# Lecture 2 Notes

Scribed by Noah Frazier-Logue

## Frege's First-Order Logic

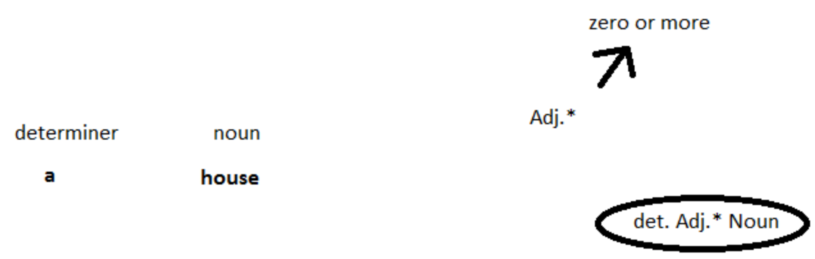
- Frege wrote a book where he introduced first-order logic
    - ∀: for all
    - ∃: there exists at least one
    - Example- S=∀ arrows A, A hits the target (assertion)
        - S=∀ arrows A, A does not hit the target (not quite a negation)
            - If there are 10 arrows, none hit
        - S=∃ arrow A, ¬A hits the target (negation)
            - At least one arrow doesn't hit the target
        - $S=\forall x \neg P(x)$
            - S'=∃A, A hits on the target
        - $\neg S=\exists x \neg P(x)$
            - S'=∀A, ¬A hits the target
- Before this logic, there was no way to talk about infinity
    - Important to appreciate theory/constructing a logic
        - More complicated logic
            - $S=\forall x\ \exists y\ y>x$ (for every number, there some number that's bigger)
            - $\neg S=\exists x\ \forall y\ \neg y>x$ (true for a finite set)
- The notation is what allows us to work within the logic
    - Ex. Multiplying with Roman numerals may be difficult

Thue Systems

- Thue: Mathematician
  - Defined a type of algebra that used strings of characters
    - $S_1$=ababaabb
    - $S_2$=bbbbb
  - Rewriting Rules: you're allowed to replace $substring_A \Leftrightarrow substring_B$
    - Rule 1) ab⇔b
    - Rule 2) ab⇔baa
      - Rewriting Rules R
        - $S_1 \equiv S_2$
        - =ababaabb
        - =abaabbb (Rule 2)
        - =baabbb (Rule 1)
        - =abbbb (Rule 2)
        - =bbbb (Rule 1)
        - Result: not equivalent
  - Someone proved that if you could always determine the answer to an equivalence problem, you could run the Haltcrazy program (see Lecture 1 notes)
    - This means that you could create the Halt program (and therefore the Haltcrazy program)

Linguistics

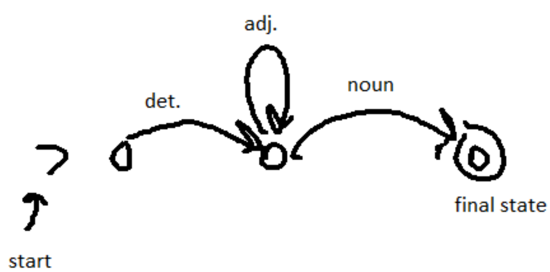
- We care about Thue systems because it's similar to what linguists do
  - Determining if certain sentences are equivalent
    - Language translators use bi-texts (used to parse texts, equate portions of text to one another)
      - I'm happy. ⇔ Je suis content. (Linguistic Rewriting Rule)
    - Languages that have fewer bi-texts are less accurate
- Noam Chomsky (1956)
  - Three Theories of Language
  - Before Chomsky:
    - Linguists went into a culture where nobody knew the language and learned it
      - Built a lexicon and established the grammar
  - Chomsky: What's happening in our minds when we create language?
    - No bound on the length of sentences
  - Noun Phrases



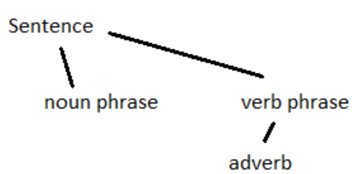

  - You can construct a sentence and know that it's syntactically permissible
    - Me eat good (bad syntax, decent semantics)

- Chomsky was asked: can you make a sentence with good syntax and bad semantics?
  - Result: “Colorless green ideas sleep furiously.”
- Certain words that seem normal to English-speakers may seem odd to foreign languages, and vice versa
  - Ex. “trickle’: something (a liquid) falling

- Finite State Automaton (Automata)
  - Chomsky’s first theory







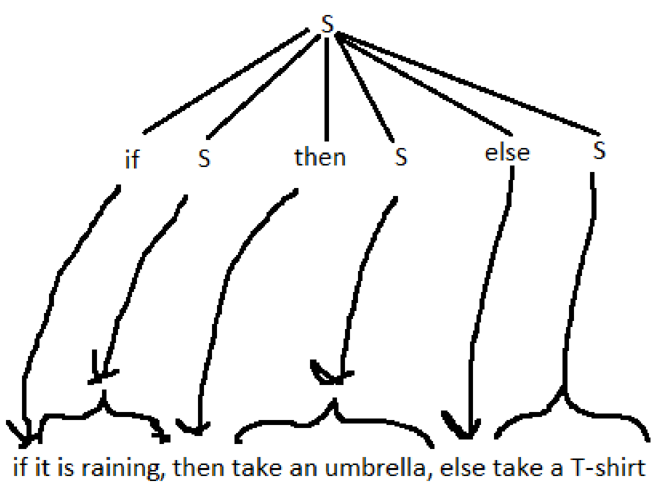

  - Consider:
    - S- if $S_1$ then $S_2$ otherwise $S_3$
    - If it is raining, then I’ll take an umbrella, otherwise I won’t
    - If it is raining, then if it is cold, then take a ski parka, otherwise I’ll take a windbreaker, otherwise I’ll take a T-shirt

- Every sentence can have its own infinite sub-routines
- If $S_1$ then if $S_2$ then if $S_3$…
  - We must have the same amount of "otherwise"'s
  - Finite State Automata cannot count, nor can they allow A and B to vary according to one another (as shown in example 3 later in the notes)
    - They will have a limit on the amount of repeats/loops it can recognize/generate
  - Therefore, a Finite State Automaton cannot be enough

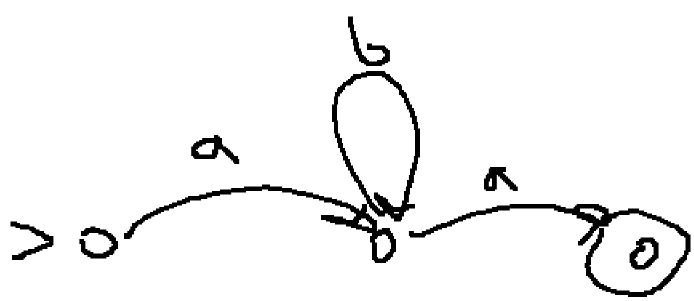
- Chomsky came up with context-free grammar
  - S → NP VP
  - NP → N | det. N | det Adj.list N ("|" means or)
    - Adj.list → Adj.|Adj. Adj.list
  - S→ if exp. then S
  - Exp. → Var relop var
    - Relop (relational operator) → = = | <= | >= | < | >
- An expression will return true or false

- If x>5, then if y>3, then z=4, else z=5
  - Ambiguous because it doesn't base definition off a specific part of the statement
    - This is why programming languages have "endif"

- The point is:
  - We can take care of ambiguity by using context-free grammar
    - A context-free grammar is powerful enough to handle arbitrary if-then statements
    - Just see if a specific sentence matches the rules
- Ex.

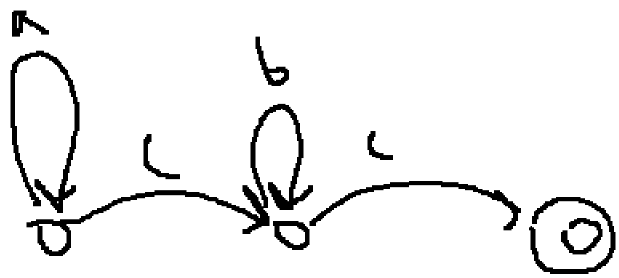



- Ex. 1 {aa, aba, abba, abbba, abbbba,…} → ab*a



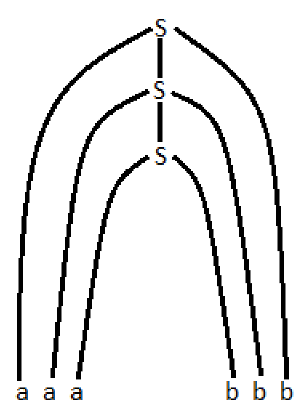

- S→ aa | a blist a
  - blist → b| b blist

- Ex. 2) a*cb*c



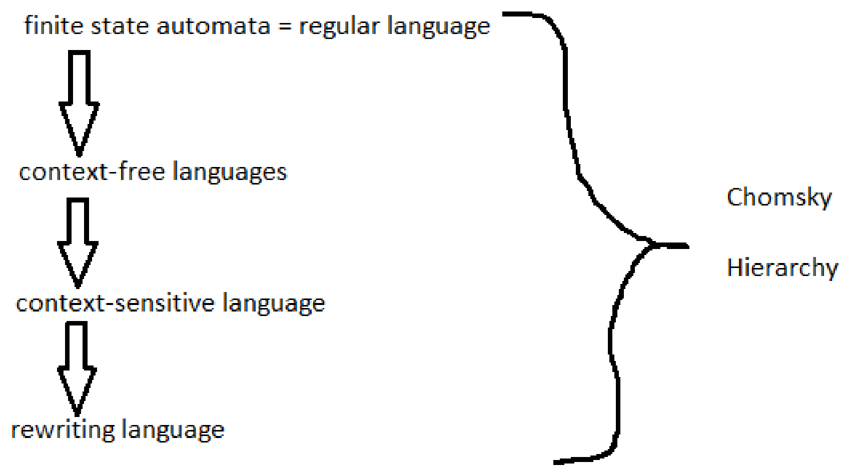

    - S→ cc | alist c blist c
        - alist → ∈ | b blist
        - blist → b | b blist
            - ∈ means null

- Ex. 3
    - {$a^n$ $b^n$ | n is a pos. integer}




- S→ ab|a S b

- Theory 3: Chomsky Hierarchy
    - Mary saw John. ⇔ John was seen by Mary.




- Computability: what one can compute with a Turing Machine

Bonus:

- The Secret to Naps

1. Don't make yourself too comfortable
2. Realize that the world can live without you for 20 or so minutes
3. Thinking about something intellectually challenging will help you fall asleep