Decidability

Here are some interesting examples of decidable languages:

Every regular language is decidable.

Every context-free language is decidable.

A_{DFA} = {<B,w> | B is a DFA that accepts w} is decidable.

A_{NFA} = {<B,w> | B is a NFA that accepts w} is decidable.

A_{REX} = {<R,w> | R is a regex that generates w} is decidable.

A_{CFG} = {<G,w> | G is a CFG that generates w} is decidable.

E_{DFA} = {<A> | A is a DFA and L(A) = $\varnothing$}, i.e., a DFA that accepts no strings, an empty language, is decidable.

EQ_{DFA} = {<A,B> | A and B are DFAs and L(A) = L(B)} is decidable.

E_{CFG} = {<G> | G is a CFG and L(G) = $\varnothing$} is decidable.

Here are a few examples of languages that are NOT decidable:

EQ_{CFG} = {<G,H> | G and H are CFGs and L(G) = L(H)} is NOT decidable.

A_{TM} = {<M,w> | M is a TM that accepts w} is NOT decidable.

$HALT_{TM}$ = {<M,w> | M is a TM that halts on w} is NOT decidable.

This last ones regard the **Halting Problem**. It is very important and means that no algorithm exists that can verify that any given program does what it should, let alone ever halt.

Importantly, A_{TM} is Turing-recognizable (obvious), but it's complement $A_{TM}{}^{C}$ is not Turing-recognizable. It can't be -- if it were, we could build a decider, but A_{TM} is not decidable. **This would make a nice final exam question, wouldn't it!**This is another way of saying Turing-recognizable languages are not closed under complements, see @147.

Below are some of the proofs of the above:

$A_{DFA} : M$ = "On input <B,w>, simulate B on w (it replicates the DFA's states and runs it on w). If the simulation ends in an accept state, accept. Else reject."

$A_{NFA} : P$ = "On input <R,w>, convert R into a DFA, B, using the GNFA procedure then run TM M above on <B,w> accepting or rejecting as M would."

$E_{DFA} : T$ = "On input <A>, in a breadth-first manner, mark reachable states from the start state. If no accept state is ever marked, then it will never accept and the language is empty so accept, else reject."

$EQ_{DFA} : F$ = "First, recognize that the symmetric difference, $L(C) = (L(A) \cap L(B)^{C}) \cup (L(A)^{C} \cap L(B)^{C})$, is empty if and only if L(A) = L(B). Thus, we first construct C using the well-known DFA closure constructions. Then we run the above machine T on <C>. If T accepts, accept; if it rejects, reject."

A_{TM} : Suppose by contradiction it is decidable, let H be the decider. On input <M,w> H accepts when M accepts w, and rejects when M does not accept w (reject or loop). Construct D(<M>), which runs H on <M,<M>> and does the opposite: rejects when M accepts <M>, accepts when M does not accept <M>. Now if we run, D on <D> it contradicts itself: accept if

D does not accept <D>, reject if D accepts <D>. Thus neither D nor H can exist, so A_{TM} is undecidable. **This would make a nice final exam question, wouldn't it!**

$HALT_{TM}$: Suppose by contradiction it is decidable, let R be the decider. Now we can decide A_{TM} by first running R in <M,w> to first check if it halts. If it halts, then go ahead and run it and accept or reject as it normally would (it is safe to run); if not, just reject (don't run it because it will loop). But deciding A_{TM} is not possible since it is undecidable! This contradiction means $HALT_{TM}$ is undecidable. **This would make a nice final exam question, wouldn't it!**