Compilers Lab III

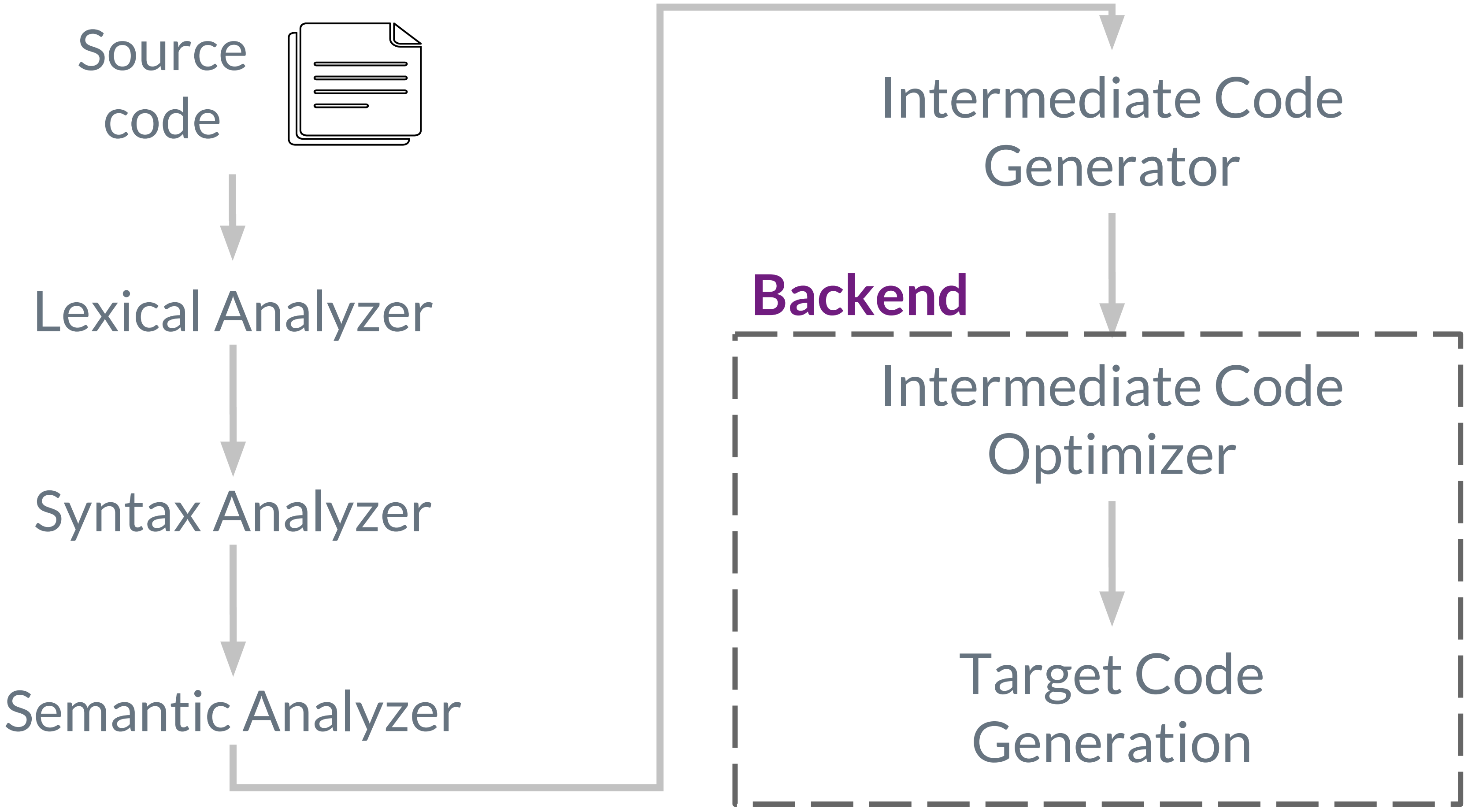
Mo.Agamia

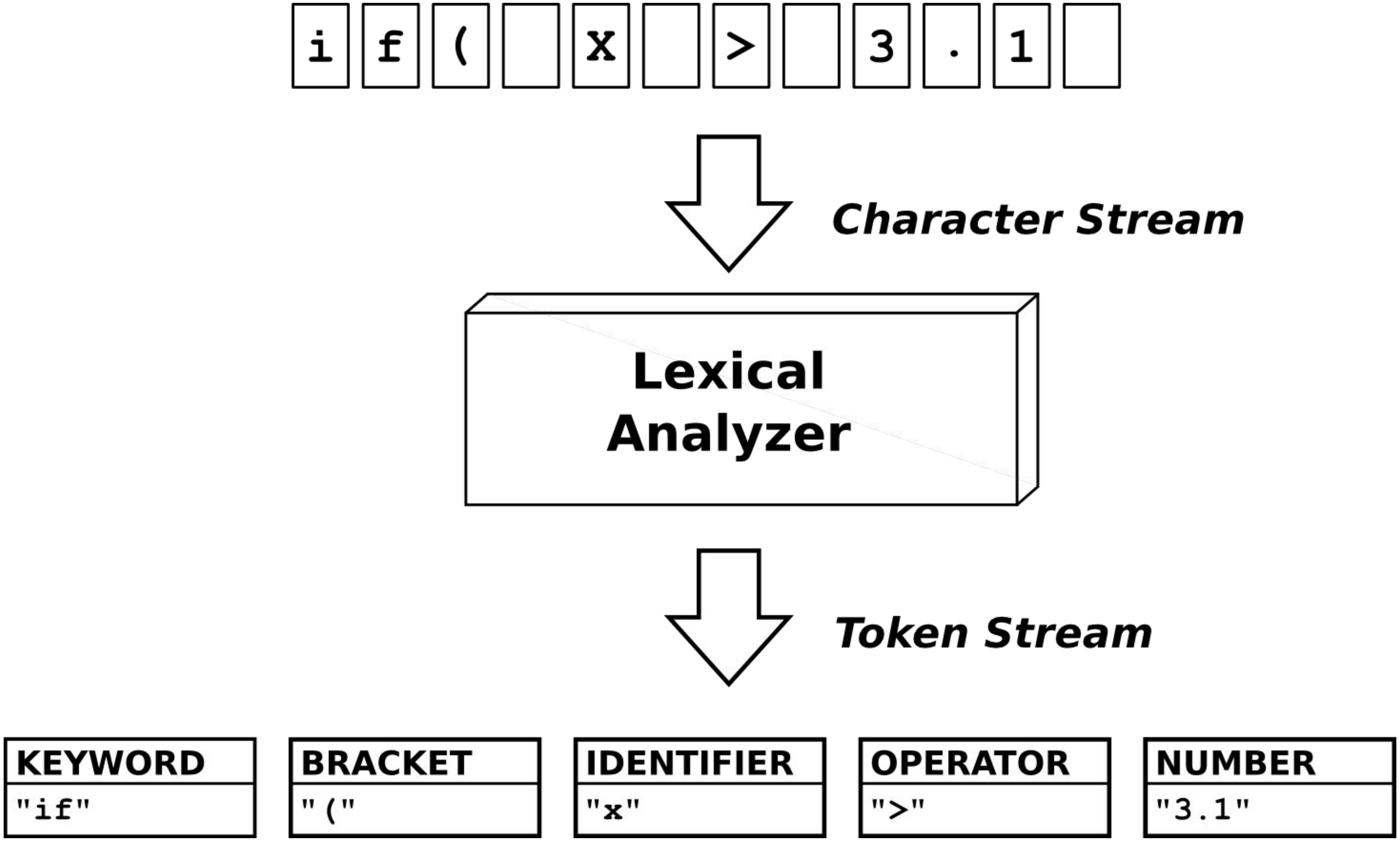
Plan

- ▷ Overview
- ▷ Fallback DFA

1.
Overview

Compiler phases



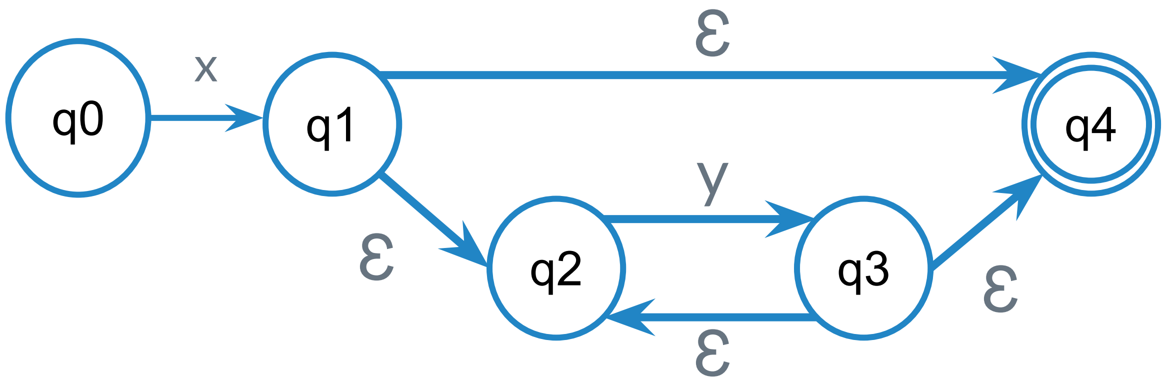


i
f
(
x
>
3
.
1
Character Stream
Lexical Analyzer
Token Stream
KEYWORD
"if"
BRACKET
"("
IDENTIFIER
"x"
OPERATOR
">"
NUMBER
"3.1"

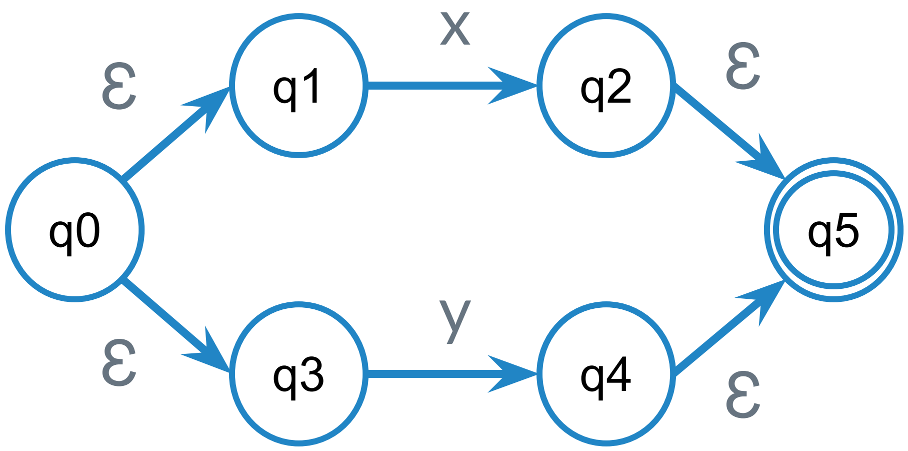
1. Write regular definition
2. Compile corresponding regular expression
3. Convert expression to NFA
4. Convert NFA to DFA

Regex to NFA

- xy* →
  print "hi"

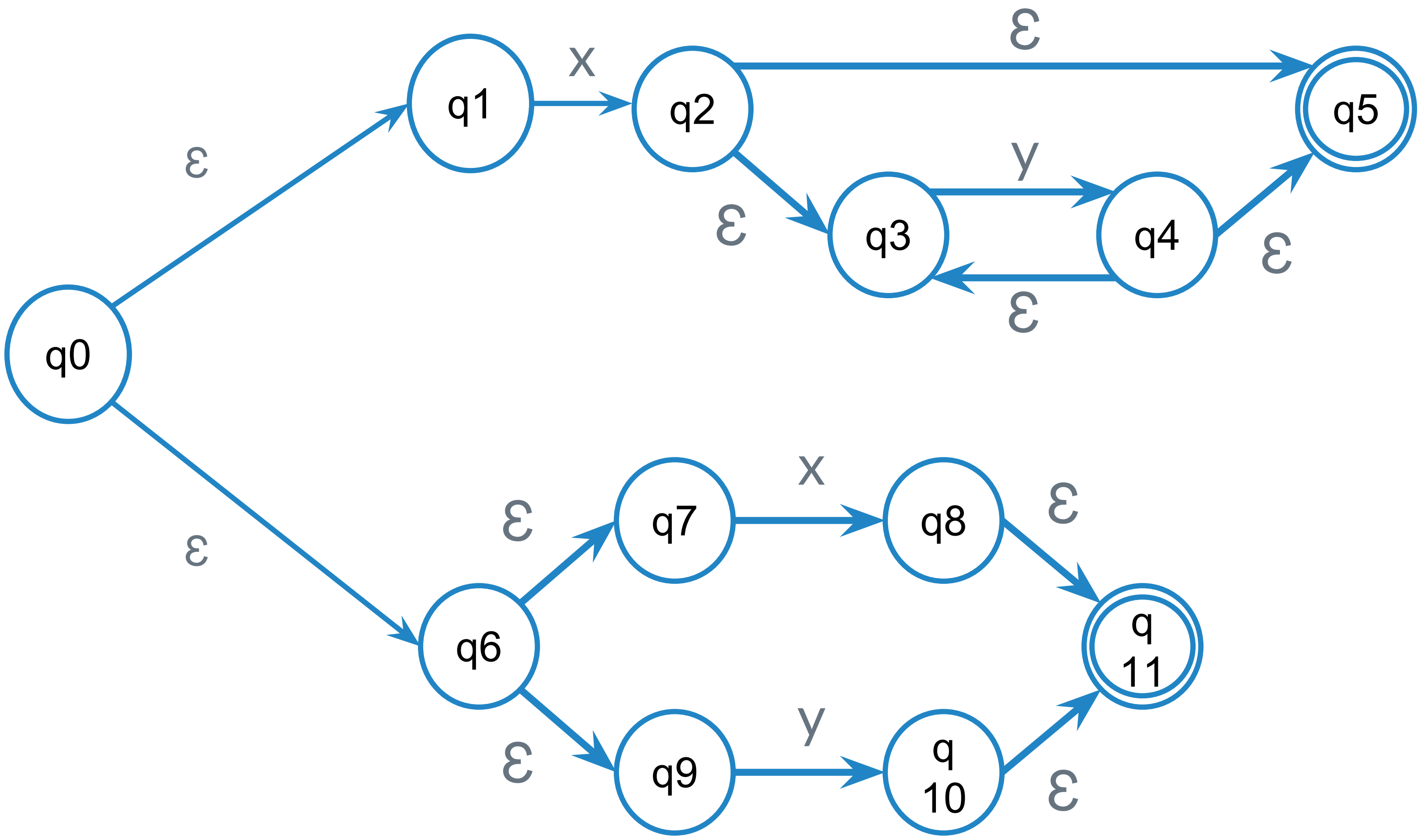


- x|y →
  print "bye"



Mo.Agamia

NFA



NFA to DFA to Fallback DFA

(xy*) | (x|y)

DFA state	NFA state	A/R	x	y	action
A	{q0, q1, q6, q7, q9}	rejected	B	C	default
B	{q2, q3 ,q5 ,q8, q11}	accepted	Dead	D	x|y
C	{q10, q11}	accepted	Dead	Dead	x|y
D	{q3, q4, q5}	accepted	Dead	D	xy*
Dead	-	rejected	Dead	Dead	default

Mo.Agamia

2.
Fallback DFA

Mo.Agamia

A fallback DFA with actions, is a 6-tuple ⟨Q, Σ, δ, q0, F, A⟩

Q, Σ, δ, q0, and F are as usual

A maps every q ∈ Q into an action.

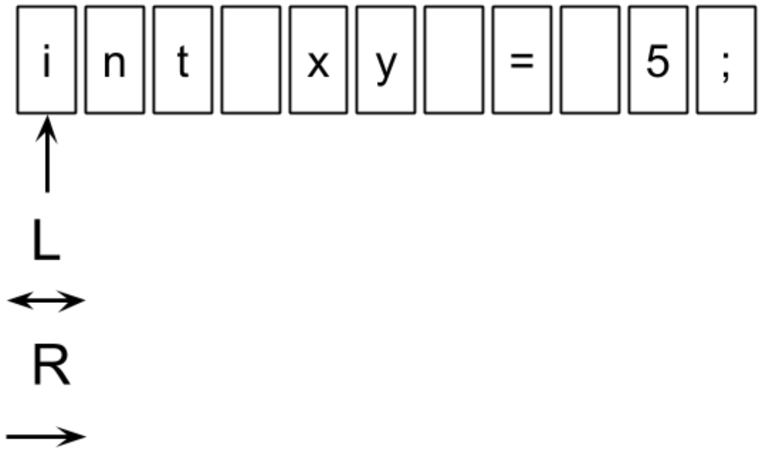
Fallback DFA

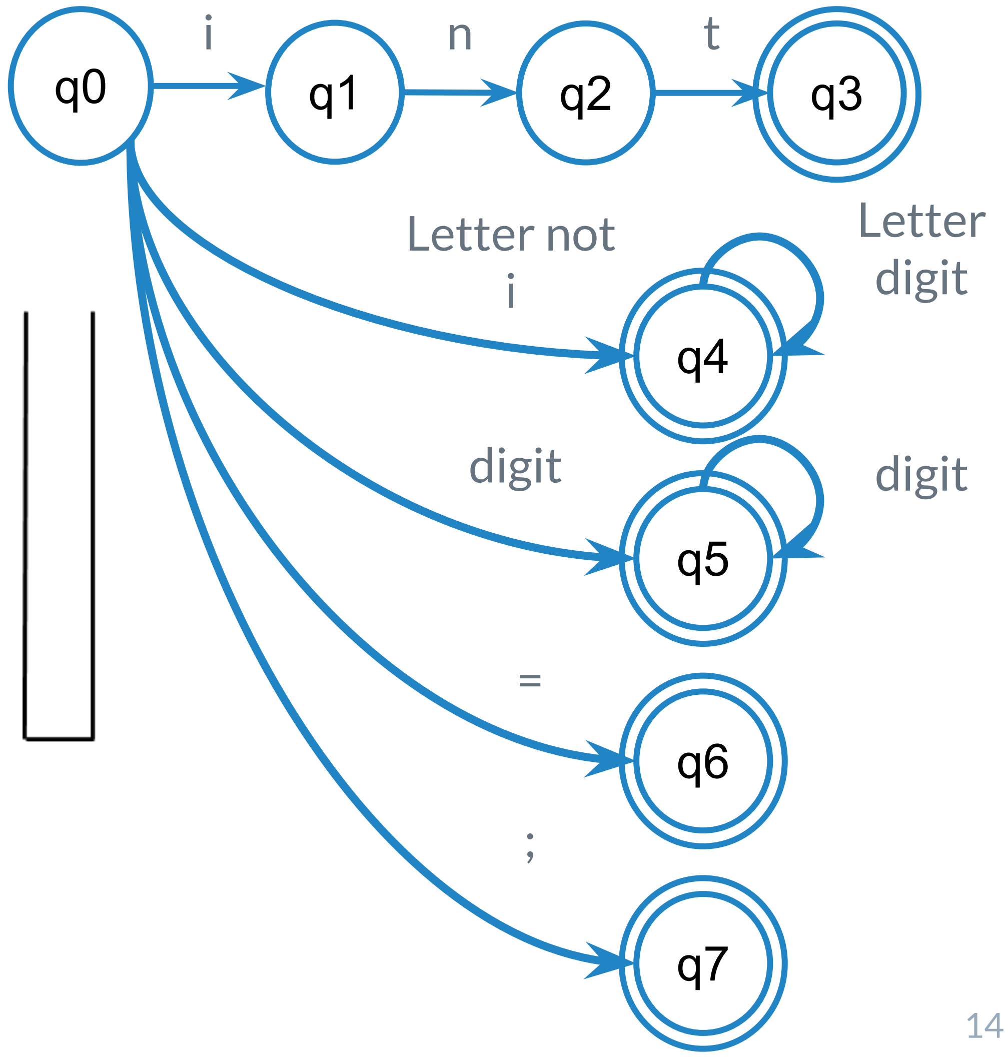
- A fallback DFA with actions consists of a stack, and two heads: L and R.
- Initially, both heads point at the left-most, where the input starts.
- R can move only to the right.
- L can move to the right and to the left.

Fallback DFA

- Push every state in the stack with every transition until the end.
- If it runs out of input at a final state, execute the action and stop.
- If it is not a final state, then:
  - Pop & move L one step to the left until the a final state or the stack is empty.
  - Stack is empty, execute the action and stop.
  - If a final state was popped, then:
    - Execute action.
    - Move L one step to the right & move R to L.
    - Empty the stack, then start over.

Fallback DFA





Mo.Agamia

Thanks!

Any questions?

You can find me at:

@piazza

mohammed.agamia@guc.edu.eg