UNDECIDABLE PROBLEMS ABOUT CONTEXT-FREE LANGUAGES

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

March 14, 2021

R.E. Languages vs Context-Free Languages

- R.E. languages are specified by Turing machines M or unrestricted grammars.
- We have seen that the following problems are undecidable.
  - whether $\mathscr{L}(M) = \emptyset$.
  - whether $\mathscr{L}(M)$ is finite.
  - whether $\mathscr{L}(M) = \Sigma^*$.
  - whether $\mathscr{L}(M)$ is recursive.
- CFLs are specified by PDA or CFGs. We ask similar questions for a CFG G.
  - whether $\mathscr{L}(G) = \emptyset$.
  - whether $\mathscr{L}(G)$ is finite.
  - whether $\mathscr{L}(G) = \Sigma^*$.
  - whether $\mathscr{L}(G)$ is a DCFL.
- Some of these CFL problems are decidable, some are not.

CFL Emptiness is Decidable

Theorem

It is decidable whether a context-free grammar G generates (or a PDA N accepts) any strings at all, that is, whether $\mathscr{L}(G) = \emptyset$ (or $\mathscr{L}(N) = \emptyset$) or not.

Proof by Pumping Lemma

- Let $L = \mathscr{L}(G)$.
- Let n be the number of non-terminal symbols of G.
- Then $k = 2^{n+1}$ is a pumping-lemma constant for L.
- The pumping lemma implies:
  - If L is finite, then all strings in L are of length $< k$.
  - If L is infinite, then L contains a string of length in the range $[k, 2k)$.
- Check whether G can generate any string of length $< 2k$.
- Each such string can be tested in finite time.
- This also establishes that the finiteness problem for CFLs is decidable.

An Efficient Procedure

- Try to mark all symbols in $\Sigma \cup N$.
- Start by marking symbols in Σ.
- Look at the rules $A \to \beta$.
- If all the symbols of β are marked, mark A.
- Continue until no further markings are possible.
- $|N|$ is finite.
- The procedure halts after a finite number of steps.
- L is non-empty if and only if the start symbol S is marked.
- This procedure is very efficient.

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

Example

$$
\begin{aligned}
S &\rightarrow ABC \mid UVS \\
A &\rightarrow aA \mid bU \mid cVW \\
B &\rightarrow caW \mid WW \\
C &\rightarrow cc \\
U &\rightarrow W \mid UWU \mid aBV \\
V &\rightarrow VC \mid WV \\
W &\rightarrow \varepsilon \mid AbcV
\end{aligned}
$$

CFL Fullness is Undecidable

Theorem

It is undecidable whether a context-free grammar G generates (or a PDA N accepts) all strings, that is, whether $\mathscr{L}(G) = \Sigma^*$ (or $\mathscr{L}(N) = \Sigma^*$) or not.

COMPUTATION-HISTORY METHOD

Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

March 14, 2021

CFL Fullness is Undecidable

Theorem

It is undecidable whether a context-free grammar G generates (or a PDA N accepts) all strings, that is, whether $\mathscr{L}(G) = \Sigma^*$ (or $\mathscr{L}(N) = \Sigma^*$) or not.

Reduction Idea

- $\overline{\mathrm{HP}} \leqslant_m \left\{ G \;\middle|\; G \text{ is a CFG over } \Delta \text{ with } \mathscr{L}(G) = \Delta^* \right\}$.
- Input: A Turing machine M and an input w for M.
- Output: A context-free grammar G.
- M does not halt on $w \Rightarrow \mathscr{L}(G) = \Delta^*$.
- M halts on $w \Rightarrow \mathscr{L}(G) \subsetneqq \Delta^*$.
- G incorporates the computation histories of M on w.
- If M halts on w, it has one or more finite computation histories.
- If M does not halt on w, it has only infinite computation histories.
- Infinite computation histories cannot be encoded as strings.
- Only finite computation histories ending in a halting configuration are called *valid*.

Encoding Configurations of M

- Σ is the input alphabet for M.
- Γ is the tape alphabet for M.
- Q is the set of states of M.
- A configuration of M is encoded as a string over $\Gamma \times (Q \cup \{-\})$.
- For the configuration

$$C = (p, \triangleright aubva\underline{c}bvwua\square vc\square^{\omega}),$$

the encoding is:

$$\begin{array}{ccccccccccccccc} \triangleright & a & u & b & v & a & c & b & v & w & u & a & \square & v & c \\ - & - & - & - & - & - & p & - & - & - & - & - & - & - & - \end{array}$$

- The initial configuration C_0 on input $w = a_1a_2a_3\ldots a_n$ is

$$\begin{array}{cccccc} \triangleright & a_1 & a_2 & a_3 & \cdots & a_n \\ s & - & - & - & \cdots & - \end{array}$$

Encoding Computation Histories

- A computation history is a sequence of configurations $C_0, C_1, C_2, \ldots, C_N$.
- Each $C_i \in (\Gamma \times (Q \cup \{-\}))^*$.
- Each C_i must contain only one state.
- C_0 should be the initial configuration.
- Two consecutive configurations must be consistent with the transition function of M.
- The last configuration should have the accept state t or the reject state r.
- We encode this history as

$$\#C_0\#C_1\#C_2\#C_3\#\cdots\#C_N\#.$$

- We allow t or r to appear in C_i for $i < N$. If so, the states in $C_i, C_{i+1}, C_{i+2}, \ldots, C_N$ must be the same.

Valid Computation Histories

- Let $\Delta = (\Gamma \times (Q \cup \{-\})) \cup \{\#\}$.
- Define

$$\mathrm{VALCOMP}(M,w) = \left\{\alpha \in \Delta^* \;\middle|\; \alpha \text{ is a valid computation history of } M \text{ on input } w\right\}.$$

- Let $L = \overline{\mathrm{VALCOMP}(M,w)} = \Delta^* \setminus \mathrm{VALCOMP}(M,w)$.

Theorem

L is a context-free language.

- A total Turing machine R, given M and w, can prepare a CFG for L.
- R does not have to simulate M on w.

Validity of this Reduction

If M halts on w

- M has one or more valid computation histories.
- $\mathrm{VALCOMP}(M, w) \neq \emptyset$.
- $L = \overline{\mathrm{VALCOMP}(M, w)} \neq \Delta^*$.

If M does not halt on w

- M has only infinite (so invalid) computation histories.
- $\mathrm{VALCOMP}(M, w) = \emptyset$.
- $L = \overline{\mathrm{VALCOMP}(M, w)} = \Delta^*$.

This is a valid reduction $\overline{\mathrm{HP}} \leqslant_m \left\{ G \;\middle|\; G \text{ is a CFG with } \mathscr{L}(G) = \Delta^* \right\}$.

VALCOMP(M,w) and its Complement L

- A string $\alpha \in \Delta^*$ is in VALCOMP(M,w) if and only if the following five conditions hold:
    1. α is of the form $\#C_0\#C_1\#C_2\#\cdots\#C_N\#$ with each $C_i \in (\Gamma \times (Q \cup \{-\}))^*$.
    2. Each C_i must contain only one state.
    3. C_0 must be the start configuration.
    4. C_N is a halting configuration (in state t or r).
    5. Each C_{i+1} follows from C_i by the transition rules of M.
- Let $A = \left\{\alpha \in \Delta^* \;\middle|\; \alpha \text{ satisfies Conditions } 1\text{–}4\right\}$.
- Let $B = \left\{\alpha \in \Delta^* \;\middle|\; \alpha \text{ satisfies Condition } 5\right\}$.
- We have $\mathrm{VALCOMP}(M,w) = A \cap B$.
- So $L = \overline{\mathrm{VALCOMP}(M,w)} = \overline{A} \cup \overline{B} = \overline{A} \cup (A \cap \overline{B})$.
- To show: $\overline{A}$ and $A \cap \overline{B}$ (or $\overline{B}$) are context-free.

A is Regular

- Let $w = a_1 a_2 \ldots a_n$.
- Notations:
  $\Delta_{-} = \Gamma \times \{-\}$, $\Delta_Q = \Gamma \times Q$, $\Delta_t = \Gamma \times \{t\}$, and $\Delta_r = \Gamma \times \{r\}$.
- Regular subexpressions:
  - $\beta_0 = \begin{array}{ccccc} \triangleright & a_1 & a_2 & \cdots & a_n \\ s & - & - & \cdots & - \end{array}.$
  - $\beta_i = \Delta_{-}^{*} \; \Delta_Q \; \Delta_{-}^{*}$.
  - $\beta_t = \Delta_{-}^{*} \; \Delta_t \; \Delta_{-}^{*}$.
  - $\beta_r = \Delta_{-}^{*} \; \Delta_r \; \Delta_{-}^{*}$.
- A is generated by the regular expression $\#\beta_0(\#\beta_i)^*\#(\beta_t + \beta_r)\#$.
- So A is regular, and $\overline{A}$ is regular too.
- $\overline{A}$ can be specified by a right-linear grammar.

$A \cap \overline{B}$ is Context-Free

- C_i and C_{i+1} are two consecutive configurations.
- We need to check C_{i+1} does *not* follow from C_i.
- Two positions in C_i and C_{i+1} are corresponding if they are equidistant from their preceding hashes.
- Change in configuration is only local.
- Changes in corresponding positions consistent with the transitions of M.
  - No change: $\begin{array}{ccc} a & b & c \\ \_ & \_ & \_ \end{array}$ remains as $\begin{array}{ccc} a & b & c \\ \_ & \_ & \_ \end{array}$.
  - State enters: $\begin{array}{ccc} a & b & c \\ \_ & \_ & \_ \end{array}$ changes to $\begin{array}{ccc} a & b & c \\ q & \_ & \_ \end{array}$ or $\begin{array}{ccc} a & b & c \\ \_ & \_ & q \end{array}$.
  - $\delta(p,b) = (q,d,L)$: $\begin{array}{ccc} a & b & c \\ \_ & p & \_ \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ q & \_ & \_ \end{array}$.
  - $\delta(p,b) = (q,d,R)$: $\begin{array}{ccc} a & b & c \\ \_ & p & \_ \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ \_ & \_ & q \end{array}$.

$A \cap \overline{B}$ is Context-Free

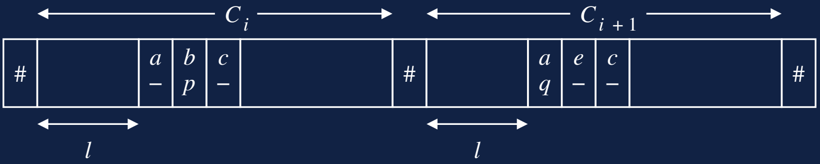
- Changes in corresponding positions not consistent with the transitions of M.
  - $\delta(p,b) = (q,d,L)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ q' & - & - \end{array}$.
  - $\delta(p,b) = (q,d,L)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & e & c \\ q & - & - \end{array}$.
  - $\delta(p,b) = (q,d,L)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ - & - & q \end{array}$.

  - $\delta(p,b) = (q,d,R)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ - & - & q' \end{array}$.
  - $\delta(p,b) = (q,d,R)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & e & c \\ - & - & q \end{array}$.
  - $\delta(p,b) = (q,d,R)$: $\begin{array}{ccc} a & b & c \\ - & p & - \end{array}$ changes to $\begin{array}{ccc} a & d & c \\ q & - & - \end{array}$.

  - The state in C_i is t or r, but that in C_{i+1} is not the same.

An NPDA to Detect an Inconsistent Change

- The NPDA non-deterministically chooses:
  - Two consecutive configurations, and
  - The corresponding positions.



- After reading the hash before C_i, the NPDA does these:
  - Push l symbols to its stack.
  - Read the three elements of Δ in its finite control.
  - Skip the rest of C_i and the next hash.
  - Move ahead exactly l positions in C_{i+1} by popping from its stack until the stack becomes empty (or some marker is exposed).
  - Read the next three symbols, and confirm inconsistency.

The Final Points about the Reduction

- $L = \overline{\text{VALCOMP}(M,w)} = \overline{A} \cup \overline{B} = \overline{A} \cup (A \cap \overline{B})$.
- If $\alpha \in \Delta^*$ is in $A \cap \overline{B}$, there is at least one inconsistency.
- The NPDA can nondeterministically find that, and accept α.
- $\overline{A}$ has a right-linear grammar.
- Convert the NPDA for $A \cap \overline{B}$ to a CFG.
- CFLs are closed under union.

Tutorial Exercises

1. You are given two CFGs G and G'. Prove that the following problems are undecidable.
   (a) whether $\mathscr{L}(G) = \mathscr{L}(G')$,
   (b) whether $\mathscr{L}(G) \subseteq \mathscr{L}(G')$,
   (c) whether $\mathscr{L}(G) = \mathscr{L}(G)\mathscr{L}(G)$.
2. Prove that the following problems are undecidable.
   (a) whether a CFL is a DCFL.
   (b) whether the intersection of two CFLs is a CFL.
   (c) whether the complement of a CFL is a CFL.
3. Prove that the finiteness problem for regular and context-free languages is decidable.