CSE 105
Theory of Computation

- Professor Jeanne Ferrante
Today’s Agenda
– Algorithms and Decidability (Continued)
  • Using TM’s to decide problems about regular languages
– Review for Exam
  • Exam Practice Problem

Reminders and announcements:
Exam 2 this Wed May 11, 8 pm - 9:50 pm in WLH 2001
• Bring
• Your ID for checking
• Your seat assignment (available on TritonEd under Exam Resources)
• Your own 3 in by 5 in index card
• More Office Hours before Exam, Check it out on Google Calendar on class web page
• Get a good night’s sleep!
Review: Encoding Input for TM’s

• Many objects can be represented as strings:
  – Integers, Tuples, Polynomials, Graphs, Finite Automata, PDA’s, Regular Expressions,…

Notation:

• <O> is encoding of object O as string
• <O₁…Oₙ> is encoding of tuple of objects O₁…Oₙ as a single string
• Think of <…> as “Convert to String(…)”
• Ex: <B,w> for DFA B and input string w
TM’s used to solve problems!

PROBLEMS ABOUT REGULAR LANGUAGES
ADFA  =  \{<B,w> | B is a DFA that accepts input string w\}

THM 4.1: ADFA is decidable.

Proof: We show that there is a TM that decides ADFA by giving a high-level description of M.

M = “On input <B,w>:
1. Check whether B is a legal encoding of a DFA, and that w is valid input for B. If not, reject.
2. Simulate B on w by keeping track of B’s current state and current position in the input string w, using the tape. The simulation starts with B in start state, reading the first input symbol of w, and gets updated according to the transition function δ.
3. If the simulation of B on w ends in a final state, M accepts. If the simulation of B ends in a non-final state, M rejects. “

Correctness: M is a decider, since simulating a DFA on a fixed string w will always terminate in some state after reading the input; M will either accept or reject.

M accepts <B,w> iff DFA B accepts w. Therefore M decides ADFA.
Th. 4.4. $E_{DFA} = \{ <A> \mid A \text{ is a DFA with } L(A) = \emptyset \}$ is decidable

Proof: A DFA accepts some string iff it is possible to reach a final state from the start state along edges of the DFA. We’ll define a TM $M$ that follows the DFA edges starting from start state to check if there is any way to end in a final state.

$M = “On \text{ input } <A>: \quad$
1. Check that $A$ is a legal encoding of a DFA. If not, reject.
2. Mark the start state of $A$.
3. Repeat until no new states get marked:
   Mark any unmarked state with an incoming edge from a marked state
4. If no final state of $A$ is marked, then $L(A)$ is empty; accept. Otherwise, reject.”

$M$ will mark
A. All the states of DFA $A$
B. All the states of DFA $A$ reachable from the start state
C. States in the DFA more than once
D. None of the above
Other Decidable Problems

• We can use decidable problems or construction algorithms to show new problems are decidable
• We know $E_{\text{DFA}}$ (checking if the language of a DFA is empty) is decidable (Th. 4.4)
• Can show $N_{\text{DFA}} = \{ <D> \mid D \text{ is a DFA and } L(D) \text{ is nonempty} \}$ is decidable.

“On input $<D>$:
1. Run the decider $M$ for $E_{\text{DFA}}$ on $<D>$.
2. If $M$ accepts, reject; if $M$ rejects, accept”

• Therefore $N_{\text{DFA}}$ is decidable
Other Decidable Problems for Regular Languages

The following are also decidable:

• Whether a regular expression generates a string \( w \)

• Whether an NFA generates a string \( w \)

We can conclude the above problems are also decidable because:

A. A regular expression is a DFA
B. A regular expression or an NFA can be converted to an equivalent DFA
C. Regular expressions and NFA’s are finite
D. None or all of the above.
Th 4.5 $\text{EQ}_{\text{DFA}} = \{ <A,B> | \ A, \ B \text{ are DFA's} \ & \ L(A) = L(B) \}$ is decidable

Proof: Want to show that $\text{EQ}_{\text{DFA}}$ is decidable.

Consider $L = (L(A) \cap L(B)^c) \cup (L(B) \cap L(A)^c)$. Since $L(A)$ and $L(B)$ are regular, and the regular languages are closed under complement, intersection and union, there is an algorithm to construct a DFA $D$ accepting language $L$.

We note that $L = \emptyset$ IFF $L(A) = L(B)$. Therefore, we can use the TM $M$ of Th. 4.4 (that decides if DFA $A$ has $L(A) = \emptyset$) as a subroutine in decider $N$:

$N = \text{“On input } <A,B>:\n1. \text{ Construct DFA } D \text{ for } L \text{ from the DFA’s for } A \text{ and } B.\n2. \text{ Run TM } M \text{ from Th. 4.4 on input } <D>\n3. \text{ If } M \text{ accepts, accept. If } M \text{ rejects, reject. “}\n
Correctness: $N$ accepts $<A,B>$ iff $M$ accepts $<D>$ iff $L(D) = \emptyset$ iff $L = \emptyset$ iff $L(A) = L(B)$.

$N$ is a decider, since the construction in 1. always terminates, and $M$ is a decider. Thus $N$ decides if $\text{EQ}_{\text{DFA}}$. QED
Decidable Problems for CFL’s

• $A_{CFG} = \{<G,w> \mid G \text{ is a CFG that generates string } w\}$
• $E_{CFG} = \{<G> \mid G \text{ is a CFG and } L(G) = \emptyset\}$

Decider $S$ for $A_{CFG}$ can be used to show every CFL $L$ (with CFG $G$) is decidable:

$M = \text{“On input } w:\n1. \text{ Run TM } S \text{ that decides } A_{CFG} \text{ on } <G,w>\n2. \text{ If } S \text{ accepts, accept; if } s \text{ rejects, reject”}
What are the closure properties?

MORE CLOSURE PROPERTIES OF DECIDABLE AND TURING-RECOGNIZABLE LANGUAGES
Prove that the class of Turing-recognizable languages is closed under Union

• Given: Two T-recognizable languages A and B over alphabet Σ, and TMs that recognize them, \( M_A \) and \( M_B \).
• Want to Show: A TM \( M_U \) that recognizes A U B.
• Construction:
  – \( M_U = \) “On input w: //w is a string
    1. Run \( M_A \) in parallel with \( M_B \) on input w
       – If either accepts, accept. If both reject, reject.”
• Correctness: // TBD
• Conclusion: \( M_U \) is a TM that recognizes AUB, therefore recognizable languages are closed under union. QED.
Proving Correctness of previous closure proof:

**Correctness:**

We want to show that

1. \( L(M_\cup) = A \cup B: \)

If \( w \) is in \( L(M_\cup) \) then \( w \) is accepted by \( M_A \) or \( M_B \). Since we run both in parallel in \( M \), and accept if either one does, \( w \in A \cup B \)

If \( w \) is not in \( L(M_\cup) \) then neither \( M_A \) nor \( M_B \) accepts \( w \), or \( M_\cup \) would accept \( w \), by our construction. So \( w \) is not in \( A \cup B \). QED
Closure Properties of Collections of TM Languages

The Decidable Languages are closed under:

- Union
- Intersection
- Complement (HW 5)
- Star
- Concatenation

The Turing-Recognizable Languages are closed under:

- Union
- Intersection
- Complement
- Star
- Concatenation
It's tomorrow evening!

EXAM 2 REVIEW
Review: Design a CFG

• Design a CFG for \( L = \{ w \in \{0,1\}^* \mid w \text{ has at least three 1's} \} \)

• How do we generate string with at least 3 1’s:
  \[ R \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ R, \text{ where } R \text{ is any string over } \{0,1\} \]

• How do we generate any string over \( \{0,1\} \):
  \[ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ R \text{ or } 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ R \text{ or } \varepsilon \]

• CFG:

\[
S \rightarrow R1R1R1R \\
R \rightarrow 0R \mid 1R \mid \varepsilon
\]

Variables = \{S, R\} Start Variable: S Terminals = \{0,1\}
Language of the CFG?

- $S \rightarrow A \ B \ C$
- $A \rightarrow aA \mid \epsilon$ $B \rightarrow bB \mid \epsilon$ $C \rightarrow cC \mid \epsilon$
Pushdown Automata

At each step, transition from state $q$, input $a$, top of stack $x$ to state $r$, replace $x$ with $y$ on top of stack.

Accept: IF Have read ALL the input, AND in final state of PDA BOTH MUST HOLD! (Don’t require the stack to be empty)
Review: Th. 2.20 A language is context-free IFF some PDA recognizes it.

• If a PDA recognizes $L$, there is a CFG that generates $L$

• If a CFG generates language $L$, there is a PDA that recognizes $L$

• We won’t prove this, but you should know and be able to cite and use it in future
Review: Informal Description of PDA

L = \{w \in \{0,1\}^* \mid w \text{ has more 0's than 1's}\}

1. If the input is the empty string, reject.
2. If there is more input to be read, read the next symbol. If the stack is empty, push the symbol onto the stack. Otherwise, if the symbol is different from the top of the stack, pop the stack; If it is the same as the top of the stack, push it onto the stack. In all cases, continue with step 2 until all the input has been read.

3. If the top of the stack is 0, then there were more 0’s than 1’s, accept. If the top of the stack is 1, then there were more 1’s than 0’s, reject. If the stack is empty, then there were an equal number of 0’s and 1’s, reject.
Tracing Computation in a PDA

0001

0000
TM Deciders Vs. Recognizers

• Every TM $M$ with language $L$ halts and accepts all the strings in $L$

• The difference between TM Deciders and Recognizers is what they do on strings not in $L$

• On input strings not accepted by the TM
  – Recognizers can either reject or not halt ("loop infinitely")
  – Deciders always halt and reject

• A language $L$ is decidable if there is some TM decider that recognizes it

• A language $L$ is recognizable if there is some TM that recognizes it
TM Techniques

- **Subroutines:** Once we know that a problem is decidable, we can use it’s decision procedure as a subroutine in other algorithms
  - \( E_{DFA} = \{<B> \mid B \text{ is a DFA and } L(B) \text{ is empty}\} \) is decidable

- **Constructions:** Once we have an algorithm for a construction, we can also use those as subroutine in TM algorithms
  - Convert a Regular Expression to an NFA Sipser Lemma 1.55
Church-Turing Thesis

Intuitive notion of algorithm

Precise notion of Turing Machine

Any algorithm can be implemented on a Turing Machine
Supporting Evidence: Equivalence of Variants of TM’s

• There are many different models:
  – Nondeterministic
  – Multiple Tapes
  – 2-sided tapes
  – ...

• Good news: they are all equivalent in power to ours

• You can choose to use nondeterminism or multiple tapes as appropriate in TM constructions

All have unrestricted access to unlimited memory
## Closure Properties of Collections of TM Languages

<table>
<thead>
<tr>
<th>The Decidable Languages are closed under:</th>
<th>The Turing-Recognizable Languages are closed under:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Union</td>
<td>• Union</td>
</tr>
<tr>
<td>• Intersection</td>
<td>• Intersection</td>
</tr>
<tr>
<td>• Complement</td>
<td>• Complement</td>
</tr>
<tr>
<td>• Star</td>
<td>• Star</td>
</tr>
<tr>
<td>• Concatenation</td>
<td>• Concatenation</td>
</tr>
</tbody>
</table>
Closure Properties of Collections of Regular and CFL Languages

The Regular Languages are closed under:

• Union
• Intersection
• Complement
• Star
• Concatenation

The CF Languages are closed under:

• Union
• Intersection
• Complement
• Star
• Concatenation
Status: Language Hierarchy

- **Context-Free**: Decidable but not Regular: \(\{0^n1^n | n > 0\}\) By pumping lemma (Regular)
- **Decidable**: Turing-Recognizable but Not Decidable: ?
- **Turing-Recognizable**: Not Turing Recognizable: ? By counting

\[\text{CF but not Regular:}\quad \{0^n1^n | n > 0\}\]
\[\text{Decidable but not CF:}\quad \{ a^n b^n c^n | n > 0\}\]
\[\text{Turing-Recognizable but Not Decidable:}\quad ?\]
\[\text{Not Turing Recognizable:}\quad ?\]
EXAM 2 Practice

• Prove that the class of Turing-recognizable languages is closed under intersection.
Prove that the class of Turing-recognizable languages over $\Sigma$ is closed under Concatenation

- Given: Two Turing-recognizable languages $A$ and $B$ over $\Sigma$, and TM’s that recognize them, $M_A$ and $M_B$.
- Show: There is a deterministic TM $M$ that recognizes $A \cdot B$.
- Construction: We give a high-level description of TM $M$.
  - $M =$ “On input $w$: // $w$ is a string
    1. For every $i = 1, 2, \ldots$
    2. For every split of $w$ into $x$, $y$ in $\Sigma^*$ such that $w = xy$:
      a) Run $M_A$ $i$ steps on input $x$ and run $M_B$ $i$ steps on input $y$. If either
          rejects, go to next split. If both accept, accept.
    3. If every split of $w = xy$ fails in 2.a, then reject”
- Correctness: $w$ is in $A \cdot B$ IFF $M$ accepts $w$
  If $w$ is in $A \cdot B$ then there is some split of $w = xy$ with $x$ in $A$ and $y$ in $B$.
  When we run $M_A$ on $x$ and $M_B$ on $y$ in $M$, both halt and accept within some number $i$ steps.
  Therefore $M$ will halt and accept $w$ in 2.a.
  If $w$ is accepted by $M$, there must be some $i$ and split $w = xy$, where $M_A$
  accepts $x$ and $M_B$ accepts $y$ within $i$ steps in 2.a. So $w = xy$ is in
  $A \cdot B$.

Conclusion: We have constructed a TM that recognizes $A \cdot B$, therefore $A \cdot B$ is Turing-recognizable, and Turing-recognizable languages are closed under concatenation. QED.
Prove that the class of Turing-recognizable languages over $\Sigma$ is closed under Concatenation

- Given: Two Turing-recognizable languages $A$ and $B$ over $\Sigma$, and TM’s that recognize them, $M_A$ and $M_B$.
- Show: There is a non-deterministic TM $M$ that recognizes $A \cdot B$.
- Construction: We give a high-level description of TM $M$.
  - $M =$ “On input $w$: 
    1. Non-deterministically guess $x, y \in \Sigma^*$ such that $w = xy$ and:
      a) Run $M_A$ on input $x$. If it rejects, reject. If it accepts, go to step b.
      b) Run $M_B$ on input $y$. If it rejects, reject. If it accepts, accept.”
- Correctness: $w$ is in $A \cdot B$ IFF $M$ accepts $w$

If $w$ is in $A \cdot B$ then there is some split of $w = xy$ with $x$ in $A$ and $y$ in $B$. When we run $M_A$ on $x$ and $M_B$ on $y$, both halt and accept. Therefore $M$ will halt and accept $w$.

If $M$ halts and accepts $w$, it must occur in step 1.b for some split $w = xy$. For $M$ to accept in step 1.b, $x$ must be accepted by $M_A$ in step 1.a, and then $y$ must be accepted by $M_B$ in step 1.b. So $w = xy$ is in $A \cdot B$.

Conclusion: We have constructed a TM that recognizes $A \cdot B$, therefore $A \cdot B$ is Turing-recognizable, and Turing-recognizable languages are closed under concatenation. QED.