Μεταγλωττιστές

Συντακτική Ανάλυση

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Συντακτική Ανάλυση

Το συντακτικό μιας γλώσσας καθορίζει ποιες συμβολοσειρές ανήκουν στην γλώσσα.

Το συντακτικό των γλωσσών προγραμματισμού καθορίζεται χρησιμοποιώντας *γραμματικές χωρίς συμφραζόμενα*.

Παράδειγμα

Ένα if-else στην Java μπορεί να μοιάζει ως

```
if (expression) statement1 else statement2
```

Δηλαδή μια πρόταση if στην Java μπορεί να είναι η παράθεση των

- δεσμευμένη λέξη if
- αριστερή παρένθεση
- μια έκφραση
- δεξιά παρένθεση
- μια πρόταση
- δεσμευμένη λέξη else
- μια πρόταση

Παράδειγμα

Ένα if-else στην Java μπορεί να μοιάζει ως

```
if (expression) statement1 else statement2
```

Χρησιμοποιώντας την μεταβλητή expr για να αναπαραστήσουμε την έκφραση και stmt για την πρόταση μπορούμε να γράψουμε:

$$stmt \rightarrow \textbf{if (} expr \textbf{)} \ stmt \ \textbf{else} \ stmt$$

- Η παραπάνω μορφή ονομάζεται **κανόνας παραγωγής**.
- Λεκτικά στοιχεία όπως το *if* και οι παρενθέσεις ονομάζονται **τερματικά σύμβολα** ενώ οι μεταβλητές *stmt* και *expr* ονομάζονται **μη-τερματικά σύμβολα**.

Ρόλος των Γραμματικών

- Οι γραμματικές δίνουν μια ακριβή και ταυτόχρονα εύκολα κατανοητή περιγραφή του συντακτικού της γλώσσας προγραμματισμού

Ρόλος των Γραμματικών

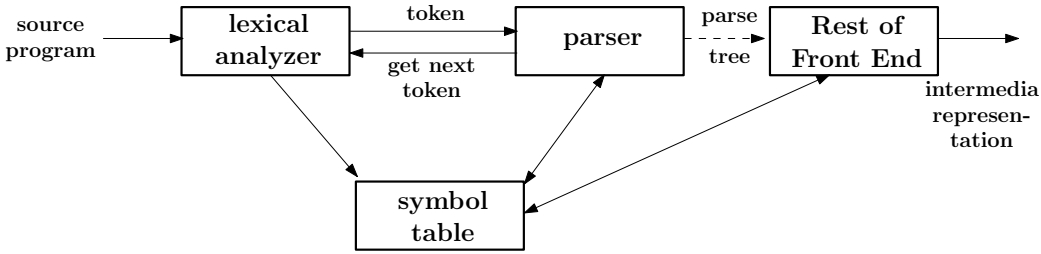
- Οι γραμματικές δίνουν μια ακριβή και ταυτόχρονα εύκολα κατανοητή περιγραφή του συντακτικού της γλώσσας προγραμματισμού
- Για συγκεκριμένου τύπου γραμματικές, μπορούμε να κατασκευάσουμε αυτόματα αποδοτικούς συντακτικούς αναλυτές

Ρόλος των Γραμματικών

- Οι γραμματικές δίνουν μια ακριβή και ταυτόχρονα εύκολα κατανοητή περιγραφή του συντακτικού της γλώσσας προγραμματισμού
- Για συγκεκριμένου τύπου γραμματικές, μπορούμε να κατασκευάσουμε αυτόματα αποδοτικούς συντακτικούς αναλυτές
- Μία γραμματική επιτρέπει σε μια γλώσσα να επεκταθεί τμηματικά ώστε να υποστηρίζει καινούριες έννοιες. Η υλοποίηση των καινούριων εννοιών είναι πιο εύκολο να πραγματοποιηθεί.

Ρόλος και Θέση του Συντακτικού Αναλυτή

Role of Parser



Γραμματικές Χωρίς Συμφραζόμενα
Context-Free Grammars

Μία γραμματική χωρίς συμφραζόμενα έχει 4 μέρη:

1. ένα σύνολο από λεκτικές μονάδες, τα *τερματικά* σύμβολα (terminal symbols)
2. ένα σύνολο από *μη-τερματικά*
3. ένα σύνολο *κανόνων παραγωγής* όπου κάθε κανόνας αποτελείται από ένα μη-τερματικό που ονομάζεται η αριστερή πλευρά του κανόνα, ένα βέλος και μια ακολουθία από λεκτικές μονάδες και/ή μη-τερματικά που ονομάζεται η δεξιά πλευρά του κανόνα
4. τον χαρακτηρισμό ενός μη-τερματικού ως το *αρχικό* σύμβολο

Μελετήθηκαν στην αρχή από γλωσσολόγους (Chomsky, 1956).

Γραμματικές

Παράδειγμα

$$
\begin{array}{rcl}
\textit{list} & \rightarrow & \textit{list} + \textit{digit} \\
\textit{list} & \rightarrow & \textit{list} - \textit{digit} \\
\textit{list} & \rightarrow & \textit{digit} \\
\textit{digit} & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

- Γράφουμε με πλάγια γράμματα τα μη-τερματικά.
- Ο πρώτος κανόνας μας λέει το αρχικό σύμβολο.

Οι πρώτοι 3 κανόνες θα μπορούσαν να είναι γραμμένοι και ως

$$
\textit{list} \rightarrow \textit{list} + \textit{digit} \mid \textit{list} - \textit{digit} \mid \textit{digit}
$$

Η γλώσσα που ορίζεται από την γραμματική αυτή είναι το σύνολο των συμβολοσειρών που περιέχουν μια λίστα ψηφίων που χωρίζονται από το + ή το -.

Μια γραμματική παράγει συμβολοσειρές ξεκινώντας από το αρχικό σύμβολο και αντικαθιστώντας ένα μη-τερματικό σύμβολο με την δεξιά πλευρά ενός κανόνα για αυτό το σύμβολο.

π.χ

μέρος μιας γραμματικής για την κλήση συναρτήσεων στην Java

$$
\begin{array}{rcl}
\textit{call} & \rightarrow & \text{id (} \textit{optparams} \text{)} \\
\textit{optparams} & \rightarrow & \textit{params} \mid \epsilon \\
\textit{params} & \rightarrow & \textit{params} \text{ , } \textit{param} \mid \textit{param}
\end{array}
$$

Τρόπος Αναπαράστασης Γραμματικών

BNF (Backus Naur Form)

- μη-τερματικά σύμβολα με γωνιακές παρενθέσεις, π.χ <expr>
- το σύμβολο $\rightarrow$ αντικαθίσταται από το σύμβολο ::=
- συνδυασμός πολλών δεξιών μελών σε έναν κανόνα παραγωγής με το σύμβολο |

  π.χ <digit> ::= 0 | 1

EBNF (Extended Backus Naur Form)

- επιπλέον σύμβολα όπως $+$ και $*$ με την σημασιολογία που έχουν και στην περίπτωση των κανονικών εκφράσεων

Συντακτική Ανάλυση
Parsing

Είναι η διαδικασία που έχει ως είσοδο ένα σύνολο από τερματικά σύμβολα και πρέπει να αποφασίσει πως αυτή η είσοδος προέκυψε από το αρχικό σύμβολο μιας γραμματικής.

- Εαν δεν είναι δυνατό, τότε πρέπει να αναφέρει συντακτικά λάθη.

Συντακτική Ανάλυση
Parsing

Υπάρχουν 3 γενικές κατηγορίες συντακτικής ανάλυσης:

1. *καθολικές* (universal) μέθοδοι
2. καθοδικές (top-down) μέθοδοι
3. ανοδικές (bottom-up) μέθοδοι

Συντακτική Ανάλυση
Parsing

Υπάρχουν 3 γενικές κατηγορίες συντακτικής ανάλυσης:

1. *καθολικές* (universal) μέθοδοι
2. καθοδικές (top-down) μέθοδοι
3. ανοδικές (bottom-up) μέθοδοι

- Οι καθολικές δουλεύουν με όλες τις γραμματικές αλλά είναι αργές για χρήση στον προγραμματισμό.

Υπάρχουν 3 γενικές κατηγορίες συντακτικής ανάλυσης:

1. *καθολικές* (universal) μέθοδοι
2. καθοδικές (top-down) μέθοδοι
3. ανοδικές (bottom-up) μέθοδοι

- Οι καθολικές δουλεύουν με όλες τις γραμματικές αλλά είναι αργές για χρήση στον προγραμματισμό.
- Οι καθοδικές και ανοδικές μέθοδοι είναι οι μέθοδοι που χρησιμοποιούνται πιο συχνά στους μεταγλωττιστές.

Υπάρχουν 3 γενικές κατηγορίες συντακτικής ανάλυσης:

1. *καθολικές* (universal) μέθοδοι
2. καθοδικές (top-down) μέθοδοι
3. ανοδικές (bottom-up) μέθοδοι

- Οι καθολικές δουλεύουν με όλες τις γραμματικές αλλά είναι αργές για χρήση στον προγραμματισμό.
- Οι καθοδικές και ανοδικές μέθοδοι είναι οι μέθοδοι που χρησιμοποιούνται πιο συχνά στους μεταγλωττιστές.
- Οι πιο αποδοτικές μέθοδοι λειτουργούν μόνο για υποσύνολα των γραμματικών, αλλά αυτά τα υποσύνολα είναι συνήθως αρκετά για να εκφράσουμε γλώσσες προγραμματισμού.

Η έξοδος ενός συντακτικού αναλυτή είναι μια αναπαράσταση ενός **συντακτικού δέντρου**. Το δέντρο αυτό περιέχει τις λεκτικές μονάδες που έχουν προκύψει από τον λεκτικό αναλυτή.

Συνήθως κατά την διάρκεια της συντακτικής ανάλυσης συμβαίνουν και άλλες διαδικασίες όπως η συλλογή πληροφοριών στον πίνακα συμβόλων σχετικά με τα σύμβολα που προέκυψαν από τον λεκτικό αναλυτή, κ.τ.λ.

Διαχείριση Συντακτικών Λαθών

Syntax Error Handling

Η δουλειά ενός μεταγλωττιστή θα ήταν πολύ ευκολότερη αν είχε να διαχειριστεί μόνο σωστά προγράμματα.

Είναι όμως απαραίτητη λειτουργία ενός μεταγλωττιστή να

- βοηθάει στην εύρεση του λάθους (γραμμή, θέση)
- βοηθάει στην εύρεση του τύπου του λάθους
- παρέχει συμβουλές διόρθωσης λαθών.

Τα προγραμματιστικά λάθη μπορούν να συμβούν σε πολλά επίπεδα:

1. *Λεκτικά* λάθη όπως ορθογραφικά λάθη δεσμευμένων λέξεων έως παράληψη εισαγωγικών σε συμβολοσειρές.

Τα προγραμματιστικά λάθη μπορούν να συμβούν σε πολλά επίπεδα:

1. *Λεκτικά* λάθη όπως ορθογραφικά λάθη δεσμευμένων λέξεων έως παράληψη εισαγωγικών σε συμβολοσειρές.
2. *Συντακτικά* λάθη όπως παράληψη ερωτηματικών, ανοιχτές παρενθέσεις, κ.τ.λ

Τα προγραμματιστικά λάθη μπορούν να συμβούν σε πολλά επίπεδα:

1. *Λεκτικά* λάθη όπως ορθογραφικά λάθη δεσμευμένων λέξεων έως παράληψη εισαγωγικών σε συμβολοσειρές.
2. *Συντακτικά* λάθη όπως παράληψη ερωτηματικών, ανοιχτές παρενθέσεις, κ.τ.λ
3. *Σημασιολογικά* λάθη όπως προβλήματα τύπων, για παράδειγμα επιστροφή τιμής από συνάρτηση Java η οποία είναι δηλωμένη ως void.

Τα προγραμματιστικά λάθη μπορούν να συμβούν σε πολλά επίπεδα:

1. *Λεκτικά* λάθη όπως ορθογραφικά λάθη δεσμευμένων λέξεων έως παράληψη εισαγωγικών σε συμβολοσειρές.
2. *Συντακτικά* λάθη όπως παράληψη ερωτηματικών, ανοιχτές παρενθέσεις, κ.τ.λ
3. *Σημασιολογικά* λάθη όπως προβλήματα τύπων, για παράδειγμα επιστροφή τιμής από συνάρτηση Java η οποία είναι δηλωμένη ως void.
4. *Λογικά* λάθη που μπορεί να είναι από απλά λάθη λογικής από την μεριά του προγραμματιστή έως την χρήση του τελεστή = αντί για == σε ένα πρόγραμμα *C*.

Ανάνηψη
Error Recovery

Η διαδικασία διαχείρισης λαθών έχει τους εξής υψηλού επιπέδου στόχους:

- ξεκάθαρη και σαφή αναφορά των λαθών
- γρήγορη ανάνηψη από κάθε λάθος ώστε να είναι δυνατή η αναγνώριση και επόμενων λαθών
- μικρή καθυστέρηση σε περίπτωση σωστών προγραμμάτων

Ευτυχώς τα πιο κοινά λάθη είναι απλά και ένας σχετικά απλός μηχανισμός διαχείρισης λαθών είναι συχνά αρκετός.

1. **Panic-Mode Recovery**
   - αφαιρούμε σύμβολα από την είσοδο μέχρι να βρούμε ένα σύμβολο από ένα σύνολο *συγχρονισμού*.
   - το σύνολο συγχρονισμού περιέχει συνήθως σύμβολα όπως ; ή }
   - ο σχεδιαστής του μεταγλωττιστή είναι υπεύθυνος για τον ορισμό των συμβόλων συγχρονισμού
   - μπορεί να αφαιρέσει πολλά σύμβολα χωρίς να βρει επιπλέον λάθη
   - απλή μέθοδος

Στρατηγικές Ανάνηψης

1. **Panic-Mode Recovery**
   - αφαιρούμε σύμβολα από την είσοδο μέχρι να βρούμε ένα σύμβολο από ένα σύνολο *συγχρονισμού*.
   - το σύνολο συγχρονισμού περιέχει συνήθως σύμβολα όπως ; ή }
   - ο σχεδιαστής του μεταγλωττιστή είναι υπεύθυνος για τον ορισμό των συμβόλων συγχρονισμού
   - μπορεί να αφαιρέσει πολλά σύμβολα χωρίς να βρει επιπλέον λάθη
   - απλή μέθοδος
2. **Phrase-Level Recovery**
   - τοπική αλλαγή ενός προθέματος της εισόδου
   - π.χ αλλαγή ενός κόμματος με ερωτηματικό ή προσθήκη ερωτηματικού
   - ο σχεδιαστής του μεταγλωττιστή ορίζει τις αλλαγές

Στρατηγικές Ανάνηψης

1. **Panic-Mode Recovery**
   - αφαιρούμε σύμβολα από την είσοδο μέχρι να βρούμε ένα σύμβολο από ένα σύνολο *συγχρονισμού*.
   - το σύνολο συγχρονισμού περιέχει συνήθως σύμβολα όπως ; ή }
   - ο σχεδιαστής του μεταγλωττιστή είναι υπεύθυνος για τον ορισμό των συμβόλων συγχρονισμού
   - μπορεί να αφαιρέσει πολλά σύμβολα χωρίς να βρει επιπλέον λάθη
   - απλή μέθοδος
2. **Phrase-Level Recovery**
   - τοπική αλλαγή ενός προθέματος της εισόδου
   - π.χ αλλαγή ενός κόμματος με ερωτηματικό ή προσθήκη ερωτηματικού
   - ο σχεδιαστής του μεταγλωττιστή ορίζει τις αλλαγές
3. **Error Productions**
   - προσθέτουμε στην γραμματική κανόνες για κοινά λάθη

Στρατηγικές Ανάνηψης

1. **Panic-Mode Recovery**
   - αφαιρούμε σύμβολα από την είσοδο μέχρι να βρούμε ένα σύμβολο από ένα σύνολο *συγχρονισμού*.
   - το σύνολο συγχρονισμού περιέχει συνήθως σύμβολα όπως ; ή }
   - ο σχεδιαστής του μεταγλωττιστή είναι υπεύθυνος για τον ορισμό των συμβόλων συγχρονισμού
   - μπορεί να αφαιρέσει πολλά σύμβολα χωρίς να βρει επιπλέον λάθη
   - απλή μέθοδος
2. **Phrase-Level Recovery**
   - τοπική αλλαγή ενός προθέματος της εισόδου
   - π.χ αλλαγή ενός κόμματος με ερωτηματικό ή προσθήκη ερωτηματικού
   - ο σχεδιαστής του μεταγλωττιστή ορίζει τις αλλαγές
3. **Error Productions**
   - προσθέτουμε στην γραμματική κανόνες για κοινά λάθη
4. **Global Correction**
   - αλγόριθμοι που βρίσκουν τις ελάχιστες αλλαγές που χρειάζονται για να γίνει σωστή η είσοδος
   - γενικά πολύ χρονοβόρες για τους μεταγλωττιστές

Συντακτικά Δέντρα

Parse Trees

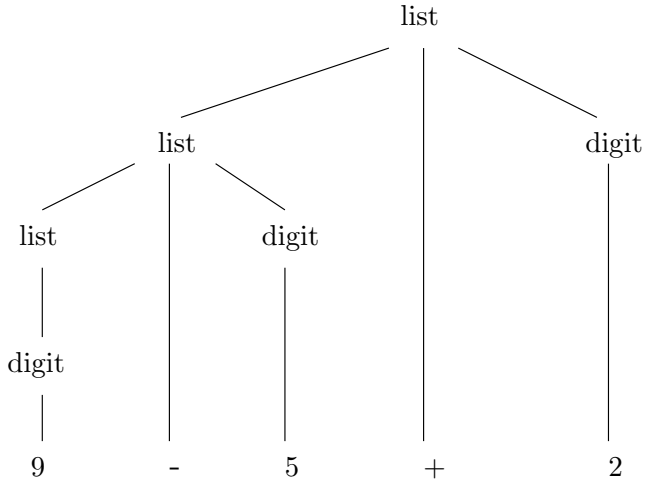
Ένα συντακτικό δέντρο δείχνει γραφικά πως ένα αρχικό σύμβολο μιας γραμματικής οδηγεί σε μια συμβολοσειρά της γλώσσας.

Δεδομένου μιας γραμματικής χωρίς συμφραζόμενα, ένα **συντακτικό δέντρο** είναι ένα δέντρο με τις εξής ιδιότητες:

1. Η ρίζα έχει ως ετικέτα το αρχικό σύμβολο
2. Κάθε φύλλο έχει ως ετικέτα μια λεκτική μονάδα ή ϵ
3. Κάθε εσωτερικός κόμβος έχει ως ετικέτα ένα μη-τερματικό
4. Αν A είναι το μη-τερματικό ενός εσωτερικού κόμβου και $X_1, X_2, \ldots, X_n$ είναι οι ετικέτες των παιδιών του από αριστερά προς τα δεξιά τότε υπάρχει ο κανόνας παραγωγής $A \rightarrow X_1 X_2 \cdots X_n$. Τα $X_1, X_2, \ldots, X_n$ μπορεί να είναι τερματικά ή μη-τερματικά σύμβολα. Ως ειδική περίπτωση, αν $A \rightarrow \epsilon$ τότε ένας κόμβος με ετικέτα A μπορεί να έχει ένα μόνο παιδί με ετικέτα ϵ.

Συντακτικά Δέντρα

Παράδειγμα



Τα φύλλα του δέντρου από αριστερά προς τα δεξιά παράγουν την συμβολοσειρά που αντιστοιχεί στο συντακτικό δέντρο.

Διφορούμενες Γραμματικές

Ambiguity

Προσοχή χρειάζεται αφού υπάρχουν γραμματικές που παράγουν συμβολοσειρές που έχουν περισσότερα από ένα συντακτικά δέντρα.

Αυτές οι γραμματικές ονομάζονται **διφορούμενες**.

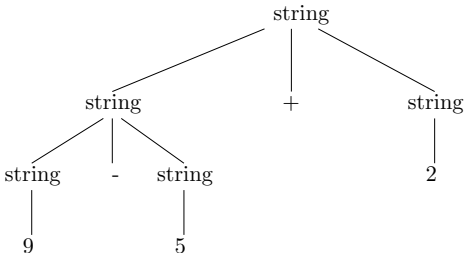
έστω η παρακάτω γραμματική

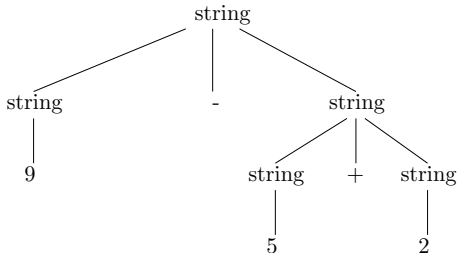
$$
\begin{array}{lcl}
\textit{string} & \rightarrow & \textit{string} + \textit{string} \\
\textit{string} & \rightarrow & \textit{string} - \textit{string} \\
\textit{string} & \rightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

η συμβολοσειρά 9-5+2 έχει περισσότερα από ένα συντακτικά δέντρα τα οποία αντιστοιχούν στους δύο τρόπους που μπορούμε να βάλουμε παρενθέσεις (9-5)+2 και 9-(5+2).

Διφορούμενες Γραμματικές

Ambiguity





Προσεταιριστικότητα Τελεστών

Associativity of Operators

Λόγω κοινής παραδοχής ξέρουμε πως η έκφραση $9 + 5 + 2$ είναι ισοδύναμη με $(9 + 5) + 2$ και η $9 - 5 - 2$ είναι ισοδύναμη με $(9 - 5) - 2$.

Λέμε πως ο τελεστής + έχει αριστερή προσεταιριστικότητα εαν στην έκφραση

$$\ldots + x + \ldots$$

το x ανήκει στον αριστερό τελεστή.

Προσεταιριστικότητα Τελεστών

Associativity of Operators

- Στις περισσότερες γλώσσες προγραμματισμού οι τελεστές +, −, *, / είναι αριστερά προσεταιριστικοί.

  Γράφοντας δηλαδή $9 + 5 + 2$ εννοούμε $(9 + 5) + 2$.

Προσεταιριστικότητα Τελεστών

Associativity of Operators

- Στις περισσότερες γλώσσες προγραμματισμού οι τελεστές +, −, *, / είναι αριστερά προσεταιριστικοί.

  Γράφοντας δηλαδή $9 + 5 + 2$ εννοούμε $(9 + 5) + 2$.

- Αντιθέτως στις περισσότερες γλώσσες προγραμματισμού ο τελεστής ανάθεσης (assignment) είναι δεξιά προσεταιριστικός.

  Γράφοντας δηλαδή $a = b = c$ εννοούμε $a = (b = c)$.

Προσεταιριστικότητα Τελεστών

Associativity of Operators

Μια γραμματική που παράγει συμβολοσειρές όπως `a = b = c = d` με ανάθεση που είναι δεξιά προσεταιριστική είναι η παρακάτω:

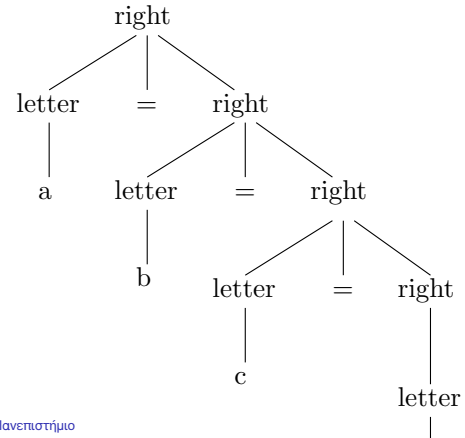
$$\begin{array}{lcl} \textit{right} & \rightarrow & \textit{letter} = \textit{right} \mid \textit{letter} \\ \textit{letter} & \rightarrow & \mathrm{a} \mid \mathrm{b} \mid \cdots \mid \mathrm{z} \end{array}$$

Προσεταιριστικότητα Τελεστών

Associativity of Operators

Μια γραμματική που παράγει συμβολοσειρές όπως `a = b = c = d` με ανάθεση που είναι δεξιά προσεταιριστική είναι η παρακάτω:

$$
\begin{array}{lcl}
\textit{right} & \rightarrow & \textit{letter} = \textit{right} \mid \textit{letter} \\
\textit{letter} & \rightarrow & \mathrm{a} \mid \mathrm{b} \mid \cdots \mid \mathrm{z}
\end{array}
$$



Υπάρχει μόνο ένας τρόπος να περιγραφεί αυτή η είσοδος με την γραμματική αυτή.

Προτεραιότητα Τελεστών

Precedence of Operators

Η προσεταιριστικότητα των τελεστών λύνει το πρόβλημα όταν υπάρχει ο ίδιος τελεστής περισσότερες από μια φορές. Τι συμβαίνει όμως όταν έχουμε διαφορετικούς τελεστές;

π.χ η έκφραση $9 + 5 * 2$ θα μπορούσε να είναι $(9 + 5) * 2$ ή $9 + (5 * 2)$.

Όταν έχουμε περισσότερους από έναν τελεστές χρειαζόμαστε *προτεραιότητα* τελεστών.

Προτεραιότητα Τελεστών

Precedence of Operators

Λέμε πως ο τελεστής $*$ έχει μεγαλύτερη προτεραιότητα από τον τελεστή $+$ εαν πέρνει τα ορίσματα του πριν από τον τελεστή $+$.

Στην έκφραση λοιπόν $9 + 5 * 2$ ο τελεστής $*$ πέρνει πρώτος ορίσματα και άρα είναι ισοδύναμη με $9 + (5 * 2)$.

Παράδειγμα Γραμματικής

Έστω λοιπόν πως θέλουμε να γράψουμε μια γραμματική για αριθμητικές εκφράσεις με τους τελεστές $+, -, *, /$ οι οποίοι είναι όλοι αριστερά προσεταιριστικοί.

- Θα δημιουργήσουμε 2 μη-τερματικά σύμβολα για τα δύο επίπεδα προτεραιότητας, *expr* και *term*.
- Επίσης θα δημιουργήσουμε ένα μη-τερματικό σύμβολο *factor* για την βασική μονάδα των εκφράσεων που είναι τα ψηφία και οι εκφράσεις με παρενθέσεις.

$$\begin{aligned} \textit{factor} &\rightarrow \textit{digit} \mid (\ \textit{expr}\ ) \\ \textit{digit} &\rightarrow 0 \mid 1 \mid 2 \mid \cdots \mid 9 \end{aligned}$$

Παράδειγμα Γραμματικής

Στην συνέχεια προσθέτουμε κανόνες για την πρόσθεση και την αφαίρεση, φροντίζοντας να είναι αριστερά προσεταιριστικοί

$$
\begin{array}{lcl}
\textit{expr} & \rightarrow & \textit{expr} + \textit{term} \\
\textit{expr} & \rightarrow & \textit{expr} - \textit{term} \\
\textit{expr} & \rightarrow & \textit{term}
\end{array}
$$

Παράδειγμα Γραμματικής

Στην συνέχεια προσθέτουμε κανόνες για την πρόσθεση και την αφαίρεση, φροντίζοντας να είναι αριστερά προσεταιριστικοί

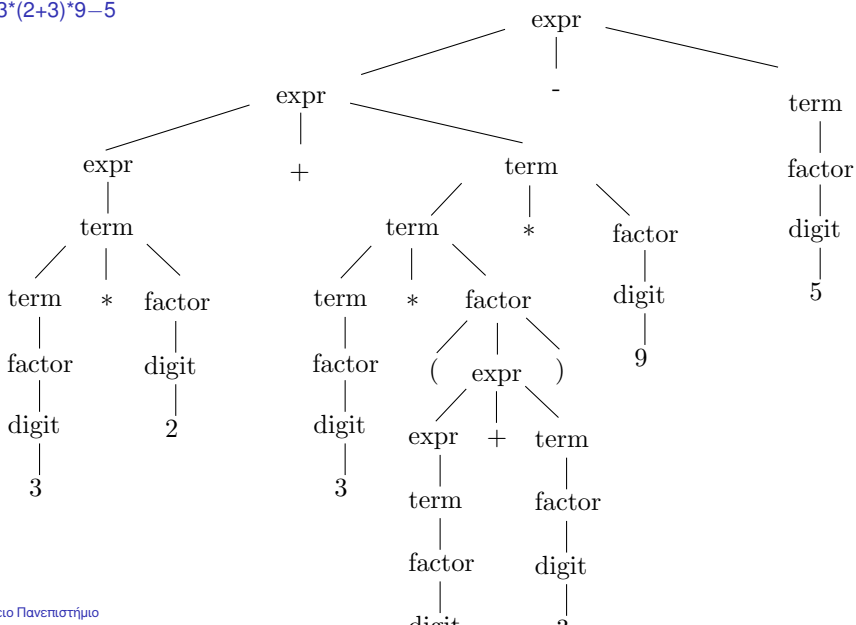
$$
\begin{array}{lcl}
\textit{expr} & \rightarrow & \textit{expr} + \textit{term} \\
\textit{expr} & \rightarrow & \textit{expr} - \textit{term} \\
\textit{expr} & \rightarrow & \textit{term}
\end{array}
$$

και αντίστοιχα προσθέτουμε κανόνες για τον πολλαπλασιασμό και την διαίρεση, φροντίζοντας να είναι αριστερά προσεταιριστικοί

$$
\begin{array}{lcl}
\textit{term} & \rightarrow & \textit{term} * \textit{factor} \\
\textit{term} & \rightarrow & \textit{term} / \textit{factor} \\
\textit{term} & \rightarrow & \textit{factor}
\end{array}
$$

Παράδειγμα Γραμματικής

3*2+3*(2+3)*9−5



Παραγωγές
Derivations

έστω η γραμματική

$$E \quad \rightarrow \quad E + E \mid E * E \mid - E \mid (E) \mid \mathbf{id}$$

Ξεκινώντας από το αρχικό σύμβολο και χρησιμοποιώντας τους κανόνες παραγωγής μπορούμε να παράγουμε συμβολοσειρές που ανήκουν στην γλώσσα που αναπαριστά η γραμματικη.

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(\mathbf{id})$$

Η παραπάνω διαδικασία ονομάζεται παραγωγή του $-(\mathbf{id})$ από το E.

Παραγωγές
Derivations

Το σύμβολο $\Rightarrow$ σημαίνει ”παράγει σε ένα βήμα”.

Όταν θέλουμε να πούμε ”παράγει σε μηδέν ή περισσότερα βήματα” χρησιμοποιούμε το σύμβολο $\overset{*}{\Rightarrow}$.

Αντίστοιχα το σύμβολο $\overset{+}{\Rightarrow}$ για να πούμε ”παράγει σε ένα ή περισσότερα βήματα”.

Πρόταση

Μια συμβολοσειρά χωρίς μη-τερματικά σύμβολα που παράγεται από το αρχικό σύμβολο μιας γραμματικής χρησιμοποιώντας κανόνες παραγωγής ονομάζεται *πρόταση* της γραμματικής.

Ορισμοί

Πρόταση

Μια συμβολοσειρά χωρίς μη-τερματικά σύμβολα που παράγεται από το αρχικό σύμβολο μιας γραμματικής χρησιμοποιώντας κανόνες παραγωγής ονομάζεται *πρόταση* της γραμματικής.

Γλώσσα

Η *γλώσσα* που παράγεται από μια γραμματική είναι το σύνολο των προτάσεων που μπορούν να παραχθούν από την γραμματική.

Ορισμοί

Πρόταση

Μια συμβολοσειρά χωρίς μη-τερματικά σύμβολα που παράγεται από το αρχικό σύμβολο μιας γραμματικής χρησιμοποιώντας κανόνες παραγωγής ονομάζεται *πρόταση* της γραμματικής.

Γλώσσα

Η *γλώσσα* που παράγεται από μια γραμματική είναι το σύνολο των προτάσεων που μπορούν να παραχθούν από την γραμματική.

Γλώσσα Χωρίς Συμφραζόμενα

Μια γλώσσα που παράγεται από μια γραμματική χωρίς συμφραζόμενα ονομάζεται *γλώσσα χωρίς συμφραζόμενα*.

Ορισμοί

Ισοδύναμες Γραμματικές

Δύο γραμματικές που παράγουν την ίδια γλώσσα, ονομάζονται *ισοδύναμες*.

Επιλογές Παραγωγών

Κατά την διάρκεια μιας παραγωγής έχουμε δύο επιλογές

1. επιλογή του μη-τερματικού προς αντικατάσταση
2. επιλογή του κανόνα παραγωγής που έχει αυτό το τερματικό ως αριστερό μέρος

π.χ

η παραγωγή

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(E+\mathbf{id}) \Rightarrow -(\mathbf{id}+\mathbf{id})$$

διαφέρει από την παραγωγή

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\mathbf{id}+E) \Rightarrow -(\mathbf{id}+\mathbf{id})$$

αριστερότερη παραγωγή

Μια παραγωγή ονομάζεται *αριστερότερη παραγωγή* (leftmost derivation) εαν επιλέγεται προς αντικατάσταση πάντα το πρώτο μη-τερματικό από αριστερά. Συμβολίζουμε τέτοιες παραγωγές με $\alpha \underset{lm}{\Rightarrow} \beta$.

Παραγωγές

αριστερότερη παραγωγή

Μια παραγωγή ονομάζεται *αριστερότερη παραγωγή* (leftmost derivation) εαν επιλέγεται προς αντικατάσταση πάντα το πρώτο μη-τερματικό από αριστερά. Συμβολίζουμε τέτοιες παραγωγές με $\alpha \underset{lm}{\Rightarrow} \beta$.

δεξιότερη παραγωγή

Μια παραγωγή ονομάζεται *δεξιότερη παραγωγή* (rightmost derivation) εαν επιλέγεται προς αντικατάσταση πάντα το πρώτο μη-τερματικό από δεξιά. Συμβολίζουμε τέτοιες παραγωγές με $\alpha \underset{rm}{\Rightarrow} \beta$.

Συντακτικά Δέντρα και Παραγωγές

- Τα συντακτικά δέντρα είναι μια γραφική αναπαράσταση παραγωγών που δεν αναπαριστά την σειρά που εκτελέστηκαν οι παραγωγές.

Συντακτικά Δέντρα και Παραγωγές

- Τα συντακτικά δέντρα είναι μια γραφική αναπαράσταση παραγωγών που δεν αναπαριστά την σειρά που εκτελέστηκαν οι παραγωγές.

- Μας ενδιαφέρουν μόνο αριστερότερες ή δεξιότερες παραγωγές μιας και αυτές έχουν 1-1 σχέση με συντακτικά δέντρα.

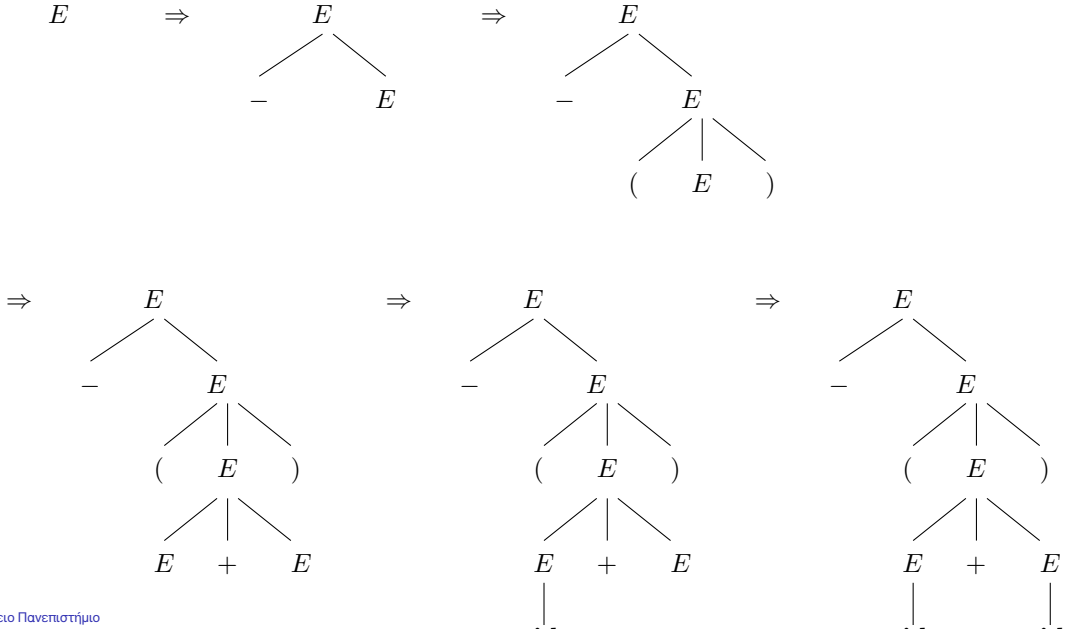
Συντακτικά Δέντρα και Παραγωγές

- Τα συντακτικά δέντρα είναι μια γραφική αναπαράσταση παραγωγών που δεν αναπαριστά την σειρά που εκτελέστηκαν οι παραγωγές.

- Μας ενδιαφέρουν μόνο αριστερότερες ή δεξιότερες παραγωγές μιας και αυτές έχουν 1-1 σχέση με συντακτικά δέντρα.

- Μπορούμε λοιπόν να ορίσουμε την έννοια της διφορούμενης γραμματικής και σε σχέση με την ύπαρξη δύο η περισσότερων αριστερότερων (ή δεξιότερων) παραγωγών για μία πρόταση.

Συντακτικά Δέντρα και Παραγωγές



Γραμματικές Χωρίς Συμφραζόμενα και Κανονικές Εκφράσεις

Οποιαδήποτε γλώσσα μπορεί να περιγραφεί από μια κανονική έκφραση μπορεί να περιγραφεί και από μια γραμματική χωρίς συμφραζόμενα. Το αντίθετο όμως δεν ισχύει.

π.χ
η κανονική έκφραση $(a|b) * abb$ και η γραμματική

$$
\begin{array}{lcl}
A_0 & \rightarrow & aA_0 \mid bA_0 \mid aA_1 \\
A_1 & \rightarrow & bA_2 \\
A_2 & \rightarrow & bA_3 \\
A_3 & \rightarrow & \epsilon
\end{array}
$$

περιγράφουν την ίδια γλώσσα.

Γραμματικές Χωρίς Συμφραζόμενα και Κανονικές Εκφράσεις

Οποιαδήποτε γλώσσα μπορεί να περιγραφεί από μια κανονική έκφραση μπορεί να περιγραφεί και από μια γραμματική χωρίς συμφραζόμενα. Το αντίθετο όμως δεν ισχύει.

Η γλώσσα $\{a^n b^n | n \geq 1\}$ δεν μπορεί να περιγραφεί με μια κανονική έκφραση ενώ μπορεί από μια γλώσσα χωρίς συμφραζόμενα.

Με μια φράση μπορούμε να πούμε πως

"τα πεπερασμένα αυτόματα δεν μπορούν να μετρήσουν"

Τέλος Αρχείου

Πριν ξεκινήσουμε να μιλάμε για μεθόδους συντακτικής ανάλυσης, πρέπει να κάνουμε έναν απαραίτητο μετασχηματισμό στην γραμματική μας ώστε να μπορεί ο συντακτικός αναλυτής να καταλάβει και το τέλος του αρχείου εισόδου.

Θα χρησιμοποιούμε το σύμβολο $\$$ για το **E**nd-**O**f-**F**ile.

Εαν στην αρχική γραμματική μας υπήρχε ένα αρχικό σύμβολο S, αλλάζουμε το αρχικό σύμβολο σε S' και προσθέτουμε έναν κανόνα

$$S' \to S\,\$$$

Καθοδική Κατασκευή Συντακτικού Δέντρου

top-down

Το πρόβλημα της κατασκευής ενός συντακτικού δέντρου από την συμβολοσειρά εισόδου.

- ξεκινάμε από την ρίζα
- επιλέγουμε συνεχόμενα το αριστερότερο μη-τερματικό σύμβολο προς αντικατάσταση (κατασκευάζουμε τους κόμβους με preorder σειρά, left-most derivation).
- διαλέγουμε τον κανόνα παραγωγής που θα χρησιμοποιήσουμε για την αντικατάσταση του μη-τερματικού.

Καθοδική Κατασκευή Συντακτικού Δέντρου

top-down

Το πρόβλημα της κατασκευής ενός συντακτικού δέντρου από την συμβολοσειρά εισόδου.

- ξεκινάμε από την ρίζα
- επιλέγουμε συνεχόμενα το αριστερότερο μη-τερματικό σύμβολο προς αντικατάσταση (κατασκευάζουμε τους κόμβους με preorder σειρά, left-most derivation).
- διαλέγουμε τον κανόνα παραγωγής που θα χρησιμοποιήσουμε για την αντικατάσταση του μη-τερματικού.

Επαναλαμβάνουμε την διαδικασία αυτή μέχρι να τελειώσουν τα μη-τερματικά ή να μην μπορούμε να αναγνωρίσουμε την είσοδο.

Καθοδική Κατασκευή Συντακτικού Δέντρου

top-down

Το πρόβλημα της κατασκευής ενός συντακτικού δέντρου από την συμβολοσειρά εισόδου.

- ξεκινάμε από την ρίζα
- επιλέγουμε συνεχόμενα το αριστερότερο μη-τερματικό σύμβολο προς αντικατάσταση (κατασκευάζουμε τους κόμβους με preorder σειρά, left-most derivation).
- διαλέγουμε τον κανόνα παραγωγής που θα χρησιμοποιήσουμε για την αντικατάσταση του μη-τερματικού.

Επαναλαμβάνουμε την διαδικασία αυτή μέχρι να τελειώσουν τα μη-τερματικά ή να μην μπορούμε να αναγνωρίσουμε την είσοδο.

Στην γενική μορφή αν οι κανόνες που έχουμε διαλέξει δεν οδηγούν σε ”αναγνώριση” όλης της εισόδου τότε κάνουμε **backtracking**.

Recursive-descent Parsing

Ένα πρόγραμμα με

- μια συνάρτηση για κάθε μη-τερματικό
- η εκτέλεση ξεκινά με την συνάρτηση για το αρχικό σύμβολο

```
void A() {
    Choose an A–production, A → X1 X2 ··· Xk;
    for( i = 1 to k ) {
        if ( Xi is a nonterminal )
            call procedure Xi();
        else if ( Xi equals the current input symbol a ) {
            advance input to next symbol;
        }
        else {
            backtrack the pointer which reads the input;
            choose another production and continue;
            if no more production rules report error;
        }
    }
}
```

Recursive-descent Parsing

Παράδειγμα

Έστω η γραμματική

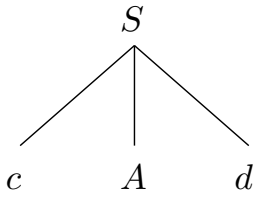
$$\begin{array}{lcl} S & \Rightarrow & c\,A\,d \\ A & \Rightarrow & a\,b \\ A & \Rightarrow & a \end{array}$$

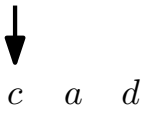
και η είσοδος

c a d

Recursive-descent Parsing

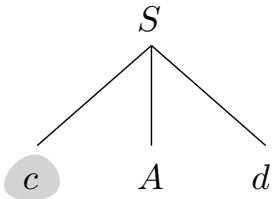
Παράδειγμα

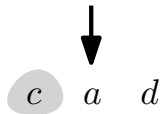




Recursive-descent Parsing

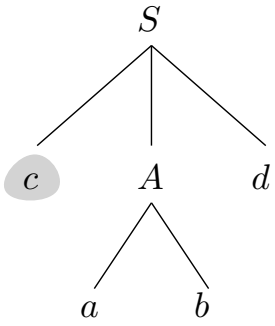
Παράδειγμα

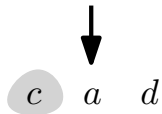




Recursive-descent Parsing

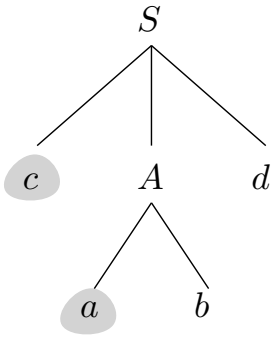
Παράδειγμα

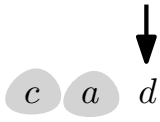




Recursive-descent Parsing

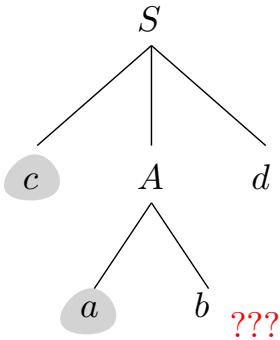
Παράδειγμα

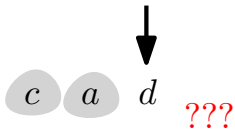




Recursive-descent Parsing

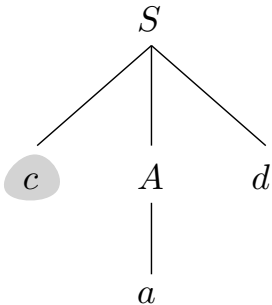
Παράδειγμα

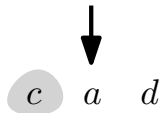




Recursive-descent Parsing

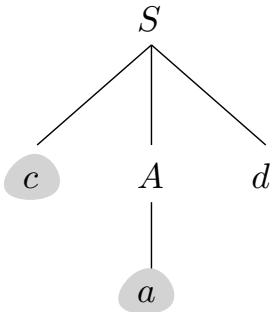
Παράδειγμα

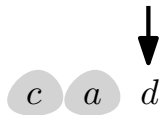




Recursive-descent Parsing

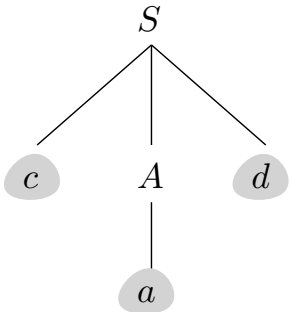
Παράδειγμα

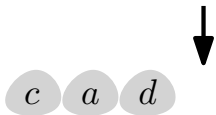




Recursive-descent Parsing

Παράδειγμα





Αριστερά Αναδρομικές Γραμματικές

Επειδή διαλέγουμε συνέχεια το πρώτο μη-τερματικό σύμβολο από αριστερά και δοκιμάζουμε κανόνες, οι μέθοδοι αυτές δεν μπορούν να λειτουργήσουν με γραμματικές που είναι *αριστερά αναδρομικές*.

αριστερά αναδρομικές γραμματικές

Μια γραμματική ονομάζεται *αριστερά αναδρομική* αν έχει ένα μη-τερματικό A για το οποίο υπάρχει παραγωγή του τύπου

$$A \stackrel{+}{\Rightarrow} A\alpha$$

Οι μέθοδοι top-down μπαίνουν σε ατέρμονο βρόγχο λόγω της προσπάθειας τους να ”περιγράψουν” το πρώτο μη-τερματικό από αριστερά.

Αριστερά Αναδρομικές Γραμματικές

π.χ

$$
\begin{array}{lcl}
E & \Rightarrow & E + T \\
E & \Rightarrow & T \\
T & \Rightarrow & T * F \\
T & \Rightarrow & F \\
F & \Rightarrow & (\ E\ ) \\
F & \Rightarrow & \mathbf{id}
\end{array}
$$

Αφαίρεση (άμεσης) Αριστερής Αναδρομής

Γενικά μπορούμε να αφαιρέσουμε την αριστερή αναδρομή κάνοντας μετασχηματισμούς στην γραμματική μας.

Στη απλή περίπτωση που υπάρχει μόνο *άμεση* αριστερή αναδρομή, δηλαδή υπάρχουν κανόνες της μορφής $A \to Aa$ μπορούμε να εφαρμόσουμε την εξής τεχνική:

- ομαδοποιούμε σε
$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$$
- και αλλάζουμε σε
$$A \to \beta_1 A' \mid b_2 A' \mid \cdots \mid \beta_n A'$$
$$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \cdots \mid \alpha_m A' \mid \epsilon$$

Αφαίρεση (άμεσης) Αριστερής Αναδρομής

π.χ

$$
\begin{array}{lcl}
E & \Rightarrow & E + T \mid T \\
T & \Rightarrow & T * F \mid F \\
F & \Rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
$$

σε

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
$$

FIRST

Η κατασκευή ενός καθοδικού αλλά και ενός ανοδικού συντακτικού αναλυτή μπορεί να βοηθηθεί από δύο συναρτήσεις, FIRST και FOLLOW.

FIRST

Ορίζουμε *FIRST*(α), όπου α είναι οποιαδήποτε συμβολοσειρά από σύμβολα της γραμματικής, ως το σύνολο των τερματικών που είναι πρώτα σε συμβολοσειρές που παράγονται από την α. Εαν $\alpha \stackrel{*}{\Rightarrow} \epsilon$ τότε $\epsilon \in$ *FIRST*(α).

Υπολογισμός FIRST

Για να υπολογίσουμε την συνάρτηση $FIRST(X)$ για κάθε σύμβολο X μιας γραμματικής, χρησιμοποιούμε τους κανόνες που ακολουθούν μέχρι να μη μπορούμε να προσθέσουμε κάποιο τερματικό ή το ϵ σε κάποιο σύνολο $FIRST$.

1. αν X είναι τερματικό τότε $FIRST(X) = \{X\}$
2. αν X είναι μη-τερματικό και $X \to Y_1 Y_2 \cdots Y_k$ είναι ένας κανόνας παραγωγής για $k \geq 1$ τότε:
   - αν για κάποιο i έχουμε πως $a \in FIRST(Y_i)$ και το ϵ ανήκει σε όλα τα $Y_1 \cdots Y_{i-1}$
   - αν $\epsilon \in FIRST(Y_j)\ \forall\ j = 1, 2, \ldots, k$ τότε πρόσθεσε το ϵ στο $FIRST(X)$
3. αν $X \to \epsilon$ είναι ένας κανόνας, πρόσθεσε το ϵ στο $FIRST(X)$

FIRST

παράδειγμα

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FIRST
παράδειγμα

- $FIRST(F) = \{(, id\}$

$$
\begin{array}{rcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FIRST
παράδειγμα

- $FIRST(F) = \{(, id\}$
- $FIRST(T) = FIRST(F) = \{(, id\}$

$$
\begin{array}{rcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FIRST
παράδειγμα

- $FIRST(F) = \{(, id\}$
- $FIRST(T) = FIRST(F) = \{(, id\}$
- $FIRST(E) = FIRST(T) = \{(, id\}$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FIRST
παράδειγμα

- $FIRST(F) = \{(, id\}$
- $FIRST(T) = FIRST(F) = \{(, id\}$
- $FIRST(E) = FIRST(T) = \{(, id\}$
- $FIRST(E') = \{+, \epsilon\}$

$$
\begin{array}{rcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FIRST

παράδειγμα

- $FIRST(F) = \{(, id\}$
- $FIRST(T) = FIRST(F) = \{(, id\}$
- $FIRST(E) = FIRST(T) = \{(, id\}$
- $FIRST(E') = \{+, \epsilon\}$
- $FIRST(T') = \{*, \epsilon\}$

$$
\begin{array}{rcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FOLLOW

Ορίζουμε *FOLLOW*(A), όπου A είναι ένα μη-τερματικό, ως το σύνολο των τερματικών a που μπορούν να εμφανιστούν αμέσως μετά το A σε κάποια συμβολοσειρά που παράχθηκε από το αρχικό σύμβολο.

Είναι δηλαδή το σύνολο από τερματικά a τέτοια ώστε να υπάρχει παραγωγή $S \stackrel{*}{\Rightarrow} qAap$ για κάποια q και p.

- Κατά την διάρκεια της παραγωγής μπορεί να υπήρχαν σύμβολα μεταξύ των A και a αλλά κατέληξαν σε ϵ.
- Επίσης αν το A μπορεί να είναι το δεξιότερο σύμβολο σε μια παραγωγή, τότε το ειδικό σύμβολο $\$$ που θεωρούμε πως δεν ανήκει σε καμία γραμματική ανήκει στο *FOLLOW*(A) συμβολίζοντας το τέλος της εισόδου (EOF).

Υπολογισμός FOLLOW

Για να υπολογίσουμε την συνάρτηση $FOLLOW(A)$ για κάθε μη-τερματικό A μιας γραμματικής, χρησιμοποιούμε τους κανόνες που ακολουθούν μέχρι να μη μπορούμε να προσθέσουμε σε κάποιο σύνολο $FOLLOW$.

1. πρόσθεσε το $ στο $FOLLOW(S)$ όπου S είναι το αρχικό σύμβολο

Υπολογισμός FOLLOW

Για να υπολογίσουμε την συνάρτηση *FOLLOW*(A) για κάθε μη-τερματικό A μιας γραμματικής, χρησιμοποιούμε τους κανόνες που ακολουθούν μέχρι να μη μπορούμε να προσθέσουμε σε κάποιο σύνολο *FOLLOW*.

1. πρόσθεσε το $ στο *FOLLOW*(S) όπου S είναι το αρχικό σύμβολο
2. εαν υπάρχει κανόνας $A \to \alpha B \beta$ τότε όλα τα *FIRST*(β) εκτός του ϵ είναι στο *FOLLOW*(B)

Υπολογισμός FOLLOW

Για να υπολογίσουμε την συνάρτηση *FOLLOW*(A) για κάθε μη-τερματικό A μιας γραμματικής, χρησιμοποιούμε τους κανόνες που ακολουθούν μέχρι να μη μπορούμε να προσθέσουμε σε κάποιο σύνολο *FOLLOW*.

1. πρόσθεσε το $ στο *FOLLOW*(S) όπου S είναι το αρχικό σύμβολο
2. εαν υπάρχει κανόνας $A \to \alpha B \beta$ τότε όλα τα *FIRST*(β) εκτός του ϵ είναι στο *FOLLOW*(B)
3. εαν υπάρχει κανόνας $A \to \alpha B$ ή κανόνας $A \to \alpha B \beta$ όπου το *FIRST*(β) περιέχει το ϵ, τότε όλα τα *FOLLOW*(A) ανήκουν στο *FOLLOW*(B)

FOLLOW

παράδειγμα

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & (\,E\,) \mid \mathbf{id}
\end{array}
$$

FOLLOW

παράδειγμα

- $FOLLOW(E) = \{), \$\}$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FOLLOW

παράδειγμα

- $FOLLOW(E) = \{), \$\}$
- $FOLLOW(E') = FOLLOW(E) = \{), \$\}$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & (\,E\,) \mid \mathbf{id}
\end{array}
$$

FOLLOW

παράδειγμα

- $FOLLOW(E) = \{), \$\}$
- $FOLLOW(E') = FOLLOW(E) = \{), \$\}$
- $FOLLOW(T) = \{+,), \$\}$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & (\,E\,) \mid \mathbf{id}
\end{array}
$$

FOLLOW

παράδειγμα

- $FOLLOW(E) = \{), \$\}$
- $FOLLOW(E') = FOLLOW(E) = \{), \$\}$
- $FOLLOW(T) = \{+,), \$\}$
- $FOLLOW(T') = FOLLOW(T) = \{+,), \$\}$

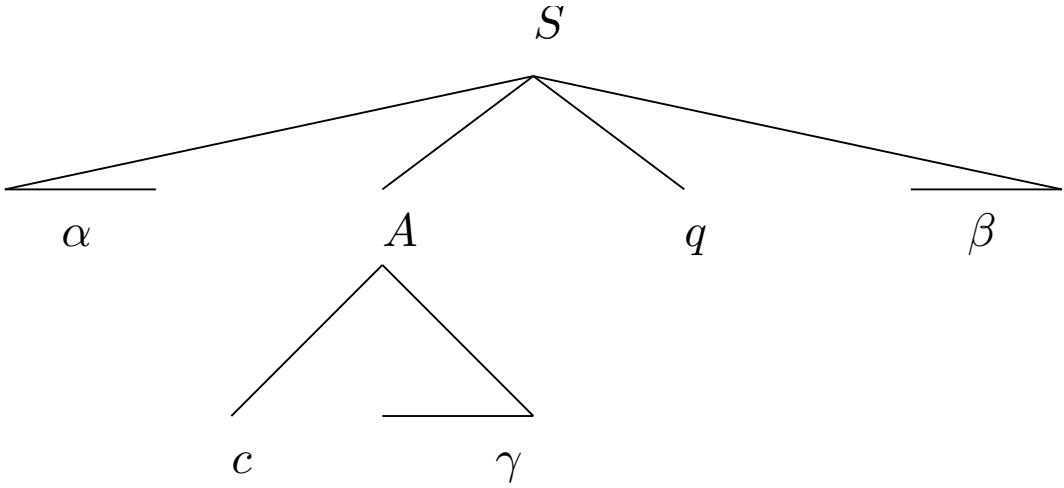
$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

FOLLOW

παράδειγμα

- $FOLLOW(E) = \{), \$\}$
- $FOLLOW(E') = FOLLOW(E) = \{), \$\}$
- $FOLLOW(T) = \{+,), \$\}$
- $FOLLOW(T') = FOLLOW(T) = \{+,), \$\}$
- $FOLLOW(F) = \{+, *,), \$\}$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \textbf{(}\,E\,\textbf{)} \mid \textbf{id}
\end{array}
$$



$c \subset FIRST(A)$

Προγνωστικός Συντακτικός Αναλυτής

Predictive Parsing

Στόχος μας είναι να κατασκευάσουμε έναν *recursive-descent parser* που να μην χρειάζεται να κάνει backtracking.

Η μέθοδος του *predictive parsing* λειτουργεί μόνο σε γραμματικές όπου το πρώτο τερματικό σύμβολο κάθε υποέκφρασης περιέχει όλη την πληροφορία που χρειαζόμαστε ώστε να αποφασίσουμε τον σωστό κανόνα παραγωγής.

Γραμματικές LL(1)

Προγνωστικοί συντακτικοί αναλυτές μπορούν να κατασκευαστούν για μια κλάση γραμματικών που ονομάζονται LL(1).

Η ονομασία LL(1) προκύπτει από τα παρακάτω:

- scanning input from **L**eft to right
- producing a **L**eftmost derivation
- using **1** input symbol of lookahead at each step to make parsing action decisions

Προγνωστικός Συντακτικός Αναλυτής

Από τα σύνολα FIRST και FOLLOW μπορούμε να κατασκευάσουμε έναν πίνακα, τον *προγνωστικό πίνακα συντακτικής ανάλυσης*.

- διδιάστατος πίνακας $M[A, a]$ όπου A μη-τερματικό και a τερματικό σύμβολο ή το σύμβολο $\$$
- η τιμή $M[A, a]$ μας λέει τον κανόνα που να ακολουθήσουμε αν ήμαστε στην συνάρτηση που αντιστοιχεί στο μη-τερματικό A και βλέπουμε στην είσοδο το τερματικό a

Προγνωστικός Συντακτικός Αναλυτής

Τρόπος υπολογισμού πίνακα προγνωστικής συντακτικής ανάλυσης από τα σύνολα FIRST και FOLLOW.

Για κάθε κανόνα παραγωγής $A \to u$ κάνε τα εξής:

- για κάθε τερματικό a που ανήκει στο $FIRST(u)$ πρόσθεσε τον κανόνα $A \to u$ στο $M[A, a]$

Προγνωστικός Συντακτικός Αναλυτής

Τρόπος υπολογισμού πίνακα προγνωστικής συντακτικής ανάλυσης από τα σύνολα FIRST και FOLLOW.

Για κάθε κανόνα παραγωγής $A \to u$ κάνε τα εξής:

- για κάθε τερματικό a που ανήκει στο $FIRST(u)$ πρόσθεσε τον κανόνα $A \to u$ στο $M[A, a]$
- εαν $\epsilon \in FIRST(u)$ τότε για κάθε τερματικό $b \in FOLLOW(A)$ πρόσθεσε τον κανόνα $A \to u$ στο $M[A, b]$.
  Εαν $\epsilon \in FIRST(u)$ και $\$ \in FOLLOW(A)$, πρόσθεσε τον κανόνα $A \to u$ επίσης στο $M[A, \$]$.

Προγνωστικός Συντακτικός Αναλυτής

Τρόπος υπολογισμού πίνακα προγνωστικής συντακτικής ανάλυσης από τα σύνολα FIRST και FOLLOW.

Για κάθε κανόνα παραγωγής $A \to u$ κάνε τα εξής:

- για κάθε τερματικό a που ανήκει στο $FIRST(u)$ πρόσθεσε τον κανόνα $A \to u$ στο $M[A, a]$
- εαν $\epsilon \in FIRST(u)$ τότε για κάθε τερματικό $b \in FOLLOW(A)$ πρόσθεσε τον κανόνα $A \to u$ στο $M[A, b]$.
  Εαν $\epsilon \in FIRST(u)$ και $\$ \in FOLLOW(A)$, πρόσθεσε τον κανόνα $A \to u$ επίσης στο $M[A, \$]$.
- εαν η θέση $M[A, u]$ είναι κενή τότε θέσε την θέση αυτή σε ”λάθος”

Παράδειγμα

Για τον κανόνα $E \to TE'$ έχουμε

- $FIRST(TE') = FIRST(T) = \{(, id\}$
- άρα μπορούμε να προσθέσουμε τον κανόνα αυτό στην θέση $M[E, (]$ και στην θέση $M[E, id]$

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

Παράδειγμα

Για τον κανόνα $E \to TE'$ έχουμε

- $FIRST(TE') = FIRST(T) = \{(, id\}$
- άρα μπορούμε να προσθέσουμε τον κανόνα αυτό στην θέση $M[E, (]$ και στην θέση $M[E, id]$

Για τον κανόνα $E' \to +TE'$ έχουμε

- $FIRST(+TE') = \{+\}$ και άρα προσθέτουμε τον κανόνα αυτό στην θέση $M[E', +]$

E	$\Rightarrow$	$T\,E'$
E'	$\Rightarrow$	$+\,T\,E' \mid \epsilon$
T	$\Rightarrow$	$F\,T'$
T'	$\Rightarrow$	$*\,F\,T' \mid \epsilon$
F	$\Rightarrow$	**(** E **)** | **id**

Παράδειγμα

Για τον κανόνα $E \to TE'$ έχουμε

- $FIRST(TE') = FIRST(T) = \{(, id\}$
- άρα μπορούμε να προσθέσουμε τον κανόνα αυτό στην θέση $M[E, (]$ και στην θέση $M[E, id]$

Για τον κανόνα $E' \to +TE'$ έχουμε

- $FIRST(+TE') = \{+\}$ και άρα προσθέτουμε τον κανόνα αυτό στην θέση $M[E', +]$

Για τον κανόνα $E' \to \epsilon$ έχουμε

- $FIRST(\epsilon) = \{\epsilon\}$ και επειδή $FOLLOW(E') = \{), \$\}$ προσθέτουμε τον κανόνα αυτό στις θέσεις $M[E',)]$ και $M[E', \$]$.

$$
\begin{array}{lcl}
E & \Rightarrow & T\,E' \\
E' & \Rightarrow & +\,T\,E' \mid \epsilon \\
T & \Rightarrow & F\,T' \\
T' & \Rightarrow & *\,F\,T' \mid \epsilon \\
F & \Rightarrow & \mathbf{(}\,E\,\mathbf{)} \mid \mathbf{id}
\end{array}
$$

Παράδειγμα

NON TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to id$			$F \to (E)$		

Άσκηση: Χρησιμοποιήστε τον παραπάνω πίνακα για να κάνετε συντακτική ανάλυση στην συμβολοσειρά

$$id + id + id \ \$$$

Γραμματικές LL(1)

Μια γραμματική ανήκει στην κλάση **LL(1)** εαν ο αλγόριθμος κατασκευής του πίνακα προγνωστικής ανάλυσης παράγει έναν πίνακα M που δεν έχει καμία θέση με περισσότερους από ένα κανόνες.

Πίνακας Προγνωστικής Ανάλυσης

Για κάθε LL(1) γραμματική ο πίνακας αυτός έχει σε κάθε θέση είτε έναν κανόνα είτε ”λάθος”.

Αυτό όμως δεν ισχύει για τις εξής γραμματικές όπου μπορούμε να έχουμε περισσότερους από έναν κανόνες ανά θέση του πίνακα:

- αριστερά αναδρομική γραμματική (left-recursive)
- διφορούμενη γραμματική (ambiguity)
- κανόνες με κοινά προθέματα (left-factoring)

Κανόνες με Κοινά Προθέματα

Left Factoring

Μερικές γραμματικές έχουν κανόνες με κοινά προθέματα, π.χ

stmt	$\Rightarrow$	**if** *expr* **then** *stmt* **else** *stmt*
stmt	$\Rightarrow$	**if** *expr* **then** *stmt*

Ένας συντακτικός αναλυτής βλέποντας το **if** δεν γνωρίζει ποιον κανόνα από τους δύο να ακολουθήσει.

Σε αυτή την περίπτωση κάνουμε μετασχηματισμούς στην γραμματική μας ώστε να πάρουμε μια ισοδύναμη γραμματική χωρίς κοινά προθέματα.

Η τεχνική είναι η εξής. Αν

$$\begin{array}{lcl} A & \to & \alpha\beta_1 \\ A & \to & \alpha\beta_2 \end{array}$$

και βλέπουμε στην είσοδο α απλά καθυστερούμε την απόφαση.

$$\begin{array}{lcl} A & \to & \alpha A' \\ A' & \to & \beta_1 \\ A' & \to & \beta_2 \end{array}$$

Έχουμε ήδη δει πως διφορούμενες γραμματικές είναι εκείνες που έχουν δύο ή περισσότερα συντακτικά δέντρα για την ίδια έκφραση.

Το πιο γνωστό παράδειγμα είναι το *ξεκρέμαστο (dangling) else*.

stmt	→	**if** *expr* **then** *stmt* **else** *stmt*
	|	**if** *expr* **then** *stmt*
	|	other

Η συμβολοσειρά

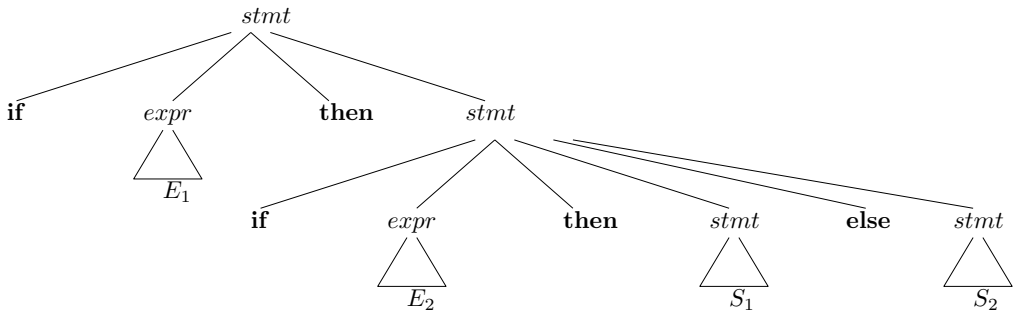
$$\textbf{if } E_1 \textbf{ then if } E_2 \textbf{ then } S_1 \textbf{ else } S_2$$

έχει δύο συντακτικά δέντρα.

Η συμβολοσειρά

if E_1 **then if** E_2 **then** S_1 **else** S_2

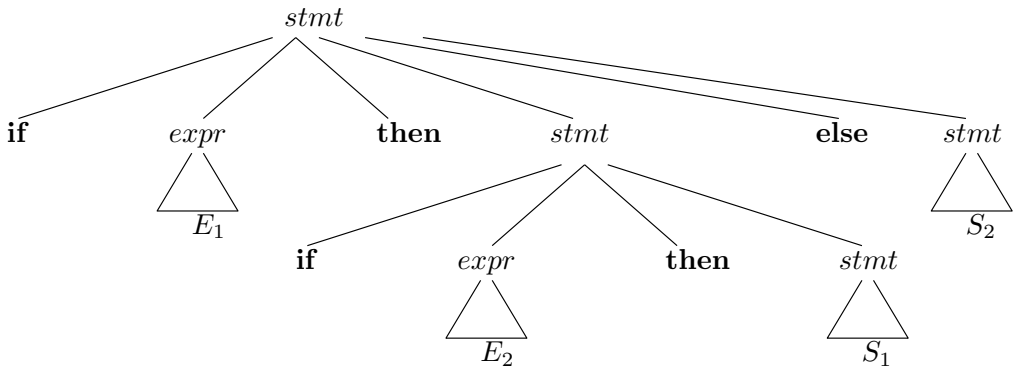
έχει δύο συντακτικά δέντρα.



Η συμβολοσειρά

if E_1 **then if** E_2 **then** S_1 **else** S_2

έχει δύο συντακτικά δέντρα.



Dangling-else

Στις γλώσσες προγραμματισμού που υποστηρίζουν **if** με αυτή την μορφή, το πρώτο συντακτικό δέντρο είναι το σωστό.

κανόνας

Ταίριαξε κάθε **else** με το αταίριαστο **then** που είναι πιο κοντά.

Αυτός ο κανόνας μπορεί να ενταχθεί μέσα στην γραμματική, αλλά στην πράξη δεν συμβαίνει.

Dangling-else

Η ιδέα για την μετατροπή της γραμματικής σε μη-διφορούμενη είναι πως μια πρόταση που εμφανίζεται μεταξύ ενός **then** και ενός **else** πρέπει να είναι ”ταιριασμένη”.

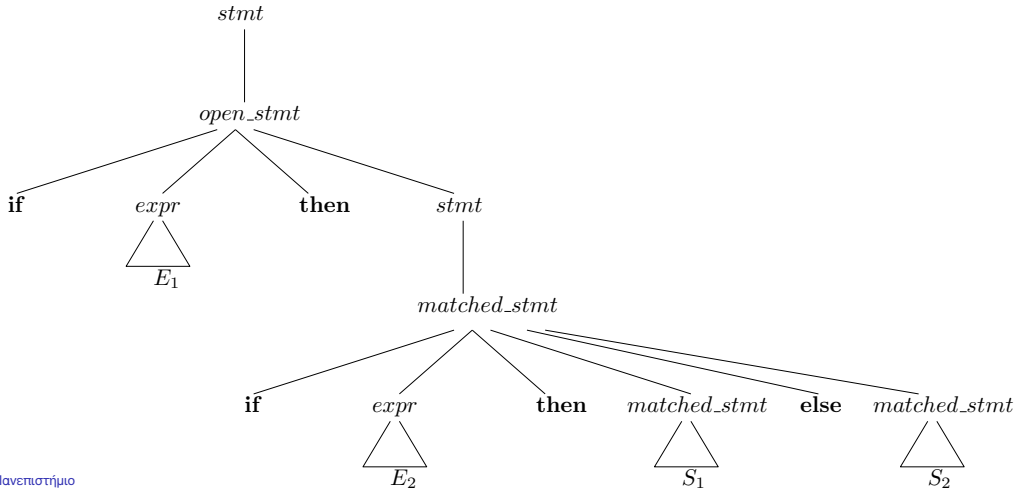
stmt	$\rightarrow$	*matched_stmt*
	|	*open_stmt*
matched_stmt	$\rightarrow$	**if** *expr* **then** *matched_stmt* **else** *matched_stmt*
	|	other
open_stmt	$\rightarrow$	**if** *expr* **then** *stmt*
	|	**if** *expr* **then** *matched_stmt* **else** *open_stmt*

Dangling-else

Η συμβολοσειρά

if E_1 **then if** E_2 **then** S_1 **else** S_2

έχει τώρα μόνο ένα συντακτικό δέντρο.



Άλλες γραμματικές

Μερικά χαρακτηριστικά των γλωσσών προγραμματισμού δεν μπορούν να περιγραφούν μόνο με γραμματικές χωρίς συμφραζόμενα.

Άλλες γραμματικές

Μερικά χαρακτηριστικά των γλωσσών προγραμματισμού δεν μπορούν να περιγραφούν μόνο με γραμματικές χωρίς συμφραζόμενα.

Η γλώσσα $L = \{wcw | w \text{ is in } (a|b)*\}$ και το γεγονός πως δεν είναι γλώσσα χωρίς συμφραζόμενα δείχνει πως και οι γλώσσες *C* και *Java* δεν είναι γλώσσες χωρίς συμφραζόμενα.

Άλλες γραμματικές

Μερικά χαρακτηριστικά των γλωσσών προγραμματισμού δεν μπορούν να περιγραφούν μόνο με γραμματικές χωρίς συμφραζόμενα.

Η γλώσσα $L = \{wcw | w \text{ is in } (a|b)*\}$ και το γεγονός πως δεν είναι γλώσσα χωρίς συμφραζόμενα δείχνει πως και οι γλώσσες *C* και *Java* δεν είναι γλώσσες χωρίς συμφραζόμενα.

Η γλώσσα *L* αντιπροσωπεύει το γεγονός πως τα αναγνωριστικά στις γλώσσες αυτές μπορεί να έχουν όσο μήκος θέλουν και ταυτόχρονα πρέπει να δηλώνονται πριν χρησιμοποιηθούν.

Άλλες γραμματικές

Μερικά χαρακτηριστικά των γλωσσών προγραμματισμού δεν μπορούν να περιγραφούν μόνο με γραμματικές χωρίς συμφραζόμενα.

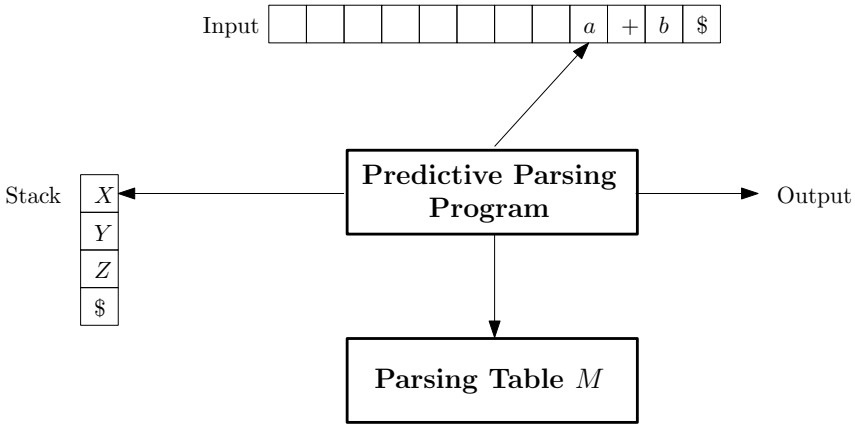
Η γλώσσα $L = \{wcw | w \text{ is in } (a|b)*\}$ και το γεγονός πως δεν είναι γλώσσα χωρίς συμφραζόμενα δείχνει πως και οι γλώσσες *C* και *Java* δεν είναι γλώσσες χωρίς συμφραζόμενα.

Η γλώσσα *L* αντιπροσωπεύει το γεγονός πως τα αναγνωριστικά στις γλώσσες αυτές μπορεί να έχουν όσο μήκος θέλουν και ταυτόχρονα πρέπει να δηλώνονται πριν χρησιμοποιηθούν.

Οι μεταγλωττιστές αντιπροσωπεύουν όλα τα αναγνωριστικά με **id** και ελέγχουν σε επόμενες φάσεις (σημασιολογική ανάλυση) πως έχουν δηλωθεί πριν χρησιμοποιηθούν.

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

Μπορούμε να αντικαταστήσουμε την αναδρομή με μια στοίβα



Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

Αλγόριθμος

Στην αρχή ο συντακτικός αναλυτής έχει $w\$$ στην είσοδο και το αρχικό σύμβολο S της γραμματικής στην κορυφή της στοίβας επάνω από το σύμβολο $\$$.

έστω a το πρώτο σύμβολο της w
έστω X το σύμβολο στην κορυφή της στοίβας
while $X \neq \$$ **do**
  if $X = a$ **then**
    αφαίρεσε την κορυφή της στοίβας και έστω a το επόμενο σύμβολο της w
  else if X *είναι τερματικό σύμβολο* **then**
    error()
  else if $M[X, a]$ *είναι κενή (λάθος)* **then**
    error()
  else if $M[X, a] = X \to Y_1 Y_2 \cdots Y_k$ **then**
    τύπωσε τον κανόνα $X \to Y_1 Y_2 \cdots Y_k$
    αφαίρεσε την κορυφή της στοίβας
    πρόσθεσε τα $Y_k, Y_{k-1}, \ldots, Y_1$ με αυτή την σειρά στην στοίβα
  έστω X το σύμβολο που είναι στην κορυφή της στοίβας
end

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\,T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\,T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match $\mathbf{id}$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\,T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$E'\$$	$\$$	output $T' \to \epsilon$

Μη-αναδρομικός Προγνωστικός Συντακτικός Αναλυτής

MATCHED	STACK	INPUT	ACTION
	$E\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
	$TE'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $E \to TE'$
	$FT'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
	$\mathbf{id}\,T'E'\$$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id}$	$T'E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id}$	$E'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $T' \to \epsilon$
$\mathbf{id}$	$+TE'\$$	$+\,\mathbf{id} * \mathbf{id}\$$	print $E' \to +TE'$
$\mathbf{id} +$	$TE'\$$	$\mathbf{id} * \mathbf{id}\$$	match $+$
$\mathbf{id} +$	$FT'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $T \to FT'$
$\mathbf{id} +$	$\mathbf{id}\,T'E'\$$	$\mathbf{id} * \mathbf{id}\$$	print $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$T'E'\$$	$* \mathbf{id}\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id}$	$*FT'E'\$$	$* \mathbf{id}\$$	output $T' \to *FT'$
$\mathbf{id} + \mathbf{id} *$	$FT'E'\$$	$\mathbf{id}\$$	match $*$
$\mathbf{id} + \mathbf{id} *$	$\mathbf{id}\,T'E'\$$	$\mathbf{id}\$$	output $F \to \mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$T'E'\$$	$\$$	match $\mathbf{id}$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$E'\$$	$\$$	output $T' \to \epsilon$
$\mathbf{id} + \mathbf{id} * \mathbf{id}$	$\$$	$\$$	output $E' \to \epsilon$

Γραμματικές LL(k)

Μπορούμε να γενικοποιήσουμε τις γραμματικές χρησιμοποιώντας τα *k* πρώτα σύμβολα για να πάρουμε αποφάσεις.

Προγνωστικοί συντακτικοί αναλυτές μπορούν να κατασκευαστούν για μια κλάση γραμματικών που ονομάζονται LL(k).

Η ονομασία LL(k) προκύπτει από τα παρακάτω:

- scanning input from **L**eft to right
- producing a **L**eftmost derivation
- using **k** input symbol of lookahead at each step to make parsing action decisions

Ο πίνακας προγνωστικής ανάλυσης που προκύπτει (στην χειρότερη περίπτωση) είναι πολύ μεγάλος αφού μπορεί να έχει εκθετικό μέγεθος.