Turing Machines

A PDA uses a stack, which is a restricted type of memory that can be written and read only at its end.

A Turing Machine (TM) is obtained by giving the automaton unrestricted access to memory. By doing this, we obtain unrestricted computational power.

Memory is organized as a tape over a finite alphabet Γ. The TM has a tape head that stands at a position on the tape. The TM can read the current position of the tape head, and it can overwrite the current position.

The TM can also move the tape head left and right on the tape without restriction.

On the next slides, we extend the standard definition, to make the TM more usable in examples.

Requirements on the Specification

Our specification of Turing Machines must meet the following requirements:

1. Easy to use in example constructions.
2. Not too far removed from the definition in Michael Sipser Introduction to the Theory of Computation, and the definition of the online TM simulator `https://turingmachinesimulator.com/`
3. It must subsume Sipser's definition, which means that every TM that is valid in Sipser's definition, must be valid in our definition.
4. It must support non-deterministic TM's.

Turing Machines

A Turing Machine M has form $(Q, \Sigma, \Gamma, \sqcup, \delta, q_s, F)$, where

- Q is a finite set of states.
- Σ is the input alphabet.
- Γ is the tape alphabet. We require that $\Sigma \subset \Gamma$.
- $\sqcup \in \Gamma \backslash \Sigma$ is a special space (or blank) symbol.
- δ is the transition relation, which must be a subset of

$$(Q \times \Gamma) \times (Q \times \Gamma^* \times \mathcal{Z}).$$

- $q_s \in Q$ is the starting state.
- $F \subseteq Q$ are the accepting states.

Explanation

In the beginning, the input word is on the tape. We assume that the tape is infinite. To the left and to the right of the input word are infinitely many spaces ($\sqcup$).

The tape head starts at the position of the first letter of the input word, or a $\sqcup$ when the input word is empty.

The TM starts in state q_s. At each moment in time, the TM looks at its state, and the symbol at the head of the tape to decide what to do next.

The TM accepts the input, if it reaches a state in F. There are no dedicated rejecting states. The TM can explicitly reject by getting stuck, or implicitly by looping forever.

Transition Relation

The transition relation δ consists of 5-tuples of form $(q_1, \sigma_1, q_2, \sigma_2, z)$.

1. If the current state is q_1, and the tape starting from the current position of the tape head contains the word σ_1, then the transition is possible.
2. In that case, σ_1 is deleted from the tape and replaced by σ_2. If $\|\sigma_1\| < \|\sigma_2\|$, then word on the tape becomes longer. If $\|\sigma_1\| > \|\sigma_2\|$, it becomes shorter. Otherwise, the length stays the same.
3. After that, the tape head is moved over z positions. If $z > 0$, the head is moved to the right. If $z < 0$, it is moved to the left. If $z = 0$, it is not moved.
4. Finally, the new state becomes q_2.

Where are the Generalizations?

The generalizations are as follows:

In order to decide about making a transition, the TM can look at more than one tape cell.

The TM can insert or remove tape cells at the position of the tape head.

The TM can move the tape head over more than one position in a single transition.

Recognizing non-context free languages

Consider the language $\mathcal{L} = \{\ w^i \# w^i \mid i \geq 0, w \in \{a, b\}^*\ \}$. Is it context-free?

It can be recognized by a Turing Machine.

What about $\mathcal{L} = \{a^i b^i c^i \mid i \geq 0\}$?

Is it context free?

Can you define a Turing Machine for it?

Can you modify the Turing Machine in such a way that it deletes one a, deletes one b, deletes one c as long as possible, until the word is empty.