Léonard Brice, Mathieu Sassolas
leonard.brice@ulb.be, mathieu.sassolas@ulb.be

Session 1:
Languages, problems, and Turing machines

Reminders

An *alphabet* is a set Σ, always assumed to be finite. A *word* over Σ is a finite sequence $w = w_1 \dots w_n$, with $w_1, \dots, w_n \in \Sigma$. The set of words over Σ is written Σ^*. A *language* over Σ is a subset $L \subseteq \Sigma^*$.

A *decision problem* is an ordered pair (I, P), where $I \subseteq \Sigma^*$ is a set of *instances* (or *inputs*), and $P \subseteq I$ is a the set of *positive instances*. Intuitively, a decision problem is a yes-or-no question asked over a set of instances.

Instances of decision problems are always words. For problems about other types of objects, it is important to know how such objects are *encoded* as words (e.g., integers will usually be encoded as a sequence of 0s and 1s).

A decision problem is assimilated to the language of its positive instances.

A *Turing machine* is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$, where:

- Q is a set of *states*,
- Σ is the *input alphabet*,
- Γ is the *tape alphabet*, containing the letter $\_$ and the alphabet Σ,
- $\delta : (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\} \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$ is a *transition function*,
- $q_0 \in Q$ is the initial state, $q_{\mathsf{acc}} \in Q$ is the *accepting state* and $q_{\mathsf{rej}} \in Q$ is the *rejecting state* (satisfying $q_{\mathsf{acc}} \neq q_{\mathsf{rej}}$).

A *configuration* of the machine M is triple $(u, q, v) \in \Gamma^* \times Q \times \Gamma^+$. Intuitively, the configuration (u, q, v) must be understood as follows: the machine M is in the state q, the content of its tape is the word uv, and the reading head is on the first letter of v. The *computation* of the machine M on the word w is the sequence: $(u_0, q_0, v_0), (u_1, q_1, v_1), \dots$ of configurations of M satisfying:

- $u_0 = \varepsilon$ and $v_0 = w$ (the machine starts in the the state q_0, with the word w written on the tape, and the reading head on its first letter);
- for every i, we have $v_i = av$ with $a \in \Gamma$ and $v \in \Gamma^*$ such that either $\delta(q_i, a) = (q_{i+1}, b, \mathsf{R})$ and $u_{i+1} = u_i b$ and $v_{i+1} = v$ (the machine reads the letter a, writes the letter b, and moves the reading head to the right), or $\delta(q_i, a) = (q_{i+1}, b, \mathsf{L})$, $u_i = uc$ with $c \in \Gamma$ and $u \in \Gamma^*$, $u_{i+1} = u$ and $v_{i+1} = cbv$ (the machine reads the letter a, writes the letter b, and moves the reading head to the left);
- and the sequence is either infinite (the machine does not *terminate*), or ends with a configuration (u_n, q_n, v_n) such that $q_n \in \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}$ (the machine *accepts* w or *rejects* it)[1].

A language L is *recognized* by a Turing machine if every word $w \in \Sigma^*$ is accepted by M if and only if it belongs to L (a word that does not belong to L may be rejected, or the computation of M on it may be infinite). The language L is *decided* by M if every word $w \in L$ is accepted by M, and every word $w \in \Sigma^* \setminus L$ is rejected by M (in other words, the machine M decides L if it recognizes L *and* terminates on every word).

[1]The function δ can also be defined as a partial function — the non-existence of $\delta(q, a)$ is then equivalent to a transition to the state q_{rej}. Such a definition is heavy for formal works, but is often used for examples, since transitions to the rejecting state can then be omitted.

Less formally, a Turing machine is an abstract machine that uses an infinite tape, and whose transition function says, when reading some letter a while being in some state q, what should be written instead of a, to which state should the machine shift, and whether the reading head should move to the left or to the right.

A *multi-tape Turing machine* is defined as a Turing machine, but with n tapes instead of 1. A transition reads then n letters $a_1, \ldots, a_n$, decides which letter should be written instead of each of those, and for each of the n reading heads, whether it should move to the left, to the right, or stay where it is.

A *non-deterministic Turing machine* is defined as a Turing machine, but several transitions may be available from the same state, reading the same letter. A word w is accepted by a non-deterministic machine M if one of the possible runs of M on w is accepting.

The *Church-Turing* thesis states that a decision problem can be decided by an algorithm if and only if it is decided by a Turing machine. It is not a theorem, because algorithms are an intuitive notion, linked to everyday life, not a formal notion. By linking it to a formal notion, the Church-Turing thesis must be understood as the *raison d'être* of the theory of computability and complexity.

1 Encoding problems as languages

1. Propose a way to encode each of these objects with the given alphabet. For example, an integer $n \in \mathbb{N}$ can be encoded on the alphabet $\{\mathtt{0}, \mathtt{1}\}$ by using the classical binary encoding.

   (a) A pair (u, v), where $u, v \in \{\mathtt{a}, \mathtt{b}\}^*$, as a word on the alphabet $\{\mathtt{a}, \mathtt{b}, \#\}$;

   (b) a list $(u_1, \ldots, u_n)$ of words on the alphabet $\{\mathtt{a}, \mathtt{b}\}$, as a word on the alphabet $\{\mathtt{a}, \mathtt{b}, \#\}$;

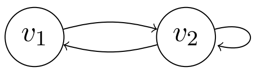
   (c) a word $w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^*$, as a word on the alphabet $\{\mathtt{0}, \mathtt{1}\}$;

   (d) an integer $z \in \mathbb{Z}$, as a word on the alphabet $\{\mathtt{0}, \mathtt{1}\}$;

   (e) a matrix M of natural integers, as a word on the alphabet $\{\mathtt{1}, \mathtt{0}, \#\}$;

   (f) a picture, as a word on the alphabet $\{\mathtt{1}, \mathtt{0}, \#\}$;

   (g) a directed graph, as a word on an alphabet of your choosing. What is your encoding for the following graph, with 2 vertices?



2. Give the language associated to each of the following problems, under the encoding you proposed in the previous question. For example, the language associated to the problem "Given a natural integer $n \in \mathbb{N}$, is n even?" is $L = \{w \cdot \mathtt{0} \mid w \in \{\mathtt{0}, \mathtt{1}\}^*\}$.

   (a) Given a pair (u, v) of words over the alphabet $\{\mathtt{a}, \mathtt{b}\}$, do we have $u = v$?

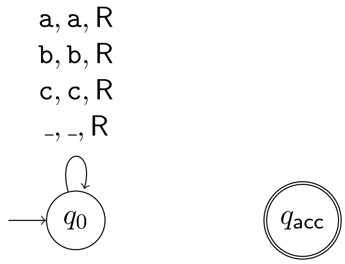
   (b) Given a word $w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^*$, does the letter $\mathtt{a}$ appear in w?

   (c) Given a directed graph G, is there a clique of size 2 in G? (That is, are there two vertices $u \neq v$ with an edge from u to v, and an edge from v to u?)

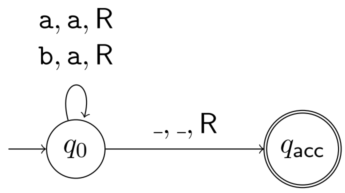
2 Turing machines

1. For each of the following Turing machines (with input alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ and tape alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{A}, \mathtt{B}\}$), give the language that it recognizes, tell whether it decides it, and describe its execution on the word $\mathtt{aab}$.

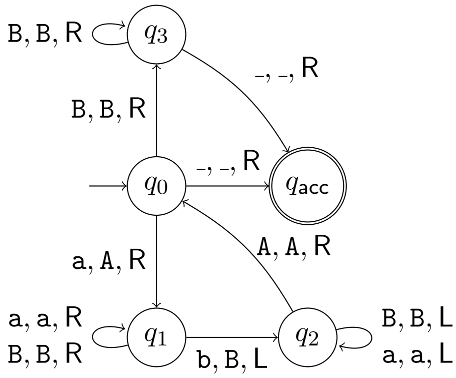
   (a) The Turing machine:



(b) The Turing machine:



(c) The Turing machine:



2. Give a Turing machine that recognizes the language of the words on the alphabet $\{\mathtt{a}, \mathtt{b}\}$ that have an even number of a's.

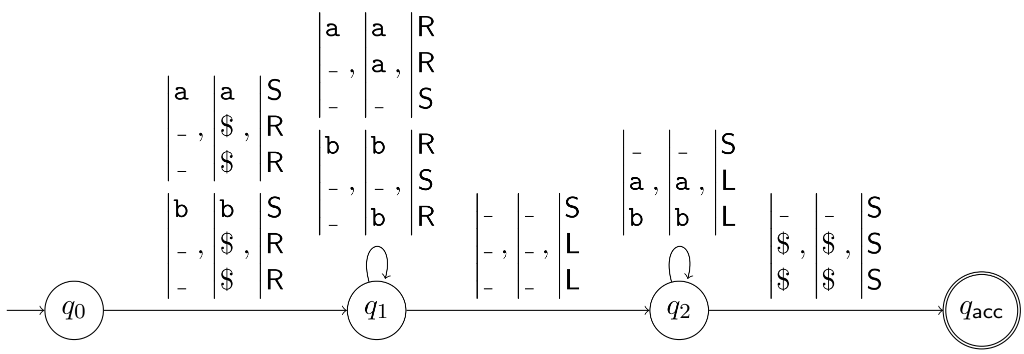
3. Give the language associated to the following problem:

   - input: a number n given in binary
   - question: Is n a multiple of 4?

   Give a Turing machine that decides that problem.

3 Variants

1. Which language does the following 3-tape Turing machine decide?



2. Give a 2-tape machine that decides the same language.

3. Is there a single-tape machine that decides the same language? Why?

4. Describe a non-deterministic Turing machine that decides the language $\{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$.

5. Among the following classes of languages, which one are included in which ones?

   (a) Languages decided by a deterministic Turing machine;
   (b) languages that can be enumerated by a deterministic Turing machine;
   (c) languages decided by a non-deterministic Turing machine;
   (d) languages recognized by a deterministic finite automaton;
   (e) languages recognized by a non-deterministic finite automaton;
   (f) languages recognized by a deterministic Turing machine;
   (g) languages recognized by a deterministic pusdhown automaton;
   (h) languages defined by a regular expression;
   (i) languages defined by a context-free grammar;
   (j) languages whose complement is recognized by a deterministic Turing machine;
   (k) languages that can be decided by a Python program;
   (l) languages that can be decided using a computer whose total memory (RAM, mass storage, ...) is at most 1000GB, and that receives the input at once
   (m) languages that can be decided using a computer whose total memory is at most 1000GB, and that receives the input letter by letter.