CSE 396 lecture 4/20 Spr 2017

REG = the <u>class</u> of regular languages.

DCFL = the class of languages accepted by DPDAs aka "deterministic CFLs" but there is no universally agreed notion of "determ. CFG."

CFL = the class of L(G) for CFGs G, which <u>equals</u> the class of L(N) for <u>N</u>PDAs N.

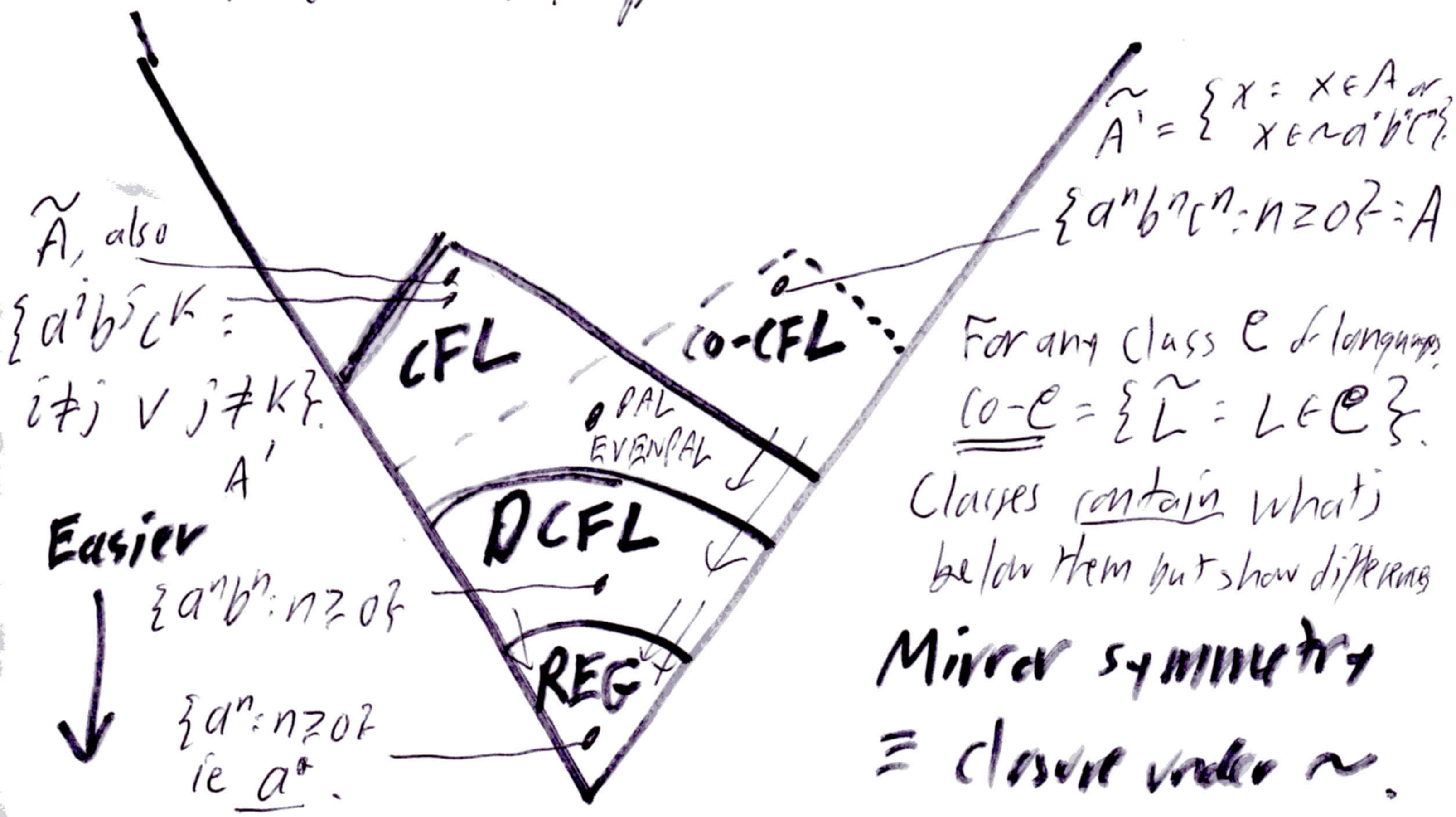
Theorem in text but proof skipped.

Brief Idea: $S \to AB$ ≡ Pop S, push B, push A.

$\ldots \Rightarrow \underline{S}AC \Rightarrow ABAC$ (stack ←)

$A \to a$ ≡ read a and pop A.

This is the CFG-to-PDA part. The converse is <u>very</u> hard.



For any class $\mathcal{C}$ of languages, <u>co-$\mathcal{C}$</u> $= \{\tilde{L} : L \in \mathcal{C}\}$.

Classes <u>contain</u> what's below them but show differences.

Mirror symmetry ≡ closure under ~.

The diagram does not tell about closure under $\cup$ or $\cap$, only $\sim$.

__Fact:__ REG is closed under all of them. DCFL is closed under $\sim$

($S \to S_1 | S_2$ idea) early CFL lecture $\to$ CFL is closed under $\cup$

Also: $A_1 = \{a^i b^j c^k : i \neq j\}$ is a DCFL. A DPDA can compare a's & b's, ignore c's.

$A_2 = \{a^i b^j c^k : j \neq k\}$ ditto. $\therefore A' = A_1 \cup A_2$ is a union of two DCFLs, but not a DCFL.

__And:__ A is the intersection of two DCFLs but is not even a CFL. $A = \underbrace{\{a^i b^j c^k : i = j\}}_{= \tilde{A}_1 \cap a^* b^* c^*} \cap \underbrace{\{a^i b^j c^k : j = k\}}_{= \tilde{A}_2 \cap a^* b^* c^*}$.

$\boxed{\text{Finally: MARKEDPAL} = \{w\#w^R\} \text{ is a DCFL but EVENPAL} = \{ww^R : w \in \{0,1\}^*\} \text{ is not but both it and its complement are CFLs}}$

idea shown on Tue, proof is __hard__.

Some New Nomenclature and Classes.

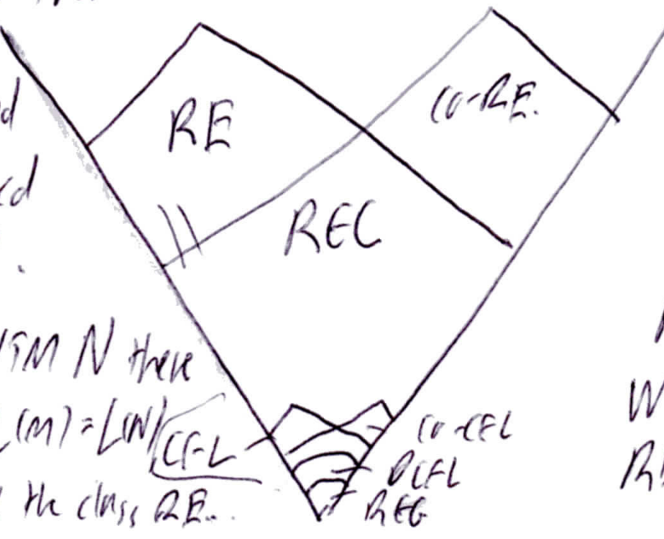
Defn: A language L is

Synonyms
- Turing-recognizable
- Turing-acceptable
- recursively enumerable (r.e.)
- computably enumerable (c.e.)

if there is a DTM M such that $L = L(M)$. The class of such languages is (always!) written __RE__.

If also the TM M is __total__, i.e. __halts for all inputs__, then $L = L(M)$ is called {__decidable__, __recursive__}. The class of such languages is usually called __REC__, sometimes __DEC__.

"Cone Diagram" for them:



Also true: REC and RE are both closed under $\cap$ and $\cup$.

__Thm:__ For every NTM N there is a DTM M s.t. $L(M) = L(N)$ so NTMs also define the class RE.

Facts we __will__ prove:

REC is closed under $\sim$

RE $\cap$ co-RE does = REC

RE $\neq$ co-RE, so RE $\neq$ REC

We will fill all regions above REC with __undecidable__ problems.

New Content: Algorithms that Decide Problems

An alg^m that always halts is called a decision procedure and (technically by the Church-Turing Thesis) can be converted into a Turing machine that always halts; total TM, aka a decider.

Key Tool: Breadth-First Search.

Ch 4: Problems about machines / grammars / regexps M: Is $L(M) \neq \emptyset$? $= \emptyset$?
Is $L(M) = \Sigma^*$?
Does M accept a given $x \in \Sigma^*$?

Sipser's Problem Naming Scheme.

A_{DFA}: Given a DFA M and an input $x \in \Sigma^*$, does M accept x?
A for Acceptance Problem. The language of this problem is $\{\langle M, x\rangle : M$ is a DFA and M accepts $x\}$.

$\langle \cdots \rangle$ refers to some "transparent" way of coding the contents as a string.

E_{DFA}: Given just the DFA M, is $L(M) = \emptyset$? (E for empty)
Language: $\{\langle M\rangle : L(M) = \emptyset$ where M is a DFA$\}$. Call the language $L_{E_{DFA}}$ or just E_{DFA}.

Generally,
The language L_Π of a problem $\Pi = \{z \in \Sigma^* :$ the answer to Π for z is YES$\}$.

Note $\underline{\underline{NE}}_{DFA} = \{$DFAs $M : L(M) \neq \emptyset\}$. "Nonemptiness Problem".

Theorem: A_{DFA}, E_{DFA}, and NE_{DFA} are all decidable problems/languages.

A_{DFA}: decision procedure is simply "Given M and x, just run $M(x)$ and say yes if and only if M accepted x."

NE_{DFA}: Fact: $L(M) \neq \emptyset \iff$ some final state f is reachable from s.
Breadth first search thus gives a decision procedure to tell. ⊠