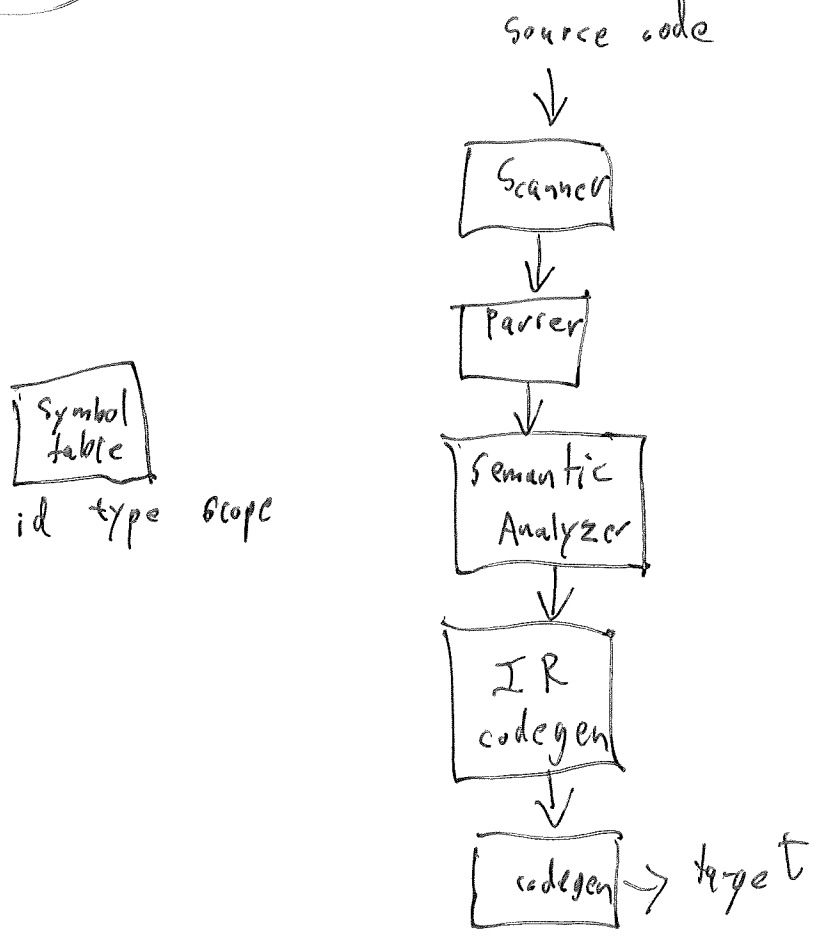
Last time


Source code
Scanner
Parser
Semantic Analyzer
IR codegen
codegen
target
Symbol table
id type scope

Outline

A bit of history
- Why this layout

FSM. (Finite state Machines)
- Scanner

Compiler History

At minimum... Compiler needs to:

- handle complex input

$(x++)+(y++)$

- Be relatively efficient

```
int fn(int);
```

Abstractions handle complexity

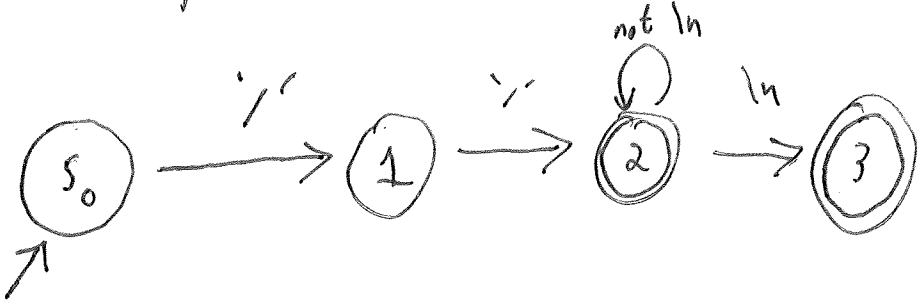
FSMs: Finite State Machines

In purest form, recognizes sequences of characters that form a token

- Powerful enough for regular languages
- "Good enough" for tokenization

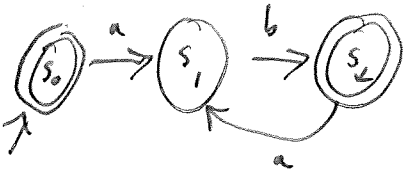
FSM : Example

// single line comments



"//"

FSM : Another Example



What Language does this accept?

"ab"

"ab ab"

"ab ab ab"

Let's call this L (C-minor)

FSM, Formally

$$(Q, \Sigma, \delta, q, F)$$

Q is a finite set of states

Σ is the alphabet, a finite set of characters

$q \in Q$ is the start state

$F \subseteq Q$ is the set of final states

$\delta : Q \times \Sigma \to Q$ is the transition relation

FSM accepts $x = x_1 x_2 x_3 \ldots x_n \iff$

$$\delta(\delta(\ldots\delta(\delta(\delta(s_0, x_1), x_2), x_3)\ldots, x_{n-1}), x_n \in F$$

Formal Defn L(C-minor)

$Q = \{ s_0, s_1 \}$

$\Sigma = \{ \text{'a'}, \text{'b'}, \text{'c'} \}$

$q = s_0$

$F = \{ s_0 \}$

$$\delta = \begin{cases} s_0, \text{'a'} \rightarrow s_1 \\ s_1, \text{'b'} \rightarrow s_0 \end{cases}$$

anything else $\rightarrow$ stuck / Dead

transition table

	a	b	c
s_0	s_1		
s_1		s_0	

FSM types: DFA & NFA

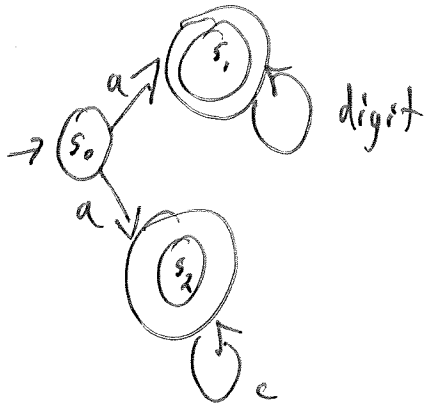
Deterministic

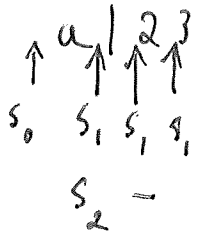
- No state has >1 outgoing edge with the same label

Nondeterministic

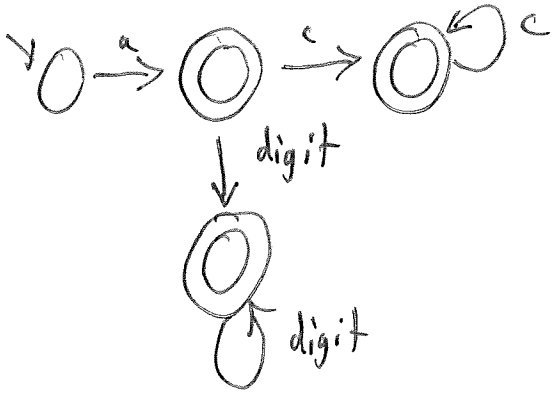
- States may have multiple outgoing edges with the same label
- Edges may be labelled with the special symbol ε, the empty string. ε-transitions can happen without looking at input

NFA example



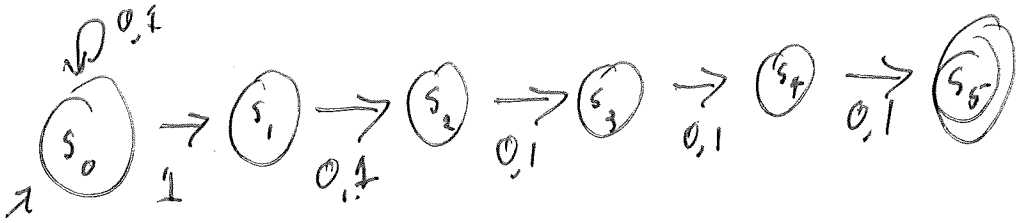


Equivalent DFA



Why NFAs?

Much more compact



An equivalent DFA ...

2^5 states