Turing Machines

Formal Definition

A Turing Machine is represented with a quadruple: (Q, Σ, q_I, δ) where:

- Q finite set of states,
- Σ finite set of symbols that include $\#$ that represents a blank.
- $q_I \in Q$ Initial state
- $\delta : (Q \times \Sigma) \to (Q \cup \{\bigcirc\} \times (\Sigma \times (\{R, L, S\}))$ is a partial function.

Behavior

- $\delta(q, \sigma) = (q_1, \rho, R)$, on state q,reading σ, overwrite ρ, move right and go to state q_1.
- $\delta(q, \sigma) = (q_1, \rho, S)$, on state q,reading σ, overwrite ρ, go to state q_1, and don't move.
- $\delta(q, \sigma) = (q_1, \rho, L)$, on state q,reading σ, overwrite ρ, move left and go to state q_1.
- $\delta(q, \sigma) = (\bigcirc, \rho, R)$, on state q,reading σ, overwrite ρ, move right, and , and halt execution.
- $\delta(q, \sigma) = (\bigcirc, \rho, S)$, on state q,reading σ, overwrite ρ, don't move, and halt execution
- $\delta(q, \sigma) = (\bigcirc, \rho, L)$, on state q,reading σ, overwrite ρ, move left, and halt execution.

Configuration

A configuration $C \in (Q \cup \{\bigcirc\} \times \Sigma^* \times \mathbb{N}) : \langle q, \omega, n \rangle$

- q is the state
- ω is the string written on the beginning of the tape afterwhich there are only blanks. It may contain blanks; it may even end with blanks, but after ω there are only blanks.
- n is the position of the head.

Configuration

- Given a configuration $\langle q, \omega, i\rangle$
- Let $\omega = \sigma_0 \ldots \sigma_{n-1}$; , $0 \leq i < n$
- We can omit i and represent the configuration as follows: $\langle q, \sigma_0 \ldots \underline{\sigma_i} \ldots \sigma_{n-1}\rangle$

One-step Transitions

- $\langle q, \sigma_0 \ldots \underline{\sigma_i} \ldots \sigma_{n-1}\rangle \Rightarrow \langle q', \sigma_0 \ldots \rho\underline{\sigma_{i+1}} \ldots \sigma_{n-1}\rangle$ if $\delta(q, \sigma i) = \langle q', \rho, R\rangle$
- $\langle q, \sigma_0 \ldots \underline{\sigma_i} \ldots \sigma_{n-1}\rangle \Rightarrow \langle q', \sigma_0 \ldots \underline{\sigma_{i-1}}\rho \ldots \sigma_{n-1}\rangle$ if $\delta(q, \sigma i) = \langle q', \rho, L\rangle$
- $\langle q, \sigma_0 \ldots \underline{\sigma_i} \ldots \sigma_{n-1}\rangle \Rightarrow \langle q', \sigma_0 \ldots \underline{\rho}\sigma_{i+1} \ldots \sigma_{n-1}\rangle$ if $\delta(q, \sigma i) = \langle q', \rho, S\rangle$

λ-Transitions

λ-transitions are used to ignore the tape: don't read and/or don't write.

$$\delta : (Q \times (\Sigma \cup \{\lambda\}) \to (Q \cup \{\bigcirc\} \times (((\Sigma \cup \{\lambda\}) \times (R, L, S))$$

Where:

- $\delta(q, \lambda) = (p, \rho, Op)$: writes ρ and executes Op no matter what symbol is on the tape, then switch to state p
- $\delta(q, \sigma) = (p, \lambda, op)$ leaves σ,
- $\delta(q, \lambda) = (p, \lambda, op)$: executes Op,then switch to state p.

You should not confuse $\#$ with λ, as $\#$ is a symbol of the alphabet.
We can add non-determinism. To ensure determinism:

- If there is a transition from q reading λ, there should not be any other transition from q.
- If there is a transition from q to r executes op and writes λ, there should not be another transition from q to r executes op and writes something else.

Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

Equivalence to machines without λ-transitions

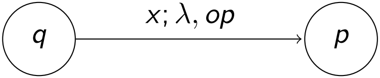
If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

Replace each transition of the form: $\delta(q, x) = (p, \lambda, op)$ by $\delta(q, x) = (p, x, op)$.

Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

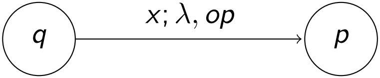
Replace each transition of the form: $\delta(q, x) = (p, \lambda, op)$ by $\delta(q, x) = (p, x, op)$.

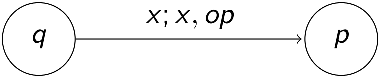


Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

Replace each transition of the form: $\delta(q, x) = (p, \lambda, op)$ by $\delta(q, x) = (p, x, op)$.





Equivalence to machines without λ-transitions

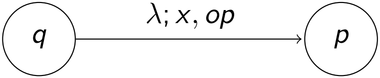
If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.

Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.
Replace each transition of the form: $\delta(q, \lambda) = (p, x, op)$ with the following transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, x, op)\}$ Remember!! $\# \in \Sigma$

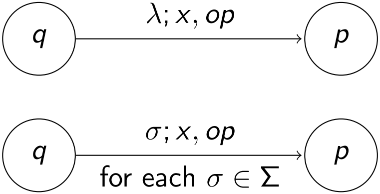
Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.
Replace each transition of the form: $\delta(q, \lambda) = (p, x, op)$ with the following transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, x, op)\}$ Remember!! $\# \in \Sigma$



Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine without $M = (Q, \Sigma, q_I, \delta)$ λ-transitions.
Replace each transition of the form: $\delta(q, \lambda) = (p, x, op)$ with the following transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, x, op)\}$ Remember!! $\# \in \Sigma$

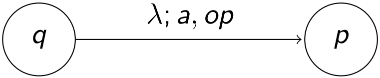


Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$

Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$



Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$

$$q \xrightarrow{\lambda;\, a,\, op} p$$

$$q \xrightarrow[(\#;\, a,\, op)]{(a;\, a,\, op)\ (b;\, a,\, op)} p$$

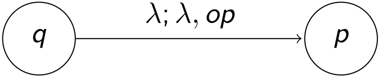
Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine $M = (Q, \Sigma, q_I, \delta)$ without λ-transitions.
Replace each transition of the form: $\delta(q, \lambda) = (p, \lambda, op)$ by the following set of transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, \sigma, op)\}$

Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine $M = (Q, \Sigma, q_I, \delta)$ without λ-transitions.

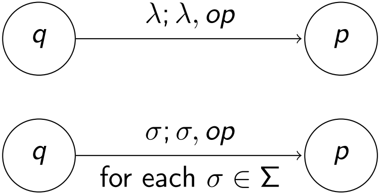
Replace each transition of the form: $\delta(q, \lambda) = (p, \lambda, op)$ by the following set of transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, \sigma, op)\}$



Equivalence to machines without λ-transitions

If $M_\lambda = (Q, \Sigma, q_I, \delta_\lambda)$ is a Turing Machine with λ-transitions, we can define an equivalent Turing Machine $M = (Q, \Sigma, q_I, \delta)$ without λ-transitions.

Replace each transition of the form: $\delta(q, \lambda) = (p, \lambda, op)$ by the following set of transitions $\{\sigma : \Sigma | \delta(q, \sigma) = (p, \sigma, op)\}$



Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$

Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$

$$\begin{tikzcd} q \arrow[rr, "\lambda; \lambda, op"] & & p \end{tikzcd}$$

Equivalence to machines without λ-transitions

Example $\Sigma = \{a, b, \#\}$

$$q \xrightarrow{\lambda; \lambda, op} p$$

$$q \xrightarrow[(\#; \#, op)]{(a; a, op)\ (b; b, op)} p$$

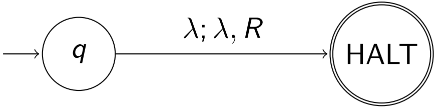
Combining Machines

We can combine machines. A machine can invoke another machine. If we define small simple machines, we can build larger machines combining these machines.

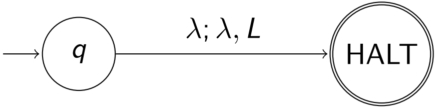
Simple Machines

- Move Right $\boxed{R}$
- Move Left $\boxed{L}$
- Move Right until $\boxed{R_\sigma}$
- Move Left until $\boxed{L_\sigma}$
- Move Right until not $\boxed{R_{!\sigma}}$
- Move Left until not $\boxed{L_{!\sigma}}$
- Write $\boxed{W_\sigma}$
- Invoke another machine $\boxed{C_M}$

Simple Machines: Move Right $\boxed{R}$



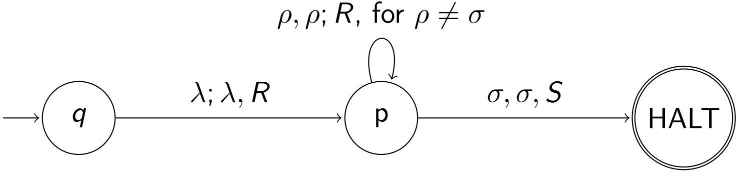
Simple Machines: Move Left $\boxed{L}$



Simple Machines: Move Right until σ: $\boxed{R_\sigma}$

This machine executes the following operations:

1. Move Right
2. If reading σ, then HALT else start again.



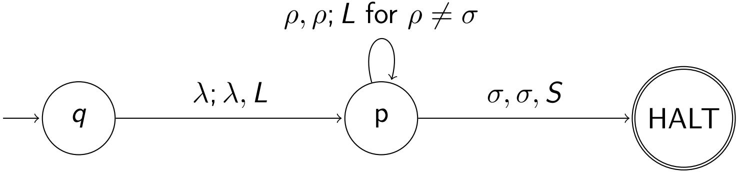
Note that it moves to the right once!

- Start configurationn: $xyz\underline{a}bccad$
- End configurationn: $xyzabcc\underline{a}d$

Simple Machines: Move Left until σ: $\boxed{L_\sigma}$

This machine executes the following operations:

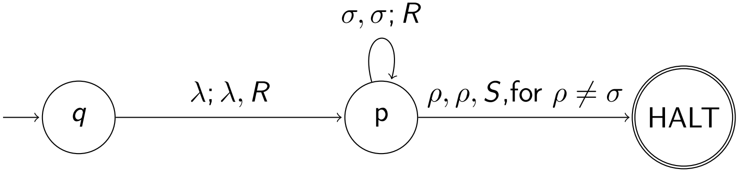
1. Move Left
2. If reading σ, then HALT else start again.



Simple Machines: Move Right until not σ: $\boxed{R_{!\sigma}}$

This machine executes the following operations:

1. Move Right
2. If not reading σ, then HALT else start again.

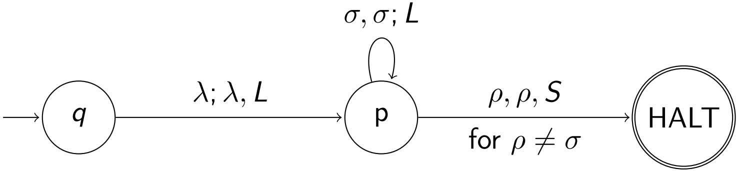


- Start configuration: $xyz\underline{b}aaadf$
- End configurationn: $xyzbaaa\underline{d}f$

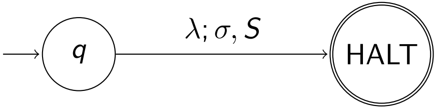
Simple Machines: Move Left until not σ: $\boxed{L_{!\sigma}}$

This machine executes the following operations:

1. Move Left
2. If not reading σ, then HALT else start again.



Simple Machines: Write σ: $\boxed{W_\sigma}$



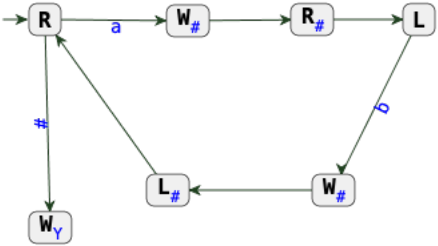
New Formalism

Nodes are operations:

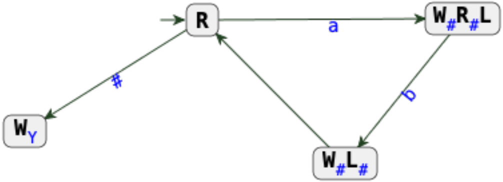
- Move Right $\boxed{R}$
- Move Left $\boxed{L}$
- Move Right until $\boxed{R_\sigma}$
- Move Left until $\boxed{L_\sigma}$
- Move Right until not $\boxed{R_{!\sigma}}$
- Move Left until not $\boxed{L_{!\sigma}}$
- Write $\boxed{W_\sigma}$
- Invoke another machine $\boxed{C_M}$

Arcs may be labeled with symbols

$\#a^n b^n$

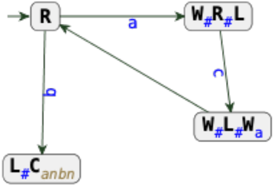


$\#a^n b^n$:Boxing together operations

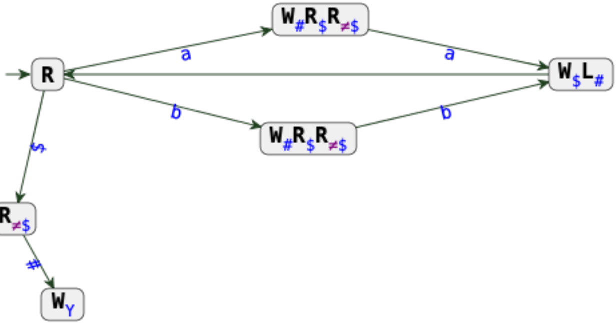


$\#a^n b^n c^n$

First : $\#a^n b^n c^n \Rightarrow \#a^n b^n$ Then call $\#a^n b^n$

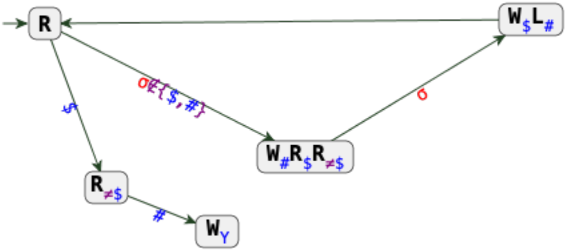


$W\$W,\ W \in \{a, b\}*$

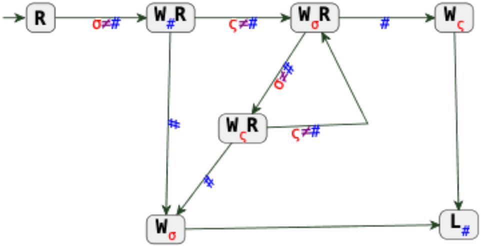


Using variables

$W\$W,\ W \in (\Sigma \setminus \{\$, \#\})*$



Buildiing Blocks: ShiftRight



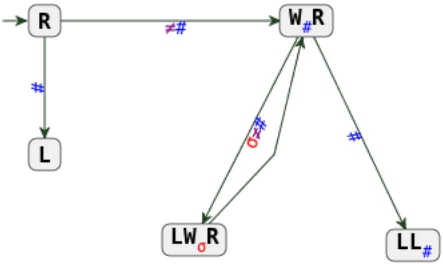
Initial configuration:

$$\sigma_0 \ldots \underline{\sigma_i}\sigma_{i+1} \ldots \sigma_{n-1}$$

Final Configuration:

$$\sigma_0 \ldots \sigma_i \underline{\#} \sigma_{i+1} \ldots \sigma_{n-1}$$

Buildiing Blocks: ShiftLeft



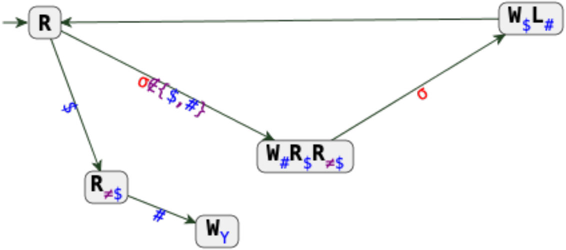
Initial Configuration:

$$\sigma_0 \ldots \underline{\#}\sigma_{i+1} \ldots \sigma_n - 1$$

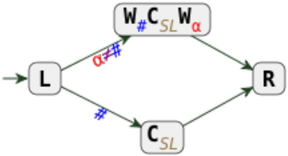
Final Configuration:

$$\sigma_0 \ldots \underline{\#}\sigma_{i+2} \ldots \sigma_n - 1$$

Buildiing Blocks: ShiftLeft



Buildiing Blocks: Delete This



Initial configuration:

$$\sigma_0 \ldots \sigma_{i-1}\underline{\sigma_i}\sigma_{i+1} \ldots \sigma_n - 1$$

Final configuration:

$$\sigma_0 \ldots \sigma_{i-1}\underline{\sigma_{i+1}} \ldots \sigma_n - 1$$