Theory of Formal Languages and Automata

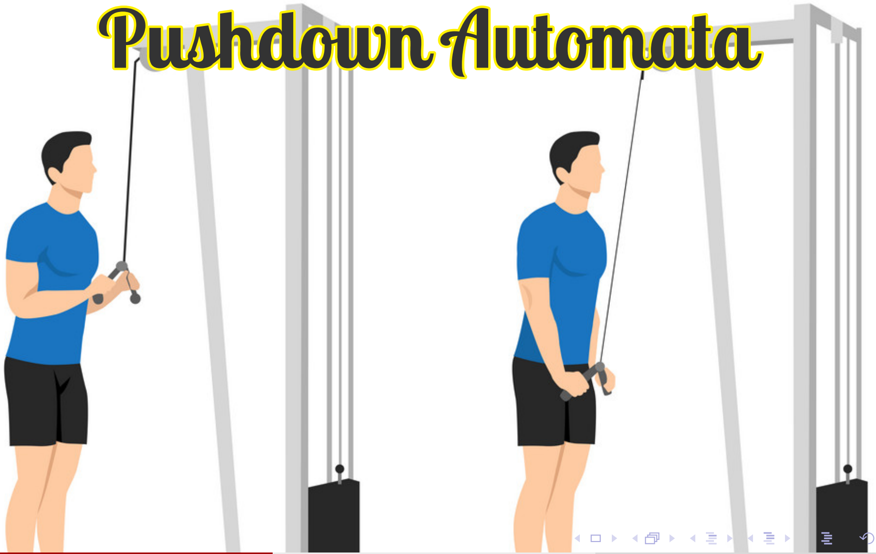
Lecture 12

Mahdi Dolati

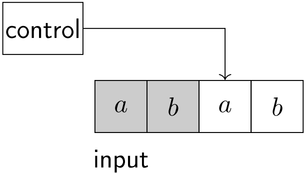
Sharif University of Technology

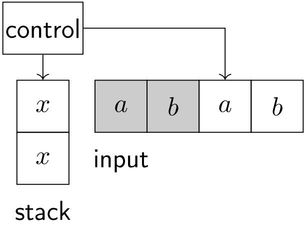
Fall 2023

November 27, 2023


Pushdown Automata

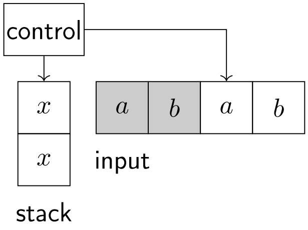
- Pushdown automata (PDA): A new computational model similar to NFA but have a stack for additional memory
- Pushdown automata are equivalent in power to CFG
    - Two methods to prove a language is CF: recognize it or generate it





Pushdown Automata

- Control: States and transitions,
- Input: contains a string,
- Arrow: Next symbol to be read,
- Write a symbol on the stack and read them later:
    - Push,
    - Pop.
- The stack can hold an unlimited number of symbols.



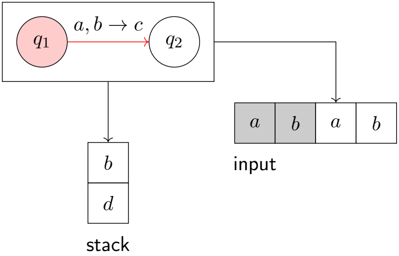
Pushdown Automata

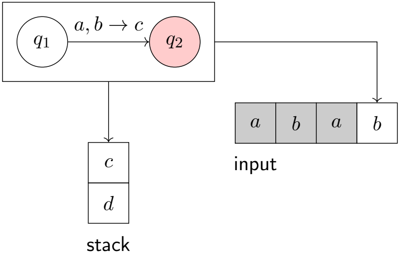
$$q_1 \xrightarrow{a, b \to c} q_2$$

- Read symbol is a,
- Pop symbol b off the stack, and
- Push symbol c onto the stack.

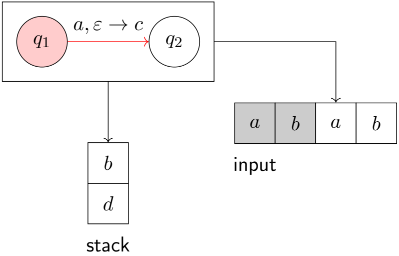
Pushdown Automata

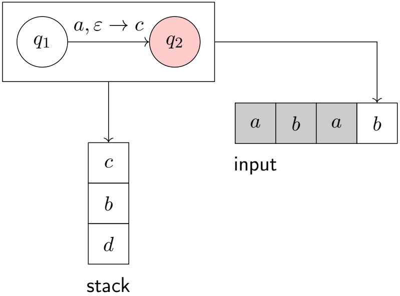
- Replace:





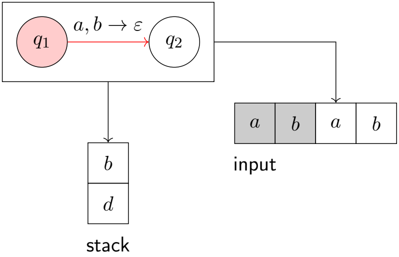
- Push:

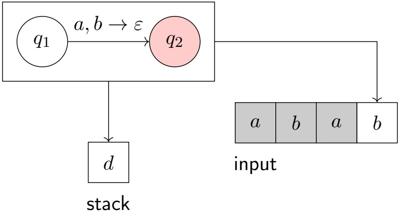

$a, \varepsilon \to c$
q_1
q_2
a b a b
input
b
d
stack


$a, \varepsilon \to c$
q_1
q_2
a b a b
input
c
b
d
stack

Pushdown Automata

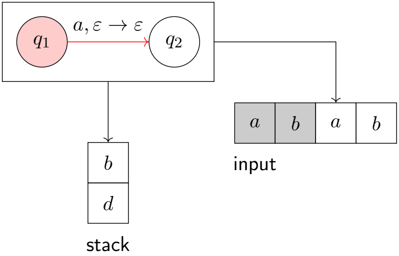
- Pop:

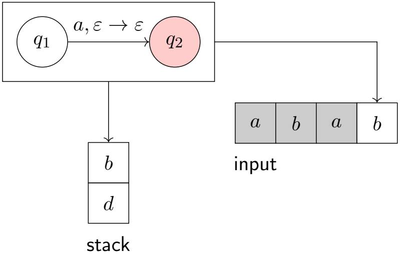




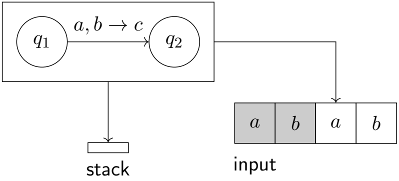
Pushdown Automata

- No change:





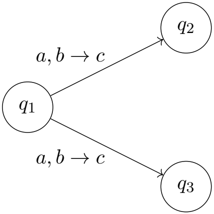
- Pop from an empty stack:

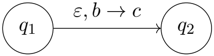


Pushdown Automata

- PDAs may be nondeterministic,
- In terms of power:

$$\text{Deterministic PDA} < \text{Nondeterministic PDA},$$





Definition (PDA)

A PDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, and F are finite sets, and

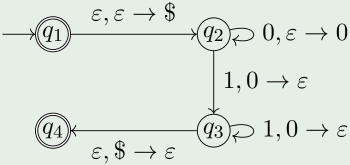
1. Set of states: Q,
2. Input alphabet: Σ,
3. Stack alphabet: Γ,
4. Transition function: $\delta : \; Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$,
5. Start state: $q_0 \in Q$, and
6. Set of accept states: $F \subseteq Q$,

Automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts w if:

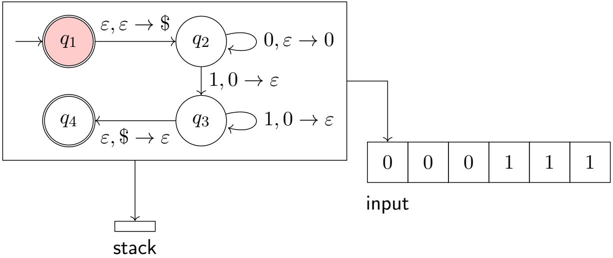
- Can write $w = w_1 w_2 \dots w_m$, where $w_i \in \Sigma_\varepsilon$,
- There exists sequence of states $r_0, r_1, \dots, r_m \in Q$,
- There exists strings (stack content) $s_0, s_1, \dots, s_m \in \Gamma^*$, such that,
  1. $r_0 = q_0$ and $s_0 = \varepsilon$,
  2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$,
     - $s_i = at$
     - $s_{i+1} = bt$
     - $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
  3. $r_m \in F$.

Example

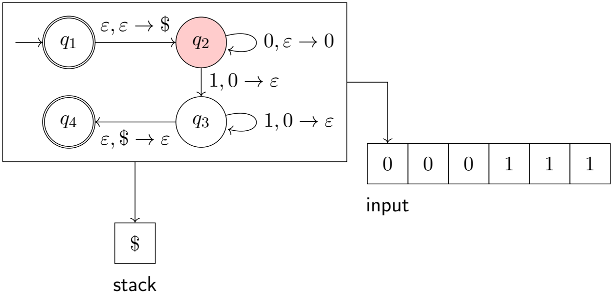
1. Read symbols from the input:
   1. Push each zero onto the stack until zero is read,
   2. Upon reading a one, pop a zero off the stack for each one,
   3. If a zero is read: Reject.
   4. If input is finished and stack is empty: Accept.
   5. If input is finished and stack is not empty: Reject.
   6. If stack becomes empty and input is not finished: Reject.



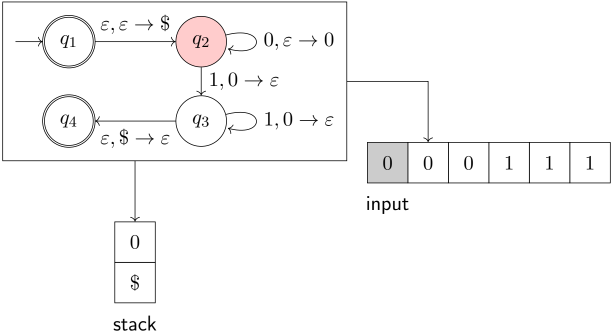
Pushdown Automata



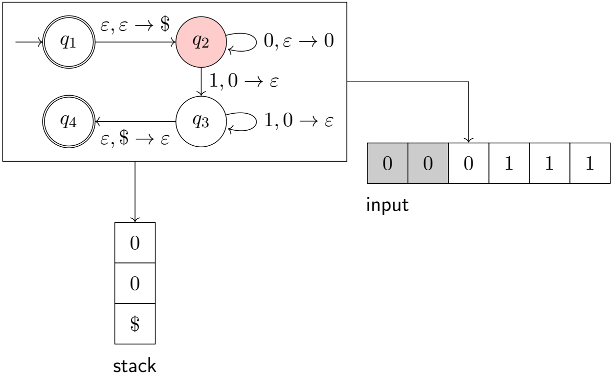
Pushdown Automata



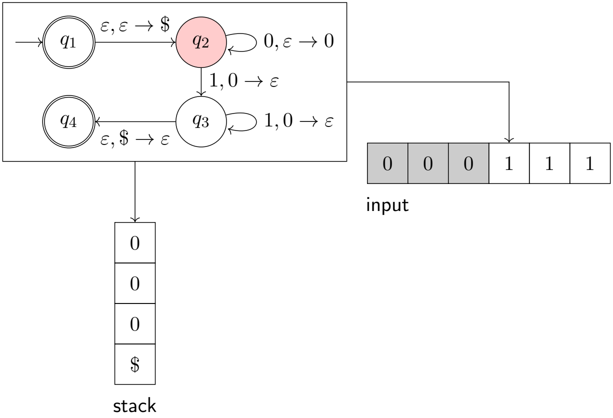
Pushdown Automata



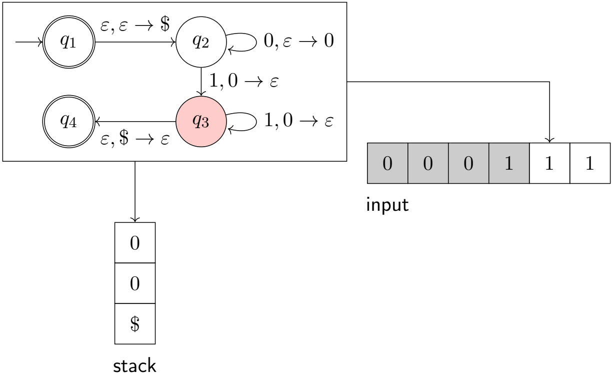
Pushdown Automata



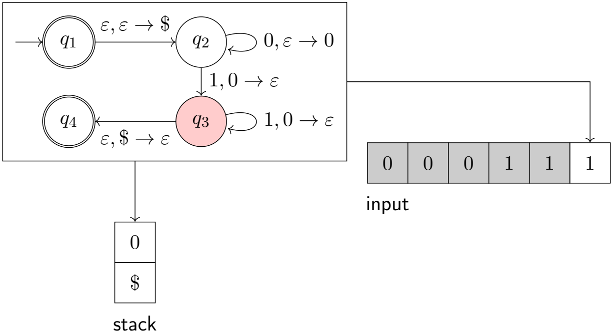
Pushdown Automata



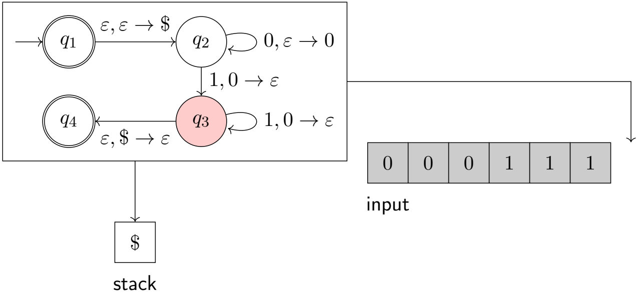
Pushdown Automata



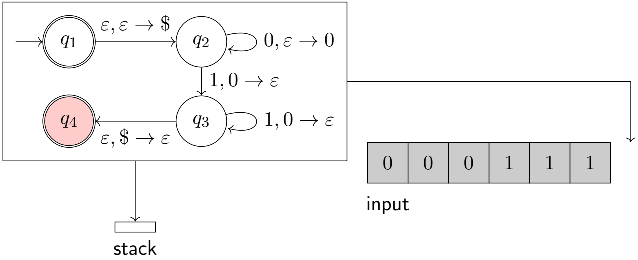
Pushdown Automata



Pushdown Automata

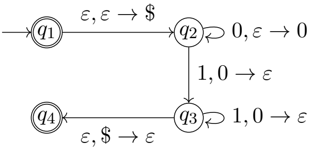


Pushdown Automata



Pushdown Automata

- $Q = \{q_1, q_2, q_3, q_4\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, \$\}$,
- Start state: q_1,
- $F = \{q_1, q_4\}$, and
- Following table gives δ, where blank entries signify $\emptyset$.



Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Theorem

A language is context free iff some pushdown automaton recognizes it.

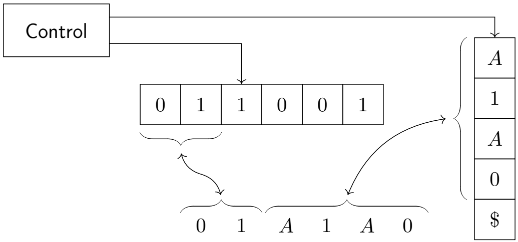
- The theorem has two directions.
- First, we prove the forward direction.
- Then, we prove the reverse direction.

Lemma

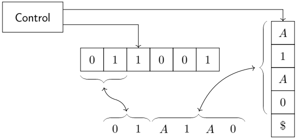
If a language is context-free, then some pushdown automaton recognizes it.

Proof idea:

- If a language is CF, then some CF grammar generates it.
- Construct PDA P that recognizes input string w, if CF grammar G generates that input (converting a CFG into a PDA).

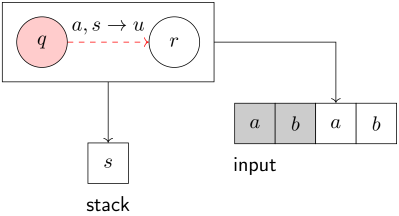


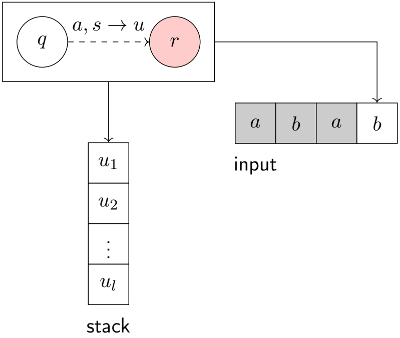
1. Push \$ and the start variable onto the stack.
2. Repeat:
    1. If the top of stack is a variable symbol A, nondeterministically select a substitution rule for A and substitute A with the right-hand side of the rule,
    2. If the top of stack is a terminal symbol a and is equal to the next symbol from the input, read the symbol from the input and pop the symbol. Otherwise, reject on this branch,
    3. If the top of stack is \$, enter the accept state. The string is accepted if it has all been read.



A handy notation:

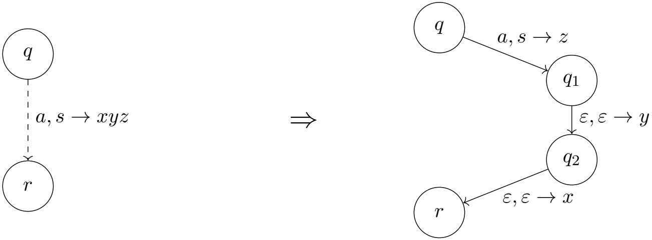
- Write the entire string $u = u_1 \dots u_l$ on the stack.





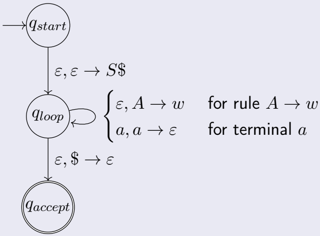
$\delta(q, a, s)$:

- Add (q_1, u_l) to $\delta(q, a, s)$,
- $\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, u_{l-1})\}$,
- $\delta(q_2, \varepsilon, \varepsilon) = \{(q_3, u_{l-2})\}$,
- $\vdots$
- $\delta(q_{l-1}, \varepsilon, \varepsilon) = \{(r, u_1)\}$.



Proof.

- $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup E$,
- Σ: Set of terminals in the grammar,
- Γ: Set of terminals, variables, and symbol $\$$,
- S: q_{start},
- $F = \{q_{accept}\}$, and
- δ:



$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_{loop}, S\$)\}, \tag{1}$$

$$\delta(q_{loop}, \varepsilon, A) = \{(q_{loop}, w) \mid A \to w \text{ is a rule}\}, \tag{2}$$

$$\delta(q_{loop}, a, a) = \{(q_{loop}, \varepsilon)\}, \tag{3}$$

$$\delta(q_{loop}, \varepsilon, \$) = \{(q_{accept}, \varepsilon)\}. \tag{4}$$

□

Lemma

If a pushdown automaton recognizes some language, then it is context free.

Assumptions:

- The automaton has a single accept state, q_{accept}.
- The automaton empties its stack before accepting.
- Each transition of the automaton either pushes a symbol or pops a symbol, but never both at the same time.

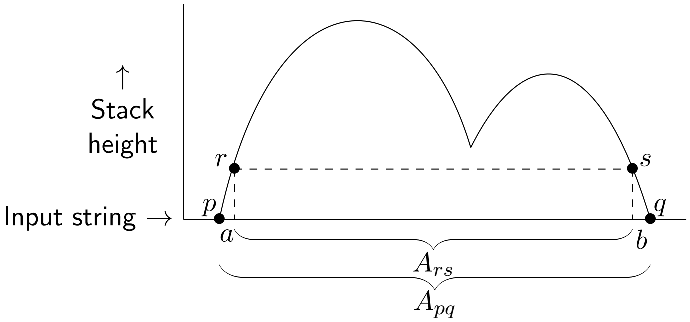
Proof idea:

- Build a CFG G that generates all strings accepted by the PDA P:
- For each pair of states $p, q \in P$, introduce a variable A_{pq},
- A_{pq} generates all strings that can take P from p with an empty stack to q with an empty stack.
- Strings that A_{pq} generates do not change the state of stack from p to q.
- Let x be a string generated by A_{pq}:
  - P's first move on x includes a push,
  - P's last move on x includes a pop.

Pushdown Automata

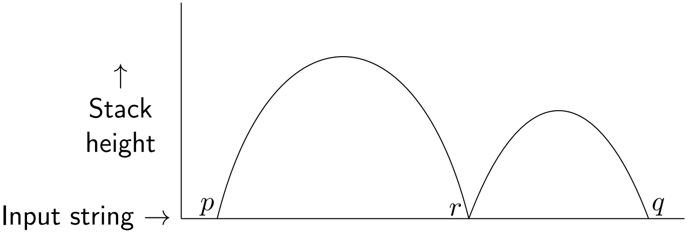
Proof idea (cont.):

- If at the end P pops the symbol that was pushed at the start:
  - Stack is only empty at the beginning and at the end,
  - $A_{pq} \to aA_{rs}b$,
  - a and b are input symbols,
  - r is the state after p and s is the state before q.



Proof idea (cont.):

- If at the end P doesn't pop the symbol that was pushed at the start:
  - Stack is also empty at a point besides the beginning and the end,
  - $A_{pq} \to A_{pr}A_{rq}$,
  - r is the state where the stack becomes empty.



Proof.

The equivalent grammar G of a given $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$ has variables:

- $\{A_{pq} \mid p, q \in Q\}$,

and, the start variable is $A_{q_0, q_{accept}}$. Rules of G are:

1. For $p, q, r, s \in Q$, $u \in \Gamma$, and $a, b \in \Sigma_\varepsilon$:
    1. If $\delta(p, a, \varepsilon)$ contains (r, u),
    2. If $\delta(s, b, u)$ contains (q, ε), then
    3. Put the rule $A_{pq} \to aA_{rs}b$ in G.
2. For $p, q, r \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$.
3. For $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

□

Claim

If A_{pq} generates x, then x can bring P from p with empty stack to q with empty stack.

Proof by induction:

- Basis: Consider a 1 step derivation. The only rules in G with no variable on the right-hand side is $A_{pp} \rightarrow \varepsilon$. Clearly, ε takes P from p with empty stack to q with empty stack.
- Induction step: Assume the claim is true for derivations of length at most k. Suppose $A_{pq} \overset{*}{\Rightarrow} x$ with $k+1$ steps. The first step in the derivation is either $A_{pq} \Rightarrow aA_{rs}b$ or $A_{pq} \Rightarrow A_{pr}A_{rq}$:

Proof by induction (Cont.):

- $A_{pq} \Rightarrow aA_{rs}b$: Let $x = ayb$ where A_{rs} generates y in k steps. Hypothesis tells y can bring P from r with empty stack to s with empty stack. By construction, $A_{pq} \Rightarrow aA_{rs}b$ means that $\delta(p, a, \varepsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ε) for some u. Therefore, x can bring P from p with empty stack to q with empty stack.
- $A_{pq} \Rightarrow A_{pr}A_{rq}$: Let $x = yz$ where $A_{pr} \overset{*}{\Rightarrow} y$ and $A_{rq} \overset{*}{\Rightarrow} z$ each in at most k steps, respectively. Hence, x can bring P from p with empty stack to q with empty stack.

Claim

If x can bring P from p with empty stack to q with empty stack, then A_{pq} generates x.

Proof by induction:

- Basis: Consider a 0 step computation, not changing the state and not reading any characters. Thus, x can only be ε. Since G has rules $A_{pp} \to \varepsilon$ the basis is proved.
- Induction step: Assume the claim is true for computations of length at most k. Suppose P has a computation on x that brings p with empty stack to q with empty stack in $k + 1$ steps. Either (1) the stack in only empty at the beginning and end of this computation, or (2) it becomes empty elsewhere, too:

Proof by induction (Cont.):

- Case 1: Let first and last read symbols be a and b, respectively. Symbol u that is pushed at the beginning is popped at the end. Thus, $\delta(p, a, \varepsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ε) for some r and s. Therefore, $A_{pq} \to aA_{rs}b$ is in G. Let $x = ayb$ where y can bring from r with empty stack to s with empty stack in $k-1$ steps (should be able to pop u at the end). According to the hypothesis, $A_{rs} \overset{*}{\Rightarrow} y$ and hence $A_{pq} \overset{*}{\Rightarrow} x$.
- Case 2: Let r be the state where the stack becomes empty. Then, computations from p to r and r to q have at most k steps with empty stacks at the start and the end. According to the hypothesis, $A_{pr} \overset{*}{\Rightarrow} y$ and $A_{rq} \overset{*}{\Rightarrow} z$ where $x = yz$. By construction, $A_{pq} \to A_{pr}A_{rq}$ is in G and thus $A_{pq} \overset{*}{\Rightarrow} x$.