Exam 1 Review (2019)

Yongjun Park
Hanyang University

Logistics

- **When, Where:**
  - Wednesday, Oct 30, 1:00pm –2:30pm
  - Room: ITBT 508
- **Type:**
  - Open book/note
- **What to bring:**
  - Text book, reference books, lecture notes
    - But must print these out
  - Pencils
  - No laptops or cell phones

Topics Covered

- **Lexical analysis: ~50%**
- **Syntax analysis: ~50%**
  - Percentages are VERY approximate
- **Not covered**
  - Detailed flex/bison syntax
    - However, contents on the slides will be covered
    - And general questions about flex/bison are possible

Textbook

- **What have we covered: Ch 1 – 4**
- **Things you should know / can ignore**
  - Ch 1, 2 – Just overviews, don't worry about it
  - Ch 3 – Lexical analysis
  - Ch 4 – Syntax analysis
- **+ Project 1/2 related stuffs on Louden book**

Open Book Exams

- **OPEN BOOK != DON'T STUDY**
- **Open book means**
  - Memorizing things is not that important
  - If you forget something, you can look it up
- **But**
  - If you are trying to learn (or re-learn) stuff during the test, you won't finish
  - Assume the test is not open book
    - Don't rely on book/notes
    - Treat them as a backup

How to Study

- **Re-familiarize yourself with all the material**
  - Where to find things if you get confused
- **Practice solving problems**
  - Do them w/o looking at the answer!
  - Class problems/examples from lectures
  - Examples in book
    - If you are ambitious, exercises at the end of each chapter
  - Practice so that you can do them without thinking much

Exam Format

- **Short answer**
  - Explain something
  - Short problems to work out
- **Longer design problems**
  - E.g., construct a parse table
- **Range of questions**
  - Simple – Were you conscience in class?
  - Grind it out – Can you solve problems
  - Challenging – How well do you really understand things
- **Important! - move on if you get stuck!**

Lexical Analysis (aka Scanning)

- **Ch 2-3**
- **Regular expressions**
  - How to write an RE from a textual description
- **NFAs**
  - RE to NFA (Thompson construction)
- **DFAs**
  - NFA to DFA
- **State minimization**
- **How does lex/flex work**

Example Problem: Regular Expressions

- **Write a RE for strings over the alphabet {a,b,c} where the first ‘a’ precedes the first ‘b’**

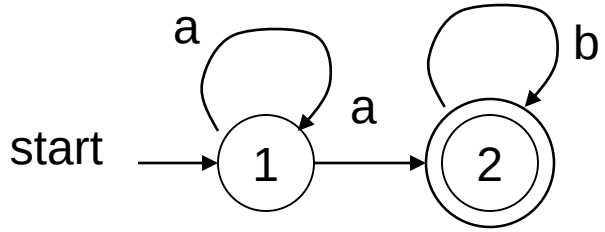
Example Problem: NFA

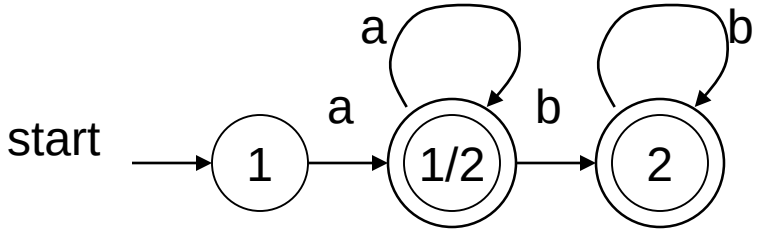
- **What's an ε transition and why do we allow them?**
- **Create an NFA for the following RE:**
  - a ((b | c | a+ c) x)* | x+ a

Recall: Convert the NFA to a DFA

- **Problem 1: Multiple transitions**
  - Move (S,a) is relabeled to target a new state whenever single input goes to multiple states

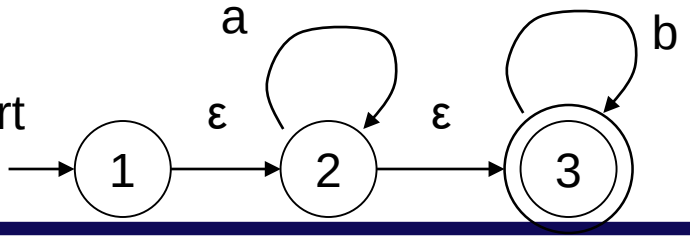
a+b*

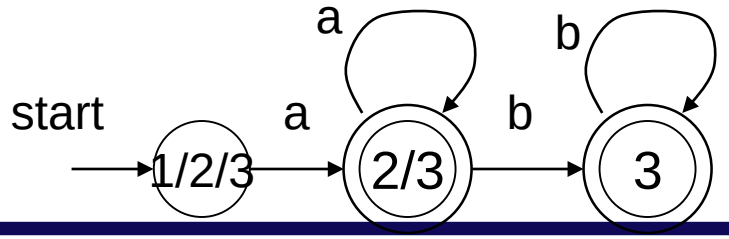




- **Problem 2: ε transitions**
  - Any state reachable by an ε transition is "part of the state"
  - ε-closure - Any state reachable from S by ε transitions is in the ε-closure; treat ε-closure as 1 big state

a*b*





Example Problem: DFA

- **Convert the previous NFA into a DFA**

Syntax Analysis (aka Parsing)

- **Ch. 4**
- **Context free grammars**
  - Derivations, ambiguity, associativity
- **Parsing**
  - Top-down
    - LL(1), building parse tables (FIRST, FOLLOW)
  - Bottom-up
    - LR(0), LR(1), SLR, LALR
    - Building parse tables (Closure, Goto), shift/reduce
- **Abstract syntax tree**

Example Problem: CFG

- **Is the following grammar ambiguous? Explain why or why not.**
  - S → E + E
  - E → num

Recall: Computing FIRST/FOLLOW

- **Determining FIRST(X)**
  1. if X is a terminal, then add X to FIRST(X)
  2. if X→ ε then add ε to FIRST(X)
  3. if X is a nonterminal and X → Y1Y2...Yk then a is in FIRST(X) if a is in FIRST(Yi) and ε is in FIRST(Yj) for j = 1...i-1
  4. if ε is in FIRST(Y1Y2...Yk) then ε is in FIRST(X)
- **Determining FOLLOW(X)**
  1. if S is the start symbol then $ is in FOLLOW(S)
  2. if A → αBβ then add all FIRST(β) != ε to FOLLOW(B)
  3. if A → αB or αBβ and ε is in FIRST(β) then add FOLLOW(A) to FOLLOW(B)

Example Problem: FIRST/FOLLOW

- **Calculate FIRST and FOLLOW for:**
  - S → uBDz
  - B → Bv | w
  - D → EF
  - E → y | ε
  - F → x | ε

Example Problem: LL(1) Parsing Table

- Construct the LL(1) parsing table for the previous grammar
- Can the grammar be parsed by an LL(1) parser?

Recall: Closure and Goto

- **Closure of a parser state:**
  - Start with Closure(S) = S
  - Then for each item in S:
    - X → α . Y β
    - Add items for all the productions Y → γ to the closure of S: Y → . γ
- **Goto operation = describes transitions between parser states, which are sets of items**
  - If the item [X → α . Y β] is in I, then
  - Goto(I, Y) = Closure([X → α Y . β])

Example Problem: LR(0) DFA

- **Build the LR(0) DFA for:**
  - S → E $
  - E → id
  - E → id (E)
  - E → E + id