Chapter 18: Decidability *

Peter Cappello  
Department of Computer Science  
University of California, Santa Barbara  
Santa Barbara, CA 93106  
cappello@cs.ucsb.edu

• Please read the corresponding chapter before attending this lecture.
• These notes are not intended to be complete. They are supplemented with figures, and material that arises during the lecture period in response to questions.

*Based on Theory of Computing, 2nd Ed., D. Cohen, John Wiley & Sons, Inc.
EMPTINESS & USELESSNESS

THEOREM

Let \( G \) be a CFG. “Is \( L(G) = \emptyset \)” is decidable.

PROOF

1. We give an algorithm to answer this question.

2. \( \Lambda \in L(G) \iff S \xrightarrow{*} \Lambda \).

3. The proof of Theorem 23 of Chapter 13 shows how to decide if a nonterminal, \( N \), is nullable: \( N \xrightarrow{*} \Lambda \).

4. Therefore, we know how to decide if \( \Lambda \in L(G) \).

5. We henceforth assume without loss of generality that \( \Lambda \notin L(G) \).
6. Assume without loss of generality that $G$ is in CNF.

7. Execute the following procedure:

   (a) For each nonterminal $N$ with a production of the form $N \rightarrow t$, for $t \in T^+$, for terminal set $T$:
       i. replace occurrences of $N$ on the RHS of all productions with $t$
       ii. remove $N$ and all $N$-productions from the grammar.

   (b) Repeat the step above until either:
       • $S$ is eliminated, or
       • there are no nonterminals with a production of the form $N \rightarrow t$, for $t \in T^+$, for terminal set $T$.

8. The algorithm terminates after finitely many steps (no more than the number of nonterminals in the $G$).

9. If $S$ is eliminated by the procedure, then $L(G) \neq \emptyset$. 
(When $S$ was eliminated, there was a production of the form $S \rightarrow t$, for $t \in T^*$. That $t \in L(G)$.)

10. If $L(G) \neq \emptyset$, then $S$ is eliminated by the procedure.

(a) If $L(G) \neq \emptyset$, then there is some $w \in L(G)$.

(b) Consider a parse tree for $w$.

(c) Its parse tree shows how the above procedure eliminates $S$.

   Illustrate.
Example

• Consider this CFG:

\[
\begin{align*}
S & \rightarrow XY \\
X & \rightarrow AX \mid AA \\
Y & \rightarrow BY \mid BB \\
A & \rightarrow a \\
B & \rightarrow b
\end{align*}
\]

• Step 1 yields:

\[
\begin{align*}
S & \rightarrow XY \\
X & \rightarrow aX \mid aa \\
Y & \rightarrow bY \mid bb
\end{align*}
\]

• Applying Step 1 to the resulting set of productions yields:
$S \rightarrow aabb$

- $S$ is eliminated during the next iteration.
  Thus, this grammar generates a non-empty language.
  In particular, it generates $aabb$. 
EXAMPLE

• Consider this CFG:

\[ S \rightarrow XY \]
\[ X \rightarrow AX \]
\[ Y \rightarrow BY \mid BB \]
\[ A \rightarrow a \]
\[ B \rightarrow b \]

• Step 1 yields:

\[ S \rightarrow XY \]
\[ X \rightarrow aX \]
\[ Y \rightarrow bY \mid bb \]

• Applying step 1 again yields:
\[ S \rightarrow Xbb \]
\[ X \rightarrow aX \]

- At this point, no nonterminal has a production of the form \( N \rightarrow t \), for some terminal \( t \in T^+ \).

The algorithm terminates with the answer “No.”
**Theorem**

Let $G$ be a CFG, and $N$ be a nonterminal in $G$. “Is $N$ used to generate any $w \in L(G)$?” is decidable.

**Proof**

1. A nonterminal that cannot generate a string of terminals is **unproductive**.

2. Given a nonterminal, $N$, decide if it is unproductive as follows:
   
   (a) Apply the algorithm of the previous theorem.
   
   (b) If $N$ is eliminated by this algorithm, then $N$ is productive.

3. Use the following algorithm to decide if a $N$ is used to generate some $w \in L(G)$:
   
   (a) Find all unproductive nonterminals.
(b) Eliminate all productions involving unproductive nonterminals (i.e., on either the RHS or the LHS).
(c) Paint all $N$’s blue.
(d) If any nonterminal, $M$, on the LHS has a production that has any blue on the RHS, paint all occurrences of $M$ blue.

```
Explain any.
```
(e) Repeat the above step until nothing new is painted.
(f) Return “Yes” if $S$ is blue; else return “No”.

4. The algorithm terminates after finitely many steps, since each iteration paints at least 1 nonterminal blue.

5. If $N$ is used in the generation of some $w \in L(G)$, then $S$ is painted blue:
(a) If $N$ is used in the generation of some $w \in L(G)$, then let $S \Rightarrow w$ be the derivation.
(b) In the parse tree for this derivation, there is a path from the root \((S)\) to a \(N\) node.

(c) Going up from the \(N\) node, we have a sequence productions in which a nonterminal is painted blue.

(d) This sequence of productions culminates in \(S\) being painted blue.

6. If \(S\) is painted blue, then \(N\) is used in the generation of some \(w \in L(G)\):

(a) Since \(S\) is blue, there is a sequence of \(j\) productions that made it blue:

\[
N_1 \rightarrow (V + T)^* N (V + T)^*
\]

\[
N_2 \rightarrow (V + T)^* N_1 (V + T)^*
\]

and so on until

\[
S \rightarrow (V + T)^* N_J (V + T)^*
\]
(b) Thus, there is a derivation $S \Rightarrow (V + T)^* N_J (V + T)^* \Rightarrow (V + T)^* N_{J-1} (V + T)^* \Rightarrow \cdots \Rightarrow N_1 \rightarrow (V + T)^* N (V + T)^*$.

(c) Since $N$ is productive, $S \Rightarrow^* w$, for some $w \in L(G)$.

If a nonterminal is not useful, it is **useless**.

A corollary of the previous theorem is that the question “Is nonterminal $N$ useless?” is decidable.
Example

• Is $X$ useful in the CFG below?

\[
S \rightarrow ABa \mid bAZ \mid b \\
A \rightarrow Xb \mid bZa \\
B \rightarrow bAA \\
X \rightarrow aZa \mid aaa \\
Z \rightarrow ZAbA
\]

• $Z$ is unproductive.

• After removing unproductive productions, we have:

\[
S \rightarrow ABa \mid b \\
A \rightarrow Xb \\
B \rightarrow bAA
\]
\[ X \rightarrow aaa \]

- Paint A blue. Therefore, paint S blue.

Therefore, answer “Yes.”
Theorem

Let $G$ be a CFG. The question “Is $L(G)$ finite?” is decidable.

Proof

1. $L(G)$ is infinite $\iff$ there is a $w \in L(G)$ that can be pumped (producing infinitely many other words in $L(G)$).

2. There is a $w \in L(G)$ that can be pumped, if there is a self-embedded nonterminal that is not useless.

3. Use the algorithm of the previous theorem to identify all useless nonterminals in $G$.

4. Remove all productions that involve useless nonterminals from $G$.

5. Use the following algorithm to decide if any remaining nonterminal is self-embedded:
For each nonterminal $N$, do:

(a) Change all occurrences of $N$ on the LHS of productions to the letter $\aleph$.
(b) Paint all occurrences of $N$ on the RHS of productions blue.
(c) If $Y$ is the LHS of a production with some blue on the RHS, paint all occurrences of $Y$ blue.
(d) Repeat the above step until nothing new is painted blue.
(e) If $\aleph$ is blue, $N$ is self-embedded.

6. A nonterminal is self-embedded $\iff L(G)$ is infinite.

The algorithm is finite and works for the same reasons that the algorithm of the previous theorem is finite and works.
Example

• Does the grammar below generate an infinite language?

\[
S \rightarrow ABa \mid bAZ \mid b \\
A \rightarrow Xb \mid bZA \\
B \rightarrow bAA \\
X \rightarrow aZa \mid bA \mid aaa \\
Z \rightarrow ZAbA
\]

• Z is unproductive, hence useless.

• After removing productions involving the useless Z, we have:

\[
S \rightarrow ABa \mid b \\
A \rightarrow Xb \\
B \rightarrow bAA
\]
\[ X \to bA \mid aaa \]

- We search for a self-embedded nonterminal, starting with \( X \).

- Change \( X \) on the LHS to \( \aleph \):
  
  \[
  S \to ABa \mid b \\
  A \to Xb \\
  B \to bAA \\
  \aleph \to bA \mid aaa
  \]

- Paint \( X \) blue.
  
  Therefore, paint \( A \) blue.
  
  Therefore, paint \( \aleph \) blue.
  
  Therefore, \( X \) is self-embedded.
  
  Therefore, \( L(G) \) is infinite.
Theorem

Let $G$ be a CFG and $w$ a string of $G$’s terminals. The question “Is $w \in L(G)$?” is decidable.

Proof

1. Let $w = w_1w_2 \cdots w_n$.

2. WLOG, assume $G$ is in CNF.

3. For each $w_i$, create a list of nonterminals that can generate it.
   That is, $N$ is on $w_i$’s list, if there is a production of the form $N \rightarrow w_i$. 
4. For $2 \leq i \leq n$ do:

   (a) For each substring of length $i$ do:

      i. For $1 \leq j < i$ do:

         A. break the substring into 2 pieces, the 1st of length $j$, the
         2nd of length $i - j$;

         B. Check to see if there is a production of the form $N_k \rightarrow N_lN_m$
         such that:

         • $N_l$ generates the 1st subsubstring
         • $N_m$ generates the 2nd subsubstring

         C. If so, add $N_k$ to the list of nonterminals that can generate
            this substring;

5. If $S$ is on the list of nonterminals that can generate $w_1w_2 \cdots w_n$, then
   $w \in L(G)$. 
Example

- Is $baaba \in L(G)$, where $G$ is given by the following set of productions:

\[
\begin{align*}
S & \rightarrow AB \mid BC \\
A & \rightarrow BA \mid a \\
B & \rightarrow CC \mid b \\
C & \rightarrow AB \mid a
\end{align*}
\]
• The following table is the result of applying the algorithm.

Row $i$ concerns substrings of length $i$.

Column $j$ concerns substrings that with $w_j$.

The entry in row $i$, column $j$ is the set of nonterminals that can generate the substring of length $i$, that starts with $w_j$.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>${S, A, C}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>${S, A, C}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>${B}$</td>
<td>${B}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>${S, A}$</td>
<td>${B}$</td>
<td>${S, C}$</td>
<td>${S, A}$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>${B}$</td>
<td>${A, C}$</td>
<td>${A, C}$</td>
<td>${B}$</td>
<td>${A, C}$</td>
</tr>
</tbody>
</table>

• Since $S$ is in row 5 column 1, $baaba \in L(G)$.