Closure properties of RE, Rec

Recall that a language L is
- recursively enumerable (RE) if there exists a TM for L,
- recursive (Rec) if there exists a TM for L which halts on every input (i.e. also on strings not from L).

Is the class RE closed under union? And intersection?

Is the class Rec closed under union? And intersection?

What can we say about complement?  YES for Rec
Closure properties of RE, REC

Lemma: RE languages are closed under union.

Option 1: create a NTM $T_4$ with $Q_4 = Q_1 \cup Q_2 \cup \{q_o\}$, assuming all disjoint.

Given TM $T_1, T_2$, create a TM $T_3$ s.t.
$L(T_3) = L(T_1) \cup L(T_2)$

Then convert $T_4$ into a det. $T_3$.

Option 2: run the TMs simultaneously (one step for both each time).

Lemma: RE languages are closed under intersection.

Given TM $T_1, T_2$, create a TM $T_3$ s.t.
$L(T_3) = L(T_1) \cap L(T_2)$

Clean up the tape after $\varepsilon$ (including it) and go to the first $\Delta$.

Creates a copy of the input:

```
\$\Delta x \Delta$
```

Assume $\varepsilon \in T_1$ when going to $T_1$. 

Also, $\varepsilon$
Lemma: Recursive languages are closed under union.

Lemma: Recursive languages are closed under intersection.

idea: same as before (for RE)
Lemma : Recursive languages are closed under complement.

   easy: switch $L_a$ and $L_r$

Lemma : RE languages are not closed under complement.

   See next Thm

   Pf: if RE is closed under complement, then for $L \in RE$, $L' \in RE$ by the Thm: $L \in \text{rec}$.
   but we will see that $\text{rec} \nsubseteq \text{RE}$ so it is not possible.
Closure properties of RE, Rec

Thm: L and L' are RE iff L is recursive.

If: easy: if L is rec. then (by def.) L is RE
    then L' rec. therefore RE.

⇒ assumption: L, L' are RE, i.e. ∃TM T₁, T₂ s.t. L(T₁) = L
    L(T₂) = L'

goal: create T₃ s.t. L(T₃) = L and
    never goes to an infinite loop

idea: run T₁ & T₂ simultaneously:
    if T₁ accepts then x ∈ L and we go to ha
    if T₂ accepts then x ∈ L' and we go to hr

NO OTHER CHOICES since T₁ accepts L and T₂ accepts L'
Noam Chomsky studied grammars as potential models for natural languages. He classified grammars according to these four types:

- **Type 0 Grammars**: *Unrestricted Grammars* (generate RE languages)
- **Type 1 Grammars**: *Context-sensitive (monotone) Grammars* (generate context-sensitive languages)
- **Type 2 Grammars**: *Context-free Grammars* (generate context-free languages)
- **Type 3 Grammars**: *Regular Grammars* (generate regular languages)
Def: An unrestricted grammar is a 4-tuple $G = (V, \Sigma, S, P)$ where
- $V$ is a finite set of variables
- $\Sigma$ is a finite set of terminal symbols
- $S \in \Sigma$ is the start symbol
- $P$ is a finite set of productions of the form $\alpha \rightarrow \beta$ where $\alpha \in (V \cup \Sigma)^+ \text{ and } \beta \in (V \cup \Sigma)^*$

$(V \text{ and } \Sigma \text{ are assumed to be disjoint})$
Example: Give an unrestricted grammar for \( \{ a^k b^k c^k \mid k \geq 0 \} \)
Example: Give an unrestricted grammar for \( \{ a^j \mid j = 2^k, k \geq 0 \} \)
Context-sensitive Gram. (Type 1)

Def: A type 0 grammar $G = (V, \Sigma, S, P)$ is context-sensitive if for every production rule $\alpha \rightarrow \beta$ in $P$, $|\alpha| \leq |\beta|$.

Which of our examples of type 0 grammars are context-sensitive?
Context-sensitive Gram. (Type 1)

Def: A type 0 grammar $G=(V, \Sigma, S, P)$ is context-sensitive if for every production rule $\alpha \rightarrow \beta$ in $P$, $|\alpha| \leq |\beta|$.

Lemma: Every context-free language which does not contain $\Lambda$ is context-sensitive.
**Context-sensitive Gram. (Type 1)**

**Def:** A type 0 grammar $G=(V, \Sigma, S, P)$ is **context-sensitive** if for every production rule $\alpha \rightarrow \beta$ in $P$, $|\alpha| \leq |\beta|$.

**Lemma:** Every context-free language which does not contain $\Lambda$ is context-sensitive.

**Def:** A linear-bounded automaton $A$ is a TM which never rewrites a blank to a non-blank symbol.

**Lemma:** A language $L$ is context-sensitive iff there exists a linear-bounded automaton accepting $L$. 

[Section 10.3]
Def: A type 0 grammar $G=(V, \Sigma, S, P)$ is regular if every production rule in $P$ is of the form $A \rightarrow \sigma B$ or $A \rightarrow \sigma$, where $A, B \in V$ and $\sigma \in \Sigma$.

Lemma: A language $L$ is regular iff there exists a regular grammar for $L-\{\Lambda\}$. 