Decidability

Bernhard Nebel and Christian Becker-Asano

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

To decide whether a particular DFA accept a given string w, we express this in a language: A_{DFA}.
A_{DFA} contains the encodings of all DFAs together with the string w the DFAs accept:

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

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_{DFA} .

Proof idea:
Decider M will simulate B on input w. If B would accept w, then M accepts; if B rejects, then M rejects.

Acceptance problem for DFAs

- Convert B $= (Q, \Sigma, \delta, q_0, F)$ and input w into string for TM's tape.
  - E.g.:

q0	q1	q2	q3	#	0	1	#	<table>	…
#	q0	#	q2	q3	#	#	0	1	1
0	1	1	1	0	#	q0			

- **Theorem 4.1**:
  A_{DFA} is a decidable language.
- Proof:
  M= „On input $\langle B, w \rangle$, where B is a DFA and w 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)

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

- **Theorem 4.2:**
  A_{NFA} is a decidable language.
- Proof:
  N = "On input $\langle B, w \rangle$ 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 1.39.
  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 of N means incorporating M into the design of N as a subprocedure.

Acceptance problem for regular expressions

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

- **Theorem 4.3:**
  A_{REX} is a decidable language.
- Proof sketch:
  1. Convert regular expression R into an NFA A according to Theorem 1.54.
  2. Use Theorem 4.2 to run TM N on input $\langle A, w \rangle$.
  3. If N accepts…

Emptiness testing problem for DFAs

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

- **Theorem 4.4**
  E_{DFA} is a decidable language.
- Proof:
  A DFA accepts some string iff reaching an accept state from the start state using valid transitions 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 graph presented in the end of the lecture about "Turing machines."
  T=„On input $\langle A \rangle$ where A is a DFA:
  1. Mark the start state of A.
  2. Repeat until no new states get marked:
     3. Mark any state that has a transition coming into it from any state that is already marked.
  4. If no accept state is marked, *accept*; otherwise, *reject*."

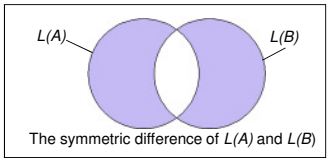
Equivalence problem for DFAs

$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$

Theorem 4.5

EQ_{DFA} is a decidable language

Proof



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

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 $L(C) = \varnothing$ 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="On input $\langle A, B\rangle$, 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

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

Theorem 4.7

A_{CFG} 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

$E_{CFG} = \{\langle G\rangle \mid G \text{ is a CFG for which } L(G) = \varnothing\}$

Theorem 4.8

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 $A \to U_1 U_2 ... U_k$ and each symbol $U_1,...,U_k$ has already been marked.
4. If the start symbol is not marked, *accept*; otherwise *reject*."

Equivalence problem for CFGs

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

- In fact:
  EQ_{CFG} is not decidable!
- The proof follows later.
- A method along the lines of the proof for DFAs is bound to fail, because the class of context free languages is not closed under complementation or intersection!

Every CFL is decidable

Theorem 4.9

Every context free language is decidable

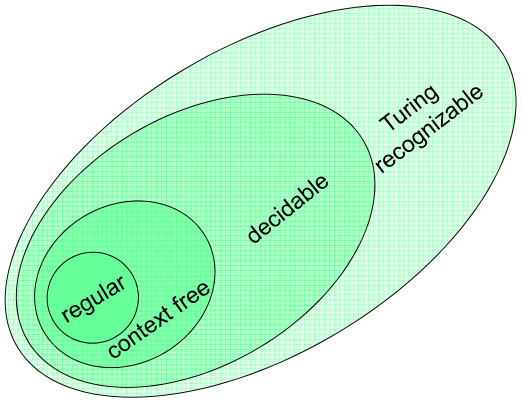
Proof

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 T4.7) on input $\langle G, w \rangle$
2. If this machine accepts, *accept*; if it rejects, *reject*."

The relationship amoung classes of languages



The halting problem

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

The halting problem

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

- **Theorem 4.11**
  A_{TM} is undecidable.
- Proof given later.
- Observe that A_{TM} is turing-recognizable. Thus, recognizers are more powerful than deciders, because requiring a TM to halt on all inputs (decider) restricts the kinds of languages that it can recognize. The following TM U recognizes A_{TM}.
  U=„On input $\langle M, w\rangle$ where M is a TM and w is a string:
  1. Simulate M on input w.
  2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*.“
- Note: This machine loops on input $\langle M, w\rangle$ if M loops on w, which is why it does not decide A_{TM}.

Diagonalization

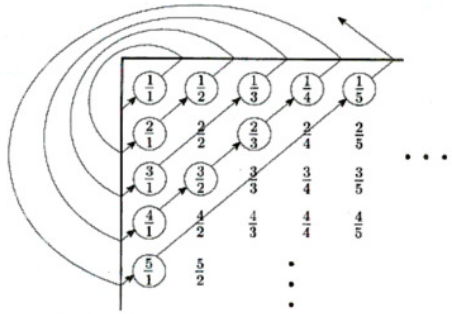
- Georg Cantor 1873
- Measure the size of (infinite) sets

n	f(n)= 2n
1	2
2	4
3	6
…	…
I	2i

- Consider the function *f: A→B*
  - f is ***injective*** (*one-to-one*), if *f(a) ≠ f(b)* whenever *a ≠ b*
  - f is ***surjective*** (*onto*), if for every *b ∈ B* there is an *a ∈ A: f(a) = b*
  - f is ***bijective*** (*corresponence*) 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: N(natural numbers) → 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 N*.

Example 4.15: rational numbers

- Let $\mathbb{Q} = \left\{\frac{m}{n} \mid m, n \in \mathbb{N}\right\}$ be the set of positive rational numbers.
- $\mathbb{Q}$ seems to be much larger than $\mathbb{N}$, but
- $\mathbb{Q}$ is countable



ℝ is uncountable

- ℝ = the set of real numbers (have a decimal representation)
- **Theorem 4.17**
  ℝ is uncountable.
- Proof idea:
  We proof (by contradiction) that there is no correspondence between ℝ and ℕ.
  Assume that there were a correspondence f, then f must pair all members of ℕ with all members of ℝ.
  → We construct $x \in \mathbb{R}$ that is not paired with any element of ℕ by choosing the i^{th} fractional digit of x to be different from the i^{th} fractional digit of $f(i)$.

Example

n	*f(n)*
1	3.1414…
2	5.567…
3	0.888888…
…	…

x = 0.275…

So, $x \neq f(n)$ for all n

B = the set of all infinite binary strings

Lemma

B is uncountable

Proof idea

By analogy to □

Let L be the set of all languages over Σ

Lemma

L is uncountable

Proof idea

We define a correspondence between L and B

Let $\Sigma^* = \{s_1, s_2, \ldots\}$; which is countable

Each language A in L has a unique characteristic sequence χ_A in B defined as follows

$\Sigma^* = \{ \quad \varepsilon, \quad 0, \quad 1, \quad 00, \quad 01, \quad 10, \quad 11, \quad 000, \quad 001, \quad \ldots \quad \}$;

$A = \{ \quad\quad 0, \quad\quad 00, \quad 01, \quad\quad\quad 000, \quad 001, \quad \ldots \quad \}$;

$\chi_A = \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad \ldots \quad .$

$f : L \to B : f(A) = \chi_A$ is a correspondence

Some languages are not Turing-recognizable

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 (Proof)

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

Theorem 4.11 (revisited)

A_{TM} is undecidable

Proof by contradiction; assume A_{TM} is decidable

Suppose H is a decider for A_{TM}

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

A_{TM} is undecidable (ctd.)

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 ;

Further Explanations

H accepts $\langle M, w\rangle$ when M accepts w

D rejects $\langle M \rangle$ when M accepts $\langle M \rangle$

D rejects $\langle D \rangle$ when D accepts $\langle D \rangle$

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

and

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

This is impossible !

Entry *i,j* is *accept* if M_i accepts $<M_j>$

	<M1>	<M2>	<M3>	<M4>	...
M1	*accept*		*accept*		
M2	*accept*	*accept*	*accept*	*accept*	
M3					...
M4	*accept*		*accept*		
⋮		⋮			

Entry *i,j* is the value of *H* on input $\langle M_i, \langle M_j \rangle \rangle$

	<M1>	<M2>	<M3>	<M4>	...
M1	accept	reject	accept	reject	
M2	accept	accept	accept	accept	
M3	reject	reject	reject	reject	...
M4	accept	reject	accept	reject	
⋮		⋮			

What happens if *D* occurs?

	<M1>	<M2>	<M3>	<M4>	...	<D>	...
M1	accept	reject	accept	reject		accept	
M2	accept	accept	accept	accept		accept	
M3	reject	reject	reject	reject	...	reject	...
M4	accept	reject	accept	reject		accept	
⋮		⋮			⋱		
D	reject	reject	accept	accept		?	
⋮		⋮					⋱

Co-Turing-recognizable and decidability

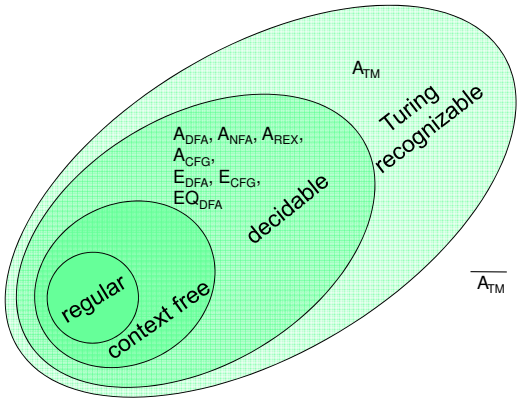
- **Theorem 4.22**
  A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.
- Proof:
  Forward direction: If A is decidable, we can easily see that both A and its complement $\bar{A}$ are Turing-recognizable. Any decidable language is Turing-recognizable and the complement of a decidable language also is decidable.
  Backward direction:
  - Assume both A and $\bar{A}$ are Turing-recognizable.
  - Let M_1 be the recognizer for A and M_2 the recognizer for $\bar{A}$.
  - The following TM M is a decider for A:

  M = „On input w:
  1. Run both M_1 and M_2 on input w in parallel.
  2. If M_1 accepts, *accept*; if M_2 accepts, *reject*.“

$\overline{A_{TM}}$ is not Turing-recognizable

- **Corollary 4.23**
  $\overline{A_{TM}}$ is not Turing-recognizable.
- Proof:
  - We know that A_{TM} is Turing-recognizable.
  - Assume that $\overline{A_{TM}}$ were also Turing-recognizable.
  - → A_{TM} would be decidable.
  - → Theorem 4.11 tells us that A_{TM} is undecidable.
  - → Contradiction! → $\overline{A_{TM}}$ is not Turing-recognizable.

Summary



The relationship amoung languages