ΗΥ280 / ΘΕΩΡΙΑ ΥΠΟΛΟΓΙΣΜΟΥ

Γραμματικές Χωρίς Συμφραζόμενα

Αυτόματα Στοίβας (PDA) και Λήμμα Άντλησης

Χαράλαμπος Ε. Τσουρακάκης



Διαλέξεις 14, 15, 16, 17

HY280 · Θεωρία Υπολογισμού

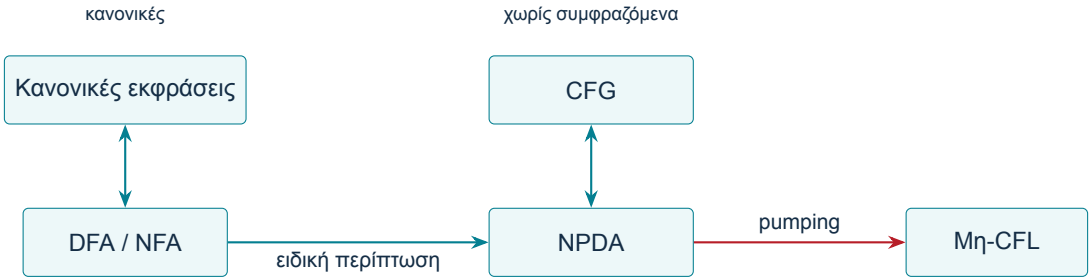
1. Βασικές έννοιες CFG

Γραμματικές, παράγωγα, δέντρα ανάλυσης

Πέρα από τις κανονικές γλώσσες

- Είδαμε πως μια γλώσσα είναι κανονική (regular) αν χτίζεται από μεμονωμένες συμβολοσειρές με
  1. *συνένωση* (concatenation)
  2. *ένωση* (union)
  3. *επανάληψη*
- Όπως είδαμε, δεν μπορούμε να αναγνωρίσουμε εξαιρετικά απλές γλώσσες (πχ. ίσο πλήθος αριστερών και δεξιών παρενθέσεων) με τις κανονικές εκφράσεις.
- Στο κεφάλαιο αυτό μελετάμε μια ευρύτερη κλάση: τις γλώσσες χωρίς συμφραζόμενα (context-free languages), που χτίζονται με
  1. *συνένωση* (concatenation)
  2. *ένωση* (union)
  3. αναδρομή
- Τυπικά: μια γλώσσα είναι χωρίς συμφραζόμενα αν και μόνο αν έχει μια αναδρομική περιγραφή ενός συγκεκριμένου τύπου, μια γραμματική χωρίς συμφραζόμενα (context-free grammar, CFG).
- Οι κανονικές γλώσσες είναι ειδική περίπτωση γλώσσας χωρίς συμφραζόμενα

Που βρισκόμαστε ως τώρα στο HY280;



Βιβλίο Sipser

Κεφάλαιο 2 (Ασυμφραστικές Γλώσσες), και συγκεκριμένα 2.1-2.3 (όχι το 2.4 αιτιοκρατικές ασυμφραστικές γλώσσες)

Γραμματική χωρίς συμφραζόμενα

Ορισμός 1

Μια γραμματική χωρίς συμφραζόμενα (context-free grammar) αποτελείται από:

- ένα πεπερασμένο σύνολο Σ, τα στοιχεία του οποίου λέγονται σύμβολα ή τερματικά (terminals),
- ένα πεπερασμένο σύνολο Γ, ξένο προς το Σ, τα στοιχεία του οποίου λέγονται μη τερματικά (non-terminals),
- ένα πεπερασμένο σύνολο R κανόνων παραγωγής (production rules) της μορφής $A \to w$, όπου $A \in \Gamma$ και $w \in (\Sigma \cup \Gamma)^*$,
- ένα αρχικό μη τερματικό (starting non-terminal), συνήθως S.

Συμβάσεις συμβολισμού

Στο υπόλοιπο κεφάλαιο χρησιμοποιούμε σχεδόν πάντα τις εξής συμβάσεις:

- Ψηφία και σύμβολα σε γραμματοσειρά σταθερού πλάτους (`0`, `1`, `2`, . . ., `(`, `$`, `#`, . . .) είναι συγκεκριμένα τερματικά.
- Πρώτα πεζά λατινικά γράμματα ($a, b, c, \ldots$) είναι άγνωστα ή αυθαίρετα τερματικά του Σ.
- Κεφαλαία λατινικά γράμματα ($A, B, C, \ldots$) και το S είναι μη τερματικά του Γ.
- Τελευταία πεζά λατινικά γράμματα ($\ldots, w, x, y, z$) είναι συμβολοσειρές στο $(\Sigma \cup \Gamma)^*$, δηλαδή μπορεί να περιέχουν τερματικά και μη τερματικά.

Γραμματικές και φυσική γλώσσα

Μπορεί μια γραμματική να παράγει ελληνικές προτάσεις;

Μια γραμματική χωρίς συμφραζόμενα μπορεί να περιγράψει ένα απλό υποσύνολο της ελληνικής γλώσσας.

Για παράδειγμα, θέλουμε να παράγουμε προτάσεις όπως:

- **ένα αγόρι βλέπει**
- **το αγόρι βλέπει ένα λουλούδι**
- **το κορίτσι με το λουλούδι αγαπά ένα αγόρι**

Βασική ιδέα

Μια πρόταση αποτελείται από ένα **ονοματικό σύνολο** και ένα **ρηματικό σύνολο**:

$$\langle \text{ΠΡΟΤΑΣΗ} \rangle \to \langle \text{ΟΝΟΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \ \langle \text{ΡΗΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle .$$

Ερώτηση: Ποια μέρη μιας πρότασης μπορούν να αντικατασταθούν από μεγαλύτερες συντακτικές δομές;

Μια CFG για απλές ελληνικές προτάσεις (1/2, βιβλίο Sipser 2.1)

$$
\begin{aligned}
\langle \text{ΠΡΟΤΑΣΗ} \rangle &\to \langle \text{ΟΝΟΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \langle \text{ΡΗΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \\
\langle \text{ΟΝΟΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle &\to \langle \text{ΑΠΛΟ-ΟΣ} \rangle \mid \langle \text{ΑΠΛΟ-ΟΣ} \rangle \langle \text{ΠΡΟΘΕΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \\
\langle \text{ΡΗΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle &\to \langle \text{ΑΠΛΟ-ΡΣ} \rangle \mid \langle \text{ΑΠΛΟ-ΡΣ} \rangle \langle \text{ΠΡΟΘΕΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \\
\langle \text{ΠΡΟΘΕΤΙΚΟ-ΣΥΝΟΛΟ} \rangle &\to \langle \text{ΠΡΟΘΕΣΗ} \rangle \langle \text{ΑΠΛΟ-ΟΣ} \rangle \\
\langle \text{ΑΠΛΟ-ΟΣ} \rangle &\to \langle \text{ΑΡΘΡΟ} \rangle \langle \text{ΟΥΣΙΑΣΤΙΚΟ} \rangle \\
\langle \text{ΑΠΛΟ-ΡΣ} \rangle &\to \langle \text{ΡΗΜΑ} \rangle \mid \langle \text{ΡΗΜΑ} \rangle \langle \text{ΟΝΟΜΑΤΙΚΟ-ΣΥΝΟΛΟ} \rangle \\
\langle \text{ΑΡΘΡΟ} \rangle &\to \text{ενα} \mid \text{το} \\
\langle \text{ΟΥΣΙΑΣΤΙΚΟ} \rangle &\to \text{αγορι} \mid \text{κοριτσι} \mid \text{λουλουδι} \\
\langle \text{ΡΗΜΑ} \rangle &\to \text{αγγιζει} \mid \text{αγαπα} \mid \text{βλεπει} \\
\langle \text{ΠΡΟΘΕΣΗ} \rangle &\to \text{με}
\end{aligned}
$$

Η G_2 έχει 10 μεταβλητές (τους γραμματικούς όρους που είναι τυπωμένοι με πεζοκεφαλαία στοιχεία εντός αγκυλών), 25 τερματικά σύμβολα (τα γράμματα του ελληνικού αλφαβήτου συν το σύμβολο του διαστήματος), και 18 κανόνες. Οι λέ-

Συντομογραφία

Για συντομία, χρησιμοποιούμε τις μεταβλητές:

$$P = \text{ΠΡΟΤΑΣΗ}, \qquad O = \text{ΟΝΟΜΑΤΙΚΟ-ΣΥΝΟΛΟ},$$
$$R = \text{ΡΗΜΑΤΙΚΟ-ΣΥΝΟΛΟ}, \qquad D = \text{ΠΡΟΘΕΤΙΚΟ-ΣΥΝΟΛΟ},$$
$$N = \text{ΑΠΛΟ-ΟΣ}, \qquad V = \text{ΑΠΛΟ-ΡΣ } \textbf{κτλ.}$$

Σημαντική διάκριση

Τα $P, O, R, \ldots$ είναι **μεταβλητές**. Οι λέξεις «ένα», «αγόρι», «βλέπει», «με» κλπ. είναι **τερματικά σύμβολα**.

Πώς θα δείξουμε ότι

$$\text{ένα αγόρι βλέπει} \in L(G_2);$$

- Ξεκινάμε από την αρχική μεταβλητή P.
- Σε κάθε βήμα αντικαθιστούμε μία μεταβλητή.
- Στο τέλος απομένουν μόνο τερματικά σύμβολα.

Άσκηση: Φτιάξτε μια μεγαλύτερη πρόταση

Ερώτηση κατανόησης

Η γραμματική μπορεί επίσης να παράγει την πρόταση «το λουλούδι αγαπά ένα αγόρι»;
Αν ναι, τι μας λέει αυτό για τη διαφορά μεταξύ **συντακτικής ορθότητας** και **σημασιολογικής ορθότητας**;

Τι είναι το Σ; Δύο επιλογές μοντελοποίησης

Ίδιες συντακτικές κατηγορίες

$$\mathcal{V} = \{P, O, R, D, N, V, A, T, H, M\}, \qquad \text{αρχική μεταβλητή } P.$$

P: πρόταση, *O*: ονοματικό σύνολο, *R*: ρηματικό σύνολο, *D*: προθετικό σύνολο, *N*: απλό ΟΣ, *V*: απλό ΡΣ, *A*: άρθρο, *T*: ουσιαστικό, *H*: ρήμα, *M*: πρόθεση.

Εκδοχή A (Sipser): χαρακτήρες

$$\Sigma_{\text{char}} = \{\alpha, \beta, \gamma, \ldots, \omega, \sqcup\}, \quad |\Sigma_{\text{char}}| = 25.$$

Η λέξη **αγορι** είναι *συμβολοσειρά* πέντε τερματικών.
Το $\sqcup$ είναι το κενό.

Εκδοχή B: tokens

$$\Sigma_{\text{tok}} = \{\text{ένα}, \text{το}, \text{αγόρι}, \text{κορίτσι}, \text{λουλούδι}, \text{αγγίζει}, \text{αγαπά}, \text{βλέπει}, \text{με}\}.$$

$|\Sigma_{\text{tok}}| = 9$.
Κάθε **ολόκληρη λέξη** είναι ένα τερματικό.

Προσοχή

Και οι δύο ορισμοί είναι έγκυροι, αλλά ορίζουν **διαφορετικά τυπικά αλφάβητα**.

Εκδοχή Α: τα τερματικά είναι χαρακτήρες

$$G_{\text{char}} = (\mathcal{V}, \Sigma_{\text{char}}, \mathcal{R}_{\text{char}}, P), \qquad |\Sigma_{\text{char}}| = 25.$$

Συντακτικοί κανόνες	Λεξικοί κανόνες
$P \to O \sqcup R$	$A \to$ ενα | το
$O \to N \mid N \sqcup D$	$T \to$ αγορι | κοριτσι
$R \to V \mid V \sqcup D$	| λουλουδι
$D \to M \sqcup N$	$H \to$ αγγιζει | αγαπα
$N \to A \sqcup T$	| βλεπει
$V \to H \mid H \sqcup O$	$M \to$ με

$$P \Rightarrow^* \text{ενα} \sqcup \text{αγορι} \sqcup \text{βλεπει} \in \Sigma_{\text{char}}^*.$$

Κρίσιμο: Το $\sqcup$ πρέπει να εμφανίζεται ρητά στους κανόνες. Οι λέξεις γράφονται χωρίς τόνους, όπως στο παράδειγμα του βιβλίου.

Εκδοχή Β: τα τερματικά είναι λέξεις–tokens

$$G_{\text{tok}} = (\mathcal{V}, \Sigma_{\text{tok}}, \mathcal{R}_{\text{tok}}, P), \qquad |\Sigma_{\text{tok}}| = 9.$$

Συντακτικοί κανόνες	Λεξικοί κανόνες
$P \to OR$	$A \to$ ένα | το
$O \to N \mid ND$	$T \to$ αγόρι | κορίτσι
$R \to V \mid VD$	| λουλούδι
$D \to MN$	$H \to$ αγγίζει | αγαπά
$N \to AT$	| βλέπει
$V \to H \mid HO$	$M \to$ με

$$P \Rightarrow^* \underbrace{\text{ένα}}_{\in \Sigma_{\text{tok}}} \underbrace{\text{αγόρι}}_{\in \Sigma_{\text{tok}}} \underbrace{\text{βλέπει}}_{\in \Sigma_{\text{tok}}}.$$

Κρίσιμο: Τα κενά ανάμεσα στις λέξεις είναι μόνο τρόπος παρουσίασης της ακολουθίας των tokens, όχι τερματικά σύμβολα.

Προγραμματιστικές γλώσσες: γιατί χρειαζόμαστε CFG;

Java: η βασική σύνταξη περιγράφεται με CFG

Η Java διαθέτει γραμματική για τη **συντακτική δομή** εντολών, εκφράσεων και εμφωλευμένων blocks. Για παράδειγμα, ένα απλουστευμένο υποσύνολο είναι:

$$
\begin{aligned}
\mathit{Block} &\to \{\ \mathit{List}\ \} \\
\mathit{List} &\to \mathit{Stmt}\ \mathit{List} \mid \varepsilon \\
\mathit{Stmt} &\to \texttt{id} = \mathit{Expr}\ ; \mid \mathit{Block} \\
\mathit{Expr} &\to \mathit{Expr} + \mathit{Term} \mid \mathit{Term} \\
\mathit{Term} &\to \texttt{id} \mid \texttt{num}
\end{aligned}
$$

Γιατί είναι χρήσιμη η αναδρομή;

Ο κανόνας $\mathit{Stmt} \to \mathit{Block}$ επιτρέπει **απεριόριστα εμφωλευμένα blocks**.

Και η C;

Η βασική σύνταξή της είναι επίσης γραμματική-βασισμένη, αλλά ορισμένες αποφάσεις ανάλυσης εξαρτώνται από προηγούμενες δηλώσεις.

C: η ίδια γραμμή, δύο διαφορετικές ερμηνείες

Περίπτωση A: το T είναι τύπος

```
typedef int T;
void f(void) {
T * p;
}
```

Ερμηνεία: δήλωση της p ως δείκτη σε `int`.

Περίπτωση B: το T είναι μεταβλητή

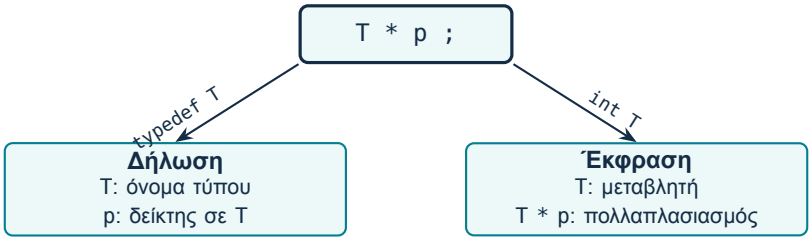
```
int T = 2, p = 3;
void f(void) {
T * p;
}
```

Ερμηνεία: εντολή έκφρασης που υπολογίζει το γινόμενο `T * p`.

Πού βρίσκεται η δυσκολία;

Η ακολουθία χαρακτήρων `T * p;` είναι **ίδια**.
Για να αναγνωρίσουμε τη δομή της, πρέπει να ξέρουμε αν το T έχει δηλωθεί ως `typedef-name` ή ως μεταβλητή.

C: γιατί δεν αρκεί η απλή συντακτική ανάλυση;



Τι κάνουν οι μεταγλωττιστές;

Χρησιμοποιούν πληροφορία από τις **δηλώσεις και τα ορατά ονόματα** (*symbol table / scope*) για να αποσαφηνίσουν τη σύνταξη, είτε κατά το parsing είτε σε επόμενο στάδιο.

Προσοχή: Το παράδειγμα *δεν αποδεικνύει* ότι η γλώσσα όλων των προγραμμάτων C δεν είναι context-free. Δείχνει γιατί μια απλή CFG χωρίς επιπλέον πληροφορία δεν αρκεί για τον συνήθη τρόπο ανάλυσης της C.

Συντακτικά σωστό $\neq$ σημασιολογικά σωστό

Πρόγραμμα Α: έγκυρο

```
void f(void) {
int x = 5;
x = x + 1;
}
```

Η μεταβλητή x έχει δηλωθεί.

Πρόγραμμα Β: σφάλμα

```
void f(void) {
int x = 5;
y = x + 1;
}
```

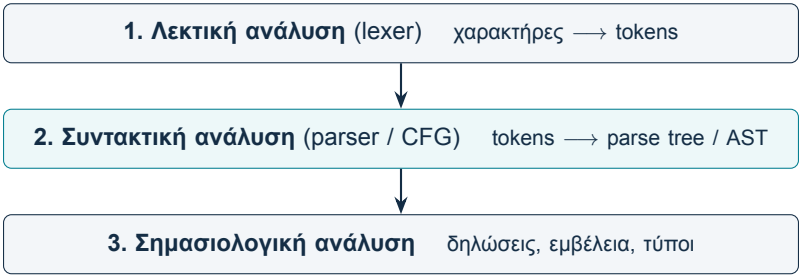
Η μεταβλητή y **δεν έχει δηλωθεί**.

Τι αναγνωρίζει η CFG;

Και τα δύο προγράμματα έχουν την ίδια **συντακτική μορφή**: δήλωση μεταβλητής και ανάθεση τιμής.
Ο parser μπορεί να κατασκευάσει *parse tree / AST* και για τα δύο.

Το σφάλμα του Β εντοπίζεται από τη **σημασιολογική ανάλυση** (name resolution / έλεγχος δηλώσεων).

Ο μεταγλωττιστής χρειάζεται περισσότερα από μια CFG



Δύο διαφορετικά ερωτήματα

- **Σύνταξη:** «Έχει το πρόγραμμα επιτρεπτή γραμματική δομή;»
- **Σημασιολογία:** «Έχουν δηλωθεί τα ονόματα; Είναι οι τύποι συμβατοί;»

Ερώτηση στην τάξη: Σε ποιο στάδιο εντοπίζουμε το `y` που δεν έχει δηλωθεί; Και γιατί το `typedef` της C μπορεί να επηρεάζει ήδη το parsing;

Ποιες γλώσσες είναι «CFG»;

Τι συγκρίνουμε;

Πόσο καλά περιγράφεται η **σύνταξη** από μια απλή CFG (όχι αν ένα πρόγραμμα περνά όλους τους σημασιολογικούς ελέγχους).

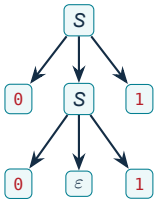
Γλώσσα	Σχέση με CFG	Γιατί;
Pascal	**CFG (πυρήνας)**	Η σύνταξη εντολών και blocks εκφράζεται φυσικά με BNF.
Java	**CFG (πυρήνας)**	Γραμματική για εκφράσεις, εντολές, κλάσεις· οι τύποι ελέγχονται χωριστά.
C	**Κοντά σε CFG**	Το `typedef` επηρεάζει την ερμηνεία του `T * p;`.
Python	**Κοντά σε CFG**	Χρησιμοποιεί PEG, (από v3.9) οι εσοχές γίνονται tokens `INDENT/DEDENT`.
JavaScript	**Κοντά σε CFG**	Επιπλέον κανόνες για tokens και αυτόματη εισαγωγή `;`.
C++	**Πιο μακριά**	Templates και αναζήτηση ονομάτων επηρεάζουν το parsing.
Perl	**Πιο μακριά**	Η αναγνώριση tokens εξαρτάται έντονα από τα συμφραζόμενα.

Προσοχή: «κοντά/μακριά» είναι *διδακτικές περιγραφές*, όχι αυστηρή ταξινόμηση. Η **σημασιολογική ανάλυση** ελέγχει δηλώσεις, εμβέλεια και τύπους.

$\{0^n 1^n \mid n \geq 0\}$

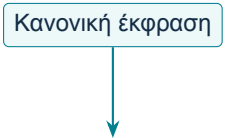
$L_1 = \{0^n 1^n \mid n \geq 0\}$,
$L_2 = \{\text{ορθά εμφωλευμένες παρενθέσεις}\}$.

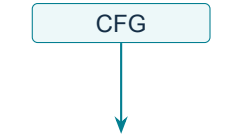
- Δεν υπάρχει DFA (pumping lemma)
- CFG: πώς; Αναδρομικά!



δύο επίπεδα αναδρομής: 0011

Η διαφορά CFG / κανονικής έκφρασης σε μία εικόνα





Π.χ. `00111` γίνεται δεκτή.

Μόνο `0011`, όχι `00111`.

Η γραμματική *συνδέει* την παραγωγή ενός `0` με την παραγωγή ενός `1`.

Οι τέσσερις συνιστώσες της CFG: παράδειγμα

Γραμματική $G = (V, \Sigma, R, S)$

$$V = \{S\}, \quad \Sigma = \{0, 1\}, \quad R = \{S \to 0S1,\ S \to \varepsilon\}, \quad S \text{ αρχικό.}$$

- S είναι *μεταβλητή*, όχι χαρακτήρας εισόδου.
- ε είναι η *κενή λέξη*, όχι τερματικό.
- $0, 1$ είναι τα τερματικά.

Άμεσος έλεγχος

$S \Rightarrow 0S1 \Rightarrow 01$.
$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011$.

Ένα βήμα, πολλά βήματα, μηδέν βήματα

$$\alpha A \beta \Rightarrow \alpha\gamma\beta \quad \text{αν } (A \to \gamma) \in R,$$
$$\alpha \Rightarrow^* \beta \quad \text{αν υπάρχουν } 0, 1, 2, \ldots \text{ βήματα.}$$

Γιατί μετρά το "μηδέν";

Για κάθε α, ισχύει $\alpha \Rightarrow^* \alpha$. Η σχέση $\Rightarrow^*$ είναι *ανακλαστική και μεταβατική*.

Τρεις διαφορετικές προτάσεις

$S \Rightarrow \texttt{0}S\texttt{1}$, $S \Rightarrow^* \texttt{0011}$, $S \Rightarrow^* S$.

Σύντομη ερώτηση: ποια γραμματική είναι CFG;

Γραμματική G_1

$S \to \mathtt{0}S\mathtt{1} \mid \varepsilon$
$\mathtt{0}S \to \mathtt{00}S\mathtt{1}$.

Γραμματική G_2

$S \to \mathtt{0}S \mid A$
$A \to A\mathtt{1} \mid \varepsilon$.

Ερώτηση: είναι έγκυρη η G_1, η G_2, και οι δύο ή καμία;

Απάντηση / συζήτηση: Μόνο η G_2. Σε CFG η αριστερή πλευρά *κάθε* κανόνα είναι ένα και μόνο μη τερματικό. Η μορφή $\mathtt{0}S \to \cdots$ δεν επιτρέπεται.

Παραγωγή

Ορισμός 2

Η z παράγεται (derives) από την x, συμβολικά $x \overset{*}{\rightsquigarrow} z$, αν μπορούμε να μετασχηματίσουμε την x στη z με πεπερασμένη ακολουθία εφαρμογών κανόνων· τυπικά, αν

- $x = z$, ή
- $x \rightsquigarrow y$ και $y \overset{*}{\rightsquigarrow} z$ για κάποια $y \in (\Sigma \cup \Gamma)^*$.

- Απλό «κυνήγι ορισμών» δίνει: αν $x \overset{*}{\rightsquigarrow} y$, τότε $wxz \overset{*}{\rightsquigarrow} wyz$ για κάθε $w, z \in (\Sigma \cup \Gamma)^*$.

Η γλώσσα μιας γραμματικής

Ορισμός 3

Η γλώσσα $L(w)$ μιας συμβολοσειράς $w \in (\Sigma \cup \Gamma)^*$ είναι το σύνολο των συμβολοσειρών του Σ^* που παράγονται από την w:

$$L(w) := \{ x \in \Sigma^* \mid w \overset{*}{\rightsquigarrow} x \}.$$

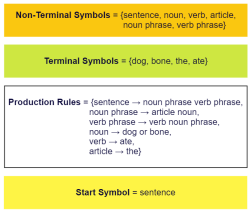
Η γλώσσα που παράγει η γραμματική G, συμβολικά $L(G)$, είναι η γλώσσα του αρχικού μη τερματικού της.

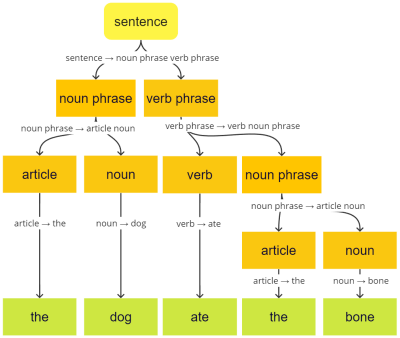
Ορισμός 4

Μια γλώσσα είναι χωρίς συμφραζόμενα (context-free) αν παράγεται από κάποια γραμματική χωρίς συμφραζόμενα.

Παράδειγμα: φυσική γλώσσα

- Wikipedia Formal Grammar





Αριστερότερη παραγωγή (Leftmost Derivation)

Έστω CFG $G = (V, \Sigma, R, S)$.

Τυπικός ορισμός

Γράφουμε

$$uAv \Rightarrow_{\text{lm}} u\gamma v$$

αν

$$A \to \gamma \in R,$$
$$u \in \Sigma^*,$$
$$v \in (V \cup \Sigma)^*.$$

Δηλαδή, σε κάθε βήμα αντικαθιστούμε την αριστερότερη μη τερματική μεταβλητή.
Η συνθήκη $u \in \Sigma^*$ εξασφαλίζει ότι δεν υπάρχει μεταβλητή αριστερά του A.

Παράδειγμα

Έστω η γραμματική $S \to AB$, $A \to \texttt{0}A \mid \texttt{0}$, $B \to \texttt{1}B \mid \texttt{1}$.
Αριστερότερη παραγωγή της λέξης 0011:

$$S \Rightarrow_{\text{lm}} AB \Rightarrow_{\text{lm}} 0AB \Rightarrow_{\text{lm}} 00B \Rightarrow_{\text{lm}} 001B \Rightarrow_{\text{lm}} 0011.$$

Δεξιότερη παραγωγή (Rightmost Derivation)

Έστω CFG $G = (V, \Sigma, R, S)$.

Τυπικός ορισμός

Γράφουμε

$$uAv \Rightarrow_{\text{rm}} u\gamma v$$

αν

$$A \to \gamma \in R,$$
$$u \in (V \cup \Sigma)^*,$$
$$v \in \Sigma^*.$$

Δηλαδή, σε κάθε βήμα αντικαθιστούμε τη δεξιότερη μη τερματική μεταβλητή.
Η συνθήκη $v \in \Sigma^*$ εξασφαλίζει ότι δεν υπάρχει μεταβλητή δεξιά του A.

Παράδειγμα

Έστω η ίδια γραμματική: $S \to AB$, $A \to 0A \mid 0$, $B \to 1B \mid 1$.
Δεξιότερη παραγωγή της λέξης 0011:

$$S \Rightarrow_{\text{rm}} AB \Rightarrow_{\text{rm}} A1B \Rightarrow_{\text{rm}} A11 \Rightarrow_{\text{rm}} 0A11 \Rightarrow_{\text{rm}} 0011.$$

Σύνοψη: Αριστερότερη και δεξιότερη παραγωγή

Γραμματική $S \to AB$, $A \to 0A \mid 0$, $B \to 1B \mid 1$.

Αριστερότερη παραγωγή της 0011

$$S \Rightarrow AB \Rightarrow 0AB \Rightarrow 00B \Rightarrow 001B \Rightarrow 0011.$$

Δεξιότερη παραγωγή της ίδιας λέξης

$$S \Rightarrow AB \Rightarrow A1B \Rightarrow A11 \Rightarrow 0A11 \Rightarrow 0011.$$

Γιατί μας ενδιαφέρει;

Οι δύο παραγωγές διαφέρουν μόνο στη **σειρά αντικατάστασης των μεταβλητών**.

Αντιστοιχούν όμως στο **ίδιο δέντρο ανάλυσης**.

Προσοχή: Μια γραμματική είναι αμφίσημη όταν κάποια λέξη έχει δύο διαφορετικά δέντρα ανάλυσης, όχι απλώς δύο διαφορετικές σειρές παραγωγής.

Δέντρα ανάλυσης

Ορισμός 5

Το δέντρο ανάλυσης (parse tree) μιας συμβολοσειράς $w \in L(G)$ είναι ένα διατεταγμένο δέντρο με ρίζα, όπου:

- Κάθε *φύλλο* έχει ετικέτα ένα τερματικό ή το ε. Η παράθεση των φύλλων από αριστερά προς τα δεξιά δίνει την w.
- Κάθε *εσωτερικός κόμβος* έχει ετικέτα ένα μη τερματικό· η ρίζα έχει ετικέτα το αρχικό μη τερματικό S.
- Για κάθε εσωτερικό κόμβο v υπάρχει κανόνας $A \to \omega$, όπου A η ετικέτα του v και τα σύμβολα της ω οι ετικέτες των παιδιών του v από αριστερά προς τα δεξιά.

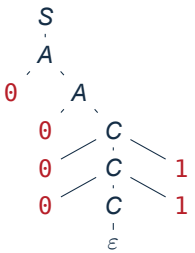
Παράδειγμα δέντρου ανάλυσης

Η γραμματική

$$S \to A \mid B$$
$$A \to \mathtt{0}A \mid \mathtt{0}C$$
$$B \to B\mathtt{1} \mid C\mathtt{1}$$
$$C \to \varepsilon \mid \mathtt{0}C\mathtt{1}$$

δίνει το διπλανό δέντρο για τη `000011`.

Το δέντρο κρατά τη *δομή*, όχι τη *σειρά* εφαρμογής των κανόνων (τι σημαίνει αυτό;).



HY280 · Θεωρία Υπολογισμού

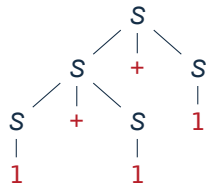
2. Κατασκευή γραμματικών και αποδείξεις

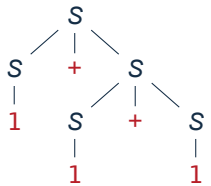
Αμφισημία

Ορισμός 6

Μια συμβολοσειρά w είναι αμφίσημη (ambiguous) ως προς μια γραμματική αν έχει περισσότερα από ένα δέντρα ανάλυσης. Μια γραμματική G είναι αμφίσημη αν κάποια συμβολοσειρά είναι αμφίσημη ως προς την G.

Η γραμματική $S \to 1 \mid S{+}S$ είναι αμφίσημη: η $1+1+1$ έχει δύο δέντρα (και η $1+1+1+1$ έχει πέντε).





Εγγενής αμφισημία

Ορισμός 7

Μια γλώσσα L είναι εγγενώς αμφίσημη (inherently ambiguous) αν *κάθε* γραμματική χωρίς συμφραζόμενα που παράγει την L είναι αμφίσημη.

Παράδειγμα 8

Η γλώσσα της γραμματικής $S \to 1 \mid S{+}S$ (η κανονική γλώσσα $(1{+})^*1$) *δεν* είναι εγγενώς αμφίσημη, γιατί η μη αμφίσημη γραμματική

$$S \to 1 \mid 1{+}S$$

παράγει την ίδια γλώσσα.

- Η αμφισημία είναι ιδιότητα της *γραμματικής*· η εγγενής αμφισημία είναι ιδιότητα της *γλώσσας*.

Ποια γλώσσα παράγει η παρακάτω γραμματική;

$$C \to \varepsilon \mid \mathtt{0}C\mathtt{1} \qquad A \to \mathtt{0}A \mid \mathtt{0}C \qquad B \to B\mathtt{1} \mid C\mathtt{1}$$

- Οι κανόνες του C δεν αναφέρονται σε άλλο μη τερματικό: ξεκινάμε από το $L(C)$.
- Πειραματιζόμενοι με το $C \to \varepsilon \mid \mathtt{0}C\mathtt{1}$ μαντεύουμε $L(C) = \{\varepsilon, \mathtt{01}, \mathtt{0011}, \ldots\} = \{\mathtt{0}^n\mathtt{1}^n \mid n \geq 0\}$.

Παράδειγμα 9

Παραγωγή της `00001111`:

$$C \rightsquigarrow \mathtt{0}C\mathtt{1} \rightsquigarrow \mathtt{00}C\mathtt{11} \rightsquigarrow \mathtt{000}C\mathtt{111} \rightsquigarrow \mathtt{0000}C\mathtt{1111} \rightsquigarrow \mathtt{0000}\varepsilon\mathtt{1111} = \mathtt{00001111}$$

Λήμμα 1: $C \overset{*}{\rightsquigarrow} 0^n1^n$

Η απόδειξη ότι $L(C) = \{0^n1^n \mid n \geq 0\}$ έχει δύο στάδια ($X \subseteq Y$ και $Y \subseteq X$), όπως πάντα για ισότητα συνόλων.

Λήμμα 10

$C \overset{*}{\rightsquigarrow} 0^n1^n$ για κάθε μη αρνητικό ακέραιο n.

Απόδειξη.

Έστω αυθαίρετο $n \geq 0$. Υποθέτουμε ότι $C \overset{*}{\rightsquigarrow} 0^k1^k$ για κάθε $k < n$.

- Αν $n = 0$, τότε $0^n1^n = \varepsilon$. Ο κανόνας $C \to \varepsilon$ δίνει $C \rightsquigarrow \varepsilon$, άρα $C \overset{*}{\rightsquigarrow} \varepsilon$.
- Αν $n > 0$, η επαγωγική υπόθεση δίνει $C \overset{*}{\rightsquigarrow} 0^{n-1}1^{n-1}$. Άρα ο κανόνας $C \to 0C1$ δίνει $C \rightsquigarrow 0C1 \overset{*}{\rightsquigarrow} 0(0^{n-1}1^{n-1})1 = 0^n1^n$.

Και στις δύο περιπτώσεις $C \overset{*}{\rightsquigarrow} 0^n1^n$. □

Λήμμα 2: Κάθε $w \in L(C)$ είναι της μορφής 0^n1^n

Λήμμα 11

Για κάθε $w \in L(C)$ ισχύει $w = 0^n1^n$ για κάποιο $n \geq 0$.

Απόδειξη.

Έστω, προς άτοπο, ότι υπάρχει λέξη $w \in L(C)$ που δεν είναι της μορφής 0^n1^n.
Επιλέγουμε μια τέτοια λέξη w με ελάχιστο μήκος.

Η γραμματική είναι

$$C \to 0C1 \mid \varepsilon.$$

1. Δεν μπορεί να ισχύει $w = \varepsilon$, αφού $\varepsilon = 0^0 1^0$.
2. Επομένως, αναγκαστικά
$$w = 0x1 \quad \text{για κάποιο } x \in L(C).$$
3. Όμως $|x| = |w| - 2 < |w|$. Από την **ελαχιστικότητα** του w, το x πρέπει να έχει τη μορφή $x = 0^k1^k$.
4. Άρα
$$w = 0^{k+1}1^{k+1},$$
άτοπο, αφού το w ήταν αντιπαράδειγμα.

Άσκηση: Προσοχή με τα ε

Η επαγωγή στο μήκος της συμβολοσειράς δεν δουλεύει για κάθε γραμματική. Θεωρήστε την αμφίσημη γραμματική

$$S \to \varepsilon \mid SS \mid \mathtt{0}S\mathtt{1} \mid \mathtt{1}S\mathtt{0}.$$

- $L(S)$ = όλες οι δυαδικές συμβολοσειρές με ίσο πλήθος $\mathtt{0}$ και $\mathtt{1}$.
- Υπάρχουν αυθαίρετα μακριές (αλλά πεπερασμένες!) παραγωγές

  $$S \rightsquigarrow SS \rightsquigarrow S \rightsquigarrow SS \rightsquigarrow S \rightsquigarrow SS \rightsquigarrow SS \rightsquigarrow \cdots \rightsquigarrow w,$$

  που εναλλάσσουν τους κανόνες $S \to SS$ και $S \to \varepsilon$.
- Ακόμη κι αν ξέρουμε ότι $w = xy$ με $x, y \in L(S)$, δεν εγγυόμαστε $|x| < |w|$ και $|y| < |w|$: δεν εφαρμόζεται η επαγωγική υπόθεση.

Τεχνική: παραγωγή ελάχιστου μήκους

Τροποποιημένο πρότυπο επαγωγής

Θεωρούμε μια παραγωγή ελάχιστου μήκους της w και κάνουμε ανάλυση περιπτώσεων ως προς τον πρώτο κανόνα αυτής της παραγωγής.

Λήμμα 12

Για κάθε $w \in L(S)$ *ισχύει* $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$.

Απόδειξη (αρχή).

Έστω $w \in L(S)$ και μια παραγωγή ελάχιστου μήκους της w. Υποθέτουμε ότι κάθε συντομότερη $x \in L(S)$ έχει $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$. Τέσσερις περιπτώσεις, ανάλογα με τον πρώτο κανόνα της παραγωγής.

Τεχνική: παραγωγή ελάχιστου μήκους: 4 περιπτώσεις

Απόδειξη (συνέχεια).

- $S \to \varepsilon$: τότε $w = \varepsilon$ και $\#(\mathtt{0}, w) = \#(\mathtt{1}, w) = 0$.
- $S \to SS$: τότε $w = xy$ με $x, y \in L(S)$. Τα x, y είναι μη κενά, αλλιώς θα μπορούσαμε να συντομεύσουμε την παραγωγή. Άρα είναι συντομότερα από w και η επαγωγική υπόθεση δίνει $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + \#(\mathtt{0}, y) = \#(\mathtt{1}, x) + \#(\mathtt{1}, y) = \#(\mathtt{1}, w)$.
- $S \to \mathtt{0}S\mathtt{1}$: τότε $w = \mathtt{0}x\mathtt{1}$, $x \in L(S)$, και $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + 1 = \#(\mathtt{1}, x) + 1 = \#(\mathtt{1}, w)$.
- $S \to \mathtt{1}S\mathtt{0}$: συμμετρικά.

Σε όλες τις περιπτώσεις $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$. □

Αμοιβαία επαγωγή: γιατί τη χρειαζόμαστε;

Έστω η γραμματική

$$S \to \mathtt{0}A\mathtt{1} \mid \varepsilon, \qquad A \to \mathtt{1}S\mathtt{0} \mid \varepsilon.$$

Παρατήρηση

Για να καταλάβουμε τι παράγει το S, πρέπει να ξέρουμε τι παράγει το A.
Αντίστοιχα, για να καταλάβουμε τι παράγει το A, πρέπει να ξέρουμε τι παράγει το S.

Λήμμα 13

Ισχύουν ταυτόχρονα:

$$\boxed{L(S) = (\mathtt{01})^*, \qquad L(A) = (\mathtt{10})^*.}$$

Ιδέα της απόδειξης.

Θα αποδείξουμε και τις δύο ισότητες, εξετάζοντας χωριστά τις δύο εγκλείσεις:

$$(\mathtt{01})^* \subseteq L(S), \quad (\mathtt{10})^* \subseteq L(A),$$
$$L(S) \subseteq (\mathtt{01})^*, \quad L(A) \subseteq (\mathtt{10})^*.$$

Κρίσιμο: Σε κάθε βήμα συλλογιζόμαστε ταυτόχρονα για τις δύο μεταβλητές. □

Αμοιβαία επαγωγή: πρώτη έγκλειση

Στόχος

Να αποδείξουμε με επαγωγή στο $n \geq 0$ ότι

$$P(n): \quad (\texttt{01})^n \in L(S) \text{ και } (\texttt{10})^n \in L(A).$$

Απόδειξη.

Βάση ($n = 0$). Από τους κανόνες $S \to \varepsilon$ και $A \to \varepsilon$: $(\texttt{01})^0 = \varepsilon \in L(S)$, $\quad (\texttt{10})^0 = \varepsilon \in L(A)$.
Επαγωγική υπόθεση. Έστω ότι ισχύει $P(n)$, δηλαδή

$$S \overset{*}{\rightsquigarrow} (\texttt{01})^n, \qquad A \overset{*}{\rightsquigarrow} (\texttt{10})^n.$$

Επαγωγικό βήμα ($n \to n+1$).
Από την παραγωγή του A προκύπτει: $S \rightsquigarrow \texttt{0}A\texttt{1} \overset{*}{\rightsquigarrow} \texttt{0}(\texttt{10})^n\texttt{1} = (\texttt{01})^{n+1}$.
Αντίστοιχα, από την παραγωγή του S: $A \rightsquigarrow \texttt{1}S\texttt{0} \overset{*}{\rightsquigarrow} \texttt{1}(\texttt{01})^n\texttt{0} = (\texttt{10})^{n+1}$.
Άρα ισχύει $P(n+1)$ και ολοκληρώνεται η επαγωγή. □

Αμοιβαία επαγωγή: δεύτερη έγκλειση (1/2)

Στόχος

Να αποδείξουμε ότι

$$L(S) \subseteq (\texttt{01})^*, \qquad L(A) \subseteq (\texttt{10})^*.$$

Απόδειξη.

Υποθέτουμε, προς άτοπο, ότι τουλάχιστον μία από τις δύο εγκλείσεις είναι ψευδής.
Τότε υπάρχει είτε

$$w \in L(S) \setminus (\texttt{01})^*$$

είτε

$$w \in L(A) \setminus (\texttt{10})^*.$$

Επιλέγουμε ένα τέτοιο αντιπαράδειγμα w με ελάχιστο μήκος, ανάμεσα στα αντιπαραδείγματα **και των δύο γλωσσών**.

Επειδή ε ανήκει και στα $(\texttt{01})^*$ και $(\texttt{10})^*$, αναγκαστικά $w \neq \varepsilon$.

Θα εξετάσουμε σε ποια από τις δύο γλώσσες ανήκει το w. □

Αμοιβαία επαγωγή: δεύτερη έγκλειση (2/2)

Συνέχεια της απόδειξης.

Περίπτωση 1: $w \in L(S) \setminus (\texttt{01})^*$.
Αφού $w \neq \varepsilon$, από τον μοναδικό μη κενό κανόνα του S έχουμε

$$w = \texttt{0}x\texttt{1}, \qquad x \in L(A).$$

Όμως $|x| = |w| - 2 < |w|$. Από την ελαχιστικότητα του w, το x δεν είναι αντιπαράδειγμα. Άρα

$$x = (\texttt{10})^k \quad \text{για κάποιο } k \geq 0.$$

Επομένως

$$w = \texttt{0}(\texttt{10})^k\texttt{1} = (\texttt{01})^{k+1},$$

άτοπο.

Περίπτωση 2: $w \in L(A) \setminus (\texttt{10})^*$.
Ομοίως, $w = \texttt{1}x\texttt{0}$ με $x \in L(S)$ και $|x| < |w|$.
Από την ελαχιστικότητα, $x = (\texttt{01})^k$, οπότε

$$w = \texttt{1}(\texttt{01})^k\texttt{0} = (\texttt{10})^{k+1},$$

Πώς σχεδιάζουμε (CFG) γραμματικές;

- Θα δούμε μερικά παραδείγματα του πώς σχεδιάζουμε μια CFG γραμματική.
- Οι αποδείξεις πληρότητας και ορθότητας είναι ασκήσεις και μέρος του φροντιστηρίου της Παρασκευής

Για να δείξουμε ότι $L(G) = L$

1. **Πληρότητα:** για κάθε $w \in L$, κατασκευάζουμε παραγωγή $S \Rightarrow^* w$.
2. **Ορθότητα:** για κάθε παραγωγή $S \Rightarrow^* w$ με $w \in \Sigma^*$, αποδεικνύουμε ότι $w \in L$.

Δύο διαφορετικές επαγωγές

Πληρότητα: επαγωγή στο μέγεθος/δομή του w.
Ορθότητα: επαγωγή στο *δέντρο παραγωγής* ή στο πλήθος βημάτων.

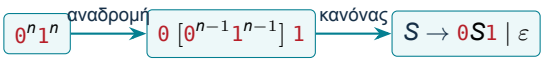
Προσοχή: η δοκιμή μερικών παραδειγμάτων δεν είναι επίσημη απόδειξη ισότητας δύο γλωσσών,

Άσκηση: σχεδιάζουμε γραμματική για 0^n1^n

Πώς προκύπτει ο κανόνας;

$$0^n1^n = 0\,(0^{n-1}1^{n-1})\,1 \qquad (n > 0).$$

Από την αναδρομική ταυτότητα παίρνουμε $S \to 0S1$ και από το $n = 0$ παίρνουμε $S \to \varepsilon$.



Παλίνδρομα: η βασική ιδέα

Τι είναι ένα παλίνδρομο;

Μια λέξη w είναι παλίνδρομη αν διαβάζεται το ίδιο από αριστερά προς τα δεξιά και αντίστροφα:

$$w = w^R.$$

Για παράδειγμα:

$$\texttt{0110}, \quad \texttt{01010}, \quad \texttt{0}, \quad \varepsilon.$$

Πώς κατασκευάζουμε ένα παλίνδρομο;

Αν η λέξη w είναι παλίνδρομη, τότε και οι λέξεις

$$\boxed{\texttt{0}w\texttt{0}} \qquad \text{και} \qquad \boxed{\texttt{1}w\texttt{1}}$$

είναι παλίνδρομες.

Δηλαδή, προσθέτουμε **το ίδιο σύμβολο στην αρχή και στο τέλος**.

Αναδρομική γραμματική

$$\boxed{S \to \texttt{0}S\texttt{0} \mid \texttt{1}S\texttt{1} \mid \varepsilon \mid \texttt{0} \mid \texttt{1}}$$

Παλίνδρομα: άρτιο και περιττό μήκος

$$S \to 0S0 \mid 1S1 \mid \varepsilon \mid 0 \mid 1.$$

Γιατί υπάρχουν τρεις κανόνες τερματισμού;

Κάθε αναδρομικό βήμα

$$S \to 0S0 \quad \text{ή} \quad S \to 1S1$$

προσθέτει δύο σύμβολα.
Επομένως, η παρατηρούμενη διαφορά άρτιου και περιττού μήκους καθορίζεται από τον **τελικό κανόνα**.

Άρτιο μήκος

$$S \to \varepsilon$$

Ξεκινάμε από μήκος 0.
Μετά από k αναδρομικά βήματα:

$$|w| = 2k.$$

Περιττό μήκος

$$S \to 0 \mid 1$$

Ξεκινάμε από μήκος 1.
Μετά από k αναδρομικά βήματα:

$$|w| = 2k + 1.$$

Παλίνδρομα: δύο αναλυτικές παραγωγές

$$S \to \texttt{0}S\texttt{0} \mid \texttt{1}S\texttt{1} \mid \varepsilon \mid \texttt{0} \mid \texttt{1}.$$

Άρτιο παράδειγμα: `0110`

$$\begin{aligned} S &\Rightarrow \texttt{0}S\texttt{0} \\ &\Rightarrow \texttt{01}S\texttt{10} \\ &\Rightarrow \texttt{01}\varepsilon\texttt{10} \\ &= \boxed{\texttt{0110}}. \end{aligned}$$

Τερματίζουμε με $S \to \varepsilon$.

Περιττό παράδειγμα: `01010`

$$\begin{aligned} S &\Rightarrow \texttt{0}S\texttt{0} \\ &\Rightarrow \texttt{01}S\texttt{10} \\ &\Rightarrow \texttt{01010} \\ &= \boxed{\texttt{01010}}. \end{aligned}$$

Τερματίζουμε με $S \to \texttt{0}$.

Δύο εξαρτήσεις: μία γλώσσα χωρίς συμφραζόμενα

Ορισμός της γλώσσας

Έστω $\Sigma = \{0, 1, 2, 3\}$ και $\boxed{L = \{0^n 1^m 2^m 3^n \mid n, m \geq 0\}.}$

Μια λέξη ανήκει στην L αν:

- τα 0 είναι όσα και τα 3,
- τα 1 είναι όσα και τα 2,
- τα σύμβολα εμφανίζονται με τη σειρά $0, 1, 2, 3$.

Παραδείγματα

$$
\begin{aligned}
&0011223 3 \in L && (n = 2,\ m = 2),\\
&01123 \notin L && (\#1 \neq \#2),\\
&001233 \notin L && (n = 1,\ m = 2).
\end{aligned}
$$

Η κρίσιμη παρατήρηση

Η εξάρτηση $0^n \cdots 3^n$ **περιβάλλει** την εξάρτηση $1^m 2^m$. Οι δύο εξαρτήσεις είναι εμφωλευμένες.

Γιατί αρκεί μία στοίβα;

Θέλουμε να αναγνωρίσουμε

$$L = \{0^n 1^m 2^m 3^n \mid n, m \geq 0\}.$$

Ιδέα ενός PDA

1. Για κάθε 0, ωθούμε ένα σύμβολο X στη στοίβα.
2. Για κάθε 1, ωθούμε ένα σύμβολο Y.
3. Για κάθε 2, αφαιρούμε ένα Y.
4. Για κάθε 3, αφαιρούμε ένα X.

Παράδειγμα: 00112233

Είσοδος	ε	0	0	1	1	2	2	3	3
Στοίβα	ε	X	XX	XXY	XXYY	XXY	XX	X	ε

Η κορυφή της στοίβας εμφανίζεται δεξιά. Το PDA ελέγχει και τη σωστή σειρά των συμβόλων.

Γιατί λειτουργεί;

Η στοίβα είναι LIFO: τα τελευταία σύμβολα που εισάγονται είναι τα πρώτα που αφαιρούνται. Άρα ελέγχουμε **πρώτα την εσωτερική** και **έπειτα την εξωτερική** εξάρτηση.

Κατασκευή CFG για εμφωλευμένες εξαρτήσεις

Η γραμματική

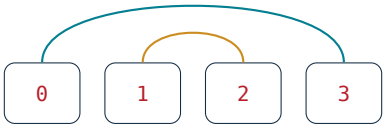
$$S \to 0S3 \mid T, \qquad T \to 1T2 \mid \varepsilon.$$

- Ο S παράγει τα εξωτερικά ζεύγη $0 \cdots 3$.
- Ο T παράγει τα εσωτερικά ζεύγη $1 \cdots 2$.
- Ο κανόνας $S \to T$ σηματοδοτεί τη μετάβαση από την εξωτερική στην εσωτερική εξάρτηση.

Παραγωγή της 00112233

$$\begin{aligned} S &\Rightarrow 0S3 \\ &\Rightarrow 00S33 \\ &\Rightarrow 00T33 \\ &\Rightarrow 001T233 \\ &\Rightarrow 0011T2233 \\ &\Rightarrow 00112233. \end{aligned}$$

Εμφωλευμένες έναντι διασταυρούμενων εξαρτήσεων



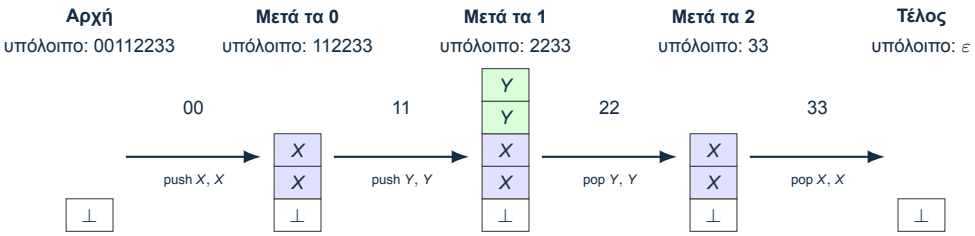
Το σχήμα είναι διαισθητικό: η μη-CF ιδιότητα της δεύτερης γλώσσας χρειάζεται απόδειξη.

Μία στοίβα αρκεί: $0^n1^m2^m3^n$

$$L = \{0^n1^m2^m3^n \mid n, m \geq 0\}, \qquad \text{παράδειγμα: } \underbrace{00}_{n=2}\,\underbrace{11}_{m=2}\,\underbrace{22}_{m=2}\,\underbrace{33}_{n=2}.$$

Ιδέα: η στοίβα είναι LIFO

0: push X 1: push Y 2: pop Y 3: pop X.



Κορυφή: το πάνω στοιχείο κάθε στοίβας

Γιατί λειτουργεί;

Τα Y (εσωτερική εξάρτηση 1^m2^m) αφαιρούνται **πριν** από τα X (εξωτερική εξάρτηση 0^n3^n). Η σειρά είναι ακριβώς LIFO.

HY280 · Θεωρία Υπολογισμού

3. Chomsky Normal Form και αναγνώριση

Noam Chomsky και Ιεραρχία Chomsky

- Noam Chomsky
- Ιεραρχία Chomsky



Noam Chomsky: από τη γλώσσα στον υπολογισμό

Μια θεμελιώδης ιδέα

Ο Noam Chomsky (1928–) άλλαξε ριζικά τη μελέτη της ανθρώπινης γλώσσας, αντιμετωπίζοντάς την ως σύστημα κανόνων που παράγει προτάσεις.

Η ιεραρχία Chomsky (1956) αποκάλυψε μια βαθιά σύνδεση:

$$\boxed{\text{Γραμματικές} \quad \longleftrightarrow \quad \text{Υπολογιστικά μοντέλα}}$$

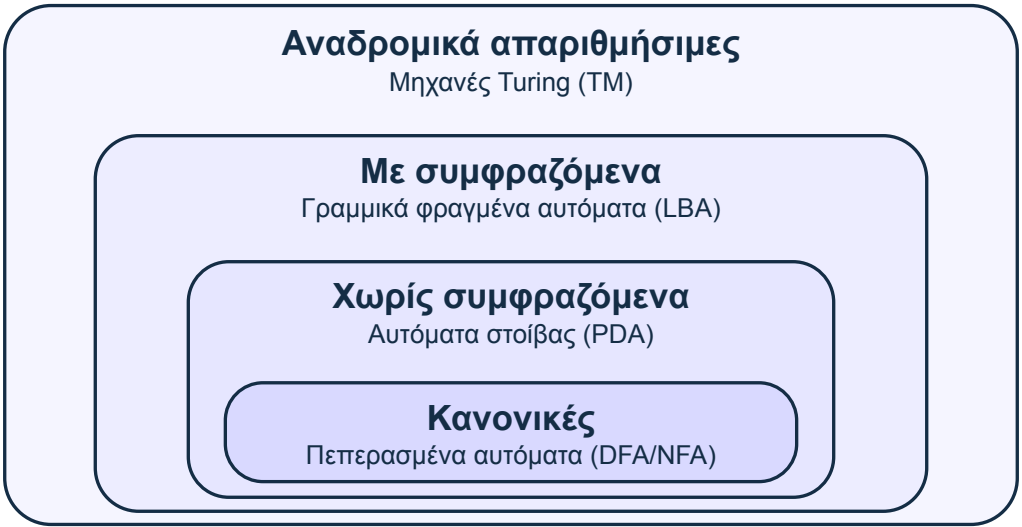
Οι περιορισμοί στους κανόνες παραγωγής καθορίζουν την εκφραστική ισχύ της γραμματικής και συνδέονται με διαφορετικούς τύπους αυτομάτων.

Ιεραρχία Chomsky: τέσσερις τύποι γραμματικών

Τύπος	Γλώσσες	Αυτόματο αναγνώρισης	Κανόνες παραγωγής	Παράδειγμα
3	Κανονικές	Πεπερασμένο (DFA / NFA)	$A \to aB \mid a$ (δεξιόγραμμη) ή $A \to Ba \mid a$ (αριστερόγραμμη)	$\{a^n : n > 0\}$
2	Χωρίς συμφραζόμενα (CFL)	Μη ντετερμινιστικό αυτόματο στοίβας (NPDA)	$A \to \alpha$	$\{a^n b^n : n > 0\}$
1	Με συμφραζόμενα (CSL)	Γραμμικά φραγμένη Μηχανή Turing (LBA)	$\alpha A \beta \to \alpha \gamma \beta$, $\gamma \neq \varepsilon$	$\{a^n b^n c^n : n > 0\}$
0	Αναδρομικά απαριθμήσιμες (RE)	Μηχανή Turing	$\gamma \to \alpha$ (γ μη κενό)	$\{\langle M \rangle :$ M τερματίζει σε είσοδο $\varepsilon\}$

$A, B \in V$, $a \in \Sigma$, $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ (με τους αντίστοιχους περιορισμούς).

Ιεραρχία Chomsky και μηχανές αναγνώρισης



Από ένα DFA σε γραμματική: η γενική ιδέα

Θεώρημα

Κάθε κανονική γλώσσα είναι και γλώσσα χωρίς συμφραζόμενα:

$$\boxed{\text{Regular} \subseteq \text{CFL}.}$$

Θα το δείξουμε **κατασκευαστικά**: από κάθε DFA φτιάχνουμε μια δεξιόγραμμη γραμματική που παράγει ακριβώς την ίδια γλώσσα.

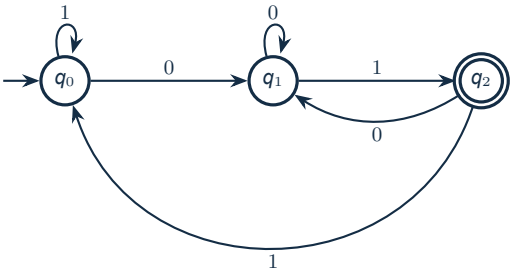
Έστω DFA $D = (Q, \Sigma, \delta, q_0, F)$. Κατασκευάζουμε $G = (V, \Sigma, R, A_{q_0})$ με:

1. **Μία μεταβλητή A_q για κάθε κατάσταση $q \in Q$.**
2. Για κάθε μετάβαση $\delta(q, a) = r$, προσθέτουμε
$$\boxed{A_q \to aA_r.}$$
3. Για κάθε τελική κατάσταση $f \in F$, προσθέτουμε
$$\boxed{A_f \to \varepsilon.}$$

Διαίσθηση: το A_q παράγει ακριβώς τις συνέχειες της εισόδου που οδηγούν από την κατάσταση q σε αποδοχή.

Παράδειγμα: δυαδικές λέξεις που τελειώνουν σε 01

$L = \{w \in \{0,1\}^* : \text{το } w \text{ τελειώνει σε } 01\}$.



Αποδεκτές λέξεις

01, 001, 101, 1101, 00001.

Μη αποδεκτές λέξεις

ε, 0, 11, 1100, 010.

Ερώτηση: Ποια κατάσταση είναι τελική; Γιατί το q_2 έχει εξερχόμενες μεταβάσεις;

Μετατρέπουμε τις μεταβάσεις σε κανόνες

Κατάσταση	$\delta(q, 0)$	$\delta(q, 1)$	Κανόνες γραμματικής
q_0	q_1	q_0	$A_0 \to 0A_1 \mid 1A_0$
q_1	q_1	q_2	$A_1 \to 0A_1 \mid 1A_2$
q_2	q_1	q_0	$A_2 \to 0A_1 \mid 1A_0 \mid \varepsilon$

Η γραμματική G

$$\begin{aligned} A_0 &\to 0A_1 \mid 1A_0 \\ A_1 &\to 0A_1 \mid 1A_2 \\ A_2 &\to 0A_1 \mid 1A_0 \mid \varepsilon \end{aligned}$$

Αρχική μεταβλητή: A_0. Αλφάβητο: $\Sigma = \{0, 1\}$.

Γιατί υπάρχει το $A_2 \to \varepsilon$; Επειδή το q_2 είναι τελική κατάσταση: εκεί μπορούμε να *σταματήσουμε*.

Η ίδια λέξη: εκτέλεση DFA και παραγωγή CFG

Παράδειγμα: $w = 1101$

Εκτέλεση του DFA:

$$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2.$$

Παραγωγή της γραμματικής:

$$\begin{aligned} A_0 &\Rightarrow 1A_0 \\ &\Rightarrow 11A_0 \\ &\Rightarrow 110A_1 \\ &\Rightarrow 1101A_2 \\ &\Rightarrow \boxed{1101}. \end{aligned}$$

Κάθε βήμα έχει το αντίστοιχό του

Η μετάβαση $q_i \xrightarrow{a} q_j$ αντιστοιχεί στην παραγωγή $A_i \to aA_j$. Το τελικό $A_2 \to \varepsilon$ αφαιρεί τη μεταβλητή.

Δοκιμάστε το 1100**:** η εκτέλεση τελειώνει στο q_1, άρα δεν υπάρχει παραγωγή $A_0 \Rightarrow^* 1100$.

Γιατί η κατασκευή είναι σωστή;

Κεντρική ιδιότητα

Για κάθε κατάσταση q και κάθε $w \in \Sigma^*$:

$$\boxed{A_q \Rightarrow^* w \quad \Longleftrightarrow \quad \delta^*(q, w) \in F.}$$

Απόδειξη με επαγωγή στο $|w|$:

1. **Βάση $w = \varepsilon$:** $A_q \Rightarrow^* \varepsilon$ αν και μόνο αν υπάρχει ο κανόνας $A_q \to \varepsilon$, δηλαδή αν $q \in F$.
2. **Βήμα $w = av$:** η πρώτη παραγωγή πρέπει να είναι $A_q \to aA_r$, όπου $r = \delta(q, a)$. Έπειτα

$$A_r \Rightarrow^* v \quad \Longleftrightarrow \quad \delta^*(r, v) \in F$$

από την επαγωγική υπόθεση.

Συμπέρασμα

$A_{q_0} \Rightarrow^* w$ αν και μόνο αν το DFA αποδέχεται το w. Άρα $\boxed{L(G) = L(D)}$.

Άσκηση: DFA για άρτιο πλήθος από 1

Το DFA

	0	1
q_e	q_e	q_o
q_o	q_o	q_e

q_e: αρχική και τελική κατάσταση.

Ερωτήματα

1. Ποιες είναι οι μεταβλητές;
2. Ποιοι κανόνες προκύπτουν;
3. Παράγεται η λέξη 110;
4. Παράγεται η λέξη 1011;

Άσκηση: DFA για άρτιο πλήθος από 1

Το DFA

	0	1
q_e	q_e	q_o
q_o	q_o	q_e

q_e: αρχική και τελική κατάσταση.

Ερωτήματα

1. Ποιες είναι οι μεταβλητές;
2. Ποιοι κανόνες προκύπτουν;
3. Παράγεται η λέξη 110;
4. Παράγεται η λέξη 1011;

Λύση

$$\boxed{\begin{aligned} A_e &\to 0A_e \mid 1A_o \mid \varepsilon \\ A_o &\to 0A_o \mid 1A_e \end{aligned}}$$

$$A_e \Rightarrow 1A_o \Rightarrow 11A_e \Rightarrow 110A_e \Rightarrow 110.$$

Το 1011 έχει τρία 1 και δεν παράγεται.

Δεύτερη απόδειξη: κάθε κανονική γλώσσα είναι CFL

Θεώρημα

$$\boxed{\text{Regular} \subseteq \text{CFL}.}$$

Κάθε κανονική γλώσσα παράγεται από κάποια γραμματική χωρίς συμφραζόμενα.

Πρώτη κατασκευή

$$\text{DFA} \longrightarrow \text{CFG}$$

Μία μεταβλητή ανά κατάσταση· μία παραγωγή ανά μετάβαση.

Νέα κατασκευή

$$\text{Regex} \longrightarrow \text{CFG}$$

Δομική επαγωγή στον σχηματισμό της κανονικής έκφρασης.

Αρκεί να αποδείξουμε:

$$\forall r \, \exists \, G_r \text{ CFG τέτοια ώστε } \boxed{L(G_r) = L(r)}.$$

Κάθε κανονική γλώσσα έχει κάποια κανονική έκφραση r.

Βάση επαγωγής: οι στοιχειώδεις εκφράσεις

Έστω αλφάβητο Σ. Οι απλούστερες κανονικές εκφράσεις είναι:

Έκφραση r	**Γλώσσα** $L(r)$	**CFG με αρχικό** S
$a \in \Sigma$	$\{a\}$	$S \to a$
ε	$\{\varepsilon\}$	$S \to \varepsilon$
$\emptyset$	$\emptyset$	$S \to S$

Γιατί ο κανόνας $S \to S$ παράγει την κενή γλώσσα;

Από το S μπορούμε να παράγουμε μόνο ξανά το S. **Δεν φτάνουμε ποτέ σε λέξη από τερματικά**, άρα $L(G) = \emptyset$.

Προσοχή: $\emptyset \neq \{\varepsilon\}$. Η πρώτη γλώσσα δεν έχει λέξεις· η δεύτερη περιέχει *μία* λέξη, μήκους 0.

Επαγωγική υπόθεση: δύο μικρότερες εκφράσεις

Δομική επαγωγή στο r

Κάθε μη στοιχειώδης έκφραση έχει μία από τις μορφές

$$\boxed{r = (st)}, \qquad \boxed{r = (s + t)}, \qquad \boxed{r = (s^*)},$$

με μικρότερες υποεκφράσεις s, t.

Επαγωγική υπόθεση: έχουμε ήδη κατασκευάσει

$$G_s = (V_s, \Sigma, R_s, S_s), \quad L(G_s) = L(s),$$
$$G_t = (V_t, \Sigma, R_t, S_t), \quad L(G_t) = L(t).$$

Πριν τις συνδυάσουμε: μετονομάζουμε μεταβλητές

Απαιτούμε $V_s \cap V_t = \emptyset$ και επιλέγουμε νέο αρχικό σύμβολο $S \notin V_s \cup V_t$.

Έτσι δεν συγχέονται οι κανόνες των δύο υπογραμματικών.

Επαγωγικό βήμα I: συνένωση $r = (st)$

Κατασκευή της νέας γραμματικής

Κρατάμε όλους τους κανόνες των G_s, G_t και προσθέτουμε

$$\boxed{S \to S_s S_t.}$$

Δηλαδή $R = R_s \cup R_t \cup \{S \to S_s S_t\}$.

Γιατί είναι σωστό;

$$S \Rightarrow S_s S_t \Rightarrow^* xy \quad \Longleftrightarrow \quad \begin{cases} x \in L(s), \\ y \in L(t). \end{cases}$$

Άρα

$$\boxed{L(G_r) = L(s)L(t) = L(st).}$$

Μικρό παράδειγμα: $r = 01$

$$A \to 0, \quad B \to 1, \quad S \to AB, \qquad S \Rightarrow AB \Rightarrow 0B \Rightarrow 01.$$

Επαγωγικό βήμα II: ένωση $r = (s + t)$

Κατασκευή της νέας γραμματικής

Κρατάμε όλους τους κανόνες των G_s, G_t και προσθέτουμε

$$\boxed{S \to S_s \mid S_t.}$$

Δηλαδή $R = R_s \cup R_t \cup \{S \to S_s,\ S \to S_t\}$.

Γιατί είναι σωστό;

$$S \Rightarrow S_s \Rightarrow^* w \quad \Longleftrightarrow \quad w \in L(s),$$
$$S \Rightarrow S_t \Rightarrow^* w \quad \Longleftrightarrow \quad w \in L(t).$$

Επομένως

$$\boxed{L(G_r) = L(s) \cup L(t) = L(s + t).}$$

Μικρό παράδειγμα: $r = (0 + 1)$

$$A \to 0, \quad B \to 1, \quad S \to A \mid B.$$

Επαγωγικό βήμα III: αστέρι Kleene $r = (s^*)$

Κατασκευή της νέας γραμματικής

Κρατάμε τους κανόνες του G_s και προσθέτουμε νέο αρχικό S:

$$\boxed{S \to \varepsilon \mid S_s S.}$$

Μηδέν επαναλήψεις:

$$S \Rightarrow \varepsilon.$$

Η κενή λέξη ανήκει πάντοτε στο $L(s^*)$.

$k \geq 1$ επαναλήψεις:

$$S \Rightarrow S_s S \Rightarrow^* w_1 S \Rightarrow^* \cdots \Rightarrow^* w_1 \cdots w_k,$$

όπου κάθε $w_i \in L(s)$.

$$\boxed{L(G_r) = \bigcup_{k \geq 0} L(s)^k = L(s^*).}$$

Μικρό παράδειγμα: $r = 0^*$

$$A \to 0, \quad S \to \varepsilon \mid AS, \qquad S \Rightarrow AS \Rightarrow 0S \Rightarrow 0AS \Rightarrow 00S \Rightarrow 00.$$

Ολοκλήρωση της επαγωγικής απόδειξης

Μορφή r	Νέος κανόνας αρχικού S	Γλώσσα
a	$S \to a$	$\{a\}$
ε	$S \to \varepsilon$	$\{\varepsilon\}$
$\emptyset$	$S \to S$	$\emptyset$
st	$S \to S_s S_t$	$L(s)L(t)$
$s + t$	$S \to S_s \mid S_t$	$L(s) \cup L(t)$
s^*	$S \to \varepsilon \mid S_s S$	$L(s)^*$

Συμπέρασμα

Οι τρεις βασικές εκφράσεις έχουν ισοδύναμες CFG. Κάθε σύνθετη κανονική έκφραση προκύπτει από τις τρεις πράξεις, για τις οποίες κατασκευάσαμε επίσης ισοδύναμη CFG.

$$\boxed{\forall r \, \exists G_r : \; L(G_r) = L(r).}$$

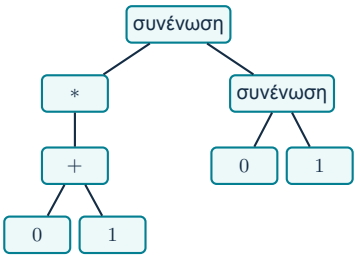
Άρα: Regular $\subseteq$ CFL. Δεν ισχύει το αντίστροφο.

Κατασκευαστικό παράδειγμα: $r = (0+1)^*01$

$$r = (0+1)^*01, \qquad L(r) = \{w01 : w \in \{0,1\}^*\}.$$

Αναλύουμε την έκφραση:

$$u = (0+1),$$
$$v = u^*,$$
$$t = (01),$$
$$r = vt.$$



Παραδείγματα

$01, 001, 101, 1101$ ανήκουν στη γλώσσα.

Στόχος: κατασκευάζουμε τη CFG *από τα φύλλα προς τη ρίζα*.

Παράδειγμα — Βήμα 1: κατασκευάζουμε το $0 + 1$

Ξεκινάμε από τις στοιχειώδεις εκφράσεις

$$\boxed{A \to 0}, \qquad \boxed{B \to 1}.$$

Για την ένωση $u = (0 + 1)$ προσθέτουμε νέα αρχική μεταβλητή U:

$$\boxed{U \to A \mid B.}$$

Επιλογή του 0

$$U \Rightarrow A \Rightarrow 0.$$

Επιλογή του 1

$$U \Rightarrow B \Rightarrow 1.$$

$$\boxed{L(U) = \{0, 1\} = L(0 + 1).}$$

Ερώτηση: γιατί δεν μπορεί το U να παράγει τη λέξη 01;

Παράδειγμα — Βήμα 2: εφαρμόζουμε το αστέρι Kleene

Τώρα θέλουμε $v = (0+1)^*$. Έχουμε ήδη $U \to A \mid B$ με $A \to 0$, $B \to 1$.

Προσθέτουμε νέα αρχική μεταβλητή K

$$\boxed{K \to \varepsilon \mid UK.}$$

Κάθε χρήση του UK προσθέτει **ένα ακόμη** 0 ή 1.

Παράδειγμα: παραγωγή της λέξης 10

$$K \Rightarrow UK \Rightarrow BK \Rightarrow 1K \Rightarrow 1UK$$
$$\Rightarrow 1AK \Rightarrow 10K \Rightarrow 10.$$

$$\boxed{L(K) = \{0,1\}^*.}$$

Γιατί παράγεται και η ε**;** Επειδή $K \to \varepsilon$.

Παράδειγμα — Βήμα 3: φτιάχνουμε το 01 και συνενώνουμε

Για το $t = 01$

Χρησιμοποιούμε **νέες** μεταβλητές:

$$\boxed{\begin{aligned} C &\to 0, \\ D &\to 1, \\ T &\to CD. \end{aligned}}$$

Άρα $L(T) = \{01\}$.

Για το $r = vt$

Συνενώνουμε τα δύο μέρη:

$$\boxed{S \to KT.}$$

$$\begin{aligned} L(S) &= L(K)L(T) \\ &= \{0,1\}^*\{01\}. \end{aligned}$$

Η τελική CFG συγκεντρωμένη

$$A \to 0, \quad B \to 1, \quad U \to A \mid B, \quad K \to \varepsilon \mid UK,$$
$$C \to 0, \quad D \to 1, \quad T \to CD, \quad S \to KT.$$

Αρχική μεταβλητή: S. Τερματικά: $\Sigma = \{0,1\}$.

Παράδειγμα — Πλήρης παραγωγή της λέξης 1101

Θέλουμε $S \Rightarrow^* 1101$. Πρώτα παράγουμε το πρόθεμα 11 από το K και έπειτα το 01 από το T.

Παραγωγή με τους κανόνες της κατασκευής

$$\begin{aligned} S &\Rightarrow KT \Rightarrow UKT \Rightarrow BKT \Rightarrow 1KT \\ &\Rightarrow 1UKT \Rightarrow 1BKT \Rightarrow 11KT \\ &\Rightarrow 11T \Rightarrow 11CD \Rightarrow 110D \Rightarrow \boxed{1101}. \end{aligned}$$

Το πρόθεμα

$K \Rightarrow^* 11$, και γενικά $K \Rightarrow^* w$ για κάθε $w \in \{0,1\}^*$.

Η κατάληξη

$T \Rightarrow CD \Rightarrow 0D \Rightarrow 01$.

Γιατί δεν παράγεται το 1110**;** Επειδή κάθε λέξη από το S τελειώνει σε 01.

Ίδια γλώσσα, διαφορετικές γραμματικές

Η προηγούμενη κατασκευή από regex έδωσε μια CFG με μεταβλητές A, B, U, K, C, D, T, S. Όμως η ίδια γλώσσα έχει και **πολύ απλούστερη** γραμματική.

Απλή δεξιόγραμμη CFG

$$\boxed{\begin{aligned} X &\to 0X \mid 1X \mid 0Y, \\ Y &\to 1. \end{aligned}}$$

Αρχικό: X.

Τι παράγει;

Οι κανόνες $0X$, $1X$ παράγουν οποιοδήποτε πρόθεμα.

Οι τελευταίοι κανόνες $X \to 0Y \to 01$ επιβάλλουν την κατάληξη 01.

$$\boxed{L(G) = \{w01 : w \in \{0,1\}^*\} = L(r).}$$

Διδακτικό συμπέρασμα

Η **κατασκευή regex** $\to$ **CFG** αποδεικνύει το θεώρημα. Η **κατασκευή DFA** $\to$ **CFG** δίνει μια άλλη απόδειξη. Δεν απαιτούμε οι δύο γραμματικές να έχουν τους ίδιους κανόνες.

Κανονική μορφή Chomsky

Ορισμός 14

Μια γραμματική είναι σε κανονική μορφή Chomsky (Chomsky normal form, CNF) αν:

- Το αρχικό μη τερματικό S δεν εμφανίζεται στο δεξιό μέλος κανενός κανόνα.
- Το S μπορεί να έχει τον κανόνα $S \to \varepsilon$.
- Κάθε άλλος κανόνας έχει τη μορφή $A \to BC$ (ακριβώς δύο μη τερματικά) ή $A \to a$ (ένα τερματικό).

Κανονική Μορφή Chomsky: το θεώρημα

Θεώρημα

Κάθε γλώσσα χωρίς συμφραζόμενα παράγεται από κάποια γραμματική σε Κανονική Μορφή Chomsky (CNF).

Ποιους κανόνες επιτρέπει η CNF;

$$A \to BC \qquad \text{ή} \qquad A \to a,$$

όπου A, B, C είναι μεταβλητές και a τερματικό. Αν $\varepsilon \in L(G)$, επιτρέπουμε επιπλέον $S_0 \to \varepsilon$, με το S_0 να μην εμφανίζεται σε κανένα δεξί μέλος.

Στρατηγική απόδειξης

Δεν περιγράφουμε απλώς τη γλώσσα: **μετασχηματίζουμε** οποιαδήποτε CFG βήμα-βήμα σε ισοδύναμη γραμματική CNF.

Η κατασκευή του Sipser: τέσσερα στάδια

1. **Νέα αρχική μεταβλητή:** $S_0 \to S$.
2. **Κατάργηση κανόνων ε:** διαγράφουμε $A \to \varepsilon$, προσθέτοντας τις παραλλαγές των κανόνων όπου παραλείπονται εμφανίσεις του A.
3. **Κατάργηση μοναδιαίων κανόνων:** διαγράφουμε $A \to B$ και μεταφέρουμε στο A τις μη μοναδιαίες παραγωγές που μπορεί να χρησιμοποιήσει το B.
4. **Τελική κανονικοποίηση:** σπάμε δεξιά μέλη μήκους ≥ 3 σε δυαδικούς κανόνες και αντικαθιστούμε τα τερματικά σε δεξιά μέλη μήκους ≥ 2 με νέες μεταβλητές.

Τι διατηρούμε;

Σε κάθε στάδιο η παραγόμενη γλώσσα παραμένει ίδια.

Παράδειγμα Sipser: η αρχική γραμματική G_6

Αρχικοί κανόνες

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

- $B \to \varepsilon$: κενή παραγωγή.
- $A \to B$, $A \to S$: μοναδιαίες παραγωγές.
- $S \to ASA$: δεξί μέλος μήκους 3.
- $S \to aB$: τερματικό μαζί με μεταβλητή.

Στόχος: να απομακρύνουμε διαδοχικά και τα τέσσερα εμπόδια.

Στάδιο 1: νέα αρχική μεταβλητή

Πριν

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

Μετά

$$S_0 \to S$$
$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

Γιατί προσθέτουμε το S_0;

Το νέο αρχικό σύμβολο δεν εμφανίζεται σε κανένα δεξί μέλος. Αυτό επιτρέπει να διαχειριστούμε με ασφάλεια την κενή λέξη κατά την απαλοιφή ε-κανόνων.

Στάδιο 2: ποιες μεταβλητές είναι μηδενίσιμες;

Ορισμός

Μια μεταβλητή X λέγεται μηδενίσιμη (nullable) αν $X \Rightarrow^* \varepsilon$.

$$B \Rightarrow \varepsilon, \qquad A \Rightarrow B \Rightarrow \varepsilon.$$

- Το B είναι μηδενίσιμο από τον κανόνα $B \to \varepsilon$.
- Το A είναι μηδενίσιμο, παρότι **δεν** έχει άμεσο κανόνα $A \to \varepsilon$.
- Το S **δεν** είναι μηδενίσιμο: ο κανόνας aB παράγει ένα a, ενώ ο ASA διατηρεί τουλάχιστον ένα S.

Κρίσιμο συμπέρασμα

Οι μεταβλητές που πρέπει να εξετάσουμε για απαλοιφή είναι $\{A, B\}$. Η αρχική γλώσσα **δεν περιέχει** το ε.

Απαλοιφή ε: πρώτα το $B \to \varepsilon$

Αρχικά

$$S \to ASA \mid aB, \qquad A \to B \mid S, \qquad B \to b \mid \varepsilon.$$

Εφόσον $B \Rightarrow \varepsilon$, ο κανόνας $S \to aB$ μπορεί επίσης να παράγει τη λέξη a.

$$\underbrace{S \Rightarrow aB \Rightarrow a}_{\text{παλιά παραγωγή}} \implies \underbrace{S \to a}_{\text{νέος κανόνας}} .$$

Μετά την απαλοιφή

$$S \to ASA \mid aB \mid a$$
$$A \to B \mid S$$
$$B \to b$$

Απαλοιφή ε: τώρα η μεταβλητή A

Επειδή $A \Rightarrow^* \varepsilon$, στον κανόνα $S \to ASA$ μπορούμε να παραλείψουμε την πρώτη, τη δεύτερη ή και τις δύο εμφανίσεις του A.

$$
\begin{array}{lcll}
ASA & \longrightarrow & ASA & \text{(κρατάμε και τις δύο)} \\
ASA & \longrightarrow & SA & \text{(παραλείπουμε την πρώτη)} \\
ASA & \longrightarrow & AS & \text{(παραλείπουμε τη δεύτερη)} \\
ASA & \longrightarrow & S & \text{(παραλείπουμε και τις δύο)}
\end{array}
$$

Προσοχή

Η απαλοιφή εφαρμόζεται **ξεχωριστά σε κάθε εμφάνιση** μιας μηδενίσιμης μεταβλητής. Δεν αφαιρούμε μηχανικά όλα τα A ταυτόχρονα.

Στάδιο 2: η γραμματική χωρίς ε-κανόνες

Αποτέλεσμα μετά τις δύο απαλοιφές

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \mid SA \mid AS \mid S \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

- Ο $S \to a$ διατηρεί τις παραγωγές όπου $B \Rightarrow \varepsilon$.
- Οι $S \to SA$, $S \to AS$, $S \to S$ διατηρούν τις παραγωγές όπου κάποια $A \Rightarrow^* \varepsilon$.
- Δεν υπάρχει πια κανόνας της μορφής $X \to \varepsilon$.

Τι μένει;

Οι $S_0 \to S$, $A \to B$, $A \to S$ και $S \to S$ είναι μοναδιαίοι κανόνες.

Στάδιο 3: τι είναι ένας μοναδιαίος κανόνας;

Ορισμός

Μοναδιαίος κανόνας (unit rule): $X \to Y$, όπου X, Y είναι μεταβλητές.

Βασική μετατροπή

Αν $A \to B$ και $B \to u$, τότε προσθέτουμε $A \to u$. Έτσι, η παραγωγή $A \Rightarrow B \Rightarrow u$ αντικαθίσταται από $A \Rightarrow u$.

- Επεκτείνουμε αυτή την ιδέα και σε **αλυσίδες** μοναδιαίων κανόνων.
- Στο τέλος αφαιρούμε τους μοναδιαίους κανόνες.
- Δεν αφαιρούμε τους κανόνες $A \to a$: αυτοί είναι έγκυροι στην CNF.

Μοναδιαίοι κανόνες του παραδείγματος

Μετά το στάδιο 2:

$$S_0 \to S, \qquad S \to S, \qquad A \to B \mid S.$$

Ποια μη μοναδιαία παραγωγή κληρονομεί κάθε μεταβλητή;

Μεταβλητή	Προσβάσιμες μέσω μοναδιαίων κανόνων
S_0	S_0, S
S	S
A	A, B, S
B	B

Παράδειγμα

Από $A \Rightarrow S \Rightarrow aB$ προσθέτουμε $A \to aB$. Από $A \Rightarrow B \Rightarrow b$ προσθέτουμε $A \to b$.

Απαλοιφή $S_0 \to S$ και $S \to S$

Ο S έχει τους μη μοναδιαίους κανόνες

$$S \to ASA \mid aB \mid a \mid SA \mid AS.$$

Αντικαθιστούμε $S_0 \to S$

Προσθέτουμε *όλους* τους μη μοναδιαίους κανόνες του S στο S_0:

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS.$$

Αφαιρούμε $S \to S$

Ο κανόνας $S \to S$ δεν συμβάλλει σε καμία πεπερασμένη παραγωγή τερματικής λέξης. Τον διαγράφουμε.

Απαλοιφή $A \to B$ και $A \to S$

Η μεταβλητή A κληρονομεί τις παραγωγές των B, S

$$B \to b$$
$$S \to ASA \mid aB \mid a \mid SA \mid AS$$

Άρα, όταν αφαιρέσουμε $A \to B \mid S$, παίρνουμε

$$\boxed{A \to b \mid ASA \mid aB \mid a \mid SA \mid AS.}$$

Τι πετύχαμε;

Κάθε παραγωγή που άρχιζε με $A \Rightarrow B$ ή $A \Rightarrow S$ μπορεί πλέον να ξεκινήσει απευθείας με μια μη μοναδιαία παραγωγή του A.

Στάδιο 3: η γραμματική χωρίς μοναδιαίους κανόνες

Ακριβές ενδιάμεσο αποτέλεσμα

$$
\begin{aligned}
S_0 &\to ASA \mid aB \mid a \mid SA \mid AS \\
S &\to ASA \mid aB \mid a \mid SA \mid AS \\
A &\to b \mid ASA \mid aB \mid a \mid SA \mid AS \\
B &\to b
\end{aligned}
$$

- Δεν υπάρχουν ε-κανόνες.
- Δεν υπάρχουν μοναδιαίοι κανόνες.
- Απομένουν δύο ειδών παραβάσεις της CNF: ASA (μήκος 3) και aB (τερματικό μαζί με μεταβλητή).

Στάδιο 4α: διασπάμε τον κανόνα $X \to ASA$

Η CNF επιτρέπει μόνο δύο μεταβλητές στα δεξιά

Ο κανόνας $X \to ASA$ έχει τρεις μεταβλητές. Εισάγουμε νέα μεταβλητή A_1 με

$$\boxed{A_1 \to SA}.$$

Έπειτα αντικαθιστούμε **κάθε** εμφάνιση του ASA από AA_1:

$$\boxed{X \to AA_1} \qquad (X \in \{S_0, S, A\}).$$

Ίδια παραγωγική δυνατότητα

$$X \Rightarrow ASA \Rightarrow^* uvw \quad \leftrightsquigarrow \quad X \Rightarrow AA_1 \Rightarrow ASA \Rightarrow^* uvw.$$

Στάδιο 4β: απομακρύνουμε τερματικό από το aB

Γιατί ο κανόνας $X \to aB$ δεν είναι CNF;

Η δεξιά πλευρά έχει **δύο** σύμβολα, αλλά το πρώτο είναι τερματικό. Στην CNF αυτό δεν επιτρέπεται.

Εισάγουμε νέα μεταβλητή U:

$$\boxed{U \to a}.$$

Και αντικαθιστούμε, για $X \in \{S_0, S, A\}$,

$$\boxed{X \to aB \qquad \rightsquigarrow \qquad X \to UB}.$$

Προσοχή στην εξαίρεση

Οι κανόνες $X \to a$ ή $X \to b$ είναι ήδη νόμιμοι στην CNF: έχουν μόνο ένα τερματικό στο δεξί μέλος και **δεν** τους αντικαθιστούμε.

Παράδειγμα Sipser: τελική γραμματική CNF

Η τελική γραμματική

$$
\begin{aligned}
S_0 &\to AA_1 \mid UB \mid a \mid SA \mid AS \\
S &\to AA_1 \mid UB \mid a \mid SA \mid AS \\
A &\to b \mid AA_1 \mid UB \mid a \mid SA \mid AS \\
A_1 &\to SA \\
U &\to a \\
B &\to b
\end{aligned}
$$

- Κάθε κανόνας είναι $X \to YZ$ ή $X \to a$.
- Το νέο αρχικό S_0 δεν εμφανίζεται δεξιά.
- Η γλώσσα **είναι ίδια** με αυτή της αρχικής G_6.

Έλεγχος κατασκευής: παράγουμε τη λέξη *bab*

Στην αρχική γραμματική

$$S \Rightarrow ASA \Rightarrow BSA \Rightarrow bSA$$
$$\Rightarrow baBA \Rightarrow baA \Rightarrow baB \Rightarrow bab.$$

Στην τελική γραμματική CNF

$$S_0 \Rightarrow AA_1 \Rightarrow bA_1 \Rightarrow bSA$$
$$\Rightarrow baA \Rightarrow bab.$$

Χρησιμοποιήσαμε $A_1 \to SA$, $S \to a$ και $A \to b$.

Τι δείχνει το παράδειγμα;

Η CNF χρησιμοποιεί **περισσότερα ενδιάμεσα βήματα**, αλλά παράγει τις ίδιες τερματικές λέξεις.

Έλεγχος και της λέξης a

Αρχική G_6

$$S \Rightarrow aB \Rightarrow a.$$

Το B μπορεί να εξαφανιστεί μέσω $B \to \varepsilon$.

Τελική CNF

$$S_0 \Rightarrow a.$$

Ο νέος κανόνας $S_0 \to a$ κωδικοποιεί την παλιά παραγωγή.

Αυτό είναι το νόημα της απαλοιφής ε

Δεν διαγράφουμε απλώς το $B \to \varepsilon$. Πρώτα προσθέτουμε **τους κανόνες που αντισταθμίζουν** την εξαφάνιση του B, ώστε να μη χαθούν λέξεις.

Γιατί λειτουργεί γενικά η μετατροπή;

1. **Νέα αρχική μεταβλητή:** δεν αλλάζει τις παραγωγές τερματικών λέξεων.
2. **Απαλοιφή ε:** κάθε δυνατότητα να εξαφανιστεί μηδενίσιμη μεταβλητή αντικαθίσταται από κατάλληλο κανόνα.
3. **Απαλοιφή μοναδιαίων:** κάθε αλυσίδα $X \Rightarrow^* Y$ μοναδιαίων βημάτων συμπτύσσεται, αντιγράφοντας τους μη μοναδιαίους κανόνες του Y στο X.
4. **Δυαδικοποίηση:** οι νέες μεταβλητές κωδικοποιούν υποσυμβολοσειρές δεξιών μελών, χωρίς αλλαγή των τερματικών λέξεων.

Συμπέρασμα

Παίρνουμε μια ισοδύναμη γραμματική με κανόνες $A \to BC$ ή $A \to a$ (και, όπου απαιτείται, $S_0 \to \varepsilon$).

Μικρή άσκηση: ελέγχουμε αν καταλάβαμε τη CNF

Δίνεται η γραμματική

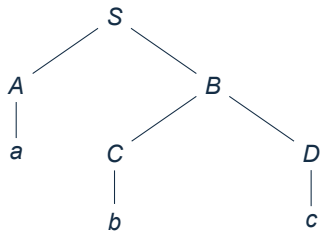
$$S \to aAb \mid b, \qquad A \to a \mid \varepsilon.$$

1. Ποιες μεταβλητές είναι μηδενίσιμες;
2. Ποιον νέο κανόνα προσθέτουμε όταν καταργούμε $A \to \varepsilon$;
3. Πώς αντικαθιστούμε το aAb με κανόνες CNF;
4. Γιατί το $S \to b$ δεν χρειάζεται αλλαγή;

Ενδεικτική απάντηση (για συζήτηση)

A μηδενίσιμη, προσθέτουμε $S \to ab$. Θέτουμε $U \to a$, $V \to b$, $T \to AV$ και $S \to UT \mid UV \mid b$, $A \to a$.

Γιατί χρήσιμη η CNF: δυαδικά δέντρα



Για λέξη μήκους $n \geq 1$, σε CNF:

- ακριβώς n εφαρμογές κανόνων $A \to a$,
- ακριβώς $n - 1$ εφαρμογές κανόνων $A \to BC$,
- άρα ακριβώς $2n - 1$ βήματα παραγωγής.

Γιατί μας ενδιαφέρει η Κανονική Μορφή Chomsky (CNF);

Η CNF επιβάλλει μια δυαδική δομή στα δέντρα παραγωγής, διευκολύνοντας τον σχεδιασμό αλγορίθμων και την απόδειξη θεωρημάτων.

- **Αλγόριθμος CYK:** Έλεγχος συμμετοχής $w \in L(G)$ με δυναμικό προγραμματισμό σε χρόνο $O(|G|n^3)$.
- **Μήκος παραγωγής:** Κάθε παραγωγή μη κενής λέξης μήκους n απαιτεί ακριβώς $2n - 1$ βήματα.
- **Αποφασισιμότητα συμμετοχής:** Αρκεί πεπερασμένη αναζήτηση παραγωγών μήκους $2n - 1$.
- **Λήμμα άντλησης για CFL**
- **Αποφασισιμότητα κενότητας:** Αν μια CFG παράγει κάποια μη κενή λέξη, υπάρχει και μία μήκους το πολύ 2^{k-1}, όπου k το πλήθος των μη τερματικών.
- ...

Ο αλγόριθμος CYK: συμμετοχή σε CFL

Πρόβλημα

Δίνεται CFG G σε CNF με αρχικό σύμβολο S_0 και λέξη $w = w_1 \cdots w_n$, όπου $n \geq 1$. Ελέγχουμε αν $w \in L(G)$.

Το κελί $T[i, j]$ περιέχει τα μη τερματικά που παράγουν ολόκληρη την υποσυμβολοσειρά από τη θέση i έως τη θέση j:

$$T[i, j] = \{A \mid A \Rightarrow^* w_i \cdots w_j\}.$$

1. Βάση: μήκος 1
Για κάθε θέση i, βρίσκουμε ποια μη τερματικά παράγουν απευθείας το w_i:

$$T[i, i] = \{A \mid A \to w_i \text{ κανόνας του } G\}.$$

2. Σειρά συμπλήρωσης
Προχωράμε κατά αύξον μήκος $\ell = 2, \ldots, n$. Για κάθε διάστημα $[i, j]$ μήκους ℓ, με $j = i + \ell - 1$, αρχικοποιούμε

$$T[i, j] \leftarrow \varnothing.$$

Έπειτα συνδυάζουμε μικρότερα διαστήματα

3. Επαγωγικό βήμα: συνδυασμός
Για κάθε $k = i, \ldots, j - 1$, χωρίζουμε την υποσυμβολοσειρά σε δύο μη κενά μέρη:

$$\underbrace{w_i \cdots w_k}_{\text{αριστερό}} \mid \underbrace{w_{k+1} \cdots w_j}_{\text{δεξί}}.$$

Για κάθε κανόνα $A \to BC$, ελέγχουμε:

- $B \in T[i, k]$: το B παράγει το αριστερό μέρος.
- $C \in T[k + 1, j]$: το C παράγει το δεξί μέρος.

Αν ισχύουν και τα δύο, τότε

CYK: Παράδειγμα για τη λέξη aabb (1/4)

Γραμματική σε CNF:

$$S_0, S \to AB \mid AT,$$
$$T \to SB,$$
$$A \to \mathrm{a}, \qquad B \to \mathrm{b}.$$

Θέλουμε να αποφασίσουμε αν $\mathrm{aabb} \in L(G)$, με αρχικό σύμβολο S_0.

Τι αποθηκεύουμε στον πίνακα;

$T[i,j]$ είναι το σύνολο των μη τερματικών που παράγουν τη συμβολοσειρά $w_i \cdots w_j$.

$$X \in T[i,j] \iff X \Rightarrow^* w_i \cdots w_j.$$

Βήμα 1: Υποσυμβολοσειρές μήκους 1

i	1	2	3	4
w_i	a	a	b	b
$T[i,i]$	$\{A\}$	$\{A\}$	$\{B\}$	$\{B\}$

CYK: Παράδειγμα για τη λέξη aabb (2/4)

Βήμα 2: Υποσυμβολοσειρές μήκους 2

- aa: κανένας κανόνας $X \to AA$, άρα $T[1,2] = \varnothing$.
- ab: $S_0, S \to AB$, άρα $T[2,3] = \{S_0, S\}$.
- bb: κανένας κανόνας $X \to BB$, άρα $T[3,4] = \varnothing$.

CYK: Παράδειγμα για τη λέξη aabb (3/4)

Βήμα 3: Υποσυμβολοσειρές μήκους 3
Για κάθε υποσυμβολοσειρά εξετάζουμε όλους τους δυνατούς διαχωρισμούς.

- Για τη aab:

$$\mathtt{a} \mid \mathtt{ab} \qquad \text{ή} \qquad \mathtt{aa} \mid \mathtt{b}.$$

  Κανένας διαχωρισμός δεν αντιστοιχεί σε κανόνα $X \to YZ$. Άρα $T[1,3] = \varnothing$.

- Για τη abb:

$$\underbrace{\mathtt{ab}}_{\{S_0,S\}} \mid \underbrace{\mathtt{b}}_{\{B\}}.$$

  Επειδή $T \to SB$, έχουμε

$$T[2,4] = \{T\}.$$

Βήμα 4: Ολόκληρη η λέξη aabb
Εξετάζουμε τους τρεις διαχωρισμούς:

$$\mathtt{a} \mid \mathtt{abb}, \qquad \mathtt{aa} \mid \mathtt{bb}, \qquad \mathtt{aab} \mid \mathtt{b}.$$

Μόνο ο πρώτος δίνει παραγωγή:

$$\underbrace{\mathtt{a}}_{\{A\}} \mid \underbrace{\mathtt{abb}}_{\{T\}}, \qquad S_0, S \to AT.$$

CYK: Παράδειγμα για τη λέξη aabb (4/4)

Ο πλήρης πίνακας δυναμικού προγραμματισμού:

Μήκος	$[1,\cdot]$	$[2,\cdot]$	$[3,\cdot]$	$[4,\cdot]$
1	$\{A\}$	$\{A\}$	$\{B\}$	$\{B\}$
2	$\varnothing$	$\{S_0, S\}$	$\varnothing$	–
3	$\varnothing$	$\{T\}$	–	–
4	$\{S_0, S\}$	–	–	–

Ανακατασκευή μιας παραγωγής

Από τα μη τερματικά του πίνακα βρίσκουμε:

$$\begin{aligned} S_0 &\Rightarrow AT \\ &\Rightarrow \mathrm{a}T \\ &\Rightarrow \mathrm{a}SB \\ &\Rightarrow \mathrm{a}ABB \end{aligned}$$

CYK: το κρίσιμο εσωτερικό βήμα

Για $w = \texttt{aabb}$ εξετάζουμε το υποδιάστημα $[2, 4]$, δηλαδή abb.

$$\underbrace{\texttt{ab}}_{[2,3]}\ \underbrace{\texttt{b}}_{[4,4]}.$$

- Στο $T[2, 3]$ υπάρχει S επειδή $S \to AB$.
- Στο $T[4, 4]$ υπάρχει B επειδή $B \to \texttt{b}$.
- Ο κανόνας $T \to SB$ μάς δίνει $T \in T[2, 4]$.

Τελικό βήμα

Για το $[1, 4]$: $A \in T[1, 1]$, $T \in T[2, 4]$ και $S_0 \to AT$, άρα αποδοχή.

CYK: ορθότητα και πολυπλοκότητα

Θεώρημα 15

Ο CYK αποφασίζει σωστά τη συμμετοχή $w \in L(G)$ για CFG σε CNF.

Ιδέα.

Επαγωγή στο μήκος $j - i + 1$. Κάθε παραγωγή από $A \to BC$ χωρίζει τη λέξη σε δύο *μη κενά* συνεχόμενα τμήματα· ο πίνακας εξετάζει ακριβώς όλους αυτούς τους χωρισμούς. □

Χρόνος

Με κατάλληλο ευρετήριο κανόνων: $O(n^3|R|)$, για δεδομένη γραμματική $O(n^3)$. Χώρος $O(n^2|V|)$. θα δούμε περισσότερα για πολυπλοκότητα αργότερα στο μάθημα

Αλγόριθμος CYK – Υλοποίηση σε Python

- Υλοποίηση του αλγορίθμου CYK σε Python.
- Παραδείγματα με διαφορετικές γραμματικές και συμβολοσειρές.

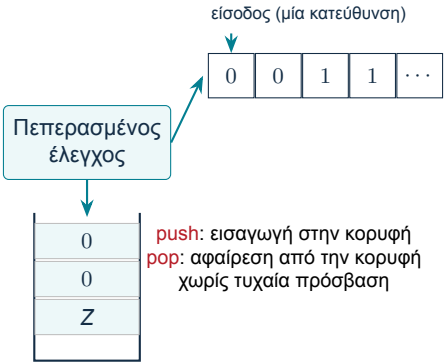
Άνοιγμα κώδικα στο Google Colab

HY280 · Θεωρία Υπολογισμού

3. Αυτόματα στοίβας (PDA)

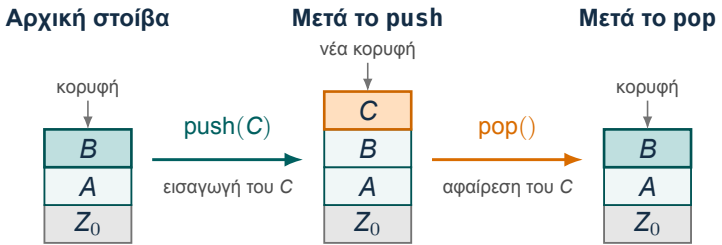
Διαμορφώσεις, ντετερμινισμός, εκτελέσεις

Από το DFA στο PDA: προσθέτουμε μία στοίβα



Στοίβα: η δομή LIFO (1/2)

Η στοίβα αποθηκεύει σύμβολα. Εισαγωγή και αφαίρεση γίνονται μόνο στην κορυφή της.



Z_0: ειδικό σύμβολο που σημειώνει τον πυθμένα της στοίβας

LIFO: τελευταίο μέσα, πρώτο έξω

Το C εισάγεται τελευταίο και αφαιρείται πρώτο. Για να φτάσουμε στο A, πρέπει πρώτα να αφαιρέσουμε όσα σύμβολα βρίσκονται από πάνω του.

Στα αυτόματα στοίβας (PDA): η στοίβα παρέχει μνήμη χωρίς προκαθορισμένο όριο ύψους.

Στοίβα: η δομή LIFO (2/2)

- **LIFO** (last in, first out): τελευταίο μέσα, πρώτο έξω.
- Διαβάζουμε και μεταβάλλουμε μόνο την *κορυφή*.
- Το πλήθος συμβόλων δεν έχει προκαθορισμένο όριο.
- Το κάτω σύμβολο Z σηματοδοτεί το άδειο ωφέλιμο περιεχόμενο.

Ενέργεια	στοίβα (κορυφή αριστερά)
αρχή	Z
push A	AZ
push B	BAZ
pop	AZ
pop	Z

Μνήμη και τάξη

Το PDA θυμάται *απεριόριστη ποσότητα* πληροφορίας, αλλά με *πολύ περιορισμένο τρόπο πρόσβασης*.

Τυπικός ορισμός PDA

Ορισμός 16

Ένα μη ντετερμινιστικό αυτόματο στοίβας είναι επτάδα

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

όπου Q πεπερασμένες καταστάσεις, Σ είσοδος, Γ αλφάβητο στοίβας, q_0 αρχική κατάσταση, Z_0 αρχικό σύμβολο στοίβας, $F \subseteq Q$ τελικές καταστάσεις, και

$$\boxed{\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)}.$$

Επιτρέπουμε αντικατάσταση μιας κορυφής με *ολόκληρη* λέξη $\gamma \in \Gamma^*$. Ισοδύναμη είναι η κλασική περιγραφή με μεμονωμένα push/pop και ενδιάμεσες καταστάσεις.

Τι σημαίνει η συνάρτηση μετάβασης;

Αν $(r, \gamma) \in \delta(q, a, X)$, μπορούμε να κάνουμε

$$\boxed{(q, aw, X\alpha) \vdash (r, w, \gamma\alpha)}$$

για $a \in \Sigma$. Αν $a = \varepsilon$, η είσοδος δεν μεταβάλλεται:

$$(q, w, X\alpha) \vdash (r, w, \gamma\alpha).$$

- Το αριστερό σύμβολο της λέξης στοίβας είναι η κορυφή.
- Η επιλογή (r, γ) μπορεί να είναι *μη ντετερμινιστική*.
- Δέχεται όταν υπάρχει εκτέλεση που *καταναλώνει όλη την είσοδο* και καταλήγει σε κατάσταση F.

PDA: τι κάνει μία μετάβαση;

Μία μετάβαση απαντά σε τέσσερα ερωτήματα

1. Σε ποια κατάσταση βρισκόμαστε;
2. Ποιο σύμβολο διαβάζουμε από την είσοδο;
3. Ποιο σύμβολο βρίσκεται στην κορυφή της στοίβας;
4. Σε ποια κατάσταση πηγαίνουμε και πώς αλλάζουμε τη στοίβα;

Ένα συγκεκριμένο παράδειγμα

$$(q_1, \texttt{0}, X) \longmapsto (q_2, YX)$$

- Βρισκόμαστε στην κατάσταση q_1.
- Διαβάζουμε το σύμβολο $\texttt{0}$.
- Η κορυφή της στοίβας είναι X.
- Πηγαίνουμε στην q_2 και αντικαθιστούμε το X με YX.

Άρα έχουμε push ενός Y πάνω από το X.

Αποκωδικοποιώντας τη συνάρτηση δ

$$\delta : \underbrace{Q}_{\text{κατάσταση}} \times \underbrace{(\Sigma \cup \{\varepsilon\})}_{\text{είσοδος ή } \varepsilon} \times \underbrace{\Gamma}_{\text{κορυφή}} \longrightarrow \mathcal{P}(\underbrace{Q}_{\text{νέα κατάσταση}} \times \underbrace{\Gamma^*}_{\text{νέα λέξη στοίβας}}).$$

Αριστερή πλευρά: τι γνωρίζουμε;

Το $\delta(q, a, X)$ εξετάζει:

- την τρέχουσα κατάσταση q,
- το επόμενο σύμβολο a (ή ε, χωρίς ανάγνωση),
- την κορυφή X της στοίβας.

Δεξιά πλευρά: τι επιτρέπεται να γίνει;

Η τιμή $\delta(q, a, X)$ είναι ένα σύνολο πιθανών αποτελεσμάτων.
Κάθε αποτέλεσμα (r, γ) σημαίνει:

- μετάβαση στην κατάσταση r,
- αντικατάσταση του X από τη λέξη γ.

Γιατί η δ επιστρέφει σύνολο;

Παράδειγμα μη ντετερμινιστικής επιλογής

Έστω

$$\boxed{\delta(q, \texttt{0}, X) = \{(q_1, YX),\ (q_2, \varepsilon)\}.}$$

Για την **ίδια** κατάσταση q, το **ίδιο** σύμβολο εισόδου $\texttt{0}$ και την **ίδια** κορυφή X, υπάρχουν δύο δυνατές μεταβάσεις.

Επιλογή 1

$$(q_1, YX)$$

Πηγαίνουμε στην q_1.
Αντικαθιστούμε το X με YX: push ενός Y.

Επιλογή 2

$$(q_2, \varepsilon)$$

Πηγαίνουμε στην q_2.
Αντικαθιστούμε το X με ε: pop του X.

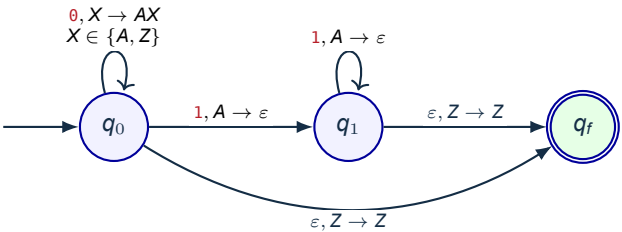
Τρεις δυνατότητες

- $\delta(q, a, X) = \emptyset$: καμία μετάβαση.
- $|\delta(q, a, X)| = 1$: μοναδική επιλογή.
- $|\delta(q, a, X)| > 1$: πολλές επιλογές.

PDA για 0^n1^n: αλφάβητα και καταστάσεις

$$L = \{0^n 1^n : n \geq 0\}, \qquad M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F).$$

- **Καταστάσεις:** $Q = \{q_0, q_1, q_f\}$.
- **Αλφάβητο εισόδου:** $\Sigma = \{0, 1\}$.
- **Αλφάβητο στοίβας:** $\Gamma = \{A, Z\}$.
- **Αρχική κατάσταση:** q_0.
- **Αρχική στοίβα:** Z, το σύμβολο πυθμένα.
- **Τελικές καταστάσεις:** $F = \{q_f\}$.



Δύο φάσεις: στο q_0 προσθέτουμε ένα A για κάθε 0. Στο q_1 αφαιρούμε ένα A για κάθε 1.

Πώς διαβάζουμε μια μετάβαση PDA;

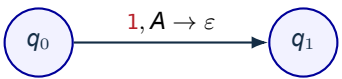
Μια ετικέτα έχει μορφή

$$a, X \to \gamma, \qquad a \in \Sigma \cup \{\varepsilon\}, \quad X \in \Gamma, \quad \gamma \in \Gamma^*.$$

1. **Είσοδος:** αν $a \neq \varepsilon$, το επόμενο σύμβολο πρέπει να είναι a και το καταναλώνουμε. Αν $a = \varepsilon$, η είσοδος παραμένει αμετάβλητη.
2. **Στοίβα:** η κορυφή πρέπει να είναι X. Αντικαθιστούμε το X με τη λέξη γ. Αν $\gamma = \varepsilon$, απλώς αφαιρούμε το X.
3. **Κατάσταση:** μεταβαίνουμε στην κατάσταση όπου καταλήγει η ακμή.

Σύμβαση: γράφουμε την κορυφή αριστερά

Αν η στοίβα είναι $X\beta$, μετά τη μετάβαση γίνεται $\gamma\beta$. Το υπόλοιπο β παραμένει αμετάβλητο.
Για παράδειγμα, η αντικατάσταση $A \to AA$ μετατρέπει τη στοίβα AZ σε AAZ: προσθέτει ένα νέο A.



Στη συγκεκριμένη ακμή, καταναλώνουμε το πρώτο 1, αφαιρούμε ένα A και αλλάζουμε φάση.

Οι μεταβάσεις του συγκεκριμένου PDA

Η συνάρτηση μεταβάσεων έχει τύπο

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma^*).$$

Για δεδομένα q, a, X, επιστρέφει τα δυνατά ζεύγη

(επόμενη κατάσταση, λέξη που αντικαθιστά την κορυφή).

q	a	X	$\delta(q, a, X)$	Ενέργεια
q_0	0	Z	$\{(q_0, AZ)\}$	Προσθήκη ενός A πάνω από το Z.
q_0	0	A	$\{(q_0, AA)\}$	Προσθήκη ενός ακόμη A.
q_0	1	A	$\{(q_1, \varepsilon)\}$	Αφαίρεση ενός A και αλλαγή φάσης.
q_1	1	A	$\{(q_1, \varepsilon)\}$	Αφαίρεση ενός A, στην ίδια φάση.
q_0	ε	Z	$\{(q_f, Z)\}$	Μετάβαση στο q_f, χωρίς ανάγνωση.
q_1	ε	Z	$\{(q_f, Z)\}$	Μετάβαση στο q_f, χωρίς ανάγνωση.

Όλες οι υπόλοιπες τιμές της δ είναι $\varnothing$. Ειδικότερα, από το q_f δεν υπάρχει εξερχόμενη μετάβαση.

Το ε δηλώνει την κενή λέξη. Δεν είναι σύμβολο του Σ ή του Γ.

Παράδειγμα αποδοχής: η λέξη 0011

Μια **διαμόρφωση** γράφεται (q, u, α): τρέχουσα κατάσταση q, αδιάβαστη είσοδος u και περιεχόμενο στοίβας α.

Η κορυφή της στοίβας βρίσκεται αριστερά.

Βήμα	Κατάσταση	Υπόλοιπο εισόδου	Στοίβα	Μετάβαση που έγινε
0	q_0	0011	Z	–
1	q_0	011	AZ	$0, Z \to AZ$
2	q_0	11	AAZ	$0, A \to AA$
3	q_1	1	AZ	$1, A \to \varepsilon$
4	q_1	ε	Z	$1, A \to \varepsilon$
5	q_f	ε	Z	$\varepsilon, Z \to Z$

Τι μετρούν τα A;

Κάθε A αντιστοιχεί σε ένα 0 που δεν έχει ακόμη αντιστοιχιστεί με ένα 1. Στο βήμα 3, το πρώτο 1 αφαιρεί ένα A καθώς αλλάζουμε από q_0 σε q_1.

Πότε ακριβώς αποδέχεται το PDA;

Αποδοχή με τελική κατάσταση

Αρκεί να υπάρχει ένας υπολογισμός που καταναλώνει ολόκληρη την είσοδο και καταλήγει στο q_f. Στο συγκεκριμένο PDA, απομένει τότε το Z στη στοίβα.

Παράδειγμα 1: η κενή λέξη ε

$$(q_0, \varepsilon, Z) \vdash (q_f, \varepsilon, Z).$$

Αυτή είναι η περίπτωση $n = 0$.

Παράδειγμα 2: η λέξη `01`

$$(q_0, \texttt{01}, Z) \vdash (q_0, \texttt{1}, AZ) \vdash (q_1, \varepsilon, Z)$$
$$\vdash (q_f, \varepsilon, Z).$$

Η ε-μετάβαση δεν ελέγχει το τέλος της εισόδου.
Επιτρέπεται ακόμη και η πρόωρη κίνηση

$$(q_0, \texttt{01}, Z) \vdash (q_f, \texttt{01}, Z).$$

Ο συγκεκριμένος κλάδος δεν αποδέχεται: απομένει η είσοδος `01` και από το q_f δεν υπάρχει συνέχεια. Ο προηγούμενος επιτυχημένος υπολογισμός αρκεί για να γίνει δεκτή η λέξη.

Παραδείγματα απόρριψης στο ίδιο PDA

1. `001`**: περισσότερα** `0` **από** `1`**.**

$$(q_0, \texttt{001}, Z) \vdash^* (q_1, \varepsilon, AZ).$$

Η είσοδος τελείωσε, αλλά περισσεύει ένα A. Δεν υπάρχει διαθέσιμη μετάβαση και το q_1 δεν είναι τελική κατάσταση.

2. `011`**: περισσότερα** `1` **από** `0`**.**

$$(q_0, \texttt{011}, Z) \vdash^* (q_1, \texttt{1}, Z) \vdash (q_f, \texttt{1}, Z).$$

Δεν υπάρχει A για το τελευταίο `1`. Η μετάβαση στο q_f αφήνει αδιάβαστη είσοδο.

3. `0101`**: λανθασμένη σειρά συμβόλων.**

$$(q_0, \texttt{0101}, Z) \vdash^* (q_1, \texttt{01}, Z) \vdash (q_f, \texttt{01}, Z).$$

Στη φάση q_1 δεν υπάρχει μετάβαση που να διαβάζει `0`. Παρότι τα πλήθη είναι ίσα, η είσοδος δεν καταναλώνεται ολόκληρη.

Τυχόν πρόωρη μετάβαση από το αρχικό q_0 στο q_f επίσης αφήνει αδιάβαστη είσοδο. Επομένως, κανένας κλάδος δεν αποδέχεται αυτές τις λέξεις.

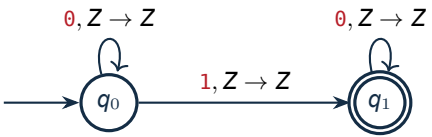
Μπορούμε να προσομοιώσουμε DFA με PDA;

Ναι: αγνοούμε τη στοίβα

Για κάθε ακμή DFA $q \xrightarrow{a} r$ εισάγουμε στο PDA την κίνηση

$$(r, Z) \in \delta(q, a, Z).$$

Το περιεχόμενο παραμένει πάντοτε Z.



Αναγνώριση λέξεων που περιέχουν ένα `1` και μετά μόνο `0` (κατάλληλο παράδειγμα υπογλώσσας).

Παράδειγμα εκτέλεσης: η λέξη 0011

Βήμα	κατάσταση	αδιάβαστη είσοδος	στοίβα
0	q_0	0011	Z
1	q_0	011	AZ
2	q_0	11	AAZ
3	q_1	1	AZ
4	q_1	ε	Z
5	q_f	ε	Z

Παρατήρηση: για να γίνει δεκτό το 00111 θα χρειαζόταν ένα τρίτο A, που δεν υπάρχει.

Παγίδα: τελική κατάσταση πριν τελειώσει η είσοδος

Λανθασμένη σκέψη

«Μόλις φτάσουμε στο q_f, η λέξη γίνεται δεκτή».

Σωστός κανόνας

$$w \in L(M) \iff \exists f \in F, \alpha \in \Gamma^* : \quad (q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha).$$

Η *υπόλοιπη* είσοδος πρέπει να είναι ε.

Αντιπαράδειγμα

Αν σε οποιοδήποτε PDA φτάσουμε σε τελική κατάσταση πριν διαβάσουμε τα δύο τελευταία `1` της `0011`, αυτό ακόμη δεν αποτελεί αποδοχή.

Αποδοχή με τελική κατάσταση ή με κενή στοίβα;

Αποδοχή με τελική κατάσταση

$$(q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha), \quad f \in F.$$

Δεν απαιτούμε κενή στοίβα.

Αποδοχή με κενή στοίβα

$$(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon).$$

Δεν απαιτούμε τελική κατάσταση.

Σημαντικό

Για *μη ντετερμινιστικά* PDA οι δύο τρόποι έχουν *ίδια εκφραστική ισχύ*: αναγνωρίζουν ακριβώς τις CFL. Για DPDA απαιτούνται προσεκτικότερες συμβάσεις.

Σωστά εμφωλευμένες παρενθέσεις: η γλώσσα

Ορισμός

Έστω $\Sigma = \{(,)\}$. Η γλώσσα D περιέχει όλες τις συμβολοσειρές σωστά εμφωλευμένων παρενθέσεων.
Μια λέξη w ανήκει στην D αν:

1. Κάθε πρόθεμα της w έχει τουλάχιστον τόσες ανοιχτές όσες κλειστές παρενθέσεις.
2. Συνολικά, οι ανοιχτές και οι κλειστές παρενθέσεις είναι ισάριθμες.

Παραδείγματα

$\varepsilon \in D$ (κενή λέξη)
$() \in D$ (ένα ζεύγος)
$(()) \in D$ (εμφώλευση)
$()() \in D$ (διαδοχικά ζεύγη)
$(()()) \in D$ (συνδυασμός)

Βασική ιδέα

Όταν διαβάζουμε μία $($, δημιουργούμε μια **εκκρεμότητα**.
Μια επόμενη $)$ πρέπει να κλείσει μία τέτοια εκκρεμότητα.

Πώς χρησιμοποιούμε τη στοίβα;

Τι αποθηκεύει η στοίβα;

Η στοίβα περιέχει ένα σύμβολο (για κάθε ανοιχτή παρένθεση που **δεν έχει ακόμη κλείσει**.
Το Z σηματοδοτεί τον πυθμένα της στοίβας.

1. **Αρχή:** Η στοίβα περιέχει μόνο Z.
2. **Διαβάζουμε (:** Ωθούμε μία (στη στοίβα (push).
3. **Διαβάζουμε):** Αφαιρούμε μία (από την κορυφή (pop).
   Αν στην κορυφή βρίσκεται το Z, απορρίπτουμε.
4. **Τέλος εισόδου:** Αποδεχόμαστε αν η στοίβα περιέχει μόνο το Z.

Γιατί λειτουργεί;

Μετά την ανάγνωση οποιουδήποτε προθέματος u, ο αριθμός των (στη στοίβα είναι

$$\boxed{\#_{(}(u) - \#_{)}(u).}$$

Το πλήθος αυτό δεν επιτρέπεται να γίνει αρνητικό.

Εκτέλεση PDA: η λέξη (()())

Θα παρακολουθήσουμε την είσοδο και τη στοίβα μετά από κάθε σύμβολο.

Βήμα προς βήμα

Βήμα	Διαβάζουμε	Στοίβα	Ενέργεια
0	–	Z	Αρχικοποίηση
1	(	$Z($	push
2	(	$Z(($	push
3	)	$Z($	pop
4	(	$Z(($	push
5	)	$Z($	pop
6	)	Z	pop

Η κορυφή της στοίβας εμφανίζεται δεξιά.

Αποτέλεσμα

Η είσοδος καταναλώθηκε πλήρως και στη στοίβα απέμεινε μόνο το Z.

Πότε αποτυγχάνει το PDA;

Παράδειγμα 1: (()

Είσοδος	Στοίβα
ε	Z
(	$Z($
((	$Z(($
(()	$Z($

Στο τέλος υπάρχει μία ανοιχτή παρένθεση.

Απόρριψη.

Παράδειγμα 2: ())

Είσοδος	Στοίβα
ε	Z
(	$Z($
()	Z
())	αποτυχία

Η τελευταία) δεν έχει αντίστοιχη (.

Απόρριψη.

Δύο διαφορετικοί λόγοι απόρριψης

1. Τελείωσε η είσοδος, αλλά παραμένουν ανοιχτές παρενθέσεις.
2. Εμφανίστηκε κλειστή παρένθεση χωρίς διαθέσιμη ανοιχτή παρένθεση.

PDA για σωστά εμφωλευμένες παρενθέσεις

Έστω $\Sigma = \{(,)\}$, $\Gamma = \{Z, (\}$, όπου Z είναι το σύμβολο πυθμένα της στοίβας.



Τυπικές μεταβάσεις

1.	$(,\ Z \to (Z$	Push της πρώτης ανοιχτής παρένθεσης.
2.	$(,\ (\to (($	Push μίας επιπλέον παρένθεσης.
3.	$),\ (\to \varepsilon$	Pop μίας ανοιχτής παρένθεσης.
4.	$\varepsilon,\ Z \to Z$	Μετάβαση στην τελική κατάσταση.

Πότε αποδεχόμαστε;

Όταν έχει καταναλωθεί **ολόκληρη η είσοδος** και η στοίβα περιέχει μόνο το Z.
Δεν υπάρχει μετάβαση για $)$ με κορυφή Z.

Δύο είδη παρενθέσεων: η στοίβα θυμάται τον τύπο

Κανόνες μετάβασης στο ίδιο q

$$
\begin{aligned}
(, X &\to (X \quad (X \in \Gamma),\\
[, X &\to [X \quad (X \in \Gamma),\\
), (&\to \varepsilon,\\
], [&\to \varepsilon.
\end{aligned}
$$

όπου $\Gamma = \{Z, (, [\}$.

Για την $([)]$

Μετά το πρόθεμα $([$ η κορυφή είναι $[$. Ένα $)$ απαιτεί $($· δεν υπάρχει επιτρεπτή μετάβαση.

Ντετερμινιστικά και μη ντετερμινιστικά PDA

NPDA

Μπορεί να υπάρχουν πολλοί δυνατοί επόμενοι υπολογισμοί από την ίδια διαμόρφωση. Αρκεί *ένας* να καταλήξει σε αποδοχή.

DPDA

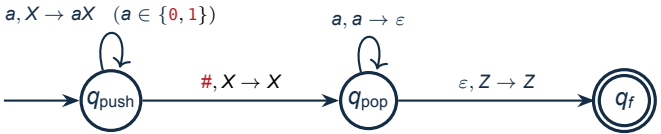
Κάθε διαμόρφωση καθορίζει το πολύ μία κίνηση. Δεν επιτρέπεται επιλογή ανάμεσα σε ε-κίνηση και κίνηση που καταναλώνει είσοδο με το ίδιο (q, X).

Μεγάλη διαφορά από τα NFA/DFA

Στα πεπερασμένα αυτόματα, ντετερμινισμός και μη ντετερμινισμός έχουν ίδια ισχύ. Στα PDA: DCFL $\subsetneq$ CFL.

Παλίνδρομοι με διαχωριστή #: ένας ντετερμινιστικός PDA

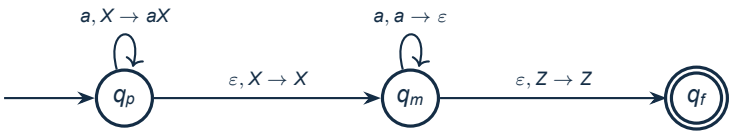
$$L_{\#} = \{w\#w^R \mid w \in \{0,1\}^*\}.$$



Το # μάς λέει ακριβώς πότε πρέπει να σταματήσουμε το push και να αρχίσουμε pop.

Χωρίς διαχωριστή: πρέπει να μαντέψουμε το κέντρο

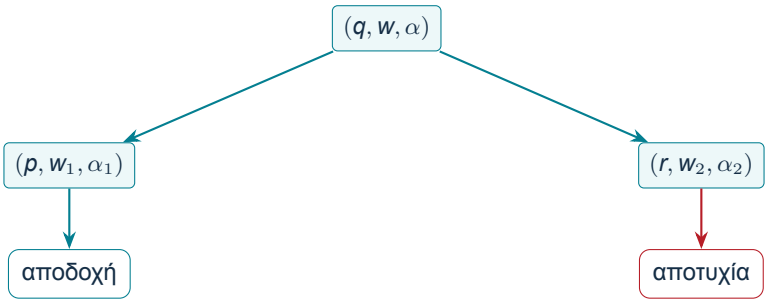
$$L_{\text{even}} = \{ww^R \mid w \in \{0,1\}^*\}.$$



Το κρίσιμο βήμα

Σε οποιαδήποτε στιγμή το NPDA *μαντεύει* με ε-μετάβαση ότι έφθασε στο μέσο. Οι λανθασμένες επιλογές αποτυγχάνουν· μία σωστή αρκεί.

Μη ντετερμινισμός: τι είναι μία διακλάδωση;



Υπαρξιακή σημασιολογία

$w \in L(M)$ αν υπάρχει *τουλάχιστον ένα* πεπερασμένο μονοπάτι αποδοχής. Δεν «τρέχουμε μαγικά» μόνο τον σωστό κλάδο: η σημασιολογία εξετάζει όλες τις δυνατότητες.

DPDA και γραμματικές LR(1): η βασική ιδέα

Γραμματική LR(1)

Μια γραμματική χωρίς συμφραζόμενα που επιτρέπει ντετερμινιστική συντακτική ανάλυση:

- **L (Left-to-right):** διαβάζουμε την είσοδο από αριστερά προς τα δεξιά.
- **R (Rightmost):** ανακατασκευάζουμε μια δεξιότερη παραγωγή με αντίστροφη σειρά.
- **1:** χρησιμοποιούμε ένα σύμβολο προεπισκόπησης (*lookahead*), δηλαδή το επόμενο αδιάβαστο σύμβολο, για να επιλέξουμε την ενέργεια.

Πώς συνδέονται;

Οι DPDA αναγνωρίζουν ακριβώς τις γλώσσες που έχουν **κάποια γραμματική LR(1)**.
Αυτές ονομάζονται **ντετερμινιστικές γλώσσες χωρίς συμφραζόμενα** (DCFL).

Τι να θυμάστε;

Κανονικές Γλώσσες $\subsetneq$ Ντετερμινιστικές Context-Free γλώσσες $\subsetneq$ Context-Free γλώσσες.

L αναγνωρίζεται από εάν DPDA $\iff$ υπάρχει LR(1) γραμματική G με $L(G) = L$.

HY280 · Θεωρία Υπολογισμού

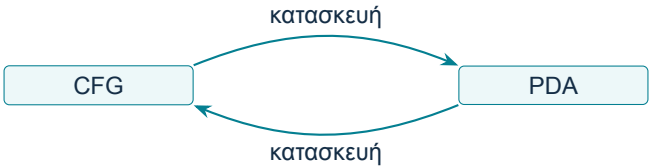
4. Ισοδυναμία CFG και PDA

Το κεντρικό θεώρημα

Θεώρημα 17 (Ισοδυναμία CFG και PDA)

Για κάθε γλώσσα $L \subseteq \Sigma^*$, τα ακόλουθα είναι ισοδύναμα:

1. $L = L(G)$ *για κάποια γραμματική χωρίς συμφραζόμενα* G.
2. $L = L(M)$ *για κάποιο μη ντετερμινιστικό αυτόματο στοίβας* M.



Απόδειξη στο Φροντιστήριο τη Παρασκευή

HY280 · Θεωρία Υπολογισμού

6. Ιδιότητες κλειστότητας των CFL

Τι διατηρείται και τι όχι

Πώς συγκρίνονται οι κλάσεις γλωσσών;



Κλειστότητα CFL: ένωση, παράθεση, Kleene star

Έστω CFG G_1, G_2 με αρχικά S_1, S_2 και *ξένα* σύνολα μη τερματικών.

Επιχείρηση	νέοι κανόνες αρχικού S
$L_1 \cup L_2$	$S \to S_1 \mid S_2$
$L_1 L_2$	$S \to S_1 S_2$
L_1^*	$S \to \varepsilon \mid S_1 S$

Κατασκευαστική απόδειξη

Δεν χρειάζεται να ανασχεδιάσουμε το PDA: χτίζουμε απευθείας γραμματική από τα υπάρχοντα συστατικά.

Παράδειγμα κλειστότητας: μία CFL και μία κανονική

$$L_1 = \{0^n 1^n \mid n \geq 0\}, \qquad L_2 = 2^*.$$

Η παράθεση $L_1 L_2$ είναι CFL

$$S \to AB,$$
$$A \to 0A1 \mid \varepsilon,$$
$$B \to 2B \mid \varepsilon.$$

Δοκιμή

Η λέξη `0011222` ανήκει· η `001212` όχι (η σειρά των μπλοκ είναι υποχρεωτική).

Τομή CFL με κανονική γλώσσα: ναι

Θεώρημα 18

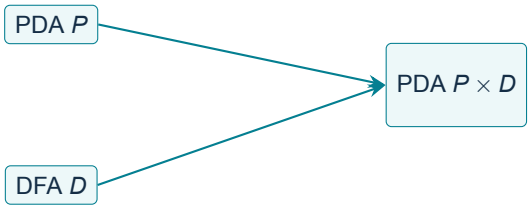
Αν L είναι CFL και R είναι κανονική, τότε $L \cap R$ είναι CFL.

Κατασκευή.

- Πάρε PDA P για το L και DFA D για το R.
- Στον πεπερασμένο έλεγχο του νέου PDA διατήρησε ζεύγος καταστάσεων (q_P, q_D).
- Σε ανάγνωση a, προχώρησε *και* στο PDA *και* στο DFA· σε ε-κίνηση PDA κράτησε ακίνητο το DFA.
- Αποδέξου μόνο αν αποδέχονται και τα δύο.

□

Προϊόν PDA $\times$ DFA: το διάγραμμα



Χρήσιμο ως τεχνική απόδειξης

Μπορούμε να «φιλτράρουμε» μία CFL με ένα κανονικό μοτίβο, π.χ. να κρατήσουμε μόνο λέξεις της μορφής $0^*1^*2^*$.

Η τομή δύο CFL μπορεί να μην είναι CFL

$$L_1 = \{0^n 1^n 2^m \mid n, m \geq 0\},$$
$$L_2 = \{0^m 1^n 2^n \mid n, m \geq 0\}.$$

- Η L_1 ελέγχει $\#0 = \#1$.
- Η L_2 ελέγχει $\#1 = \#2$.
- Και οι δύο είναι CFL με απλές γραμματικές.

Αλλά η τομή είναι

$$\boxed{L_1 \cap L_2 = \{0^n 1^n 2^n \mid n \geq 0\}},$$

η οποία δεν είναι CFL (θα αποδειχθεί με pumping).

Γιατί οι CFL δεν είναι κλειστές στο συμπλήρωμα;

Απόδειξη μέσω De Morgan

Αν οι CFL ήταν κλειστές στο συμπλήρωμα, τότε για CFL A, B θα είχαμε

$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

επίσης CFL, επειδή είναι κλειστές στην ένωση.

Αντίφαση

Γνωρίζουμε ζεύγος CFL με μη-CFL τομή. Επομένως οι CFL *δεν* είναι κλειστές στο συμπλήρωμα.

Τα συμπληρώματα λαμβάνονται ως προς το ίδιο Σ^*.

Πίνακας κλειστότητας

Πράξη	Regular	CFL
Ένωση	✓	✓
Παράθεση	✓	✓
Kleene star	✓	✓
Αντιστροφή λέξεων (reversal)	✓	✓
Τομή με regular	✓	✓
Τομή δύο γλωσσών της ίδιας κλάσης	✓	×
Συμπλήρωμα	✓	×
Διαφορά δύο γλωσσών της ίδιας κλάσης	✓	×

Το × σημαίνει «δεν ισχύει για κάθε ζεύγος», όχι «δεν συμβαίνει ποτέ».

Η αντίστροφη μιας CFL είναι CFL

Αν μια CFG περιέχει τον κανόνα

$$A \to X_1 X_2 \cdots X_k,$$

κατασκευάζουμε γραμματική που περιέχει

$$A \to X_k \cdots X_2 X_1.$$

Συμπέρασμα

Η νέα γραμματική παράγει ακριβώς $L(G)^R = \{w^R : w \in L(G)\}$.

Παράδειγμα

Η $\{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$ αντιστρέφεται στη $\{\mathtt{1}^n\mathtt{0}^n : n \geq 0\}$, που έχει γραμματική $S \to \mathtt{1}S\mathtt{0} \mid \varepsilon$.

HY280 · Θεωρία Υπολογισμού

7. Το Pumping Lemma για CFL

Πώς αποδεικνύουμε ότι μια γλώσσα δεν είναι χωρίς συμφραζόμενα

Το pumping lemma για γλώσσες χωρίς συμφραζόμενα

Θεώρημα 19 (Pumping lemma για CFL)

Για κάθε CFL L υπάρχει ακέραιος $p \geq 1$ ώστε για κάθε $s \in L$ με $|s| \geq p$ υπάρχει διαμέριση

$$\boxed{s = uvxyz}$$

που ικανοποιεί ταυτόχρονα:

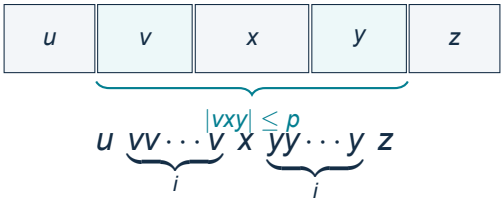
(i) $uv^i xy^i z \in L \quad$ *για κάθε* $i \geq 0$,

(ii) $|vy| > 0$,

(iii) $|vxy| \leq p$.

Τα v, y αντλούνται *συγχρονισμένα*. Επιτρέπεται το ένα από τα δύο να είναι κενό, όχι όμως και τα δύο.

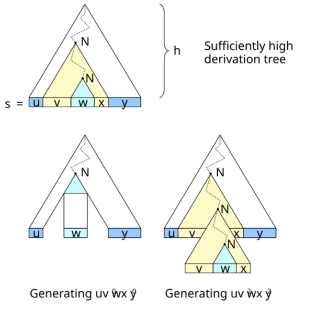
Σχηματικά: ποια κομμάτια επαναλαμβάνονται;



Δύο αλλαγές, ίδιο πλήθος επαναλήψεων

Το μήκος κάθε αντλημένης λέξης είναι $|s| + (i-1)(|v| + |y|)$.

Ιδέα απόδειξης (source: Wikipedia)



Proof idea: If s is sufficiently long, its derivation tree w.r.t. a Chomsky normal form grammar must contain some nonterminal N twice on some tree path (upper picture). Repeating n times the derivation part $N \Rightarrow ... \Rightarrow vNx$ obtains a derivation for $uv^n wx^n y$ (lower left and right picture for $n = 0$ and 2, respectively).

Ιδέα απόδειξης (1/5): Ύψος του δέντρου παραγωγής

Έστω L μια γλώσσα χωρίς συμφραζόμενα και $G = (V, \Sigma, R, S)$ μια γραμματική σε κανονική μορφή Chomsky (CNF) τέτοια ώστε $L(G) = L$.

Έστω $k = |V|$ το πλήθος των μη τερματικών συμβόλων της γραμματικής.

Βασική παρατήρηση

Στην κανονική μορφή Chomsky, οι κανόνες παραγωγής είναι της μορφής

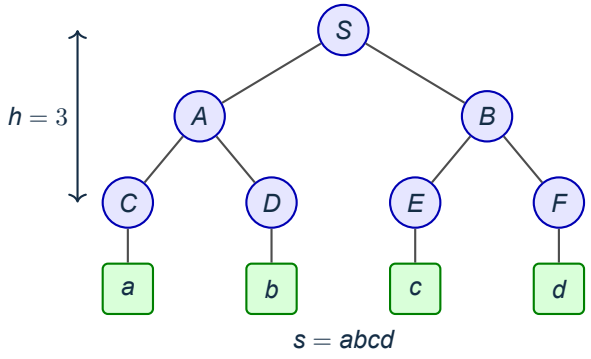
$$A \to BC \qquad \text{ή} \qquad A \to a,$$

με πιθανή εξαίρεση τον κανόνα $S \to \varepsilon$.

- Κάθε μη τερματικός κόμβος έχει το πολύ δύο παιδιά.
- Αν κάθε μονοπάτι από τη ρίζα σε φύλλο περιέχει το πολύ h μη τερματικούς κόμβους, τότε το δέντρο παράγει το πολύ 2^{h-1} τερματικά σύμβολα.

$$\boxed{|s| \leq 2^{h-1}}$$

Γιατί $|s| \leq 2^{h-1}$;



$1 = 2^0$ μη τερματικό

$2 = 2^1$ μη τερματικά

$4 = 2^2$ μη τερματικά

4 τερματικά

Παρατήρηση

Έχουμε $h = 3$ επίπεδα μη τερματικών συμβόλων, αλλά μόνο $h - 1 = 2$ επίπεδα δυαδικής διακλάδωσης. Επομένως,

$$\boxed{|s| = 4 = 2^{3-1} = 2^{h-1}.}$$

Βοηθητικό λήμμα: Πλήθος τερματικών συμβόλων

Λήμμα 20

Αν κάθε μονοπάτι από τη ρίζα σε φύλλο ενός δέντρου παραγωγής σε CNF περιέχει το πολύ h μη τερματικούς κόμβους, τότε το δέντρο παράγει το πολύ 2^{h-1} τερματικά σύμβολα.

Απόδειξη.

Με επαγωγή ως προς h.

Βάση ($h = 1$): Ο μόνος δυνατός κανόνας για παραγωγή μη κενής συμβολοσειράς είναι $A \to a$. Άρα παράγεται ένα τερματικό σύμβολο και $1 = 2^0$.

Επαγωγικό βήμα ($h > 1$): Αν η ρίζα χρησιμοποιεί κανόνα $A \to BC$, τότε καθένα από τα δύο υποδέντρα έχει το πολύ $h - 1$ μη τερματικούς κόμβους ανά μονοπάτι.
Από την επαγωγική υπόθεση, το συνολικό πλήθος τερματικών συμβόλων είναι το πολύ

$$2^{h-2} + 2^{h-2} = \boxed{2^{h-1}}.$$

Αν χρησιμοποιείται κανόνας $A \to a$, το φράγμα ισχύει τετριμμένα. □

Ιδέα απόδειξης (2/5): Αρχή του περιστερώνα

Επιλέγουμε ως μήκος άντλησης (pumping length)

$$\boxed{p = 2^k}, \qquad k = |V|.$$

Έστω οποιαδήποτε συμβολοσειρά $s \in L$ με $|s| \geq p$.

- Αν κάθε μονοπάτι του δέντρου παραγωγής περιείχε το πολύ k μη τερματικούς κόμβους, τότε
$$|s| \leq 2^{k-1} < 2^k = p,$$
άτοπο.
- Άρα, υπάρχει μονοπάτι από τη ρίζα σε φύλλο με τουλάχιστον $k+1$ μη τερματικούς κόμβους.
- Εξετάζουμε τους τελευταίους $k+1$ μη τερματικούς κόμβους ενός μακρύτερου τέτοιου μονοπατιού.

Αρχή του περιστερώνα (pigeonhole principle)

Έχουμε $k+1$ κόμβους αλλά μόνο k διαφορετικά μη τερματικά σύμβολα. Επομένως, κάποιο μη τερματικό σύμβολο $N \in V$ εμφανίζεται τουλάχιστον δύο φορές στο ίδιο μονοπάτι.

Ιδέα απόδειξης (3/5): Διάσπαση της συμβολοσειράς

Οι δύο εμφανίσεις του μη τερματικού συμβόλου N στο ίδιο μονοπάτι καθορίζουν τρεις παραγωγές:

$$S \Rightarrow^* uNy,$$

$$N \Rightarrow^* vNx,$$

$$N \Rightarrow^* w.$$

Συνεπώς, η αρχική παραγωγή της συμβολοσειράς s είναι

$$S \Rightarrow^* uNy \Rightarrow^* uvNxy \Rightarrow^* uvwxy = s.$$

Το κρίσιμο σημείο

Η παραγωγή

$$\boxed{N \Rightarrow^* vNx}$$

ξεκινά από το μη τερματικό N και καταλήγει ξανά στο ίδιο μη τερματικό, παράγοντας επιπλέον τις συμβολοσειρές v και x.

Ιδέα απόδειξης (4/5): Άντληση (pumping)

Επαναλαμβάνοντας την παραγωγή $N \Rightarrow^* vNx$ ακριβώς i φορές, έχουμε

$$N \Rightarrow^* v^i N x^i \Rightarrow^* v^i w x^i.$$

Επομένως:

$$\begin{aligned} i = 0 \quad &: \quad S \Rightarrow^* uNy \Rightarrow^* uwy, \\ i = 1 \quad &: \quad S \Rightarrow^* uNy \Rightarrow^* uvNxy \Rightarrow^* uvwxy, \\ i = 2 \quad &: \quad S \Rightarrow^* uNy \Rightarrow^* uvNxy \Rightarrow^* uvvNxxy \Rightarrow^* uvvwxxy. \end{aligned}$$

Γενική περίπτωση

Για κάθε ακέραιο $i \geq 0$,

$$S \Rightarrow^* uNy \Rightarrow^* uv^i N x^i y \Rightarrow^* uv^i w x^i y.$$

Άρα,

$$\boxed{uv^i w x^i y \in L, \qquad \forall i \geq 0.}$$

Ιδέα απόδειξης (5/5): Οι περιορισμοί μήκους

Γιατί ισχύει $|vx| > 0$;

Οι δύο εμφανίσεις του N είναι διαφορετικοί κόμβοι του δέντρου παραγωγής.
Για να μεταβούμε από την πρώτη στη δεύτερη, χρησιμοποιούμε τουλάχιστον έναν κανόνα της μορφής $A \to BC$.
Το παιδί που δεν ανήκει στο συγκεκριμένο μονοπάτι παράγει τουλάχιστον ένα τερματικό σύμβολο, το οποίο ανήκει είτε στο v είτε στο x.
Άρα $|v| + |x| > 0$.

Γιατί ισχύει $|vwx| \leq p$;

Επιλέξαμε τις δύο εμφανίσεις του N μεταξύ των τελευταίων $k+1$ μη τερματικών κόμβων ενός μακρύτερου μονοπατιού.
Επομένως, κάθε μονοπάτι του υποδέντρου με ρίζα την ανώτερη εμφάνιση του N περιέχει το πολύ $k+1$ μη τερματικούς κόμβους.
Άρα το υποδέντρο παράγει το πολύ $2^k = p$ τερματικά σύμβολα.

Συνολικά, αποδείξαμε και τις τρεις συνθήκες:

$$uv^i wx^i y \in L \ \ (\forall i \geq 0), \qquad |vx| > 0, \qquad |vwx| \leq p. \qquad \square$$

Η γλώσσα $\{0^n1^n2^n\}$: αρχή της απόδειξης

Το αποτέλεσμα που θα αποδείξουμε

Η γλώσσα

$$L = \{0^n1^n2^n \mid n \geq 0\}$$

δεν είναι γλώσσα χωρίς συμφραζόμενα.

Υποθέτουμε, για να καταλήξουμε σε αντίφαση, ότι η L είναι CFL.
Τότε το λήμμα άντλησης εγγυάται την ύπαρξη ενός ακεραίου $p \geq 1$, του μήκους άντλησης.

Επιλέγουμε τη λέξη

$$\boxed{s = 0^p1^p2^p.}$$

Η s ανήκει στη L: περιέχει ακριβώς p μηδενικά, p άσους και p σύμβολα 2, με αυτή τη σειρά.
Επιπλέον,

$$|s| = 3p \geq p,$$

οπότε μπορούμε να εφαρμόσουμε το λήμμα στη λέξη s.

Τι θα έπρεπε να ισχύει για τη λέξη s;

Σύμφωνα με το λήμμα, υπάρχει τουλάχιστον μία διαμέριση

$$s = uvxyz$$

με τις ακόλουθες ιδιότητες:

- $|vxy| \leq p$: το συνολικό μήκος των τριών μεσαίων τμημάτων είναι το πολύ p.
- $|v| + |y| \geq 1$: τουλάχιστον ένα από τα v, y δεν είναι κενό.

Επιπλέον, για αυτή τη διαμέριση πρέπει να ισχύει

$$uv^i xy^i z \in L \qquad \text{για κάθε ακέραιο } i \geq 0.$$

Θα χρησιμοποιήσουμε την τιμή $i = 0$

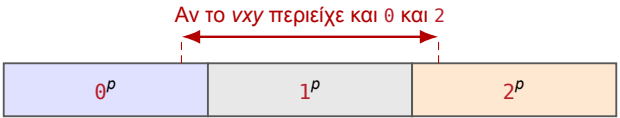
Τότε αφαιρούμε τα v, y και κρατάμε τα u, x, z:

$$uv^0 xy^0 z = uxz.$$

Θα δείξουμε ότι κάθε διαμέριση που ικανοποιεί τους δύο περιορισμούς μήκους δίνει $uxz \notin L$.

Πού μπορεί να βρίσκεται το *vxy*;

Το *vxy* είναι συνεχόμενο τμήμα της λέξης *s* και έχει μήκος το πολύ *p*.



Αν το *vxy* περιείχε ένα `0` και ένα `2`, θα περιείχε και όλους τους *p* άσους που βρίσκονται ανάμεσά τους. Θα είχαμε λοιπόν

$$|vxy| \geq 1 + p + 1 = p + 2 > p,$$

σε αντίφαση με τον περιορισμό $|vxy| \leq p$.

Επομένως, υπάρχουν μόνο πέντε δυνατότητες

- Το *vxy* περιέχει μόνο `0`, μόνο `1` ή μόνο `2`.
- Το *vxy* περιέχει `0` και `1`, ή `1` και `2`.

Περιπτώσεις 1–3: ένα μόνο είδος συμβόλου

Υποθέτουμε ότι το vxy βρίσκεται ολόκληρο μέσα σε μία από τις τρεις ομάδες συμβόλων.
Άρα και τα v, y περιέχουν μόνο αυτό το είδος συμβόλου.

Θέτουμε

$$d = |v| + |y|.$$

Από τους περιορισμούς του λήμματος γνωρίζουμε ότι

$$1 \leq d \leq |vxy| \leq p.$$

Με $i = 0$ αφαιρούμε ακριβώς d σύμβολα:

Το vxy περιέχει	Η λέξη που απομένει είναι
μόνο 0	$uxz = 0^{p-d}1^p2^p$
μόνο 1	$uxz = 0^p1^{p-d}2^p$
μόνο 2	$uxz = 0^p1^p2^{p-d}$

Σε κάθε περίπτωση, ένα πλήθος γίνεται $p - d < p$, ενώ τα άλλα δύο παραμένουν ίσα με p.

$$\boxed{uxz \notin L.}$$

Περίπτωση 4: το vxy περιέχει 0 και 1

Τώρα το vxy περνά από την ομάδα των μηδενικών στην ομάδα των άσων.
Τα v, y επομένως περιέχουν μόνο μηδενικά και άσους.

Έστω ότι στα v και y υπάρχουν συνολικά r μηδενικά και t άσοι. Τότε

$$r, t \geq 0, \qquad r + t = |v| + |y| \geq 1.$$

Μπορεί ένα από τα r, t να είναι μηδέν, αλλά δεν μπορούν να είναι και τα δύο μηδέν.

Με $i = 0$ αφαιρούμε αυτά τα σύμβολα και παίρνουμε

$$\boxed{uxz = 0^{p-r}1^{p-t}2^{p}.}$$

- Τα σύμβολα 2 παραμένουν ακριβώς p.
- Επειδή $r + t \geq 1$, τουλάχιστον ένα από τα πλήθη $p - r$ και $p - t$ είναι μικρότερο από p.

Άρα τα τρία πλήθη δεν είναι όλα ίσα, οπότε $uxz \notin L$.

Περίπτωση 5: το vxy περιέχει 1 και 2

Αυτή τη φορά το vxy περνά από την ομάδα των άσων στην ομάδα των συμβόλων 2.
Τα v, y επομένως περιέχουν μόνο αυτά τα δύο είδη συμβόλων.

Έστω ότι στα v και y υπάρχουν συνολικά r άσοι και t σύμβολα 2. Τότε

$$r, t \geq 0, \qquad r + t = |v| + |y| \geq 1.$$

Με $i = 0$ αφαιρούμε αυτά τα σύμβολα και παίρνουμε

$$\boxed{uxz = 0^p 1^{p-r} 2^{p-t}.}$$

- Τα μηδενικά παραμένουν ακριβώς p.
- Τουλάχιστον ένα από τα άλλα δύο πλήθη γίνεται μικρότερο από p, επειδή $r + t \geq 1$.

Επομένως, ούτε σε αυτή την περίπτωση τα τρία πλήθη παραμένουν όλα ίσα.

$$\boxed{uxz \notin L.}$$

Γιατί ολοκληρώθηκε η απόδειξη;

Ξεκινήσαμε με μια αυθαίρετη νόμιμη διαμέριση της ίδιας λέξης

$$s = 0^p 1^p 2^p.$$

Καλύψαμε όλες τις δυνατές θέσεις του *vxy*: τις τρεις περιπτώσεις με ένα είδος συμβόλου και τις δύο περιπτώσεις με δύο είδη.

Σε κάθε περίπτωση, η επιλογή $i = 0$ δίνει

$$uv^0xy^0z = uxz \notin L.$$

Η αντίφαση με το λήμμα

Το λήμμα εγγυάται ότι υπάρχει τουλάχιστον μία διαμέριση για την οποία

$$uv^ixy^iz \in L \qquad \text{για κάθε ακέραιο } i \geq 0.$$

Εμείς δείξαμε ότι κάθε νόμιμη διαμέριση αποτυγχάνει ήδη για $i = 0$. Άρα τέτοια διαμέριση δεν υπάρχει.

Η αρχική υπόθεση ότι η *L* είναι CFL ήταν λανθασμένη. Επομένως,

$$\boxed{L = \{0^n 1^n 2^n \mid n \geq 0\} \notin \text{CFL}.}$$

Το pumping lemma δεν αποδεικνύει ότι μία γλώσσα είναι CFL

Λογική της πρότασης

$$L \text{ CFL} \implies L \text{ έχει την pumping ιδιότητα.}$$

Η αντίστροφη συνεπαγωγή *δεν ισχύει γενικά*.

Για να αποδείξουμε ότι μία γλώσσα είναι CFL

Αρκεί να *κατασκευάσουμε CFG ή PDA* και να αποδείξουμε την ορθότητά του.

Για να αποδείξουμε ότι δεν είναι CFL

Βρίσκουμε λέξη που αποτυγχάνει *για κάθε νόμιμη διαμέριση* ή χρησιμοποιούμε άλλη αναγκαία ιδιότητα.

Δεύτερη μη-CFL: η γλώσσα αντιγραφής $\{ww\}$

$$COPY = \{ww \mid w \in \{0,1\}^*\}.$$

Τεχνική απλοποίησης

Θεωρούμε την κανονική γλώσσα $R = 0^*1^*0^*1^*$.

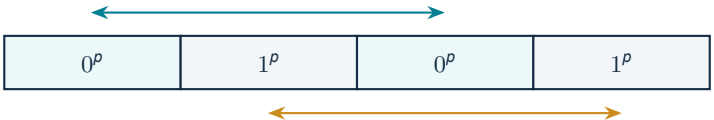
$$COPY \cap R = \{0^i1^j0^i1^j \mid i,j \geq 0\}.$$

Αν η *COPY* ήταν CFL, τότε και η τομή της με το R θα ήταν CFL.

Pumping για τη γλώσσα αντιγραφής: τέσσερα μπλοκ

Έστω $K = \{0^i 1^j 0^i 1^j \mid i, j \geq 0\}$, και διάλεξε

$$s = 0^p 1^p 0^p 1^p.$$



- *vxy* μήκους $\leq p$ αγγίζει το πολύ *δύο γειτονικά* μπλοκ.
- Θέτοντας $i = 0$, αλλάζει το πλήθος σε τουλάχιστον ένα από αυτά.
- Το *αντίστοιχο* μπλοκ στην άλλη μισή λέξη μένει αμετάβλητο.

Άρα η νέα λέξη δεν είναι δύο ίδια αντίγραφα. Συνεπώς *K* και *COPY* δεν είναι CFL.

Η αντιπαραβολή είναι ουσιώδης: palindrome vs copy

Γλώσσα	Κεντρικό εργαλείο	CFL;
$\{ww^R\}$	ένα stack με pop σε αντίστροφη σειρά	✓
$\{ww\}$	απαιτεί αντιγραφή στην ίδια σειρά	×
$\{0^n1^n\}$	μία μέτρηση	✓
$\{0^n1^n2^n\}$	τρεις ίσες μετρήσεις	×

Κρίσιμο δίδαγμα: ο αριθμός των μετρητών δεν αρκεί μόνος του ως κριτήριο· η *διάταξη και η εμφώλευση* των εξαρτήσεων έχουν σημασία.

Άσκηση: $\{0^n 1^m 2^n \mid m, n \geq 0\}$

Ερώτηση: είναι CFL ή όχι;

Πρώτη σκέψη

Έχουμε τρία μπλοκ. Αυτό *δεν* αρκεί για να ισχυριστούμε «μη-CFL».

Κατασκευή CFG

$$S \to 0S2 \mid T, \qquad T \to 1T \mid \varepsilon.$$

Απάντηση / συζήτηση: Είναι CFL. Το m είναι ανεξάρτητο· μόνο το πλήθος 0 πρέπει να ισούται με το πλήθος 2.