Decidable Problems for Context–Free Grammars

Martin Čermák, Jiří Koutný and Alexander Meduna

Deparment of Information Systems
Faculty of Informatin Technology
Brno University of Technology, Faculty of Information Technology
Božetěchova 2, Brno 612 00, Czech Republic

BRNO UNIVERSITY OF TECHNOLOGY

FIT

FACULTY OF INFORMATION TECHNOLOGY

Advanced Topics of Theoretical Computer Science

FRVŠ MŠMT FR2581/2010/G1

Convention

- ${}_{CNF-CF}\Psi$ denotes the set of all context–free grammars in Chomsky normal form.
- We supose there exists a fixed encoding and decoding of the grammars in ${}_{CNF-CF}\Psi$.
- $\langle G \rangle$ represents the code of $G \in {}_{CNF-CF}\Psi$.
- $\langle G, w \rangle$ denotes $(G, w) \in {}_{CNF-CF}\Psi \times \triangle^*$.
- $\langle G, H \rangle$ denotes $(G, H) \in {}_{CNF-CF}\Psi \times {}_{CNF-CF}\Psi$.

CF–Emptiness

Problem: *CF–Emptiness*
Question: Let $G \in {}_{CNF-CF}\Psi$. Is $L(G)$ empty?
Language: ${}_{CF\text{–}Emptiness}L = \{\langle G\rangle \mid G \in {}_{CNF-CF}\Psi, L(G) = \emptyset\}$.

Theorem

${}_{CF\text{–}Emptiness}L \in {}_{TD}\Phi$

Proof:

- Let $G \in {}_{CNF-CF}$.
- A symbol in G is terminating if it derives a string of terminals.
- $L(G)$ is non–empty iff ${}_GS$ is terminating, where ${}_GS$ denotes start symbol of G.
- Construct a Turing decider D that works on $\langle G\rangle$ in the following way:
  - D decides whether ${}_GS$ is terminating.
  - D rejects $\langle G\rangle$ if ${}_GS$ is terminating; otherwise, D accepts $\langle G\rangle$.

CF–Membership

Problem: *CF–Membership*
Question: Let $G \in {}_{CNF-CF}\Psi$ and $w \in \triangle^*$. Is w member of $L(G)$?
Language:
$_{CF\text{–}Membership}L = \{\langle G, w\rangle |\ G \in {}_{CNF-CF}\Psi, w \in \triangle^*, w \in L(G)\}$.

Lemma

Let $G \in {}_{CNF-CF}\Psi$. Then, G generates every $w \in L(G)$ by making no more than $2|w| - 1$ derivation steps.

Theorem

$_{CF\text{–}Membership}L \in {}_{TD}\Phi$.

Proof:

- From the Chomsky normal form, ${}_{CNF-CF}\Psi$ contains no grammar that generates ε.
- Construct the following Turing decider D that works on every $\langle G, w \rangle$ in either of the following two ways:
    - Let $w = \varepsilon$.
        - Clearly, $\varepsilon \in L(G)$ iff ${}_GS$ derives ε.
        - D decides whether ${}_GS$ dirives ε, and if so, D accepts $\langle G, w \rangle$; otherwise, D rejects $\langle G, w \rangle$.
    - Let $w \neq \varepsilon$. Then D works on $\langle G, w \rangle$ as follows:
        - D constructs the set of all sentences that G generates by making no more than $2|w| - 1$ derivation steps;
        - If the set contains w, D accepts $\langle G, w \rangle$; otherwise, D rejects $\langle G, w \rangle$.

CF–Infiniteness

Problem: *CF–Infiniteness*
Question: Let $G \in {}_{CNF-CF}\Psi$. Is $L(G)$ infinite?
Language: ${}_{CF\text{–}Infiniteness}L = \{\langle G \rangle |\ G \in {}_{CNF-CF}\Psi, L(G) \text{ is infinite}\}$.

Lemma

Let $G \in {}_{CNF-CF}$ be in the Chomsky normal form. $L(G)$ is infinite iff $L(G)$ contains a sentence x such that $k \leq |x| < 2k$ with $k = 2^{card({}_G N)}$.

Theorem

${}_{CF\text{–}Infiniteness}L \in {}_{TD}\Phi$

Proof:

- Construct a Turing decider D that works on every $\langle G, w \rangle$ as follows:
  - D constructs the set of all sentences in G such that $k \leq |x| < 2k$ with $k - 2^{card({}_G N)}$;
  - If this set contains w, D accepts $\langle G, w \rangle$; otherwise, it rejects $\langle G, w \rangle$.

CF–Finiteness

Problem: *CF–Finiteness*
Question: Let $G \in {}_{CNF-CF}\Psi$. Is $L(G)$ finite?
Language: ${}_{CF\text{–}Finiteness}L = \{\langle G\rangle |\ G \in {}_{CNF-CF}\Psi, L(G) \text{ is finite}\}$.

Corollary

${}_{CF\text{–}Finiteness}L \in {}_{TD}\Phi$

References

- Wayne Goddard.
  Introducing the Theory of Computation.
  Jones Bartlett Publishers, 2008.
- Jeffrey D. Ullman John E. Hopcroft, Rajeev Motwani.
  Introduction to Automata Theory, Languages, and Computation.
  Addison Wesley, 2006.
- Dexter C. Kozen.
  Automata and Computability.
  Springer, 2007.
- Dexter C. Kozen.
  Theory of Computation.
  Springer, 2010.
- John C. Martin.
  Introduction to Languages and the Theory of Computation.
  McGraw-Hill Science/Engineering/Math, 2002.

Thank you for your attention!

End