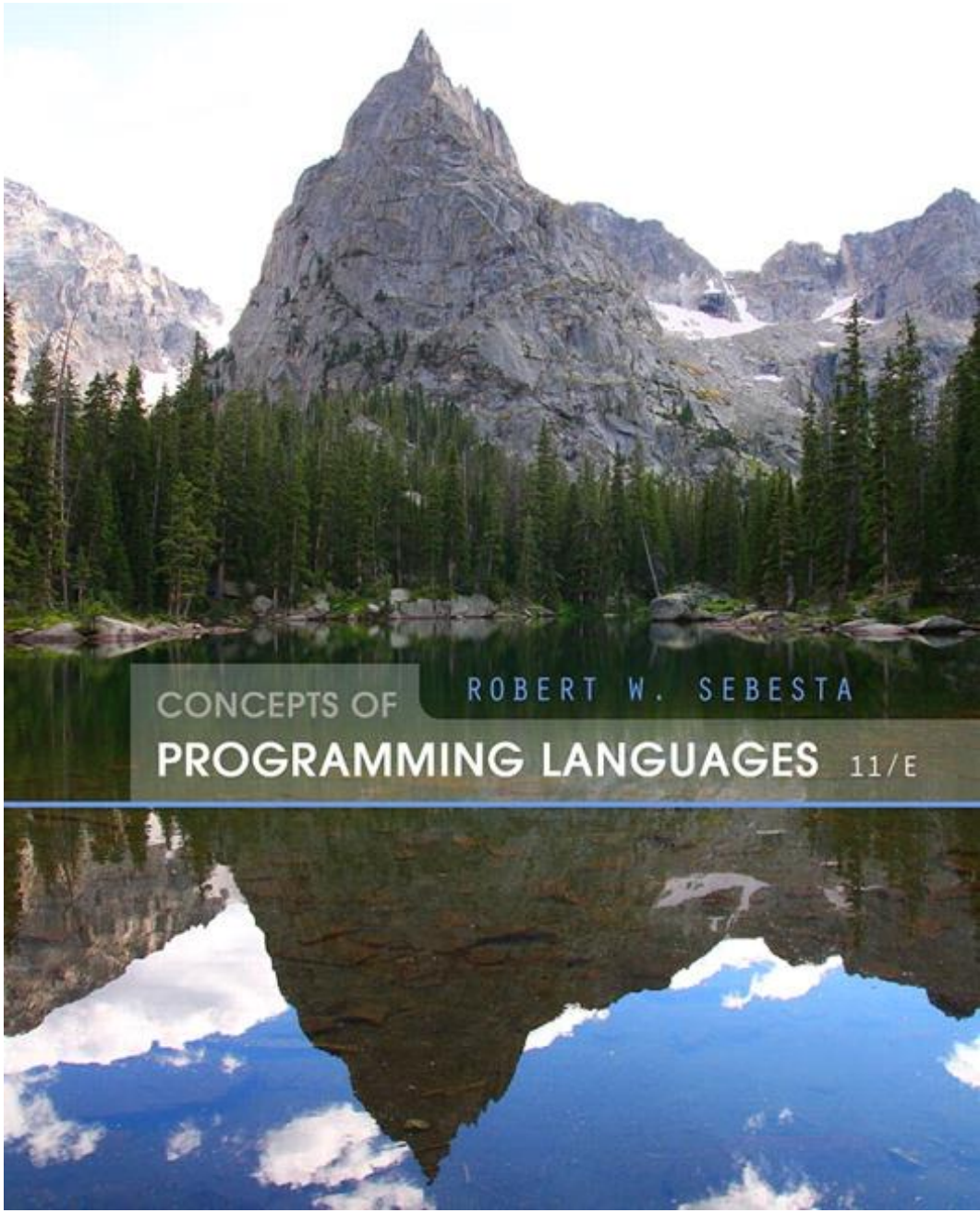
Κεφάλαιο 4

Λεκτική και συντακτική ανάλυση

Γκόγκος Χρήστος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)
Πανεπιστήμιο Ιωαννίνων



ISBN 0-321-49362-1

Κεφάλαιο 4 – Θεματικές Ενότητες

- Εισαγωγή
- Λεκτική Ανάλυση
- Το πρόβλημα της συντακτικής ανάλυσης
- Συντακτική ανάλυση αναδρομικής κατάβασης
- Συντακτική ανάλυση από κάτω προς τα πάνω

Εισαγωγή

- Τα συστήματα υλοποίησης γλωσσών θα πρέπει να αναλύουν τον πηγαίο κώδικα, ανεξάρτητα από τη συγκεκριμένη προσέγγιση υλοποίησης
- Σχεδόν όλες οι συντακτικές αναλύσεις βασίζονται σε μια τυπική περιγραφή του συντακτικού της πηγαίας γλώσσας (BNF)

Συντακτική Ανάλυση

- Το τμήμα συντακτικής ανάλυσης του επεξεργαστή μιας γλώσσας σχεδόν πάντα αποτελείται από δύο μέρη:
  - Ένα χαμηλού επιπέδου τμήμα που λέγεται λεκτικός αναλυτής (σε μαθηματική ορολογία, είναι ένα πεπερασμένο αυτόματο που βασίζεται σε μια κανονική γραμματική)
  - Ένα υψηλού επιπέδου τμήμα που ονομάζεται συντακτικός αναλυτής, ή parser (σε μαθηματική ορολογία, ένα push-down αυτόματο που βασίζεται σε μια γραμματική χωρίς συμφραζόμενα CFG ή BNF)

Πλεονεκτήματα χρήσης BNF για την περιγραφή του συντακτικού

- Παρέχει μια καθαρή και συνεπή περιγραφή του συντακτικού
- Ο parser μπορεί να βασιστεί απευθείας στο BNF
- Οι parsers που βασίζονται σε BNF συντηρούνται εύκολα

Λόγοι διαχωρισμού Λεκτικής και Συντακτικής Ανάλυσης

- *Απλότητα* : λιγότερο πολύπλοκες προσεγγίσεις μπορούν να χρησιμοποιηθούν για τη συντακτική ανάλυση – διαχωρίζοντας λεκτική και συντακτική ανάλυση απλοποιείται ο parser
- *Αποδοτικότητα* : ο διαχωρισμός επιτρέπει τη βελτιστοποίηση του λεκτικού αναλυτή
- *Μεταφερσιμότητα* : τμήματα του λεκτικού αναλυτή μπορεί να μην είναι μεταφέρσιμα, αλλά ο parser είναι πάντα μεταφέρσιμος

Λεκτική Ανάλυση

- Ένας λεκτικός αναλυτής είναι ένας μηχανισμός αντιστοίχισης μοτίβων για συμβολοσειρές
- Ένας λεκτικός αναλυτής αποτελεί το εμπρόσθιο τμήμα (front-end) για τον parser
- Αναγνωρίζει υποσυμβολοσειρές του πηγαίου προγράμματος που ανήκουν μαζί – λεξήματα (*lexemes)*
  - Τα λεξήματα ταιριάζουν με ένα μοτίβο χαρακτήρων, που σχετίζεται με μια λεκτική κατηγορία η οποία αποτελεί μια λεκτική μονάδα (*token)*
  - Το `sum` είναι ένα λέξημα, και το token του μπορεί να είναι `IDENT` (αναγνωριστικό)

Λεκτική ανάλυση (συνέχεια)

- Ο λεκτικός αναλυτής είναι συνήθως μια συνάρτηση που καλείται από τον parser όταν χρειάζεται το επόμενο token
- Υπάρχουν τρεις προσεγγίσεις στην κατασκευή ενός λεκτικού αναλυτή:
  - Συγγραφή μιας τυπικής περιγραφής των tokens και χρήση ενός εργαλείου λογισμικού που κατασκευάζει έναν καθοδηγούμενο από πίνακα λεκτικό αναλυτή από την περιγραφή
  - Σχεδίαση ενός διαγράμματος καταστάσεων που περιγράφει τα tokens και συγγραφή ενός προγράμματος που υλοποιεί το διάγραμμα καταστάσεων
  - Σχεδίαση ενός διαγράμματος καταστάσεων που περιγράφει τα tokens και χειροκίνητη κατασκευή μιας καθοδηγούμενης από πίνακα υλοποίησης του διαγράμματος καταστάσεων

Σχεδίαση Διαγράμματος Καταστάσεων

- Ένα απλοϊκό διάγραμμα καταστάσεων θα είχε μετάβαση από κάθε κατάσταση για κάθε χαρακτήρα της πηγαίας γλώσσας – ένα τέτοιο διάγραμμα θα ήταν πολύ μεγάλο!

Λεκτική ανάλυση (συνέχεια)

- Σε πολλές περιπτώσεις, οι μεταβάσεις μπορούν να συνδυαστούν για να απλοποιήσουν το διάγραμμα καταστάσεων
  - Όταν αναγνωριστεί ένα αναγνωριστικό, όλα τα κεφαλαία και τα πεζά γράμματα είναι ισοδύναμα
    - Χρήση μιας κλάσης χαρακτήρων που περιλαμβάνει όλα τα γράμματα
  - Όταν αναγνωρίζεται ένα ακέραιο κυριολεκτικό, όλα τα ψηφία είναι ισοδύναμα
    - χρήση μιας κλάσης ψηφίων που περιλαμβάνει όλα τα ψηφία

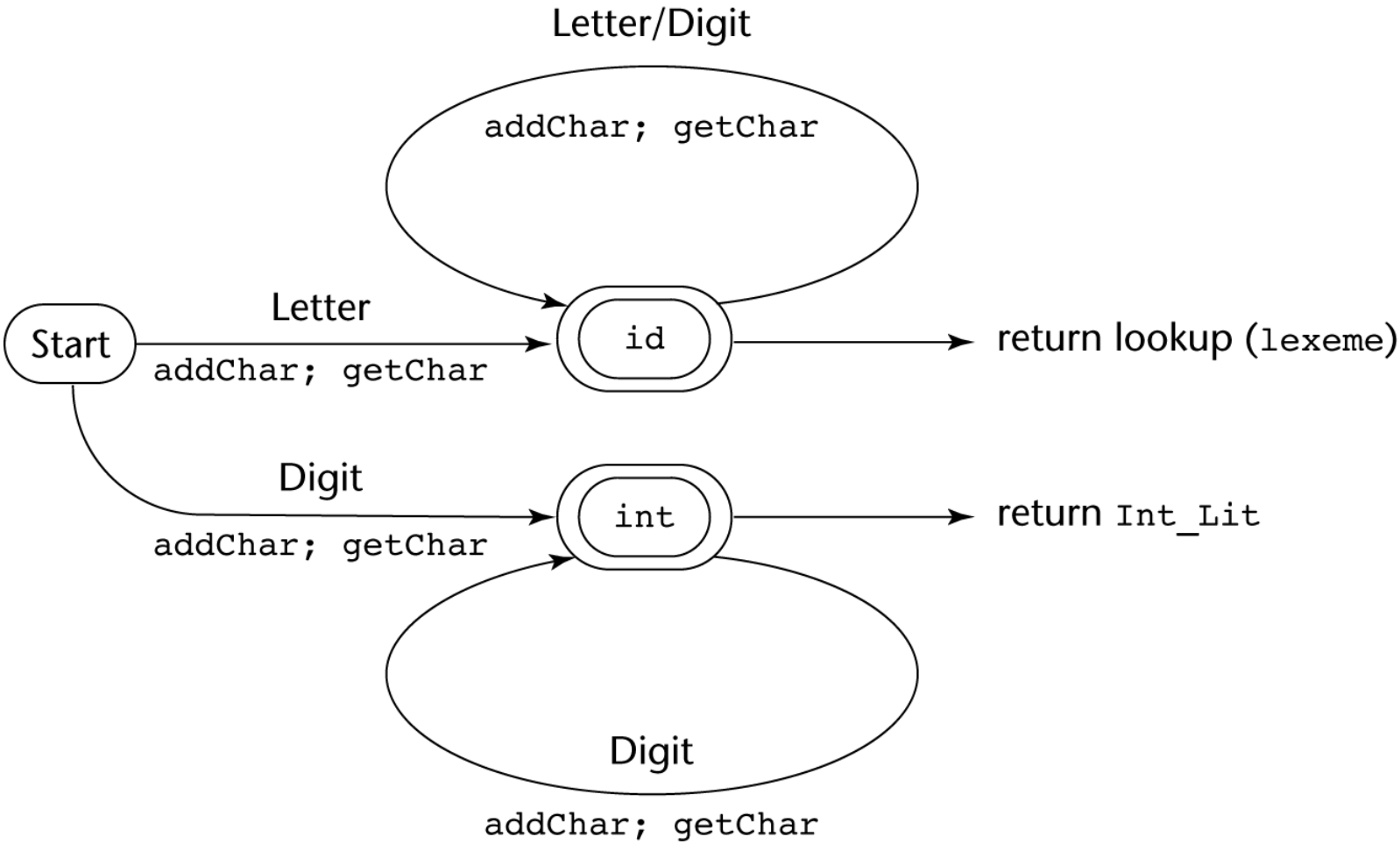
Λεκτική ανάλυση (συνέχεια)

- Οι δεσμευμένες λέξεις και τα αναγνωριστικά μπορούν να αναγνωρίζονται μαζί (αντί να υπάρχει ξεχωριστό στοιχείο του διαγράμματος για κάθε δεσμευμένη λέξη)
  - Χρήση ενός πίνακα αναζήτησης (lookup table) για να καθορίζεται εάν ένα πιθανό αναγνωριστικό είναι στην πραγματικότητα δεσμευμένη λέξη

Λεκτική ανάλυση (συνέχεια)

- Χρήσιμα υποπρογράμματα:
  - **`getChar`** – λαμβάνει τον επόμενο χαρακτήρα της εισόδου, τον τοποθετεί στο **`nextChar`**, καθορίζει την κλάση του και τοποθετεί την κλάση στην **`charClass`**
  - **`addChar`** – τοποθετεί τον χαρακτήρα από το **`nextChar`** στη θέση στην οποία συσσωρεύεται το λέξημα, **`lexeme`**
  - **`lookup`** – καθορίζει αν το λεκτικό στο **`lexeme`** είναι δεσμευμένη λέξη (επιστρέφει έναν κωδικό)

State Diagram



Λεκτικός αναλυτής

Υλοποίηση:

→ Δείτε `front.c` (σελ. 167–170)

– Ακολουθεί η έξοδος του λεκτικού αναλυτή `front.c` όταν χρησιμοποιείται στην είσοδο:
`(sum + 47) / total`

```
Next token is: 25 Next lexeme is (
Next token is: 11 Next lexeme is sum
Next token is: 21 Next lexeme is +
Next token is: 10 Next lexeme is 47
Next token is: 26 Next lexeme is )
Next token is: 24 Next lexeme is /
Next token is: 11 Next lexeme is total
Next token is: -1 Next lexeme is EOF
```

Το πρόβλημα της συντακτικής ανάλυσης

- Στόχοι ενός συντακτικού αναλυτή, δεδομένης μιας εισόδου προγράμματος:
  - Εύρεση όλων των συντακτικών λαθών, για κάθε λάθος παραγωγή του κατάλληλου διαγνωστικού μηνύματος και γρήγορη ανάκαμψη
  - Παραγωγή του δένδρου συντακτικής ανάλυσης (parse tree), ή τουλάχιστον ενός ίχνους του parse tree, για το πρόγραμμα

Το πρόβλημα της συντακτικής ανάλυσης (συνέχεια)

- Δύο κατηγορίες συντακτικών αναλυτών
  - *Από πάνω προς τα κάτω* – παράγουν το δένδρο συντακτικής ανάλυσης, ξεκινώντας από τη ρίζα
    - Η σειρά είναι αυτής της αριστερότερης παραγωγής
    - Ίχνη ή builds του δένδρου συντακτικής ανάλυσης σε preorder
  - *Από κάτω προς τα πάνω* – παράγουν το δένδρο συντακτικής ανάλυσης, ξεκινώντας από τα φύλλα
    - Η σειρά είναι η αντίστροφη της δεξιότερης παραγωγής
- Οι χρήσιμοι parsers βλέπουν μόνο ένα token μπροστά (ahead) στην είσοδο

Το πρόβλημα της συντακτικής ανάλυσης (συνέχεια)

- Από πάνω προς τα κάτω συντακτικοί αναλυτές
  - Δεδομένης μιας προτασιακής μορφής, xAα , ο συντακτικός αναλυτής θα πρέπει να επιλέξει το σωστό A-κανόνα για να λάβει την επόμενη προτασιακή μορφή στην αριστερότερη παραγωγή, χρησιμοποιώντας μόνο το πρώτο token που παράγεται από το A
- Οι πλέον διαδεδομένοι αλγόριθμοι συντακτικής ανάλυσης από πάνω προς τα κάτω:
  - Αναδρομική κατάβαση – υλοποίηση που κωδικοποιείται
  - LL parsers – υλοποίηση που οδηγείται από πίνακα

Το πρόβλημα της συντακτικής ανάλυσης (συνέχεια)

- Συντακτικοί αναλυτές από κάτω προς τα πάνω
  - Δεδομένης μιας right προτασιακής μορφής, α, καθορίζει ποια υποσυμβολοσειρά του α είναι το δεξί μέρος του κανόνα στη γραμματική που πρέπει να μειωθεί έτσι ώστε να παράξει την προηγούμενη προτασιακή μορφή στη δεξιά παραγωγή
  - Οι πλέον διαδεδομένοι αλγόριθμοι συντακτικής ανάλυσης από κάτω προς τα πάνω ανήκουν στην οικογένεια αλγορίθμων LR

Το πρόβλημα της συντακτικής ανάλυσης (συνέχεια)

- Η πολυπλοκότητα της συντακτικής ανάλυσης
  - Οι συντακτικοί αναλυτές που λειτουργούν για οποιαδήποτε σαφή γραμματική είναι πολύπλοκοι και και complex and μη αποδοτικοί ($O(n^3)$, όπου n είναι το μήκος της εισόδου)
  - Οι μεταγλωττιστές χρησιμοποιούν συντακτικούς αναλυτές που λειτουργούν μόνο για ένα υποσύνολο των σαφών γραμματικών, αλλά διαθέτουν γραμμικούς χρόνους εκτέλεσης (O(n), όπου n είναι το μήκος της εισόδου)

Συντακτική ανάλυση αναδρομικής-κατάβασης

- Υπάρχει ένα υποπρόγραμμα για κάθε μη-τερματικό της γραμματικής, που μπορεί να πραγματοποιεί συντακτική ανάλυση σε προτάσεις που μπορούν να παραχθούν από το συγκεκριμένο μη-τερματικό
- EBNF είναι ιδανική για να χρησιμοποιηθεί ως βάση σε συντακτικούς αναλυτές αναδρομικής κατάβασης, καθώς η EBNF ελαχιστοποιεί το πλήθος των μη-τερματικών

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Μια γραμματική για απλές εκφράσεις:

```
<expr> → <term> {(+ | -) <term>}
<term> → <factor> {(* | /) <factor>}
<factor> → id | int_constant | ( <expr> )
```

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Υποθέτουμε ότι διαθέτουμε έναν λεκτικό αναλυτή με όνομα `lex`, που τοποθετεί το επόμενο κωδικό token στο `nextToken`
- Η διαδικασία κωδικοποίησης όταν υπάρχει ένα μόνο RHS:
  - Για κάθε τερματικό σύμβολο στο RHS, σύγκρινέ το με το επόμενο token εισόδου, αν ταιριάζουν, συνέχισε, αλλιώς υπάρχει σφάλμα
  - Για κάθε μη τερματικό σύμβολο στο RHS, κάλεσε το αντίστοιχο υποπρόγραμμα συντακτικής ανάλυσης

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

```
/* Function expr
   Parses strings in the language
   generated by the rule:
   <expr> → <term> {(+ | -) <term>}
 */

void expr() {

/* Parse the first term */

  term();
/* As long as the next token is + or -, call
   lex to get the next token and parse the
   next term */

  while (nextToken == ADD_OP ||
         nextToken == SUB_OP){
    lex();
    term();
  }
}
```

Συντακτική ανάλυση αναδρομικής–κατάβασης (συνέχεια)

- Η συγκεκριμένη ρουτίνα δεν ανιχνεύει λάθη
- Σύμβαση: Κάθε ρουτίνα συντακτικής ανάλυσης αφήνει το επόμενο token στο **`nextToken`**

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Τα μη-τερματικά που έχουν περισσότερα από ένα RHS απαιτούν μια αρχική διαδικασία που καθορίζει ποια από τα RHS θα είναι αυτό για το οποίο θα γίνει συντακτική ανάλυση
  - Το σωστό RHS επιλέγεται με βάση το επόμενο token της εισόδου (το lookahead)
  - Το επόμενο token συγκρίνεται με το πρώτο token που μπορεί να δημιουργηθεί από κάθε RHS μέχρι να βρεθεί κάποιο που να ταιριάζει
  - Αν δεν βρεθεί ταίριασμα, τότε προκύπτει συντακτικό σφάλμα

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

```
/* term
Parses strings in the language generated by the rule:
<term> -> <factor> {(* | /) <factor>)
*/
void term() {

/* Parse the first factor */
  factor();

/* As long as the next token is * or /,
   next token and parse the next factor */
  while (nextToken == MULT_OP || nextToken == DIV_OP) {
    lex();
    factor();
  }
} /* End of function term */
```

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

```
/* Function factor
   Parses strings in the language
   generated by the rule:
   <factor> -> id  |  (<expr>)  */

 void factor() {

 /* Determine which RHS */
   if (nextToken) == ID_CODE || nextToken == INT_CODE)

 /* For the RHS id, just call lex */
     lex();

/* If the RHS is (<expr>) - call lex to pass over the left parenthesis,
   call expr, and check for the right parenthesis */
   else if (nextToken == LP_CODE) {
    lex();
      expr();
     if (nextToken == RP_CODE)
        lex();
      else
        error();
    }  /* End of else if (nextToken == ...  */

   else error(); /* Neither RHS matches */
}
```

Συντακτική ανάλυση αναδρομικής–κατάβασης (συνέχεια)

- Ιχνηλάτηση του λεκτικού και του συντακτικού αναλυτή στο `(sum + 47) / total`

```
Next token is: 25 Next lexeme is (
Enter <expr>
Enter <term>
Enter <factor>
Next token is: 11 Next lexeme is sum
Enter <expr>
Enter <term>
Enter <factor>
Next token is: 21 Next lexeme is +
Exit <factor>
Exit <term>
Next token is: 10 Next lexeme is 47
Enter <term>
Enter <factor>
Next token is: 26 Next lexeme is )
Exit <factor>
Exit <term>
Exit <expr>
Next token is: 24 Next lexeme is /
Exit <factor>
Next token is: 11 Next lexeme is total
Enter <factor>
Next token is: -1 Next lexeme is EOF
Exit <factor>
Exit <term>
Exit <expr>
```

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Η κατηγορία γραμματικών LL
  - Το πρόβλημα της αριστερής αναδρομής
    - Αν μια γραμματική έχει αριστερή αναδρομή, είτε άμεση είτε έμμεση, δεν μπορεί να αποτελέσει τη βάση ενός συντακτικού αναλυτή από πάνω προς τα κάτω
      - Μια γραμματική μπορεί να τροποποιηθεί έτσι ώστε να αφαιρεθούν οι άμεσες αριστερές αναδρομές ως εξής:

      Για κάθε μη-τερματικό, A,

      1. Ομαδοποίηση των A-κανόνων ως $A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

         όπου κανένα από τα β's δεν ξεκινούν μεA

      2. Αντικατάσταση των A-κανόνων με

         $A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$

         $A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \epsilon$

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Το άλλο χαρακτηριστικό των γραμματικών που δεν επιτρέπει τη συντακτική ανάλυση από πάνω προς τα κάτω είναι η έλλειψη των pairwise disjointness
  - Η αδυναμία να καθοριστεί το σωστό RHS με βάση ένα token του lookahead
  - Def: FIRST(α) = {a | α =>* aβ }

    (If α =>* ε, ε is in FIRST(α))

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Έλεγχος Pairwise Disjointness:
  - Για κάθε μη-τερματικό, A, στη γραμματική που έχει περισσότερα από ένα RHS, για κάθε ζεύγος κανόνων, $A \rightarrow \alpha_i$ και $A \rightarrow \alpha_j$, θα πρέπει να είναι αληθές ότι

$$\text{FIRST}(\alpha_i) \cap \text{FIRST}(\alpha_j) = \phi$$

- Παράδειγμα:

$$A \rightarrow a \mid bB \mid cAb$$

$$A \rightarrow a \mid aB$$

Συντακτική ανάλυση αναδρομικής-κατάβασης (συνέχεια)

- Η αριστερή παραγοντοποίηση μπορεί να επιλύσει το πρόβλημα

  Αντικατάσταση

<variable> → identifier | identifier [<expression>]

  με

<variable> → identifier <new>

<new> → ε | [<expression>]

  ή

<variable> → identifier [[<expression>]]

(Οι εξωτερικές αγκύλες είναι μετασύμβολα του EBNF)

Συντακτική ανάλυση από κάτω προς τα πάνω

- Το πρόβλημα συντακτικής ανάλυσης είναι η εύρεση του σωστού RHS σε μια δεξιά-προτασιακή μορφή που θα μειωθεί έτσι ώστε να ληφθεί η προηγούμενη δεξιά-προτασιακή μορφή σε μια παραγωγή

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Διαίσθηση για τα handles:
  - Ορισμός: το β είναι *handle* της δεξιάς προτασιακής μορφής $\gamma = \alpha\beta w$ αν και μόνο αν $S =>^*_{rm} \alpha A w =>_{rm} \alpha\beta w$
  - Ορισμός: το β είναι μια *phrase* της δεξιάς προτασιακής μορφής γ αν και μόνο αν $S =>^* \gamma = \alpha_1 A \alpha_2 =>+ \alpha_1\beta\alpha_2$
  - Ορισμός: το β είναι μια *simple phrase* της δεξιάς προτασιακής μορφής γ αν και μόνο αν $S =>^* \gamma = \alpha_1 A \alpha_2 => \alpha_1\beta\alpha_2$

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Διαίσθηση για τα handles (συνέχεια):
  - Το handle μιας δεξιάς προτασιακής μορφής είναι η αριστερότερη simple phrase
  - Δεδομένου ενός δένδρου συντακτικής ανάλυσης, είναι τώρα εύκολο να βρεθεί το handle
  - Η συντακτική ανάλυση μπορεί να θεωρηθεί ως κλάδεμα handle

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Αλγόριθμοι Shift-Reduce
  - Reduce είναι η ενέργεια της αντικατάστασης του handle στην κορυφή του parse stack με το αντίστοιχο LHS
  - Shift είναι η ενέργεια της μετακίνησης του επόμενου token στην κορυφή του parse stack

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Πλεονεκτήματα των LR συντακτικών αναλυτών:
  - Λειτουργούν σχεδόν για όλες τις γραμματικές που περιγράφουν γλώσσες προγραμματισμού.
  - Λειτουργούν για μεγαλύτερο εύρος γραμματικών σε σχέση με άλλους αλγορίθμους από κάτω προς τα πάνω, αλλά είναι εξίσου αποδοτικοί με οποιονδήποτε από αυτούς.
  - Μπορούν να ανιχνεύσουν συντακτικά λάθη το συντομότερο δυνατό.
  - Η κατηγορία των LR γραμματικών είναι υπερσύνολο της κατηγορίας που μπορεί να αναλυθεί συντακτικά από LL συντακτικούς αναλυτές.

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Οι LR parsers κατασκευάζονται με λογισμικό
- Η διαίσθηση του Knuth: Ένας συντακτικός αναλυτής από κάτω προς τα πάνω θα μπορούσε να χρησιμοποιήσει το πλήρες ιστορικό της συντακτικής ανάλυσης που έχει πραγματοποιηθεί μέχρι εκείνο το σημείο, έτσι ώστε να λάβει αποφάσεις συντακτικής ανάλυσης
  - Υπάρχει ένα πεπερασμένος και σχετικά μικρός αριθμός από διαφορετικές καταστάσεις συντακτικής ανάλυσης που μπορούν να υπάρξουν, συνεπώς η ιστορία θα μπορούσε να αποθηκευτεί σε μια κατάσταση του συντακτικού αναλυτή, στη στοίβα συντακτικής ανάλυσης

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

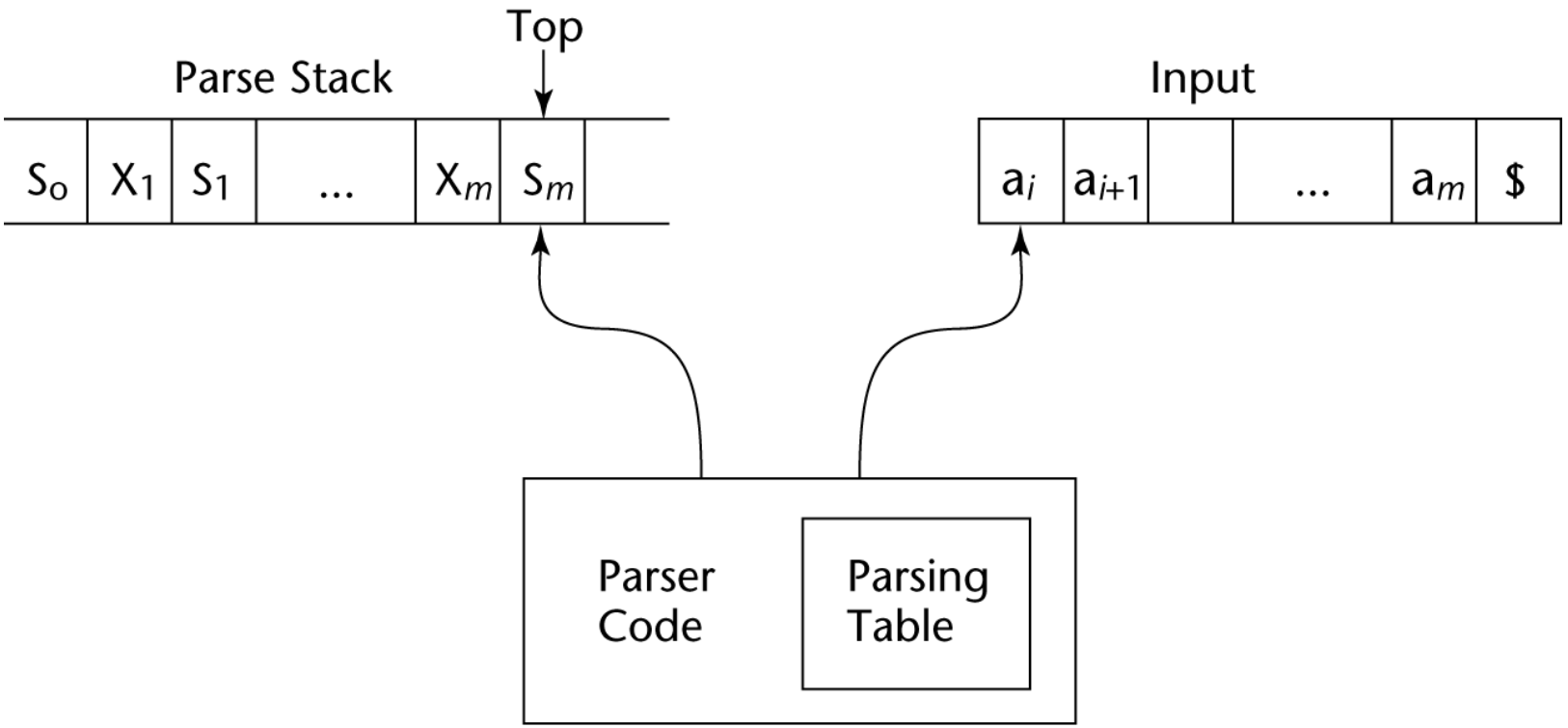
- Ένα LR configuration αποθηκεύει την κατάσταση ενός LR συντακτικού αναλυτή

$(S_0X_1S_1X_2S_2\ldots X_mS_m, a_ia_{i+1}\ldots a_n\$)$

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Οι συντακτικοί αναλυτές LR καθοδηγούνται από πίνακες, με τον πίνακα να έχει δύο επιμέρους τμήματα, τον πίνακα ACTION και τον πίνακα GOTO
  - Ο πίνακας ACTION καθορίζει την ενέργεια που θα επιτελέσει ο συντακτικός αναλυτής, δεδομένης της κατάστασης του συντακτικού αναλυτή και του επόμενου token
    - Οι γραμμές είναι τα ονόματα των καταστάσεων, οι στήλες είναι τα τερματικά
  - Ο πίνακας GOTO καθορίζει ποια κατάσταση θα τοποθετηθεί στην κορυφή της στοίβας συντακτικής ανάλυσης μετά την πραγματοποίηση μια ενέργειας reduction
    - Και στον πίνακα GOTO, οι γραμμές είναι τα ονόματα των καταστάσεων, οι στήλες είναι τα τερματικά

Η δομή ενός LR Συντακτικού Αναλυτή



Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Αρχικό configuration: (S_0, $a_1 \ldots a_n\$$)
- Ενέργειες του συντακτικού αναλυτή:
  - Για μια ολίσθηση (shift), το επόμενο σύμβολο της εισόδου ωθείται στη στοίβα, μαζί με το σύμβολο κατάστασης που αποτελεί μέρος της προδιαγραφής ολίσθησης στον πίνακα Action
  - Για μια μείωση (reduce), αφαιρείται το handle από τη στοίβα, μαζί με τα σύμβολα κατάστασης του. Ωθείται το LHS του κανόνα. Ωθείται στη στοίβα το σύμβολο κατάστασης από τον πίνακα GOTO, χρησιμοποιώντας το σύμβολο κατάστασης ακριβώς κάτω από το νέο LHS, και το LHS του νέου κανόνα ως γραμμή και στήλη στον πίνακα GOTO

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Ενέργειες του συντακτικού αναλυτή (συνέχεια):
  - Σε περίπτωση αποδοχής, η συντακτική ανάλυση θε πρέπει να είναι πλήρης και να μη βρεθούν λάθη.
  - Σε περίπτωση σφάλματος, ο συντακτικός αναλυτής καλεί μια ρουτίνα χειρισμού-λαθών.

LR Πίνακας Συντακτικής Ανάλυσης

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5		S4				1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Συντακτική ανάλυση από κάτω προς τα πάνω (συνέχεια)

- Για μια δεδομένη γραμματική, ο πίνακας συντακτικής ανάλυσης μπορεί να δημιουργηθεί με λογισμικό, π.χ., **`yacc`** ή **`bison`**

Σύνοψη

- Η συντακτική ανάλυση αποτελεί ένα βασικό τμήμα της υλοποίησης της γλώσσας
- Ο λεκτικός αναλυτής είναι ένας μηχανισμός αναγνώρισης προτύπων που απομονώνει αποσπάσματα μικρού μεγέθους του προγράμματος
- Ο συντακτικός αναλυτής
  - Ανιχνεύει συντακτικά λάθη
  - Παράγει το δένδρο συντακτικής ανάλυσης
- Ένας αναδρομικός-καθοδικός συντακτικός αναλυτής είναι ένας LL συντακτικός αναυτής
  - EBNF
- Το πρόβλημα συντακτικής ανάλυσης για τους bottom-up συντακτικούς αναλυτές: εύρεση της υποσυμβολοσειράς στην τρέχουσα προτασιακή μορφή
- Η LR οικογένεια των shift-reduce συντακτικών αναλυτών αποτελούν την πλέον κοινή προσέγγιση συντακτικής ανάλυσης από κάτω προς τα πάνω