Nondeterministic Turing machines (Section 7.7 in ILTC)

A *nondeterministic Turing machine* (NTM) is defined like a Turing machine, except that

(1) δ has the form $\delta\colon (Q - \{\mathrm{h_a}\}) \times \Gamma \to 2^{Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}}$ and

(2) the reject state $\mathrm{h_r}$ is replaced by $\mathrm{h}_\emptyset$, where $\delta(\mathrm{h}_\emptyset, a) = \emptyset$ for all $a \in \Gamma$.

An input string w is *accepted* by an NTM if there exists a transition sequence from $q_0 \Delta w$ to an accepting configuration.

Crash at the start of the tape: we define $qav \vdash \mathrm{h}_\emptyset av$ if $\delta(q, a) = (r, b, \mathrm{L})$ for some $r \in Q$ and $b \in \Gamma$.

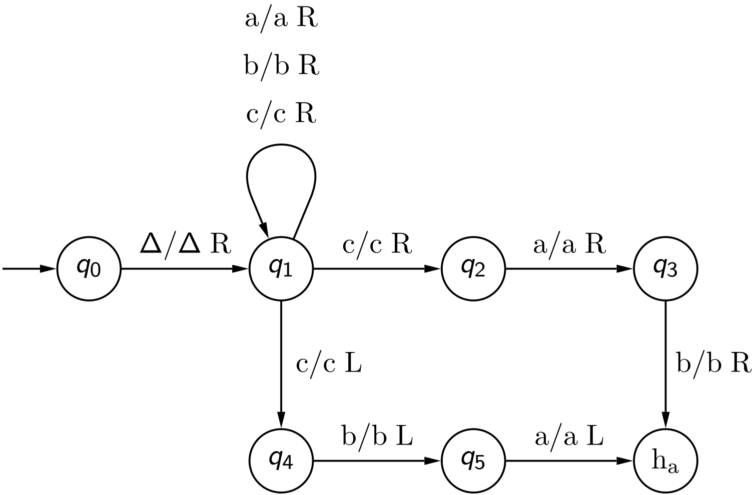
Nondeterministic Turing machines (Section 7.7 in ILTC)

Notes:

- For a given input string, there may be several transition sequences which an NTM can choose to execute.
- A reject state $\mathrm{h_r}$ wouldn't make sense: there could be transition sequences leading from an initial configuration to both a rejecting and an accepting configuration.
- The drawing convention for TMs does not apply: there are no hidden transitions.
- Given a state q and a tape symbol a such that $\delta(q, a) = \emptyset$, an NTM *halts without acceptance or rejection*: the input string may or may not be in the machine's language.
- If an NTM loops on an input string, the string may or may not be in the language.

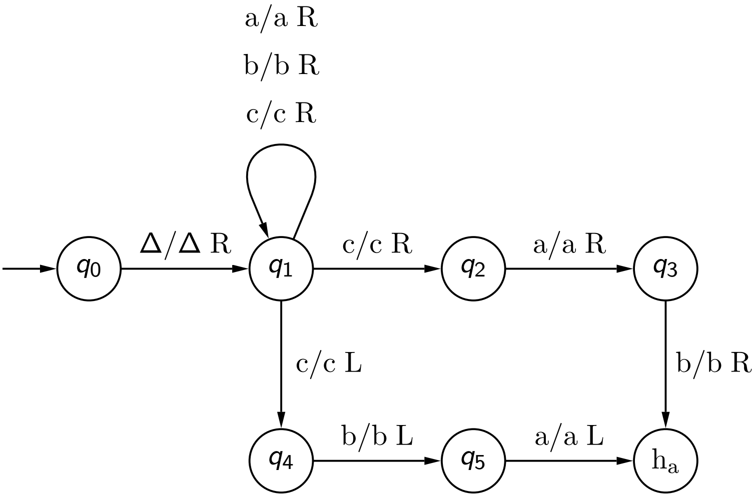
Example (Nondeterministic Turing machine)

N:



Example (Nondeterministic Turing machine)

N:



$\mathrm{L}(N) = \Sigma^*\{\texttt{abc}, \texttt{cab}\}\Sigma^*$ where $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$

Nondeterministic vs deterministic TMs

Theorem (Thm 7.31 in ILTC)

For every nondeterministic Turing machine N there exists a (deterministic) Turing machine D such that $\mathrm{L}(N) = \mathrm{L}(D)$.

Nondeterministic vs deterministic TMs

Theorem (Thm 7.31 in ILTC)

For every nondeterministic Turing machine N there exists a (deterministic) Turing machine D such that $\mathrm{L}(N) = \mathrm{L}(D)$.

Proof idea

Construct a TM D which simulates N by trying all possible branches of N's nondeterministic computation in breadth-first fashion. If D finds the accept state on one of these branches, it accepts. Otherwise, the simulation will reject or not terminate. $\square$

Multitape Turing machines (Section 7.5 in ILTC)

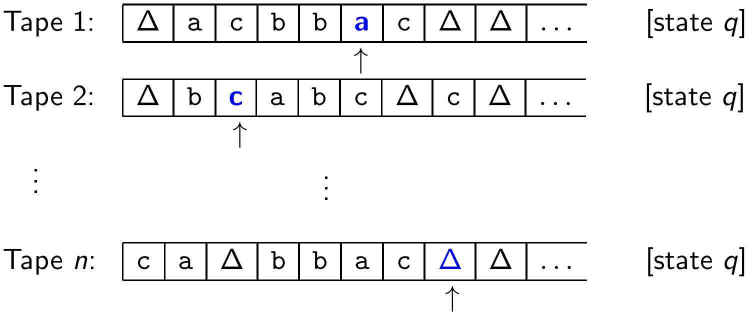
A *multitape Turing machine* (MTM) is defined like a Turing machine, except for the transition function

$$\delta\colon (Q - \{\mathrm{h_a},\, \mathrm{h_r}\}) \times \Gamma^n \to Q \times \Gamma^n \times \{\mathrm{L},\, \mathrm{R},\, \mathrm{S}\}^n,$$

where $n \geq 2$. Intuitively, the machine has n tapes on which it works simultaneously.

A *configuration* is an n-tuple $(u_1 q v_1,\, u_2 q v_2,\, \ldots,\, u_n q v_n)$, where $q \in Q$ and $u_i, v_i \in \Gamma^*$. The initial configuration for an input w is $(q_0 \Delta w,\, q_0 \Delta,\, \ldots,\, q_0 \Delta)$.

Example: snapshot of an n-tape TM



Multitape vs ordinary Turing machines

Theorem

For every multitape Turing machine N there exists a Turing machine M such that $\mathrm{L}(N) = \mathrm{L}(M)$.

Proof.

Proof of Thm 7.26 in ILTC (for the case of 2-tape machines). $\square$

Semidecidable languages (Chapter 8 in ILTC)

A language L is *semidecidable* (or *recursively enumerable* or of *type 0*) if there exists a Turing machine that accepts L.

Theorem

A language L is semidecidable if and only if L is generated by some (unrestricted) grammar.

Proof.

"If": Proof of Thm 8.13 in ILTC.

"Only if": Proof of Thm 8.14 in ILTC. $\square$

The Chomsky hierarchy

Type	Grammars/ Languages	Grammar productions	Machines
0	*Unrestricted*/ *semidecidable*	$\alpha \to \beta$ $[\alpha \in (V \cup \Sigma)^+,$ $\beta \in (V \cup \Sigma)^*]$	*Turing machine* (deterministic or nondeterministic)
1	*Context-sensitive*	$\alpha \to \beta$ $[\alpha, \beta \in (V \cup \Sigma)^+,$ $\lvert\alpha\rvert \leq \lvert\beta\rvert]$	*Linear-bounded automaton*
2	*Context-free*	$A \to \beta$ $[A \in V,\ \beta \in (V \cup \Sigma)^*]$	*Pushdown automaton*
3	*Regular*	$A \to aB,\ A \to \Lambda$ $[A, B \in V,\ a \in \Sigma]$	*Finite automaton* (deterministic or nondeterministic)

Enumerating languages by MTMs (Section 8.2 in ILTC)

A multitape Turing machine *enumerates* a language L if

(1) the computation begins with all tapes blank,

(2) the tape head on tape 1 never moves to the left,

(3) at each point in the computation, the contents of tape 1 has the form $\Delta\#w_1\#w_2\#\ldots\#w_n\#v$ where $n \geq 0$, $w_i \in L$, $\# \in (\Gamma - \Sigma)$ and $v \in \Sigma^*$,

(4) every $w \in L$ will eventually appear as one of the strings w_i on tape 1.

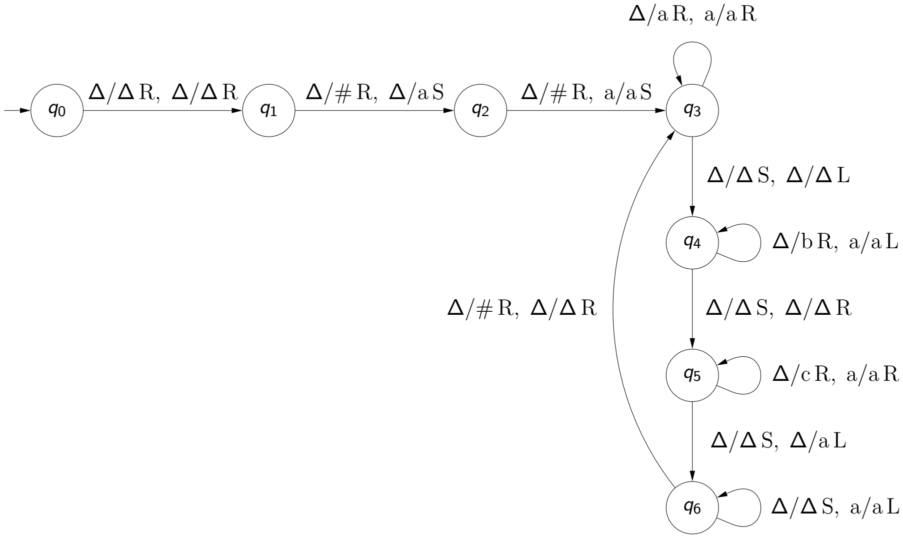
(The definition in ILTC is slightly different but equivalent.)

Enumerating languages by MTMs (Section 8.2 in ILTC)

Notes:

- There is no input
- Tape 1 is the *output tape*
- The listing may contain repeated strings
- (3) is *soundness*: every listed string belongs to L
- (4) is *completeness*: every string in L will eventually be listed

Example: 2-tape TM enumerating $\{a^n b^n c^n \mid n \geq 0\}$



Example: 2-tape TM enumerating $\{a^n b^n c^n \mid n \geq 0\}$

Notes:

- The initial transitions $q_0 \to q_1 \to q_2 \to q_3$ generate ## on Tape 1 (empty string) and a on Tape 2
- Tape 2 is used as a counter holding the current n
- The remaining transitions generate strings in rounds; in round n, Tape 2 contains $\mathtt{a}^n$ and $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ is produced on Tape 1 by scanning the string on Tape 2 three times: left $\to$ right $\to$ left $\to$ right; each scan is synchronised with the writing of $\mathtt{a}^n$, $\mathtt{b}^n$ or $\mathtt{c}^n$ on Tape 1 (by the loop $q_3 \to q_3$, $q_4 \to q_4$ or $q_5 \to q_5$)
- $q_5 \to q_6$ adds an a to Tape 2
- The loop $q_6 \to q_6$ rewinds the head on Tape 2

Example: 2-tape TM enumerating $\{a^n b^n c^n \mid n \geq 0\}$

Generation of $\#\#\mathtt{abc}\#$:

$$
\begin{array}{rcl}
(q_0\Delta,\ q_0\Delta) & \vdash & (\Delta q_1\Delta,\ \Delta q_1\Delta) \\
& \vdash & (\Delta\# q_2\Delta,\ \Delta q_2\mathtt{a}) \\
& \vdash & (\Delta\#\# q_3\Delta,\ \Delta q_3\mathtt{a}) \\
& \vdash & (\Delta\#\#\mathtt{a} q_3\Delta,\ \Delta\mathtt{a} q_3\Delta) \\
& \vdash & (\Delta\#\#\mathtt{a} q_4\Delta,\ \Delta q_4\mathtt{a}\Delta) \\
& \vdash & (\Delta\#\#\mathtt{ab} q_4\Delta,\ q_4\Delta\mathtt{a}\Delta) \\
& \vdash & (\Delta\#\#\mathtt{ab} q_5\Delta,\ \Delta q_5\mathtt{a}\Delta) \\
& \vdash & (\Delta\#\#\mathtt{abc} q_5\Delta,\ \Delta\mathtt{a} q_5\Delta) \\
& \vdash & (\Delta\#\#\mathtt{abc} q_6\Delta,\ \Delta q_6\mathtt{aa}) \\
& \vdash & (\Delta\#\#\mathtt{abc} q_6\Delta,\ q_6\Delta\mathtt{aa}) \\
& \vdash & (\Delta\#\#\mathtt{abc}\# q_3\Delta,\ \Delta q_3\mathtt{aa})
\end{array}
$$

Enumeration vs acceptance

Theorem (Thm 8.9 in ILTC)

A language L is enumerated by some multitape Turing machine if and only if L is semidecidable.