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

Session 1:
Languages, problems, and Turing machines

Solutions

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 $\{\mathtt{a}, \mathtt{b}, \#\}$;*

   ☛ The pair can be encoded by the word $u\#v$; for example, the pair $(\mathtt{aab}, \mathtt{ba})$ is encoded by the word $\mathtt{aab\#ba}$.

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

   ☛ Such a list can be encoded by the word $u_1\#u_2\#\ldots\#u_n$.

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

   ☛ Such a word can be encoded by replacing each occurrence of $\mathtt{a}$ by $\mathtt{00}$, of $\mathtt{b}$ by $\mathtt{01}$, of $\mathtt{c}$ by $\mathtt{10}$, and of $\mathtt{d}$ by $\mathtt{11}$.

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

   ☛ The integer z can be encoded by, first, a $\mathtt{1}$ if $z \geq 0$ or a $\mathtt{0}$ if $z < 0$, and then the binary encoding of $|z|$.

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

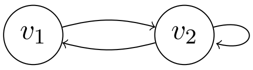
   ☛ A matrix can be seen as a list of lists, and therefore written using the encoding proposed in Question 1b. All the integers can be encoded using the usual binary encoding. Note that two different separation marks must be used: it can be, for example, the symbol # inside a list, and ## in the list of lists.

   For example, the matrix $\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$ is written $\mathtt{0\#1\#\#10\#11}$.

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

   ☛ A picture can be seen as a matrix of pixels, each pixel being itself a 3-tuple of integers giving an indication of how much red, green and blue it contains.

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



   ☛ Many answers are possible to this question. There are two canonical ones:

   - *the adjacency matrix:* a graph with n vertices, called $v_1, \ldots, v_n$, can be written using the encoding of Question 1e on the adjacency matrix $M = (e_{ij})_{i,j \in \{1,\ldots,n\}}$, where $e_{ij} = \mathtt{1}$ if there is an edge from v_i to v_j, and $\mathtt{0}$ otherwise;
   - *the adjacency list:* the same graph can be written, using the encoding of Question 1b, as a list of lists $(\ell_1, \ldots, \ell_n)$, where each for each i, j, the list ℓ_i contains the binary encoding of the integer j if and only if there is an edge from v_i to v_j.

   Both options have qualities and flaws: when the graph has many vertices and few edges, the second one is more compact. But, with the first one, it is always possible to know immediately whether there is an edge between two given vertices v_i and v_j, while with the second one, it may be necessary to read the whole list ℓ_i.

   With the first encoding, the example graph is written $\mathtt{0\#1\#\#1\#1}$. With the second encoding, it is written $\mathtt{10\#\#1\#10}$.

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 $u = v$?*
   ☛ $L = \{w\#w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$.

   (b) *Given a word $w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^*$, does the letter $\mathtt{a}$ appear in w?*
   ☛ $L = \{w_1 \cdot \ldots \cdot w_n \mid \forall i, w_i \in \{\mathtt{0}, \mathtt{1}\}^2 \text{ and } \exists i, w_i = \mathtt{00}\}$.

   (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?)*
   ☛ With our first encoding:

$$L = \{e_{11}\#e_{12}\# \ldots e_{1n}\#\#e_{21}\# \ldots \#e_{nn} \in \{\mathtt{0}, \mathtt{1}\}^* \mid n \in \mathbb{N}, \text{ and } \exists i \neq j, e_{ij} = 1 \text{ and } e_{ji} = 1\}.$$

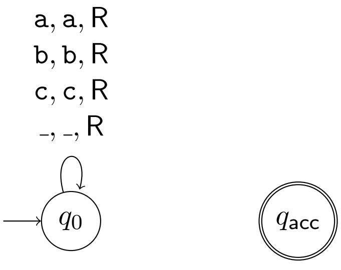
   With our second encoding:

$$L = \left\{ j_{11}\# \ldots \#j_{1k_1}\#\#j_{21}\# \ldots \#j_{nk_n} \;\middle|\; \begin{array}{c} \forall i, \ell, j_{i\ell} \in \mathbb{N} \\ \text{and } \exists i, \ell, i', \ell', j_{i\ell} = i' \text{ and } j_{i'\ell'} = i \end{array} \right\}.$$

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:*



   ☛ That machine recognizes the empty language. It does not decide it, since it does not stop, for example, on the word $\mathtt{aab}$:

$$(q_0)\ \overset{\downarrow}{\mathtt{a}}\ \mathtt{a}\ \mathtt{b}\ \_\ \_\ldots.$$

$$(q_0)\ \mathtt{a}\ \overset{\downarrow}{\mathtt{a}}\ \mathtt{b}\ \_\ \_\ldots.$$

$$(q_0)\ \mathtt{a}\ \mathtt{a}\ \overset{\downarrow}{\mathtt{b}}\ \_\ \_\ldots.$$

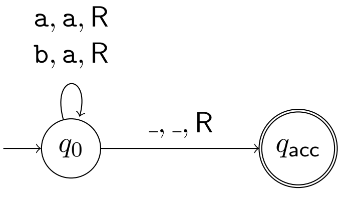
$$(q_0)\ \mathtt{a}\ \mathtt{a}\ \mathtt{b}\ \overset{\downarrow}{\_}\ \_\ldots.$$

$$(q_0)\ \mathtt{a}\ \mathtt{a}\ \mathtt{b}\ \_\ \overset{\downarrow}{\_}\ldots$$

$$(q_0)\ \mathtt{a}\ \mathtt{a}\ \mathtt{b}\ \_\ \_\ \overset{\downarrow}{\_}\ldots$$

   . . . and so on.

   (b) *The Turing machine:*



☛ This machine recognizes the language of words in which the letter `c` does never occur. It even decides it: since the reading head always go to the right, and since the machine accepts when it reads the blank symbol, it always terminates.

For example, on the word `aab`:

$$(q_0)\quad \overset{\downarrow}{\texttt{a}}\ \texttt{a}\ \texttt{b}\ \_\ \_\ .\ .\ .$$

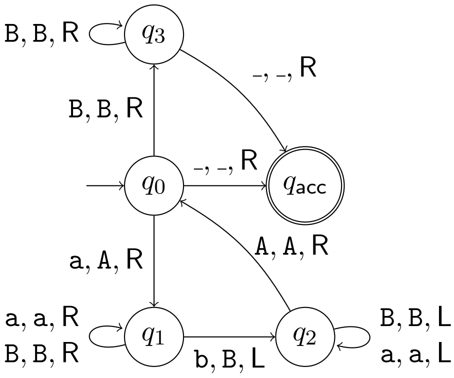
$$(q_0)\quad \texttt{a}\ \overset{\downarrow}{\texttt{a}}\ \texttt{b}\ \_\ \_\ .\ .\ .\ .$$

$$(q_0)\quad \texttt{a}\ \texttt{a}\ \overset{\downarrow}{\texttt{b}}\ \_\ \_\ .\ .\ .\ .$$

$$(q_0)\quad \texttt{a}\ \texttt{a}\ \texttt{a}\ \overset{\downarrow}{\_}\ \_\ .\ .\ .\ .$$

$$(q_{\textsf{acc}})\quad \texttt{a}\ \texttt{a}\ \texttt{a}\ \_\ \overset{\downarrow}{\_}\ .\ .\ .$$

(c) *The Turing machine:*



☛ This machine recognizes the language $L = \{\textsf{a}^n\textsf{b}^n \mid n \in \mathbb{N}\}$. As an example, let us study the (rejecting) computation of this machine on the word `aab`. First, the machine sees an `a`:

$$(q_0)\quad \overset{\downarrow}{\texttt{a}}\ \texttt{a}\ \texttt{b}\ \_\ \_\ .\ .\ .\ .$$

Then, it marks it (writing an `A` instead), and looks for a `b` to match it:

$$(q_1)\quad \texttt{A}\ \overset{\downarrow}{\texttt{a}}\ \texttt{b}\ \_\ \_\ .\ .\ .\ .$$

$$(q_1)\quad \texttt{A}\ \texttt{a}\ \overset{\downarrow}{\texttt{b}}\ \_\ \_\ .\ .\ .\ .$$

Once the `b` has been found, it marks it, and goes back to the last `A`:

$$(q_2)\quad \texttt{A}\ \overset{\downarrow}{\texttt{a}}\ \texttt{B}\ \_\ \_\ .\ .\ .\ .$$

$$(q_2)\quad \overset{\downarrow}{\texttt{A}}\ \texttt{a}\ \texttt{B}\ \_\ \_\ .\ .\ .\ .$$

That `A` is followed by a new `a`:

$$(q_0)\quad \texttt{A}\ \overset{\downarrow}{\texttt{a}}\ \texttt{B}\ \_\ \_\ .\ .\ .\ .$$

Then, the process starts again:

$$(q_1)\quad \texttt{A}\ \texttt{A}\ \overset{\downarrow}{\texttt{B}}\ \_\ \_\ .\ .\ .\ .$$

But now, it finds a blank symbol:

$$(q_1)\quad \texttt{A}\ \texttt{A}\ \texttt{B}\ \overset{\downarrow}{\_}\ \_\ .\ .\ .\ .$$

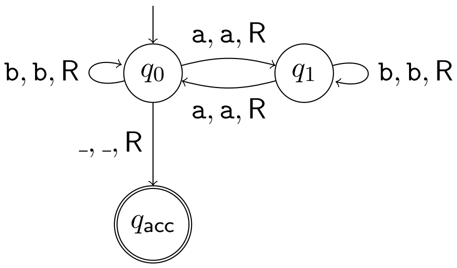
$$(q_{\textsf{rej}})$$

That Turing machine even decides that language. Indeed, an infinite run may:

- loop eternally on q_1 or on q_3: then, the reading head always goes to the right, without reading a blank symbol, which is impossible;
- loop eternally on q_2: then, the reading head always goes to the left, which is impossible;
- take eternally the cycle with q_0, q_1, and q_2: then, the run replaces infinitely often `a`'s by `A`'s, but never writes new `a`'s, which is impossible.

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.*

   ☛ The following machine decides that language:

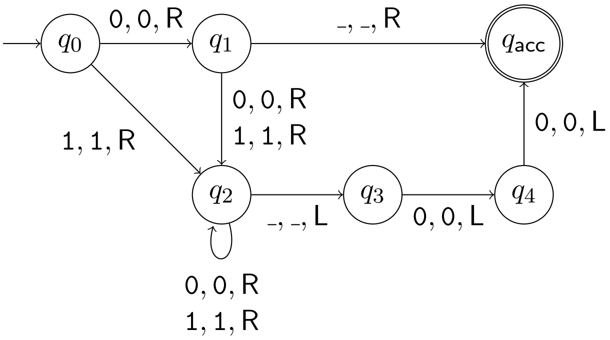


3. *Give the language associated to the following problem:*

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

   Give a Turing machine that decides that problem.

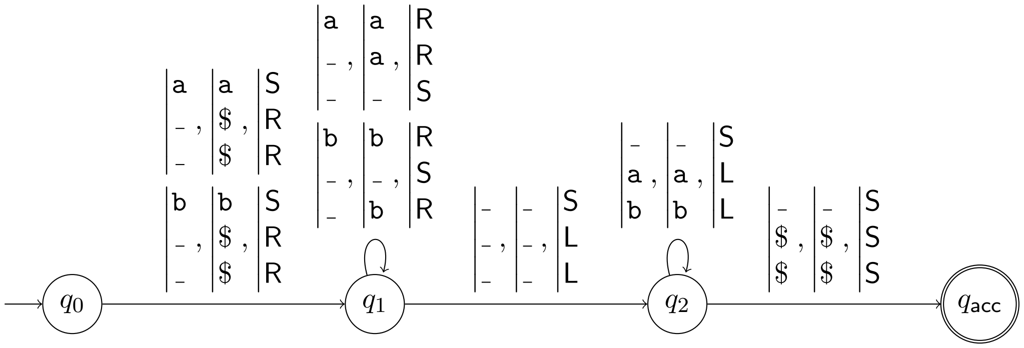
   ☛ The number n is a multiple of 4 either if it is 0, or if its two last digits are `0`s. The associated language is $\{\mathtt{0}\} \cup \{\mathtt{0}, \mathtt{1}\}^* \cdot \{\mathtt{00}\}$. A Turing machine that decides it is the following one:



   If the number is 0, written with one single digit `0`, then the machine goes to q_1 and then to q_{acc}. Otherwise, it goes to the state q_2, and the reading head moves to the end of the word by looping there, until it finds the blank symbol and goes to q_3. Then, it checks that the last digit was a `0` and goes to q_4, and that the previous one was a `0` and goes to q_{acc}.

3 Variants

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


q_0
q_1
q_2
q_{acc}
a, a, S / _, $, R / _, $, R
b, b, S / _, $, R / _, $, R
a, a, R / _, a, R / _, _, S
b, b, R / _, _, S / _, b, R
_, _, S / _, _, L / _, _, L
_, _, S / a, a, L / b, b, L
_, _, S / $, $, S / $, $, S

☛ This machine decides the language of the words $w \in \{\mathtt{a},\mathtt{b}\}^*$ that contain exactly as many a's as b's.

As an example, let us study the (accepting) computation of that machine on the word abab. At first, the second and the third tapes are empty; the machine marks their beginning with a special symbol \$:

$$(q_0)\quad\begin{cases}\overset{\downarrow}{\mathtt{a}}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\_\,\_\ldots\\ \overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\,\_\,\_\ldots\\ \overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\,\_\,\_\ldots\end{cases}$$

Then, it reads the word on the first tape from left to right, and copies all the a's on the second tape, and all the b's on the second tape:

$$(q_1)\quad\begin{cases}\overset{\downarrow}{\mathtt{a}}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\_\,\_\ldots\\ \$\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\,\_\ldots\\ \$\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\,\_\ldots\end{cases}$$

$$(q_1)\quad\begin{cases}\mathtt{a}\,\overset{\downarrow}{\mathtt{b}}\,\mathtt{a}\,\mathtt{b}\,\_\,\_\ldots\\ \$\,\mathtt{a}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\ldots\\ \$\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\,\_\ldots\end{cases}$$

$$(q_1)\quad\begin{cases}\mathtt{a}\,\mathtt{b}\,\overset{\downarrow}{\mathtt{a}}\,\mathtt{b}\,\_\,\_\ldots\\ \$\,\mathtt{a}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\ldots\\ \$\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\ldots\end{cases}$$

$$(q_1)\quad\begin{cases}\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\overset{\downarrow}{\mathtt{b}}\,\_\,\_\ldots\\ \$\,\mathtt{a}\,\mathtt{a}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\ldots\\ \$\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\,\_\ldots\end{cases}$$

$$(q_1)\quad\begin{cases}\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\ldots\\ \$\,\mathtt{a}\,\mathtt{a}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\ldots\\ \$\,\mathtt{b}\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\,\_\,\_\ldots\end{cases}$$

Now, the word has been completely read. The machine must therefore check that there are as many a's on the second tape than there are b's on the third tape. To do so, it reads what it has written on those tapes from right to left, until it finds the symbol \$:

$$(q_2)\quad\begin{cases}\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\ldots\\ \$\,\mathtt{a}\,\overset{\downarrow}{\mathtt{a}}\,\_\,\_\,\_\,\_\ldots\\ \$\,\mathtt{b}\,\overset{\downarrow}{\mathtt{b}}\,\_\,\_\,\_\,\_\ldots\end{cases}$$

$$(q_2)\quad\begin{cases}\mathtt{a}\,\mathtt{b}\,\mathtt{a}\,\mathtt{b}\,\overset{\downarrow}{\_}\,\_\ldots\\ \$\,\overset{\downarrow}{\mathtt{a}}\,\mathtt{a}\,\_\,\_\,\_\,\_\ldots\\ \$\,\overset{\downarrow}{\mathtt{b}}\,\mathtt{b}\,\_\,\_\,\_\,\_\ldots\end{cases}$$

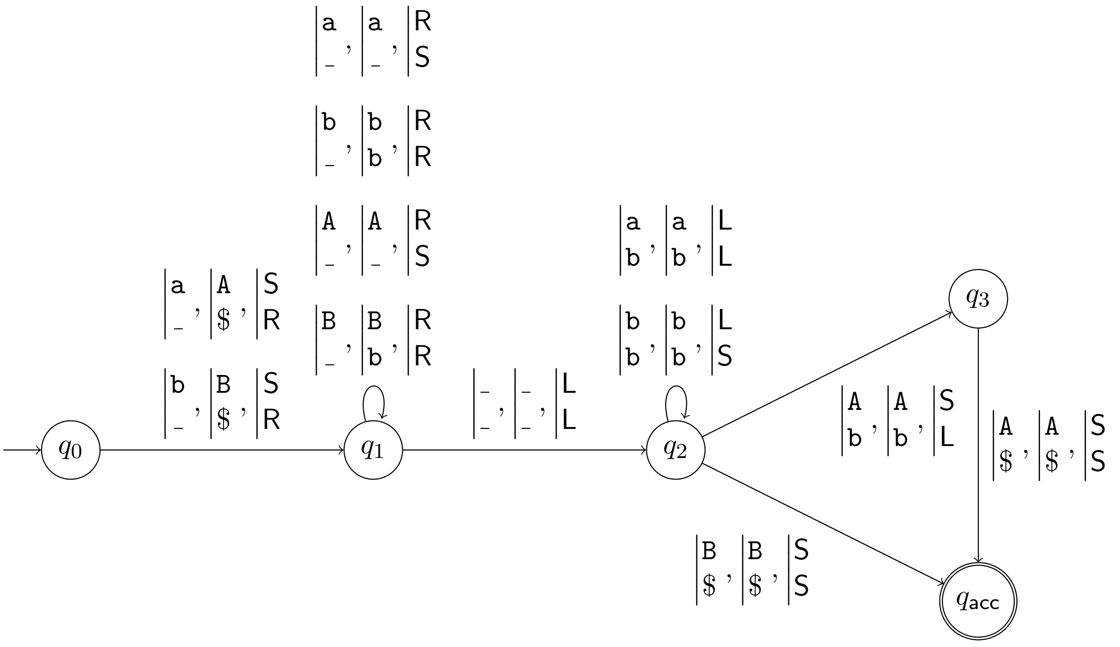
In the end, it should find the symbol $ simultaneously on the two tapes:

$$(q_2)\quad \begin{cases} \texttt{a b a b}\,\overset{\downarrow}{\_}\,\_\ldots \\ \overset{\downarrow}{\$}\ \texttt{a a}\ \_\ \_\ \_\ \_\ldots \\ \overset{\downarrow}{\$}\ \texttt{b b}\ \_\ \_\ \_\ \_\ldots \end{cases}$$

$$(q_{\mathsf{acc}})$$

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

   ☛ The following 2-tape machine copies only the `b`'s on the second tape, and then compares their number to the number of `a`'s on the first tape.



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

   ☛ Yes: every multi-tape Turing machine can be simulated by a single-tape Turing machine, in which the contents of the tapes are written one after the other.

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

   ☛ That machine remembers, first, what is the first letter $c \in \{\mathtt{a}, \mathtt{b}\}$. It marks it, replacing it by the symbol `1`, which will be used for the already read part of the first half of the word. Then, it starts reading the word from left to right, and non-deterministically chooses another c, hoping it is the beginning of the second half of the word. It marks it, replacing it by a `2`.

   The sequel is deterministic, and quite similar to the main idea of the Turing machine defined in Question 2.1c: the machine goes to the left until it finds a `1`, then moves to the right and remember the letter it reads. It replaces that letter by a `1`, moves to the right, finds the `2`'s, continues until it finds a letter, which should be the same, marks it with a `2`, and so on. Once there is no more letters in the first half, there should be no letters in the second half either.

   If the word read is of the form ww, then there is exactly one of the possible executions of the machine on it that is accepting: the one that guesses the right place to write the first `2`. But only one is enough.

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.*

☛ We have the following inclusions:

$$5l \subset 5m \subset 5h = 5d = 5e \subset 5i = 5g \subset 5a = 5c = 5k \subset 5b = 5f,$$

and $5a \subset 5j$.