Computability Theory
Decidable Problems of CFLs and beyond

Haniel Barbosa

The University of Iowa
Readings for this lecture

Chapter 4 of [Sipser 1996], 3rd edition. Section 4.1.
Decidable Problems of CFLs

We now describe algorithms to test
- whether a CFG generates a particular string
- whether the language generated by a CFG is empty

We consider the language

\[ A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates the string } w \} \]
Decidability of CFG generation

Theorem

$A_{CFG}$ is a decidable language

Proof ideas: how to check $G$ generates $w$?

► Possibly infinitely many derivations in $G$, therefore no exhaustive check
Decidability of CFG generation

**Theorem**

\[ A_{\text{CFG}} \] is a decidable language

**Proof ideas: how to check \( G \) generates \( w \)?**

- Possibly infinitely many derivations in \( G \), therefore no exhaustive check
- However, if \( G \) is in Chomsky normal form then for any \( w \in \mathcal{L}(G) \), with \( |w| = n, n > 0 \), exactly \( 2n - 1 \) steps are required for any derivation of \( w \).
Decidability of CFG generation

Theorem

$A_{\text{CFG}}$ is a decidable language

Proof ideas: how to check $G$ generates $w$?

- Possibly infinitely many derivations in $G$, therefore no exhaustive check
- However, if $G$ is in Chomsky normal form then for any $w \in \mathcal{L}(G)$, with $|w| = n, n > 0$, exactly $2n - 1$ steps are required for any derivation of $w$.
- We can then build a TM $S$ that first converts $G$ into Chomsky Normal Form and then checks every possible derivation of $2n - 1$ steps.
Decidability of CFG generation

Theorem

$A_{CFG}$ is a decidable language

Proof ideas: how to check $G$ generates $w$?

- Possibly infinitely many derivations in $G$, therefore no exhaustive check
- However, if $G$ is in Chomsky normal form then for any $w \in \mathcal{L}(G)$, with $|w| = n, n > 0$, exactly $2n - 1$ steps are required for any derivation of $w$.
- We can then build a TM $S$ that first converts $G$ into Chomsky Normal Form and then checks every possible derivation of $2n - 1$ steps

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

1. Convert $G$ to an equivalent grammar in Chomsky Normal Form
2. List all derivations with $2n - 1$ steps, for $|w| = n$, if $n > 0$, otherwise list all derivations of 1 step
3. If any of these derivations generates $w$, accept, otherwise reject.”
Decidability of emptiness testing for CFGs

**Theorem**

\[ E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } \mathcal{L}(G) = \emptyset \} \text{ is a decidable language.} \]
Decidability of emptiness testing for CFGs

Theorem

\[ E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } \mathcal{L}(G) = \emptyset \} \text{ is a decidable language.} \]

Proof ideas: check whether string of terminals derivable from start

- We proceed by solving a more general problem: whether from a nonterminal a string of terminals is derivable.
- Algorithm proceeds bottom up by marking terminals, then all nonterminals with rules whose rhs consists only of marked symbols
Decidability of emptiness testing for CFGs

Theorem

\[ E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } \mathcal{L}(G) = \emptyset \} \text{ is a decidable language.} \]

Proof ideas: check whether string of terminals derivable from start

- We proceed by solving a more general problem: whether from a nonterminal a string of terminals is derivable.
- Algorithm proceeds bottom up by marking terminals, then all nonterminals with rules whose rhs consists only of marked symbols.

\[ R = \text{“On input string } \langle G \rangle, \text{ where } G \text{ is a CFG:} \]

1. Mark all terminal symbols in \( G \)
2. Repeat until no new nonterminal gets marked:
   3. Mark any nonterminal \( A \) s.t. \( G \) contains a rule \( A \to u_1, \ldots, u_n \) and each symbol \( u_i \) has already been marked
   4. If the start nonterminal of \( G \) is marked, accept, otherwise reject.”
Decidability of equality problem for CFGs

Consider the language

$$E_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$$

Could we proceed to prove its decidability as we did with $E_{\text{DFA}}$? NO!

CFGs are not closed under intersection or complement, therefore we cannot use symmetric difference. And as we will see later on, $E_{\text{CFG}}$ is indeed undecidable.
Decidability of equality problem for CFGs

Consider the language

\[ EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } \mathcal{L}(G) = \mathcal{L}(H) \} \]

Could we proceed to prove its decidability as we did with \( EQ_{\text{DFA}} \)? \ NO! \n
- CFGs are not closed under intersection or complement, therefore we cannot use symmetric difference.
- And as we will see later on, \( EQ_{\text{CFG}} \) is indeed **undecidable**.
Theorem

Every CFL is decidable.

Proof idea: simulate generation of string by respective CFG

Let $G$ be a CFG and let $L(G) = A$. Design a TM $M_G$ that decides $A$ by building a copy of $G$ into $M_G$:

1. Run TM $S$ on input $⟨G, w⟩$
2. If this machine accepts, accept, otherwise reject.
Decidability of context-free languages

Theorem

Every CFL is decidable.

Proof idea: simulate generation of string by respective CFG

Let $G$ be a CFG and let $L(G) = A$. Design a TM $M_G$ that decides $A$ by building a copy of $G$ into $M_G$:
Decidability of context-free languages

Theorem

*Every CFL is decidable.*

Proof idea: simulate generation of string by respective CFG

Let $G$ be a CFG and let $\mathcal{L}(G) = A$. Design a TM $M_G$ that decides $A$ by building a copy of $G$ into $M_G$:

$$M_G = \text{"On input string } w, \text{ where } G \text{ is a CFG:}$$

1. Run TM $S$ on input $\langle G, w \rangle$
2. If this machine accepts, *accept*, otherwise *reject.*
Methodology for deciding languages relationships

To solve decidability problems concerning relations between languages ones should proceed as follows:

🔹 Understand the relationship

🔹 Transform the relationship into an expression using closure operators on decidable languages

🔹 Designing a TM that constructs the language thus expressed

🔹 Run a TM that decides the language represented by the expression
Example

Consider checking the equivalence of DFAs and Regular Expressions.

We express this problem as the language

\[
EQ_{DFA,REX} = \{ \langle A, R \rangle \mid A \text{ is a DFA, } R \text{ is a regex, and } \mathcal{L}(A) = \mathcal{L}(R) \}
\]

Since regexs can be reduced into DFAs and DFAs are closed under symmetric difference, we can proceed similarly as we did for proving the decidability of \( EQ_{DFA} \)
Example

Consider checking the equivalence of DFAs and Regular Expressions.

We express this problem as the language

\[ EQ_{DFA,REX} = \{ \langle A, R \rangle \mid A \text{ is a DFA, } R \text{ is a regex, and } L(A) = L(R) \} \]

Since regexs can be reduced into DFAs and DFAs are closed under symmetric difference, we can proceed similarly as we did for proving the decidability of \( EQ_{DFA} \).

E = “On input string \( \langle A, R \rangle \), where \( A \) is a DFA and \( R \) is a regex:

1. Convert \( R \) to an equivalent DFA \( B \)
2. Use the TM \( C \), for deciding \( EQ_{DFA} \), on input \( \langle A, B \rangle \)
3. If \( C \) accepts, accept, otherwise reject.”
Another example

Is the problem of testing whether a CFG generates some string $1^*$ decidable?
Another example

Is the problem of testing whether a CFG generates some string $1^*$ decidable?

We can formulate this problem as showing the decidability of

$$A = \{ \langle G \rangle \mid G \text{ is a CFG over } \{0, 1\}^* \text{ and } \mathcal{L}(1^*) \cap \mathcal{L}(G) \neq \emptyset \}$$
Another example

Is the problem of testing whether a CFG generates some string $1^*$ decidable?

- We can formulate this problem as showing the decidability of

$$A = \{ \langle G \rangle \mid G \text{ is a CFG over } \{0, 1\}^* \text{ and } \mathcal{L}(1^*) \cap \mathcal{L}(G) \neq \emptyset \}$$

- We know that $1^*$ defines a regular language, that $\mathcal{L}(G)$ is a CFL, and that $\mathcal{L}(1^*) \cap \mathcal{L}(G)$ is a CFL.

- We know how to check emptiness of CFLs!
Another example

Is the problem of testing whether a CFG generates some string $1^*$ decidable?

▷ We can formulate this problem as showing the decidability of

$$ A = \{ \langle G \rangle \mid G \text{ is a CFG over } \{0, 1\}^* \text{ and } \mathcal{L}(1^*) \cap \mathcal{L}(G) \neq \emptyset \} $$

▷ We know that $1^*$ defines a regular language, that $\mathcal{L}(G)$ is a CFL, and that $\mathcal{L}(1^*) \cap \mathcal{L}(G)$ is a CFL.

▷ We know how to check emptiness of CFLs!

$X =$“On input string $\langle G \rangle$, where $G$ is a CFG:

1. Construct CFG $H$ such that $\mathcal{L}(H) = \mathcal{L}(1^*) \cap \mathcal{L}(G)$

2. Run the TM $R$, which decides the language $E_{\text{CFG}}$, on input $\langle H \rangle$

3. If $R$ accepts, reject, otherwise accept.”
Yet another example

Is the language generated by particular regular expressions decidable? For example, is the language of regular expressions that contain at least one string that has the pattern 111 as a substring decidable?
Yet another example

Is the language generated by particular regular expressions decidable? For example, is the language of regular expressions that contain at least one string that has the pattern 111 as a substring decidable?

▷ We can formulate this problem as showing the decidability of

\[ A = \{ \langle R \rangle \mid R \text{ is a regex for a language with at least one string } w = x111y \} \]

▷ To decide \( A \) we need to be able determine whether for any regex \( R \) it is the case that \( \mathcal{L}(R) \) contains some string \( x111y \).
Yet another example

Is the language generated by particular regular expressions decidable? For example, is the language of regular expressions that contain at least one string that has the pattern 111 as a substring decidable?

▷ We can formulate this problem as showing the decidability of

$$A = \{ \langle R \rangle \mid R \text{ is a regex for a language with at least one string } w = x111y \}$$

▷ To decide $A$ we need to be able determine whether for any regex $R$ it is the case that $L(R)$ contains some string $x111y$.

▷ I.e. whether $L(\Sigma^*111\Sigma^*) \cap L(R) \neq \emptyset$. 
Yet another example

Is the language generated by particular regular expressions decidable? For example, is the language of regular expressions that contain at least one string that has the pattern 111 as a substring decidable?

▷ We can formulate this problem as showing the decidability of

\[ A = \{ \langle R \rangle \mid R \text{ is a regex for a language with at least one string } w = x111y \} \]

▷ To decide \( A \) we need to be able determine whether for any regex \( R \) it is the case that \( \mathcal{L}(R) \) contains some string \( x111y \).

▷ I.e. whether \( \mathcal{L}(\Sigma^*111\Sigma^*) \cap \mathcal{L}(R) \neq \emptyset \).

\[
Z = \text{“On input string } \langle R \rangle \text{, where } R \text{ is a regex:} \\
1. \text{Construct the DFA } E \text{ that accepts } \mathcal{L}(\Sigma^*111\Sigma^*) \\
2. \text{Construct the DFA } B \text{ that accepts } \mathcal{L}(E) \cap \mathcal{L}(R) \\
2. \text{Run the TM } T, \text{ which decides } E_{\text{DFA}}, \text{ on input } \langle B \rangle \\
3. \text{If } T \text{ accepts, reject, otherwise accept.”}
\]