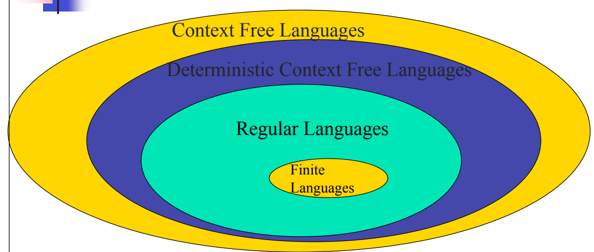
# Decision and Closure Properties of CFLs

## Now our picture looks like




## Closure Properties

- We already seen that CFLs are closed under:
  - Union
  - Concatenation
  - Kleene Star
- Regular Languages are also closed under
  - Intersection
  - Complementation
  - Difference
- What about Context Free Languages?

## Closure Properties

- Sorry, Charlie
  - CFLs are not closed under intersection
  - Meaning:
    - If $L_1$ and $L_2$ are CFLs then $L_1 \cap L_2$ is not necessarily a CFL.

## Closure Properties

- CFLs are not closed under intersection
  - Example:
    - $L_1 = \{a^ib^jc^k \mid i < j \}$
    - $L_2 = \{a^ib^jc^k \mid i < k \}$
    - Are both CFLs

## Closure Properties

- CFLs are not closed under intersection

| $L_1 = \{a^ib^jc^k \mid i < j \}$ | $L_2 = \{a^ib^jc^k \mid i < k \}$ |
|---|---|
| $S \rightarrow ABC$ | $S \rightarrow AC$ |
| $A \rightarrow aAb \mid \varepsilon$ | $A \rightarrow aAc \mid B$ |
| $B \rightarrow bB \mid b$ | $B \rightarrow bB \mid \varepsilon$ |
| $C \rightarrow cC \mid \varepsilon$ | $C \rightarrow cC \mid c$ |

## Closure Properties

- CFLs are not closed under intersection
  - $L_1 \cap L_2 = \{a^i b^j c^k \mid i < j \text{ and } i < k\}$
  - Which we just showed to be non-context free.

## Closure Properties

- Sorry, Charlie
  - CFLs are not closed under complement
  - Why?
    - $L_1 \cap L_2 = (L_1' \cup L_2')'$

## Closure Properties

- Sorry, Charlie
  - CFLs are not closed under difference
  - Why?
    - $L' = \Sigma^* - L$
    - We know $\Sigma^*$ is regular, and as such is also a CFL.
    - If CFLs were closed under difference, then $\Sigma^* - L = L'$ would always be a CFL
    - But we showed that CFLs are not closed under complement

## Closure Properties

- What went wrong?
  - Can't we apply the same construction as we did for the complement of RLs?
    - Reverse the accepting / non-accepting states
    - PDAs can "crash".
      - I.e Fail by having no place to go.
      - PDAs can "crash" in accepting or non-accepting state
      - Making non-accepting states accepting will not handle crashes.

## Closure Properties

- What went wrong?
  - Can't we apply the same construction as we did for the intersection of RLs?
    - The states of M are an ordered pair (p, q) where $p \in Q_1$ and $q \in Q_2$
    - Informally, the states of M will represent the current states of $M_1$ and $M_2$ at each simultaneous move of the machines.

## Closure Properties

- What went wrong?
  - Can't we apply the same construction as we did for the intersection of RLs?
    - The problem is the stack.
    - Although we could try the same thing for PDAs and have a combined machine keep track of where both PDAs are at any one time.
    - We can't keep track of what's on both stacks at any given tine.

## Closure Properties

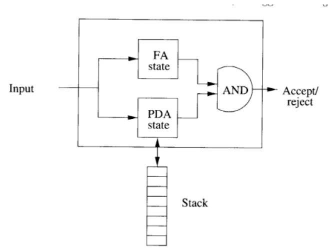
- However, if one of the CFLs does not use the stack (I.e. it is an FA), then we can build a PDA that accepts $L_1 \cap L_2$ .
- In other words:
  - If $L_1$ is a context free language and $L_2$ is a regular language, then $L_1 \cap L_2$ is context free.

## Closure Properties

- Basic idea:
  - Like with the FA construction, let the states of the new machine keep track of the states of the PDA accepting $L_1$ ($M_1$) and the FA accepting $L_2$ ($M_2$).
  - Our single stack of the new machine will operate the same as the stack of the PDA accepting $L_1$
  - Accepting states will be all states that contain both an accepting state from $M_1$ and $M_2$.

## Closure Properties

- Basic idea




## Closure Properties

- Summary
  - CFLs are closed under
    - Union, Concatenation, Kleene Star
  - CFLs are NOT closed under
    - Intersection, Difference, Complement
  - But
    - The intersection of a CFL with a RL is a CFL

## Decision Properties

- Questions we can ask about context free languages and how we answer such questions.

## Decision Properties

- Given regular languages, specified in any one of the four means, can we develop algorithms that answer the following questions:
  1. Is a given string in the language?
  2. Is the language empty?
  3. Is the language finite?

## Decision Properties

- Membership
  - Unlike FAs, we can't just run the string through the machine and see where it goes since PDAs are non-deterministic.
    - Must consider all possible paths

## Decision Properties

- Membership
  - Instead, start with your grammar in CNF.
    - The proof of the pumping lemma states that the longest derivation path of a string of size n will be 2n – 1.
    - Systematically generate all derivations with one step, then two steps, …, then 2n – 1 steps where the length of the string tested = n. If one of the derivations derive x, return true, else return false.

## Decision Properties

- Emptiness
  - Remove useless symbols and prouductions
  - If S is useless, then L(G) is empty.

## Decision Properties

- Finiteness
  - Just as with RLs, a language is infinite if there is a string x with length between n and 2n
    - With RLs n = number of states in an FA
    - With CFLs n = $2^{p+1}$ where p is the number of variables in the CFG
    - Systematically generate all strings with lengths between n and 2n
    - Run through membership algorithm
    - If one passes, L is infinite, if all fail, L is finite

## Decision Properties

- Questions?

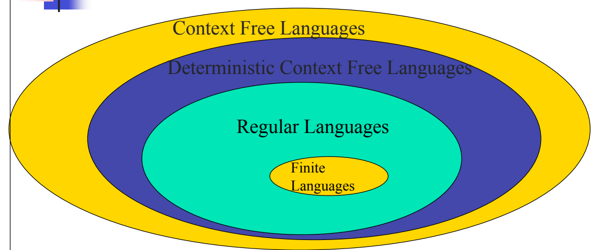
## Decision Properties

- Sad facts about CFLs
  - There is no "algorithm" to determine if, given a grammar G, G is ambiguous.
  - There is no "algorithm" to determine if two CFGs generate the same language.

## Summary

- Pumping Lemma for CFLs
- Closure Properties
- Decision Properties

## Now our picture looks like




Is there anything out here? YES

## Next Time

- Next classes of languages
- However,
  - Once again, we start with the machine rather than the language
  - Move beyond simple language acceptance into the realm of computation.
- Enter…The Turing Machine!!!