Decidability and Decidable Problems for Finite Automata

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
FACULTY OF INFORMATION TECHNOLOGY

Advanced Topics of Theoretical Computer Science

FRVŠ MŠMT FR2581/2010/G1

Consider any problem P expressed by a language.

- P is associated
  - with the set of all its instances Π and
  - with a property π.
- Each instance either satisfies or does not sutisfies property π.

Definition

- Given a particular instance $i \in \Pi$ and its string representation $\langle i \rangle$.
- P asks whether or not i satisfies π.
- Encoding language of P is defined as

$$_{P}L = \{\langle i \rangle \mid i \in \Pi, i \text{ satisfies } \pi\}.$$

A Turing decider M solves P if

1. M rejects every input that represents no instance from Π and
2. for every $\langle i \rangle$ where $i \in \Pi$, M accepts $\langle i \rangle$ iff i satisfies π.

P is stated as:

Problem: P
Question: a formulation of P
Language: ${}_{P}L$

Example

Problem: *FA–Emptiness*
Question: Let $M \in {}_{FA}\Psi$, $L(M) = \emptyset$?
Language: ${}_{FA\text{–}Emptiness}L = \{\langle M\rangle \mid M \in {}_{FA}\Psi, L(M) = \emptyset\}$.

For any finite automaton M, *FA–Emptiness* asks whether the language accepted by M is empty.

- Turing decider for ${}_{FA\text{–}Emptiness}L$ can be constructed in a trivial way.

Definition

I. Let $M \in {}_{TM}\Psi$. M is Turing decider if
- M always halts and
- M–f is a function from $x \in \triangle^*$ to $\{\varepsilon\}$.

II. Let L be a lanuage and $M \in {}_{TM}\Psi$ be a Turing decider. M is a Turing decider for L if $domain(M\text{–}f) = L$

III. A language is decidable if there exists a Turing decider for it. Otherwise, the language is undecidable.

By I, $M \in {}_{TM}\Psi$ is a Turing decider if
- it never loops and
- for every $x \in \triangle^*$, $\triangleright\blacktriangleright x\triangleleft \Rightarrow^* \triangleright iu\triangleleft$ where $i \in \{\blacksquare, \blacklozenge\}$ and $u \in \square^*$.

By II, Turing decider M for a language L satisfies
- for every $x \in L$, $\triangleright\blacktriangleright x\triangleleft \Rightarrow^* \triangleright\blacksquare u\triangleleft$ in M and
- for every $y \in \triangle^* - L$, $\triangleright\blacktriangleright y\triangleleft \Rightarrow^* \triangleright\blacklozenge v\triangleleft$ in M where $u, v \in \square^*$.

Convention

- ${}_{TD}\Psi$ denotes the set of all Turing deciders.
- ${}_{TD}\Phi = \{L(M) |\ M \in {}_{TD}\Psi\}$.

Theorem

${}_{FA}\Phi \subset {}_{CF}\Phi \subset {}_{TD}\Phi$

Example

Let $L = \{x|\ x \in \{a, b, c\}^*, occur(x, a) = occur(x, b) = occur(x, c)\}$.

- Consider Turing Machine D such that $D \in {}_{TD}\Psi$ and D accepts L.
- D can be designed by this way:
  - D repeatedly scans across the tape in a left-to-right way.
  - During every single scan, D is erasing the leftmost occurrence of a, b, and c.
  - If $\lhd$ is reached after erasing all these three occurences, D moves to $\rhd$ and makes another scan.
  - If $\lhd$ is reached while least one of the three symbol missing, D makes final return to $\rhd$ in dependency on whether its tape is blank.
- D is a Turing decider for L, so L is a decidable language.
- Symbolically, $D \in {}_{TD}\Psi$ and $L \in {}_{TD}\Phi$.

Convention

- ${}_{CS-FA}\Psi$ is the set of all completely specified finite automata.
- $\langle M \rangle$ represents the code of $M \in {}_{CS-FA}\Psi$.
- $\langle M, w \rangle$ denotes $(M, w) \in {}_{CS-FA}\Psi \times \triangle^*$.
- $\langle M, N \rangle$ denotes $(M, N) \in {}_{CS-FA}\Psi \times {}_{CS-FA}\Psi$.

FA–Emptiness

Problem: *FA–Emptiness*
Question: Let $M \in {}_{CS-FA}\Psi$. Is $L(M)$ empty?
Language: ${}_{FA\text{–}Emptiness}L = \{\langle M \rangle |\ M \in {}_{CS-FA}\Psi, L(M) = \emptyset\}$.

Theorem

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

Proof:

- We know that M is completely specified finite automaton.
- Therefore, each of its states is reachable.
- Thus, $L(M) = \emptyset$ iff ${}_{M}F = \emptyset$.
- Design a Turing decider D that work on every $\langle M \rangle$, where $M \in {}_{CS-FA}\Psi$:
  - D accepts $\langle M \rangle$ iff ${}_{M}F = \emptyset$,
  - otherwise rejects $\langle M \rangle$.

FA–Membership

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

Theorem

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

Proof:

- Proper finite automaton M reads an input symbol during every move.
- After making $|w|$ moves on $w \in \triangle^*$, M either accepts or rejects w.
- Design Turing decider D that works on every $\langle M, w\rangle$ as follows:
    - D runs M on w until M either accepts or rejects w.
    - D accepts $\langle M, w\rangle$ iff M accepts w, and
    - D rejects $\langle M, w\rangle$ iff M rejects w.

FA–Infiniteness

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

- M is completely specified finite automaton.
- It is easy to see, $L(M)$ is infinite iff its state diagram contains a cycle.
- *FA–Infiniteness* can be reformulated to terms of graph theory.
- Alternatively, it is possible to use pumping lemma for regular language in the following way:
  - For every $M \in {}_{FA}\Psi$, let ${}_{\infty ?}L(M)$ denotes finite language ${}_{\infty ?}L(M) = \{x | x \in L(M), card({}_{M}Q) \leq |x| < 2card({}_{M}Q)\}$.

Lemma

For every $M \in {}_{CS-FA}\Psi$, $L(M)$ is infinite iff ${}_{\infty ?}L(M) \neq \emptyset$.

Proof: the *if* part of the equivalence:

- Suppose that ${}_{\infty?}L(M) \neq \emptyset$.
- Take any $z \in {}_{\infty?}L(M)$.
- Pumping lemma constant k equals $card()MQ)$.
- Because $card()MQ) \leq z$, $z = uvw$, where:
  - $0 < |v| \leq |uv| \leq card()MQ)$ and
  - $uv^m w \in L$ for all $m \geq 0$.
- Hence, $L(M)$ is infinite.

Proof: the *only if* part of the equivalence:

- Assume that L is infinite.
- Let z be the shortest string such that:
  - $z \in L$ and $|z| \geq 2card()MQ)$.
- From Pumping Lemma, $z = uvw$, where:
  - $0 < |v| \leq |uv| \leq card()MQ)$ and
  - $uv^m w \in L$ for all $m \geq 0$.
- Take $uv^0 w = uw \in L(M)$.
- Observe that $2card()MQ) \geq |uw|$.
- As $0 < |v| \leq card()MQ)$, $card()MQ) \leq uw < 2card()MQ) \leq |z|$,
- so $uw \in {}_{\infty?}L(M)$ and, therefore ${}_{\infty?}L(M) \neq \emptyset$.

Theorem

${}_{FA\text{—}Infiniteness}L \in {}_{TD}\Phi$

Proof:

- Construct a Turing decider D that works on every $\langle M \rangle \in {}_{FA\text{—}Finiteness}L$ so it first construct ${}_{\infty?}L(M)$.
- D accepts $\langle M \rangle$ iff ${}_{\infty?}L(M) \neq \emptyset$, and
- D rejects $\langle M \rangle$ iff ${}_{\infty?}L(M) = \emptyset$

FA–Finiteness

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

Corollary

${}_{FA\text{—}Finiteness}L \in {}_{TD}\Phi$

FA–Equivalence

Problem: *FA–Infiniteness*
Question: Let $M, N \in {}_{CS-FA}\Psi$. Are M and N equivalent?
Language: ${}_{FA\text{–}Equivalence}L = \{\langle M, N\rangle |\ M, N \in {}_{CS-FA}\Psi, L(M) = L(N)\}$.

Theorem

${}_{FA\text{–}Equivalence}L \in {}_{TD}\Phi$

Proof:

- It is easy to prove that $L(M) = L(N)$ iff
  - $\emptyset = (L(M) \cap \sim L(N)) \cup (L(N) \cap \sim L(M))$.
- Turing decider D works on every $\langle M, N\rangle \in {}_{FA\text{–}Equivalence}L$:
  - From M and N construct finite automaton O such that $L(O) = (L(M) \cap \sim L(N)) \cup (L(N) \cap \sim L(M))$.
  - D converts O to an equivalent $P \in {}_{CS-FA}\Psi$.
  - D decides whether $L(P) = \emptyset$.
  - If $P = \emptyset$, $L(M) = L(N)$ and D accepts $\langle M, N\rangle$.
  - If $P \neq \emptyset$, $L(M) \neq L(N)$ and D rejects $\langle M, N\rangle$.

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