CS 360
Fall 2021
The Language Hierarchy

Jeffrey Shallit
School of Computer Science
University of Waterloo
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit
The hierarchy

We can arrange the language classes we’ve studied so far into a hierarchy, as follows:
The hierarchy

\[ \text{FINITE} = \text{the finite languages} \]

\[ \text{REG} = \text{the regular languages} \]

\[ \text{CFL} = \text{the context-free languages} \]

\[ \text{REC} = \text{the recursive languages, aka the Turing-decidable languages} \]

\[ \text{RE} = \text{the recursively enumerable languages, aka the Turing-recognizable languages} \]

\[ \text{ALL} = \text{all languages} \]

We want to know, is this really a hierarchy? Is each class contained in the larger class?
And is it a strict hierarchy? Is there always a language in the larger class that is not in the smaller class? Let’s answer this one first.

Example of **FINITE**: \(\{ \text{a} \}\)

Example of **REG − FINITE**: \(a^* = \{a^i : i \geq 0\}\)

Example of **CFL − REG**: \(\{a^n b^n : n \geq 0\}\)

Example of **REC − CFL**: \(\{a^n b^n c^n : n \geq 0\}\)

Example of **RE − REC**: 
\(A_{\text{DTM}} = \{e(T)e(w) : T \text{ is a TM accepting } w\}\).

Example of **ALL − RE**: \(\text{DIAG} = \{e(T) : T \text{ does not accept } e(T)\}\).
Let’s prove that each class is contained in the next larger class.

**FINITE \(\subseteq\) REG:** Given a finite language \(L = \{x_1, \ldots, x_n\}\), we can make a regular expression out of the elements of \(L\): \(\emptyset\) if \(n = 0\), \(x_1\) if \(n = 1\), and \(x_1 + x_2 + \cdots + x_n\) if \(n \geq 2\).

**REG \(\subseteq\) CFL:** A DFA accepting \(L\) can be considered a PDA, just one that does not use the stack at all. So if \(L\) is recognized by a DFA, it’s also recognized by a PDA.

**REC \(\subseteq\) RE:** A TM that recognizes \(L\) and always halts still recognizes \(L\).

**RE \(\subseteq\) ALL:** By definition.

Only one is left to do (on the next slide).
It remains to prove that $\text{CFL} \subseteq \text{REC}$.

We need to show that if $L$ is a CFL, it can be decided by some TM, that is, recognized by a halting TM $T$.

We know that if $L$ is a CFL, there is a CFG for it.

Even more, there is a CFG $G$ in Chomsky normal form generating $L - \{\epsilon\}$.

And we know that a CFG $G$ in Chomsky normal form generates the string $w$ iff there is a derivation of length $2|w| - 1$ for it.

So we just make a TM that tries all possible derivations of this length to see if any equal $w$. 
Now, more details: Suppose the largest number of right-hand sides of any variable is $B$.

We now create a 5-tape TM $T$ to behave as follows: if the input is $\epsilon$, we either accept or reject, based on whether $\epsilon \in L$. Otherwise ...

Tape 1 holds the input $w$. It is never changed.

Tape 2 holds the grammar $G$.

Tape 3 is a work tape, where we carry out the various possible leftmost derivations.

Tape 4 holds, successively, the strings of length $2|w| - 1$ over the alphabet \{0, 1, \ldots, B - 1\}. These strings represent which choices to make in any particular derivation.

Tape 5 is a counter to count the number of steps in a derivation.
We now do the following:

1. We write $S$ on Tape 3.

2. We scan Tape 3, looking for a variable to replace. When we find one, we use the next symbol $a$ on Tape 4 to decide which of the various rhs to use, namely the $a$'th choice.

3. We replace the variable on Tape 3 with its rhs, shifting symbols down if necessary, to make room.
4. We update the counter on Tape 5.

   ▶ If it is less than $2|w| - 1$, we go back to step 2.
   ▶ If it is equal to $2|w| - 1$, compare the contents of Tape 3 to the input on Tape 1, and accept if they agree.
   ▶ If it is greater than $2|w| - 1$, we erase Tape 3, and use a subroutine to create the next string on Tape 4 (like in our binary incrementer).

If the current string of choices on Tape 4 was $(B - 1)(B - 1) \cdots (B - 1)$, we’re done and we reject.

Otherwise, we go back to step 1.
More about the hierarchy

So now we have completely justified this picture:

But to be honest, it's a little deceptive. This is what it really looks like:
The largest class of languages we can access computationally, namely RE, is just a tiny insignificant dot in an ocean of languages we can never access that way.