بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۲۱)

نظریهٔ زبان‌ها و ماشین‌ها

حسین فلسفین

Part 2: Computability Theory

Chapter 3: The Church-Turing Thesis

☞ *Section 3.1 Turing Machines*

مقدمه

So far in our development of the theory of computation, we have presented several models of computing devices. Finite automata are good models for devices that have a small amount of memory. Pushdown automata are good models for devices that have an unlimited memory that is usable only in the last in, first out manner of a stack. We have shown that some very simple tasks are beyond the capabilities of these models. Hence, they are too restricted to serve as models of general purpose computers.

مقدمه

Both of the simple devices we studied earlier, finite automata and pushdown automata, are models of computation. A machine of either type can receive an input string and execute an algorithm to obtain an answer, following a set of rules specific to the machine type. In both cases, the machine can execute only very specialized algorithms: in the case of an FA, algorithms in which there is an absolute limit on the amount of information that can be remembered, and in the case of a PDA, algorithms in which it is sufficient to access information following the rules imposed by a stack.

مقدمه

We turn now to a much more powerful abstract model, first proposed by Alan Turing in 1936, called the Turing machine. Similar to a finite automaton but with an unlimited and unrestricted memory, a Turing machine is a much more accurate model of a general purpose computer. A Turing machine can do everything that a real computer can do. Nonetheless, even a Turing machine cannot solve certain problems. In a very real sense, these problems are beyond the theoretical limits of computation. The Turing machine model uses an infinite tape as its unlimited memory. It has a tape head that can read and write symbols and move around on the tape.

مقدمه

A Turing machine is not just the next step beyond a pushdown automaton. It is, according to the Church-Turing thesis, a general model of computation, potentially able to execute any algorithm.

We introduce the important concept of a Turing machine, a finite-state control unit to which is attached a one-dimensional, unbounded tape. Even though a Turing machine is still a very simple structure, it turns out to be very powerful and lets us solve many problems that cannot be solved with a pushdown automaton. This leads to Turing's Thesis, which claims that Turing machines are the most general types of automata, in principle as powerful as any computer.

مقدمه

Turing's objective was to demonstrate the inherent limitations of algorithmic methods. He began with the idea of formulating a model capable in principle of carrying out any algorithm whatsoever; this allowed him to assert that any deficiencies of the model were in fact deficiencies of the algorithmic method. The Turing machine was intended as a theoretical model, not as a practical way to carry out a computation. However, it anticipated in principle many of the features of modern electronic computers.

مقدمه

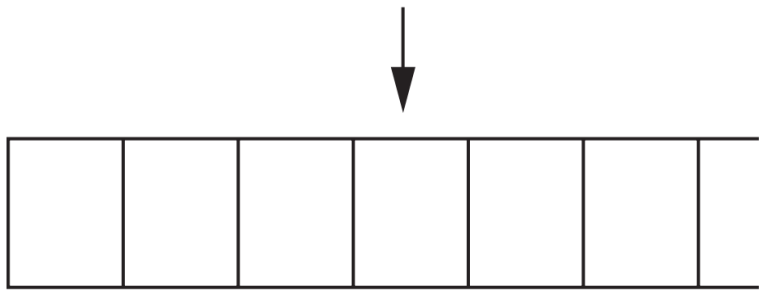
While the mechanism of a Turing machine is quite rudimentary, the concept is broad enough to cover very complex processes. The discussion culminates in the Turing thesis, which maintains that any computational process, such as those carried out by present-day computers, can be done on a Turing machine.

نوار (*tape*)

A Turing machine has a finite alphabet of symbols (actually two alphabets, an input alphabet and a larger one for use during the computation), and a finite set of states. Turing specified a linear tape, which has a left end and is potentially infinite to the right. The tape is marked off into squares, each of which can hold one symbol; if a square has no symbol on it, we say that it contains the blank symbol (␣).

The tape serves as the input device (the input is the finite string of nonblank symbols on the tape originally), the memory available for use during the computation, and the output device. The output, if it is relevant, is the string of symbols left on the tape at the end of the computation.

نوار (*tape*)



A Turing machine's storage is actually quite simple. It can be visualized as a single, one-dimensional array of cells, each of which can hold a single symbol. This array extends indefinitely in only one direction and the other end is fixed (semi-infinite or one-way infinite). The tape is therefore capable of holding an unlimited amount of information. The information can be read and changed in any order. We will call such a storage device a tape because it is analogous to the magnetic tapes used in older computers.

نوار (*tape*)

One crucial difference between a Turing machine and the two simpler machines we have studied is that a Turing machine is not restricted to one left-to-right pass through the input, performing its computation as it reads. The input string is present initially, before the computation starts. Because the tape head will normally start out at the beginning of the tape, we think of the machine as "reading" its input from left to right, but it might move its tape head over the squares containing the input symbols simply in order to go to another portion of the tape.

نوار (*tape*)

The computation of a Turing machine begins at the following situation: consecutive leftmost squares of the tape contain an input string followed by an infinite number of the blank symbols; the machine is in the start state; the head is over the leftmost square of the tape. Once started, the computation proceeds: the machine repeats changing the tape contents, the state, and the head position according to the rules specified by the transition function.

Initially the tape contains only the input string and is blank everywhere else. If the machine needs to store information, it may write this information on the tape. To read the information that it has written, the machine can move its head back over it.

وضعیت سوم: افتادن در دام حلقه!

☞ *A Turing machine has the entire string to start with, and once the machine decides to accept or reject the input, the computation stops. For this reason, a Turing machine never needs more than two halt states, one that denotes acceptance and one that denotes rejection (q_{accept} and q_{reject}).*

☞ *With Turing machines there is a new issue, which did not arise with finite automata and did not arise in any serious way with PDAs. If a Turing machine decides to accept or reject the input string, it stops. But it might not decide to do this, and so it might continue moving forever. This will turn out to be important!*

☞ *The machine continues computing until it decides to produce an output. The outputs accept and reject are obtained by entering designated accepting and rejecting states. If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting.*

وضعیت سوم: افتادن در دام حلقه!

For a finite automaton the computation halts when the automaton has read the entire input, whereas for a Turing machine the computation halts when the machine goes to an accept state or a reject state unless otherwise stated. As an outcome of computation, we have the three possibilities that the machine goes to an accept state, goes to a reject state, or never goes to an accept state or a reject state, moving on forever. Corresponding to these three possibilities, a Turing machine is said to accept, reject, or loop, respectively.

وضعیت سوم: افتادن در دام حلقه!

A string initially given on the tape is accepted if the machine eventually goes to the accept state and is rejected if it eventually goes to the reject state. A Turing machine fails to accept a string initially given on the tape if the machine eventually goes into the reject state or goes on forever, never going to the accept state or the reject state.

A Turing machine M, on input w, is not guaranteed to halt.

تفاوت‌های *TM*ها و *FA*ها

The following list summarizes the differences between finite automata and Turing machines.

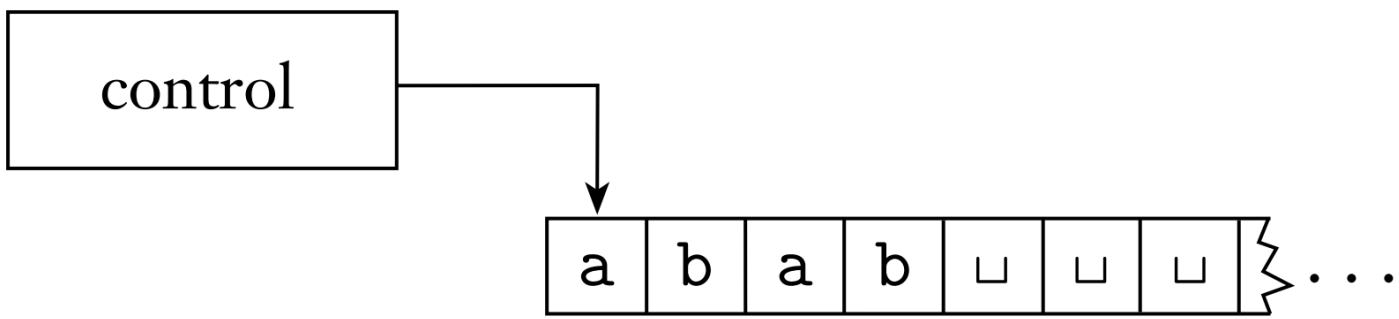
1. A Turing machine can both write on the tape and read from it.

2. The read-write head can move both to the left and to the right.

3. The tape is infinite.

4. The special states for rejecting and accepting take effect immediately.

Schematic of a Turing machine:



یک مثال

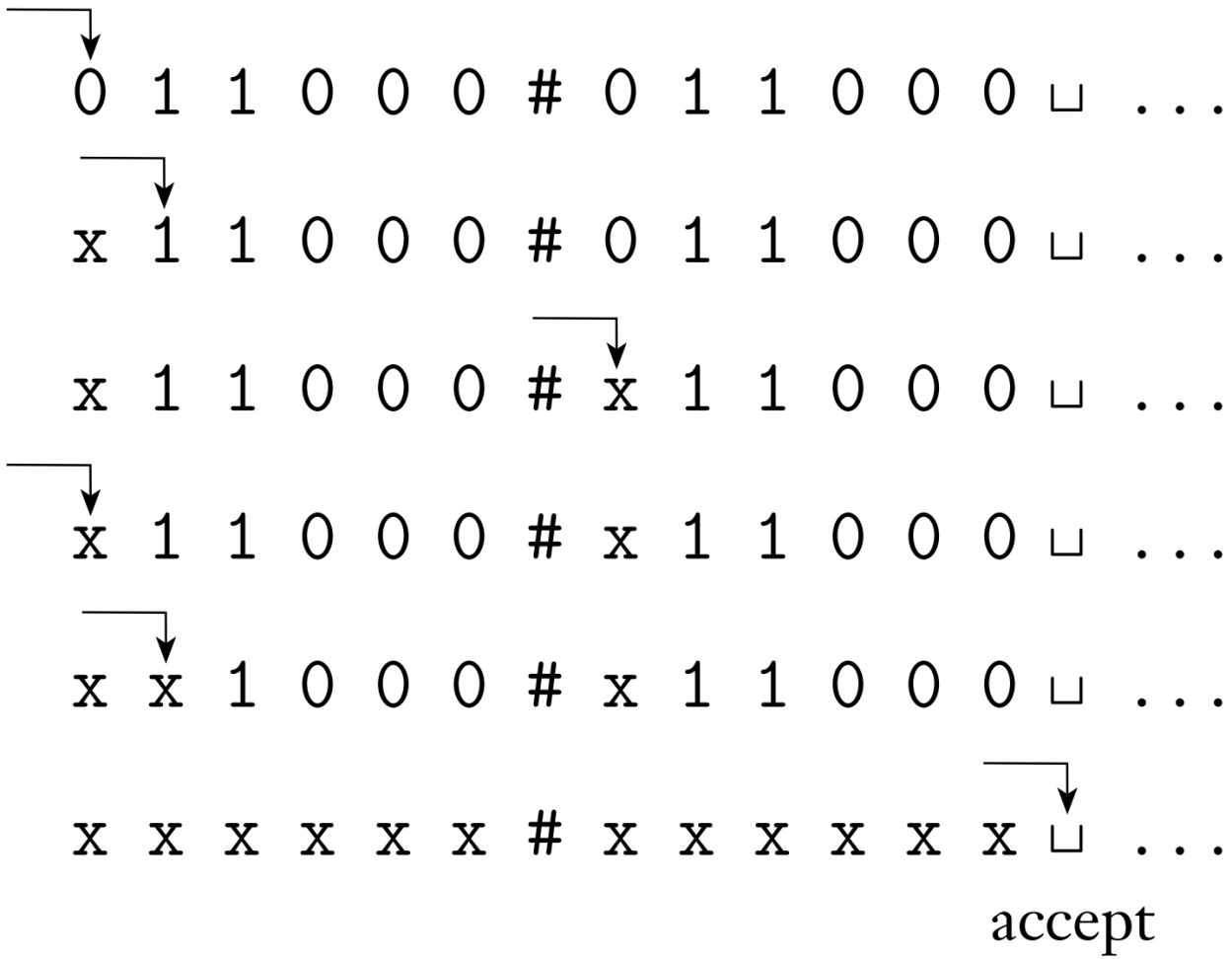
Let's introduce a Turing machine M_1 for testing membership in the language $B = \{w\#w | w \in \{0,1\}^*\}$:

We are allowed to move back and forth over the input and make marks on it. The obvious strategy is to zig-zag to the corresponding places on the two sides of the # and determine whether they match. Place marks on the tape to keep track of which places correspond. We design M_1 to work in that way. It makes multiple passes over the input string with the read-write head. On each pass it matches one of the characters on each side of the # symbol. To keep track of which symbols have been checked already, M_1 crosses off each symbol as it is examined. If it crosses off all the symbols, that means that everything matched successfully, and M_1 goes into an accept state. If it discovers a mismatch, it enters a reject state.

In summary, M_1's algorithm is as follows:

M_1 = "On input string w:
1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, reject. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, reject; otherwise, accept."

The following figure contains several nonconsecutive snapshots of M_1's tape after it is started on input 011000#011000:



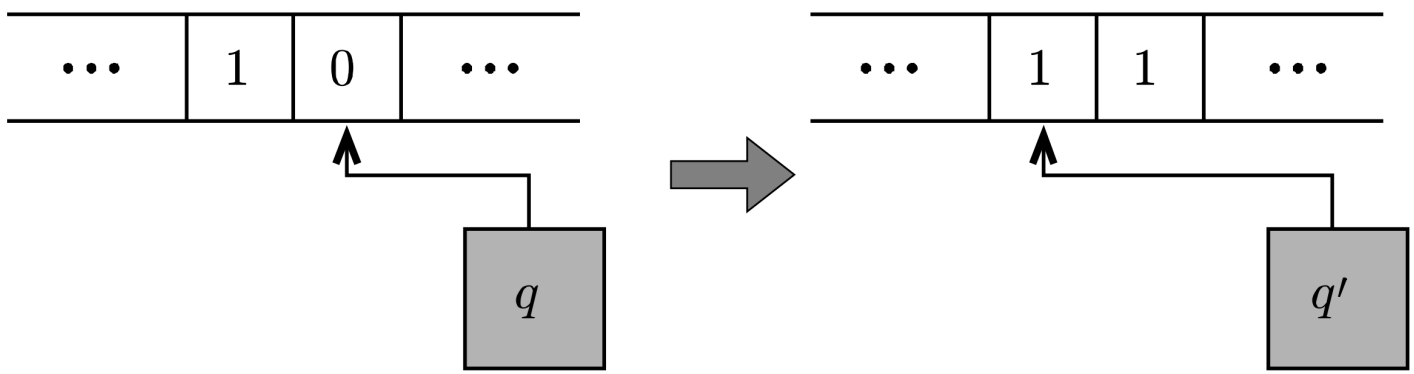
دربارهٔ تابع انتقال

In our version of a Turing machine, a single move is determined by the current state and the current tape symbol, and consists of three parts:

1. *Changing from the current state to another, possibly different state;*
2. *Replacing the symbol in the current square by another, possibly different symbol;*
3. *Moving the tape head one square to the right or moving it one square to the left if it is not already on the leftmost square.*

For a Turing machine, δ takes the form: $Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$. That is, when the machine is in a certain state q and the head is over a tape square containing a symbol a, and if $\delta(q, a) = (r, b, L)$, the machine writes the symbol b replacing the a, and goes to state r. The third component is either L or R and indicates whether the head moves to the left or right after writing. In this case, the L indicates a move to the left.

Example: *If δ is specified as $\delta(q, 0) = (q', 1, L)$, then the machine goes to state q', writes 1 replacing the 0, and moves the head to the left, which is shown in the following figure:*



تعریف رسمی یک ماشین تورینگ

Definition: ***A Turing machine is a 7-tuple,*** $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, ***where*** Q, Σ, Γ ***are all finite sets and***

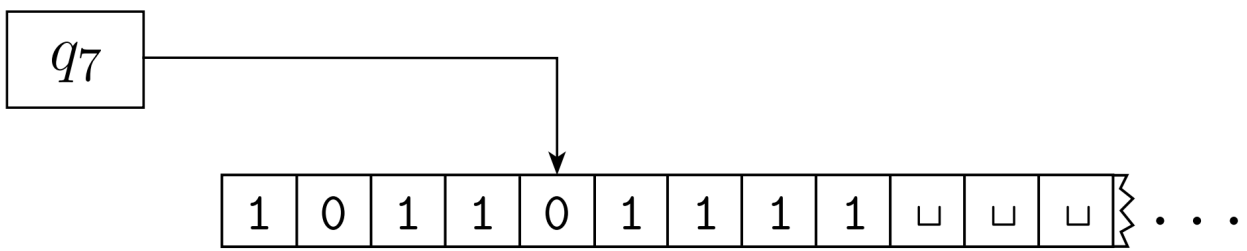
1. Q ***is the set of states,***
2. Σ ***is the input alphabet not containing the blank symbol*** $\sqcup$,
3. Γ ***is the tape alphabet, where*** $\sqcup \in \Gamma$ ***and*** $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ ***is the transition function,***
5. $q_0 \in Q$ ***is the start state,***
6. $q_{\text{accept}} \in Q$ ***is the accept state, and***
7. $q_{\text{reject}} \in Q$ ***is the reject state, where*** $q_{\text{reject}} \neq q_{\text{accept}}$.

دقت کنید که $\Sigma \subseteq \Gamma$، $\sqcup \notin \Sigma$، و $\sqcup \in \Gamma$.

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ computes as follows. Initially, M receives its input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled with blank symbols). The head starts on the leftmost square of the tape. Note that Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input. Once M has started, the computation proceeds according to the rules described by the transition function. If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L. The computation continues until it enters either the accept or reject states, at which point it halts. If neither occurs, M goes on forever.

A configuration of the Turing machine

As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a configuration of the Turing machine. Configurations often are represented in a special way. For a state q and two strings u and v over the tape alphabet Γ, we write uqv for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v. The tape contains only blanks following the last symbol of v. For example, $1011q_701111$ represents the configuration when the tape is 101101111, the current state is q_7, and the head is currently on the second 0.



☞ *Say that configuration C_1 yields configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step.*

☞ *Suppose that we have a, b, and c in Γ, as well as u and v in Γ^* and states q_i and q_j. In that case, uaq_ibv and uq_jacv are two configurations. Say that uaq_ibv yields uq_jacv if in the transition function $\delta(q_i, b) = (q_j, c, L)$. That handles the case where the Turing machine moves leftward. For a rightward move, say that uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$.*

☞ *A move from one configuration to another will be denoted by $\vdash$. The symbol $\vdash^*$ has the usual meaning of an arbitrary number of moves. Subscripts, such as $\vdash_M$, are used in arguments to distinguish between several machines.*

☞ *The collection of strings that M accepts is the language of M, or the language recognized by M, denoted $L(M)$.*

$$L(M) = \{x \in \Sigma^* : q_0x \vdash_M^* \alpha q_{\text{accept}}\beta \text{ for some } \alpha, \beta \in \Gamma^*\}.$$

Special cases occur when the head is at one of the ends of the configuration. For the left-hand end, the configuration $q_i bv$ yields $q_j cv$ if the transition is left-moving (because we prevent the machine from going off the left-hand end of the tape), and it yields $cq_j v$ for the right-moving transition. For the right-hand end, the configuration uaq_i is equivalent to $uaq_i\sqcup$ because we assume that blanks follow the part of the tape represented in the configuration. Thus we can handle this case as before, with the head no longer at the right-hand end.

☞ *The start configuration of M on input w is the configuration $q_0 w$, which indicates that the machine is in the start state q_0 with its head at the leftmost position on the tape.*

☞ *In an accepting configuration, the state of the configuration is q_{accept}. In a rejecting configuration, the state of the configuration is q_{reject}.*

☞ *Accepting and rejecting configurations are halting configurations and do not yield further configurations. Because the machine is defined to halt when in the states q_{accept} and q_{reject}, we equivalently could have defined the transition function to have the more complicated form $\delta : Q' \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$, where Q' is Q without q_{accept} and q_{reject}.*

تمرین:

Examine the formal definition of a Turing machine to answer the following questions, and explain your reasoning.

a. Can a Turing machine ever write the blank symbol $\sqcup$ on its tape?

b. Can the tape alphabet Γ be the same as the input alphabet Σ?

*c. Can a Turing machine's head **ever** be in the same location in two successive steps?*

d. Can a Turing machine contain just a single state?

بیان رسمی‌تر مفهوم محاسبه

A Turing machine M accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
1. C_1 is the start configuration of M on input w,
2. each C_i yields C_{i+1}, and
3. C_k is an accepting configuration.

A TM accepts string w if there exist configurations $C_1, \ldots, C_m$ such that $C_1 \vdash C_2 \vdash \cdots \vdash C_m$, and such that C_1 is expressed as $q_0 w$ and C_m is expressed as $u q_{\text{accept}} v$, where q_0 is the start state, q_{accept} is an accept state, and u and v are strings in Γ^*. We call such configurations C_1 and C_m a start configuration and an accepting configuration, respectively.

Terminology

Definition: *A Turing machine M accepts an input string $w \in \Sigma^*$ iff it enters q_{accept}.*

Definition: *A Turing machine M rejects an input string $w \in \Sigma^*$ iff it enters q_{reject}.*

Definition: *A Turing machine M halts an input string $w \in \Sigma^*$ iff it enters either q_{accept} or q_{reject}.*

Definition: *A Turing machine M loops an input string w iff it never enters either q_{accept} or q_{reject}.*

Definition: *Call a language Turing-recognizable if some Turing machine recognizes it. (If a Turing machine accepts a language, the language is said to be Turing-recognizable or simply recognizable.)*

*The definition of accepting does not require the Turing machine to enter q_{reject} for strings **not** in L. It only forbids that it enter q_{accept}. It can do this by entering q_{reject}, by moving right forever, by alternating between two adjacent tape squares, or by some other set of moves. Therefore, if $w \in L$, then M accepts w, and if $w \notin L$, then M does not accept w. Note that M may fail to accept either by rejecting or by failing to halt.*

عبارات معادل دیگری که به‌کار برده می‌شوند:

(Turing) acceptable or (Turing) recognizable or semidecidable or recursively enumerable (RE)

Deciding & Deciders

When we start a Turing machine on an input, three outcomes are possible. The machine may accept, reject, or loop. By loop we mean that the machine simply does not halt. Looping may entail any simple or complex behavior that never leads to a halting state. A Turing machine M can fail to accept an input by entering the q_{reject} state and rejecting, or by looping. Sometimes distinguishing a machine that is looping from one that is merely taking a long time is difficult. For this reason, we prefer Turing machines that halt on all inputs; such machines never loop. These machines are called deciders because they always make a decision to accept or reject. A decider that recognizes some language also is said to decide that language.

Definition: *Call a language Turing-decidable or simply decidable if some Turing machine decides it. (If a Turing machine that eventually halts for all inputs accepts a language, the language is said to be recursive or (Turing) decidable.)* ***In other words,*** *A Turing machine M decides language L over Σ iff for any string w in Σ^*, M enters q_{accept} if w is in L, and enters q_{reject} if w is not in L.*

If a Turing machine decides a language L, then it accepts L. If it accepts L, it might also decide L, but would not necessarily do so. "Deciding" is a stronger condition than "accepting." Every decidable language is Turing-recognizable. We will present examples of languages that are Turing-recognizable but not decidable.

What is the difference between accepting a language and deciding a language?

When a Turing machine M decides a language, the machine M, when a string is given as input, decides whether the string is accepted or rejected because M eventually halts, going to an accept state or a reject state. On the other hand, when a Turing machine M accepts a language, the machine M, when a string is given as input, eventually goes to an accept state if the string belongs to $L(M)$, but if the string does not belong to $L(M)$, there is a possibility that M never makes a decision, continuing transitions forever. In such a case, no matter how long we run the machine, it never tells whether the input is accepted or rejected.

Computing a function (Acting as a transducer)

In this course we mostly think of a Turing machine as an accepting machine which accepts a language that consists of strings accepted by the Turing machine. However, we sometimes think of a Turing machine as a transformer which, given a string as input, yields a string as output when the machine eventually halts. Such a Turing machine can be thought of as computing a function $f : \Sigma^* \mapsto \Sigma^*$ *when the machine starts its computation with string* w *on the tape, and eventually goes to a halting state with string* $f(w)$ *left on the tape, where the tape contains string* $f(w)$ *on its leftmost squares and blank symbols everywhere else.*

Definition: *A Turing machine* M *computes a function* $f : \Sigma^* \mapsto \Sigma^*$ *if, given a string* w *as input,* M *starts with its start configuration and eventually halts with just* $f(w)$ *on its tape.*