

# CMSC 330: Organization of Programming Languages

## Pushdown Automata and Turing Machines

### This Lecture

- ▶ Regular grammars
- ▶ Pushdown automata
- ▶ Turing machines
- ▶ Decidability
- ▶ Turing tests

### Last Lecture

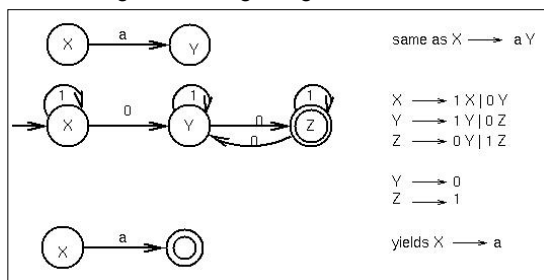
- ▶ Parsing
  - Recursive descent
  - FIRST sets
- ▶ Rewriting grammars
  - Left factoring
  - Eliminating left recursion
- ▶ Question
  - Where do regular expressions and context free grammars stand in the world of complexity?

### Regular Expressions and CFGs

- ▶ Regular grammar
  - Subset of context free grammar
  - All productions are of the form
    - >  $A \rightarrow aB$  // 1 terminal followed by 1 nonterminal
    - >  $A \rightarrow a$  // 1 terminal
- ▶ Regular grammars are just as powerful as regular expressions
  - For any regular expression R, there is a regular grammar G that accepts the exact same language

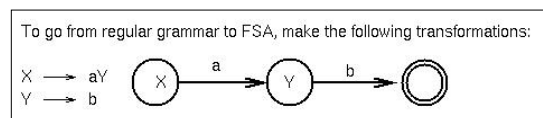
### Equivalence of DFA and Regular Grammar

- ▶ Reducing DFA to regular grammar



### Equivalence of DFA and Regular Grammar

- ▶ Reducing regular grammar to DFA



## Regular Expressions and CFGs

- CFGs are strictly more powerful than REs
  - Since CFGs are superset of regular grammar
- This is good, since programming languages are not regular
  - No regular expression for  $(n)^n$
- But programming languages are usually (almost) context-free
  - With a few language constructs being exceptions

CMSC 330

7

## Regular Expressions and CFGs

	Description	Machine
regular languages	regular expressions	DFA, NFAs
context-free languages	context-free grammars	pushdown automata (PDAs)

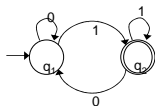
- We can implement REs with DFAs
- Question
  - How can we implement CFGs?

CMSC 330

8

## Pushdown Automaton (PDA)

- A PDA is an abstract machine similar to a DFA
  - Has a finite set of states
  - But also has a stack



CMSC 330

9

## Moves of a PDA

- Reads
  - An input symbol is read
  - Top symbol on the stack
- Based on both inputs, the machine
  - Enters a new state, and
  - Writes zero or more symbols onto the pushdown stack
  - Or pops zero or more symbols from the stack
- String accepted if
  - The stack is empty, AND
  - The string has ended

CMSC 330

10

## Power of PDAs

- PDAs are more powerful than DFAs
  - $a^n b^n$ , which cannot be recognized by a DFA, can easily be recognized by the PDA
    - Stack all  $a$  symbols and, for each  $b$ , pop an  $a$  off the stack.
    - If the end of input is reached at the same time that the stack becomes empty, the string is accepted
- PDAs can recognize some L(CFG)
  - Table-driven predictive parser
    - Table = Finite automaton
    - Stack = Stack
  - Limited to LR(1) grammars in general

CMSC 330

11

## Power of PDAs and NDPDA

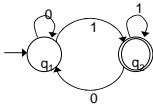
- There are also nondeterministic PDA (NDPDA)
  - NDPDA are more powerful than DPDA
  - NDPDA can recognize even length palindromes over  $\{0,1\}^*$ , but a DPDA cannot. Why?
    - Hint: Consider palindromes over  $\{0,1\}^2\{0,1\}^*$  vs  $\{0,1\}^* \{0,1\}^*$
- NDPDAs can recognize all L(CFG)
  - Equivalent in power to context-free languages

CMSC 330

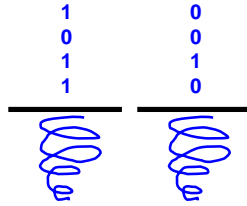
12

## Build a More Powerful PDA?

- ▶ What if we add another stack to the PDA?
- ▶ A 2-stack PDA
  - Has a finite set of states
  - Plus 2 stacks
- ▶ How powerful is this?



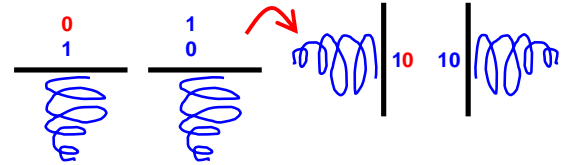
CMSC 330



13

## Build a More Powerful PDA?

- ▶ Answer – very powerful!
- ▶ Trick
  - Flip two stacks so they face each other
  - Pop a symbol off stack 1, push it right back onto stack 2
  - 2 stacks is just like a tape!

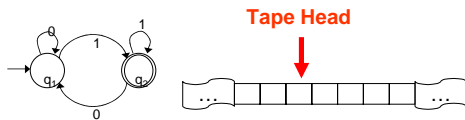


CMSC 330

14

## Turing Machine

- ▶ Defined by Alan Turing in 1936
- ▶ Finite state machine + tape
- ▶ Tape
  - Infinite storage
  - Read / write one symbol at tape head
  - Move tape head one space left / right



CMSC 330

15

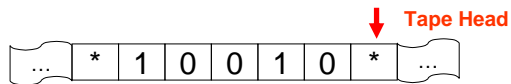
## Turing Machine

- ▶ Allowable actions
  - Read symbol from current square
  - Write symbol to current square
  - Move tape head left
  - Move tape head right
  - Go to next state

CMSC 330

16

## Turing Machine



Current State	Current Content	Value to Write	Direction to Move	New state to enter
START	*	*	Left	MOVING
MOVING	1	0	Left	MOVING
MOVING	0	1	Left	MOVING
MOVING	*	*	No move	HALT

CMSC 330

17

## Turing Machine

- ▶ Operations
  - Read symbol on current square
  - Select action based on symbol & current state
  - Accept string if in accept state
  - Reject string if halts in non-accepting state
  - Reject string if computation does not terminate
- ▶ Halting problem
  - It is undecidable in general whether long-running computations will eventually accept

CMSC 330

18

## “Enhanced” Turing Machines

- Attempts to increase power of Turing machine
  - Additional tapes
  - Additional tape heads
  - 2-dimensional tape
  - BlueGene/L @ Livermore Lab
    - World's fastest computer in 2007
    - 106K IBM PowerPC nodes (596 x10<sup>12</sup> Flops)
- Turns out any computation on enhanced machine
  - Can be computed on standard Turing machine
  - Though may require more time



CMSC 330

19

## Computability

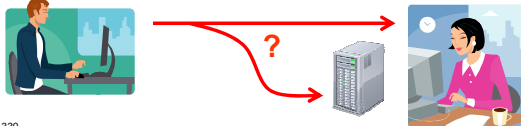
- Computability
  - A language is **computable** if it can be recognized by some algorithm with finite number of steps
- Church-Turing thesis
  - Turing machine can recognize any language computable on any machine
- Intuition
  - Turing machine captures essence of computing
    - Both in a formal sense, and in an informal practical sense

CMSC 330

20

## Computability – Turing Test

- Turing Test
  - Test machine's capability to demonstrate intelligence
  - Proposed by Alan Turing in 1950
  - Practical test for “Can machines think?”
- Test procedure
  - Judge converses via text (e.g., instant messaging)
  - Pass if can't reliably tell whether machine or human



CMSC 330

## More Turing Tests



22

## Computability – Reverse Turing Test

- Reverse Turing test
  - Computer determines whether human or machine
- CAPTCHA
  - Completely Automated Public Turing test to tell Computers and Humans Apart
  - Challenge-response asking for word identification
    - Useful for foiling scripts



CMSC 330

23

## More Reverse Turing Tests



CMSC 330

24