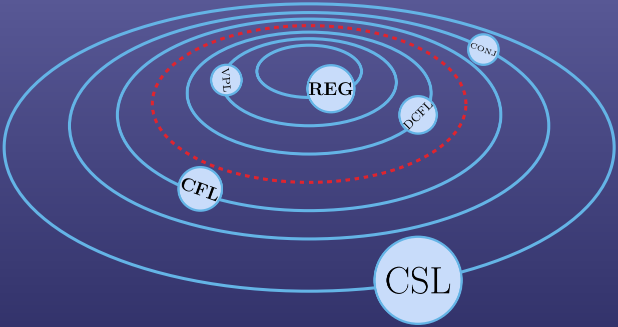
Bauman Moscow State University
Th. Computer Science Dept.

Pushdown Machines: Visible and Not



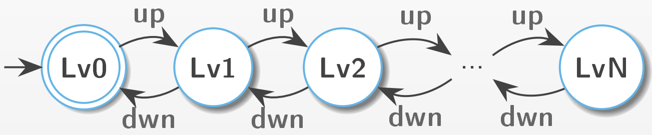
Antonina Nepeivoda
a_nevod@mail.ru

Lecture Outline

Finite Automata is Enough?

Real-world machines are finite. Do finite models suffice?

Recall "elevator automaton" with a unique final state on the "ground floor", breaking if asked to reach an non-existing floor:

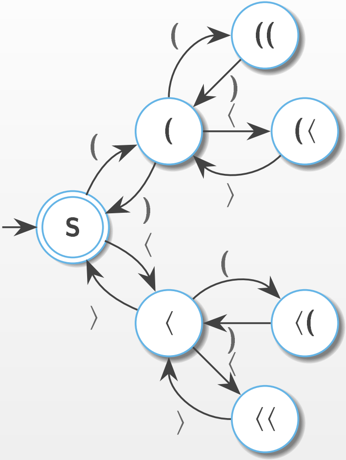


"**up**" and "**dwn**" instructions can be interpreted wrt a parentheses structure. That is, parsing string "((" we move to **Lv2**, and "(()())" returns us to **Lv1**.

Real-world nesting depth is limited (even in Lisp-like languages), and linear blow-up in state size seems satisfactory.

Until we decide to use several sorts of brackets...

Myhill–Nerode Congruence for Many-Sorted Brackets



Congruence Table

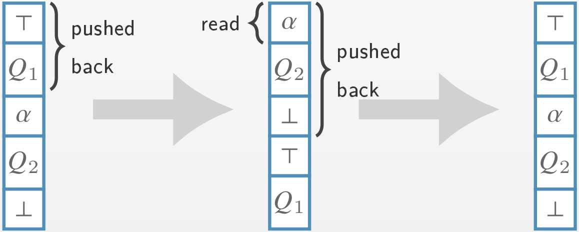
	ε	$)$	$\rangle$	$))$	$\rangle)$	$)\rangle$	$\rangle\rangle$
ε	+	−	−	−	−	−	−
$($	−	+	−	−	−	−	−
$\langle$	−	−	+	−	−	−	−
$(($	−	−	−	+	−	−	−
$(\langle$	−	−	−	−	+	−	−
$\langle($	−	−	−	−	−	+	−
$\langle\langle$	−	−	−	−	−	−	+

- N-depth balanced sequences of 2 sorts of brackets $\Rightarrow$ $2^{N+1}-1$ states in a min NFA.
- N-depth balanced sequences of K sorts of brackets $\Rightarrow$ $\frac{K^{N+1}-1}{K-1}$ states in a min NFA.

Finite automata cannot track nested structures efficiently.

Memoising Counters via Additional Memory

- Queue as a memory — can be considered as an additional tape with the write access, since it can be "re-rolled" to any wanted position with no memory loss.



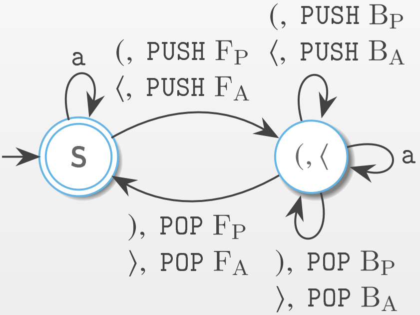
- Stack as a memory — information given in Q_1 cannot be stored except in states when α is read. More restrictive, natural for tracking nested structures.

Natural Idea: Call–Return Counters

Input alphabet Σ is split into disjoint union $\Sigma_I \cup \Sigma_C \cup \Sigma_R$, where:
Σ_I is internal alphabet (symbols not affecting the stack),
Σ_C is call alphabet (symbols that push on the stack),
Σ_R is return alphabet (symbols that pop from stack).

Balanced parentheses language:

- Σ_I — all non-brackets;
- Σ_C — all opening brackets;
- Σ_R — all closing brackets.



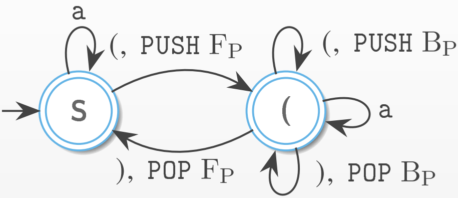
With several sorts of brackets:

- state space does not increase;
- transitions set grows linearly.

For simplicity, henceforth we usually use an unique sort of brackets.

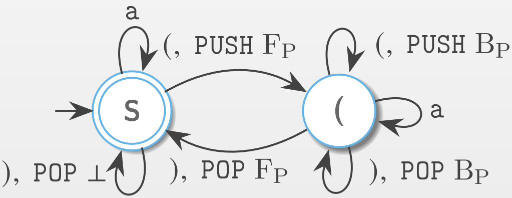
Handling Imbalance

Language of valid prefixes of balanced parentheses?



The final configuration can admit non-empty stack.

Language of valid suffixes of balanced parentheses?



Return symbols can be safely popped from the empty stack ($\perp$).

Visibly Pushdown Automaton

👀 Definition

A VPA is a tuple $\mathcal{A} = \langle Q, \Sigma_C \cup \Sigma_R \cup \Sigma_I, \Gamma, \delta, q_0, F \rangle$, *where:*

- Q *is a finite set of states,* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of final states.*
- Γ *is a finite stack alphabet, containing bottom-of-stack symbol* $\perp$,
- δ *is the transitions set, consisting of the three subsets:*
  - *call transitions:* $\delta_C \subseteq Q \times \Sigma_C \times (\Gamma \setminus \{\perp\}) \times Q$,
  - *return transitions:* $\delta_R \subseteq Q \times \Sigma_R \times \Gamma \times Q$,
  - *internal transitions:* $\delta_I \subseteq Q \times \Sigma_I \times Q$.

The subsets are disjoint, since alphabets Σ_C, Σ_R, Σ_I are disjoint.
The transitions depend on, and change, the stack, and VPA configuration is

$$\Big\langle \overbrace{\ q_i\ }^{\text{current state}}, \underbrace{a_k \ldots a_n}_{\text{suffix to parse}}, \overbrace{\gamma_m \ldots \gamma_1 \perp}^{\text{current stack}} \Big\rangle$$

Visibly Pushdown Automaton

The transitions depend on, and change, the stack, and VPA configuration is

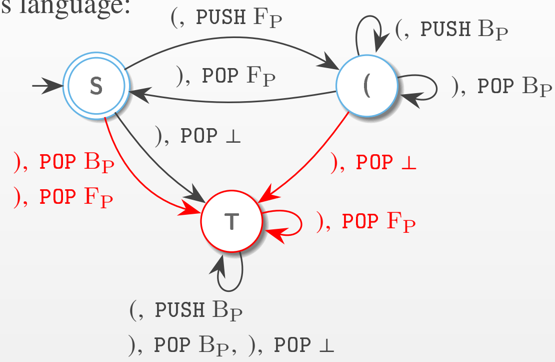
$$\left\langle \overbrace{\;q_i\;}^{\text{current state}}, \underbrace{a_k \ldots a_n}_{\text{suffix to parse}}, \overbrace{\gamma_m \ldots \gamma_1 \bot}^{\text{current stack}} \right\rangle$$

Given such a configuration, next configurations are defined as follows:

- if $a_k \in \Sigma_C$ and $\langle q_i, a_k, \gamma_{m+1}, q_j \rangle \in \delta$, then $\langle a_{k+1} \ldots a_n, \gamma_{m+1}\gamma_m \ldots \gamma_1 \bot \rangle$. Note that the stack bottom $\bot$ can never be pushed.
- if $a_k \in \Sigma_I$ and $\langle q_i, a_k, q_j \rangle \in \delta$, then $\langle a_{k+1} \ldots a_n, q_j, \gamma_m \ldots \gamma_1 \bot \rangle$. ε-transitions are forbidden, even for the internal case.
- if $a_k \in \Sigma_R$, and $m > 0$, and $\langle q_i, a_k, \gamma_m, q_j \rangle \in \delta$, then $\langle a_{k+1} \ldots a_n, q_j, \gamma_{m-1} \ldots \gamma_1 \bot \rangle$.
- if $a_k \in \Sigma_R$, and $m = 0$, and $\langle q_i, a_k, \bot, q_j \rangle \in \delta$, then $\langle a_{k+1} \ldots a_n, q_j, \bot \rangle$. I.e. the stack bottom is a fixed point of `POP` operation.

VPA for Dyck's Language, Revisited

Suppose we want to add the trap state to the VPA for balanced parentheses language:



Formally, we are required to add all combinations of stack symbols on the stack top and symbols from Σ_R (i.e., the closing parenthesis). But the transitions given in red are actually unreachable.

By what means we can get rid of them?

Balanced Words

Definition

Given a word $\omega \in (\Sigma_C \cup \Sigma_R \cup \Sigma_I)^*$, ω is said to be a balanced word iff $h(\omega)$ is a balanced parentheses sequence, given the following morphism h:

- $\forall a \in \Sigma_C (h(a) = \texttt{(}\,)$;
- $\forall a \in \Sigma_R (h(a) = \texttt{)}\,)$;
- $\forall a \in \Sigma_I (h(a) = \varepsilon)$.

Hence, every balanced word ω acts on any given configuration as if $\omega \in \Sigma_I^*$.