CS 3313:
Foundations of Computing
Part II

http://gw-cs3313.github.io

1

Regular Languages - Summary

- Finite state machines
  - DFA, NFA
- Regular expressions
- DFA as an algorithm
  - Example: substring search (and application to DNA sequence matching)
    Above all: about problem solving and algorithmic thinking
- Properties of regular languages
  - Closure properties
  - Decision properties
- Pumping lemma – to prove a language is not regular
  - Limits of finite state machines
    - Ex: we now know that checking syntax of programming languages cannot be done by a Finite state machine (using $a^n b^n$ to model the property)

2

Course Schedule - Topics

- Part 1: Regular Languages and Finite State Automata. (Weeks 1-5). familiar territory but now from a math perspective
  - Finite Automata ….same as Finite State Machines in Hardware!
  - Regular expressions to denote regular languages (same as Unix RegEx)
  - Properties of regular languages
- Part 2: Context Free Languages and Grammars (weeks 5-10)
  - Pushdown Automata – adding simple ”memory” to finite state machines
  - Formal grammars – context free grammars and a parsing algorithm
- Part 3: Turing Machines and Computability
  - Turing machine model and Universal Turing machine
  - What is computable ? Proving a problem is not solvable
  - Computational complexity models.

3

Recall definition: Automata

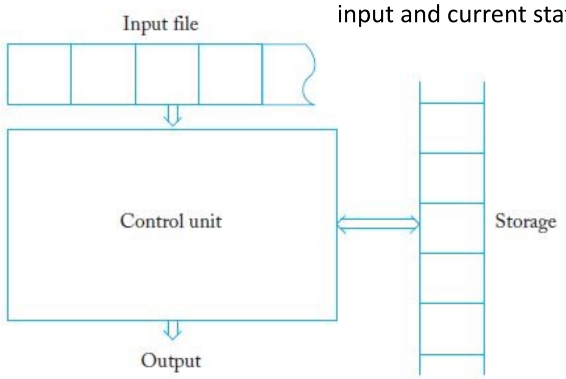
- An ***automaton*** is an abstract model of a digital computing device
- An automaton consists of
  - An input mechanism
  - A control unit
  - Possibly, a storage mechanism
  - Possibly, an output mechanism
- Control unit can be in any number of internal *states*, as determined by a *next-state* or *transition* function.
- There are a finite number of states

4

Illustration of a General Automaton

At each **step**:

- Machine reads one symbol from input,
- Determines next state based on input and current state (and storage)



Note: this is one model of the “type” of external storage
The storage model changes based on the automata model
Question: Is there storage in a finite state machine ?

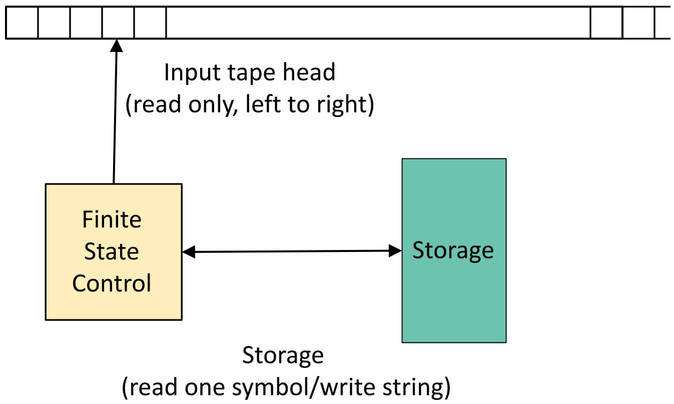
5

Modifying Finite State Automata

- Finite Automata (Deterministic & Non-deterministic)
  - These model Finite State Machines
- Observation: no external memory
  - State can summarize past events but cannot operate as an arbitrary size memory (eg. Register/counter to store any value, buffer to store arbitrary length strings, etc.)
- Changing Finite Automata model: Add the simplest form of memory to a Finite state machine
- What is the simplest form of storage ?

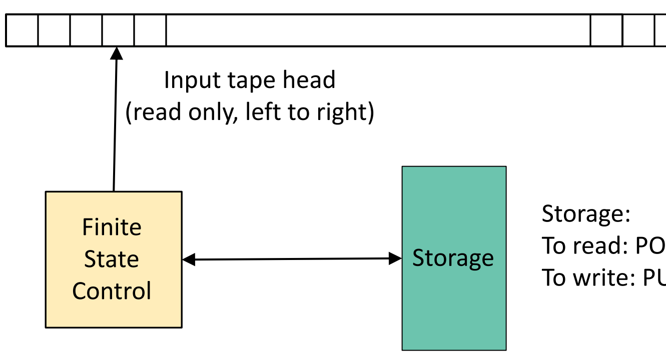
6

Automata with Storage



7

Automata with Stack Storage



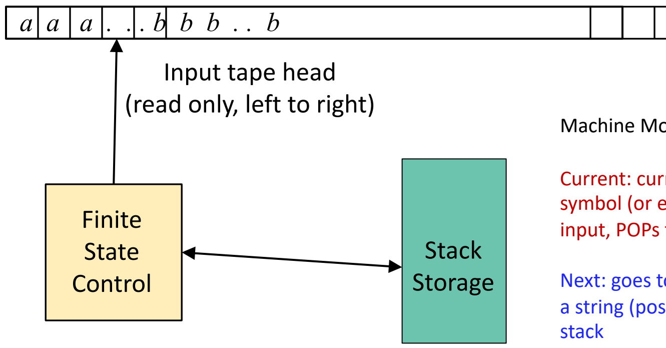
Storage:
To read: POP one symbol
To write: PUSH string

Machine Model: In one step
Current: current state, reads symbol (or empty string) from input, reads top of stack (POPs top of stack)
Next: goes to next state, PUSHes a string (possibly empty) to the stack

Accepts: if machine is in a Final state (or if stack is empty)

8

Automata with Stack Storage



Machine Model: In one step

Current: current state, reads symbol (or empty string) from input, POPs top of stack

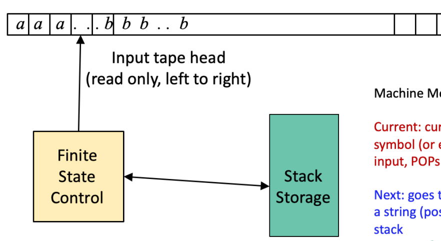
Next: goes to next state, PUSHes a string (possibly empty) to the stack

Accepts: if machine in Final state

Question: Can you design an "algorithm" for this machine model to accept the language $\{ a^n b^n \mid n > 0 \}$

9

(Stack machine) Algorithm for L= $\{ a^n b^n \mid n > 0 \}$



Machine Model: In one step

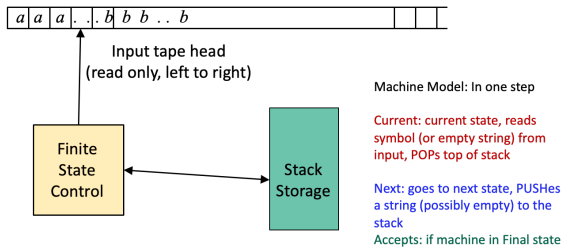
Current: current state, reads symbol (or empty string) from input, POPs top of stack

Next: goes to next state, PUSHes a string (possibly empty) to the stack

Accepts: if machine in Final state

10

(Stack machine) Algorithm for L= { wcw^R | w in {a,b}* }



11

Next Topic: Pushdown Automaton (PDA)

- This machine model (NFA with stack storage) is formally known as a Pushdown Automaton (PDA).
  - The default PDA definition is a non-deterministic machine

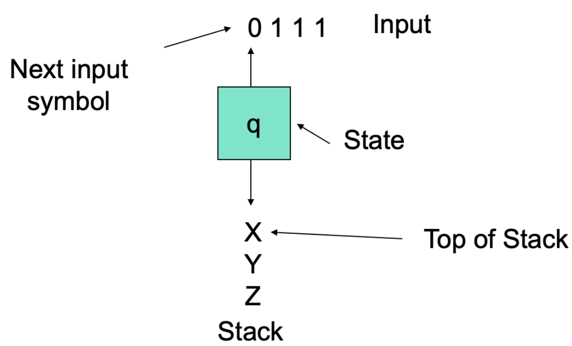
    from current configuration, it has number of choices for next move

    each choice specifies: next state, push string to stack
- The PDA is an automaton that accepts Context free languages
  - and equivalent to Context Free Grammars in language-defining power.
- Only the nondeterministic PDA defines all the CFL's.
- But the deterministic version models parsers.
  - Syntax of most programming languages have deterministic PDA's.

12

Intuition: PDA

- Think of an λ-NFA with the additional power that it can manipulate a stack.
- Its moves are determined by:
  1. The current state (of its "NFA"),
  2. The current input symbol (or λ), and
  3. The current symbol on top of its stack.



13

13

Intuition: PDA – (2)

- Being nondeterministic, the PDA can have a choice of next moves.
- In each choice, the PDA can:
  1. Change state, and also
  2. Replace the top symbol on the stack by a sequence of zero or more symbols.
     - Zero symbols = "pop" and no push
     - Many symbols = "push" a string

14

14

PDA Formalism

- A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is described by:
  1. A finite set of *states* Q (same as before).
  2. An *input alphabet* Σ (same as before).
  3. A *stack alphabet* Γ (typically assume Γ disjoint from Σ).
  4. A *transition function* δ
     - $\delta: (Q \times (\Sigma \cup \lambda) \times \Gamma) \rightarrow 2^{(Q \times \Gamma^*)}$ (subset of $Q \times \Gamma^*$)
       - Number of choices (i.e., non-deterministic)
       - Ex: $\delta(q_1, 0, X) = \{ (q_1, XX), (q_2, \lambda) \}$
  5. A *start state* q_0, in Q (same as before).
  6. A *start symbol* Z_0, in Γ (to indicate bottom of stack).
  7. A set of *final states* $F \subseteq Q$

  Need a few more definitions and notations to define acceptance….will return to this after Exam 1

15

15

Next 5 weeks

- Pushdown Automata
  - Formal definitions
  - Designing PDAs – can use JFLAP again!
- Grammars – Context Free grammars
  - Parse trees (there we go again with graphs and trees!!!)
  - Normal Forms for grammars
  - A simple parsing algorithm
- Equivalence of PDAs and CFGs
  - Definitions…we skip the details in this course
- Properties of Context Free Languages
  - We will pump you up again !!

16

Exam 1 – Thursday Feb. 10th

- Content: ALL topics on regular languages
  - Weeks 1—4
  - HW 1—3 and Quizzes 1—3
- Do you need to memorize all the theorems ?
- Will there be proofs ?
- Will you have enough time ?
- Will there be a review session before the exam ?
- Will you pass the exam

17

Review: When is a language regular

- A language is regular iff there is a finite state machine (DFA or NFA) or regular expression for the language
  - To prove a language is regular, you have to provide a NFA/DFA or Reg. Expr.
- To prove a language is not regular, use the pumping lemma to derive a contradiction
  - Assume language is regular
  - Use pumping lemma to show a contradiction

18

The Pumping Lemma for Regular Languages

For every regular language L

There is an integer n, such that

For every string w in L of length $\geq n$

We can write $w = xyz$ such that:

1. $|xy| \leq n$.
2. $|y| > 0$.
3. For all $i \geq 0$, xy^iz is in L.

Number of states of DFA for L

Labels along first cycle on path labeled w

19

19

How do use the pumping lemma: 2 person adversarial game

- **For all** regular languages L, **there exists** n…**for all** w in L

….**there exists** xyz….

- To show a contradiction, pick a string (express in terms of n – the constant of the lemma), and then find *one* value of i >= 0 such that xy^iz does not belong to the language

20

Pumping Lemma…another example

- Can we check if a number is a prime by using a DFA ?
- Formally: Is $L = \{ a^j \mid j \text{ is a prime number}\}$ regular ?
  - L uses “unary” notation to represent an integer
  - Why is this useful…checking for primality is a (very) important step in crypto algorithms

- Check the tutorial video for the proof…
  - Key observation: this is an example where we need to find that exact one value of i such that xy^iz is not in L to find a contradiction using the pumping lemma

21

Is the language regular

- $L_1 = \{ a^j b^j \mid j >=0 \} \cup \{a^j b^k \mid j,k >=0 \}$

- $L_2 = \{0^i 1^j 2^k 3^l \mid i+j = k + l \}$

- $L_3 = \{ w \mid w \in \{a,b,c\}^* \text{ and } |w| = 3\, n_a(w) \text{ where } n_a(w) \text{ is number of a's in } w \}$

22

More examples…is L regular ?

- $L_4 = \{a^j b^j \mid j <= 20\}$

- $L_5 = \{ a^j b^k c^l \mid j+k+l > 5 \}$

- $L_6 = \{ a^j b^l \mid j >=100 \text{ and } l <= 100 \}$

23

Pumping Lemma – Another example

- $L_7 = \{ a^k \mid k = j^2$ *for some* $j > 0 \}$ *…check notes/book*
- $L_8 = \{ a^j b^k a^l \mid j=k$ *or* k *is not equal to* $l\}$

- $L_9 = \{ a^{k!} \mid k > 0\}$ *(where k! is k factorial)*

24