Turing machines and languages (Configurations)

Representing configurations

We represent configurations as $usiv$ where u and v are strings in the tape alphabet $\tilde{A}$ and s_i is the current state of the machine. The tape contents are then uv and the current location of the tape head contents one then uv and the current location of the tape head is the first Symbol of v. The assumption here is that the tape contains only blanks after the last symbol in v.

EXAMPLE

$\epsilon i001$ is the configuration
[0][0][1][_]
-^
as we start examining the string 001 in our previous example of a turing machine

DEF: let C_1, C_2 be two configurations of a given Turing machine. We say that the configuration C_1 yields the configuration C_2 if the turing machine can go from C_1 to C_2 in one step.

EXAMPLE: If S_i, S_j are states, u and v are steps in the tape alphabet $\tilde{A}$ and a, b, c $\in \tilde{A}$.
A configuration $C_1 = uas_i$ bv yields a configuration $C_2 = us_jacv$ if the transition mapping specifies a transition $t(s_i, b) = (s_j, c, L)$. In other words, the Turing machine is in state s_i, it needs __ b, writes character c in its place, enters state s_j ands its head moves left.

Types of Configurations

Initial configuration:

Input u, is iu or (ϵiu), which indicates that the machine is in its initial state i, with its head at the leftmost position on the tape (which is the reason why this configuration has no string left of the state)

Accepting configurations:

$us_{accept}v$ for $u, v \in \tilde{A}^*$, (u,v strings in $\tilde{A}$), namely the machine is in its accept state.

Rejecting configuration:

$us_{reject}v$ for $u, v \in \tilde{A}^* \Leftarrow$ the machine is in its reject state.

Halting configurations:

Yield no further configurations, no transitions are defined out of their states. Accepting and Rejecting configurations are examples of Halting configurations.

DEF: A Turing machine M accepts input $w \in A\star$ (Strings in the input alphabet) if $\exists$ a sequence of configurations such that:

1. C_1 is the start configuration with input w $(c_1 = \epsilon iw)$
2. Each C_i yields C_{i+1} for i = 1, ... k-1.
3. C_k is an accepting configuration

DEF: Let M be a turing machine $L(M) = \{w \in A^* | M \text{ accepts } w\}$ is the language recognized by M.
DEF: A Language $L \subset A^*$ is **Turing-recognizable** if $\exists M$ a turing machine that recognizes L.

> NB Some textbooks use the terminology **Recursively enumerable language**

Turning Recognizable

Turing recognizable is not as strong a notion as we might perceive as a turing machine can:

- Accept
- Reject
- ***Loop***

Looping

Looping is very simple or complex behavior that does not lead to a halting state (Accept OR Reject). The problem with looping is that the user does not have infinite time. It can be difficult to distinguish between looping or taking a very long time to compute. We thus prefer deciders.

Deciders

DEF: A decider is a Turing machine that always enters either an accept state or a reject state for every input in A^*
DEF: A decider that recognizes some language $L \subset A^*$ is said to decide that language.
DEF: A language $L \subset A^*$ is called Turing decidable if $\exists$ a Turing machine M that decides L.

> NB Some textbooks use the terminology **Recursive language** instead of Turing decidable

EXAMPLE: $L = $\{0^m 1^m | m \in N, m \geq 1\}$ is turing decidable because the Turing machine we built that recognized it was in fact a decider.
Turing Decidable $\to$ Turing recognizable, but the converse is not true.
We'll construct an example of a language that is Turing-recognizable but NOT turing decidable in the last lecture before the final review