CS 536 Announcements for Monday, February 26, 2024

Last Time

- Java CUP
  - specification format
  - handling precedence and associativity
  - translating lists
  - handling unary minus

Today

- review for Midterm 1

Next Time

- approaches to parsing
- bottom-up parsing
- CFG transformations

Midterm 1
Thursday, February 29, 7:30 – 9 pm
S429 Chemistry

Covers

- lectures through 2/19
- programming assignments 1 & 2

Additional practice

- Homeworks 0, 1, & 2
- sample midterm (esp questions 1, 3a, 3b, 4)

Format

- closed-book, closed-notes
- paper and pencil/pen
- question formats
  - multiple-choice
  - short-answer
  - written questions

Make sure to bring your student ID

See also Exam Information page

Midterm 1 Topics

Scanning

- general :
  - what does a scanner do
  - how does it fit into the design of a compiler
- underlying model :
  - FSMs, DFAs vs NFAs
  - translating regex → NFA
  - translating NFA → DFA
- specification of a scanner :
  - regular expressions, JLex specifications*
    *you do not need to know all of JLex's special characters

Context-Free Grammars

- specification of a language's syntax via a CFG
- derivations (left-most, right-most)
- parse trees
- expression grammars (precedence, associativity)
- list grammars
- ambiguous grammars
- recursive grammar (left recursive, right recursive)

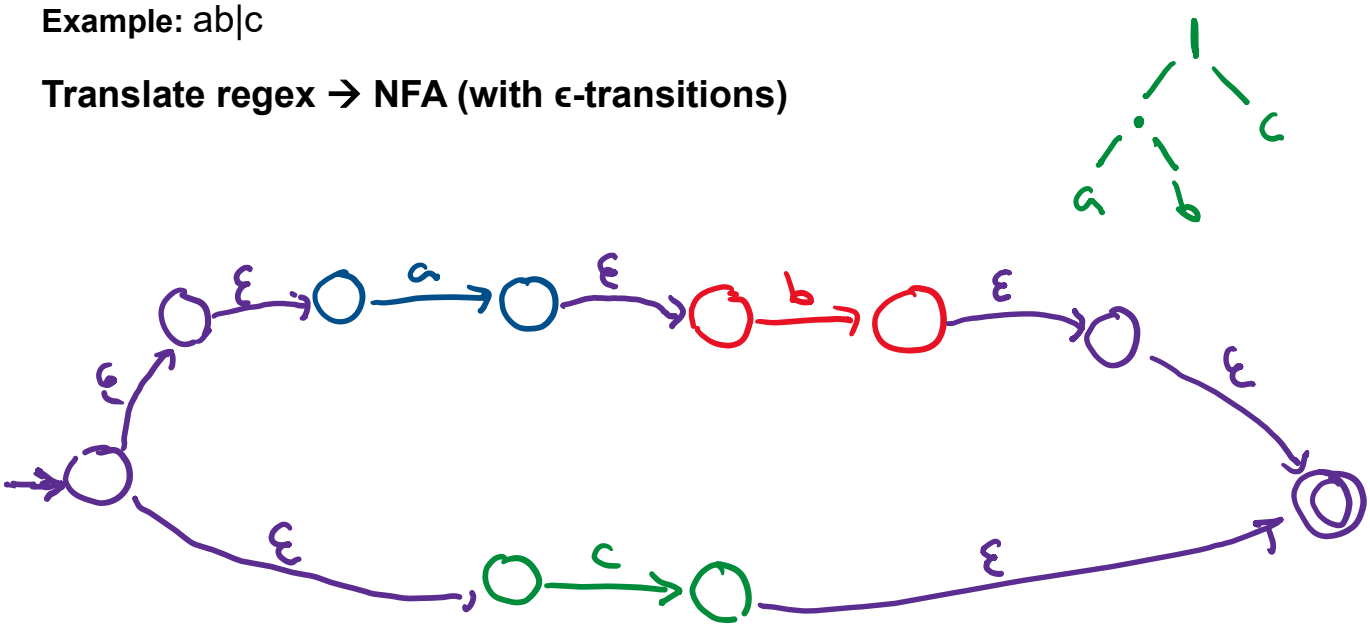
Syntax-Directed Translation

- "plain" translations
  - writing rules of the form "s1.trans ="
- being able to define translations of any types (integer, AST nodes, etc.)

Translating a regular expression into a DFA

Example: ab|c

Translate regex → NFA (with ϵ-transitions)



Removing ϵ-transitions from NFAs

Let M be an NFA with ϵ-transitions. Goal: construct ϵ-free NFA M* that is equivalent to M

Recall: eclose(s) = set of all states reachable from s using 0 or more ϵ - transitions

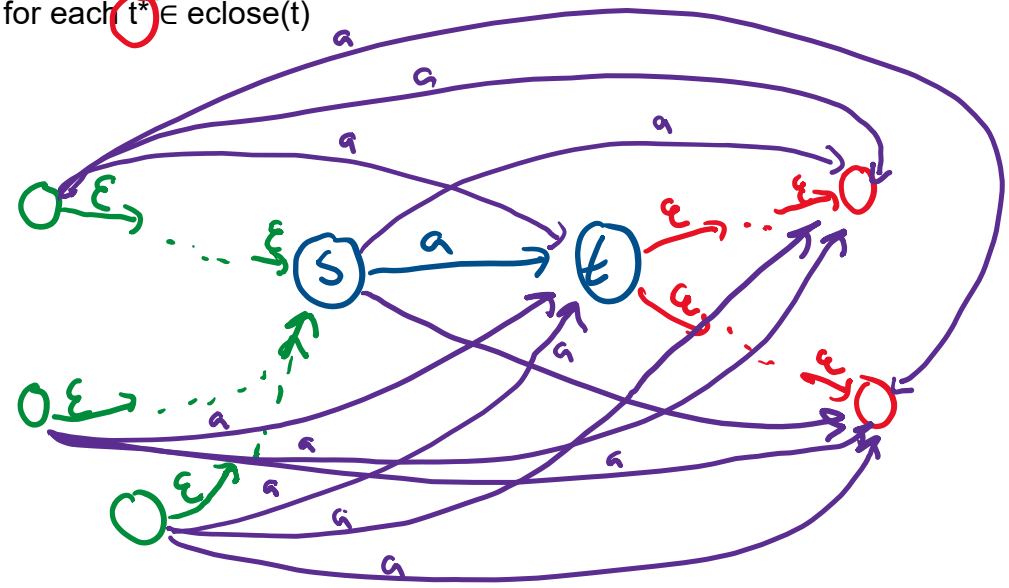
Overview:

1) determine eclose(s) for each state s ∈ M
2) initialize M* to M
3) determine additional final states of M* – using eclose
4) add edges to M*
5) remove ϵ-transitions

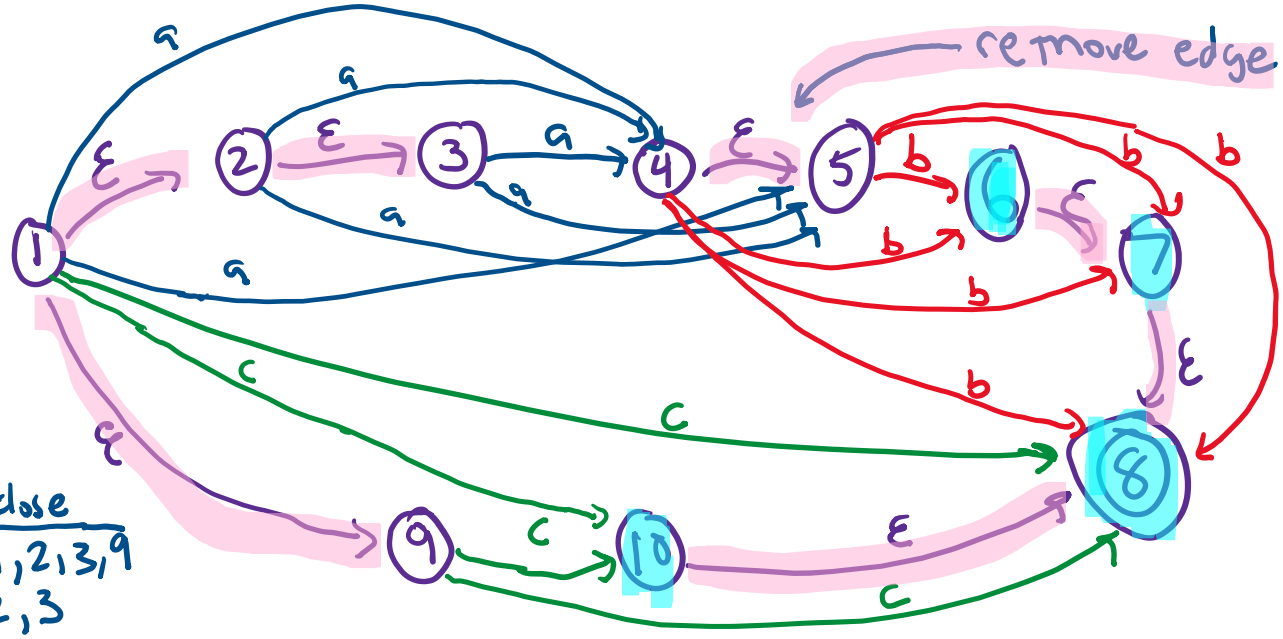
Final states of M*: If eclose(s) contains a final state in M, then s is a final state in M*

Adding edges to M*: If there is edge s –^{a}→ t in M (a ≠ ϵ), then add edge s* –^{a}→ t* in M*

- for each s* such that s ∈ eclose(s*) and
- for each t* ∈ eclose(t)



Translate NFA w/ ϵ-transitions → NFA w/o ϵ-transitions



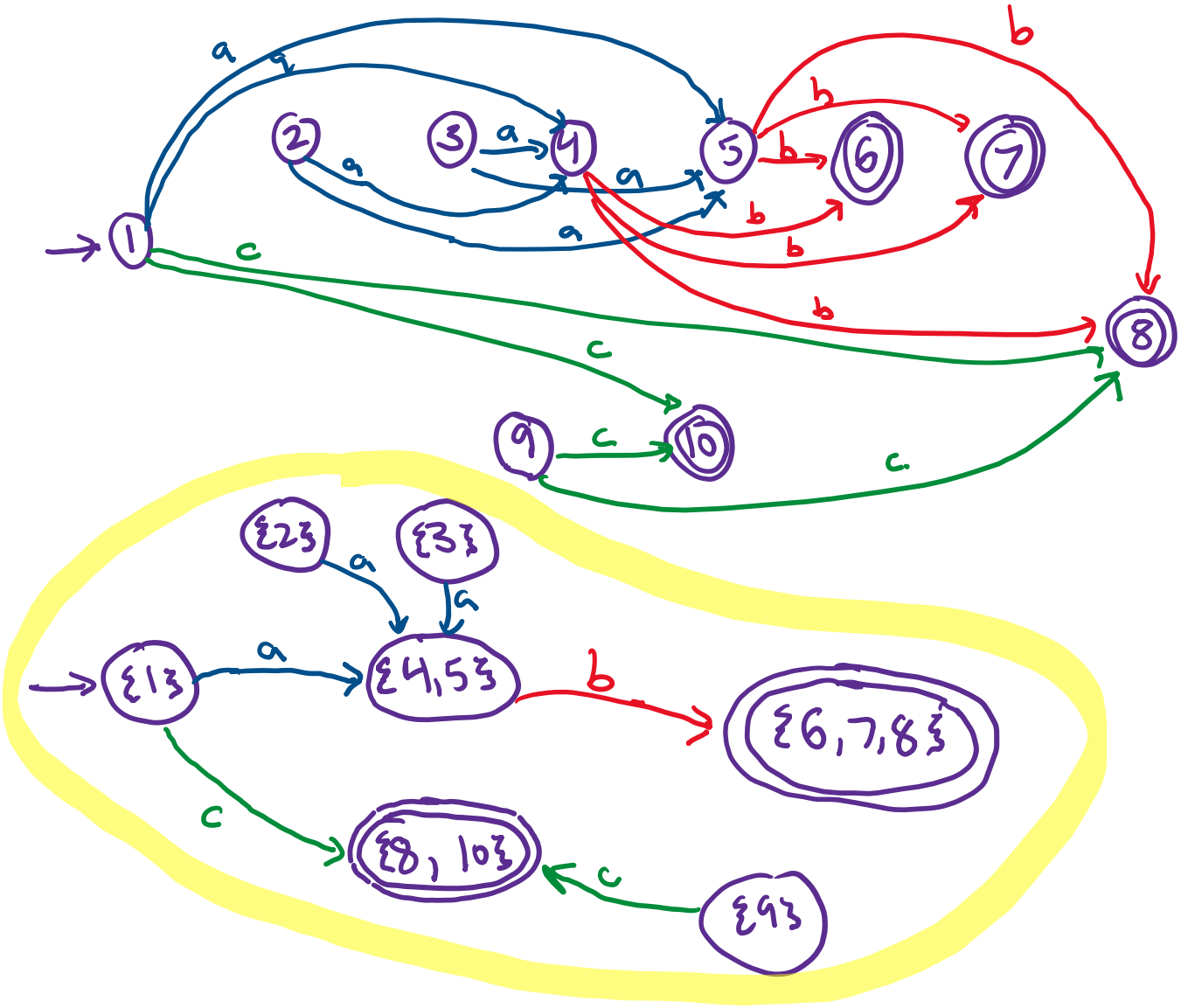
	eclose
1	1,2,3,9
2	2,3
3	3
4	4,5
5	5
6	6,7,8
7	7,8
8	8
9	9
10	10,8

Consider (3) $\xrightarrow{a}$ (4) $3 \in eclose(x)$ for $x \in \{1,2,3\}$
$eclose(4) = \{4,5\}$

Consider (5) $\xrightarrow{b}$ (6) $5 \in eclose(x)$ for $x \in \{4,5\}$
$eclose(6) = \{6,7,8\}$

Consider (9) $\xrightarrow{c}$ (10) $9 \in eclose(x)$ for $x \in \{1,9\}$
$eclose(10) = \{10,8\}$

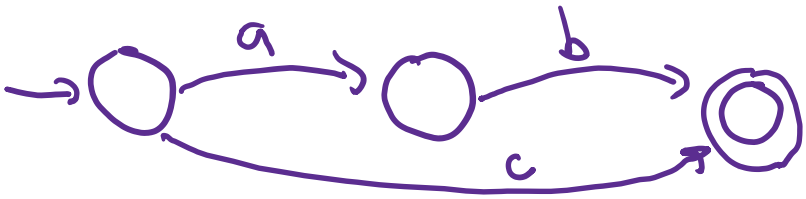
NFA w/o ϵ-transitions → DFA



Optimize DFA

{2}, {3}, {9} are unreachable → remove them

{6, 7, 8}, {8, 10} are equivalent → merge them



≡ a|b|c

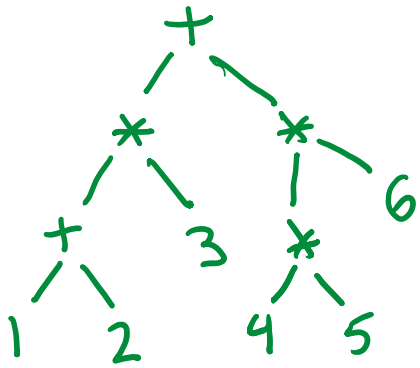
Syntax-directed translation

CFG			Translation rules
$expr_1$	→	$expr_2$ + term	$expr_1.trans = expr_2.trans + term.trans$
	|	term	$expr.trans = term.trans$
$term_1$	→	$term_2$ * factor	$term_1.trans = term_2.trans * factor.trans$
	|	factor	$term.trans = factor.trans$
factor	→	INTLIT	$factor.trans = INTLIT.value$
	|	(expr)	$factor.trans = expr.trans$

Write a syntax-directed translation for the CFG given above so that the translation of a sequence of tokens is numeric value of the expression (i.e., the expression evaluated).

(1+2)*3 + 4*5*6

AST



Parse tree