Modelling Computations by Humans — Turing Machines I
Useful Turing Machine Variants

Introduction

It can be frustrating that so many descriptions of "Turing machines" describe slightly different models of computation: It can be challenge to find two or more references that describe precisely the same thing! This has happening almost since Turing machines were introduced [2]: By 1947, Emil Post had observed that Turing's original description was unclear and some points and proposed modifications of his own [1].

While may differences in definitions are quite minor, some "Turing machine variants" are extremely useful. The examination of proofs of their equivalence (they can all be used to define the same sets of "Turing-computable functions" for Turing machines that compute functions, or of "Turing-recognizable languages" and "Turing-decidable languages" for Turing machines that accept, reject or loop on strings) can help to see why the other, more minor, differences in definitions do not matter. These more significant variants are also useful in proofs of equivalence of other — very different looking — models of computation.

Ideally, all of this is ***review*** — that is, ideally, CPSC 513 students have already seen Turing machine variants, like these, before this. However, that is not guaranteed; quite a bit of material that follows might (initially) seem to be unfamiliar because time has passed since it was last presented, or because it was presented differently than it is here.

Turing Machines with One-Way Infinite Tapes

To begin, consider a variant of Turing machine with a ***one-way infinite*** tape instead of a two-way infinite tape:

- There is a "leftmost-cell". When applying a transition that would move the tape head left, and the tape head is already pointing to the leftmost cell, the tape head simply stays where it is. In all other cases, the tape head is moved as it would be with a two-way infinite tape.

- When a computation begins the symbols in the input strings are written on the leftmost cells of the tape and the tape head is also pointing to the leftmost cell. Thus a copy of "$\sqcup$" is the initially visible and, indeed, every cell of the tape stores a copy of "$\sqcup$" if the input string is the empty string. If the input is a non-empty string

  $$\omega = \alpha_1\alpha_2\ldots\alpha_n$$

  with positive length n, then the first n cells of the tape stores the symbols $\alpha_1, \alpha_2, \ldots, \alpha_n$ of the input string, all other cells of the tape store a copy of "$\sqcup$", and the leftmost symbol α_1 of the input is initially visible.

- Other definitions (including definitions of "accepting" a string, "rejecting" a string, "looping on" a string, "recognizing" a language and "deciding" a language) are, otherwise, unchanged.

One Simulation

Claim 1. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If L is Turing-recognizable then there exists a Turing machine, with a one-way infinite tape, whose language is L.

(b) If L is Turing-decidable then there exists a Turing machine, with a one-way infinite tape, that decides L.

Sketch of Proof. Let $L \subseteq \Sigma^\star$, for an alphabet $\Sigma^\star$. Suppose that L is Turing-recognizable; then there exists a (standard) Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

whose language is L. In order to prove part (a) of the claim it is necessary, and sufficient, to describe a Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

with a one-way infinite tape whose language is L, as well. A ***simulation*** will be given to show such a Turing machine, $\widehat{M}$, exists.

Representation of Information: The current state of M will be remembered using $\widehat{M}$'s finite control. In order to represent the content of M's tape, and the location of its tape head, $\widehat{M}$'s tape alphabet, $\widehat{\Gamma}$, will include each of the following symbols:

- All the symbols in $\Sigma \cup \{\sqcup\}$ (since all the symbols in this set might be visible on both M's and $\widehat{M}$'s tape, when executions on an input string in $\Sigma^\star$ begin).

- A "dotted" copy, $\dot{\sigma}$, of every symbol $\sigma \in \Gamma$. These will be used on the *leftmost* cell of $\widehat{M}$'s tape, immediately after $\widehat{M}$'s execution begins.
- The set $\Gamma \times \Gamma$ of *ordered pairs* of symbols in M's tape alphabet — so that a cell of $\widehat{M}$'s tape can represent information on a *pair* of cells on M's tape.

Suppose, in particular, that the non-blank portion of M's tape (extended, if necessary, to include the cell that the tape head initially pointed to) stores a string

$$\gamma_k \gamma_{k-1} \cdots \gamma_1 \alpha \beta_1 \beta_2 \cdots \beta_\ell \tag{1}$$

where

- α is stored in the cell that M's tape head initially pointed to,
- either $\gamma_k \neq \sqcup$ or the tape head points to this symbol (and $k \geq 1$) or $k = 0$,
- and either $\beta_\ell \neq \sqcup$ or the tape head points to this symbol (and $\ell \geq 1$) or $\ell = 0$.

Let $m = \max(k, \ell)$, set γ_i to be $\sqcup$ for $k + 1 \leq i \leq m$, and set β_j to be $\sqcup$ for $\ell + 1 \leq j \leq m$. Then the non-blank part of $\widehat{M}$'s tape should be as follows, then this is represented:

$\dot{\alpha}$	β_1	β_2	$\cdots$	β_m	$\sqcup$	$\cdots$
	γ_1	γ_2	$\cdots$	γ_m		

$\widehat{M}$'s finite control, $\widehat{Q}$, should include the set

$$(Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}) \times \{\text{top}, \text{bottom}\}$$

so that $\widehat{Q}$ include states (q, top) and (q, bottom) for every non-halting state in Q.: The following properties are satisfied, if the non-blank part of M's tape is as shown at line (1) and M is in state q.

- If M's tape head points to a symbol, γ_i, for some integer i such that $1 \leq i \leq m$ (so that the tape head is to the ***left*** of its original position) then $\widehat{M}$'s tape head points to the cell storing the ordered pair (β_i, γ_i) and $\widehat{M}$ is in state (q, bottom) when the simulation of the next move is about to begin.
- If M's tape head points to a symbol, β_i, for some integer i such that $1 \leq i \leq m$ (so that the tape head is to the ***right*** of its original position) then $\widehat{M}$'s tape head points to the cell storing the ordered pair (β_i, γ_i) and $\widehat{M}$ is in state (q, top) when the simulation of the next move is about to begin.
- If M's tape head points to the symbol α (so that the tape head is at is original position) then $\widehat{M}$'s tape points to the cell storing $\dot{\alpha}$ and $\widehat{M}$ is in one of states (q, top) or (q, bottom) when the simulation of the next move is about to begin.[1]

[1] The choice of these states will probably depend on where M's tape head was located, immediately before this.

Question #3, in the set of practice problems for this lecture, asks you to provide additional details of this simulation (including an initialization phase as well as the simulation of a move of M). This includes the statement of a claim, concerning how M and $\widehat{M}$ are related (and which is reasonably easy to prove, after additional details of the simulation have been given), and which implies this claim.

Once a proof of part (a) has been completed it suffices to note that $\widehat{M}$ decides L, if M decides L, in order to prove part (b) as well. □

Another Simulation

Claim 2. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If there exists a Turing machine, with a one-way infinite tape, whose language is L, then L is Turing-recognizable.

(b) If there exists a Turing machine, with a one-way infinite tape, that decides L, then L is Turing-decidable.

Sketch of Proof. Let $L \subseteq \Sigma^\star$, for an alphabet $\Sigma^\star$. Suppose that there is a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

with a one-way infinite tape whose language is L. In order to prove part (a) of the claim it is necessary, and sufficient, to describe a "standard" Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

— with a two-way infinite tape — whose language is L, as well. A ***simulation*** will be given to show such a Turing machine, $\widehat{M}$, exists.

Representation of Information: The current state of M will be remembered using $\widehat{M}$'s finite control. Indeed, it is sufficient for $\widehat{Q}$ to include the states in Q, along with two more states that are (only) used for a very simple initialization process. $\widehat{M}$'s tape alphabet, $\widehat{\Gamma}$, will include each of the following symbols:

- All the symbols in Γ.
- A "dotted" copy, $\dot{\sigma}$, for every symbol $\sigma \in \Gamma$. These are used to mark the cell on $\widehat{M}$'s tape that represents the *leftmost* cell on M's tape.

Suppose, in particular, that the non-blank part of M's tape (extended, if necessary, to include the location of M's tape head) stores a string

$$\alpha_0 \alpha_1 \alpha_2 \dots \alpha_k \tag{2}$$

where

- α_0 is stored at the leftmost cell of M's tape, α_1 is stored at the cell immediately to the right of that one, and so on,
- either $\alpha_k \neq \sqcup$ or M's tape head points to the cell storing α_k (or both), and
- all cells to the right of the cell storing α_k store copies of "$\sqcup$".

Then $\widehat{M}$'s tape stores a string

$$\dot{\alpha}_0 \alpha_1 \alpha_2 \dots \alpha_k$$

with copies of "$\sqcup$" on all cells to the left and to the right of this string. If M's tape head points to the cell storing α_0 then $\widehat{M}$'s tape head head points to the cell storing $\dot{\alpha}_0$ and, for $1 \leq i \leq k$, if M's tape head points to the cell storing α_i then $\widehat{M}$'s tape head points to the cell storing α_i as well.

Initialization: Consider the execution of these Turing machines on an input string

$$\omega = \beta_1 \beta_2 \dots \beta_n \in \Sigma^\star$$

with length n (so that $\beta_1, \beta_2, \dots, \beta_n \in \Sigma$). If $n \geq 1$ then M's tape head initially points to the rightmost copy of "$\sqcup$" to the *left* of the cell storing β_1 — while $\widehat{M}$'s tape head should initially point to the cell storing β_1, instead.

Under these circumstances, the "initialization" phase should consist of two moves (using two new states of $\widehat{Q}$ that are never entered after the initialization phase ends): In the first move (which begins in a new start state, $\widehat{q}_0$), $\widehat{M}$'s tape should move right, without changing the contents of the tape, and moving to another new state, $\widehat{q}_1$, in the process. In the second move, $\widehat{M}$ should replace the symbol $\sigma \in \Sigma \cup \{\sqcup\}$ that is now visible with the corresponding "dotted" symbol, $\dot{\sigma}$, without moving the position of the tape head, and entering M's start state, q_0. $\widehat{M}$ now represents M's initial configuration on the input string, as desired.

Simulation of a Step of M: Consider a move of M when this Turing machine is in a state $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and a symbol $\sigma \in \Gamma$ is visible on its tape. Then $\widehat{M}$ is also in state q at this point, and either σ or $\dot{\sigma}$ is visible on *this* machine's tape.

- If σ is visible on $\widehat{M}$'s tape then M's tape head is not presently located at this tape's leftmost cell, so that both Turing machines should change state, update tape contents, and move their tapes in exactly the same way. This can be accomplished by defining $\widehat{M}$'s transition function so that

  $$\widehat{\delta}(q, \sigma) = \delta(q, \sigma)$$

  for every state $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and for every symbol $\sigma \in \Gamma$.
- On the other hand, if $\dot{\sigma}$ is visible on $\widehat{M}$'s tape then M's tape head is located at its leftmost cell: A "dotted" symbol should be written instead of an "undotted" one, and attempts to move the tape head left should be replaced by transitions that leave the location of the

tape head unchanged. This can be accomplished by defining $\widehat{M}$'s transition function so that

$$\widehat{\delta}(q,\dot{\sigma}) = \begin{cases} (r,\dot{\tau},\mathtt{S}) & \text{if } \delta(q,\sigma) = (r,\tau,\mathtt{L}) \text{ for a symbol } \tau \in \Gamma \text{ and state } r \in Q, \\ (r,\dot{\tau},\mathtt{R}) & \text{if } \delta(q,\sigma) = (r,\tau,\mathtt{R}) \text{ for a symbol } \tau \in \Gamma \text{ and state } r \in Q, \\ (r,\dot{\tau},\mathtt{S}) & \text{if } \delta(q,\sigma) = (r,\tau,\mathtt{S}) \text{ for a symbol } \tau \in \Gamma \text{ and state } r \in Q. \end{cases}$$

The description of this simulation is almost complete. Question #4, in the set of practice problems, for this lecture, asks you to finish it (noting what else must be done).

Once a proof of part (a) has been completed, it suffices to note that if M decides L then $\widehat{M}$ decides L too, in order to prove part (b). □

Turing Machines That Compute Functions

Turing machines, with one-way infinite functions, that compute functions can also be described:

- The initial configuration for an input string $\omega \in \Sigma^\star$ is as described above.
- As described above, if the tape head is already pointing to the leftmost cell of the tape, and a transition moving left is applied, then the location of the tape is unchanged — so that the transition function is applied as described above, as well.

Now, consider a partial or total function $f : \Sigma_1^\star \to \Sigma_2^\star$, for alphabets Σ_1 and Σ_2. A Turing machine with a one-way infinite tape, that computes f, is a Turing machine

$$M = (Q, \Sigma_1, \Sigma_2, \Gamma, \delta, q_0, q_{\text{halt}})$$

that satisfies the following additional properties, concerning an execution of M on an input string $\omega \in \Sigma_1^\star$:

- If $f(\omega)$ is defined then this string should be written on the tape (with infinitely many copies of "␣" to the right) when the computation ends — with the tape head back at the leftmost cell of the tape, and
- if $f(\omega)$ is not defined then the computation should not end at all.

Claim 3. *Let $f : \Sigma_1^\star \to \Sigma_2^\star$ for alphabets Σ_1 and Σ_2.*

(a) If f is Turing-computable then there exists a Turing machine, with a one-way infinite tape, that computes f.

(b) If there exists a Turing machine, with a one-way infinite tape, that computes f then f is Turing-computable.

Questions #3 and #4, in the set of practice problems for this lecture, ask you to complete a proof of this claim.

Multi-Tape Turing Machines

For any (fixed) integer k such that $k \geq 1$, a ***k*-tape Turing machine** is a generalization of a Turing machine that has k tapes — all of whose tape heads can move independently.

- Since transitions must now depend on symbols visible on k tapes — but must also specify how k tapes should be updated, the ***transition function*** is now a partial function

$$\delta : Q \times \Gamma^k \to Q \times (\Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\})^k$$

  such that, for all symbols $\alpha_1, \alpha_2, \ldots, \alpha_k \in \Gamma$, $\delta(q, \alpha_1, \alpha_2, \ldots, \alpha_k)$ is defined for every state $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ (where q_{accept} and q_{reject} are the Turing machine's "accept" and "reject" state, respectively) and $\delta(q_{\text{accept}}, \alpha_1, \alpha_2, \ldots, \alpha_k)$ and $\delta(q_{\text{reject}}, \alpha_1, \alpha_2, \ldots, \alpha_k)$ are both undefined.

- The ***initial configuration*** for an input $\omega \in \Sigma^\star$ has the Turing machine in its initial state (often called q_0). The first tape is as it is for a standard Turing machine for input ω — so that the symbols in ω are written in contiguous cells, with these cells surrounded by cells storing "$\sqcup$" to the left and to the right, and with the tape head pointing to to the cell, storing "$\sqcup$", immediately to the left of the cell storing the first symbol in ω.[2] All cells on all other tapes initially store a copy of "$\sqcup$" with the tape heads for these other tapes pointing to one of these cells.

- Definitions (including definitions of "accepting" a string, "rejecting" a string, "looping on" a string, "recognizing" a language and "deciding" a language) are, otherwise, unchanged.

One Simulation

Claim 4. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If L is Turing-recognizable then there is a k-tape Turing machine, for some positive integer k, whose language is L.

(b) If L is Turing-decidable then there is a k-tape Turing machine, for some positive integer k, that decides L.

Sketch of Proof. Let $L \subseteq \Sigma^\star$, for an alphabet $\Sigma^\star$. Suppose that L is Turing-recognizable; then there exists a (standard) Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

[2] If ω is the empty string then every cell on the tape stores a copy of "$\sqcup$" and the tape head points to one of these cells.

whose language is L. In order to prove part (a) of the claim it is necessary, and sufficient, to describe a k-Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}}),$$

for some positive integer k, whose language is L, as well.

This is easy: All you need to do is set k to be 1 and set $\widehat{M}$ to be M. A comparison of definitions of acceptance of strings, rejection of strings, looping on strings, and recognizing and deciding languages, for "standard" Turing machines and k-tape Turing machines (when $k = 1$) is sufficient to confirm that $\widehat{M}$ has the same language as M,

Furthermore, $\widehat{M}$ decides this language if M does, as needed to establish this claim. □

Another Simulation

Claim 5. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If there exists a k-tapeTuring machine with language L, for some positive integer k, then L is Turing-recognizable.

(b) If there exists a k-tape Turing machine, for some positive integer k, that decides L, then L is Turing-decidable.

Sketch of Proof. Let $L \subseteq \Sigma^\star$, for an alphabet $\Sigma^\star$. Suppose that there exists a k-tape Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}),$$

for some positive integer k, whose language is L. In order to prove part (a) of the claim it is necessary, and sufficient, to describe a (standard, one-tape) Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

whose language is L. A ***simulation*** will be given to show such a Turing machine, $\widehat{M}$, exists.[3]

Representation of Information: Once again, the current state of M will be remembered using $\widehat{M}$'s finite control. $\widehat{M}$'s single tape will be used to store the contents of all of M's tapes, along with the locations of tape heads, by significantly expanding the tape alphabet, $\widehat{\Gamma}$:

$$\widehat{\Gamma} = \Sigma \cup \{\sqcup\} \cup (\{\sqcup, \downarrow\} \times \Gamma)^k.$$

This will allow us to view the nonblank part of $\widehat{M}$'s tape as having $2k$ ***tracks*** that can be used to store all the relevant information on M's multiple tapes (including locations of tape heads) at any times.

[3] Students who completed CPSC 313 or 351 might have seen a similar simulation that involves Turing machines with one-way infinite tapes. The simulation described here is modified so that it uses two-way infinite tapes instead.

In particular, after an "initialization" phase has been carried out, every cell in the non-blank part of $\widehat{M}$'s will a store a symbol in $(\{\sqcup, \downarrow\} \times \Gamma)^k$:

- For $1 \leq i \leq k$, the symbol in track $2i - 1$ is "$\sqcup$" if the tape head for the i^{th} tape *does not* point to the cell, in the i^{th} tape, that is being represented – and this symbol is "$\downarrow$" if the tape head for the i^{th} tape *does* point to this cell.
- For $1 \leq i \leq k$, the symbol in track $2i$ is the symbol, $\sigma \in \Gamma$, that is stored in the cell, in the i^{th} tape, that is being represented.

At the beginning of the simulation of each step $\widehat{M}$'s tape head should point to the rightmost copy of "$\sqcup$" to the *left* of the leftmost non-blank symbol on its tape.

Initialization: Suppose that M is being executed on an input string $\omega \in \Sigma^\star$.

- If $\omega = \sqcup$ then $\widehat{M}$'s tape should be initialized by writing a symbol

$$(\downarrow, \sqcup, \downarrow, \sqcup, \ldots, \downarrow, \sqcup) \tag{3}$$

  representing 2 tracks — with "$\downarrow$" in track $2i - 1$ and with "$\sqcup$" in track $2i$, for $1 \leq i \leq k$ — to represent the cells of the i tape heads that each tape head points to. $\widehat{M}$'s tape head should be moved to the left, so that it rests to the left of the non-blank part of the tape, as required.

- Suppose, instead, that

$$\omega = \alpha_1 \alpha_2 \ldots \alpha_n$$

  is a string in $\Sigma^\star$ with length $n \geq 1$ (so that $\sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma$). In this case $\widehat{M}$ should begin by writing the symbol shown at line (3) once again — but moving *right* instead of left. Then, for $1 \leq i \leq n$, $\widehat{M}$ should write the symbol

$$(\sqcup, \alpha_i, \sqcup, \sqcup, \ldots, \sqcup\sqcup)$$

  with α_i in track 2 and with "$\sqcup$" in every track — representing the contents of the tapes in the cells that are each i positions to the right of the tape head — moving *right* in the process.

  This ends when "$\sqcup$" is visible on the tape instead of a symbol in Σ. $\widehat{M}$ should then move its tape left, past non-blank symbols, until another "$\sqcup$" is visible. At this point $\widehat{M}$ should enter a state corresponding to M's start state q_0 (also corresponding to the beginning of the simulation of a step) so that it represents M's initial configuration.

Simulation of a Step of M: A simulation of a step of M includes the following phases.

- A ***read*** phase includes k sweeps right, and back. left, over part of the non-blank portion of $\widehat{M}$'s tape, in search of the symbol that is currently visible on M's i^{th} tape: The tape moves right until a symbol (representing $2k$ tracks) is found such that the symbol in track $2i-1$ is "$\downarrow$"' — so that the symbol in track $2i$ is the symbol, in Γ, that is currently visible on M's i^{th} tape.

  Since k (and $|\Gamma|$) is finite, $\widehat{M}$'s finite control can be used to remember all the symbols on tapes that have been discovered, along with M's current state. Thus, after the k^{th} sweep, $\widehat{M}$'s finite control stores all the information needed to determine the transition of M that should next be applied. Since M is a *fixed* Turing machine one can think of its transition function being included (or represented) in $\widehat{M}$'s transition function. Thus the new state that M should enter, the symbols that should be written onto M's tapes, and the directions in which M's tapes should move, are all available (that is, remembered using $\widehat{M}$'s finite control) at this point.

- An ***update*** phase also includes k series of moves. For $1 \leq i \leq k$ the i^{th} of these includes a sweep right to locate the cell corresponding to the location of M's i^{th} tape head (that is, a cell where "$\downarrow$" is found in track $2i-1$); a finite number of moves to write information in track $2i$ (updating M's i^{th} tape) and updating information in track $2i-1$, in this and adjoining cells, to reflect a movement of M's i^{th} tape head. This is followed by a sweep back to the left, so that $\widehat{M}$'s tape head is to the left of the non-blank part of its tape.

  Occasionally it will be necessary to extend the non-blank part of $\widehat{M}$'s tape, the cell that it should update stores "$\sqcup$" instead of a cell representing $2k$ tracks. Whenever this happens the same update should be made as for the case that the cell, now visited, stored a symbol

  $$(\sqcup, \sqcup, \sqcup, \sqcup, \ldots, \sqcup, \sqcup)$$

  representing $2k$ tracks, where each track stores "$\sqcup$".

- While it is not necessary, the simulation can also include a ***cleanup*** phase, where cells storing

  $$(\sqcup, \sqcup, \sqcup, \sqcup, \ldots, \sqcup, \sqcup)$$

  at the left or right end of the non-blank part of $\widehat{M}$'s tape are overwritten with "$\sqcup$" — so that the non-blank part of the tape is no larger than necessary.

- Finally, $\widehat{M}$ should change its state, so that is finite control represents the state that M should now move to.

Question #5, in the set of practice problems for this lecture, concerns the completion of a proof of part (a) of the claim. Once a proof of part (a) has been completed, it suffices to note that $\widehat{M}$ decides L if M does, in order to prove part (b). □

Turing Machines That Compute Functions

Once again let Σ_1 and Σ_2 be alphabets — finite and nonempty sets — and suppose that $\sqcup$ does not belong to either Σ_1 or Σ_2. Consider a (partial or total) function $f : \Sigma_1^\star \to \Sigma_2^\star$. If k is a positive integer then a ***k*-tape Turing machine M, that computes the function $f : \Sigma_1^\star \to \Sigma_2^\star$** can be defined. Formally,

$$M = (Q, \Sigma_1, \Sigma_2, \Gamma, \delta, q_0, q_{\text{halt}})$$

where Σ_1 and Σ_2 are as described above and where Q, Γ, δ, q_0 and q_{halt} satisfy the following conditions. These concern what happens if M is executed on input ω — so that ω is initially stored on the first tape (surrounded by copies of "$\sqcup$" to the left and to the right), with the tape head on the rightmost copy of "$\sqcup$" that is to the left of the input, and with all other tapes initially filled with copies of "$\sqcup$".

- If $f(\omega)$ is defined then this execution eventually ends. When it does, $f(\omega)$ is stored on the k^{th} tape (surrounded by copies of "$\sqcup$" to the left and to the right), with the tape head on the rightmost copy of "$\sqcup$" to the left of the output. All other tapes are filled with copies of "$\sqcup$".
- On the other hand, if $f(\omega)$ is undefined then this execution never ends at all.

Claim 6. *Let $f : \Sigma_1^\star \to \Sigma_2^\star$ for alphabets Σ_1 and Σ_2.*

(a) If f is Turing-computable then there exists a k-tape Turing machine, for some positive integer k, that computes f.

(b) If there exists a k-tape Turing machine that computes f, for some positive integer k, then f is Turing-computable.

Question #5, in the set of practice problems for this lecture, asks you to extend the simulation(s), given above, to prove this claim.

Nondeterministic Turing Machines

A ***nondeterministic*** Turing machine is the same as a regular Turing machine except that there can be **zero, one, or *many*** moves that might be possible at any time — so that the transition function is now a (partial) function

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}).$$

- If $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ then $\delta(q, \sigma)$ should be defined for every symbol $\sigma \in \Gamma$ — where (as usual) q_{accept} and q_{reject} are Turing machine's accept and reject states, respectively.

- If $q \in \{q_{\text{accept}}, q_{\text{reject}}\}$ then $\delta(q, \sigma)$ should *not* be defined for *any* symbol $\sigma \in \Gamma$.

The ***initial configuration*** of a Turing machine, for an input string $\omega \in \Sigma^\star$, is the same as for a standard (deterministic) Turing machine.

A computation of a nondeterministic Turing machine M on an input string $\omega \in \Sigma^\star$ can be represented as a ***tree***:

- The start configuration for ω is at the root.
- Each path down the tree shows one of the possible sequence of configurations that the machine can have when processing ω.

A nondeterministic Turing machine M ...

- ***accepts*** ω if ***at least one accepting configuration*** appears on the computation tree for ω.
- ***rejects*** ω if the computation tree for ω is ***finite*** and there is ***no accepting configuration*** on it.
- ***loops on*** ω otherwise — that is, when the computation tree is infinite and does not include any accepting configuration.

The definition of a Turing machine's ***recognizing*** a language (which depends on the definitions of "accepting", "rejecting" and "looping on" a string) is (otherwise) unchanged. However, the definition of a nondeterministic Turing machine's "deciding" a language is somewhat different: Let us say that a nondeterministic Turing machine M is a ***decider*** if ***all*** branches of the computation tree for M and an input $\omega \in \Sigma^\star$ are finite — so that the entire *computation tree* for ω is finite — for ***every*** input string $\omega \in \Sigma^\star$. We will say that a nondeterministic Turing machine M ***decides*** a language $L \subseteq \Sigma^\star$ if and only if

- M is a decider, and
- L is the language of M — that is, the set of strings in $\Sigma^\star$ that M accepts.

One Simulation

Claim 7. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If L is Turing-recognizable then there exists a nondeterministic Turing machine whose language is L.

(b) If L is Turing-decidable then there exists a nondeterministic Turing machine that decides L.

Proof. Let

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

be a deterministic Turing machine M that recognizes a language $L \subseteq \Sigma^\star$. It suffices to note that if we set

$$\widehat{M} = (Q, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

to be a *nondeterministic* Turing machine with the same set Q of states, tape alphabet Γ, and with the same start state, accept state and reject state, then it suffices to define $\widehat{\delta}$ by setting

$$\widehat{\delta}(q, \sigma) = \{\delta(q, \sigma)\}$$

for each state $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and every symbol $\sigma \in \Gamma$. The state diagrams for M and $\widehat{M}$ will look exactly the same. It also easy proved, then (by induction on $|\omega|$) that

$$\widehat{\delta}^\star(q_0, \omega) = \{\delta^\star(q_0, \omega)\}$$

for every string $\omega \in \Sigma^\star$. Since M and $\widehat{M}$ have the same start state, accept state, and reject state, it follows from this that $L(\widehat{M}) = L(M) = L$. Furthermore, $\widehat{M}$ ***decides*** L if M does — as needed to establish the claim. □

Another Simulation

Claim 8. *Let $L \subseteq \Sigma^\star$ (for an alphabet Σ).*

(a) If there exists a nondeterministic Turing machine whose language is L then L is Turing-recognizable.

(b) If there exists a nondeterministic Turing machine that decides L then L is Turing-decidable.

Sketch of Proof. Let $L \subseteq \Sigma^\star$, for an alphabet $\Sigma^\star$. Suppose that there exists a nondeterministic Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

whose language is L. In order to prove part (a) of the claim it is necessary, and sufficient, to describe a 2-tape (deterministic) Turing machine

$$\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}}),$$

whose language is L, as well — and such that if M decides L then so does $\widehat{M}$: The result will then follow by Claim 5, above.

A ***simulation*** will be given to show such a Turing machine, $\widehat{M}$, exists: $\widehat{M}$ will simulate *all* possible executions of M on an input string $\omega \in \Sigma^\star$ by performing a ***search*** over M's computation tree on this input string.

Representation of Information: In order to do this, $\widehat{M}$ will need to represent ***configurations*** of M.

$\widehat{M}$'s tape alphabet, $\widehat{\Gamma}$, will therefore include

- All the symbols in M's tape alphabet, Γ;
- all the symbols in M's set Q of states — and it will be assumed that $Q \cap \Gamma = \emptyset$;
- one more symbol, ♠, that is not in either Γ or Q.

$\widehat{M}$ will keep the following information on its tapes.

- *Tape #1:* This initially stores the input string $\omega \in \Sigma^\star$. During this simulation it will store a sequence of ***configurations*** of M, separated by (and ending with) ♠ — and with this tape starting with two ♠'s, so that the left end of the tape can be detected.
- *Tape #2:* This is a "work tape" that will be used to update tape #1.

How Not To Do This: One way to try to simulate M's behaviour is to perform a ***depth-first search*** — simulating computations down branches of the computation tree until each ends.

Unfortunately this will not always work if M is not a "decider" — because it is possible that an infinite branch might get searched — and $\widehat{M}$ will end up "looping on ω" — even though there is another *finite* branch in M's computation tree that ends with an accepting configuration — so that M *accepts* ω, instead.

How To Do This: Use ***breadth-first search*** instead: After t stages of this simulation all the configurations reachable from the start configuration by making t moves of M will be shown on Tape #1. Since there are only finitely many configurations that are reachable after t moves of M, for any non-negative integer t, this guarantees that an accepting configuration will be discovered, so that the input string will be accepted, whenever it belongs to the language of M.

Question #6, in the set of practice problems, for this lecture, asks you to describe this simulation and detail and use it to complete a proof of this claim. □

Turing Machines That Compute Functions

There is no straightforward — and useful — way to define "nondeterministic" computations of functions, so "nondeterministic Turing machines that compute functions" will not be defined or considered.

Combinations of the Above

Combinations of these extensions can also be considered. For example, one can also consider

- multi-tape Turing machines with one-way infinite tapes,
- nondeterministic Turing machines with one-way infinite tapes, or
- nondeterministic multi-tape Turing machines.

In each simulations described, in this document, can be combined and adopted to prove that the sets of Turing-recognizable languages and Turing-decidable languages — and, for deterministic computations — Turing-computable functions would be the same if they were defined using these variants of Turing machines instead of the standard model.

References

[1] Emil L. Post. Recursive unsolvability of a problem of Thue. *The Journal of Symbolic Logic*, 12:1–11, 1947.

[2] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42:230–365, 1937.