Problems about CFLs

Membership problem. Given a CFG G and a string x, is $x \in L(G)$?

Problems about CFLs

Membership problem. Given a CFG G and a string x, is $x \in L(G)$?

Answer. Decidable — CKY algorithm

Problems about CFLs

Membership problem. Given a CFG G and a string x, is $x \in L(G)$?

Answer. Decidable — CKY algorithm

Emptiness problem. Given a CFG G, is $L(G) = \varnothing$?

Answer. Decidable.

Problems about CFLs

Membership problem. Given a CFG G and a string x, is $x \in L(G)$?

Answer. Decidable — CKY algorithm

Emptiness problem. Given a CFG G, is $L(G) = \emptyset$?

Answer. Decidable.

Inductive marking procedure – Determine if a nonterminal generates some string.

1. Mark all terminal symbols in Σ.
2. Repeat till no change
   - Mark $A \in N$ if $\exists\ A \to \beta \in P$ s.t all symbols in β are marked

By Induction: A is marked iff $\exists\ x \in \Sigma^*$ s.t $A \xrightarrow[G]{*} x$.

$L(G)$ is non-empty iff S is marked.

Problems about CFLs

Membership problem. Given a CFG G and a string x, is $x \in L(G)$?

Answer. Decidable — CKY algorithm

Emptiness problem. Given a CFG G, is $L(G) = \emptyset$?

Answer. Decidable.

Universality problem. Given a CFG G, is $L(G) = \Sigma^*$?

Answer. Undecidable

Valid Computation Histories

Configurations of a Turing machine

A configuration of a Turing machine M is a triple (q, y, n) where

- q is a state,
- y describes the content of the tape,
- n an integer describing the head position.

Encoding configurations. We can encode configurations as finite strings over the alphabet $\Gamma \times (Q \cup \{-\})$.

Example. Configuration (q, y, k)

$$\begin{array}{ccccccccc} \vdash b_1 & b_2 & b_3 & \cdots & b_k & \cdots & b_m \\ - \ - & - & - & & q & \cdots & - \end{array}$$

Machine is scanning the k^{th} cell

Valid Computation Histories

Configurations of a Turing machine

A configuration of a Turing machine M is a triple (q, y, n) where

- q is a state,
- y describes the content of the tape,
- n an integer describing the head position.

Encoding configurations. We can encode configurations as finite strings over the alphabet $\Gamma \times (Q \cup \{-\})$.

Example. Configuration (q, y, k)

$$\begin{array}{cccccccc} \vdash & b_1 & b_2 & b_3 & \cdots & b_k & \cdots & b_m \\ - & - & - & - & & q & \cdots & - \end{array}$$

Machine is scanning the k^{th} cell

Start Configuration.

$$\begin{array}{ccccc} \vdash & a_1 & a_2 & \cdots & a_n \\ s & - & - & \cdots & - \end{array}$$

where $x = a_1 a_2 \cdots a_n$

Valid Computation Histories

Alphabet: $\Gamma \times (Q \cup \{-\})$.

A valid computation history of M on x is a string

$$\#\alpha_0\#\alpha_1\#\alpha_2\#\cdots\#\alpha_N\#$$

- α_0 is a start configuration of M on x,
- α_N is a halting configuration (state is either the accept state t or reject state r),
- α_{i+1} follows in one step from α_i according to δ of M. That is, for $0 \leq i \leq N-1$,

$$\alpha_i \xrightarrow[M]{1} \alpha_{i+1}.$$

Valid Computation Histories

Alphabet: $\Gamma \times (Q \cup \{-\})$.

A valid computation history of M on x is a string

$$\#\alpha_0\#\alpha_1\#\alpha_2\#\cdots\#\alpha_N\#$$

- α_0 is a start configuration of M on x,
- α_N is a halting configuration (state is either the accept state t or reject state r),
- α_{i+1} follows in one step from α_i according to δ of M. That is, for $0 \leq i \leq N-1$,

$$\alpha_i \xrightarrow[M]{1} \alpha_{i+1}.$$

Let $\Delta = \{\#\} \cup (\Gamma \times (Q \cup \{-\}))$, then

$$\text{VALCOMPS}(M,x) = \{\text{valid computation histories of } M \text{ on } x\} \subseteq \Delta^*.$$

$$\text{VALCOMPS}(M,x) = \varnothing \text{ iff } M \text{ does not halt on } x.$$

$\overline{\text{VALCOMPS}(M,x)} = \Delta^*$ iff M does not halt on x.

Valid Computation Histories

Claim 1. $\overline{\text{VALCOMPS}(M, x)}$ is a CFL.

Claim 2. We can construct a CFG G for $\overline{\text{VALCOMPS}(M, x)}$ from the description of M and x.

Observation. $L(G) = \Delta^*$ iff M does not halt on x.

Reduction. $\overline{\text{HP}} \leq_m \underbrace{\{G \mid G \text{ is a CFG and } L(G) = \Delta^*\}}_{\text{not r.e.}}.$

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,

$\forall i$, α_i is a string over $(\Delta - \#)^*$

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,
2. each α_i is a string of symbols of the form

   a   a

            or

   -                q

   where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,

2. each α_i is a string of symbols of the form

   a   a

            or

   \-   q

   where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,

3. α_0 represents the start configuration of M on x,

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,
2. each α_i is a string of symbols of the form

   a   a

          or

   -                q

   where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,
3. α_0 represents the start configuration of M on x,
4. a halt state, either t or r appears somewhere in z. That is, α_N is a halting configuration,

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,
2. each α_i is a string of symbols of the form

   a   a

       or

   \-   q

   where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,
3. α_0 represents the start configuration of M on x,
4. a halt state, either t or r appears somewhere in z. That is, α_N is a halting configuration,
5. $\alpha_i \xrightarrow[M]{1} \alpha_{i+1}$ for $0 \leq i \leq N-1$.

Conditions for a string $z \in \Delta^*$ to be a valid computation history of M on x:

1. z must begin and end with a $\#$. It must be of the form $\#\alpha_0\#\alpha_1\#\cdots\#\alpha_N\#$,
2. each α_i is a string of symbols of the form

   $\begin{matrix} a \\ - \end{matrix}$ or $\begin{matrix} a \\ q \end{matrix}$

   where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,
3. α_0 represents the start configuration of M on x,
4. a halt state, either t or r appears somewhere in z. That is, α_N is a halting configuration,
5. $\alpha_i \xrightarrow[M]{1} \alpha_{i+1}$ for $0 \leq i \leq N - 1$.

$$A_i = \{x \in \Delta^* \mid x \text{ satisfies conditions } (i)\}, \quad 1 \leq i \leq 5$$

$VALCOMPS(M,x) = \bigcap_{1 \leq i \leq 5} A_i$; $\overline{VALCOMPS(M,x)} = \bigcup_{1 \leq i \leq 5} \overline{A_i}$

Claim. Sets A_1, A_2, A_3, A_4 are regular sets.

↳ $\exists$ right linear CFGs.

- z must begin and end with a $\#$.

Claim. Sets A_1, A_2, A_3, A_4 are regular sets.

- z must begin and end with a $\#$.

  Observation. A_1 is the regular set $\#\Delta^*\#$.

- each α_i is a string of symbols of the form

$$\begin{array}{c} a \\ - \end{array} \quad \text{or} \quad \begin{array}{c} a \\ q \end{array}$$

where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,

- each α_i is a string of symbols of the form

  a   a

         or

  \-   q

  where exactly one symbol of α_i has an element of Q on the bottom and others have $-$, and only the leftmost has a $\vdash$ on top,

  Suffices to check that between every two $\#$'s there is exactly one symbol with state q on the bottom and $\vdash$ occurs on the top immediately after each $\#$ (except the last) and nowhere else.

  Observation. A_2 is the regular set.

- α_0 represents the start configuration of M on x,

  Observation. A_3 is the regular set

- α_0 represents the start configuration of M on x,

  Observation. A_3 is the regular set

- a halt state, either t or r appears somewhere in z. That is, α_N is a halting configuration,

  Observation. Suffices to check that t or r appears somewhere in the string.

Condition 5. $\alpha_i \xrightarrow[M]{1} \alpha_{i+1}$ for $0 \leq i \leq N - 1$.

Claim. $\overline{A_5}$ is a CFL.

Condition 5. $\alpha_i \xrightarrow[M]{1} \alpha_{i+1}$ for $0 \le i \le N-1$.

Note. α_i & α_{i+1} should agree on most symbols except a few near the current head position.

$$\alpha \xrightarrow[M]{1} \beta$$

$$\begin{array}{cccccccccccccccccccccccc}
\cdots & \# & \vdash & a & b & a & b & b & a & b & a & \# & \vdash & a & b & a & a & b & a & b & a & \# & \cdots \\
 & & - & - & - & - & q & - & - & - & - & & & - & - & - & p & - & - & - & - & - &
\end{array}$$

if $\delta(q,b) = (p,a,L)$

To check: $\alpha \xrightarrow[M]{1} \beta$: check all 3 element substring u of α and corresponding substring v of β

↳ occuring at same distance from #

$$\overbrace{\begin{array}{ccc} a & b & b \\ - & q & - \end{array}}^{\text{in } \alpha} \quad \text{and} \quad \overbrace{\begin{array}{ccc} a & a & b \\ p & - & - \end{array}}^{\text{in } \beta} \Bigg\} \text{ are consistent with } \delta \text{ since } \delta(q,a) = \delta(p,b,L).$$

The pair $\underbrace{\begin{array}{ccc} a & b & b \\ - & - & - \end{array}}_{\text{in } \alpha}$ and $\underbrace{\begin{array}{ccc} a & b & b \\ - & - & - \end{array}}_{\text{in } \beta}$

To check that $\alpha \xrightarrow[M]{1} \beta$ does not hold, need to check: $\exists$ a length 3 substring of α s.t the corresponding length 3 substring of β is not consistent with δ.

An NPDA M s.t $L(M) = \overline{A_5}$

M works as follows:

- Guess α_i non-deterministically.

- Guess length 3 substring u is α_i, check that the corresponding substring v α_{i+1} is not consistent.

- Identify the corresponding substring in v using the stack.
    - Push the prefix of α_i till u in the stack.
    - Pop the symbols in α_{i+1} after # to match the length of the prefix to find the corresponding substring v in α_{i+1}.