Lecture 7 – The CYK algorithm, Pushdown automata

NTIN071 Automata and Grammars

Jakub Bulín (KTIML MFF UK)

Spring 2024

** Adapted from the Czech-lecture slides by Marta Vomlelová with gratitude. The translation, some modifications, and all errors are mine.*

Recap of Lecture 6

- Reducing a grammar: removing ϵ-productions, unit productions, useless symbols
- Chomsky Normal Form of a context-free grammar
- Pumping lemma for context-free languages, application: proving non-context-freeness

2.8 The CYK algorithm

Testing membership in a context-free language

Given a context-free grammar G in Chomsky Normal Form and a word $w = a_1 \dots a_n \in T^*$, determine if $w \in L(G)$.

Naive, inefficient algorithm:

Construct all parse trees from G of appropriate depth ($\lceil log_2 |w| \rceil$), check if the yield is w.

The Cocke-Younger-Kasami algorithm:

Use dynamic programming to compute, for every $1 \leq i \leq j \leq n$, the set X_{ij} of all variables of G that generate the subword $a_i \dots a_j$.

Then check if $S \in X_{1n}$.

(Very efficient, worst-case time complexity $\mathcal{O}(n^3 |G|)$.)

The CYK algorithm

- **input:** $G = (V, T, \mathcal{P}, S)$ in ChNF, $w = a_1 \ldots a_n \in T^*$
- **decide:** $w \in L(G)$?

Compute for $1 \leq i \leq j \leq n$:

$$X_{ij} = \{A \in V \mid A \Rightarrow^* a_i a_{i+1} \ldots a_j\}$$

using dynamic programming
(storing results in a table)

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

1. **Initialize:** $X_{ii} = \{A \in V \mid A \rightarrow a_i \in \mathcal{P}\}$
2. **Fill upwards:**

$$X_{ij} = \{A \in V \mid A \rightarrow BC \in \mathcal{P}, B \in X_{ik}, C \in X_{k+1,j}\}$$

3. **Check:** Is $S \in X_{1n}$?

The CYK algorithm: an example

Example

$G = (\{S, A, B, C\}, \{a, b\}, \mathcal{P}, S)$ with

$\mathcal{P} = \{S \to AB \mid BC, A \to BA \mid a, B \to CC \mid b, C \to AB \mid a\}$

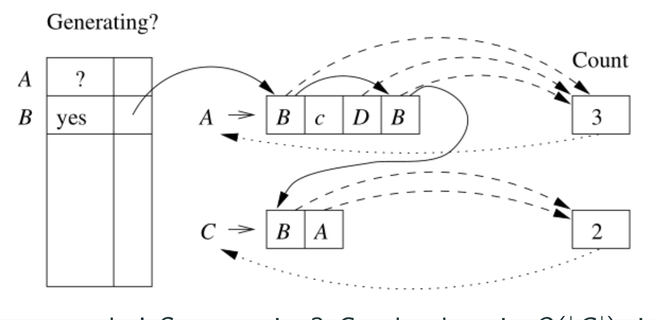
Rules reversed:

$$\begin{array}{lll} AB \leftarrow \{S, C\} & BC \leftarrow \{S\} & b \leftarrow \{B\} \\ BA \leftarrow \{A\} & CC \leftarrow \{B\} & a \leftarrow \{A, C\} \end{array}$$

Fill upwards:

$\{S, A, C\}$				
-	$\{S, A, C\}$			
-	$\{B\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

Testing emptiness of a context-free language



Is the start symbol S generating? Can be done in $O(|G|)$ time.

- For each variable a chain of all body positions where it appears
- For each production two-way link to a count of body positions with variables that have not yet been marked as generating

Once we mark a variable as generating, follow the chain and decrease counts by 1. If a count reaches 0, mark the head as generating. Process all generating variables using a stack.

Testing finiteness of a context-free language

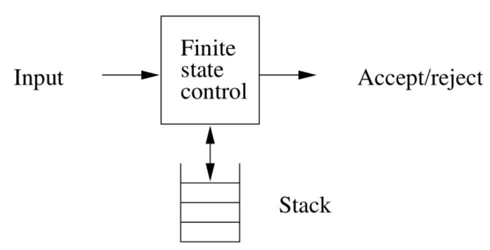
Let G be a Chomsky normal form grammar for L, i.e. $L \setminus \{\lambda\} = L(G)$. Construct the following oriented graph:

- nodes: variables in G
- edges: $\{(A, B), (A, C) \mid A \to BC \text{ is a production rule in } G\}$

Now L is infinite, if and only if the graph contains an oriented cycle. Can be tested in $O(|G|)$.

2.9 Pushdown automata

Pushdown automaton (PDA)



- an extension of ϵ–NFA, additional feature: a stack memory
- the stack has its own stack alphabet Γ (can contain Σ or not)
- at each step we pop the top stack symbol X, make a decision based on (q, a, X), push some word $\gamma \in \gamma^*$
- the stack can rememeber an infinite amount of information
- PDA define context-free languages, nondeterminism is important: deterministic PDA only recognize a proper subset of context-free languages (unlike DFA vs. NFA)

The definition

A pushdown automaton (PDA): $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

- Q is finite, nonempty set of states
- Σ is a finite, nonempty input alphabet
- Γ is a finite, nonempty stack alphabet
- δ is the transition function,

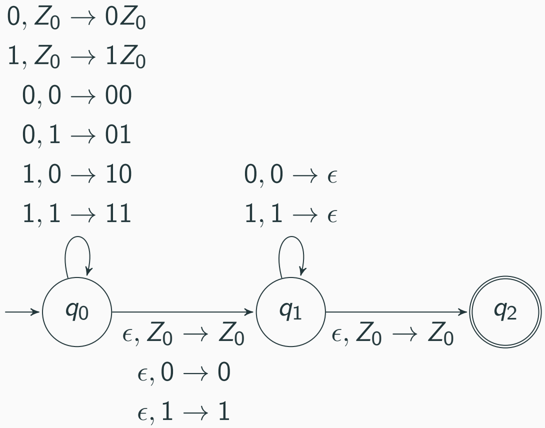
  $$\delta\colon Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}_{FIN}(Q \times \Gamma^*)$$

  $\delta(q, a, X) \ni (p, \gamma)$ where p is the new state and γ a finite string of stack symbols that replace X on top of the stack
- $q_0 \in Q$ is the initial state
- $Z_0 \in \Gamma$ is the initial stack symbol (bottom of the stack); the only symbol on the stack at the beginning
- F is a set of accepting (final) states; may be undefined if our PDA accepts by empty stack

One transition of a PDA

- read one input letter ($a \in \Sigma$) or do an ϵ-transition ($a = \epsilon$)
- pop X from the top of the stack
- based on a, X, and the current state q nondeterministically choose one of finitely many options $(p, \gamma) \in \delta(q, a, X)$
- switch to the new state p
- push the finite string γ to the stack (the first symbol of Γ is now on top)
- pop: $\gamma = \epsilon$, read only: $\gamma = X$, push: $\gamma = \gamma' X$

Example: $L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$



q_0 read input letters pushing them onto the stack; guess the middle (nondeterministically), jump to q_1

q_1 compare input with stack, consuming both; if empty stack (we see the bottom), accept by jumping to q_2; no input can remain

Example cont'd: full description of the PDA

$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$ $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$	push input onto stack, leave the bottom
$\delta(q_0, 0, 0) = \{(q_0, 00)\}$ $\delta(q_0, 0, 1) = \{(q_0, 01)\}$ $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ $\delta(q_0, 1, 1) = \{(q_0, 11)\}$	stay in q_0, push input onto stack
$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)\}$ $\delta(q_0, \epsilon, 0) = \{(q_1, 0)\}$ $\delta(q_0, \epsilon, 1) = \{(q_1, 1)\}$	jump to q_1 without changing stack
$\delta(q_1, 0, 0) = \{(q_1, \epsilon)\}$ $\delta(q_1, 1, 1) = \{(q_1, \epsilon)\}$	pop stack and match with input
$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$	we have ww^R, go to accepting state

Notation

a, b, c	symbols of the input alphabet
q, p, r	states
u, w, x, y, z	words over input alphabet
X, Y, A, B	stack symbols
Z_0	bottom of the stack symbol
α, β, γ	words over stack alphabet

Transition diagram:

- nodes are states, initial and final denoted as usual
- a transition $\delta(q, a, X) \ni (p, \alpha)$: arc from p to q labelled $a, X \to \alpha$

The languages of a PDA

Configurations and moves (computation graph)

A configuration of a PDA is a triple (q, w, γ), where

q is the current state

w is the remaining input and

γ is the stack contents (the top is on the left)

We define moves between configurations ($\vdash_P$ or $\vdash$) thus: for any transition $\delta(q, a, X) \ni (p, \alpha)$ and all $w \in \Sigma^*$ and $\beta \in \Gamma^*$ we have

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

We use the symbol $\vdash_P^*$ or $\vdash^*$ to represent zero or more moves, i.e.

- $I \vdash^* I$ for any configuration I
- $I \vdash^* J$ if there exists K such that $I \vdash K$ and $K \vdash^* J$

Initial and accepting configurations, the languages of a PDA

The initial configuration of $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ for input word $w \in \Sigma^*$ is (q_0, w, Z_0). Which configurations are accepting?

Two options:

1. Acceptance by final state: (f, ϵ, γ) for some final state $f \in F$ and arbitrary stack contents $\gamma \in \Gamma^*$

$L(P) = \{w \mid (q_0, w, Z_0) \vdash_P^* (f, \epsilon, \gamma)$ for some $f \in F$ and $\gamma \in \Gamma^*\}$

2. Acceptance by empty stack: (q, ϵ, ϵ) for an arbitrary $q \in Q$

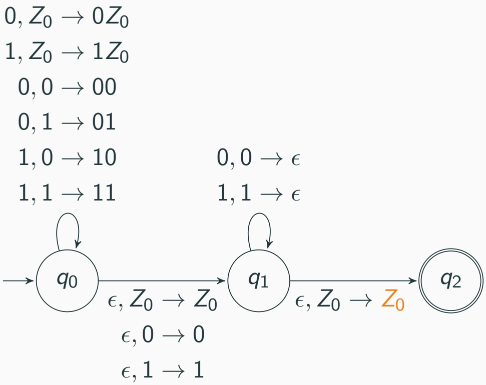
$N(P) = \{w \mid (q_0, w, Z_0) \vdash_P^* (q, \epsilon, \epsilon)$ for any $q \in Q\}$

In this case we can write only $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

Configurations for the input $w = 1111$

$$
\begin{tikzcd}[column sep=small]
 & & & & \boxed{(q_0, 1111, Z_0)} \arrow[dl] \arrow[dr] & \\
 & & & (q_0, 111, 1Z_0) \arrow[dl] \arrow[dr] & & (q_1, 1111, Z_0) \arrow[d] \\
 & & (q_0, 11, 11Z_0) \arrow[dl] \arrow[dr] & & (q_1, 111, 1Z_0) \arrow[d] & (q_2, 1111, Z_0) \\
 & (q_0, 1, 111Z_0) \arrow[dl] \arrow[dr] & & (q_1, 11, 11Z_0) \arrow[d] & (q_1, 11, Z_0) \arrow[d] & \\
(q_0, \epsilon, 1111Z_0) \arrow[d] & & (q_1, 1, 111Z_0) \arrow[d] & (q_1, 1, 1Z_0) \arrow[d] & (q_2, 11, Z_0) & \\
(q_1, \epsilon, 1111Z_0) & & (q_1, \epsilon, 11Z_0) & (q_1, \epsilon, Z_0) \arrow[d] & & \\
 & & & \boxed{(q_2, \epsilon, Z_0)} & &
\end{tikzcd}
$$

Our example

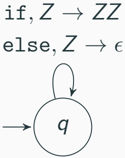


- acceptance by final state: $L(P) = L_{wwr}$
- to accept by empty stack: modify $\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$ to $\delta(q_1, \epsilon, Z_0) = \{(q_2, \epsilon)\}$ (erase bottom of the stack symbol), then also $N(P') = L_{wwr}$

Another example: if-else

Stop (accept) at first error, e.g. more `else`'s than `if`'s

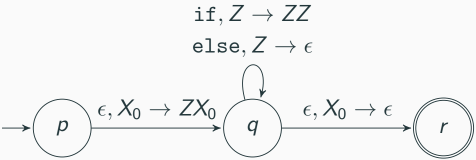
By empty stack: $P_N = (\{q\}, \{\mathtt{if}, \mathtt{else}\}, \{Z\}, \delta_N, q, Z)$



$$\delta_N(q, \mathtt{if}, Z) = \{(q, ZZ)\} \quad \text{(push)}$$

$$\delta_N(q, \mathtt{else}, Z) = \{(q, \epsilon)\} \quad \text{(pop)}$$

By final state: $P_F = (\{p, q, r\}, \{\mathtt{if}, \mathtt{else}\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\})$



$$\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\} \quad \text{(start)}$$

$$\delta_F(q, \mathtt{if}, Z) = \{(q, ZZ)\} \quad \text{(push)}$$

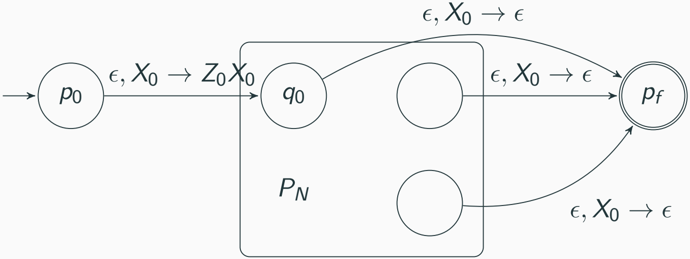
$$\delta_F(q, \mathtt{else}, Z) = \{(q, \epsilon)\} \quad \text{(pop)}$$

$$\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\} \quad \text{(accept)}$$

From empty stack to final state

Lemma

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F such that $L = L(P_F)$.



Idea: Make Z_0 a fake bottom (insert a new bottom X_0 below), so that we can tell when P_N's stack was empty. Add ϵ-transitions upon seeing X_0 from all states to a new, accepting state.

The proof

$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_F\})$, where δ_F is

- $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$ (start).
- $\forall(q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)$, $\delta_F(q, a, Y) = \delta_N(q, a, Y)$.
- In addition, $\delta_F(q, \epsilon, X_0) \ni (p_f, \epsilon)$ for every $q \in Q$.

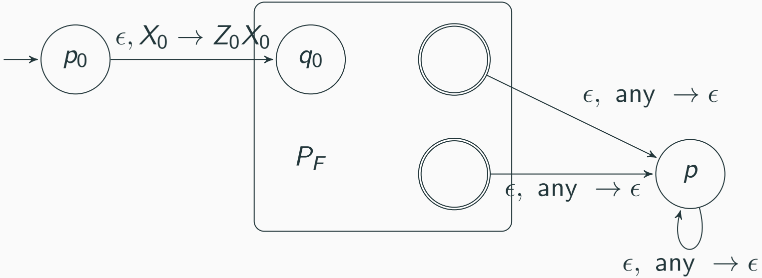
We must show that $w \in L(P_N)$ iff $w \in L(P_F)$.

- (If) P_F accepts as follows:
  $(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0 X_0) \vdash^*_{P_F = N_F} (q, \epsilon, X_0) \vdash_{P_F} (p_f, \epsilon, \epsilon)$.
- (Only if) No other way to go to p_F than the above. $\square$

From final state to empty stack

Lemma

If $L = L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, then there exists a PDA P_N such that $L = N(P_N)$.



Idea: Make Z_0 a fake bottom (insert a new bottom below), because P_F could accidentally empty stack in a nonfinal state. Add ϵ-transitions (upon any stack symbol) from final states to a new state, there empty the stack without reading any input symbols.

The proof

Let $P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$, where

- $\delta_N(p_0, \epsilon, X_0) = \{(q, Z_0 X_0)\}$ (start)
- $\forall (q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma)\ \delta_N(q, a, Y) = \delta_F(q, a, Y)$ (simulate)
- $\forall (q \in F, Y \in \Gamma \cup \{X_0\})$, $\delta_N(q, \epsilon, Y) \ni (p, \epsilon)$ (i.e. accept if P_F accepts)
- $\forall (Y \in \Gamma \cup \{X_0\}), \delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$ clean the stack.

The proof $w \in N(P_N)$ iff $w \in L(P_F)$ is similar as before. $\square$

Unseen data cannot affect computation

Lemma

If $(q, x, \alpha) \vdash_P (p, y, \beta)$*, then for any* $w \in \Sigma^*$ *and* $\gamma \in \Gamma^*$ *we also have* $(q, xw, \alpha\gamma) \vdash_P^* (p, yw, \beta\gamma)$*. (In particular,* $\gamma = \epsilon$ *or* $w = \epsilon$*.)*

Proof: Induction on the number length of the sequence of configurations that take $(q, xw, \alpha\gamma)$ to $(p, yw, \beta\gamma)$. Each of the moves $(q, x, \alpha) \vdash_P^* (p, y, \beta)$ is justified without using w and/or γ in any way. The moves are still valid with w, γ on the input/stack. $\square$

Lemma

If $(q, xw, \alpha) \vdash_P^* (p, yw, \beta)$*, then also* $(q, x, \alpha) \vdash_P^* (p, y, \beta)$*.*

NB: Not true for stack, the computation may require γ on the stack and then push it back. (E.g. $L = \{0^i 1^i 0^j 1^j\}$, configuration $(p, 0^{i-j} 1^i 0^j 1^j, 0^j Z_0) \vdash^* (q, 1^j, 0^j Z_0)$, inbetween clear the stack.)

Summary of Lecture 7

- Testing membership in a context-free language: the Cocke-Younger-Kasami algorithm
- Testing emptiness and finiteness of a context-free language
- Pushdown automaton: extend an ϵ-NFA with a stack memory (potentially infinite), pop the top symbol, decide based on (q, a, X), can push a finite string of stack symbols
- Acceptance by final state $L(P)$ and by empty stack $N(P)$, conversion between the two options