CS 361
FINAL REVIEW

Chapter 1

- Vocabulary
  - String, Language, alphabet, empty string
- Regular Languages
  - Definition
  - Closed Operations
- Finite Automaton
  - Deterministic
  - Non-deterministic
  - ~~NFA to DFA~~
  - ~~Relationship between FA and Regular Languages (regular expressions~~)
- Pumping Lemma

Chapter 2

- Context Free Grammar
  - Create grammars to describe languages
  - Generate (trace) strings using grammars
- Push-Down Automata
  - Notation is very important
  - Understanding under which conditions a strings is accepted/rejected is key
- ~~Pumping Lemma for Context-free Languages~~

Chapter 3

- Vocabulary
  - Power of automata, Turing decidable, Turing recognizable
- Turing Machines
  - Ordinary TM
  - Variations of TM
  - Deciding versus recognizing a language
- Halting Problem
- Comparisons between FA, PDA, and TM

Chapter 4

- Vocabulary
  - Decider, simulate TM, halt
- What is a decidable problem?
- What is a decider?
- Constructing deciders to determine whether a problem is decidable or not is key
  - Understanding and knowing how to prove that a problem is decidable is the most important aspect of this chapter
- Undecidability
- Relationship among classes of languages

- Consider the following statements
  1. Recursive languages (Turing -Decidable) are closed under complement **TRUE**
  2. Every language can be recognized by a Turing Machine **FALSE**
  3. L={<M,w,t> | M is a TM that accepts string w in t transitions} is decidable **TRUE**
  4. L={<M> | M is a TM that does not accept the string "SUMMER"} is not decidable **TRUE**

Chapter 5

- What is reduction?
- What is the point of reduction?

Vocabulary

- A **decision problem** consists of a set of questions whose answers are either yes or no and is undecidable if no algorithm that can solve the problem; otherwise, it is decidable.

- An **unsolvable (or undecidable) problem** is a problem such that there does not exist any TM that can solve the problem.

Vocabulary

- A **recursively enumerable language** L is a formal language for which there exists a TM that will halt and accept an input string in L, and may either (i) halt and reject, or (ii) loop forever, otherwise.

- A **recursive language** L is a formal language for which there exists a TM that will halt and accept an input string in L, and halt and reject, otherwise.

Decidable/Recognizable /Not Recognizable

- B={<M> | M is a TM that accepts the string "SUMMER"}

  Recognizable

- C={<M> | M is a TM that does not accept the string "SUMMER"}

  Not Recognizable

- D={<M> | M is a TM that accepts the string "SUMMER" in 10 steps}

  Decidable

Chapter 7

- Computational versus Complexity Theory
- Solving problems versus solving them efficiently
- P, NP, NP-Hard, and NP-Complete
  - These are the key concepts for this chapter

Vocabulary

- A problem is **efficiently** solvable if there is a deterministic, 1-tape TM that solves the problem whose computations are polynomially bounded
- **P**: the set of all problems that can be solved in polynomial time using a deterministic TM
- **NP**: the set of all problems that can be solved in polynomial time by a non-deterministic TM
- **NP-Hard**: A problem which all M ϵ NP can be reduced to in polynomial time
- **NP-Complete**: An NP-Hard problem that is in NP

P and NP

- True or false:
  - P is a proper subset of NP
    - FALSE, we cannot be sure if P and NP are disjoint sets
  - If M_1 can be reduced to M_2 and $M_2 \epsilon P$, then $M_1 \epsilon P$
    - FALSE, unless the reduction is done in polynomial time

Some practice problems

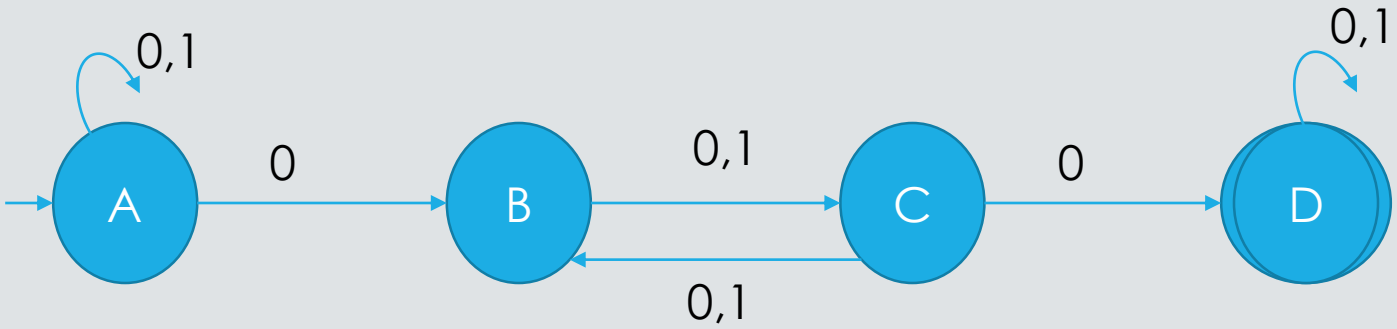
1. Consider the following language L={w over {0,1}* | w includes two 0's separated by an odd number of symbols}. If it is regular, create the **NFA**, if not, use the pumping lemma to show it is not regular.
2. Consider the following language L={$a^n b^n$ | n >0}. If it is regular, create the DFA, if not, use the pumping lemma to show it is not regular.
3. Create a grammar for language L={$a^n b^n$ | n >0}.
4. Use set notation to describe the language generated by the grammar G below.

$S \rightarrow AD$
$A \rightarrow aAb \mid bB$
$B \rightarrow bbBcc \mid eCe$
$C \rightarrow eCe \mid$
$D \rightarrow \#D \mid \varepsilon$

Some practice problems

5. Consider the following language L={$a^n b^m$ | n>m, n≥0, m>0}. If it is context free, create the PDA, if not, use the pumping lemma to show it is not context-free.
6. Construct the TM that accepts the language L={a^{2^n} | n≥0}, using a 1-tape TM.
7. Given ALL_{DFA}={<D> | D is a DFA that accepts all the strings over Σ}, show that it is decidable.
8. Show that the following language is decidable, by constructing a decider for it: L = {<M, w> | M is TM that moves its head left at any point of the computation of w}

- Consider the following language L={w over {0,1}* | w includes two 0's separated by an odd number of symbols}. If it is regular, create the **NFA**, if not, use the pumping lemma to show it is not regular.



- Consider the following language L={$a^n b^n$ | n >0}. If it is regular, create the DFA, if not, use the pumping lemma to show it is not regular.

Assume L is regular and let *p* be the pumping length.
Consider string $s=a^p b^p \in L$, which can be separated into 3 substrings, such that s=xyz, provided that |y| > 0 and |xy| ≤ p.

$$s=\begin{cases} x = a^k \\ y = a^j \\ z = a^{p-k-j} b^p \end{cases}$$

Since L is regular, then $xy^i z \in L$ for i>0.
Lets explore the case when i=2, since xy^2z should be part of L.

However $s = \begin{cases} x = a^k \\ y^2 = a^j a^j \\ z = a^{p-k-j} b^p \end{cases} = a^{p+j} b^p$. Since |y| >0, then so is j,

thus *p+j* > *p*, which means that s ∉ L, which is a contradiction.
Therefore, L is not regular.

Grammars

- Create a grammar for language L={$a^n b^n$ | n >0}.

S→aSb | aAb
A → ε

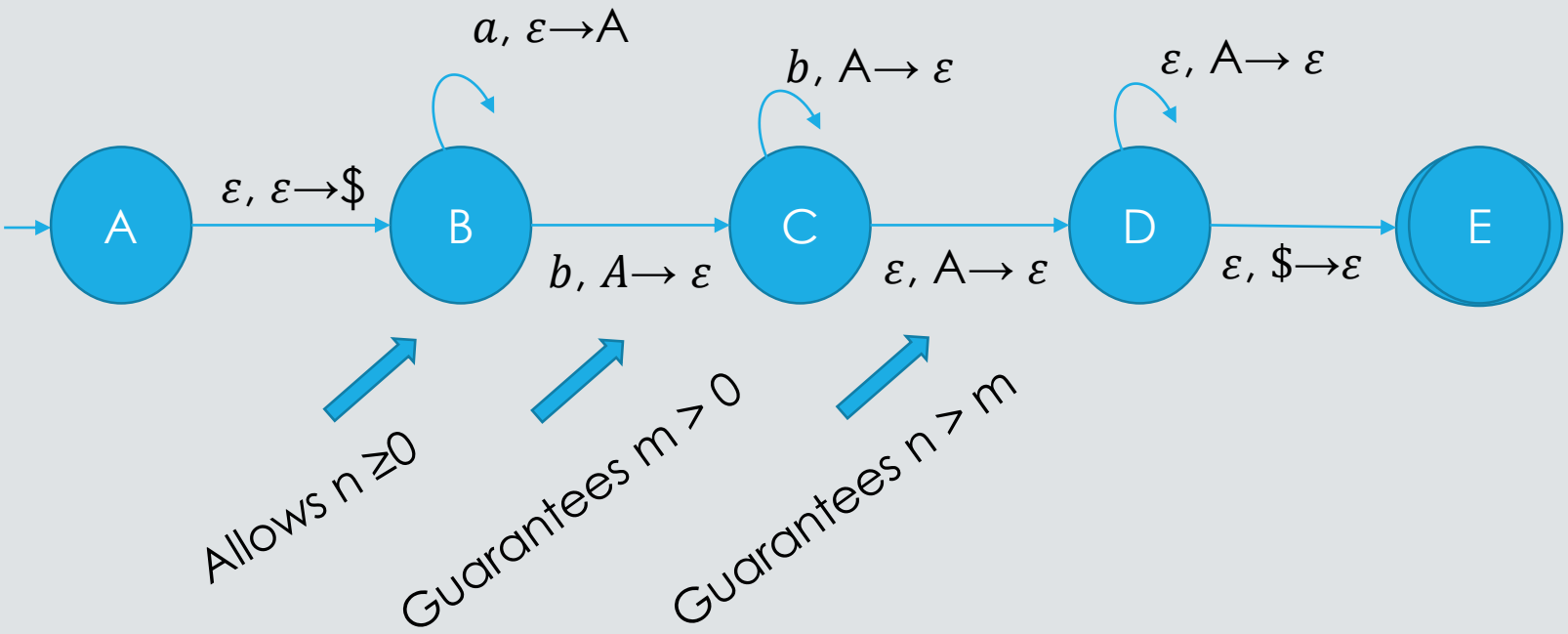
S→aAb
A → aAb | ε

- Use set notation to describe the language generated by the grammar G below.

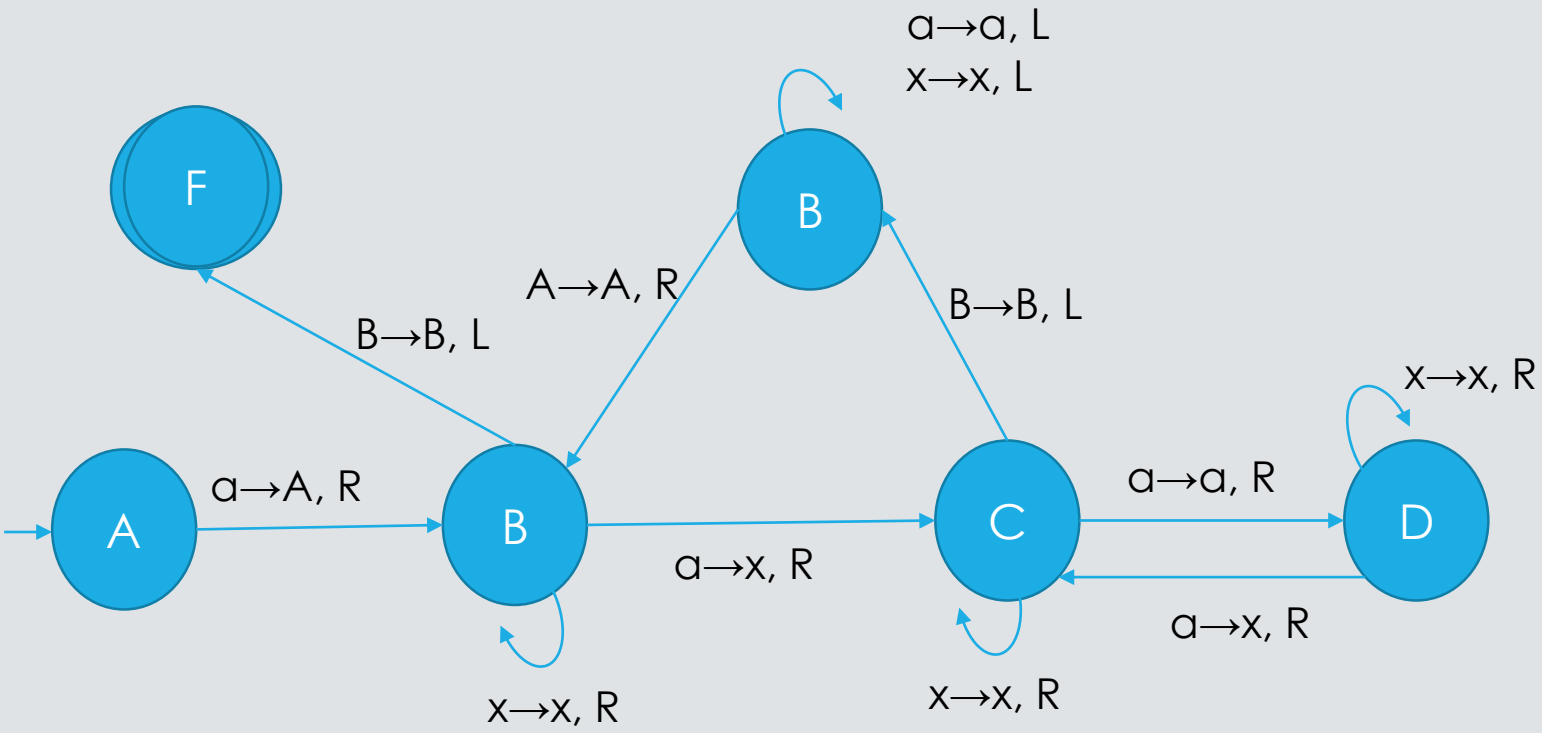
S → AD
A→aAb | bB
B →bbBcc | eCe
C →eCe |
D →#D | ε

$$L(G) = \{a^n\ b^{2m+1}(ee)^p c^m b^n \#^q \mid n, m, q \geq 0, p \geq 1\}$$

- Consider the following language L={$a^n b^m$ | n>m, n≥0, m>0}. If it is context free, create the PDA, if not, use the pumping lemma to show it is not context-free.

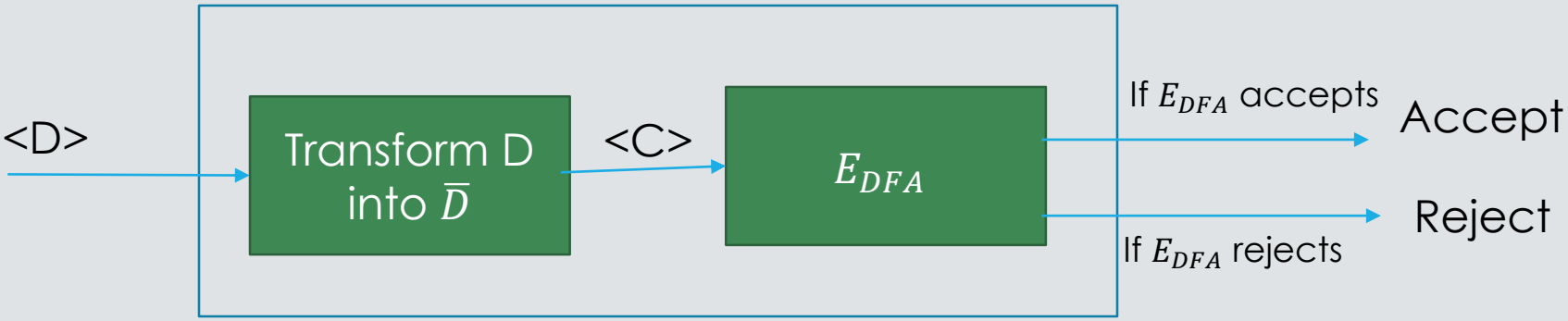


- Construct the TM that accepts the language L={a^{2^n} | n≥0}, using a 1-tape TM.



- Given ALL_{DFA}={<D> | D is a DFA that accepts all the strings over Σ }, show that it is decidable.

1. On input <D>, create a TM (decider) R
    1. R constructs a DFA C that recognizes the complements of L(D), which is regular, since regular languages are closed under the compliment operation
    2. Since L(C)= L($\overline{D}$), then for D to be part of ALL_{DFA} means that the language of its complement should be empty, i.e., L(C)= L($\overline{D}$)=ϕ. Hence R simulates the decider for E_{DFA} (the languages of the emptiness of a DFA) on <C>
    3. If E accepts, R accepts, Otherwise, R rejects.



- Show that the following language is decidable, by constructing a decider for it: L = {<M, w>| M is TM that moves its head left at any point of the computation of w}
- S: on input <M,w> =
  1. Compute the number of states of M, denoted n
  2. Simulate M on input w for at most |w| + n + 1 steps
     1. If M goes left during the simulation, accept
     2. If M did not go left during the simulation, reject

S decides L because it is bounded by a finite computation length

- |w| + n + 1 steps are sufficient, since after "visiting" |w| states and moving, in the worst case, to the right each time, all further computation will be spent reading blank symbols (having passed the input w length). Because the input symbol will be constant hereafter during the simulation while reading to the right, we can consider what will happen over the following n+1 states. These n+1 states incur the longest possible transition sequence the TM can traverse and still go to unique states. Thus, if the TM does not move left during this number of transitions, it will then loop (or may already be looping) through the states on the same input, and will loop forever moving right