CSCI 340: Computational Models

Decidability

Chapter 18

Department of Computer Science

Decidability

1. How can we tell whether two CFGs define the same languages?
2. Given a CFG, how can we tell whether it is ambiguous?
3. Given an ambiguous CFG, how can we tell there exists a non-ambiguous CFG accepting the same language?
4. How can we tell whether the complement of a CFG is also context-free?
5. How can we tell whether the intersection of two CFGs is also context-free?
6. Given two CFGs, how can we tell whether they have a word in common?
7. Given a CFG, how can we tell whether there are any words it *does not* generate?

Decidability

1. How can we tell whether two CFGs define the same languages?
2. Given a CFG, how can we tell whether it is ambiguous?
3. Given an ambiguous CFG, how can we tell there exists a non-ambiguous CFG accepting the same language?
4. How can we tell whether the complement of a CFG is also context-free?
5. How can we tell whether the intersection of two CFGs is also context-free?
6. Given two CFGs, how can we tell whether they have a word in common?
7. Given a CFG, how can we tell whether there are any words it *does not* generate?

Which of these questions are *decidable*?

Decidability

None of the prior questions are *decidable*!

There are no algorithms to answer any of these *for any CFG*

What Exists

- What is known
- What will be known
- What might have been known but nobody will ever care enough to figure it out

What Does Not Exist

- Married bachelors
- Algorithms for Questions 1-7
- A good 5-cent cigar
- A funny joke from Professor Killian

So what questions can we answer about Context-Free Grammars?

Three Fundamental Questions We Can Answer

1. **Emptiness**
   Given a CFG, can we tell whether or not it generates any words at all?
2. **Finiteness**
   Given a CFG, can we tell whether or not the language it generates is finite of infinite?
3. **Membership**
   Given a CFG and a particular string of characters w, can we tell whether or not w can be generated by the CFG?

Emptiness

Theorem

Given any CFG, there is an algorithm to determine whether or not it can generate ***any*** *words at all.*

Proof.

- We can already show whether or not Λ can be produced
- Convert the grammar to CNF
- If there is a production of the form $S \longrightarrow t$, then t is a word in the language
- If there are no such productions, then we propose the following:

Step 1 For each Nonterminal N that has productions of the form $N \longrightarrow t$ where t is a terminal **or** string of terminals, replace N with t across all productions

Step 2 Repeat Step 1 until either S is eliminated or no new terminals are eliminated. If S has been eliminated, then the CFG produces some words; if not, then it does not.

Emptiness

Proof (Continued).

- The algorithm is finite since we will at most run Step 1 for every unique non-terminal in the original CNF form of the grammar.
- The string of nonterminals that will eventually replace S is a word that could be generated by the CFG.
- Some sequence of these "backwards replacements" (Step 1) will eventually reach back to S if there is **any** word in the language.

□

Example

$S \rightarrow XY$	$S \rightarrow XY$
$X \rightarrow AX \mid AA$	$X \rightarrow AX$
$Y \rightarrow BY \mid BB$	$Y \rightarrow BY \mid BB$
$A \rightarrow a$	$A \rightarrow a$
$B \rightarrow b$	$B \rightarrow b$

Usage of a Nonterminal Production (Uselessness)

Theorem

There is an algorithm to decide whether or not a given nonterminal X in a CFG is ever used in the generation of words.

A Clever Trick

Just for a minute, reverse S and X in all the production rules in the grammar. Use the "emptiness" algorithm to see whether we can derive a working string involving X that leads to a word.

Definition

A nonterminal that *cannot* ever produce a string of terminals is **unproductive**

Usage of a Nonterminal Production (Uselessness)

Algorithm (Deciding if X is Useless)

1. Find all nonproductive nonterminals
2. Purify the grammar by eliminating all productions from Step 1
3. Paint all X's blue
4. If any nonterminal is the left side of a production with **anything** blue on the right hand side, paint it (and any occurrences) blue
5. Repeat Step 4 until nothing blue is painted
6. If S is blue, then X is a useful member of the CFG. If not, X is useless

Example

$$S \to ABa \mid bAZ \mid b$$
$$A \to Xb \mid bZa$$
$$B \to bAA$$
$$X \to aZa \mid aaa$$
$$Z \to ZAbA$$

Finiteness

Theorem

There is an algorithm to decide whether a given CFG generates an infinite language or a finite language

Proof.

- There exists a procedure (next slide)
- If any word in the language is long enough to apply the pumping lemma to, we can produce an infinite sequence
- If the language is infinite, then the pumping lemma must be applicable
- We must find a self-embedded nonterminal X in our algorithm

□

Finiteness

Algorithm

1. Use the “usefulness” algorithm to determine which nonterminals are useless. Eliminate all productions involving them
2. Use the following algorithm to test each of the remaining nonterminals, in term, to see whether they are self-embedded. When a self-embedded one is discovered, stop. To test X:
   1. Change all X’s on the left side of productions into the Russian letter Ж, but leave all X’s on the right hand side of productions alone.
   2. Paint all X’s blue.
   3. If Y is any nonterminal that is the left side of any production with **some** blue on the right, paint all Y’s blue.
   4. Repeat step 2(iii) until nothing new is painted blue
   5. If Ж is blue, then X is self-embedded; if not, then it is not.
3. If any nonterminal left in the grammar after step 1 is self-embedded, the language generated is infinite. If not, then the language is finite.

Finiteness

Example

$$
\begin{aligned}
S &\rightarrow ABz \mid bAZ \mid b \\
A &\rightarrow Xb \mid bZA \\
B &\rightarrow bAA \\
X &\rightarrow aZa \mid bA \mid aaa \\
Z &\rightarrow ZAbA
\end{aligned}
$$

Membership

Theorem

Given a CFG and a string x in the same alphabet, we can decide whether or not x can be generated by the CFG.

- Strategy created by Cocke, Kasami, and Younger (CKY)
- Out of scope for this class (Compilers)

Homework 10a

1. Decide whether or not the following grammars generate any words. Show work! (2 points each)

   i.
   $$S \longrightarrow aSa \mid bSb$$

   ii.
   $$S \longrightarrow XY \mid SY$$
   $$X \longrightarrow SY \mid a$$
   $$Y \longrightarrow SX \mid b$$

   iii.
   $$S \longrightarrow AB$$
   $$A \longrightarrow BC \mid b$$
   $$B \longrightarrow CD$$
   $$C \longrightarrow DA$$
   $$D \longrightarrow a$$

   iv.
   $$S \longrightarrow XS$$
   $$X \longrightarrow YX \mid a$$
   $$Y \longrightarrow YY \mid XX$$

   v.
   $$S \longrightarrow AB$$
   $$A \longrightarrow BSB \mid CC \mid a \mid b$$
   $$B \longrightarrow AAS \mid CC$$
   $$C \longrightarrow SS \mid b \mid bb$$

Homework 10a

2. Decide whether or not the following grammars generate finite or infinite languages. Show work! (2 points each)

   i.
   $$\begin{aligned} S &\rightarrow XS \mid b \\ X &\rightarrow YZ \\ Y &\rightarrow ab \\ Z &\rightarrow XY \end{aligned}$$

   ii.
   $$\begin{aligned} S &\rightarrow XY \mid bb \\ X &\rightarrow YX \\ Y &\rightarrow XY \mid SS \end{aligned}$$

   iii.
   $$\begin{aligned} S &\rightarrow XY \\ X &\rightarrow AA \mid YY \mid b \end{aligned}$$

   iv.
   $$\begin{aligned} S &\rightarrow XY \\ X &\rightarrow AA \mid XY \mid b \\ A &\rightarrow BC \\ B &\rightarrow AC \\ C &\rightarrow BA \\ Y &\rightarrow a \end{aligned}$$

   v.
   $$\begin{aligned} S &\rightarrow SS \mid b \\ X &\rightarrow SS \mid SX \mid a \end{aligned}$$