Only If Part

Theorem: If a PDA accepts a language, then that language is context-free.

- For each pair of states p and q in P, we have variable A_{pq}.
- variable generates all strings that take P from p with an empty stack to q with empty stack.
- same string also takes p with any stack to q with same stack!

PDA Implies CFL

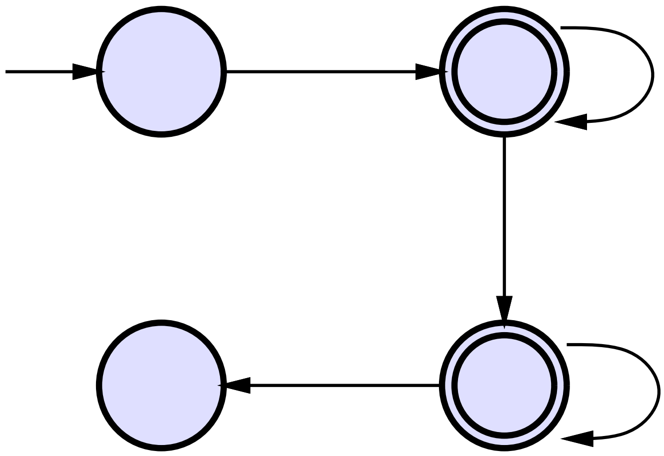
Modify P to make things easier

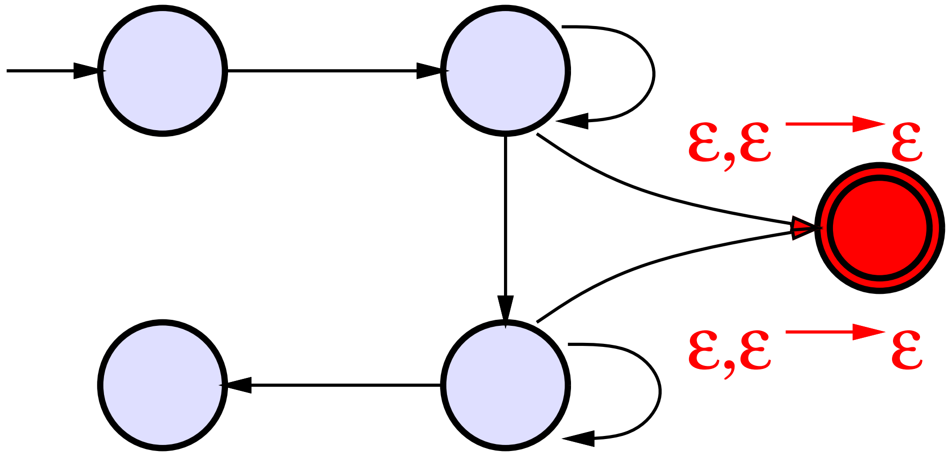
- single accept state q_a
- empties stack before accepting
- each transition either pushes or pops, but not both.

PDA Implies CFL

Modify P to make things easier

- single accept state q_a



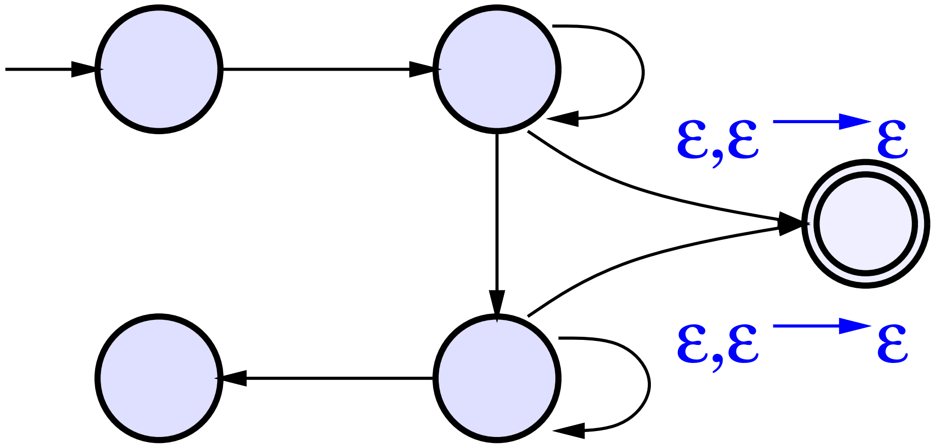


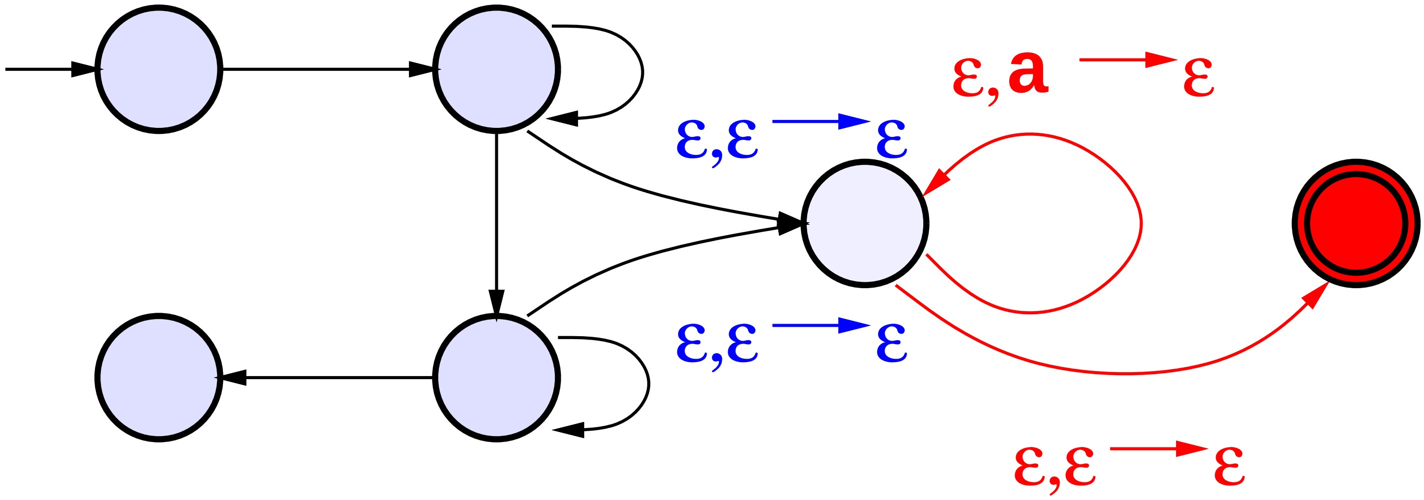
- empties stack before accepting
- each transition either pushes or pops, but not both.

PDA Implies CFL

Modify P to make things easier

- single accept state q_a
- empties stack before accepting



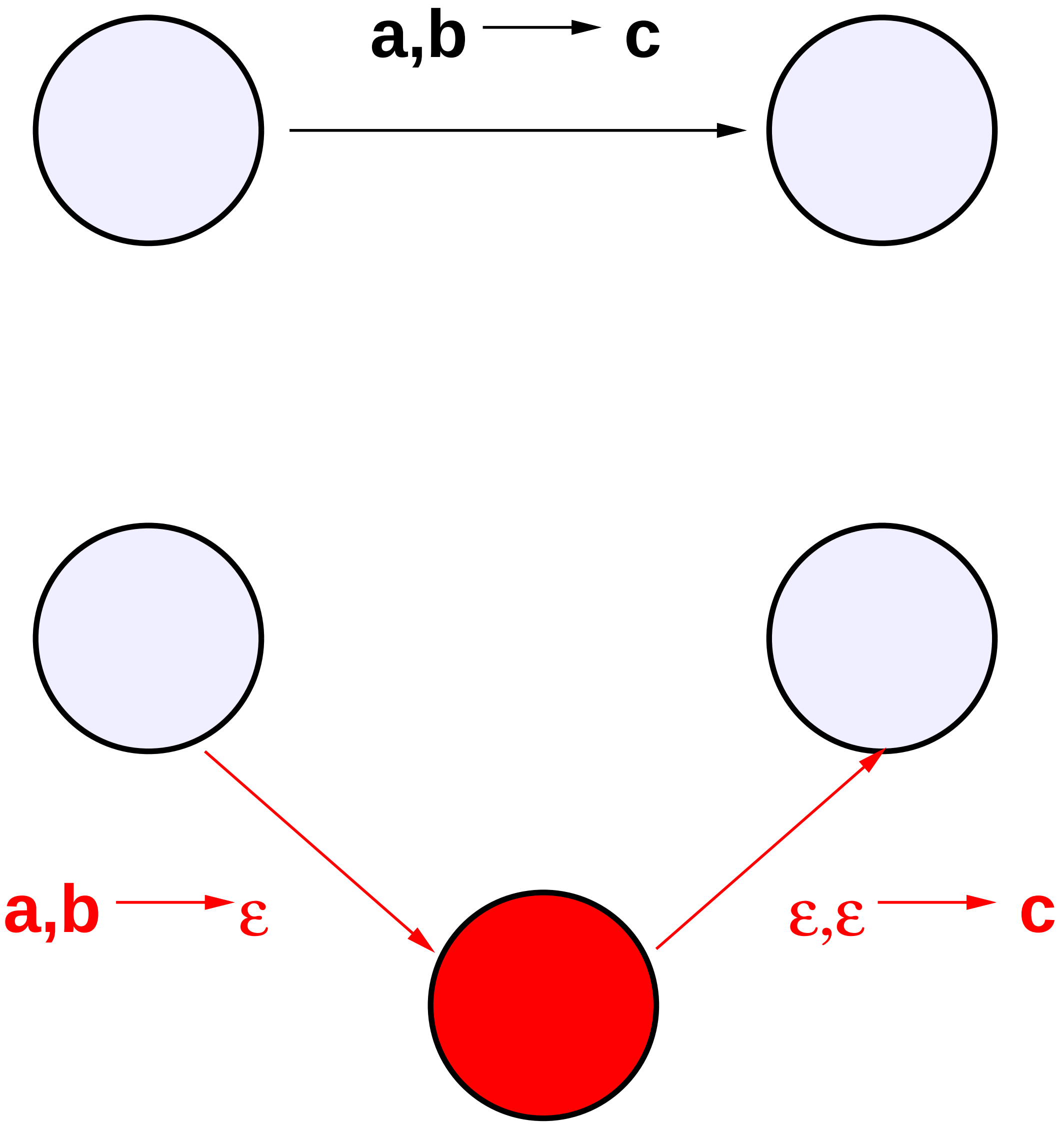


- each transition either pushes or pops, but not both.

PDA Implies CFL

Modify P to make things easier

- single accept state q_a
- empties stack before accepting
- each transition either pushes or pops, but not both.



Basic Idea

Suppose x takes P from p to q, leaving stack unchanged.

First move is push, last is pop.

In between, two possibilities:

- Stack empty at start and finish, but not in middle.
- Stack was empty in between.

Basic Idea

Suppose x takes P from p to q, leaving stack unchanged.

First move is push, last is pop.

In between, two possibilities:

- Stack empty at start and finish, but not in middle. Simulate by:

$$A_{pq} \to aA_{rs}b$$

  where a, b first and last input symbols, r follows p, and s precedes q.
- Stack was empty in between. Simulate by:

$$A_{pq} \to A_{pr}A_{rs}$$

  where r is intermediate state with empty stack.

Proof

Given PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_a\})$, construct G.

Variables are $\{A_{pq} | p, q \in Q\}$

Start variable is $A_{q_0 q_a}$.

Rules:

- [1] For $p, q, r, s \in Q$, $t \in Gamma$, and $a, b \in Sigma$, if $(r, t) \in \delta(p, q, \varepsilon)$ and $(q, \varepsilon) \in \delta(s, b, t)$ add rule

$$A_{pq} \to a A_{rs} b.$$

- [2] for $p, q, r \in Q$ add rule

$$A_{pq} \to A_{pr} A_{rq}.$$

- [3] for each $p \in Q$, add rule

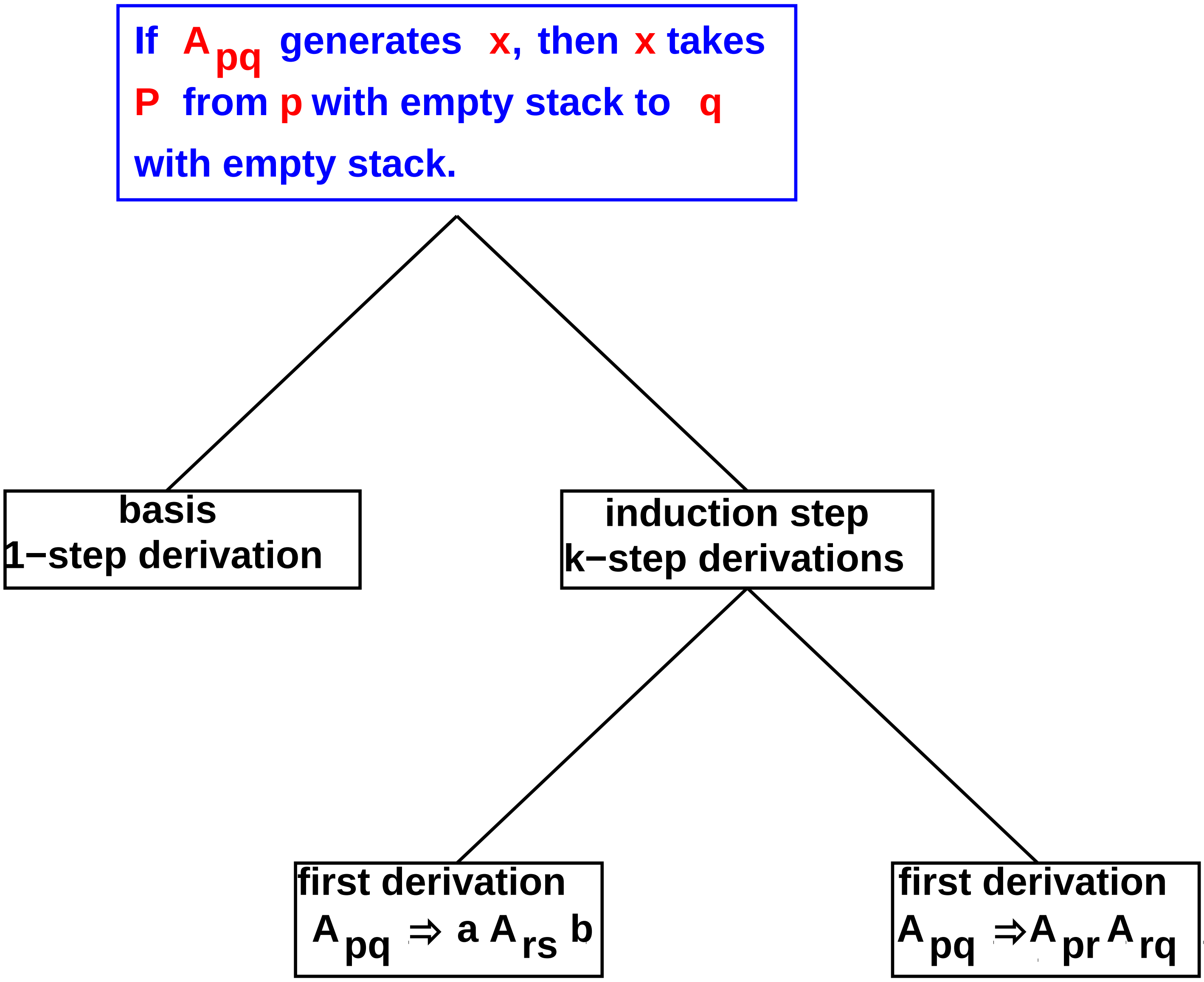
$$A_{pp} \to \varepsilon.$$

Overall Structure

Claim: A_{pq} generates x if and only if x brings P from p with empty stack to q with empty stack.

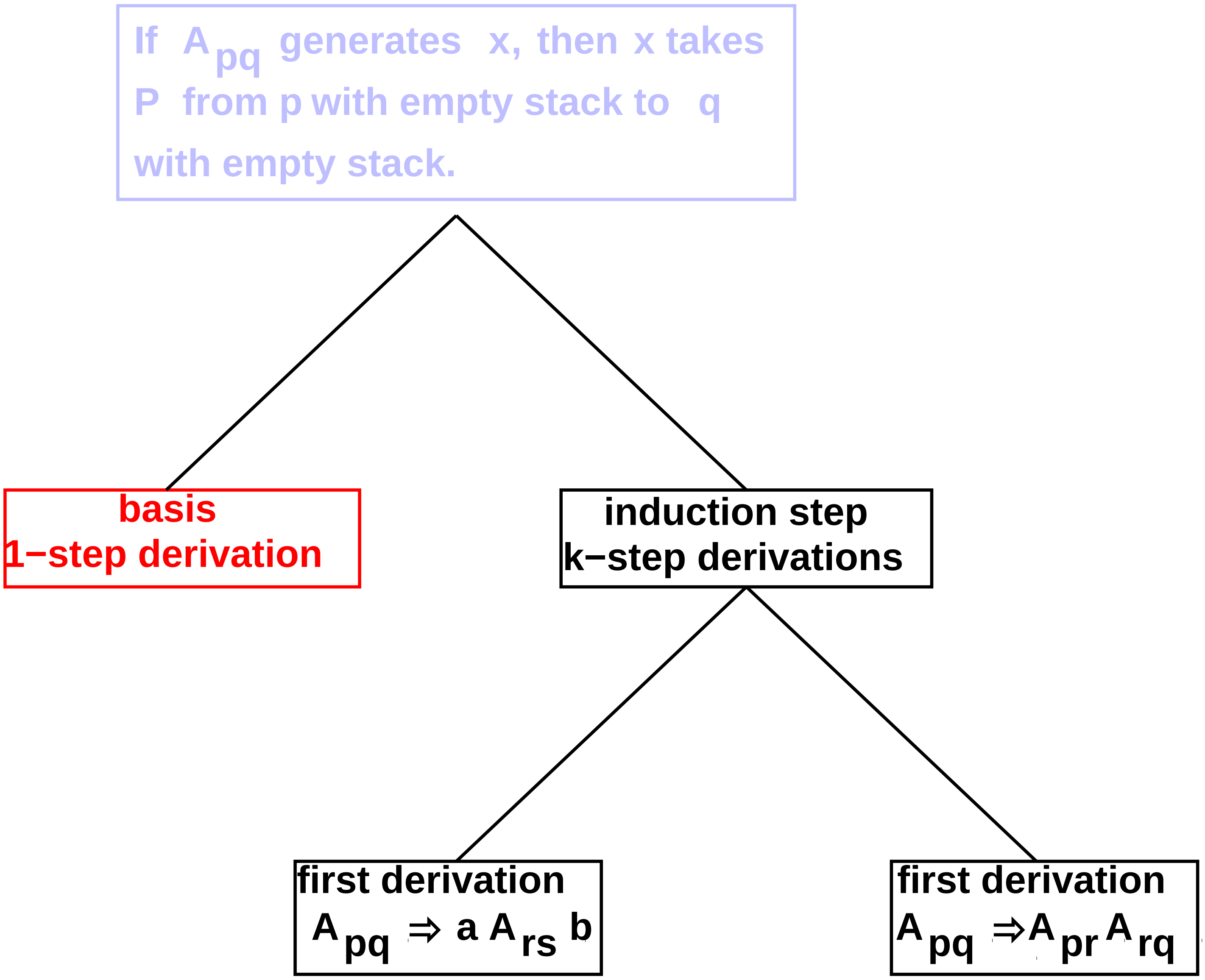
First Part

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.



First Part

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.



Basis

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.

Induction on number of steps in derivation.

Basis: The only single-step derivation is $App = \varepsilon$.

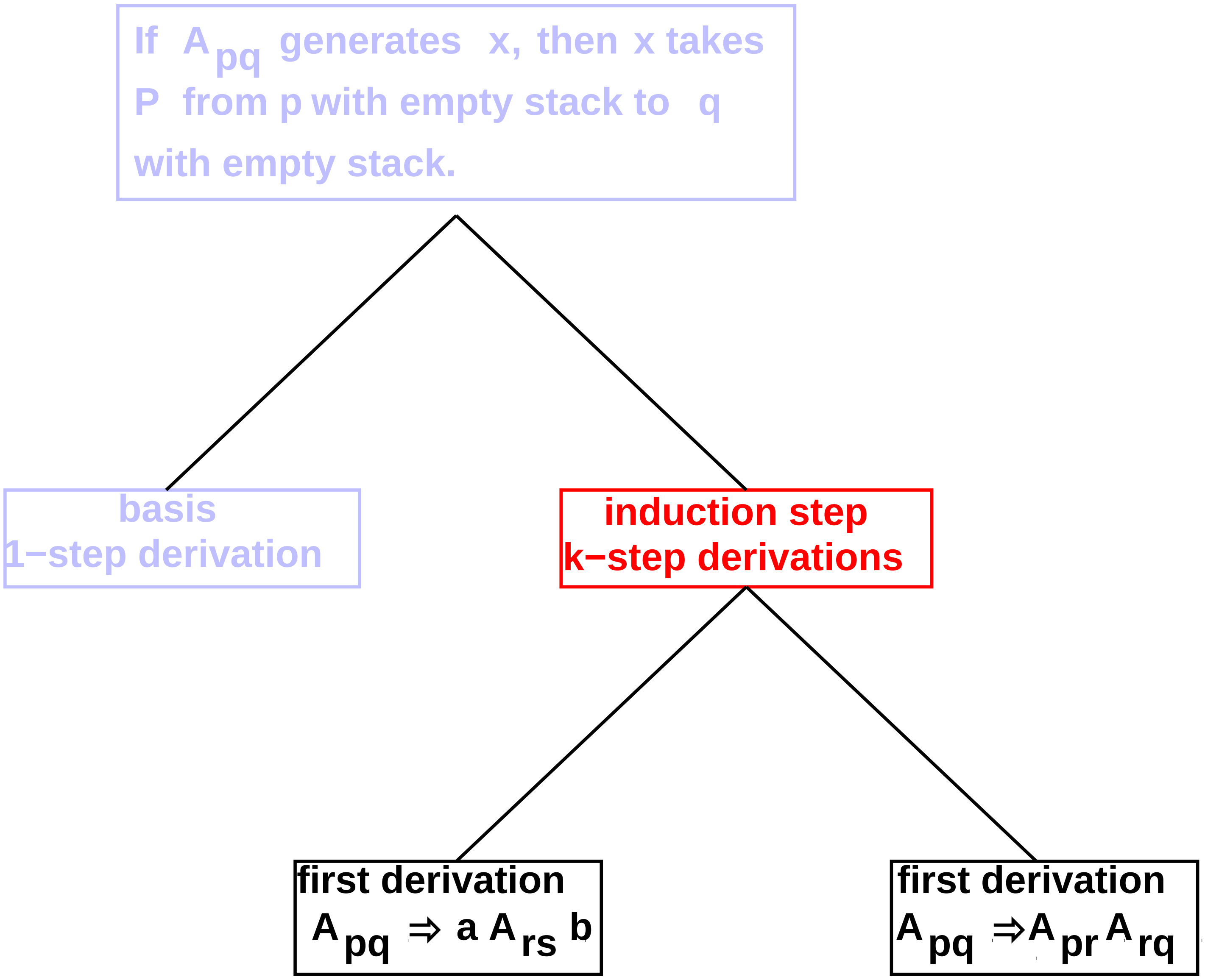
This is a rule in G:

- [3] for each $p \in Q$, add rule

$$A_{pp} \to \varepsilon.$$

Induction Step

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.



Proof

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.

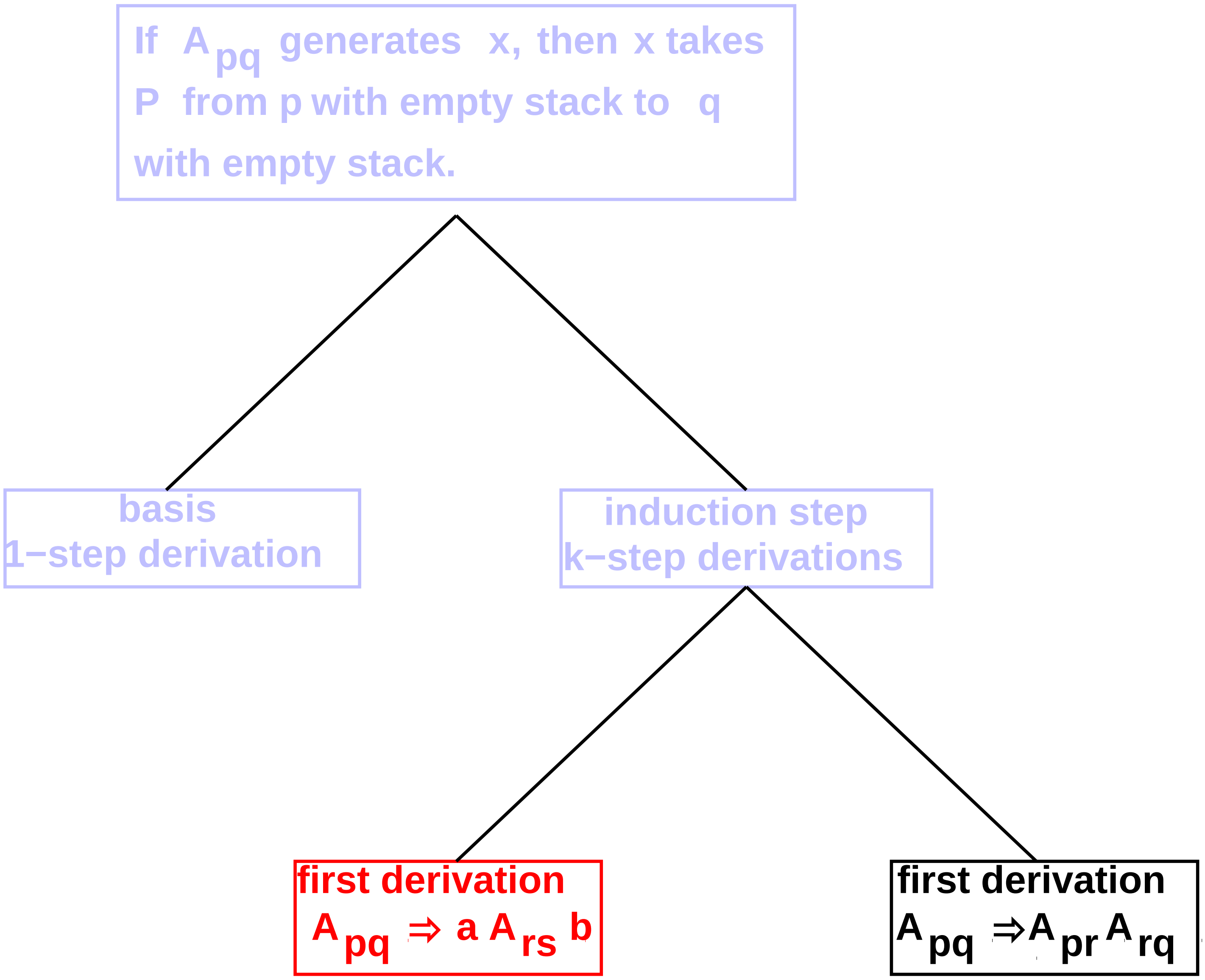
Induction Step: Assume claim for k steps, and let $A \stackrel{*}{\Rightarrow} x$ in $k+1$ steps.

Two cases:

1. first derivation is $A_{pq} \to aA_{rs}b$.
2. first derivation is $A_{pq} \to A_{pr}A_{rq}$.

First Part

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.



Proof

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.

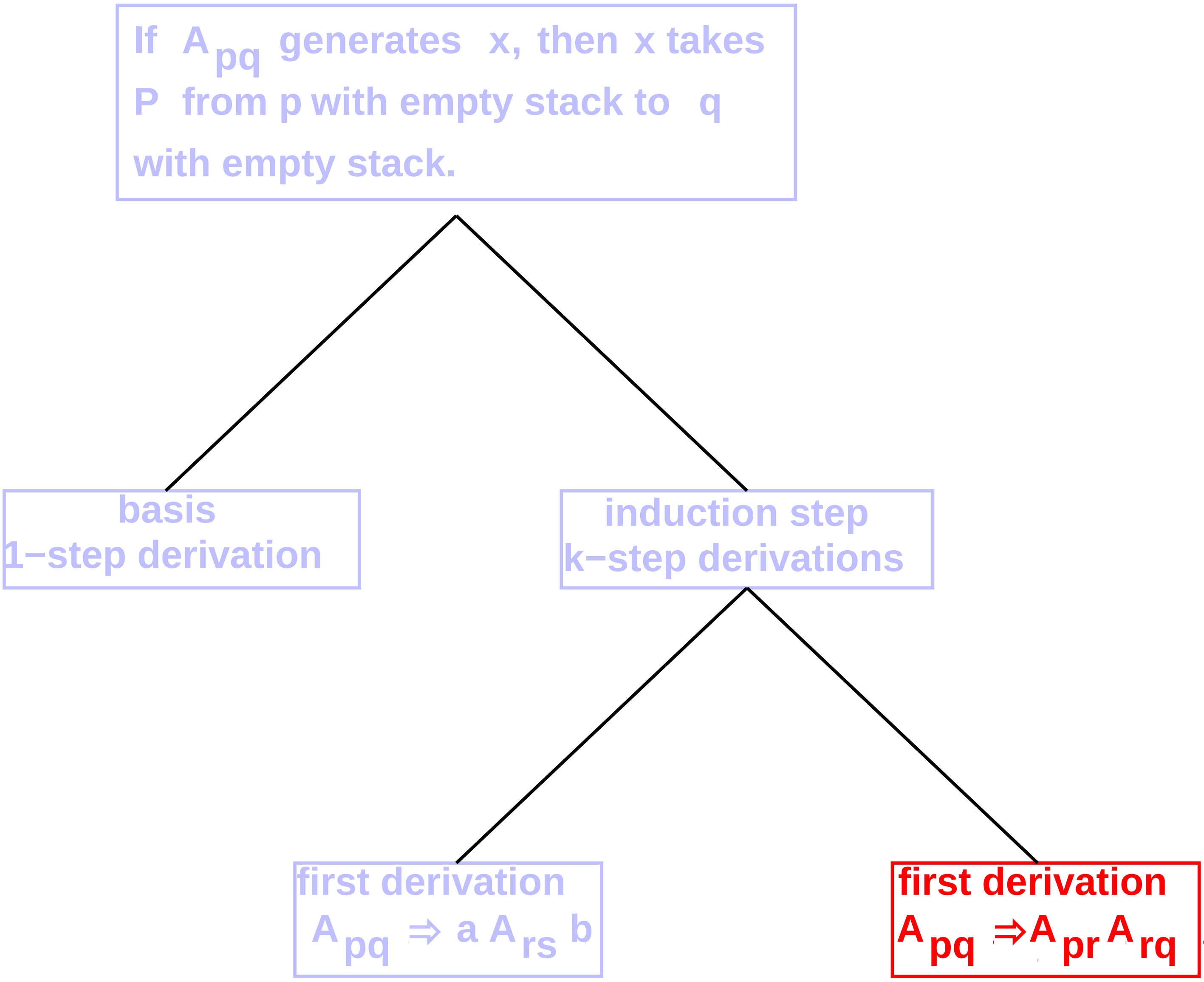
Induction Step: Assume claim for k-step, and let $A \stackrel{*}{\Rightarrow} w$.

Assume first derivation is $A_{pq} \to aA_{rs}b$.

- $x = ayb$, where $A_{rs} \stackrel{*}{\Rightarrow} y$ in k steps.
- P starts in p with empty stack
- reads a, pushes t, enters r.
- by induction hypothesis, P reaches s with t on stack
- P pops t and reaches q with empty stack.

First Part

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.



Proof

Claim: If A_{pq} generates x, then x takes P from p with empty stack to q with empty stack.

Induction Step: Assume claim for k-step, and let $A \stackrel{*}{\Rightarrow} w$.

First derivation is $A_{pq} \to A_{pr} A_{rq}$.

- $x = yz$, where $A_{pr} \stackrel{*}{\Rightarrow} y$ and $A_{rr} \stackrel{*}{\Rightarrow} z$ in $\leq k$ steps.
- induction hypothesis: y carries P from q with empty stack to r with empty stack
- induction hypothesis: z carries P from r with empty stack to q with empty stack

This completes proof of claim.

Other Direction

Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.

Induction on number of steps in computation.

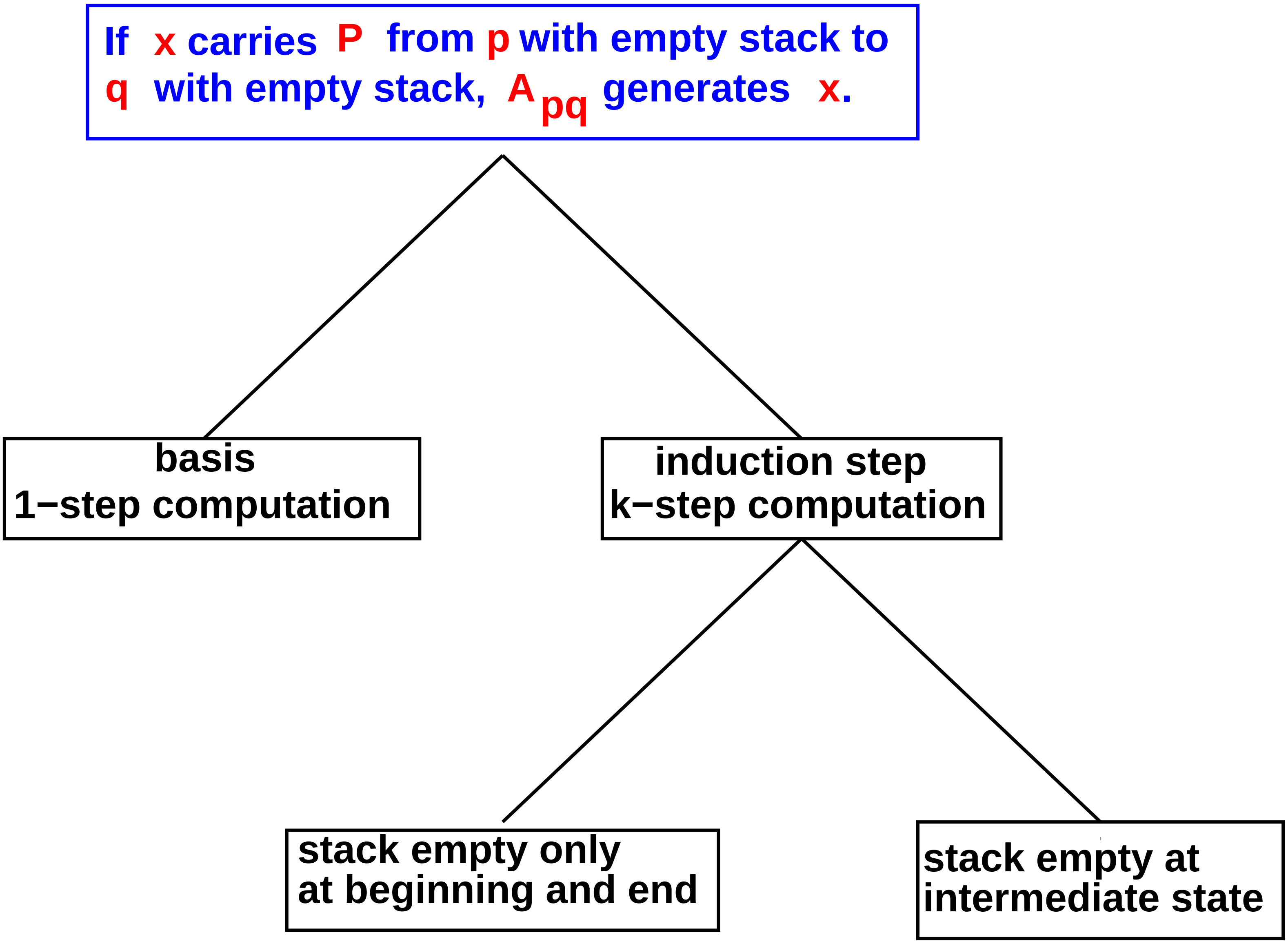
Base Case: zero-step computation.

Induction Step: k-step computation. Two cases:

1. stack never empty during computation.
2. stack empty at intermediate state r.

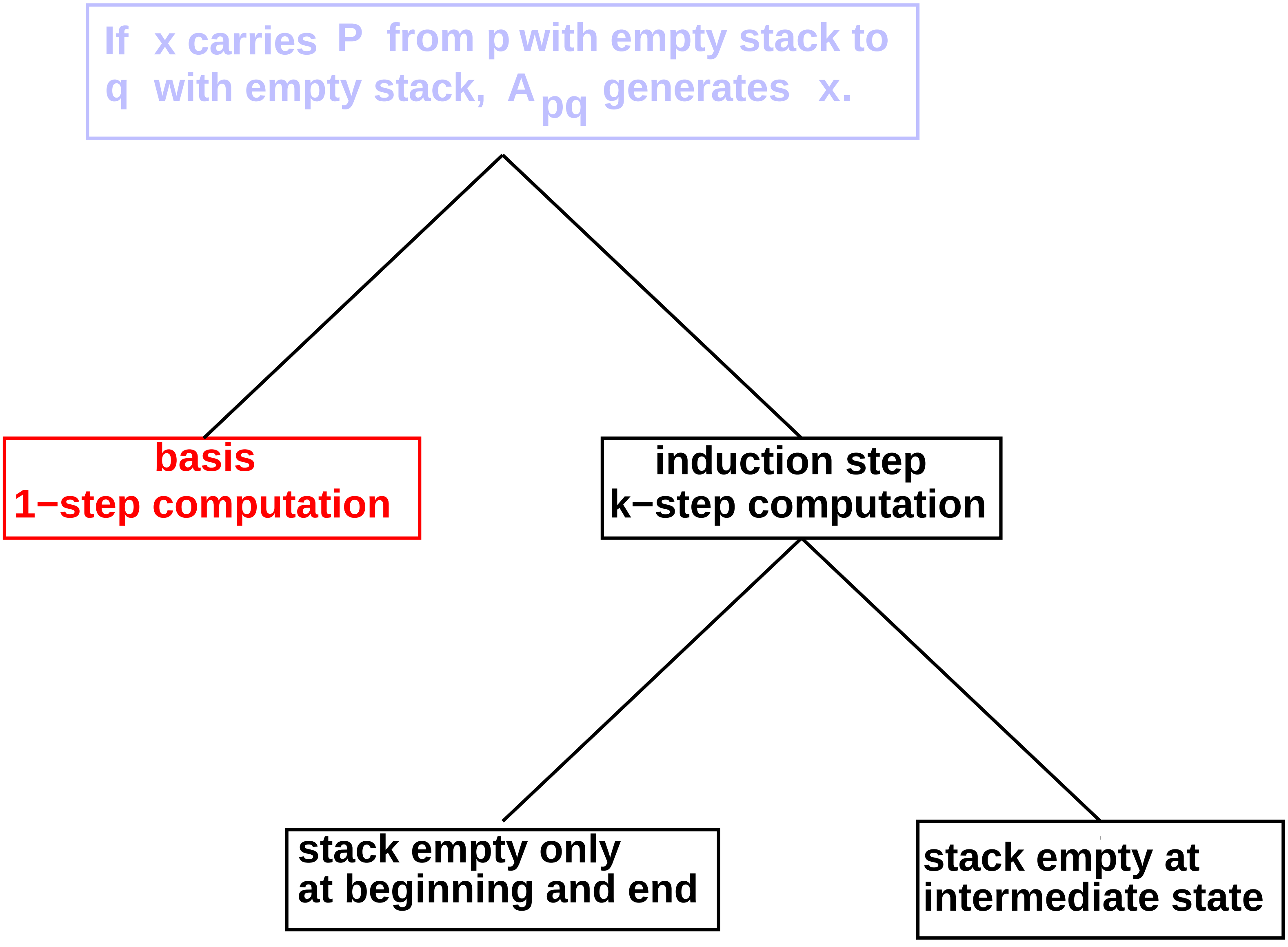
Other Direction

Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.



Basis

Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.



Basis

Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.

Induction on number of steps in computation.

Basis: In any zero-step computation, P reads only ε.

This is a rule in G:

- [3] for each $p \in Q$, add rule

$$A_{pp} \to \varepsilon.$$

Induction Step

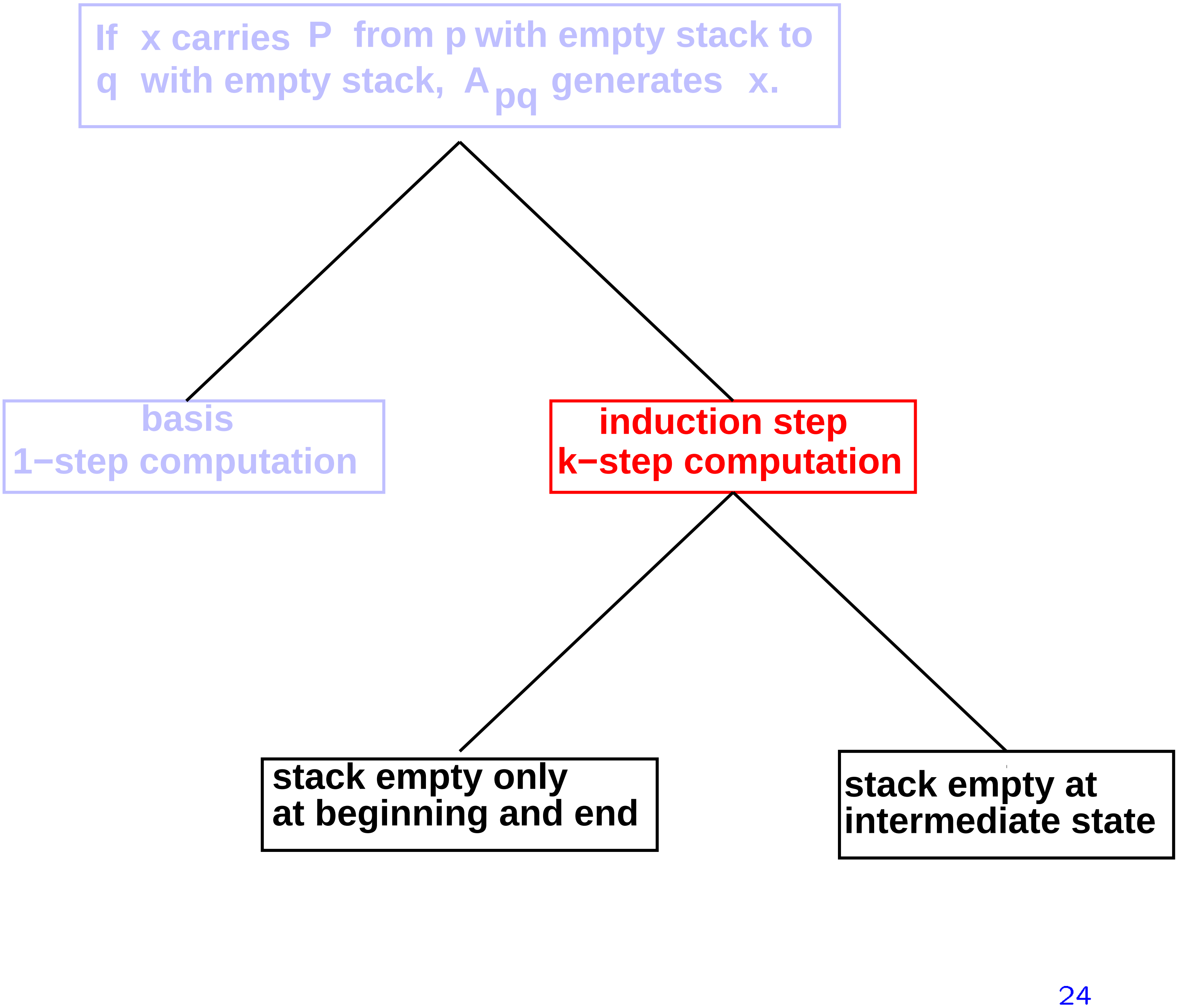
Assume claim for k-step computation.

Suppose x brings P from p with empty stack to q with empty stack in $k+1$ steps.

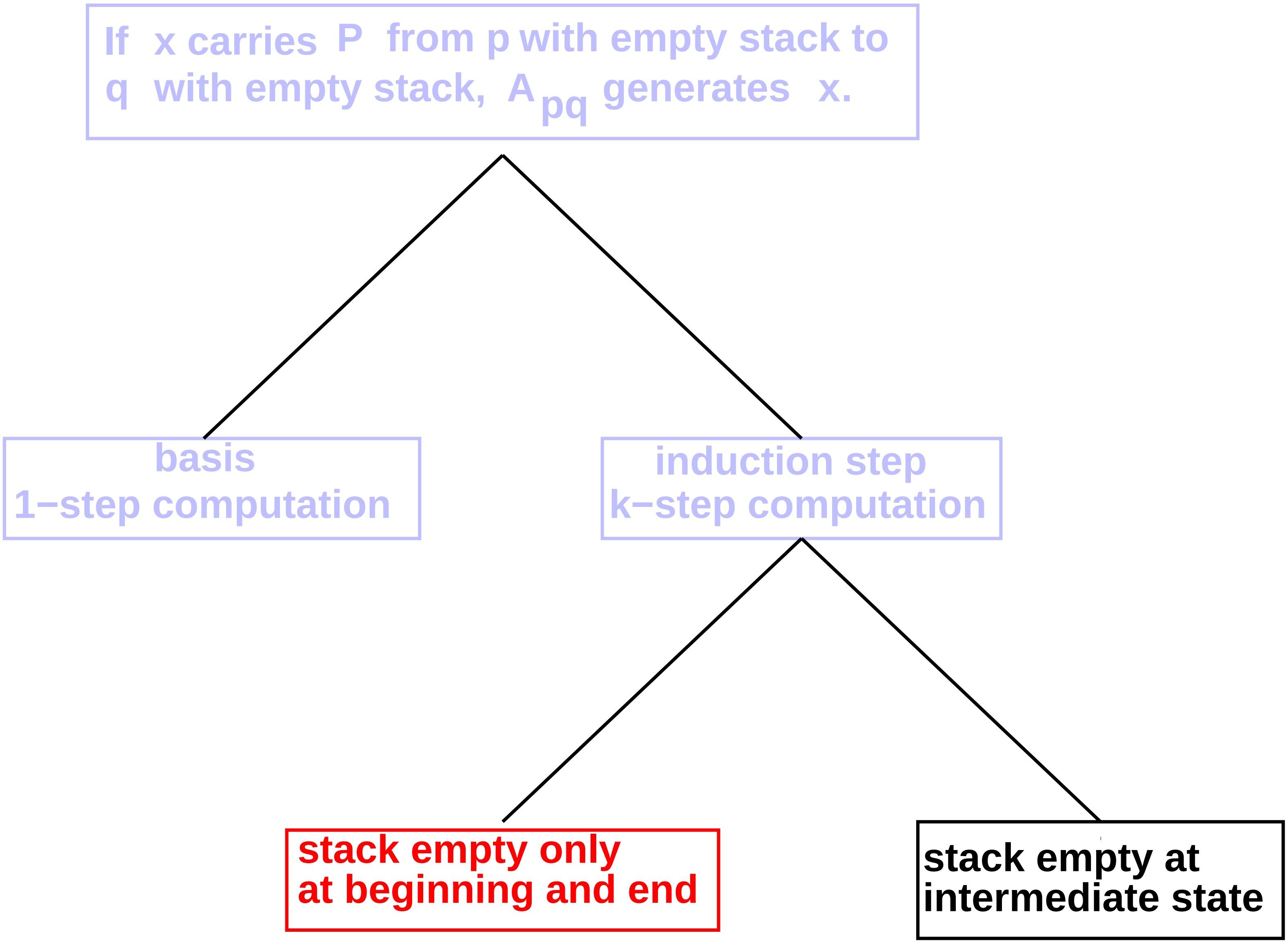
Two cases:

1. stack never empty during computation.
2. stack empty at intermediate state r.

Induction Step



Induction Step



Proof

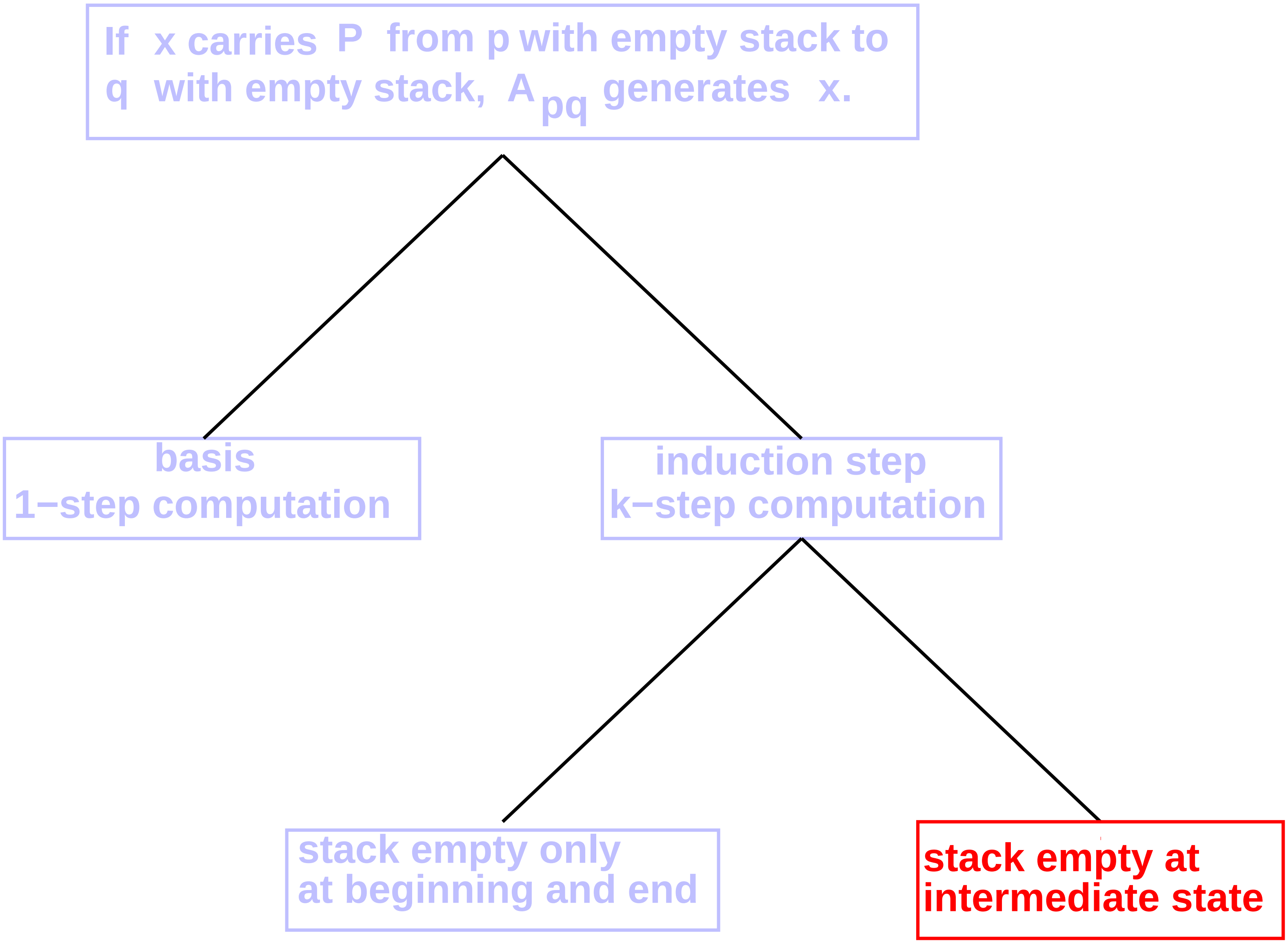
Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.

Suppose x brings P from p with empty stack to q with empty stack in $k + 1$ steps.

Suppose stack never empty during computation.

- first symbol pushed t is last symbol popped.
- let a be input read in first move
- r state state after first move
- s state before last move
- $(r, t) \in \delta(p, a, \varepsilon)$ and $(q, \varepsilon) \in \delta(s, b, t)$ so $A_{pq} \to aA_{rs}b$ is a rule.

Induction Step



Induction Step

Claim: If x brings P from p with empty stack to q with empty stack, then A_{pq} generates x.

Induction on number of steps in computation.

Suppose x brings P from p with empty stack to q with empty stack in $k+1$ steps.

Suppose stack empty at intermediate state r.

- Computations from p to r and from r to q each contain $\leq k$ steps.
- Let y be input in first portion, and z in second.
- Induction hypothesis says that $A_{pr} \stackrel{*}{\Rightarrow} y$ and $A_{rq} \stackrel{*}{\Rightarrow} z$.
- But we have rule $A_{pq} \to A_{pr} A_{rq}$.

This completes the proof that pushdown automata accept all and only context-free languages.

Corollary: Every regular language is context-free.