Decidability

Andreas Karwath und Malte Helmert

Overview

* An investigation into the solvable/decidable
* Decidable languages
* The halting problem (undecidable)

Decidable problems?

* **Acceptance** problem:
  * decide whether an automaton accepts a string

* **Equivalence** problem:
  * Decide whether two automata are equivalent, i.e. accept the same language

* **Emptiness testing** problem:
  * Decide whether the language of an automaton is empty

* Can be applied to
  * DFA, NFA, REX, PDA, CFG, TM,…

Acceptance problem for DFAs (T 4.1)

To decide whether a particular DFA accept a given string $w$, we express this in a language: $A_{\text{DFA}}$.

$A_{\text{DFA}}$ contains the encodings of all DFAs together with the string $w$ the DFAs accept:

$$A_{\text{DFA}} = \{ (B, w) \mid B \text{ is a DFA that accepts input string } w \}$$

The problem of testing whether a DFA $B$ accepts $w$ is the same as the problem of whether $<B, w>$ is a member of language $A_{\text{DFA}}$.

**Theorem 4.1**

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

**Proof**

$M = "\text{On input } \langle B, w \rangle, \text{ where } B \text{ is a DFA and } w \text{ is a string:}"

1. Simulate $B$ on input $w$.
2. If the simulation ends in an accept state, accept. If it ends in a nonaccepting state, reject."
Acceptance problem for NFAs (T 4.2)

\[ A_{\text{NFA}} = \{ \langle B, w \rangle \mid B \text{ is an NFA that accepts input string } w \} \]

**Theorem 4.2**

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

**Proof**

\( N = \langle B, w \rangle \) where \( B \) is an NFA, and \( w \) is a string:

1. Convert NFA \( B \) to an equivalent DFA \( C \) using the procedure for this conversion given in Theorem 1.19 (TS2, slide 30 ff).
2. Run TM \( M \) from Theorem 4.1 on input \( \langle C, w \rangle \).
3. If \( M \) accepts, accept; otherwise reject.

Running TM \( M \) in stage 2 means incorporating \( M \) into the design of \( N \) as a subprocedure.

---

Acceptance problem for Regular Expressions (T 4.3)

\[ A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression that generates input string } w \} \]

**Theorem 4.3**

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

**Proof**

The following TM \( P \) decides \( A_{\text{REX}} \):

\( P = \langle R, w \rangle \) where \( R \) is a regular expression and \( w \) is a string:

1. Convert regular expression \( R \) to an equivalent DFA \( A \) by using the procedure for this conversion given in Theorem 1.28.
2. Run TM \( M \) on input \( \langle A, w \rangle \).
3. If \( M \) accepts, accept; if \( M \) rejects, reject.

---

Emptiness testing problem for DFAs (T 4.4)

\[ E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is DFA for which } L(A) = \emptyset \} \]

**Theorem 4.4**

\[ E_{\text{DFA}} \text{ is a decidable language} \]

**Proof**

A DFA accepts some string if and only if reaching an accept state from the start state by traveling along the arrows of the DFA is possible. To test this condition we can design a TM \( T \) that uses a marking algorithm similar to that used in the example about connected graphs in acs-06, slide 33. \( T = \langle A \rangle \) where \( A \) is a DFA:

1. Mark the start state of \( A \).
2. Repeat until no new states get marked:
   1. Mark any state that has a transition coming into it from any state that is already marked.
3. If no accept state is marked, accept; otherwise reject.

---

Equivalence problem for DFAs (T 4.5)

\[ E_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \]

**Theorem 4.5**

\[ E_{\text{DFA}} \text{ is a decidable language} \]

**Proof**

The symmetric difference of \( L(A) \) and \( L(B) \)

\[ L(C) = (L(A) \cap \overline{L(B)}) \cup (L(A) \cap L(B)) \]

This expression is sometimes called the symmetric difference of \( L(A) \) and \( L(B) \). Here \( \overline{L(A)} \) is the complement of \( L(A) \). The symmetric difference is useful here because \( L(C) = \emptyset \) if and only if \( L(A) = L(B) \).

One can construct \( C \) from \( A \) and \( B \) with the constructions for proving the class of regular languages are closed under the complement, union, and intersection. These constructions are algorithms that can be carried out by Turing machines. Once \( C \) has been constructed one can use Theorem 4.4 to test whether \( L(C) \) is empty.

If it is empty, \( L(A) \) and \( L(B) \) must be equal.

\( F = \langle A, B \rangle \), where \( A \) and \( B \) are DFA's:

1. Construct DFA \( C \) as described.
2. Run TM \( T \) from Theorem 4.4 on input \( \langle C \rangle \).
3. If \( T \) accepts, accept. If \( T \) rejects, reject.
Acceptance problem for CFGs (T 4.6)

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

**Theorem 4.6**

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

**Proof**

Relies on the following property:

If \( G \) is in Chomsky Normal Form, then any derivation of \( w \) has length at most \( 2|w| - 1 \).

There are only finitely many derivations of length less than \( n \).

The TM \( S \) for \( A_{CFG} \) follows.

\( S = \) "On input \( \langle G, w \rangle \), 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, where \( n \) is the length of \( w \),
   except if \( n = 0 \), then instead list all derivations with 1 step.
3. If any of these derivations generate \( w \), accept; if not, reject."

Emptiness testing problem for CFGs (T 4.7)

\[ E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG for which } L(G) = \emptyset \} \]

**Theorem 4.7**

\( E_{CFG} \) is a decidable language

**Proof**

Determine for each variable whether that variable is capable of generating a string of terminals

\( R = \) "On input \( \langle G \rangle \), where \( G \) is a CFG:

1. Mark all terminal symbols in \( G \).
2. Repeat until no new variables get marked:
3. Mark any variable \( A \) where \( G \) has a rule \( A \rightarrow U_1 \ldots U_k \) and each symbol \( U_1, \ldots, U_k \) has already been marked.
4. If the start symbol is not marked, accept; otherwise reject."

Equivalence problem for CFGs

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

**Theorem**

\( E_{EQ_{CFG}} \) is not decidable

**Proof**

Follows later

The problem with adapting the proof for DFAs is that the class of context free languages is not closed under complementation or intersection!

Every CFL is decidable (T 4.8)

**Theorem 4.8**

Every context free language is decidable

**Proof**

Let \( G \) be a CFG for \( A \) and design a TM \( M_G \) that decides \( A \). We build a copy of \( G \) into \( M_G \). It works as follows.

\( M_G = \) "On input \( w \):

1. Run TM \( S \) (from T4.6) on input \( \langle G, w \rangle \)
2. If this machine accepts, \( \text{accept} \); if it rejects, \( \text{reject} \)."
The relationship among classes of languages

- Regular
- Context-free
- Decidable
- Turing recognizable

The halting problem

- There is a specific problem that is algorithmically unsolvable (undecidable), e.g. the halting problem
- Philosophical implications: computers are fundamentally limited

Diagonalization

- Georg Cantor 1873
- Measure the size of (infinite) sets

Consider the function \( f: A \to B \)
- \( f \) is injective (one-to-one), if \( f(a) \neq f(b) \) whenever \( a \neq b \)
- \( f \) is surjective (onto), if for every \( b \in B \) there is an \( a \in A \): \( f(a) = b \)
- \( f \) is bijective (correspondence) if it is injective and surjective

A and \( B \) are said to be the same size, if there exists a bijective function \( f \), i.e. for every element in \( A \) there exists an unique element in \( B \).

Example: \( f: \mathbb{N}(natural numbers) \to E(even numbers) \)
- \( f(n) = 2n \) is a bijective function
- Both sets have the same size

Definition: A set is countable, if it is finite or has the same size as \( \mathbb{N} \).
\[ Q = \{ \frac{m}{n} \mid m, n \in \mathbb{N} \} \] the positive rational numbers

**Theorem**

\( Q \) is countable

**Proof** idea

We prove (by contradiction) that there is no correspondence between \( R \) and \( \mathbb{N} \).

Assume that there were a correspondence \( f \).

We now construct an \( x \in R \) that is not paired with any element of \( \mathbb{N} \).

Choose \( i-th \) fractional digit of \( x \) different from \( i-th \) frac. digit of \( f(i) \).

\[ x = 0.275... \]

So, \( x \neq f(n) \) for all \( n \).
Some languages are not Turing-recognizable (T 4.15)

**Theorem (T4.15)**

Some languages are not Turing recognizable

**Proof**

There is a countable number of Turing Machines (Each Turing Machine can be encoded in a string; the set of all strings over a finite alphabet is countable; not all strings need to encode legal TMs)

The set of all languages is uncountable

Therefore there is no correspondence between the set of all TMs and the set of all languages.

---

**A_{TM} is undecidable (T4.9)**

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \} \]

**Theorem (T4.9)**

\[ A_{TM} \text{ is undecidable} \]

**Proof** by contradiction; assume \( A_{TM} \) is decidable

Suppose \( H \) is a decider for \( A_{TM} \)

\[ H(< M, w >) = \begin{cases} 
\text{accept if } M \text{ accepts } w \\
\text{reject if } M \text{ does not accept } w
\end{cases} \]

Further Explanations

**Entry \( i,j \) is accept if \( M_i \) accepts \( <M_j> \)**

Use \( H \) to define \( D \):

On input \( \langle M \rangle \), where \( M \) is a TM

1. Run \( H \) on input \( <M, <M> > \)
2. Output the opposite of what \( H \) outputs

So, \( D(<M>) = \begin{cases} 
\text{accept if } M \text{ does not accept } <M> \\
\text{reject if } M \text{ accepts } <M>
\end{cases} \)

and

\[ D(<D>) = \begin{cases} 
\text{accept if } D \text{ does not accept } <D> \\
\text{reject if } D \text{ accepts } <D>
\end{cases} \]

This is impossible!
Entry $i,j$ is the value of $H$ on input $<M_i,M_j>$. 

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td></td>
</tr>
</tbody>
</table>

What happens if $D$ occurs?

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>...</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
</tbody>
</table>

A language is co-Turing recognizable if it is the complement of a language that is Turing recognizable.

**Theorem (T 4.16)**
A language is decidable if and only if it is both Turing-recognizable and co-Turing recognizable.

**Proof**
1. If $A$ is decidable then $A$ and $\overline{A}$ Turing recognizable
   Trivial
2. If $A$ and $\overline{A}$ are Turing recognizable then $A$ is decidable
   Let $M_A$ and $M_\overline{A}$ be TMs for $A$ and $\overline{A}$
   Define $M$ such that $M$ decides $A$
   On input $w$
   1. Run both $M_A$ and $M_\overline{A}$ on $w$ in parallel
   2. If $M_A$ accepts, then accepts;
      If $M_\overline{A}$ accepts, then reject;
   3. All strings are either in $A$ or $\overline{A}$
   4. Either $M_A$ or $M_\overline{A}$ must accept any given string $w$
   5. $M$ always terminates with correct answer

$A_{TM}$ is not Turing-recognizable

**Theorem (T 4.17)**
$A_{TM}$ is not Turing-recognizable

**Proof**
$A_{TM}$ is Turing-recognizable
If $A_{TM}$ were also Turing-recognizable
Then $A_{TM}$ would be decidable.
Summary

The relationship among languages

Informatik Theorie II (A) WS2009/10