Turing Machines (cont'd)

- Last time we introduced a new computational model called Turing Machine (TM). Our goal for the next two lectures is to convince ourselves that TMs are a good model to study properties of intuitively *reasonable* computation. The word *reasonable*, among other things, implies the following:
  - The computation in the model is controlled by a finite set of instructions.
  - Each instruction can be carried out in a finite number of steps.
  - The computation in the model is deterministic (i.e. the effect of each instruction is predictable).
- We will proceed in stages:
  - Give a formal definition of TM and a computation of TM.
  - Show that Turing Machine can emulate a Turing Machine with multiple tapes.
  - Show that Turing Machine with multiple tapes can emulate a Random Access Machine (RAM).

  RAMs are appealing because the control logic of RAMs looks very much like computer programs written in assembly language, and so, hopefully, the equivalence between RAMs and TMs will make a strong point for the appropriateness of studying TMs.

Formal definition of Turing Machine.

- **Definition:** A *Turing Machine* is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where
  - Q is a finite set of *states*;
  - Σ is an *input alphabet*, $\sqcup \notin \Sigma$;
  - Γ is a *tape alphabet*, $\sqcup \in \Gamma$, and $\Sigma \subset \Gamma$;
  - $\delta : Q \times \Gamma \mapsto (Q \cup \{q_{accept}, q_{reject}\}) \times \Gamma \times \{L, R\}$ is the *transition function*;
  - $q_0 inQ$ is the *start state*;
  - q_{accept}, q_{reject} are the *accept* and *reject* states, respectively.
- A configuration of TM consists of the state, the content of the tape, and the current position of the head. Think of the configuration as a "snapshot" of the current state of the TM. Formally, we define a configuration as follows:

  Definition: A *configuration* of a Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, is a triple (u, q, v), written as $u\ q\ v$, where $u, v \in \Gamma^*$, and $q \in Q \cup \{q_{accept}, q_{reject}\}$.

The intended meaning is: M is in the state q, with the tape content uv, and the head pointing to the first symbol of v.

Definition: A configuration $u\ q_{accept}\ v$ is called an *accepting* configuration. A configuration $u\ q_{reject}\ v$ is called a *rejecting* configuration. A *halting* configuration is either an accepting or a rejecting configuration.

- Now we need to formalize a "legal" transition from one configuration to another.

  Definition: Let $x, y \in \Gamma^*$ (note: could be empty), and let $a, b \in \Gamma$. A configuration C_1 *yields* the configuration C_2, in symbols $C_1 \vdash C_2$, if one of the following 4 cases hold:

  – $C_1 = xq_iay$, $C_2 = xbq_jy$, and $\delta(q_i, a) = (q_j, b, R)$ – this is the right move when the head is not at right end of the string;

  – $C_1 = xq_ia$, $C_2 = xbq_j\sqcup$, and $\delta(q_i, a) = (q_j, b, R)$ – this is the right move when the head is at right end of the string;

  – $C_1 = xaq_iy$, $C_2 = xq_jby$, and $\delta(q_i, a) = (q_j, b, L)$ – this is the left move when the head is not at the left end of the tape;

  – $C_1 = q_iay$, $C_2 = q_jby$, and $\delta(q_i, a) = (q_j, b, L)$ – this is the left move when the head is at left end of the tape;

  Examples:

- And finally, we can talk about what does it mean for TM to compute:

  Definition: A *computation* of a TM $(Q, \Gamma, \Sigma, \delta, q_0, q_{accept}, q_{reject})$ on string $w \in \Sigma^*$, is a possibly infinite sequence of configurations $C_1, C_2, \ldots$ such that $C_1 = q_0w$, and for all i, $C_i \vdash C_{i+1}$.

  Definition: A *computation* of a TM $(Q, \Gamma, \Sigma, \delta, q_0, q_{accept}, q_{reject})$ on string $w \in \Sigma^*$, is a (possibly infinite) sequence of configurations $C_1, C_2, \ldots$ such that $C_1 = q_0w$, and for all i, $C_i \vdash C_{i+1}$.

  Definition: An *accepting* computation is a finite computation $C_1, \ldots, C_k$, where C_k is an accepting configuration. Similarly, a *rejecting* computation is a finite computation $C_1, \ldots, C_k$, where C_k is a rejecting computation. A computation which is either accepting or a rejecting is called a *halting* computation.

- Similarly to FAs and PDAs we define a language recognized by a TM:

  Definition: A language *recognized* by a TM M, in symbols $L(M)$, is the set of all strings accepted by M, i.e.

$$L(M) = \{x \mid M \text{ accepts } x\}.$$

Definition: A language L is called a *Turing-recognizable* (or, *recursively enumerable*) if there exists a TM M such that $L = L(M)$.

- *Important point:* as opposed to FAs and PDAs, Turing Machines may not halt on certain inputs. This means that some Turing-recognizable languages represent decision problems that, in a sense, are unsolvable by TMs – this is because given an input to a TM for Turing-recognizable language one can never know whether a TM sits in the infinite loop, or just computes for a very long time.

  On the other hand, we do feel that there are many languages for which it is possible to create a Turing machine that will always halt. Among those languages there are many that are not recognizable by FAs and PDAs (such as $L = \{0^n 1^n 2^n \mid n \geq 0\}$). Languages of this type are called *Turing-decidable*, or *recursive*.

  Definition A Turing Machine $M = (Q, \Gamma, \Sigma, \delta, q_0, q_{accept}, q_{reject})$ is said to *decide* a language L if for every string $w \in L$, M accepts w, *and* for every string $w \in \overline{L}$, M rejects w. In other words M halts on every input.

  Definition A language L is called a *Turing-decidable* (or, *recursive*) if there exists a Turing Machine that decides it.

  It should be clear that every Turing-decidable language is Turing-recognizable. However, as we shall see shortly, the converse is far from being true.

- To reiterate the previous point:

  - Turing-decidable languages correspond to problems "solvable" by Turing Machines.
  - Turing-recognizable languages that are not Turing-decidable, correspond to problems that are "unsolvable" by Turing Machines.

  Next time, we will see some evidence that "solvable" and "unsolvable" by Turing Machine means "solvable" and "unsolvable" by digital computers, and may also mean algorithmically "solvable" or "unsolvable".