Computability and Complexity

Instituto de Matemáticas

Exam

Turing machines and decidability

Guidelines

This is an oral exam. You have 2 hours to prepare exercises, and then 45 minutes to explain your answers. **You have to solve 1 exercise per part**, so in total you will have to prepare 6 exercises.

1 Turing machines

Exercise 1 : *Logic 101*.
Give an implementation-level description of a Turing machine that, from an input $\#w_1\#w_2\#$ (where w_1 and w_2 are strings over $\{0,1\}$, with $|w_1| = |w_2| > 0$), compute **one** of the following functions:

1. The logical AND (x AND $y = 1$ if and only if $x = y = 1$);
2. The logical OR (x OR $y = 0$ if and only if $x = y = 0$);
3. The logical XOR (x XOR $y = 1$ if and only if $x \neq y$).

Those functions are **bit-by-bit** operations. So, for instance, 111 AND 101 = 101; 000 OR 101 = 101; 111 XOR 101 = 010.
Answer:
In all cases, we write the output on a second tape.

1. *Here, the idea is that we write a 1 only if we have two 1's.*

M_{AND} := *On input $\#w_1\#w_2\#$:*

1. *Go to the first character after #. If it is a 0, go to step 2; if it is a 1, go to step 3 ; if it is a blank, accept.*
2. *Write 0 and move Right on the second tape. On the first tape, replace the character by a #. Move Right to the first character after #. Replace it by a #. Move the head back to the leftmost cell, and go to step 1.*
3. *Replace the character by a #. Move Right to the first character after #. If it is a 0, write 0 and move Right on the second tape; otherwise, write 1 and move Right on the second tape. On the first tape, replace the character by a #. Move the head back to the leftmost cell, and go to step 1.*

2. *Here, the idea is that we write a 0 only if we have two 0's.*

M_{OR} := *On input $\#w_1\#w_2\#$:*

1. *Go to the first character after #. If it is a 1, go to step 2; if it is a 0, go to step 3 ; if it is a blank, accept.*
2. *Write 1 and move Right on the second tape. On the first tape, replace the character by a #. Move Right to the first character after #. Replace it by a #. Move the head back to the leftmost cell, and go to step 1.*

3. *Replace the character by a #. Move Right to the first character after #. If it is a 0, write 0 and move Right on the second tape; otherwise, write 1 and move Right on the second tape. On the first tape, replace the character by a #. Move the head back to the leftmost cell, and go to step 1.*

3. *Here we write a 1 only if the two digits are different.*

$M_{XOR} :=$ *On input* $\#w_1\#w_2\#$*:*

1. *Go to the first character after #. If it is a 0, go to step 2; if it is a 1, go to step 3 ; if it is a blank, accept.*
2. *Replace the character by a #. Move Right to the first character after #. If it is a 0, write 0 and move Right on the second tape; otherwise, write 1 and move Right on the second tape. On the first tape, replace the character by a #. Move the head back to the leftmost cell, and go to step 1.*
3. *Replace the character by a #. Move Right to the first character after #. If it is a 1, write 0 and move Right on the second tape; otherwise, write 1 and move Right on the second tape. On the first tape, replace the character by a #. Move the head back to the leftmost cell, and go to step 1.*

□

Exercise 2 : *ABC is easy as 123.*
For a word w and a character ℓ, denote by $|w|_\ell$ the number of ℓ's in w.
Decide $L = \{w \in \{a, b, c\}^* \mid |w|_a > |w|_b > |w|_c\}$.
Answer:
We use a Turing machine with four tapes. The idea is to count the number of a's (resp. b's, c's) on the second (resp. third, fourth) tape. We then compare the tapes and accept if and only if the word verifies the condition.

□

2 Computation models

Exercise 3 : *Without Left.*
What class of languages is recognized by a Turing machine where the head can write and then move Right or stay Still, but not move Left? Explain your answer.
Answer:
It is easy to see that such Turing machines can only read their input (by moving Right) or follow ϵ-transitions (by staying Still). Thus, they are equivalent to nondeterministic finite automatas, and recognize the class of regular languages.

□

Exercise 4 : *Challenge: Resetting.*
What class of languages is recognized by a Turing machine where the head can write and then move Right or reset to the leftmost cell of the tape, but not move Left? Explain your answer.
Answer:
Those Turing machines are equivalent to classical Turing machines, so they recognize Turing-recognizable languages. Here is how we can simulate a Left-move:

1. *Mark the current character, then reset.*
2. *If the leftmost character is marked, then unmark it and Reset, the Left-move has been simulated.*
3. *Otherwise, mark it and move Right.*
4. *If the current character is unmarked, then Reset. Move Right until you reach a marked character. Unmark it, move Right. If the current character is marked, then go to step 5, otherwise go to step 4.*

5. *If the current character is marked, then the previous character was the one reached by the Left-move: unmark it, Reset and move Right until you reach the marked character, and we are done.*

The idea is to mark the character you started from, and then Reset and mark every character one by one until you have two marked characters side-by-side, in which case the leftmost of the two is the one reached by the Left-move.
Since we can simulate Left-moves and still have access to Right-moves, we can simulate classical Turing machines using this variant.

□

3 Closure under complement

Exercise 5 : *Complementation.*
Recall that the complement of a language L over Σ is $\overline{L} = \Sigma^* \setminus L$.

1. Prove that decidable languages are closed under complement.
2. Prove that Turing-recognizable languages are **not** closed under complement.

Answer:

1. *Assume that L is decided by a Turing machine M. We construct a Turing machine $\overline{M}$ that decide $\overline{L}$. Simply execute M on the input. If M accepts, then reject; and if M rejects, then accept.*
2. *We reason by contradiction. Let L be a Turing-recognizable language, then $\overline{L}$ is Turing-recognizable too. But then, we can use the Turing machines that recognize L and $\overline{L}$ to* decide *L: execute both Turing machines in parallel (basically, have each of them alternate doing one computation step), if the one that recognizes L accept then accept, and if the one that recognizes $\overline{L}$ accepts then reject. This Turing machine is correct since both languages are Turing-recognizable, and it is a decider for L. Thus, if the Turing-recognizable language class is closed under complement, then every Turing-recognizable language is decidable. This is a contradiction, since A_{TM} is Turing-recognizable and undecidable.*

□

4 Other closures

Exercise 6 : *Union.*
Prove that decidable languages **and** Turing-recognizable languages are closed under union.
Answer:

1. *Assume that L_1 and L_2 are decided by Turing machines M_1 and M_2. We construct a Turing machine M that decide $L_1 \cup L_2$. Have two tapes, copy the input on the second tape, execute M_1 on the first tape. If M_1 accepts the word, then accept. Otherwise, execute M_2 on the second tape. If M_2 accepts the word, then accept. Otherwise, reject. Note that this works since both languages are decidable, so the computation will end.*
2. *Assume that L_1 and L_2 are recognized by Turing machines M_1 and M_2. We construct a Turing machine M that recognizes $L_1 \cup L_2$. Have two tapes, copy the input on the second tape. Then execute M_1 and M_2 at the same time on their respective tapes (basically, have each of them alternate doing one computation step). Since both languages are Turing-recognizable, if the input is in one of the languages, then the corresponding Turing machine will halt and accept it, in which case we accept the input.*

□

Exercise 7 : *Intersection*.
Prove that decidable languages **and** Turing-recognizable languages are closed under intersection.
Answer:

1. *Assume that L_1 and L_2 are decided by Turing machines M_1 and M_2. We construct a Turing machine M that decide $L_1 \cap L_2$. Have two tapes, copy the input on the second tape, execute M_1 on the first tape. If M_1 rejects the word, then reject. Otherwise, execute M_2 on the second tape. If M_2 rejects the word, then reject. Otherwise, accept. Note that this works since both languages are decidable, so the computation will end.*

2. *Assume that L_1 and L_2 are recognized by Turing machines M_1 and M_2. Note that if a word is in the intersection of the two languages, then both machines will halt. So, execute M_1. If it halts and accepts, then execute M_2. If it also halts and accepts, then accept. If any of the two machines reject, then we reject; and if any of them loop then our machine will loop and thus not accept the input.*

□

Exercise 8 : *Concatenation*.
Prove that decidable languages **and** Turing-recognizable languages are closed under concatenation.
Answer:

1. *Assume that L_1 and L_2 are decided by Turing machines M_1 and M_2. We construct a Turing machine M that decide L_1L_2. The idea is that, for an input w, we use two other tapes, where we will try decompositions of w into x and y (the first tape will keep note of w). For each decomposition, we run M_1 on x and M_2 on y. If both machines accept, then we accept w. Otherwise, we go to the next possible decomposition. If all decompositions have been rejected by either of the two machines, then M rejects w.*

2. *Assume that L_1 and L_2 are recognized by Turing machines M_1 and M_2. Again, we use other tapes to compute all possible decompositions of an input w into x and y. However, this time, we cannot try decompositions one by one (since the machines may loop), but we can try all decompositions in parallel. So we do one step of M_1 on the first x, one step of M_2 on the first y, one step of M_1 on the second x, and so on until the last y; and we iterate. If, for a given decomposition, the two Turing machines halt and accept, then we accept. This recognizes L_1L_2.*

□

5 Reducibility

Exercise 9 : *United states*.
Prove that the following language is undecidable:

$L = \{< M, q, w > \mid M$ is a Turing machine, q is a state of M, and M reaches q in its computation on $w\}$.

.
Answer:
Assume by contradiction that L is decidable, and let D_L be a Turing machine that decides L. We construct the following Turing machine:

$S :=$ *On input $< M, w >$:*

1. *Let q_a be the accepting state of M. Run D_L on $< M, q_a, w >$.*

2. *If D_L accepts, then accept. Otherwise, reject.*

It is easy to see that S decides A_{TM}, a contradiction.

□

Exercise 10 : *Inclusivity*.
Prove that the following language is undecidable:

$$L = \{< M_1, M_2 > \mid M_1 \text{ and } M_2 \text{ are Turing machines, and } L(M_1) \subseteq L(M_2)\}.$$

Answer:
Assume by contradiction that L is decidable, and let D_L be a Turing machine that decides L. We construct the following Turing machine:

$S :=$ *On input* $< M >$*:*

1. *Construct a Turing machine R that rejects every input (thus, $L(R) = \emptyset$).*
2. *Run D_L on $< M, R >$. If D_L accepts, then accept.*

It is easy to see that S accepts $< M >$ if and only if $L(M) \subseteq \emptyset$, that is if $L(M) = \emptyset$. Hence, S decides the problem E_{TM}, a contradiction.

□

6 Rice's Theorem

Exercise 11 : *Context-freedom*.
Prove that the following language is undecidable:

$$L = \{< M > \mid M \text{ is a Turing machine, and } L(M) \text{ is context-free}\}.$$

Answer:
Being context-free is a nontrivial property (there are Turing machines that recognize context-free languages, and some that recognize non-context-free languages), and depends only on the language. By using Rice's Theorem, we obtain the result.

□

Exercise 12 : *Reverting*.
Prove that the following language is undecidable:

$$\{< M > \mid M \text{ is a Turing machine, and } L(M) = L(M)^R\}.$$

Answer:
Having its language being equal to its reverse is a nontrivial property, and depends only on the language. By using Rice's Theorem, we obtain the result.

□