Compilation

CM2 - Analyse syntaxique

ISTIC, Université de Rennes 1
Sebastien.Ferre@irisa.fr

COMP, M1 info

Plan

Plan

Rappels

- langage = ensemble de mots
- mot = suite de caractères/symboles/terminaux
- hiérarchie de langages
  - type 3 : langages dits réguliers
    - engendrés par des expressions régulières
    - reconnus par des automates finis
  - type 2 : langages dits algébriques
    - engendrés par des grammaire hors-contexte
    - reconnus par des automates à pile

Des grammaires aux automates

Problème

Comment construire un automate à pile M à partir d'une grammaire hors-contexte G tel que $\mathcal{L}(M) = \mathcal{L}(G)$?

- quels états (Q) ?
- quelles transitions (Δ) ?

- grammaire = spécification du langage (génération)
- automate = implémentation du langage (reconnaissance)

C'est un problème de compilation !

- langage source : grammaires
- langage cible : automates à pile
- préservation de la sémantique : $\mathcal{L}(G) = \mathcal{L}(M)$
- efficacité : automate déterministe, si possible

Des grammaires aux automates

Problème

Comment construire un automate à pile M à partir d'une grammaire hors-contexte G tel que $\mathcal{L}(M) = \mathcal{L}(G)$?

- quels états (Q) ?
- quelles transitions (Δ) ?

- grammaire = spécification du langage (génération)
- automate = implémentation du langage (reconnaissance)

C'est un problème de compilation !

- langage source : grammaires
- langage cible : automates à pile
- préservation de la sémantique : $\mathcal{L}(G) = \mathcal{L}(M)$
- efficacité : automate déterministe, si possible

Des grammaires aux automates

Décomposition du problème :

1. traduction : dérivation d'un automate des items non-contextuels (AINC), indéterministe
2. optimisation : réduction de l'indéterminisme par application de restrictions
   - différentes restrictions
   - différents types d'analyses pour différents types de grammaires
     ex. : descendante, ascendante, tabulée

Grammaire augmentée

Forme de la grammaire : on suppose

- que l'axiome S n'apparaît pas à droite des règles
- que la fin des phrases est facilement reconnaissable

Si ce n'est pas le cas, on augmente la grammaire

Definition (grammaire augmentée)

$G' = (V'_T, V'_N, S', P')$ est la grammaire augmentée de $G = (V_T, V_N, S, P)$ où :

- $V'_T = V_T \cup \{\$\}$ ajout symbole de fin
- $V'_N = V_N \cup \{S'\}$ nouvel axiome
- $P' = P \cup \{S' \rightarrow S\ \$\}$ règle pour le nouvel axiome

Grammaire augmentée

Forme de la grammaire : on suppose

- que l'axiome S n'apparaît pas à droite des règles
- que la fin des phrases est facilement reconnaissable

Si ce n'est pas le cas, on augmente la grammaire

Definition (grammaire augmentée)

$G' = (V'_T, V'_N, S', P')$ est la grammaire augmentée de $G = (V_T, V_N, S, P)$ où :

- $V'_T = V_T \cup \{\$\}$ ajout symbole de fin
- $V'_N = V_N \cup \{S'\}$ nouvel axiome
- $P' = P \cup \{S' \to S\ \$\}$ règle pour le nouvel axiome

Exemple de grammaire augmentée

$$0 : S' \to S\ \$$$

$$1 : S \to X\ Y$$

$$2 : X \to a\ X$$
$$3 : X \to c$$

$$4 : Y \to b$$

Items non-contextuels

L'AINC est basé sur la notion d'item (non-contextuel) (INC) qui représente un corps de règle partiellement reconnu.

Definition (item non-contextuel (INC))

Si $A \to \alpha\beta$ est une production d'une grammaire G alors $[A \to \alpha \bullet \beta]$ est un INC.

- α est l'histoire de l'item : ce qui a été reconnu
- β est le futur de l'item : ce qui reste à reconnaître pour avoir reconnu un A
- $[A \to \alpha\bullet]$ est un item complet
- $\mathcal{I}_G$ est l'ensemble des INC de G

Exemple : . . . .

Les items comme états de l'automate des items

Deux items particuliers :

- l'item $[S' \to \bullet S\$]$ correspond au début de l'analyse
  rien de reconnu
- l'item $[S' \to S \bullet \$]$ correspond à la fin de l'analyse
  une phrase S complète a été reconnue

Ce qu'on en déduit :

- ces items représentent l'état initial et l'état final
- les états de l'AINC sont des INC (items) !
- l'AINC va manipuler des piles d'items !

Automate des items non-contextuels

Definition

Soit $G = (V_T, V_N, S, P)$, et sa grammaire augmentée G', on appelle automate des items non-contextuels l'automate à pile $M_G = (V'_T, \mathcal{I}_G, \delta, [S' \to \bullet S\$], \{[S' \to S \bullet \$]\})$ où
δ est l'ensemble des transitions d'un des trois types suivants :

- (E)xpansion : futur : $Y\gamma \Rightarrow \alpha\gamma$
  pour tout $(Y \to \alpha) \in P$, indéterministe
  $\delta([X \to \beta \bullet Y\gamma], \epsilon) = [Y \to \bullet\alpha][X \to \beta Y \bullet \gamma]$
- (L)ecture (ou (D)écalage) : histoire : $\beta \Rightarrow \beta a$
  $\delta([X \to \beta \bullet a\gamma], a) = [X \to \beta a \bullet \gamma]$ déterministe
- (R)éduction : histoire : $\beta\alpha \Rightarrow \beta Y$
  $\delta([Y \to \alpha\bullet][X \to \beta Y \bullet \gamma], \epsilon) = [X \to \beta Y \bullet \gamma]$ déterministe

Analogie

Analogie avec l'exécution d'un programme procédural :

production	$\rightarrow$	définition de procédure
pile d'items	$\rightarrow$	pile d'appels
$\bullet$	$\rightarrow$	compteur ordinal
expansion	$\rightarrow$	appel de procédure Y de corps α
lecture	$\rightarrow$	exécution d'une instruction a
réduction	$\rightarrow$	retour de procédure Y

Indéterminisme si plusieurs productions α pour un même non-terminal Y
plusieurs définitions d'une même procédure

Arbre de dérivation

Construction possible de l'arbre de dérivation

- pendant l'exécution de l'automate
- pendant le processus de reconnaissance du mot

L'arbre «pousse» avec les expansions :

- Expansion : . . . . . .
- Lecture : . . . . . .
- Réduction : . . . . . .

Exemple : grammaire et automate des items

......

Exemple : Analyse d’un mot avec l’AINC

……

Correction de l'automate des items

Theorem (préservation de la sémantique)

L'automate des INC M_G accepte un mot m ssi la grammaire G engendre m.

$$\mathcal{L}(M_G) = \mathcal{L}(G)$$

Efficacité de l'automate des items

- Cet automate a deux inconvénients :
  1. indéterminisme des expansions
  2. complexité des piles d'items
- Des analyseurs plus efficaces peuvent être obtenus en imposant des restrictions aux transitions
  - ⇒ restrictions au niveau des grammaires et des langages
  - ⇒ perte de complétude par rapport aux langages de type 2

Remarque

On recherche un compromis entre efficacité de l'analyse et expressivité de la famille d'analyseurs.

Analyse syntaxique descendante

- Le principe de l'analyse descendante est de construire l'arbre de dérivation
  1. en partant de la racine et
  2. de la gauche vers la droite
- Plutôt que de procéder par essai-erreur, on va tenter de prédire la structure de l'arbre en fonction des symboles terminaux lus
- En terme d'automate des items, le parti pris est de ne regarder que le futur des items, qui correspond aux parties de l'arbre restant à construire

Analyse syntaxique descendante

- Le principe de l'analyse descendante est de construire l'arbre de dérivation
  1. en partant de la racine et
  2. de la gauche vers la droite
- Plutôt que de procéder par essai-erreur, on va tenter de prédire la structure de l'arbre en fonction des symboles terminaux lus
- En terme d'automate des items, le parti pris est de ne regarder que le futur des items, qui correspond aux parties de l'arbre restant à construire

Analyse syntaxique descendante

- Le principe de l'analyse descendante est de construire l'arbre de dérivation
  1. en partant de la racine et
  2. de la gauche vers la droite
- Plutôt que de procéder par essai-erreur, on va tenter de prédire la structure de l'arbre en fonction des symboles terminaux lus
- En terme d'automate des items, le parti pris est de ne regarder que le futur des items, qui correspond aux parties de l'arbre restant à construire

Simplification de l'automate des items

On va donc «oublier» l'histoire et la tête des items non-contextuels dans la relation de transition δ :
$[X \rightarrow \alpha \bullet \beta] \rightsquigarrow [\beta]$

- Expansion : $\delta([Y\gamma], \epsilon) = [\alpha][\gamma]$, pour tout $(Y \rightarrow \alpha) \in P$
- Lecture : $\delta([a\gamma], a) = [\gamma]$
- Réduction : $\delta([][\gamma], \epsilon) = [\gamma]$

Simplification de l'automate des items

Dans chaque transition, on regarde au plus un symbole de l'item en somme de pile.

- $\Rightarrow$ on peut oublier les [ ] délimitant les items
- $\Rightarrow$ une pile de symboles suffit
  au lieu d'une pile de séquences de symboles (items)

Transitions simplifiées, sur piles de symboles $(V_T \cup V_N)$:

- transition d'un ancien futur vers un nouveau futur
- Expansion : $\delta(Y\gamma, \epsilon) = \alpha\gamma$, pour tout $(Y \to \alpha) \in P$
- Lecture : $\delta(a\gamma, a) = \gamma$
- Réduction : $\delta(\gamma, \epsilon) = \gamma$ inutile

On s'est ramené à un automate plus simple :

- analyse par expansion-lecture (plus de réduction)
- une pile de symboles (le futur) état = symbole !

Simplification de l'automate des items

Dans chaque transition, on regarde au plus un symbole de l'item en somme de pile.

$\Rightarrow$ on peut oublier les [ ] délimitant les items

$\Rightarrow$ une pile de symboles suffit
au lieu d'une pile de séquences de symboles (items)

Transitions simplifiées, sur piles de symboles ($V_T \cup V_N$) :

- transition d'un ancien futur vers un nouveau futur
- Expansion : $\delta(Y\gamma, \epsilon) = \alpha\gamma$, pour tout $(Y \to \alpha) \in P$
- Lecture : $\delta(a\gamma, a) = \gamma$
- Réduction : $\delta(\gamma, \epsilon) = \gamma$ inutile

On s'est ramené à un automate plus simple :

- analyse par expansion-lecture (plus de réduction)
- une pile de symboles (le futur) état = symbole !

Simplification de l'automate des items

Dans chaque transition, on regarde au plus un symbole de l'item en somme de pile.

$\Rightarrow$ on peut oublier les [ ] délimitant les items

$\Rightarrow$ une pile de symboles suffit
au lieu d'une pile de séquences de symboles (items)

Transitions simplifiées, sur piles de symboles $(V_T \cup V_N)$:

- transition d'un ancien futur vers un nouveau futur
- Expansion : $\delta(Y\gamma, \epsilon) = \alpha\gamma$, pour tout $(Y \to \alpha) \in P$
- Lecture : $\delta(a\gamma, a) = \gamma$
- Réduction : $\delta(\gamma, \epsilon) = \gamma$ inutile

On s'est ramené à un automate plus simple :

- analyse par expansion-lecture (plus de réduction)
- une pile de symboles (le futur) état = symbole !

Exemple d'analyse descendante

Avec le même exemple que précédemment : ......

Déterminisation de l'automate des items

Problème

On a simplifié l'automate, mais il reste le même indéterminisme sur les expansions

- choix de la production $Y \to \alpha$ pour un Y donné
- choix des productions 2 ou 3 pour expansions de X

Solution

- On va autoriser la transition d'expansion à consulter les premiers symboles de ce qui reste à lire
- Dans l'exemple, un seul symbole suffit :
  - si c'est un a, alors 2 : $X \to a\ X$
  - si c'est un c, alors 3 : $X \to c$
- On dit que la grammaire est LL(1)

Déterminisation de l'automate des items

Problème

On a simplifié l'automate, mais il reste le même indéterminisme sur les expansions

- choix de la production $Y \to \alpha$ pour un Y donné
- choix des productions 2 ou 3 pour expansions de X

Solution

- On va autoriser la transition d'expansion à consulter les premiers symboles de ce qui reste à lire
- Dans l'exemple, un seul symbole suffit :
  - si c'est un a, alors 2 : $X \to a\ X$
  - si c'est un c, alors 3 : $X \to c$
- On dit que la grammaire est LL(1)

Grammaires et langages LL(k)

Definition (grammaire LL(k))

Une grammaire est LL(k) si la lecture de au plus k symboles permet toujours de choisir la bonne production à expanser

- L = left-to-right scan — lecture de gauche à droite
- L = leftmost derivation — expansions du NT le plus à gauche

Definition (langage LL(k))

Un langage est LL(k) s'il est engendré par une grammaire LL(k).

Grammaires et langages LL(k)

Certaines grammaires (et langages) hors-contexte ne sont LL(k) pour aucun k

- exemple :
  1. S $\rightarrow$ Decl-fonction | Decl-procedure
  2. Decl-fonction $\rightarrow$ En-tete Corps-avec-return
  3. Decl-procedure $\rightarrow$ En-tete Corps-sans-return
- Le `return` d'une fonction peut se trouver à une distance arbitraire du début de la déclaration, selon la taille du corps

Le choix d'une analyse descendante déterministe impose donc des restrictions sur les langages (et leurs grammaires) que l'on peut analyser

Grammaires et langages LL(k)

Certaines grammaires (et langages) hors-contexte ne sont LL(k) pour aucun k

- exemple :
  1. S → Decl-fonction | Decl-procedure
  2. Decl-fonction → En-tete Corps-avec-return
  3. Decl-procedure → En-tete Corps-sans-return
- Le `return` d'une fonction peut se trouver à une distance arbitraire du début de la déclaration, selon la taille du corps

Le choix d'une analyse descendante déterministe impose donc des restrictions sur les langages (et leurs grammaires) que l'on peut analyser

Grammaires et langages LL(k)

1. Si un langage n'est pas LL(K), on peut le modifier :
   - S $\rightarrow$ `fun` Decl-fonction | `proc` Decl-procedure
2. Si une grammaire G n'est pas LL(k), mais que le langage $\mathcal{L}(G)$ l'est, on peut modifier la grammaire pour la rendre LL(k) sans modifier le langage engendré

Remarque

L'objet du TD1 sera de décider si une grammaire est LL(1), et si non, de la rendre LL(1) lorsque c'est possible (langage LL(1)).

Récursivité gauche

L'analyse descendante s'accomode mal des récursivités gauches car elles entrainent des boucles sans fin dans l'analyse.

- exemple : . . . . . .

Récursivité gauche

Cependant, il est toujours possible de transformer un récursivité gauche en une récursivité droite :

- avant : $X \to X\beta \mid \alpha$ $\quad X \to \alpha\beta\ldots\beta$
- après : $X \to \alpha X'$ et $X' \to \beta X' \mid \epsilon$

Compilateurs d'analyseurs descendants

Les outils suivants prennent en entrée la description d'une grammaire et produisent un analyseur descendant (avec certaines restrictions) :

- ANTLR (Java, C, C++, ...) : grammaires LL(k) DM et TP
- stream parsers (OCaml) : grammaires LL(1) DM et TP
- JavaCC (Java) : grammaires LL(k)
- DCG (Prolog) : grammaires LL(*) mais indéterministe

Analyse syntaxique ascendante

- Le principe de l'analyse ascendante est de construire l'arbre de dérivation
  - en partant des feuilles
  - de la gauche vers la droite
- En terme d'automate des items, le parti pris est de ne regarder que l'histoire des items, qui correspond aux parties de l'arbre déjà construites
- L'idée est de reconnaitre des corps de règle et de les remplacer par la tête de règle

Analyse syntaxique ascendante

- Le principe de l'analyse ascendante est de construire l'arbre de dérivation
  - en partant des feuilles
  - de la gauche vers la droite
- En terme d'automate des items, le parti pris est de ne regarder que l'histoire des items, qui correspond aux parties de l'arbre déjà construites
- L'idée est de reconnaitre des corps de règle et de les remplacer par la tête de règle

Simplification de l'automate des items

On oublie la tête et le futur des items dans la relation de transition $\delta : [X \to \alpha \bullet \beta] \rightsquigarrow [\alpha]$

- Expansion : $\delta([\beta], \epsilon) = [][\beta]$
- Lecture : $\delta([\beta], a) = [\beta a]$
- Réduction : $\delta([\alpha][\beta], \epsilon) = [\beta Y]$ si $(Y \to \alpha) \in P$

On «inverse» les piles d'items (fond de pile à gauche) et on les «applatit» en piles de symboles :

- Expansion : $\delta(\beta, \epsilon) = \beta$ inutile
- Lecture : $\delta(\beta, a) = \beta a$
- Réduction : $\delta(\beta\alpha, \epsilon) = \beta Y$ si $(Y \to \alpha) \in P$

Simplification de l'automate des items

On oublie la tête et le futur des items dans la relation de transition δ : $[X \to \alpha \bullet \beta] \rightsquigarrow [\alpha]$

- Expansion : $\delta([\beta], \epsilon) = [][\beta]$
- Lecture : $\delta([\beta], a) = [\beta a]$
- Réduction : $\delta([\alpha][\beta], \epsilon) = [\beta Y]$ si $(Y \to \alpha) \in P$

On «inverse» les piles d'items (fond de pile à gauche) et on les «applatit» en piles de symboles :

- Expansion : $\delta(\beta, \epsilon) = \beta$ inutile
- Lecture : $\delta(\beta, a) = \beta a$
- Réduction : $\delta(\beta\alpha, \epsilon) = \beta Y$ si $(Y \to \alpha) \in P$

Déterminisme de l'analyse ascendante

Plus d'indéterminisme sur les expansions, mais possiblement sur les réductions

- s'il existe une règle de la forme $Z \to \alpha$ (même corps)
- une transition de lecture est toujours possible (conflit avec réduction ?)

Exemple d'analyse ascendante

......

Plan

Devoir à la Maison (DM) - préparation TP

- Objectifs
  - découverte de l'outil choisi : ANTLR/OCaml
  - application des concepts vus en cours
  - préparation aux TPs
- travail par binôme, note de CC
- Rapport de 4 pages, pour le CM du 29 septembre
  - documents lus
  - installation et utilisation de ANTLR/OCaml
  - exemple de l'évaluateur d'expression
  - augmenter cet exemple avec
    - division : même priorité que multiplication
    - puissance : priorité supérieure à la multiplication et associativité à droite
  - résultats d'expériences