COMP3721 Tutorial 9

R.E. Languages

Q1. We know that the class of recursively enumerable languages is not closed under complementation. Show that it is closed under union and intersection.

R.E. Languages

Q1. We know that the class of recursively enumerable languages is not closed under complementation. Show that it is closed under union and intersection.

Union: If L_1 and L_2 are recursively enumerable languages then there exists two Turing machines M_1 and M_2 that semi-decide L_1 and L_2, respectively. Let M be the 2-tape Turing machine that operates as follows:

(i) Copy the input string w from the first tape to the second tape.

(ii) Simulate M_1 on the first tape and M_2 on the second tape alternatively (i.e. do one step of M_1 on first tape, then one step of M_2 on second tape, and so on). If M_1 or M_2 halts, then M halts.

We claim M semi-decides $L_1 \cup L_2$.

R.E. Languages

Q1. We know that the class of recursively enumerable languages is not closed under complementation. Show that it is closed under union and intersection.

Verification:

If $w \in L_1 \cup L_2 \Rightarrow w \in L_1$ or $w \in L_2 \Rightarrow$ (M_1 halts) or (M_2 halts) $\Rightarrow M$ halts.
If $w \notin L_1 \cup L_2 \Rightarrow w \notin L_1$ and $w \notin L_2 \Rightarrow$ (M_1 doesn't halt) and (M_2 doesn't halt) $\Rightarrow M$ doesn't halt.

R.E. Languages

Q1. We know that the class of recursively enumerable languages is not closed under complementation. Show that it is closed under union and intersection.

Intersection: If L_1 and L_2 are recursively enumerable languages then there exists two Turing machines M_1 and M_2 that semi-decide L_1 and L_2, respectively. Let M be the Turing machine that operates as follows:

(i) Simulate M_1 on the string w.

(ii) If M_1 halts, then simulate M_2 on the string w.

(iii) If M_2 halts, then M halts.

We claim M semi-decides $L_1 \cap L_2$.
Verification omitted.

Problem (a)

(a) Given a Turing machine M, a state q, and a string w, does M ever reach state q when started with input w from its initial state?

Problem (a)

(a) Given a Turing machine M, a state q, and a string w, does M ever reach state q when started with input w from its initial state?

Solution: This problem is undecidable. Suppose it was decidable, then there exists some Turing machine M_A that decides it. It can be used to solve the halting problem:
M_H: on input "M" "w"

1. Run M_A("M" "w" "h") where h is the halting state of M.
2. If M_A output y, M_H output y; If M_A output n, M_H output n.

Problem (a)

(a) Given a Turing machine M, a state q, and a string w, does M ever reach state q when started with input w from its initial state?

Verification: On input "M" "w"
If "M""w"$\in$ H, i.e. M halts on w, then M_A on input "M" "w" "h" will halt at y and so M_H will also halt at y.
If "M""w"$\notin$ H, i.e. M does not halt on w, then M_A on input "M" "w" "h" will halt at n, and so M_H will also halt at n.

Problem (b)

(b) To determine, given a Turing machine M and a symbol σ, does M ever write the symbol σ when started on the empty tape?

Problem (b)

(b) To determine, given a Turing machine M and a symbol σ, does M ever write the symbol σ when started on the empty tape?

Solution: This problem is undecidable. Suppose it was decidable, then there exists some Turing machine M_B that decides it. Then, it can be used to solve the empty string problem:
M_E: On input "M",

1. Let a be a symbol that is not in the alphabet of M. Construct a Turing machine M^* that is identical to M except that whenever it halts it also writes an a. (Clearly, M^* writes an a when started on the empty tape if and only if M halts when started on the empty tape.)
2. Run M_B("M^*""a").
3. If M_B output y, M_E output y; If M_B output n, M_E output n.

Verification omitted.

Problem (c)

(c) Given a Turing machine M and an input string w, does M use a finite amount of tape squares on input w?

Problem (c)

(c) Given a Turing machine M and an input string w, does M use a finite amount of tape squares on input w?

Solution: This problem is undecidable. Suppose it was decidable, then there exists some Turing machine M_C that decides it. Then, it can be used to solve the halting problem:

M_H: on input "M" "w"

1. Construct a Turing machine M^* that runs M on w. At the same time, M^* uses a unary counter to record the number of steps M has run so far. If M never halts, this unary counter uses infinite number of tape squares. If M halts, then M^* uses finite number of tape squares. (M^* uses a finite number of tape squares on input w if and only if M halts on w.)
2. Run M_C("M^*" "w")
3. If M_C output y, M_H output y; If M_C output n, M_H output n.

Verification omitted.