COMP30026 Models of Computation

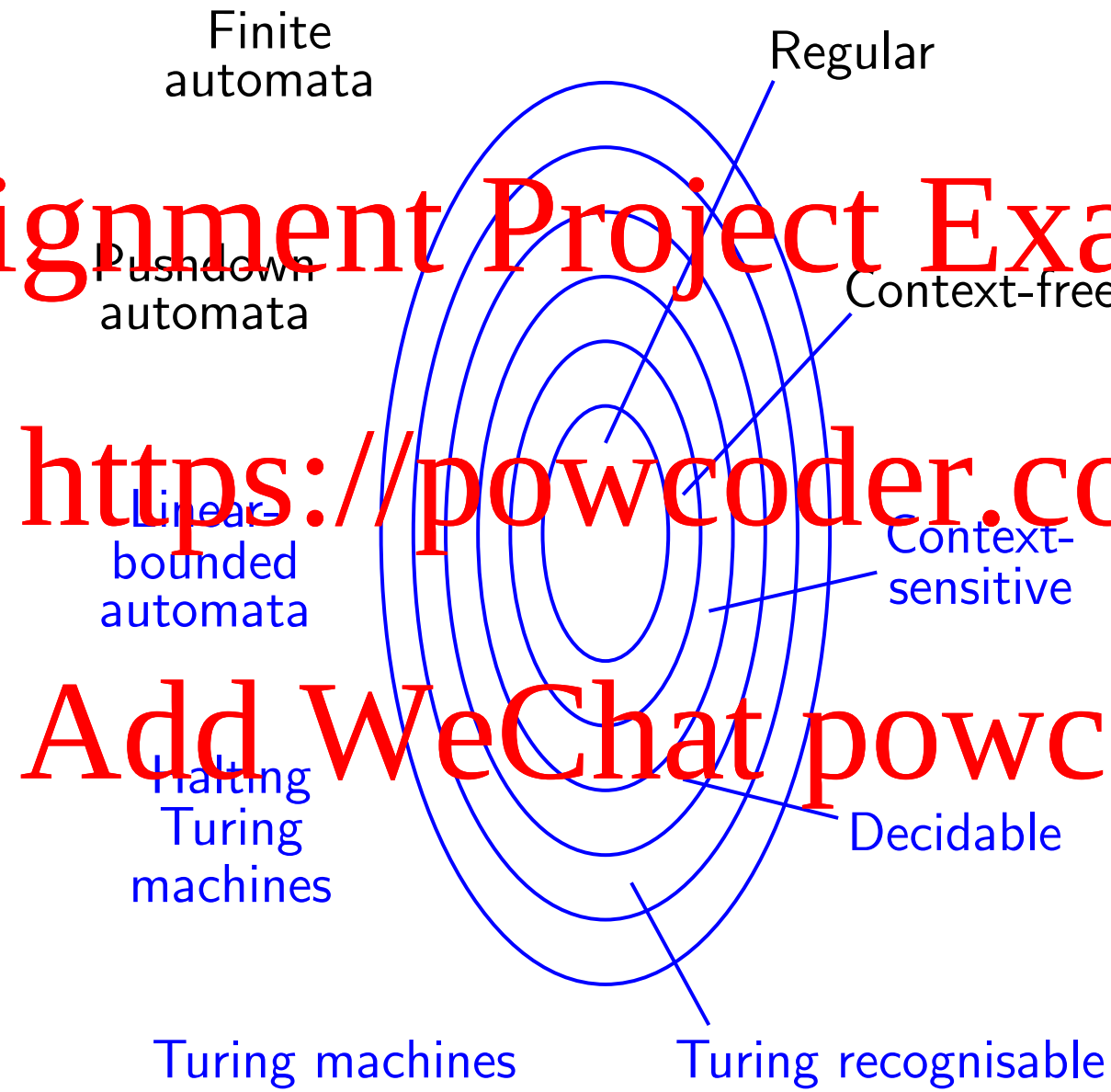
Turing Machines and Termination

Bach Le / Anna Kalenkova

Lecture Week 10

Semester 2, 2021

Chomsky Hierarchy Again



Context-Sensitive Grammars (Not Examinable)

A context-sensitive grammar G is a 4-tuple (V, Σ, R, S), where

1. V is a finite set of variables,
2. Σ is a finite set of terminals,
3. R is a finite set of rules,
4. S is the start variable, and
5. Each rule is of the form (1) $x \to y$, where $x, y \in (V \cup \Sigma)^+$ and $|x| \leq |y|$, or (2) $S \to \epsilon$.

Let $u, v, w \in (V \cup \Sigma)^*$. Then $uxw \Rightarrow uyw$ if $x \to y$ is a rule in R.

$$L(G) = \{s \in \Sigma^* \mid S \overset{*}{\Rightarrow} s\}$$

A language which can be generated by some context-sensitive grammar is a context-sensitive language.

Context-Sensitive Grammars (Not Examinable)

A context-sensitive grammar G is a 4-tuple (V, Σ, R, S), where

1. V is a finite set of variables,
2. Σ is a finite set of terminals,
3. R is a finite set of rules,
4. S is the start variable,
5. Each rule is of the form (1) $x \to y$, where $x, y \in (V \cup \Sigma)^+$ and $|x| \leq |y|$, or (2) $S \to \epsilon$.

Theorem: A context-free language can be generated by a context-sensitive grammar.

Context-Sensitive Grammars. (Not Examinable)

This context-sensitive grammar generates $L = \{a^n b^n c^n | n \geq 1\}$:

$$
\begin{array}{lcl}
S & \rightarrow & aSBC \mid aBC \\
CB & \rightarrow & BC \\
aB & \rightarrow & ab \\
bB & \rightarrow & bb \\
bC & \rightarrow & bc \\
cC & \rightarrow & cc
\end{array}
$$

$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow aaaBBCCBC \Rightarrow aaaBBCBCC \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \Rightarrow aaabbBCCC \Rightarrow aaabbbCCC \Rightarrow aaabbbcCC \Rightarrow aaabbbccC \Rightarrow aaabbbccc.$

Context-Sensitive Grammars. (Not Examinable)

A context-sensitive grammar G is a 4-tuple (V, Σ, R, S), where

1. V is a finite set of variables,
2. Σ is a finite set of terminals,
3. R is a finite set of rules,
4. S is the start variable,
5. Each rule is of the form (1) $x \to y$, where $x, y \in (V \cup \Sigma)^+$ and $|x| \leq |y|$, or (2) $S \to \epsilon$.

Let $u, v, w \in (V \cup \Sigma)^*$. Then $uxw \Rightarrow uyw$ iff $x \to y$ is a rule in R.

$$L(G) = \{s \in \Sigma^* \mid S \overset{*}{\Rightarrow} s\}$$

Context-sensitive languages are recognised by linear bounded automata (Turing machines with a tape of a bounded finite length).

Unrestricted Grammars. (Not Examinable)

An unrestricted grammar G is a 4-tuple (V, Σ, R, S), where

1. V is a finite set of variables,
2. Σ is a finite set of terminals,
3. R is a finite set of rules,
4. S is the start variable.

Let $u, v, w \in (V \cup \Sigma)^*$. Then $uxw \Rightarrow uyw$ iff $x \to y$ is a rule in R.

$$L(G) = \{s \in \Sigma^* \mid S \overset{*}{\Rightarrow} s\}$$

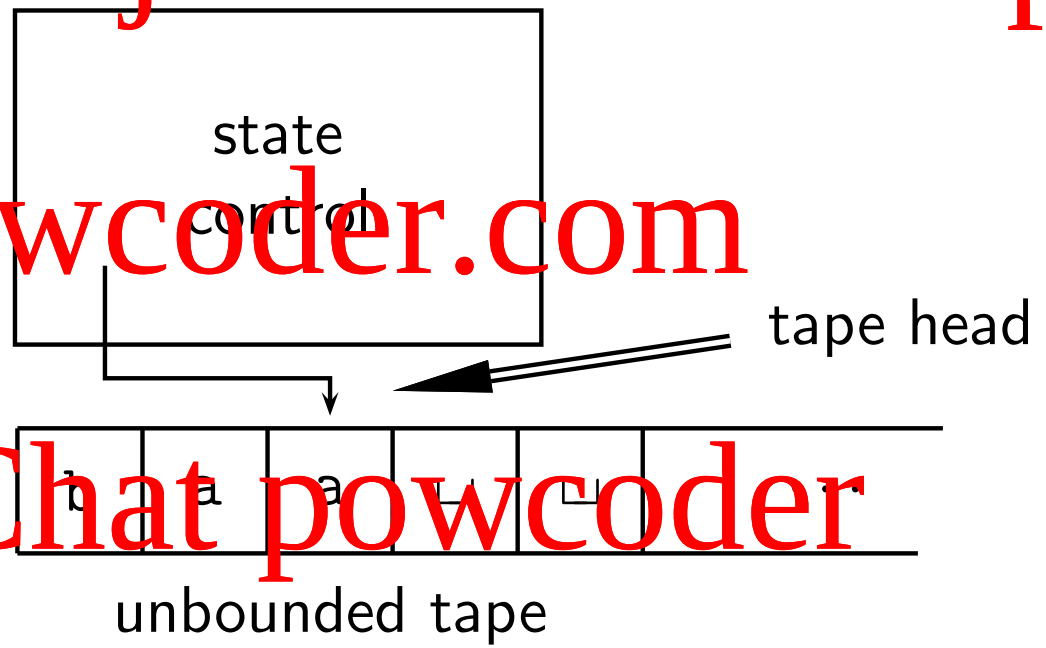
Languages generated by unrestricted grammars are recognised by Turing machines (Turing recognisable).

Turing Machines

A Turing machine has an unbounded tape through which it takes its input and performs its computations.

Unlike our previous automata it can:

- both read from and write to the tape, and
- move either left or right over the tape.



The machine has distinct accept and reject states, in which it accepts/rejects irrespective of where its tape head is.

Turing Machines Formally

A Turing machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ where

- Q is a finite set of states,
- Γ is a finite tape alphabet, which includes the blank character, $\sqcup$,
- $\Sigma \subseteq \Gamma \setminus \{\sqcup\}$ is the input alphabet,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- q_0 is the initial state,
- q_a is the accept state, and
- q_r ($\neq q_a$) is the reject state.

The Transition Function

A transition $\delta(q_i, x) = (q_j, y, d)$ depends on two things

1. current state q_i, and
2. current symbol x under the tape head

It consists of three actions

1. change state to q_j,
2. over-write tape symbol x by y, and
3. move the tape head in direction d.

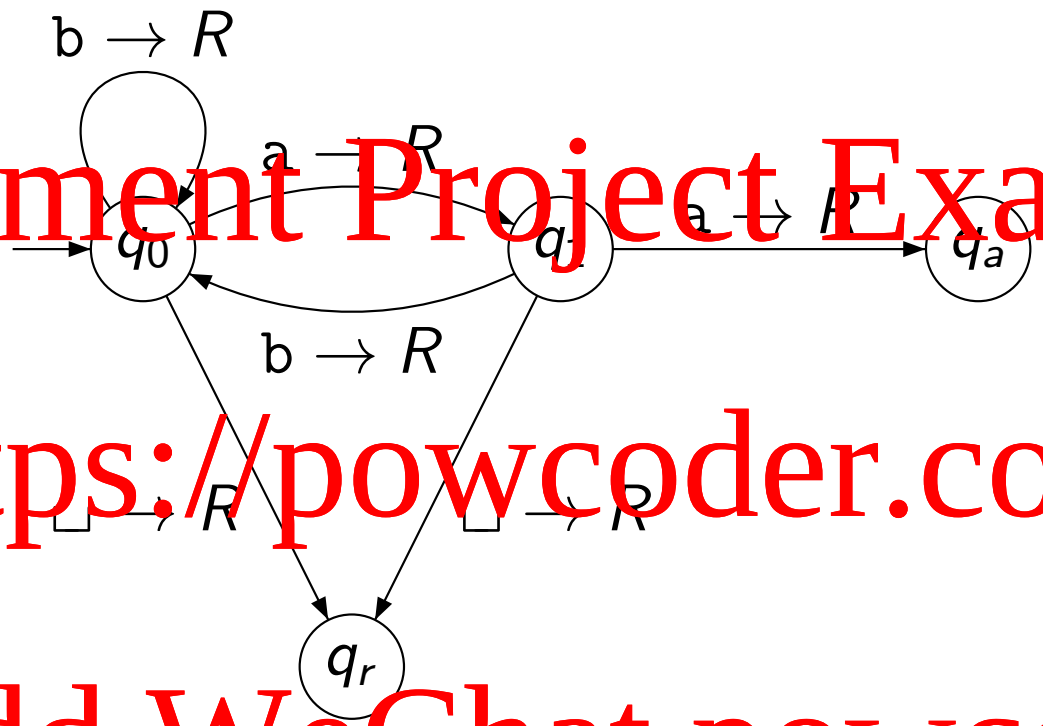
Drawing Turing Machines

We can have a graphical notation for Turing machines similar to that for finite automata.

On an arrow from q_i to q_j we write

- $x \to d$ when $\delta(q_i, x) = (q_j, x, d)$, and
- $x \to y, d$ when $\delta(q_i, x) = (q_j, y, d)$ $y \neq x$.

Turing Machine Example 1

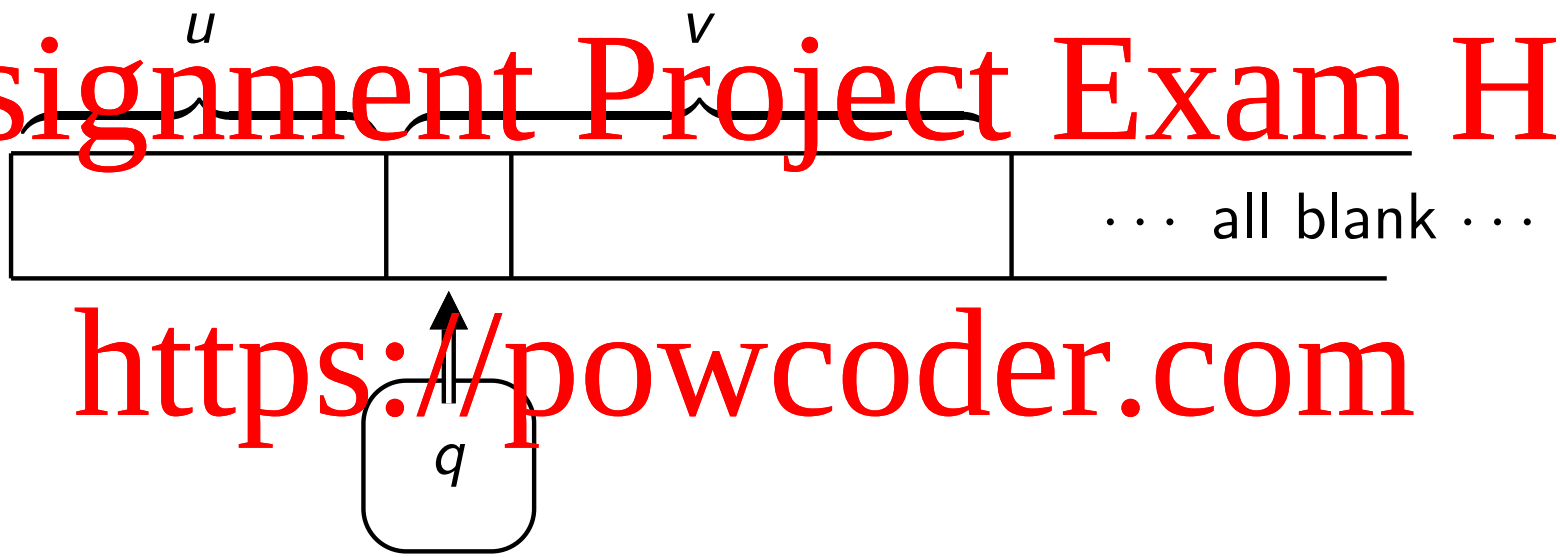


This machine recognises the regular language $(\mathtt{a} \cup \mathtt{b})^*\mathtt{aa}(\mathtt{a} \cup \mathtt{b})^*$.

We can leave the reject state q_r out with the understanding that transitions that are not specified go to q_r.

Turing Machine Configurations

We write uqv for this configuration:



On input aba, the example machine goes through 5 configurations:

$$
\begin{aligned}
& \epsilon q_0 \mathtt{aba} \qquad \text{(or just } q_0\mathtt{aba}\text{)} \\
\Rightarrow\ & \mathtt{a} q_1 \mathtt{ba} \\
\Rightarrow\ & \mathtt{ab} q_0 \mathtt{a} \\
\Rightarrow\ & \mathtt{aba} q_1 \sqcup \\
\Rightarrow\ & \mathtt{aba} \sqcup q_r
\end{aligned}
$$

Computations Formally

For all $q_i, q_j \in Q$, $a, b, c \in \Gamma$, and $u, v \in \Gamma^*$, we have

$$\begin{aligned} uq_ibv &\Rightarrow ucq_jv && \text{if } \delta(q_i, b) = (q_j, c, R) \\ q_ibv &\Rightarrow q_jcv && \text{if } \delta(q_i, b) = (q_j, c, L) \\ uaq_ibv &\Rightarrow uq_jacv && \text{if } \delta(q_i, b) = (q_j, c, L) \end{aligned}$$

The start configuration of M on input w is q_0w.

M accepts w iff there is a sequence of configurations $C_1, C_2, \ldots, C_k$ such that

1. C_1 is the start configuration q_0w,
2. $C_i \Rightarrow C_{i+1}$ for all $i \in \{1 \ldots k-1\}$, and
3. The state of C_k is q_a.

Turing Machines and Languages

The set of strings accepted by M is the language of M, $L(M)$.

A language A is Turing recognisable (or recursively enumerable, or just r. e.) iff $A = L(M)$ for some Turing machine M.

Note carefully that three behaviours are possible for M on input w: M may accept w, reject w, or fail to halt.

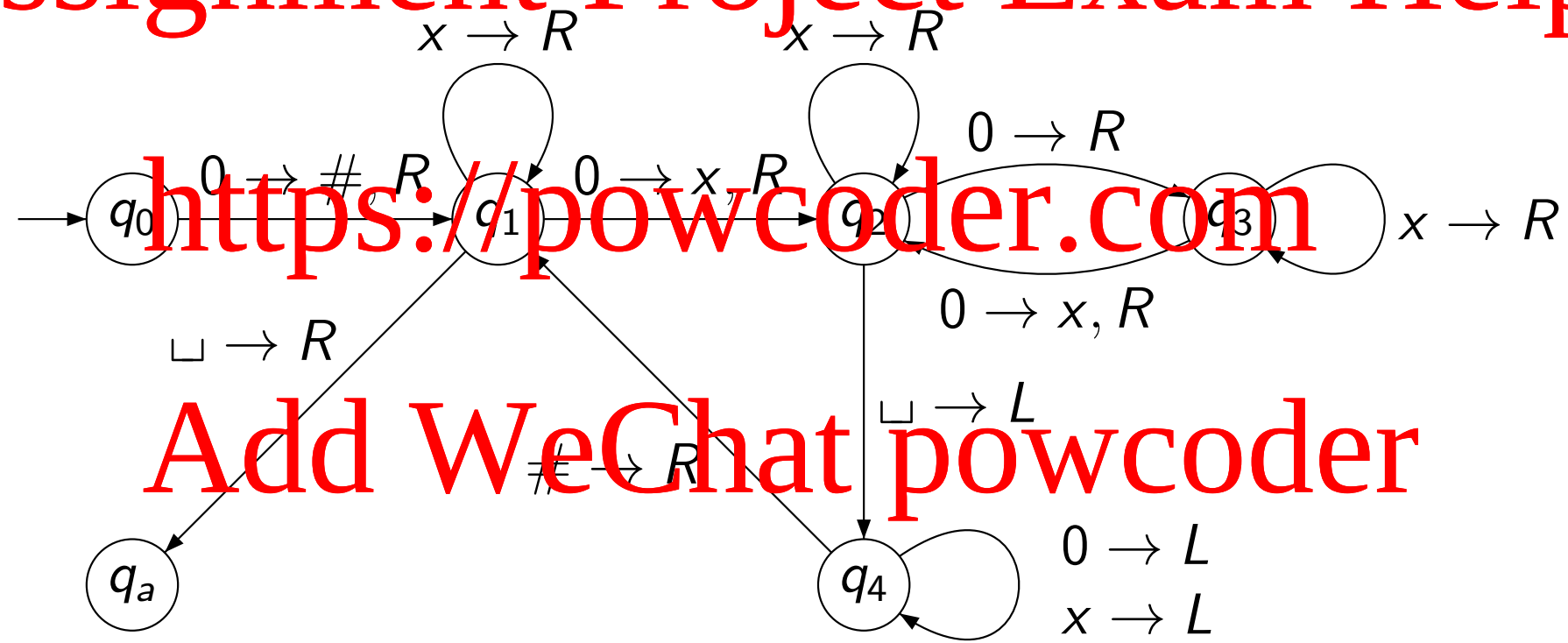
If A is recognised by a Turing machine M that halts on all input, we say that M decides A.

A language is Turing decidable (or recursive, or just decidable) iff some Turing machine decides it.

Turing Machine Example 2

This machine decides the language $\{0^{2^n} \mid n \geq 0\}$.

It has input alphabet $\{0\}$ and tape alphabet $\{\sqcup, 0, x, \#\}$.



Running the machine on input 000:

$$q_0000 \Rightarrow \#q_100 \Rightarrow \#xq_20 \Rightarrow \#x0q_3\sqcup \Rightarrow \#x0\sqcup q_r$$

The Versatility of Turing Machines

We can decide that a Turing machine produces output (not just accept/reject) through its tape. This way a Turing machine can be a general computing device, not just a language recogniser.

We can capture data other than strings via suitable representations.

For example, natural numbers can be represented as strings of (unary, binary, decimal, ...) digits, so a Turing machine can compute number theoretic functions $\mathbb{N} \to \mathbb{N}$.

Or, by suitable encoding, it can take multiple arguments, and/or return multiple results.

A Turing machine can also solve graph problems, once we decide on a suitable representation for graphs.

Robustness of Turing Machines

Different books use different definitions of Turing machines.

Most differences are minute and technical and aim at making the machines easier to program (for example, we may insist that machines start with a tape that has the first cell blank, and they try to leave that cell blank—to make it easier to compose machines).

Similarly, in addition to the two kinds of tape movement, we can allow a 'no move' option.

Turing machines are robust in the sense that such changes to the machinery do not affect what the machines are capable of computing.

Variants of Turing Machines

In an attempt to make the Turing machine more powerful we could add more radically to its features:

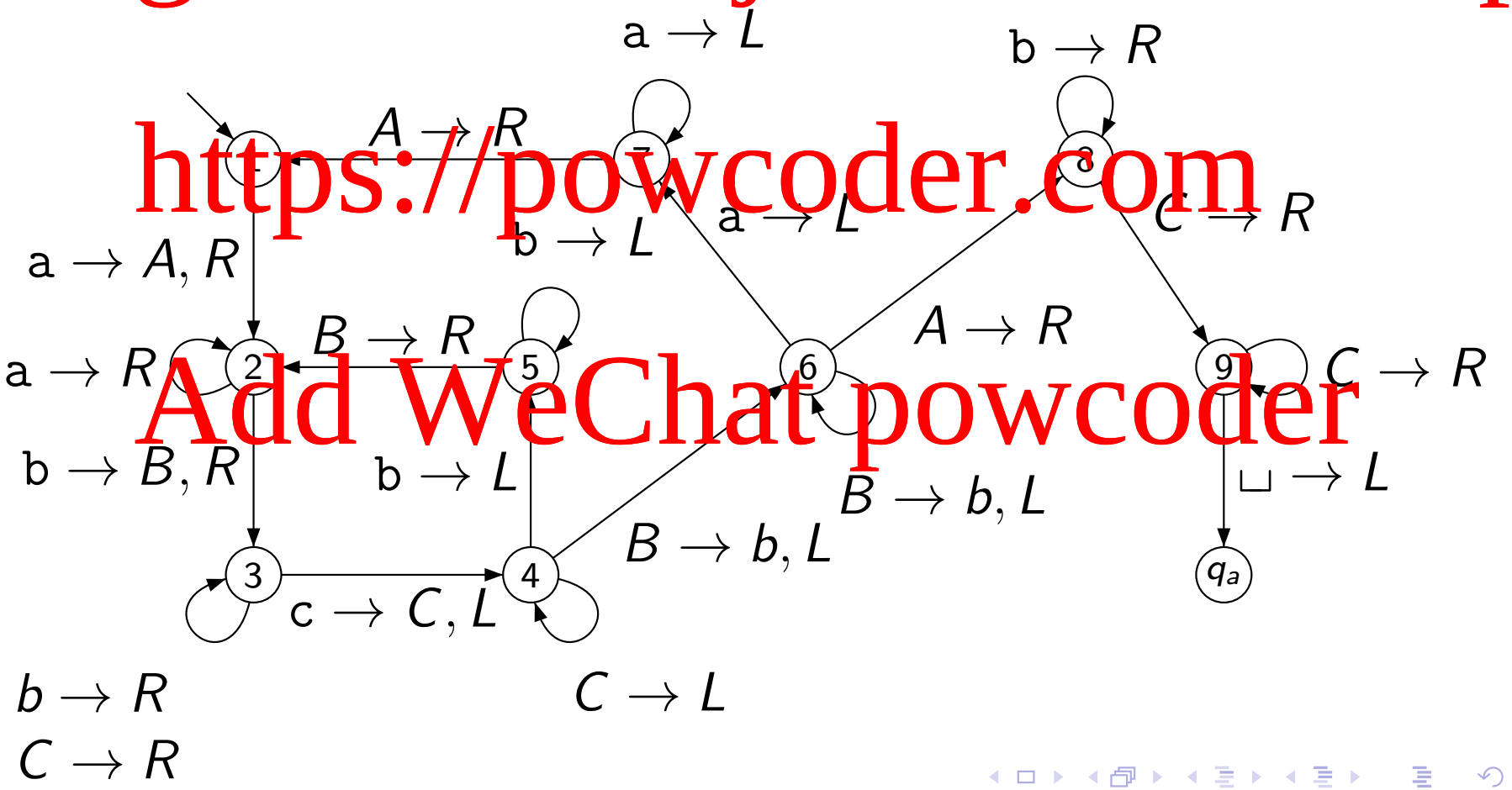
- Let its tape extend indefinitely in both directions.
- Let its tape have multiple tracks.
- Let there be several tapes, each with its independent tape head.
- Add nondeterminism.

However, none of this increases a Turing machine's capabilities as a language recogniser.

Turing Machine Example 3

This machine decides the language $\{a^i b^j c^k \mid k = i \cdot j, i, j > 0\}$.

Its tape alphabet is $\{\sqcup, a, b, c, A, B, C\}$.



Exercise

Design a Turing machine with input alphabet $\{a, b, c\}$ which decides the language: $A = \{a^i b^j c^k \mid i, j, k > 0 \wedge i + j \geq k\}$.