Ryan Darras, CS 5700 - HW09 3.(15, 16(a-d), 20)

3.15 and 3.16 are incredibly similar questions. The primary thing to note is that a language is only turing recognizable if there is a turing machine that will halt and not run forever for this language. The language is decidable if a turing machine recognizes it, but loops forever.

Problem 3.15

Show that the collection of decidable languages is closed under the operation of

a) Union
b) Concatenation
c) Star
d) Complementation
e) Intersection

Problem 3.15a Answer

For two decidable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the union of L_1 and L_2.

“On input w:

1. Run M_1 on w. If it accepts, accept.
2. Run M_2 on w. If it accepts, accept, Otherwise, reject.”

If either is accepted, M accepts the union of L_1 and L_2. If both reject, M rejects.

Problem 3.15b Answer

For two decidable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the concatenation of L_1 and L_2.

“On input w:

1. Split w into w_1 and w_2 for every possible split. (IE: aba/, ab/a, a/ba, /aba)
2. Run M_1 on w_1. If it rejects, reject.
3. Run M_2 on w_2. If it accepts, accept; if it rejects, reject.”

After trying every possible solution of splitting w into w_1 and w_2 we don’t have any accepting answers, then we know that it isn’t closed. However, if the input that forms w is the concatenation of two inputs that are accepted by M_1 and M_2 then this shows that M will accept this language. Meaning decidable languages are closed under concatenation.

Problem 3.15c Answer

For a language L, we construct a TM M that recognizes L*.

“On input w:

1. Split w into w_1 w_2 … w_k for every possible splitting scenario.
2. Run M on w_i for all i for all possible splitting scenarios. If M accepts all w_i for all i, accept; otherwise reject.

If w can be split into w_1 w_2 … w_k such that every w_i is a member of L, then M will accept w, hence decidable languages are closed by star.

Problem 3.15d Answer

For a language L which has TM M that recognizes it, we construct a TM M' that recognizes $\overline{L}$.

"On input w:

1. Run M on w, if it accepts, reject; otherwise accept"

If our machine M' simply runs M and spits out the opposite result, then we have an accurate conclusion to the complementation rule. Meaning that decidable languages are closed under complementation.

Problem 3.15e Answer

For two decidable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the intersection of L_1 and L_2.

"On input w:

1. Run M_1 on w. If it rejects, reject.
2. Run M_2 on w. If it rejects, reject; if it accepts, accept.

Both must be accepted for the intersection to be accepted.

Problem 3.16

Show that the collection of Turing-recognizable languages is closed under the operation of

a) Union
b) Concatenation
c) Star
d) Intersection

Problem 3.16a Answer

For two turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the union of L_1 and L_2.

"On input w:

1. Run M_1 and M_2 alternatively on w step by step. If either accepts, accept. If both halt and reject, reject."

If either M_1 or M_2 accepts w, M accepts w. Because we run them side by side (in parallel) we know that if we find one, we won't continue running forever searching for the other. In the case that we ran M_1 first, and it happened to run forever, it would never check M_2.

Problem 3.16b Answer

For two turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the concatenation of L_1 and L_2.

“On input w:

1. Split w into w_1 and w_2 for every possible split. (IE: aba/, ab/a, a/ba, /aba)
2. Run M_1 on w_1. If it halts and rejects, reject.
3. Run M_2 on w_2. If it accepts, accept; if it halts and rejects, reject.”

After trying every possible solution of splitting w into w_1 and w_2 we don’t have any accepting answers, then we know that it isn’t closed. However, if the input that forms w is the concatenation of two inputs that are accepted by M_1 and M_2 then this shows that M will accept this language. Because we are looking to see if the turing-recognizable language is closed under concatenation, we need to verify that M_1 and M_2 don’t halt.

Problem 3.16c Answer

For a language L, we construct a TM M that recognizes L*.

“On input w:

1. Split w into w_1 w_2 … w_k for every possible splitting scenario.
2. Run M on w_i for all i for all possible splitting scenarios. If M accepts all w_i for all i, accept; otherwise reject.

If w can be split into w_1 w_2 … w_k such that every w_i is a member of L, then M will accept w. If M either halts or rejects any w_i we reject the entire thing as a whole.

Problem 3.16d Answer

For two turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the machines that recognize them. We construct a TM M that recognizes the intersection of L_1 and L_2.

“On input w:

1. Run M_1 on w. If it halts and rejects, reject.
2. Run M_2 on w. If it halts and rejects, reject; if it accepts, accept.

Both must be accepted for the intersection to be accepted.

Problem 20

Show that single-tape TMs that cannot write on the portion of the tape containing the input string recognize only regular languages.

Problem 20 answer

Regular languages are a subset of context free languages which are a subset of turing recognizable languages. The difference between regular languages and context free languages is the fact that you need to hold some information in order to know whether a language is context free or not. In the case of a context free language, we use a stack. With the same idea, a turing-recognizable language that isn't a context free language or regular requires a tape to hold that information. Because of this, we can construct a turing machine that only ever goes right across the input string and stops when it sees an empty space to the right of its character. This machine will accept only regular languages, because all it has access to is the input string itself and it never needs to write to the tape at all.