Main topics of the week:

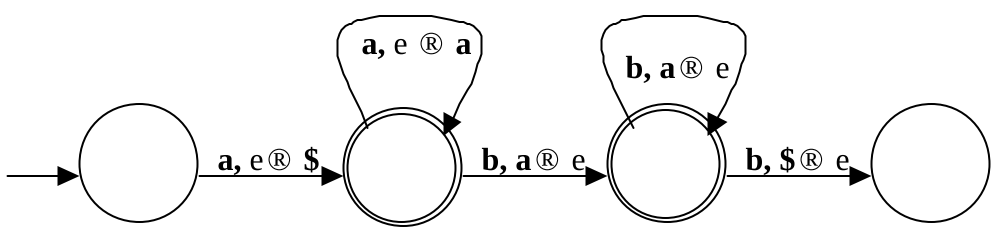
- **Countable and uncountable sets**
- **Diagonalization proof of uncountable**
- **Undecidable halting problem A_{TM}**
- **Complement of A_{TM} is not even Turing recognizable**
- **Review for final**

Review problems for final.

Let G be the grammar **S ® aS | aA | a**
A ® aAb | ab

Give a concise description of L(G) and a PDA to accept L(G).

We observe that the symbol A generates $a^n b^n$ for all $n \geq 1$. The rules for S using aA and a simply add another 'a' in front, so that gives us strings of the form $aa^n b^n$ for all $n \geq 0$. The rule aS for S allows an arbitrary number of a's to be added at the beginning, so L(G) = $\{ a^m b^n \mid m > n \geq 0 \}$.



Show that the C programming language is not a context free language.

Proof: Suppose that C is a CFL. By the pumping lemma for CFLs, there must be a pumping length p. Consider the legal C program:

```
f(){int aa…a;aa…a;aa…a;}
```

In this program, the variable is a^p, i.e., the symbol 'a' repeated p times. Suppose we realize this string as uvxyz, where $|vxy| \leq p$. If vy contains the blank or any of the symbols preceding it, then we certainly have a syntax error in uxz or at least an undeclared variable. If vy contains the last semi-colon or closing brace, then again uxz will have a syntax error. So vxy must be between the space and the last semi-colon. If vy contains a semi-colon and possibly some characters before and after, again uxz will have an undeclared variable since one set of the a's will combine to be at least p+1 a's while the other stays the same. If vxy contains no semi-colon, then uvvxyyz will have a second identifier, so again the program will not be legal since something will be undeclared. This exhausts all the possibilities; so legal C programs are not a context free language.

Consider the language A = { a^i b^j a^m b^n | i + j £ m + n }. Is A regular? Is A context free?

A is *not* regular. *Proof*: We prove this using the pumping lemma. Let s=$a^p ba^p b$, and realize this as xyz. Since $|xy| \leq p$, y must consist of just a's. Then xyyz will be of the form $a^{p+k}ba^p b$ where $k \geq 1$ from which it follows that p+k+1>p+1 showing that xyyz $\notin$ A, a contradiction to the pumping lemma. Thus A cannot be regular.

A *is* context free. *Proof*: We can construct a PDA to recognize A. The operation of this PDA pushes an end of stack marker $'. Then it moves to a state where it loops and pushes x on the stack for each a. In the next state, it pushes x on the stack for each b. The next state pop's an x for each a, or pops and pushes a '$' for each a. The next state does the same thing for b. Finally, we move to an accept state if the stack is empty.

Let A = { *w* | *w* Î {a,b,c}* and *w* has fewer a's than b's and fewer b's than c's}. Prove that A is not a context free language.

Proof: Suppose that A is context free, and let p be the pumping length. Let s=$a^p b^{p+1} c^{p+2} \in$ A. If s =uvxyz with the constraints of the pumping lemma, then we know that $|vxy| \leq p$ means that vxy can span at most two distinct symbols, i.e., it could have a's and b's but no c's, or b's and c's, but no a's.

Case 1: v or y have at least one c. If v or y contains a b, then uxz has fewer b's but the same number of a's, so there would be at least as many a's as b's and uxz would not be in A. If there are no b's in v or y, then vxz has the same number of b's, but fewer c's, which means at least as many b's as c's, so uxz would not be in A in this case either.

Case 2: there are no c's in v or y. If v or y has at least one b, then uvvxyyz has at least one more b, but the same number of c's, so there are at least as many b's as c's, and uvvxyyz would not be in A. If there are no b's in v or y, then uvvxyyz has more a's, but the same number of b's, so there are at least as many a's as b's and thus this is not in A.

Both cases led to a contradiction, so A cannot be a context free language.

Let A = {<R,S> | R and S are regular expressions and L(R) Í L(S) } Show that A is decidable.

Proof: We describe a TM that decides A:

"On input <R,S> where R and S are regular expressions,

1) Convert the regular expression R to an equivalent DFA A, using the procedure for converting a regular expression to an NFA, and the procedure for converting an NFA to a DFA.
2) Likewise convert the regular expression S to a DFA B.
3) Using the procedure from an exercise, convert B to a DFA C that recognizes the complement of L(S).
4) Using the procedure from the problem on the midterm (just like the one for the union of regular languages), construct a DFA D from A and C to recognize $L(R) \cap L(\overline{S})$.
5) Run the TM for E_{DFA} on D to determine if $L(R) \cap L(\overline{S}) = \emptyset$.
6) If it is empty, *accept*, otherwise *reject*."

Note that each stage of our TM uses procedures that we know to halt, so this is a decider. By determining if the intersection of L(R) with the complement of L(S) is empty, we determine whether $L(R) \subseteq L(S)$ or not.

Let S ={<M> | M is a DFA that accepts w^R whenever M accepts w}. Show that S is decidable.

Proof: We describe a TM that decides S. Basically we test to see whether the language recognized by M is the same as the reverse of that language.

"On input <M>, where M is a DFA,

1) Construct a DFA N to recognize $\{w^R \mid w \in L(M)\}$. (Recall that we saw in a problem how to construct an NFA to do this by adding a single accept state and reversing all the arrows, and we know how to convert an NFA to a DFA).
2) Run the TM for EQ_{DFA} with input <M,N>.

If it accepts, *accept*, otherwise *reject*."