Decision Problems as Languages

A *decision problem* is the problem of determining an answer to a class of yes/no questions about some objects of interest.

Examples of objects of interest:

- graphs
- DFAs
- CFGs
- TMs
- integers
- Boolean formulas

Examples of yes/no questions:

- Does this graph have a path that goes through every node exactly once?
- Are these two graphs isomorphic?
- Are these two DFAs equivalent?
- Does this TM accept this string?
- Is the language this TM recognizes regular?
- Does this TM halt on this input?
- Is this number prime?
- Is there a set of *true/false* values for the variables in this Boolean formula that makes the formula *true*?

A solution to a decision problem is an algorithm (i.e., a Turing Machine) that provides an answer to all yes/no questions in that specified class.

For a given decision problem:

- *Computability issue:* Does it have a solution at all? That is, is there any algorithm that can answer all yes/no questions in that class?
- *Complexity issue:* Does it have an *efficient* solution? That is, is there an algorithm that can answer all yes/no questions in that class and whose running time scales well with input size?

Our strategy:

1. Represent objects of interest as strings.
2. Formulate a given question as the question of membership in a corresponding language. (The string is in the language iff the answer to the question for that object is *yes*.)
3. Study the existence (and/or efficiency) of algorithms to answer the question in general (i.e., to solve that decision problem) by examining TMs that recognize or decide the corresponding language.

Notation For Objects Encoded as Strings

Single objects:

- Let O be some object of interest (e.g., graph, DFA, TM).
- Then its encoding as a string is denoted $\langle O \rangle$.

Multiple objects:

- Sometimes we want to construct an algorithm (i.e., TM) that takes as input a combination of several objects, $O_1, O_2, \ldots, O_n$.
- Then we encode the entire combination as a single string denoted $\langle O_1, O_2, \ldots, O_n \rangle$.

Example:

- Consider the decision problem: *Given a CFG and a string, does the CFG generate the string?*
- Input string to a TM to solve this decision problem is denoted $\langle G, w \rangle$, where G is the given CFG and w is the given string.

Technically:

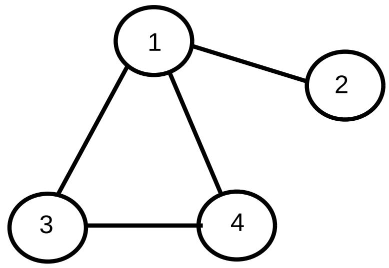
- $\langle \_, \_, \ldots, \_ \rangle$ represents a function from the Cartesian product of the classes to which the relevant objects belong to the set of strings over some alphabet.
- We want the individual objects $O_1, O_2, \ldots, O_n$ to be "recoverable" from the string $\langle O_1, O_2, \ldots, O_n \rangle$, which means this function must be one-to-one.
- One sensible approach having this property:

  $$\langle O_1, O_2, \ldots, O_n \rangle = \langle O_1 \rangle \# \langle O_2 \rangle \# \ldots \# \langle O_n \rangle ,$$

  where each string $\langle O_i \rangle$ is itself obtained in a one-to-one manner from O_i.

Encoding Objects as Strings
Example 1: Finite Undirected Graphs

Consider the following finite undirected graph G:



One way to encode this as a string is as a list of all its nodes followed by a list of all its edges, as follows:

$$\langle G \rangle = (1,2,3,4)((1,2),(1,3),(1,4),(3,4))$$

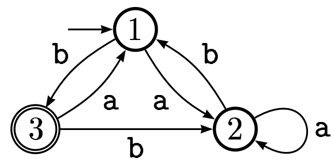
The alphabet used is $\Sigma = \{(,), \langle\text{comma}\rangle, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$.
(Use a decimal representation to give each node a unique number.)

Actually, this could be done with just a 2-symbol alphabet.

Can you see how?

Encoding Objects as Strings
Example 2: DFAs

Consider the following DFA D:



This table represents D's transition function δ:

	a	b
1	2	3
2	2	1
3	1	2

We can encode any DFA by concatenating:

- a list of its states (identified with decimal numbers)
- a list of the alphabet symbols
- a list of the rows of the transition table
- its start state
- a list of its accept states

For this example we get

$$\langle D \rangle = (1,2,3)(\mathtt{a},\mathtt{b})((2,3),(2,1),(1,2))1(3)$$

Encoding Objects as Strings
Example 3: CFGs

Consider the following CFG G:

$$\begin{array}{rcl} S & \rightarrow & XSY \mid \varepsilon \\ X & \rightarrow & \texttt{a}X \mid \texttt{a} \\ Y & \rightarrow & \texttt{b}Y \mid \varepsilon \end{array}$$

We can encode any CFG by concatenating:

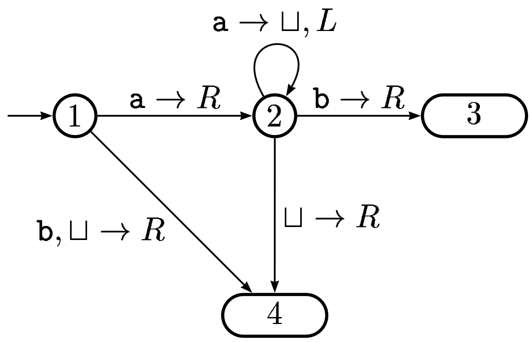
- a list of its variables
- a list of its terminal symbols
- a list of its rules as (LHS,RHS) pairs
- its start variable

For this example we get

$$\langle G \rangle = (S, X, Y)(\texttt{a}, \texttt{b})((S, XSY), (S, \varepsilon), (X, \texttt{a}X), (X, \texttt{a}), (Y, \texttt{b}Y), (Y, \varepsilon))S$$

Encoding Objects as Strings
Example 4: TMs

Consider the following TM T:



where 3 is the accept state and 4 is the reject state.

T's transition function δ is represented by the following table:

	$\mathtt{a}$	$\mathtt{b}$	$\sqcup$
1	$(2, \mathtt{a}, R)$	$(4, \mathtt{b}, R)$	$(4, \sqcup, R)$
2	$(2, \sqcup, L)$	$(3, \mathtt{b}, R)$	$(4, \sqcup, R)$

We can encode any TM by concatenating:

- a list of its states
- a list of its input alphabet symbols
- a list of its tape alphabet symbols
- a list of the rows of its transition table
- its start state
- its accept state
- its reject state

(using additional commas as delimiters if necessary).

For this example we get

$$\langle T \rangle = (1,2,3,4)(\mathtt{a},\mathtt{b})(\mathtt{a},\mathtt{b},\sqcup)(((2,\mathtt{a},R),(4,\mathtt{b},R),(4,\sqcup,R))((2,\sqcup,L),(3,\mathtt{b},R),(4,\sqcup,R)))1,3,4$$