10. Pushdown Automata and Context-free Languages

Goal: Develop an automata-theoretic understanding of context-free languages.

Motivation:
- Will make it easier to show some closure properties
- Needed for parsing.

10.1 Pushdown Automata

Goal:
- Introduce a new computational model: finite automata that have access to a stack.
- Can store an unbounded amount of information about the history of the computation, but access this information only in a very restricted way (last-in, first-out).

Definition (Syntax of a pushdown automaton):
A (non-deterministic) pushdown automaton ((N)PDA) is a tuple $M = (Q, \Sigma, \Gamma, q_0, \delta, Q_F)$ with
- Q a finite set of states (also called control states),
- $q_0 \in Q$ the initial state, and $Q_F \subseteq Q$ the set of final states,
- Σ a (finite) input alphabet (alphabets are always finite),
- Γ a (finite) stack alphabet,
- $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma^* \times Q$ a finite set of transitions.

To define the behavior of a PDA, its semantics, we need the notion of a configuration, the stack of the PDA at runtime.

The configuration should contain
- the current control state and
- the current stack content.

Definition (Semantics of a pushdown automaton):
Let $M = (Q, \Sigma, \Gamma, q_0, \delta, Q_F)$ be a PDA.

- The set of configurations of M is $Q \times \Gamma^*$.
  The initial configuration is (q_0, ε).
  A configuration is final (or accepting) if it is of the form $(q_f, \omega) \in Q_F \times \Gamma^*$.
- The labelled transition relation among configurations

$$\longrightarrow_M \subseteq (Q \times \Gamma^*) \times \Sigma \times (Q \times \Gamma^*)$$

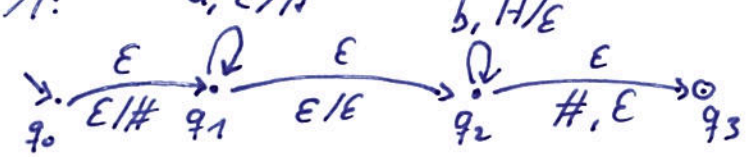
is defined by
$(q_1, \omega_1) \xrightarrow{a}_M (q_2, \omega_2)$, if $q_1 \xrightarrow[v_1/v_2]{a} q_2 \in \delta$ (this is the tuple (q_1, a, v_1, v_2, q_2))
and $\omega_1 = \omega . v_1$
and $\omega_2 = \omega . v_2$ for some $\omega \in \Gamma^*$.

We generalize the labelled transition relation to words from Σ^* in the expected way.

- The language of M is
$$L(M) := \{ w \in \Sigma^* \mid (q_0, \varepsilon) \xrightarrow{w} (q_f, u) \in Q_F \times \Gamma^* \}.$$

Example:

M:



$L(M) = \{ a^n b^n \mid n \in \mathbb{N} \}$.

Note:
If PDA M induces an infinite-state automaton

$$\mathcal{A}_M := (Q \times \Gamma^*, \Sigma, (q_0, \varepsilon), \rightarrow_M, Q_F \times \Gamma^*)$$

and $L(M) = L(\mathcal{A}_M)$
(where the language of an infinite-state automaton is defined like for NFAs).

Remark:
There are alternative definitions of PDAs in the literature.
(1) A common alternative is to define acceptance
by reaching the empty stack.
In this case, the PDA is a tuple $M = (Q, \Sigma, \Gamma, q_0, \#, \delta)$
with $\# \in \Gamma$ a symbol that is on the stack initially.
(2) In both models (acceptance via final states and via empty stack),
we may assume that transitions
- always pop an element and
- push at most two elements.

For the former restriction, we have to require
to start with an initial stack symbol
also when accepting with final states.

Theorem:
The alternative definitions accept the same languages as PDAs.

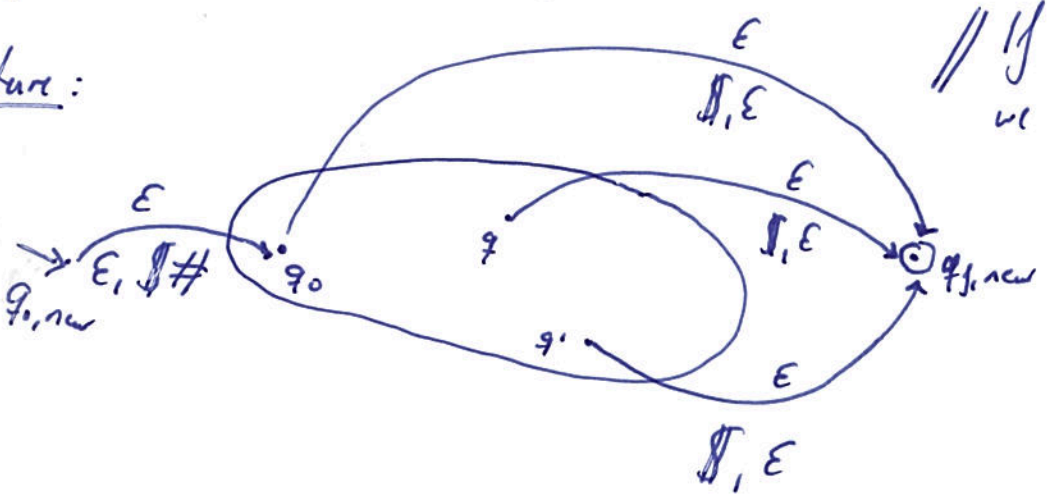
Proof (Sketch):
- To mimic empty stack acceptance with our definition,
- let $\#$ be the distinguished initial stack symbol.

We introduce
- a new initial state,
- a new final state, and
- a new bottom of stack symbol $\$$.

We introduce transitions that
- initially establish the bottom of stack $\$$
- the symbol is never removed except in the last step that takes the PDA from any state to the new final state.

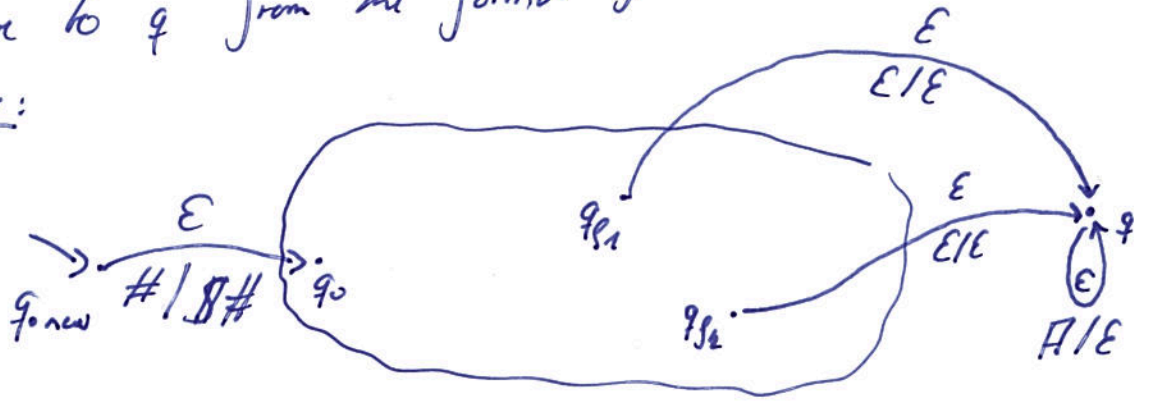
// If we can read $\$$, we are sure the stack is empty.

Picture:



- To mimic acceptance with a final state via empty stack, add a new bottom of stack symbol $\$$ so that the stack is not accidentally emptied. Moreover, add a new state q in which to empty the stack. Move to q from the former final states.

Picture:



- The remaining equivalences can be solved as homework. □

10.2 Equivalence with Context-free Languages

Goal: Show that the languages accepted by PDTAs are precisely (coincide with) the context-free languages.

We establish the two directions separately.

Theorem:
If L is context-free, then there is a PDTA A with $L = L(A)$.

Idea:
- Mimic the left-derivation relation $\Rightarrow_L$ using the stack.
- We give an elegant proof that makes use of Greibach's normal form.
- The normalization step can be avoided.

Proof:
Let $L = L(G)$ with $G = (N, \Sigma, P, S)$ a context-free grammar.
Assume the grammar is in Greibach normal form so that the productions are

$$A \to a B_1 \dots B_k \quad \text{and potentially } S \to \varepsilon.$$

We construct an equivalent PDTA A that accepts with empty stack.
Interestingly, A has only one stack.
The definition is

$$A := (\{q\}, \Sigma, N, q, \underbrace{S}_{\text{initial stack content}}, \delta)$$

(N: stack alphabet)

For every production $A \to a B_1 \dots B_k$ there is a transition

$$q \xrightarrow{a} q \quad A / B_k \dots B_1.$$

So we read letter a from the input,
read A from the top-of-stack, and
write $B_n \dots B_1$ to the top-of-stack.

Similarly, $S \to \varepsilon$ yields $q \xrightarrow[S/\varepsilon]{\varepsilon} q$.

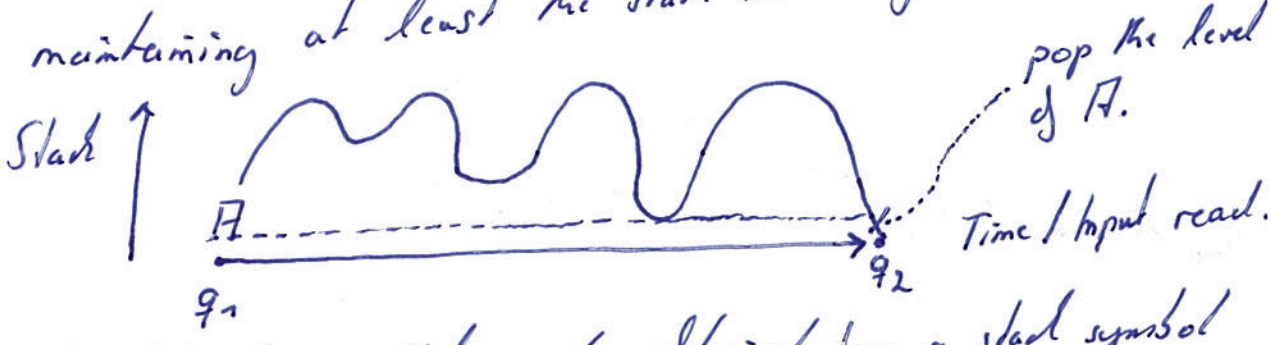
Claim: $L(M) = L(G)$. □

Discussion:

- One can give a similar construction
that does not rely on Greibach normal form.
Doing this is a good exercise.
- The benefit of the above construction is that
we definitely read a letter from the input word with every step.
So we can decide in NP whether a given word is accepted.

Theorem:
Every language $L(M)$ of a PDA M is context-free.

Idea: - We guess the changes in the control state of the PDA,
say from q_1 to q_2 (potentially far away),
that can be obtained from a stack symbol A
while maintaining at least the stack level of A.



- Guessing the stack changes that can be obtained from a stack symbol
means we use non-terminals of the form (q_1, A, q_2).

Proof:

Consider $L(M)$ with $M = (Q, \Sigma, \Gamma, q_0, \#, \delta)$ accepting with empty stack.

We assume $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma^* \times Q$,

so we definitely read a stack symbol in every transition.

We define the grammar to be

$$G := ((Q \times \Gamma \times Q) \cup \{S\}, \Sigma, P, S)$$

with

$$P := \{ S \to (q_0, \#, q) \mid q \in Q \}$$

$$\cup \{ (q, A, q') \to a \,.\, (q'', B_m, q_1)\ldots(q_{m-1}, B_1, q') \mid$$

$q_1, \ldots, q_{m-1} \in Q$ (m may be 0) and

(*) $q \xrightarrow[A / B_1 \ldots B_m]{a} q''$ with $a \in \Sigma \cup \{\varepsilon\}\}$. □

Claim: $L(M) = L(G)$.

Putting the results together, we obtain an interesting observation about the role of control states in pushdown automata.

Corollary:

For every PDA M, there is a PDA M' that only has one control state and satisfies $L(M) = L(M')$.

(*) Rephrasing the idea of the construction:

Non-terminal (q_1, A, q_2) generates $w \in \Sigma^*$ iff upon input w and with initial stack symbol A, M moves from q_1 to q_2 and ends with an empty stack.