Midterm 1

The midterm is currently scheduled for Tuesday, October 16, from 5PM to 7PM in Arjona 115. Please say if this scheduling causes you significant difficulty.
The midterm is open book, with no electronic support.
The midterm covers the material in Chapters 1 and the subset of Chapter 2 prior to Section 2.4 Deterministic Context-Free Languages, Chapter 3 and that part of Chapter 4 covered by October 10.

Languages - Regular 2

Postulate a finite set of symbols
String is a sequence of symbols, from that set, possibly infinite
Language is set of strings
Class of languages is a set of languages
Chap. 1: DFA, NFA, regular languages, regular expressions
use a single, well-chosen string in the language and pumping lemma to prove a language was not regular
by showing no way to "pump" the string such that the resulting string was in the language
Machines take strings, whether or not they are in the language, and render a decision as to whether the string is in the language or not.
Regular expressions generate strings that are in the language.
Use definitions (of DFA/NFA and of regular language) and closure properties to show a language was regular.

Languages - Context Free 3

Chap. 2: Context free grammars, (Chomsky Normal Form), context free languages, non-deterministic push down automata
use a single, well-chosen string in the language and pumping lemma to prove a language was not context-free
by showing no way to "pump" the string such that the resulting string was in the language
Machines take strings, whether or not they are in the language, and render a decision as to whether the string is in the language or not.
Context free grammars generate strings that are in the language.
Use definitions (of PDA and of context free language) and closure properties to show a language was context-free.

Languages - Decidable, or only Recognizable 4

Chap. 3: Turing machines, Turing-decidable languages, Turing recognizable languages, algorithms
Robust model – changes such as more tapes, different ability to search the tape, don't change the power of the model.
There are problem statements whose answers cannot be computed (at all), and there are problem statements for which the answer might be an unpredictably long time arriving.
We are going to learn how to prove that some problem statements cannot be decided.

Languages 5

Though it might have seemed that the description of generating the language, regular expressions or context free grammar was the primary entity, or at times it might have seemed the machine, which tells whether or not a given string is in the language, was the primary entity, really it is the language that is the central character here.

The reason we are concentrating on the language is that the language is a representation of a problem we are trying to solve. Some problems can be cast as "decision problems": either a proposed element is in a set or it is not.

For example: Is this string a member of the set "syntactically correct Java programs"?

Decidable Languages 6

We start considering decidable language with some we know already, regular languages, but there's a difference.
"Is this language empty" is about a language rather than about a string.
"Are two finite automata, A and B, equivalent?" is the same question as, "Do A and B accept exactly the same language?".
Notice: "DFA accepts language $L(A)$" implies that DFA rejects strings not in $L(A)$.

Notation for Problem Cast as Language7

Problem: Does DFA B accept string w?
Suppose we had a set of all possible DFAs, and a way to tell (such as the DFA) whether a string was accepted by it.
Then we could ask, is B a DFA, and, does it accept w?
i.e., is the pair $\langle B, w \rangle$ a matched pair?
Representation:
$A_{DFA} = \{\langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w\}$
clicker questions

Problem Cast as Language8

So, $A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$ is a language, a set of strings.
We want to think about problems as deciding whether a given string is a member of a language.
We have seen how to think about whether a DFA accepts a string as a set membership question.
Once we are able to solve the set membership problem for the language, we can solve the problem we cast as a language / set membership problem.
It is not that this formulation made that problem easier, rather, it is expressed in a mathematical formulation.
Once we master this powerful tool (through practice, if necessary), we can begin to classify problems as decidable or not.

Practice Problem Cast as Language – Regular Expression9

$A_{REX} = \{\langle R, w\rangle \mid$
R is a regular expression that generates string $w\}$
We want to show A_{REX} is a decidable language.
We show this by construction, i.e., by constructing a TM P that decides A_{REX}.
In particular, we construct that DFA that results from the regular expression,
and we use the previously established technique to check out the string w.

Practice Problem Cast as Language – Empty Regular Language10

$E_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$

Empty language means, there is no string that gets accepted.

Equivalently, there is no path from start state to any accept state

Use a connectedness algorithm to trace paths:

Start at start state, mark it;

Repeat until no new state gets marked: Check each state in Q, if it has a incoming arc from a marked state, mark it. Once an iteration in which no newly marked state occurs, check whether any accept state is marked. If there are no accept states marked, then, this is empty language,

else, it is not the empty language.

clicker question

Practice Problem Cast as Language – Equivalent DFAs11

$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

Equivalent means, there is no string in either A or B that is accepted by one, yet rejected by the other.

So, we can construct the language C that contains those strings that are accepted by one, yet rejected by the other, and reuse our test of empty language on C.

How do we know that a Turing machine can construct the language C?

We proved that regular languages are closed under complementation, union and intersection.

clicker questions

Practice Problem Cast as Language – String Member of Context Free Language12

Consider:

$A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$

Recall that it is possible to convert a context free grammar into a grammar in Chomsky Normal form, and that there is a relationship between the number of substitutions in the grammar and the length of the string, in particular, for a string of length n, there were exactly $2n - 1$ substitutions in arriving at that string.

So to check whether G generates w, we have this algorithm:

$S =$"On input $\langle G, w\rangle$, where G is a CFG and w is a string:

1. Convert G to an equivalent grammer in Chomsty normal form.
2. List all sequences of valid substitutions of length $2n - 1$, (except if $n = 0$, in which case try all substitions of length 1).
3. If any of these derivations generate w, accept; otherwise, reject.

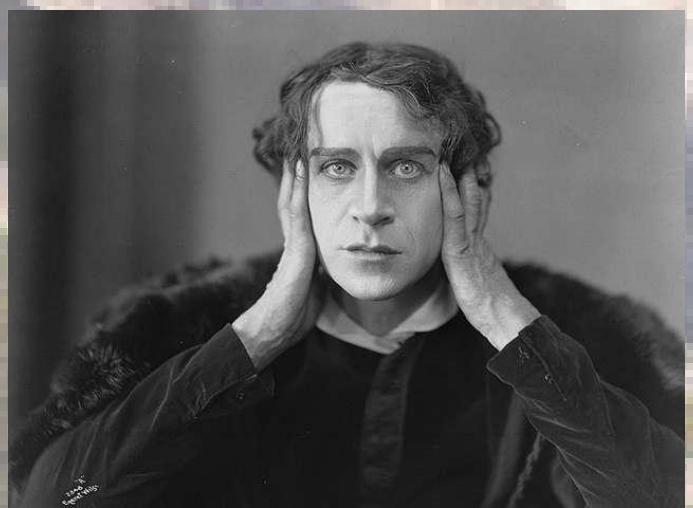
clicker question

Practice Problem Cast as Language – Context Free Language Generates No Strings13

Deciding E_{CFG}:
Recall the method for the language E_{DFA}, in which, starting with the start state, states were marked if there were a transition into the state from a state that was already marked. This marking process continued until no new states were marked. Then the accept states were checked to see whether any was marked.
With a context free grammar G in Chomsky Normal Form, we shall operate in reverse of this. We shall mark all terminals. Then we shall mark any variable A such that G contains a rule $A \rightarrow U_1 U_2 \cdots U_k$ consisting entirely of marked entities. We shall continue this process until no new variables are marked. Then the start variable is checked to see whether it was marked. If the start variable is marked, the language is not empty, implying the machine whose job it is, to decide this language, should reject; otherwise, the language is empty and the machine should accept.

Practice Problem Cast as Language – Context Free Language Equivalence14

We can state the language representing the problem:
$EQ_{CFG} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
but can we decide it?



The answer is, no, we cannot decide this language. In Chapter 5, we will discuss it further.

Context Free Languages are Decidable15

We cast this into a language as:
$A_{CFL} = \{\langle L, w\rangle \mid L \text{ is a CFL that generates string } w\}$
This looks a lot like:
$A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$
that we recently proved.
By recalling that there is an equivalence between CFG's and CFL's, (see Theorem 2.9, and the definition of the language of a grammar) we are encouraged to build a Turing machine that runs the previous Turing machine, that for A_{CFG}, and use it for A_{CFL}.