Context-Free languages

Atri Rudra

May 1

Announcement

- Pick up H/W #3
  - If you did not last class

Feedback: things I will change

- Ask questions to gauge understanding
  - More concrete/problem solving type
- Instructor stands in front of what he writes

On puzzles

- Solutions the week after ?
- My take
  - I need an indication that you have solved the problem
    - Tell the instructor/TAs about your solution
    - Email your solution
- Recommended problems
  - Turn in your puzzles, I'll "grade" them

On Examples

- Many a times too simple
- Not similar enough to the H/W problems
  - Students are not taught how to solve the type of problems on H/Ws

What is expected of a formal proof ?

- Any proof in Sipser is a formal proof
  - vs. Proof idea
  - Beware: some proofs are not "complete"
- A proof is a *convincing logical argument that a statement is true*
- "Format" of proofs
  - You are not constrained
  - More in an email

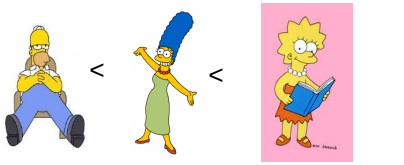
In short: ask questions/clarifications

A quote from the feedback

- "…..
  Programming > Theory"

Questions ?

Simpson family model of computation

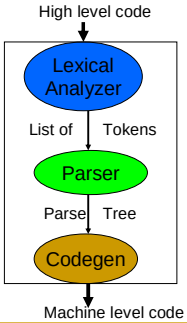


Finite State Machine < Context Free Grammars < Turing Machines

- We will prove a bunch of stuff about these models

Another application: Compiler

- *javac* for Java
- First step
  - Lexical Analyzer
  - Uses Finite state machines
- Second step
  - Parser
  - Error messages
  - Context Free grammar
- Last step: Codegen



Let's look back

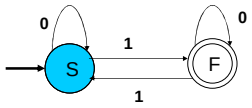
- Regular languages
  - DFA
  - NFA
  - Regular expression

As if three forms were not enough…

- We will quickly go over another!

Good old DFA

- DFA for string with odd number of 1s
- On input: 00111



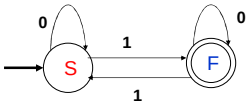
"Acceptors" vs "Generators"

- DFA / NFA
  - Given a string, tell whether it is accepted or not
- Regular expression
  - "Generates" strings in the language
- Up next: another "generator"

Welcome to the world of grammars

- Rules for generating strings in the language
  - S → 0S
  - S → 1F
  - F→ 0F
  - F→ 1S
  - F→ ε

Each line is a rule



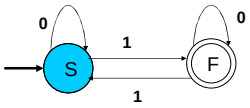
OK… how does it work ?

Generating strings from the grammar

- Start with S
- Any any step apply a rule
  - Replace S/F with the right hand side
- Stop when no S/F remain
  - S → 0S
  - S → 1F
  - F→ 0F
  - F→ 1S
  - F→ ε

S
0S
00S
001F
0011S
00111F
00111

What's going on here ?



S
0S
00S
001F
0011S
00111F
00111

Questions ?

- Don't worry if you could not copy down the slides
- We will go through this again on the whiteboard