Name ___________________________

Homework 9

SOLUTIONS

1. For each of the language classes: *regular, context-free, recursively enumerable,* and *recursive*, and each operation: *complement, union, intersection, concatenation,* and *closure, indicate whether the class is closed under that operation by filling in the corresponding spot in the following table with “yes” or “no”.

	regular	context-free	recursively enumerable	recursive
complement	**Yes**	**No**	**No**	**Yes**
union	**Yes**	**Yes**	**Yes**	**Yes**
intersection	**Yes**	**No**	**Yes**	**Yes**
concatenation	**Yes**	**Yes**	**Yes**	**Yes**
*-closure	**Yes**	**Yes**	**Yes**	**Yes**

#2. Suppose L_1, L_2, and L_3 are recursively enumerable languages over the alphabet Σ such that

(a) $L_i \cap L_j = \Phi$ when $i \neq j$; and

(b) $L1 \cup L2 \cup L3 = \Sigma^*$

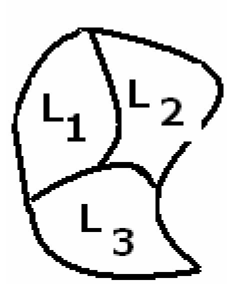
Prove that L_1 is recursive. Indicate clearly how each of the hypotheses (a) and (b) are used in your argument.

The argument is that both L1 and its complement (L2UL3) are re, so L1 is recursive. (This is equally true for L2 and L3). In more detail:

Because $L1 \cup L2 \cup L3 = \Sigma^*$, their union is all possible strings over Σ^* .

(a) says that the languages have no strings in common so

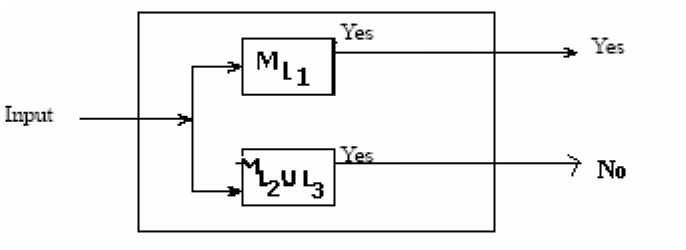
Σ^* looks like this:



Thus $\sim L_1 = L_2 \cup L_3$

Since L_1 is re, $\exists$ a TM, M_{L1} that accepts it.

Because L_2 and L_3 are re, their union is re and ∃ a TM M_{L2UL3} that accepts this union (which is the complement of L_1). Putting these together:



This shows L_1 is recursive.

#3. Suppose L is a recursive language. Prove that L^* is recursive. Write this out in words rather than using pictures.

Can be done by induction using the fact that recursive languages are close under union (think about it!)

#4. Complete each sentence with one of the answers:

- *regular*
- *context-free*
- *Recursive*
- *Recursively enumerable*
- *none of the above*

Make the strongest true assertion you can. In each case you should explain your answer by reference to the appropriate closure properties of the relevant language classes. (B^c *means the complement of B)*

(a) If A is regular and B is regular, then $A \cup B^c$ is: : **regular**

Proof **The regular languages are closed under complementation, so B^c is regular. The regular languages are closed under union, so $A \cup B^c$ is regular**

(b) If A is context-free and B is regular, then $A \cup B^c$ is: **c-f**

Proof. **The regular languages are closed under complementation, so B^c is regular. Any regular language is context-free, so B^c is context-free. The CF languages are closed under union, so $A \cup B^c$ is CF.**

(c) If A is regular and B is context-free, then $A \cup B^c$ is: recursive

***Proof*. The CF languages are not closed under complementation, so B^c is not necessarily CF; but CF languages are recursive, and the recursive languages are closed under complementation, so B^c is recursive. Regu;ar languages are recursice and recursive languages are closed under union, so $A \cup B^c$ is recursive.**

(d) If A is recursive and B is recursive, then $A \cup B^c$ is: *recursive.*

***Proof*. The recursive languages are closed under complementation, so B^c is recursive. The recursive languages are closed under union, so $A \cup B^c$ is recursive.**

(e) If A is recursive and B is Recursively enumerable, then $A \cup B^c$ is: ***none***

***Proof*. The RE languages are not closed under complementation. So take *B* to be a language whichis RE, yet B^c is not RE, (for example, let *B* be the language corresponding to the halting problem). Then take *A* to be the empty language. Then *AU* B^c is just B^c, so we cannot make any claims about it.**

(f) If A is recursively enumerable and B is recursive then $A \cup B^c$ is: re

***Proof*. The recursive languages are closed under complementation, so B^c is recursive. Every recursive language is RE, so B^c is RE. The RE languages are closed under union, so $A \cup B^c$ is RE.**

#5. Show the smallest category (regular, c-f, recursive, re, non-re) that each of the following is in. Prove that the language is in that category and prove that it is not in the next innermost category.

a) L(G) where G:

$$S \rightarrow aB \mid bA$$
$$A \rightarrow aS \mid bAA \mid a$$
$$B \rightarrow bS \mid aBB \mid b$$

Clearly context-free.

Flawed answers: 1. The grammar is not regular; therefore, the language it generates is not regular: could be another regular grammar generating L(G) that *is* regular. A couple of you said it couldn't be converted, but no one gave a reason.

2. a^kb^k is in L and when pumped yields a string not of that form. But there are lots of other strings in the language and the pumped string might be one of them.

<u>SOLUTION</u>

L(G) = {w | w has an equal number of a's and b's}

Using the pumping lemma, you can pump a^kb^k so it doesn't have an equal number of a's and b's

b) L(G) where G:

$$S \rightarrow a\,S \mid b\,S \mid \varepsilon$$

L(G) = (aUb)* so regular

c) $L_1 \cap L_2$ where L_1 and L_2 are context-free

Recursive. Not (necessarily) c-f because intersection of c-f languages is not c-f

d) The set of encodings of Turing Machines M whose time complexity is not bounded by n^2; i.e., L = {M | there exists an input string w such that M performs more than $|w|^2$ steps on input w}

Recursively enumerable

#6. Consider the following language:

HALT = {< *M, w* > | *M* is a Turing machine and $w \in L(M)$}

Here < M,w > denotes some encoding of the pair consisting of a description of machine M and of input w, so that HALT is an encoding of the set of machine-input pairs such that the machine accepts the input. The specific details of the encoding itself are irrelevant to the questions which follow.

The theorem on the undecidability of the halting problem states that HALT is not decidable. For this problem, take that theorem as given.

(a) Is the language HALT recursively enumerable? Defend your answer.

(b) Is the complement of HALT recursively enumerable? Defend your answer.

The language HALT is Turing-recognizable (RE), but its complement is not RE.

The fact that HALT is RE follows immediately from the fact that there exists a universal Turing machine. Indeed, the set HALT is precisely the domain of this universal Turing machine. For some authors being the domain of a TM is the *definition* of being RE, so in that case we are done. Some authors define RE sets as the accepted sets of special Turing machines, the "acceptors", but it is a trivial matter to build a universal acceptor from the classical universal Turing machine.

To prove that $HALT_c$ is not RE, we invoke the fact that if a set and its complement are each RE, then that set is in fact recursive. So if $HALT_c$ were RE, then by the previous paragraph it would be recursive, which we know it is not.

#7. Decidable or undecidable: whether a language is regular or not. Prove your answer. (Hint: Look at Reg_{TM} = {M | M is a TM and L(M) is regular})

Vijay, you can find this all over the web

#8. Show that there is no algorithm that determines whether an arbitrary Truing machine prints a 1 on its final transition.

#9. Show whether the following instance of PCP has a solution or not. (01,011), (001,10), (10,00)

ditto

#10. Explain the differences between NP, NP-Complete and undecidable briefly but clearly (I know we have not covered this in class)