Handout 10: More About Turing Machines

© Laura Kovács

10.1 Robustness. Many variations of the Turing machine model are equally powerful. This can be shown by TM simulations. For example, a DTM M with two tapes can be simulated by a standard DTM M' (with one tape):

> A configuration $(q, u_1, a_1, v_1, u_2, a_2, v_2)$ of the two-tape machine *corresponds* to the configuration $(q, \varepsilon, \#, u_1\$a_1v_1\#u_2\$a_2v_2)$ of the one-tape machine (the \$ symbols mark the positions of the two r/w heads). The two machines start in corresponding configurations. For every transition t of the two-tape machine M, the one-tape machine M' performs a finite sequence $t_1, \dots, t_n$ of transitions such that the two machines are again in corresponding configurations.

Note that this simulation preserves acceptance, rejection, and looping. Since the simulation preserves acceptance, it follows that the two-tape TMs accept no more than the r.e. languages. Since the simulation preserves also looping, it follows that the two-tape Turing deciders accept no more than the recursive languages.

10.2 Nondeterminism. A nondeterministic TM M can be simulated by a three-tape DTM D, as follows[1]. We view M's computation on an input w as a tree. Each branch of the tree represents one branch of nondeterminism (i.e. one possible run of M on w). Each node of the tree is a configuration of M; the root of the tree is the starting configuration of M. We then construct D to explore in a breadth first order all possible branches of M's nondeterministic computation tree. D uses its three tapes in a particular way. Tape 1 of D always contains the input word and is never altered. Tape 2 keeps track of D's location in M's nondeterministic computation tree (i.e. tape 2 remembers the tree node that we currently explore). Tape 3 maintains a copy of M's tape on some branch of its nondeterministic computation (i.e. tape 3 simulates M upto the tree node stored in tape 2). If D encounters an accepting configuration using tapes 2 and 3, then D accepts the input w stored on tape 1. The construction of D is such that for every input word w,

1. if some run of M over w accepts, then the run of D over w accepts;
2. if all runs of M over w reject, then the run of D over w rejects;
3. if no run of M over w accepts, and some run of M over w loops, then the run of D over w loops.

Further, using a simulation like the one of Section 10.1, the DTM D with three tapes (and hence M) can be simulated by a standard DTM M' with only one tape.

Therefore, the nondeterministic TMs accept the r.e. languages, and the nondeterministic Turing deciders accept the recursive languages.

10.3 Positive boolean operations. Given two DTMs M_1 and M_2, we can construct a single DTM M that encodes the two tapes of M_1 and M_2 similar to the construction of Section 10.1, and alternates between simulating a transition of M_1 and simulating a transition of M_2.

Union As soon as one of M_1 or M_2 accepts, let M accept. Once both M_1 and M_2 have rejected, let M reject. Consequently, if neither M_1 nor M_2 accepts, and M_1 or M_2 loops, then M will loop.

Intersection As soon as one of M_1 or M_2 rejects, let M reject. Once both M_1 and M_2 have accepted, let M accept. Consequently, if M_1 or M_2 loops, and neither M_1 nor M_2 rejects, then M will loop.

[1] the proof of the construction is given on pages 152–153 of M. Sipser's book

It follows that the recursive languages are closed under union and intersection, and that the r.e. languages are closed under union and intersection.

10.4 Complementarity. Given a DTM M, let $\overline{M}$ be the DTM that results from M by swapping q_a and q_r. If M is a Turing decider, then $L(\overline{M}) = \Sigma^* \backslash L(M)$. It follows that the recursive languages are closed under complementation. However, if M loops on some inputs, then $L(\overline{M}) \subset \Sigma^* \backslash L(M)$. So, the r.e. languages may not be closed under complementation. A language L is *co-r.e.* if there is a DTM M such that $L(M) = \Sigma^* \backslash L$; that is, M accepts the complement of M. For r.e. languages L, there is a deterministic TM that is guaranteed to accept all inputs that are in L, but it may not halt on inputs that are not in L. For co-r.e. languages L, there is a deterministic TM that is guaranteed to reject all inputs that are not in L, but it may not halt on inputs that are in L.

Theorem. A language L is recursive iff L is both r.e. and co-r.e.

We will soon see a language that is r.e. but not recursive, namely, the membership problem for DTMs. It follows that r.e. and co-r.e. are different language classes.