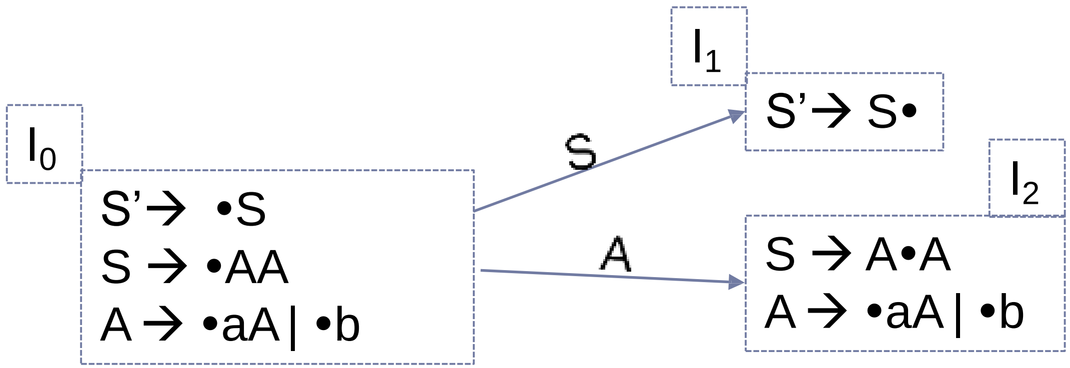
$S' \rightarrow \bullet S$
$S \rightarrow \bullet AA$
$A \rightarrow \bullet aA \mid \bullet b$

L'état I_1 = ALLER_À(I_0,S) = FERMETURE($S' \rightarrow S\bullet$)



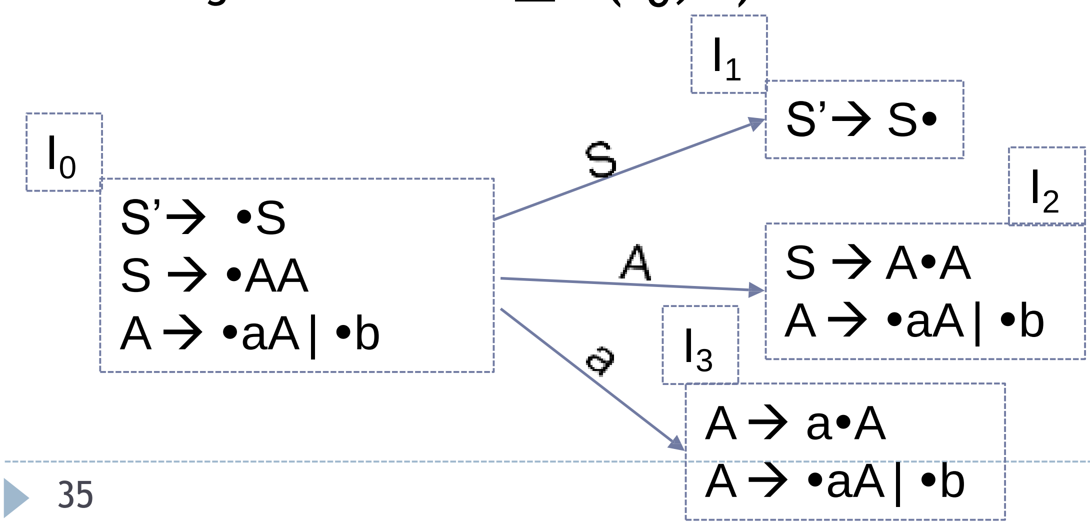
Calcul des états de l'automate LR(0)

L'état I_2 = ALLER_À(I_0,A) = FERMETURE($S \rightarrow A \bullet A$)



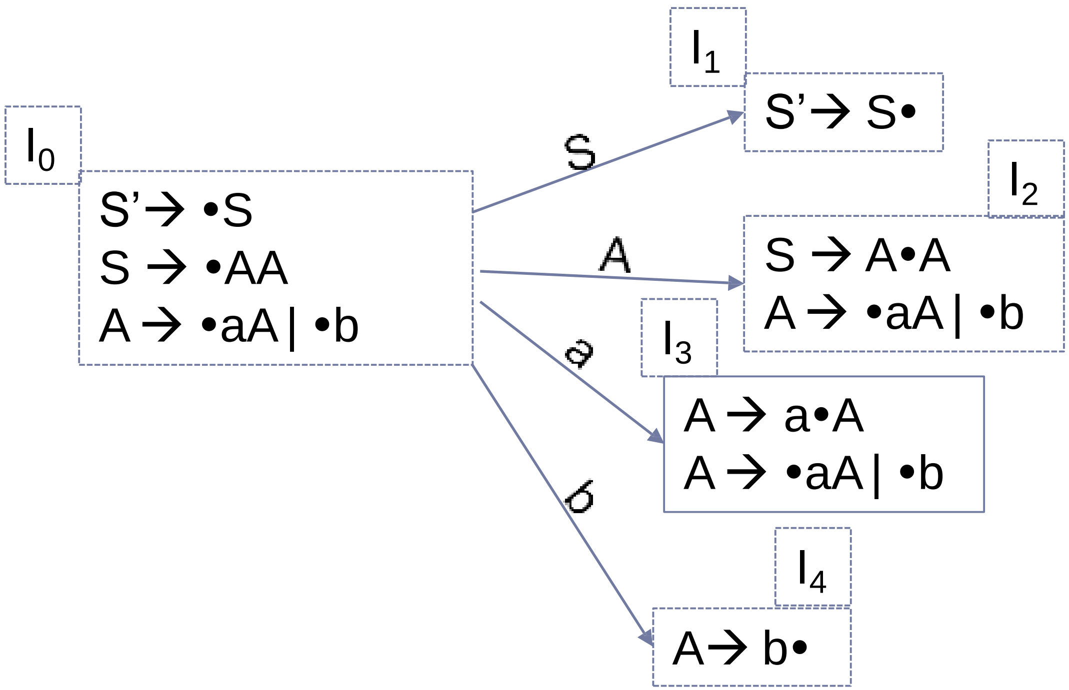
(il faut développer la FERMETURE après ALLER_À)

L'état I_3 = ALLER_À(I_0,a) = FERMETURE($A \rightarrow a \bullet A$)



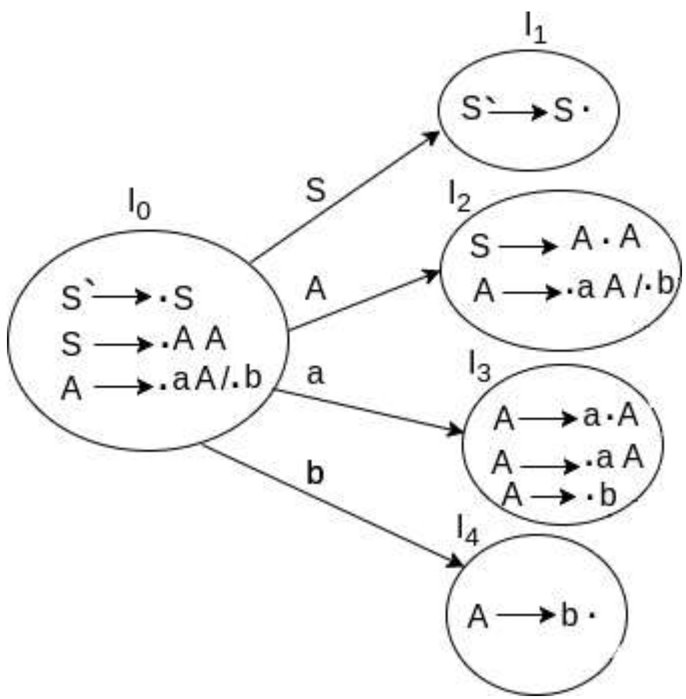
Calcul des états de l'automate LR(0)

L'état I_4 = ALLER_À(I_0,b) = FERMETURE($A \longrightarrow b\bullet$)



Calcul des états de l'automate LR(0)

L'état I_4 = ALLER_À(I_0,b) = FERMETURE($A \rightarrow b\bullet$)



Calcul des états de l'automate LR(0)

L'état I_5 = ALLER_À(I_2,A) = FERMETURE($S \rightarrow AA\bullet$)

ALLER_À(I_2, a) = FERMETURE($A \rightarrow a\bullet A$) = L'état I_3

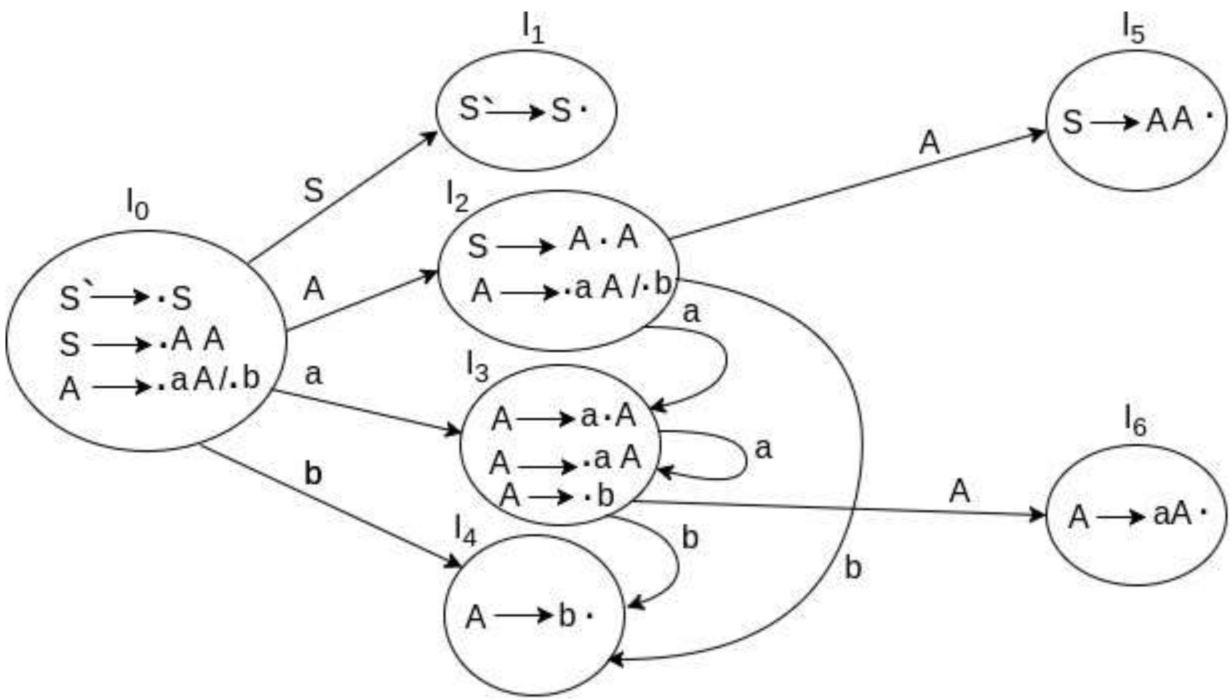
ALLER_À(I_2,b) = FERMETURE($A \rightarrow b\bullet$) = L'état I_4

L'état I_6 = ALLER_À(I_3,A) = FERMETURE($A \rightarrow aA\bullet$)

ALLER_À(I_3,a) = FERMETURE($A \rightarrow a\bullet A$) = L'état I_3

ALLER_À(I_3,b) = FERMETURE($A \rightarrow b\bullet$) = L'état I_4

Calcul des états de l'automate LR(0)



Calcul de tous les états (et transitions) de l'automate LR(0)

- On appelle l'ensemble de tous les états de l'automate LR(0) d'une grammaire G la collection des sous-ensembles d'items LR(0) (dénotée C) de G', la grammaire augmentée de G

- C est calculé par l'algorithme ITEMS(G) tel que présenté à la page suivante

C est aussi appelée collection canonique

Calcul de tous les états (et transitions) de l'automate LR(0)

- ITEMS(G)
  - C= {FERMETURE(S'→•S$)}
  - Répéter
    - Pour(chaque ensemble d'items I de C) Faire
      - Pour(chaque symbole terminal ou non-terminal X) Faire
        - Si(ALLER_À(I ,X) est ni vide ni dans C) Alors
          - C← C U {ALLER_À(I ,X)}
        - Fin
      - Fait
    - Fait
  - Jusqu'(à ce qu'aucun nouvel ensemble d'items n'est ajouté à C à la dernière itération de la boucle "Pour")
- Fin

Construire un automate à pile fini non déterministe

Maintenant que nous savons comment calculer l'automate LR(0) d'une grammaire, comment faire pour produire la table d'analyse?

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

	Actions			ALLER_À		
	Terminaux			Non-terminaux		
États		...			...	
		...			...	
...	...	...	...	...	...	...
		...			...	

Construction de la table LR(0)

- Nous allons faire référence au tableau composé des états et actions par ACTIONS et celui composé des non-terminaux avec les états par ALLER_À
- Étapes :
  1. Le signe $ est ajouté aux terminaux (tableau ACTIONS)
  2. Construire la collection canonique $C = \{I_0, I_1, \dots, I_n\}$
     - L'état j est associé à ensemble I_j
  3. La matrice ACTIONS est construite comme suit:
     - Si($A \rightarrow \alpha \bullet a\beta$ est dans I_j avec a terminal) Alors
       - ACTIONS[j, a] ← "Décaler:a" (abrégée par :"d:a")
     - Fin
     - Si($A \rightarrow \alpha \bullet$ est dans I_j) Alors
       - ACTIONS[j, *] ← "Réduire:$A \rightarrow \alpha \bullet$" (abrégée par :"r:$A \rightarrow \alpha \bullet$")
         - *: n'importe quel symbole terminal (concerne toutes les colonnes)
     - Fin
     - Si($S' \rightarrow S \bullet \$$ est dans I_j) Alors
       - ACTIONS[j, $] ← "Accepter"
     - Fin

Construction de la table LR(0)

	Actions			ALLER_À		
	Terminaux			Non-terminaux		
États		...			...	
		...			...	
...	...	...	...	...	...	...
		...			...	

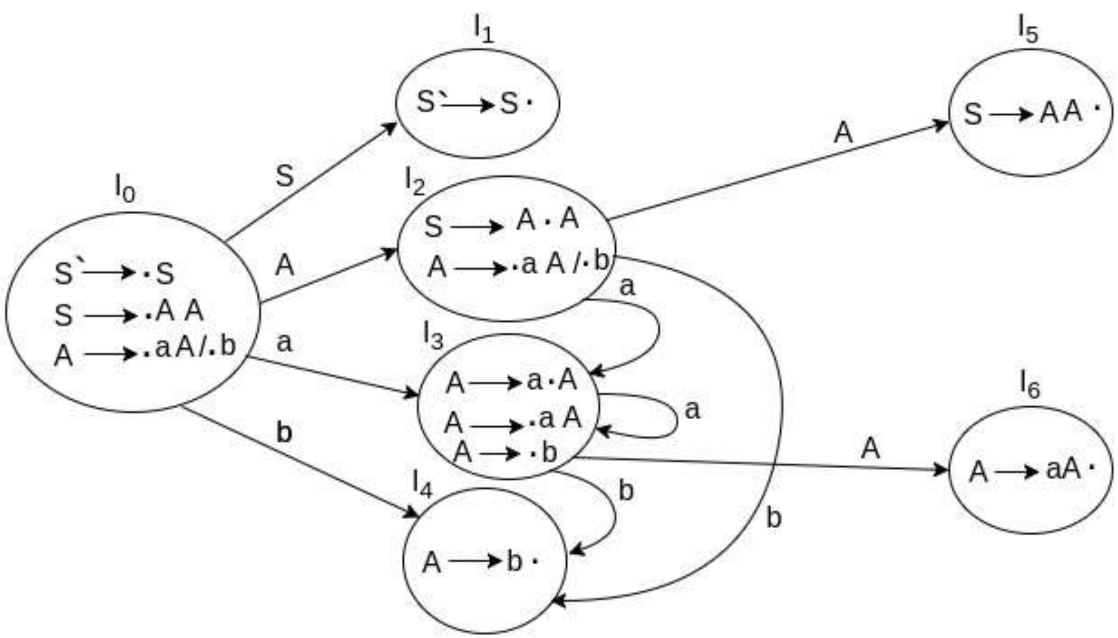
4. La matrice ALLER_À est construite comme suit:
   - Pour (chaque non-terminal A) Faire
     - Si(ALLER_À (I_j,A) = I_k) Alors
       - ALLER_À[j,A] ← k
     - Fin
5. Toutes les entrées des matrices ALLER_À et ACTIONS qui sont encore vides sont positionnées à "Erreur"
6. S'il y a des conflits (réduire/réduire ou décaler/réduire) dans la matrice ACTIONS, alors la grammaire n'est pas LR(0) et l'analyse syntaxique LR(0) ne peut être effectuée

Remarque

- Pour alléger l'écriture dans la table, une entrée "réduire:A→α" ou "r:A→α" seront écrites :"r:i« ou "ri" où i est un numéro attribué à la règle A→α
  - Autrement dit, on peut numéroter les règles de la grammaire et utiliser ces numéros à la place des règles

- Un décalage d:a peut être réduit à d, et dans ce cas le terminal a sera celui de la colonne à laquelle appartient la cellule

Automate complet



- **Étapes :**
  1. Le signe $ est ajouté aux terminaux (tableau ACTIONS)
  2. Construire la collection canonique $C = \{I_0, I_1, \ldots, I_n\}$
     - L'état j est associé à ensemble I_j
  3. La matrice ACTIONS est construite comme suit:
     - Si($A \rightarrow \alpha \bullet a\beta$ est dans I_j avec a terminal) Alors
       - ACTIONS[j, a] ← "Décaler:a" (abrégée par : "d:a")
     - Fin
     - Si($A \rightarrow \alpha \bullet$ est dans I_j) Alors
       - ACTIONS[j, *] ← "Réduire:$A \rightarrow \alpha \bullet$" (abrégée par : "r:$A \rightarrow \alpha \bullet$")
         - *: n'importe quel symbole terminal (concerne toutes les colonnes)
     - Fin
     - Si($Z \rightarrow E \bullet \$$ est dans I_j) Alors
       - ACTIONS[j, $] ← "Accepter"
     - Fin
  4. La matrice ALLER_À est construite comme suit:
     - Pour(chaque non-terminal A)Faire
       - Si(ALLER_À(I_j, A) = I_k) Alors
         - ALLER_À[j,A] ← k
       - Fin
  5. Toutes les entrées des matrices ALLER_À et ACTIONS qui sont encore vides sont positionnées à "Erreur"
  6. S'il y a des conflits (réduire/réduire ou décaler/réduire) dans la matrice ACTIONS, alors la grammaire n'est pas LR(0) et l'analyse syntaxique LR(0) ne peut être effectuée

Exemple de table LR(0)

- Table LR(0) associée à la grammaire donnée en exemple :

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
0 (I_0)	S3	S4			
1 (I_1)			accept		
2 (I_2)	S3	S4			
3 (I_3)	S3	S4			
4 (I_4)					
5 (I_5)					
6 (I_6)					

Terminaux → Shift État (Ex: S3)

Exemple de table LR(0)

- Table LR(0) associée à la grammaire donnée en exemple :

Règles numérotées:

0 S' → S$
1 S → AA
2 A → aA
3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
0				2	1
1					
2				5	
3				6	
4					
5					
6					

Non Terminaux → État (Ex:2)

Exemple de table LR(0)

- Table LR(0) associée à la grammaire donnée en exemple :

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
0					
1					
2					
3					
4	r3	r3	r3		
5	r1	r1	r1		
6	r2	r2	r2		

États finaux (Ex: I4 I5 I6) → appliquer la règle de réduction Ex:r3)

Exemple de table LR(0)

- Table LR(0) associée à la grammaire donnée en exemple :

Règles numérotées:

0 S' → S$
1 S → AA
2 A → aA
3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Terminaux → Shift État (Ex: S3) et Non Terminaux → État (Ex:2)
États finaux (Ex: I4 I5 I6) → appliquer la règle de réduction Ex:r3)

Construction de la table LR(0)

Comment exploiter la table d'analyse LR(0) pour analyser une séquence de symboles terminaux?

Plan

- Analyse ascendante
  - Analyse ascendante
  - Réductions
  - Décalage/réduction
- Analyse syntaxique LR
  - Principe
  - Étapes
    - Construire un automate à pile fini non déterministe
    - Construire la table d'analyse à partir de cet automate
    - Exploiter une table d'analyse LR

Algorithme d'un analyseur syntaxique LR(0)

- Soient le mot w$ à valider et a le premier symbole de w$
- La pile est initialisée avec l'état initial de l'automate
- Tant que(Vrai) Faire
  - Soit s l'état situé au sommet de la pile
  - Si(ACTIONS[s,a]=="décaler: t")Alors
    - Empiler t
    - Lire le prochain symbole et le mettre dans a
  - Sinon Si(ACTIONS[s, a]=="réduire:A→β")Alors
    - Dépiler |β| symboles
    - Soit t le nouveau sommet de la pile
    - Empiler ALLER_À[t,A]
    - Émettre la production A→β
  - Sinon Si(ACTIONS[s, a]=="accepter")
    - Arrêter et sortir
  - Sinon
    - Erreur et sortir
  - Fin
- Fait

|β| = nombre de symboles de β (cardinalité)

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	

- Soient le mot w$ à valider et a le premier symbole de w$
- La pile est initialisée avec l'état initial de l'automate
- Tant que(Vrai) Faire
  - Soit s l'état situé au sommet de la pile
  - Si(ACTIONS[s, a]=="décaler:t")Alors
    - Empiler t
    - Lire le prochain symbole et le mettre dans a
  - Sinon Si(ACTIONS[s, a]=="réduire:A→β")Alors
    - Dépiler |β| symboles
    - Soit t le nouveau sommet de la pile
    - Empiler ALLER_À[t, A]
    - Émettre la production A→β
  - Sinon Si(ACTIONS[s, a]=="accepter")
    - Arrêter et sortir
  - Sinon
    - Erreur et sortir
  - Fin

Fait

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3:A → b (|b|=1 → dépiler 2)
b$	0a3a3A	

Règles numérotées:

0 S' → S$
1 S → AA
2 A → aA
3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	

Règles numérotées:

0 S' → S$
1 S → AA
2 A → aA
3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	b à l'état 2 → empiler b et 4
$	0A2b4	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	b à l'état 2 → empiler b et 4
$	0A2b4	$ à l'état 4 → r3
$	0A2A	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	b à l'état 2 → empiler b et 4
$	0A2b4	$ à l'état 4 → r3
$	0A2A	A à l'état 2 → 2
$	0A2A5	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	b à l'état 2 → empiler b et 4
$	0A2b4	$ à l'état 4 → r3
$	0A2A	A à l'état 2 → 2
$	0A2A5	$ à l'état 5 → r1
$	0S	

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Exemple d'analyse syntaxique en utilisant la table LR(0)

- Nous allons utiliser la table d'analyse LR(0) construite dans l'exemple pour analyser la chaîne "aabb"

Entrée	Pile	Règle
aabb$	0	Empiler a et État courant = 3
abb$	0a3	Empiler a et État courant = 3
bb$	0a3a3	Empiler b et État courant = 4
b$	0a3a3b4	r3: A → b (|b|=1 → dépiler 2)
b$	0a3a3A	A à l'état 3 → 6
b$	0a3a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0a3A	A à l'état 3 → 6
b$	0a3A6	b à l'état 6 → r2 (|aA| =2, dépiler 4)
b$	0A	A à l'état 0 → 2
b$	0A2	b à l'état 2 → empiler b et 4
$	0A2b4	$ à l'état 4 → r3
$	0A2A	A à l'état 2 → 2
$	0A2A5	$ à l'état 5 → r1
$	0S	S à l'état 0 → 1
$	0S1	$ à l'état 1 → Accepter

Règles numérotées:

0 S' → S$

1 S → AA

2 A → aA

3 A → b

	Actions			ALLER_À	
	Terminaux			Non-Terminaux	
État	a	b	$	A	S
I_0	S3	S4	Err	2	1
I_1	Err	Err	accept	Err	Err
I_2	S3	S4	Err	5	Err
I_3	S3	S4	Err	6	Err
I_4	r3	r3	r3	Err	Err
I_5	r1	r1	r1	Err	Err
I_6	r2	r2	r2	Err	Err

Références

- A.Aho, R. Sethi, J.Ullman,"Compilateurs : principes, techniques et outils". Deuxième édition, Pearson Education. 2007 (chapitre 4)
- Andrew W. Appel and Jens Palsberg, "Modern Compiler Implementation in Java, Second Edition". Second Edition, Cambridge University Press, 2002
- Ronald Mak, "Writing Compilers and Interpreters: A Modern Software Engineering Approach Using Java". Third Edition, John Wiley & Sons, 2009