Multi-Tape Turing Machines

The transition function for a k-tape TM has the form

$$\delta : Q' \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k.$$

Here is an informal description (partly high-level and partly implementation-level) of a 3-tape Turing machine (transducer) A for doing binary addition. It assumes that the input (which appears initially on tape 1) has the form $w_1 \# w_2$, where w_1 and w_2 are the binary numbers to be added, with each appearing in reverse order (least significant bit first). The result appears on tape 3, also in reverse order.

$A =$ "On input string $w_1 \# w_2$:

1. Copy w_2 to tape 2.
2. Erase all but w_1 from tape 1.
3. Initialize the carry bit c (recorded in the finite control) to 0.
4. Position all three tape heads at the beginning of each tape.
5. Let b_1 = contents of current cell of tape 1 and b_2 = contents of current cell of tape 2 (recorded in the finite control).
6. If b_1 and b_2 are not both blank:
7. Perform a full-adder operation with b_1, b_2, and c, writing the sum bit to tape 3 recording the updated carry bit c in the finite control, and move all three tape heads one cell to the right.
   (If either b_1 or b_2 is blank, treat it like 0 when performing the addition.)
8. Go to step 5.
9. If b_1 and b_2 are both blank:
10. If the carry bit $c = 1$, write 1 on tape 3.
11. *Halt.*"

Multi-Tape TM vs. (Single-Tape) TM

The transition function for a k-tape TM has the form

$$\delta : Q' \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k.$$

Theorem. For any k-tape TM there is an equivalent (single-tape) TM.

Proof. Let M be a multi-tape TM with k tapes. Here is a high-level description of a (single-tape) TM S that simulates the behavior of M.

$S =$ "On input string $w = a_1 a_2 \ldots a_n$:

1. Format the tape as follows:
$$\#\dot{a}_1 a_2 \ldots a_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$
Here # is used as a delimiter between the parts that represent each tape and there is also one at the beginning and the end of the entire non-blank portion of the tape, making a total of $k + 1$ of them. A "marked" version of each tape symbol (including the blank) is used to indicate the current position of the "virtual head" for the corresponding tape.

2. Starting at the left end, scan to the right to determine all the symbols under the simulated tape heads and record this combination in the finite control. (There may be many possible combinations, implying a large number of states, but all that matters is that this is a finite number.)

3. Go back to the left end again and sweep to the right to update the simulated tapes according to the transition function δ. This includes moving the marks to indicate movement of the virtual tape heads.

4. If one of the virtual heads gets moved right onto a #, write a blank into this cell and shift everything from here to the end of the tape one cell to the right.

5. If M would enter its accept state, *accept*; if M would enter its reject state, *reject*. (Or when M is a transducer, if it would enter its halt state, *halt*.)

6. Go to stage 2."

Corollary.

1. A language is Turing-recognizable iff some multi-tape TM recognizes it.

2. A language is decidable iff some multi-tape TM decides it.

Nondeterministic TM vs. (Deterministic) TM

The transition function for a nondeterministic TM (NTM) has the form

$$\delta : Q' \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L,R\}}.$$

An NTM accepts a string iff *some* branch of its computation tree leads to the accept state. (Other branches might lead to the reject state and still others might never terminate).

Theorem. For any NTM there is an equivalent TM.

Proof. Let N be a nondeterministic TM. Here is a high-level description of a (deterministic) 3-tape TM D that explores the computation tree of N in a breadth-first manner:

First, its use of the 3 tapes is as follows:

- Tape 1 holds the input string. Its contents are never changed.
- Tape 2 is the *simulation tape*, used to represent the tape contents for the particular computation branch of N currently being simulated.
- Tape 3 is the *address tape*, used to keep track of which branch of the N's computation tree is currently being simulated.

$D =$ "On input w:

1. Copy tape 1 to tape 2.[1]
2. Use tape 2 to simulate N on the computation path from the root to the node represented by the address on tape 3. This is done one step at a time by consulting the next symbol to the right on tape 3 after each simulated transition. If, at some point during the simulation, the address is found to be invalid or a rejecting configuration is encountered, go to stage 4.
3. If an accepting configuration is encountered, *accept*.
4. If all branches of the tree are exhausted, *reject*.
5. Replace the string on tape 3 by the lexicographically next string.
6. Go to stage 1."

Since any 3-tape TM has an equivalent TM, the theorem follows.

Definition. An NTM is a *decider* if all branches halt on all inputs.

Corollary.

1. A language is Turing-recognizable iff some nondeterministic TM recognizes it.
2. A language is decidable iff some nondeterministic TM decides it.

[1] This should also involve copying (at least) one terminating blank since an earlier simulated branch of the computation may have written to parts of this tape beyond the space originally occupied by the input string.

Nondeterministic Multi-Tape Turing Machines

We could also consider combining the use of multiple tapes with nondeterminism. The transition function for a k-tape NTM has the form

$$\delta : Q' \times \Gamma^k \longrightarrow 2^{Q \times \Gamma^k \times \{L,R,S\}}.$$

Theorem. For any multi-tape NTM there is an equivalent TM.

Proof. If the multitape NTM has k tapes, we use the construction of the previous proof, this time with k simulation tapes instead of just one. Thus the overall deterministic machine should have $k+2$ tapes, the k simulation tapes along with a tape that holds the input and is never changed plus an address tape that keeps track of the part of the computation tree currently being simulated. Since this $(k+2)$-tape machine can itself be simulated by a single-tape TM, the result follows.

Corollary.

1. A language is Turing-recognizable iff some multi-tape nondeterministic TM recognizes it.
2. A language is decidable iff some multi-tape nondeterministic TM decides it.

This means that whenever we want to prove a language is decidable or Turing-recognizable, it is sufficient to show how to construct a TM that uses any combination of multiple tapes or nondeterminism to perform the desired computation. This is useful because it may be easier to describe the design of such a machine than it is to describe the design of an ordinary (single-tape, deterministic) TM to do the job.