Computational problems

A computational problem is \textbf{decidable} iff the language encoding the problem instances is decidable.
Computational problems

Sample computational problems and their encodings:

• $A_{DFA}$ "Check whether a string is accepted by a DFA."
  \[ \{ <B,w> \mid B \text{ is a DFA over } \Sigma, w \in \Sigma^*, \text{ and } w \text{ is in } L(B) \} \]

• $E_{DFA}$ "Check whether the language of a DFA is empty."
  \[ \{ <A> \mid A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \} \]

• $EQ_{DFA}$ "Check whether the languages of two DFA are equal."
  \[ \{ <A, B> \mid A \text{ and } B \text{ are DFA over } \Sigma, L(A) = L(B) \} \]

We will show that all of these problems are decidable!
Proving decidability: DFA Equality

Claim: \( \text{EQ}_{\text{DFA}} \) is decidable
Proof: WTS that \( \{ <A, B> \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. Idea: give high-level description
Step 1: construction

Will we be able to simulate A and B?
What does set equality mean?
Can we use our previous work?
Claim: $\text{EQ}_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A, B> | A, B \text{ are DFA over } \Sigma, L(A) = L(B) \}$ is decidable. Idea: give high-level description

Step 1: construction

Very high-level:
Build new DFA recognizing symmetric difference of $L(A)$ and $L(B)$. Check if this set is empty.
Proving decidability: DFA Equality

Claim: $\text{EQ}_{\text{DFA}}$ is decidable

Proof: WTS that \{ <A, B> | A, B are DFA over $\Sigma$, $L(A) = L(B)$ \} is decidable. Idea: give high-level description

Step 1: construction

Define TM J by: J = "On input <A,B>:

1. Check whether A,B are valid encodings of DFA; if not, reject.
2. Construct a new DFA, D, from A,B using algorithms for complement, union, intersection of regular languages such that $L(D) = \text{symmetric difference of } L(A) \text{ and } L(B)$.
3. Run machine K on <D>.
4. If K accepts, then accept; if K rejects, then reject."
Proving decidability: DFA Equality

Step 1: construction
Define TM J by: $J = \text{"On input } <A,B>:\text{"}:
1. Check whether $A, B$ are valid encodings of DFA; if not, reject.
2. Construct a new DFA, $D$, from $A, B$ using algorithms for complement, union, intersection of regular languages such that $L(D) = \text{symmetric difference of } L(A) \text{ and } L(B)$.
3. Run machine $K$ on $<D>$.
4. If $K$ accepts, then accept; if $K$ rejects, then reject."

Step 2: correctness proof
WTS (1) $L(J) = \text{EQ}_{\text{DFA}}$ and (2) $J$ is a decider.
Proving decidability: More Examples

All proven in Sipser 4.1.

- **A\textsubscript{NFA}** "Check whether a string is accepted by an NFA."
  \{ <B,w> | B is an NFA over \( \Sigma \), w in \( \Sigma^* \), and w is in \( L(B) \) \}

- **A\textsubscript{REX}** "Check whether a regular expression generates a string."
  \{ <R,w> | R is a regular expression over \( \Sigma \) that generates w in \( \Sigma^* \) \}

- **A\textsubscript{CFG}** "Check whether a string is generated by a CFG."
  \{ <G,w> | G is a CFG over \( \Sigma \) that generates w in \( \Sigma^* \) \}

- **E\textsubscript{CFG}** "Check whether the language of a CFG is empty."
  \{ <G> | G is a CFG over \( \Sigma \) and \( L(G) \) is empty \}

- **EQ\textsubscript{CFG}** "Check whether the languages of two CFGs are equal."
  \{ <G, H> | G and H are CFGs over \( \Sigma \), \( L(G) = L(H) \) \}"
Techniques for proving decidability

- **Subroutines**: can use decision procedures of decidable problems as subroutines in other algorithms
  - Examples: $A_{DFA}$, $E_{DFA}$, $EQ_{DFA}$

- **Constructions**: can use algorithms for constructions as subroutines in other algorithms
  - Converting DFA to DFA recognizing complement (or Kleene star).
  - Converting two DFA/NFA to one recognizing union (or intersection, concatenation).
  - Converting NFA to equivalent DFA.
  - Converting regular expression to equivalent NFA.
  - Converting DFA to equivalent regular expression.
Undecidable?

- There are many ways to prove that a problem is decidable.
- How do we find (and prove) that a problem is not decidable?
Counting arguments

Before we proved the Pumping Lemma …

We proved there was a language that was not regular because

All sets of strings

Regular Languages

Countable

Uncountable
Counting arguments

Recall: sets A and B have the same size, $|A| = |B|$ means there is a one-to-one and onto function between them.

A set is countable iff it is either
- finite, or
- has the same size as $\mathbb{N}$ (can list all and only the elements of the set in a sequence)
Which of the following is true?

A. Any two infinite sets have the same size.
B. If A is a strict subset of B and then A and B do not have the same size.
C. If A is a subset of B and B is countable, then A is countable.
D. If A is countable then AxA is not countable.
E. I don't know.
Examples of countable sets

\[ N \]
\[ Z \]
\[ Q \]
\[ \{0,1\}^* \]
\[ \Sigma^* \text{ for any alphabet } \Sigma \]

**Corollary:** The set of all TMs is countable.  
Sipser 4.18

**Proof Idea:** Each TM, \( M \), has an encoding as a string \(<M>\).  
Set of all strings is countable, so is a subset of it.
Examples of uncountable sets

$\mathbb{R}$

$[0,1]$

\{ infinite sequences of 0s and 1s \}

$P(\{0,1\}^*)$

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.
Examples of uncountable sets

$\mathbb{R}$

$[0,1]$

\{ infinite sequences of 0s and 1s \}

$P(\{0,1\}^*)$

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

What type of elements are in the set $P(\{0,1\}^*)$?

A. Binary strings

B. Regular expressions

C. Sets of binary strings

D. Sets of regular expressions

E. I don’t know
Proof that $\mathcal{P}({0,1}^*)$ is not countable

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

| 1) | Some set of binary strings, $A_1$ |
| 2) | Some set of binary strings, $A_2$ |
| 3) | Some set of binary strings, $A_3$ |
| 4) | Some set of binary strings, $A_4$ |
| ... | ... |

Define a set of binary strings $A$ that can’t be in this list to get a contradiction.
Proof that $P(\{0,1\}^*)$ is not countable

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

| 1) | Some set of binary strings, $A_1$ | 0 in $A_1$ $\iff$ 0 not in $A_1$ |
| 2) | Some set of binary strings, $A_2$ | 00 in $A_2$ $\iff$ 00 not in $A_2$ |
| 3) | Some set of binary strings, $A_3$ | 000 in $A_3$ $\iff$ 000 not in $A_3$ |
| 4) | Some set of binary strings, $A_4$ | 0000 in $A_4$ $\iff$ 0000 not in $A_4$ |
| ... | ... | $0^n$ in $A_n$ $\iff$ $0^n$ not in $A_n$ |
Proof that $P(\{0,1\}^*)$ is not countable

Diagonalization Proof: Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

<table>
<thead>
<tr>
<th></th>
<th>Some set of binary strings, $A_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2)</td>
<td>Some set of binary strings, $A_2$</td>
</tr>
<tr>
<td>3)</td>
<td>Some set of binary strings, $A_3$</td>
</tr>
<tr>
<td>4)</td>
<td>Some set of binary strings, $A_4$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Since $A$ is a set of binary strings, it must appear in our list at some position, $p$. $A_p = A$.

But then is $0^p$ in $A$?

$0^n \in A \iff 0^n \not\in A_n$
Proof that $\mathcal{P}(\{0,1\}^*)$ is not countable

Diagonalization Proof:
Assume towards a contradiction that the set is countable. This gives a correspondence with $\mathbb{N}$, but we can derive a contradiction.

1) Some set of binary strings, $A_1$
2) Some set of binary strings, $A_2$
3) Some set of binary strings, $A_3$
4) Some set of binary strings, $A_4$
...

Since $A$ is a set of binary strings, it must appear in our list at some position, $p$. $A_p = A$.

But then is $0^p$ in $A$?
Why is the set of Turing-recognizable languages countable?

A. It's equal to the set of all TMs, which we showed is countable.
B. It's a subset of the set of all TMs, which we showed in countable.
C. Each Turing-recognizable language is associated with a TM, so there can be no more Turing-recognizable languages than TMs.
D. More than one of the above.
E. I don't know.
Satisfied?

• Maybe not …

• What is a specific example of a language that is not Turing-recognizable? or not Turing-decidable?

• Idea: consider a set that, were it to be Turing-decidable, would have to "talk" about itself.