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_{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} .

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)

$A_{NFA} = \{\langle B, w\rangle \mid B$ 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 1.19 (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)

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

Theorem 4.3

A_{REX} is a decidable language

Proof The following TM P decides A_{REX}

P="On input $\langle R,w\rangle$ 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)

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

Theorem 4.4

E_{DFA} 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 = "On input ⟨A⟩ 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."
```

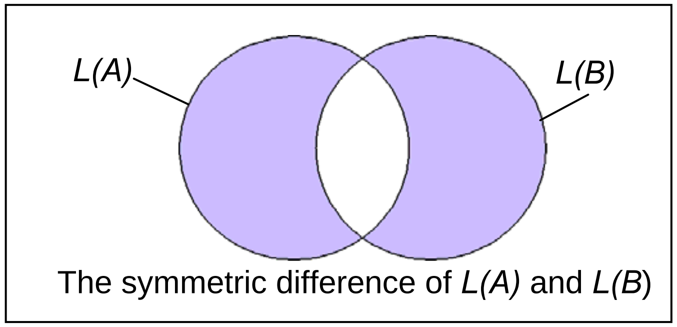
Equivalence problem for DFAs (T 4.5)

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

Theorem 4.5

EQ_{DFA} is a decidable language

Proof



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

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

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 $L(C) = \emptyset$ 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.

$F =$"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 (T 4.6)

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

Theorem 4.6

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 $2|\mathrm{w}|-1$

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 ⟨G⟩, 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 $A \to U_1U_2...U_k$ 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

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

Theorem

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

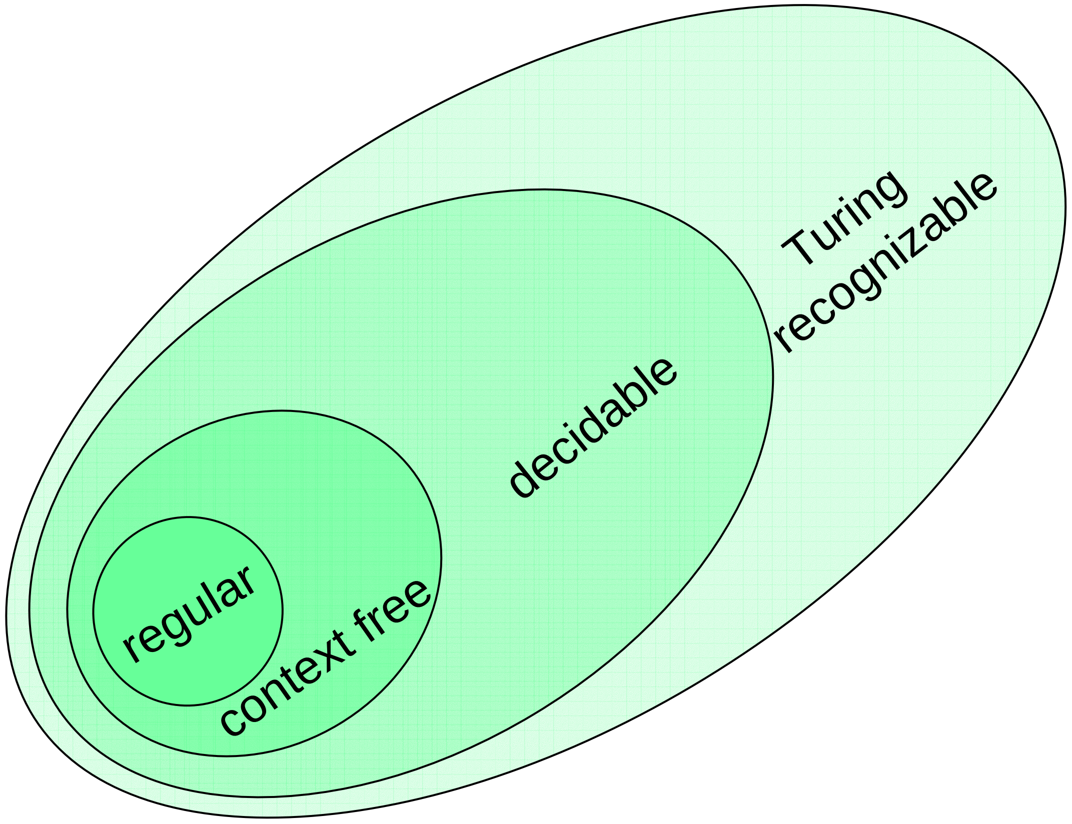
Proof

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, *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 (T 4.9)

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

Theorem

A_{TM} is Turing recognizable

Proof

Consider U : (*Universal Turing Machine*)

On input $<M, w>$, where M is a TM and w a string

1. Simulate M on w
2. If M ever enters its accept state, *accept*,
   if M ever enters its reject state, *reject*

U loops when M does, the halting problem:

Theorem A_{TM} is undecidable

shows that recognizers are more powerful than deciders requires quite involved proof

Diagonalization

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

- 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

n	*f(n)= 2n*
1	2
2	4
3	6
…	…
I	2i

- Definition: A set is ***countable***, if it is *finite* or has *the same size as N*.

$Q = \{\frac{m}{n} \mid m, n \in N\}$ the positive rational numbers

Theorem

Q is countable

Proof idea

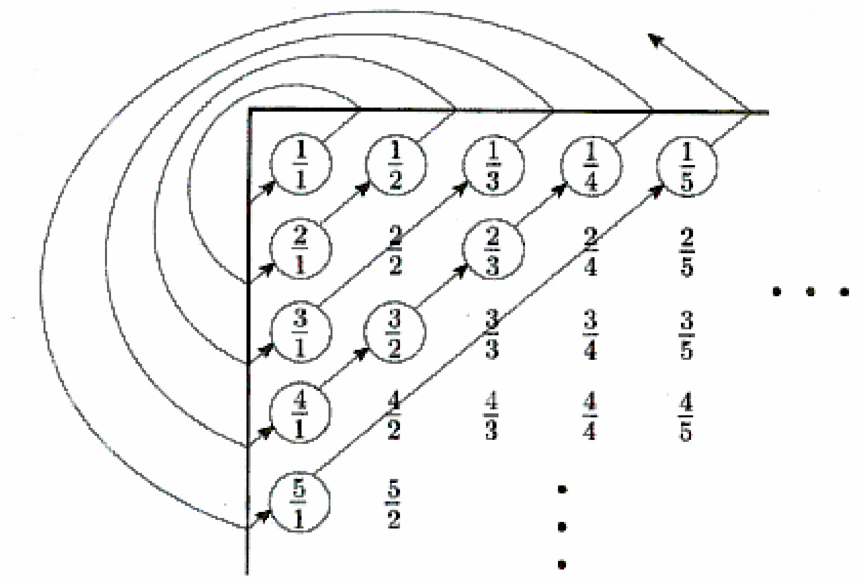


FIGURE 4.3
A correspondence of $\mathcal{N}$ and $\mathcal{Q}$

$\mathbb{R}$ is uncountable (T 4.14)

$R =$ the set of real numbers (have a decimal representation)

Theorem (T 4.14)

R is uncountable

Proof idea

We prove (by contradiction) that there is no correspondence between R and N

Assume that there were a correspondence f

We now construct an $x \in R$ that is not paired with any element of N

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

Example

n	*f(n)*
1	3.$\underline{1}$414…
2	5.5$\underline{6}$7…
3	0.88$\underline{8}$888…
…	…

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 R

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

$$\begin{array}{lllllllllllll}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \ldots & \} & ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \ldots & \} & ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \ldots & . &
\end{array}$$

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} is undecidable

Proof by contradiction; assume A_{TM} is decidable

Suppose H is a decider for A_{TM}

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

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

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 ;

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

and

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

This is impossible !

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 $\langle D \rangle$

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 $<M_i,<M_j>>$

	<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*		?	
⋮		⋮					⋱

T 4.16

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_1 and M_2 be TMs for A and $\overline{A}$

   Define M :

```
On input w
    1. Run both $M_1$ and $M_2$ on $w$ in parallel
    2. If $M_1$ accepts, then accepts;
       If $M_2$ accepts, then reject;
```

M decides A

all strings are either in A or $\overline{A}$

either M_1 or M_2 must accept any given string

M always terminates with correct answer

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

Theorem (T4.17)

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

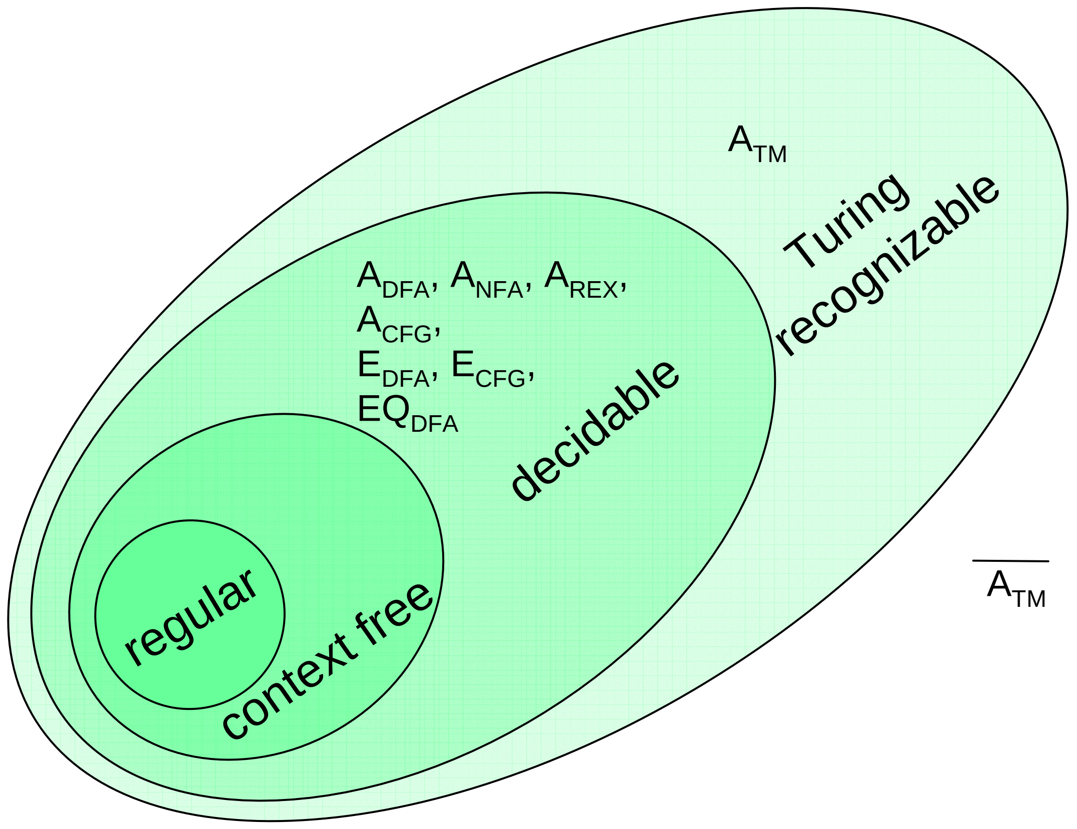
Proof

A_{TM} is Turing-recognizable

If $\overline{A_{TM}}$ were also Turing-recognizable

Then A_{TM} would be decidable.

Summary



The relationship amoung languages