Theory of Programming Languages

COMP360
“Sometimes it is the people no one imagines anything of, who do the things that no one can imagine”

Alan Turing
What can be computed?

• Before people even built computers, there was considerable work on the theory of computing
• The concepts of “computing” and “accepting a string that is a member of a language” are almost the same
History

• In the seventeenth century Gottfried Leibniz formulated the Entscheidungsproblem or decision problem
• In 1928, mathematician David Hilbert published a paper on the decision problem, what can be computed
• In 1936, Alonzo Church and Alan Turing published independent papers showing that a general solution to the Entscheidungsproblem is impossible
Alan Turing

• Alan Turing was a British mathematician
• He invented the Turing machine
• According to the Church–Turing thesis, Turing machines are capable of computing anything that is computable
Noam Chomsky

• Noam Chomsky is an American linguist
• In the 1956 he developed a theory on the classes of formal grammars
• The Chomsky hierarchy defines a several classes of languages, each embedded in the next

• Chomsky has also been known for his activism
Chomsky Hierarchy

• There are four classes of languages
• Each class is a subset of another class
• Regular languages are the simplest while recursively enumerable (RE) languages are the most complex
Theoretical Computers

• Each class of languages can be recognized by a type of abstract machine

• The classes can be defined by those languages that can be recognized by the machine for that class
<table>
<thead>
<tr>
<th>Language</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Expressions</td>
<td>Deterministic Finite State Automata (DFA)</td>
</tr>
<tr>
<td>Context Free</td>
<td>Push Down Automata (PDA)</td>
</tr>
<tr>
<td>Context Sensitive</td>
<td>Bounded Turing Machine</td>
</tr>
<tr>
<td>Recursively Enumerabe</td>
<td>Turing Machine</td>
</tr>
</tbody>
</table>
Click the correct choice

A. All regular languages are context free
B. All context free languages are regular
C. No regular languages are context free
D. No context free languages are regular
E. All Recursively Enumerable languages are regular
Finite state automaton

• A finite state automaton (FSA) is a graph with directed labeled arcs, two types of nodes (final and non-final state), and a unique start state:

• This is also called a state machine

• The language accepted by machine M is set of strings that move from start node to a final node
FSA Execution

• The FSA has a “current state” which starts at the start state
• The input string is examined left to right one character at a time
• Based on the current input character, the FSA can move from the current state to a new state following an arrow labeled with the current input character
• Execution ends at the end of the input or if the FSA cannot move
More on FSAs

• An FSA can have more than one final state:
Example FSA

• This FSA recognizes a Java or C++ character string
• Remember that "" is an escape sequence that allows you to put a quote character in a string
Try It

• Write a DFA to recognize /* */ style comments
Deterministic FSAs

• **Deterministic FSA**: For each state and for each input symbol, there is exactly one transition

• **Non-deterministic FSA (NDFSA)**: Remove this restriction

• At each node there is 0, 1, or more than one transition for each alphabet symbol

• A string is accepted if there is *some* path from the start state to some final state

• Example nondeterministic FSA (NDFSA): 01 is accepted via path: ABD even though 01 also can take the paths: ACC or ABC and C is not a final state
Equivalence of FSA and NDFSA

• Important early result: NDFSA = DFSA

• Let subsets of states be states in DFSA

• Keep track of which subset you can be in

• Any string from \{A\} to either \{D\} or \{CD\} represents a path from A to D in the original NDFSA.
Try It

• Write a DFA to recognize // style comments
• You can use \n as “new line” or end of line
Regular expressions

- Can write regular language as an expression:

\[ 0^*11^*(0|100^*1)1^*|0^*11^*1 \]

Operators:
- Concatenation (adjacency)
- OR written as \(|\)
- Kleene closure (* is 0 or more instances)
- Repeat one or more is written as superscript plus \(+\)
FSA and Regular Languages

• An FSA can recognize any regular language
• Regular languages are often used for pattern matching or searches
• A popular tool for searching files for a word is grep, General Regular Expression Parser
Try It

• Write a DFA to recognize both // style and /* */ style comments
Equivalence of FSA and regular grammars

To go from regular grammar to FSA, make the following transformations:

- $X \rightarrow aY$
- $Y \rightarrow b$

- $X \rightarrow aY$
- $Y \rightarrow b$

- $X \rightarrow 1X|0Y$
- $Y \rightarrow 1Y|0Z$
- $Z \rightarrow 0Y|1Z$

- $X \rightarrow a$
- $Y \rightarrow 0$
- $Z \rightarrow 1$

- $X \rightarrow aY$
- $Y \rightarrow b$

Programming Language design and Implementation -4th Edition Copyright © Prentice Hall, 2000
Why do we care about regular languages?

• Programs are composed of tokens:
  • Identifier
  • Number
  • Keyword
  • Special symbols

• Each of these can be defined by regular grammars
Compiling

• There are several steps in compiling a program
• The first step is usually for a FSA program to read the program source code and create a list of tokens
• The list of tokens is then used by the following context free parser
FSA Limits

• The only “memory” that a FSA has is the current state
• As a FSA inputs symbols, it moves to different states
• An FSA cannot count or compare an arbitrary number of symbols
Push Down Automata

• A Push Down Automata (PDA) has a FSA and a stack memory

• The top of the stack, input symbol and FSA state determine what a PDA will do

• The input is examined one character at a time

• A PDA can:
  • Push a new symbol on the stack
  • Pop the top symbol from the stack
  • Change the FSA state
Push Down Automata Languages

• A Push Down Automata (PDA) can recognize context free grammars
• Most programming languages are context free grammars
• All regular languages are a subset of context free grammars
• A PDA can “count” things
A PDA looks at the input character, the top of stack character and FSA state then makes a decision.

It can push or pop the stack and change FSA state.

Like a FSA, it “accepts” the input if it ends in an accepting state.

It is a FSA with limited memory.
Turing Machine

• A Turing Machine (TM) has a FSA and an infinite tape
• The input is on the tape
• TM can take an action based on the current symbol on the tape and the current FSA state
  • Overwrite the symbol on the tape
  • Move left or right
  • Change the FSA state
Bounded Turing Machine

• A Bounded Turing Machine (BTM) has a limited tape
• A BTM can recognize a context-sensitive grammar (CSG)
Classical Languages Recognized

<table>
<thead>
<tr>
<th>Machine</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic Finite State Automata (DFA)</td>
<td>Regular expressions</td>
</tr>
<tr>
<td>Push Down Automata (PDA)</td>
<td>$a^n b^n$</td>
</tr>
<tr>
<td>Bounded Turing Machine</td>
<td>$a^n b^n c^n$</td>
</tr>
<tr>
<td>Turing Machine</td>
<td>Anything that can be recognized</td>
</tr>
</tbody>
</table>


Most Powerful

• A Turing Machine can compute anything that any computer can compute
• If something cannot be computed on a Turing Machine, it cannot be computed with any computer
Beyond Turing Machines

• There are some problems that cannot always be solved by a Turing Machine or any other computer
• These are non-computable problems
What is a Non-Computable Problem?

• Non-Computable Problems are well defined problems that are impossible to solve for all cases, by any algorithm

• These problems are provably non-computable. They cannot be solved with a faster computer or a better algorithm

• The best known undecidable problem is the Halting Problem
Halting Problem

• Can you write a program that inputs another program and some data to determine if the other program will eventually terminate (halt) when using the data?

• Proof by contradiction: Assume we have such a program (called Q)

figure from “Algorithmics, The Spirit of Computing”, 2nd ed. by David Harel
Halting Problem Proof

• Create program S calling program Q as a method.
• If Q answers Yes, go into an infinite loop

figure from “Algorithmics, The Spirit of Computing”, 2nd ed. by David Harel
Halting Problem Proof

• Call program S using S as the input
• If Q says S halts, then it doesn’t
• If Q says S does not halt, then it does
• This is a contradiction
• Therefore Q cannot exist

figure from “Algorithmics, The Spirit of Computing”, 2nd ed. by David Harel
Sometimes but not Always

• You can frequently solve a problem known to be non-computable
• The simple Hello World program will halt
  • `while (true) {}` never halts
• While you may be able to solve specific instances of the problem, you cannot write a program to always solve the problem
Non-Computable Problems

• There are a number of problems that have been proved non-computable including:
  • Post Correspondence Problem
  • Wang Tiles
  • Domino Snakes
Post correspondence problem

• Given a set of domino-like tiles with numbers on the top and bottom, can a series of tiles be placed side by side so that the string of numbers on the top matches the string of numbers on the bottom
• Assume you have an infinite number of each type of domino
Simple solution to this instance of the problem requiring seven dominos
More Difficult Problem

• If you change the bottom number on the last tile so that it only contains one zero, the problem becomes much more difficult.

• This simple change has a solution, but it requires 75 dominoes.
Wang tiles

• When using a set of square tiles with colored edges, can the tiles be arranged without rotation or reflection so that they tile a plane with adjacent tiles having edges of the same color?
• Assume you have an infinite supply of each tile
Tiling Solution
Periodic and Aperiodic

- The previous solution has a 2 by 4 block that can be repeated forever.
- Some sets of tiles can tile an infinite plane without repeating.
Aperiodic Tiling
Domino Snakes

• Using tiles similar to Wang Tiles, can a given set of tiles form a path from two given points so that all adjacent tile edges have matching colors?
Snake Solution