The Fundamentals: Algorithms

Diagonalization & Halting Problem

April 18, 2022

Outline

Languages

alphabet: A finite set of **symbols**; *e.g.* the *binary alphabet* is $\{0, 1\}$.

string: A sequence of zero or more symbols from an alphabet; *e.g.* ε (the empty string), 01011100010, 0, 101
Σ is used to represent the alphabet as a whole. ε (or λ) stand for the empty string.

language: A set of **strings**; *e.g.* $\{w | w \text{ starts with } 1\}$, $\{00, 01, 10, 11\}$.

Σ^*: The star (Kleene's star operator) means zero or more copies of the symbol before the star. This is short hand for the set of **all** the strings across the alphabet Σ.
Note: that means Σ^* is a *set*.

Cardinality of $\{0, 1\}^*$

$\{0, 1\}^*$ is infinite. Consider the set of just strings containing only '1' symbols. The lengths of different strings in this language range across non-negative integers. That is infinite.

Is Σ^* **countable**?

If so, how to prove it.

Cardinality of $\{0, 1\}^*$

$\{0,1\}^*$ is infinite. Consider the set of just strings containing only '1' symbols. The lengths of different strings in this language range across non-negative integers. That is infinite.

Is Σ^* **countable**?

If so, how to prove it.

Find bijection $f : \Sigma^* \rightarrow \mathbb{Z}^+$.

$f : \{0,1\}^* \to \mathbb{Z}^+$

Define the *length-then-value* ordering for binary strings: w_1 comes before w_2 if $|w_1| < |w_2|$ or $|w_1| = |w_2| \wedge$ the unsigned number represented by w_1 is less than the number represented by w_2.
So, Σ^* can be put in order by the above ordering:

$$\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...\}$$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f : \{0,1\}^* \to \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

z	s	$z - p$	$f(z)$
1	0	0	"" = ε
2	1	0	"0"
3	1	1	"1"
4	2	0	"00"
5	2	1	"01"
6	2	2	"10"
7	2	3	"11"
8	3	0	"000"
9	3	1	"001"
10	3	2	"010"
11	3	3	"011"

$f : \{0,1\}^* \rightarrow \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.
$f(23) =$

$f : \{0,1\}^* \to \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f(23) =$0111

$f(32) =$

$f : \{0, 1\}^* \to \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f(23) =$0111

$f(32) =$00000

$f^{-1}(\varepsilon) =$

$f : \{0,1\}^* \to \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f(23) =$0111
$f(32) =$00000
$f^{-1}(\varepsilon) =$0
$f^{-1}(1000) =$

$f : \{0,1\}^* \to \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f(23) =$0111

$f(32) =$00000

$f^{-1}(\varepsilon) =$0

$f^{-1}(1000) =$24

$f^{-1}(00010) =$

$f : \{0,1\}^* \rightarrow \mathbb{Z}^+$

$\forall z \in \mathbb{Z}$ let $s = \lfloor \log_2(z) \rfloor$ and $p = 2^s$. Then $f(z) = z - p$ written as a binary number s characters long.

$f(23) =$0111
$f(32) =$00000
$f^{-1}(\varepsilon) =$0
$f^{-1}(1000) =$24
$f^{-1}(00010) =$34

Cardinality of $\mathbb{P}(\Sigma^*)$

Review of Terms

- $\Sigma = \{0, 1\}$ = the *binary* alphabet
- $\Sigma^* =$

Cardinality of $\mathbb{P}(\Sigma^*)$

Review of Terms

- $\Sigma = \{0, 1\}$ = the *binary* alphabet
- $\Sigma^* = \{$ all *binary* strings$\}$
  $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...\}$
- $\mathbb{P}(\Sigma^*)$ = Set of all *subsets* of Σ^*

Cardinality of $\mathbb{P}(\Sigma^*)$

Review of Terms

- $\Sigma = \{0, 1\}$ = the *binary* alphabet
- Σ^* = { all *binary* strings}
  $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...\}$
- $\mathbb{P}(\Sigma^*)$ = Set of all *subsets* of Σ^*
  { all *binary* **languages**}

Cardinality of $\mathbb{P}(\Sigma^*)$

$\mathbb{P}(\Sigma^*)$ is *uncountable*.
$|\mathbb{P}(\Sigma^*)| > |\mathbb{Z}^+|$
$|\mathbb{P}(\Sigma^*)| > \aleph_0$

$\mathbb{P}(\Sigma^*)$ is Infinite

TBP: $\mathbb{P}(\Sigma^*)$ is *infinite*.

Let $S = \{L | L \in \mathbb{P}(\{0,1\}^*) \wedge |L| = 1\}$ S is the set of all *singleton* languages. Since there is one element in S for each element in $\{0,1\}^*$, $|S| = |\{0,1\}^*|$.

S is (countably) infinite.

$S \subset \mathbb{P}(\{0,1\}^*) \ \Rightarrow \ |S| \leq |\mathbb{P}(\{0,1\}^*)|$

$\therefore \mathbb{P}(\{0,1\}^*)$ is **infinite**. $\square$

$\mathbb{P}(\Sigma^*)$ is Uncountable

TBP: $\mathbb{P}(\Sigma^*)$ is *uncountable*.

TBP: $|\mathbb{P}(\Sigma^*)| \neq |\mathbb{Z}^+|$

FSOC: $|\mathbb{P}(\Sigma^*)| = |\mathbb{Z}^+|$ — Countably Infinite

1. bijection $\exists\, f : \mathbb{Z}^+ \to \mathbb{P}(\Sigma^*)$ — Same cardinality
2. f can be represented as a table

$\square$

Looking at f

	ε	0	1	00	01	10	11	000	001	…
1	0	1	0	1	0	1	0	1	0	…
2	0	1	0	1	1	1	0	0	0	…
3	0	0	1	0	0	0	0	0	0	…
4	1	0	1	0	1	0	0	0	1	…
⋮										

Each row represents a *subset* of Σ^* or an element of $\mathbb{P}(\Sigma^*)$. A sequence of Boolean values whether the string atop the column is/is not in the language in that row.

Looking at f

	ε	0	1	00	01	10	11	000	001	…
1	0	1	0	1	0	1	0	1	0	…
2	0	1	0	1	1	1	0	0	0	…
3	0	0	1	0	0	0	0	0	0	…
4	1	0	1	0	1	0	0	0	1	…
⋮										

Let $f(i) = b_{i1}b_{i2}b_{i3}b_{i4}b_{i5}\ldots$

Looking at f

	ε	0	1	00	01	10	11	000	001	…
1	0	1	0	1	0	1	0	1	0	…
2	0	1	0	1	1	1	0	0	0	…
3	0	0	1	0	0	0	0	0	0	…
4	1	0	1	0	1	0	0	0	1	…
⋮										

Let $f(i) = b_{i1}b_{i2}b_{i3}b_{i4}b_{i5}\ldots$

Let d, the diagonal language, be $b_{11}b_{22}b_{33}b_{44}\ldots = 0110\ldots$

Looking at f

	ε	0	1	00	01	10	11	000	001	…
1	0	1	0	1	0	1	0	1	0	…
2	0	1	0	1	1	1	0	0	0	…
3	0	0	1	0	0	0	0	0	0	…
4	1	0	1	0	1	0	0	0	1	…
⋮										

Let $f(i) = b_{i1} b_{i2} b_{i3} b_{i4} b_{i5} \ldots$
Let d, the diagonal language, be $b_{11} b_{22} b_{33} b_{44} \ldots = 0110\ldots$
Let $\overline{d}$, the anti-diagonal language, be $\overline{b_{11} b_{22} b_{33} b_{44}} \ldots = 1001\ldots$

Looking at f

	ε	0	1	00	01	10	11	000	001	…
1	0	1	0	1	0	1	0	1	0	…
2	0	1	0	1	1	1	0	0	0	…
3	0	0	1	0	0	0	0	0	0	…
4	1	0	1	0	1	0	0	0	1	…
⋮										

Let $f(i) = b_{i1}b_{i2}b_{i3}b_{i4}b_{i5}\ldots$

Let d, the diagonal language, be $b_{11}b_{22}b_{33}b_{44}\ldots = 0110\ldots$

Let $\overline{d}$, the anti-diagonal language, be $\overline{b_{11}b_{22}b_{33}b_{44}}\ldots = 1001\ldots$

$\overline{d} \notin range(f)$.

Looking at f

TBP: $\overline{d} \notin range(f)$.

FSOC:	$\overline{d} \in range(f)$.	
1.	$\exists z \in \mathbb{Z}+ \ni f(z) = \overline{d}$	Definition of range
2.	$f(z)_z = b$	$f(z)$ is a sequence of bits
3.	$\overline{d}_z = \overline{b}$	by construction
$\rightarrow\leftarrow$	$f(z)$ cannot equal $\overline{d}$	2. and 3.
$\therefore$	$\overline{d} \notin range(f)$	

f is **not** an onto function.

$\square$

$\mathbb{P}(\Sigma^*)$ is Uncountable

TBP: $\mathbb{P}(\Sigma^*)$ is *uncountable*.

TBP:	$\lvert\mathbb{P}(\Sigma^*)\rvert \neq \lvert\mathbb{Z}^+\rvert$	Countably Infinite
FSOC:	$\lvert\mathbb{P}(\Sigma^*)\rvert = \lvert\mathbb{Z}^+\rvert$	
1.	bijection $\exists\, f : \mathbb{Z}^+ \to \mathbb{P}(\Sigma^*)$	Same cardinality
2.	*f* can be represented as a table	
3.	The *f* in the table is not **onto**.	
$\rightarrow\leftarrow$	*f* both is and is not onto	1. and 3.
$\therefore$	$\lvert\mathbb{P}(\Sigma^*)\rvert \neq \lvert\mathbb{Z}^+\rvert$	
$\therefore$	$\mathbb{P}(\Sigma^*)$ is **uncountably** infinite	

$\square$

Halting Problem

Definition

The Halting Problem is the problem of constructing a program, H, that takes two parameters: P, another computer program and I, input for P. H should report "halts" or "loops forever" depending on whether or not P halts on input I.

$$H(P, I) = \begin{cases} \text{"halts"} & \text{if } P(I) \text{ halts} \\ \text{"loops forever"} & \text{if } P(I) \text{ does not halt} \end{cases}$$

Programs as Data

A program is encoded in some way (Fortran, pseudo-code, Java, bytecode). An encoding can be expressed as a sequence of symbols across some alphabet.
Any alphabet can be re-encoded using strings of bits.
Any program can be expressed as a sequence of bits.
A program, encoded as a sequence of bits, can be given as input to a program.
The receiving program may or may not respect the “right” interpretation.

Programs as Data

A program is encoded in some way (Fortran, pseudo-code, Java, bytecode). An encoding can be expressed as a sequence of symbols across some alphabet.
Any alphabet can be re-encoded using strings of bits.
Any program can be expressed as a sequence of bits.
A program, encoded as a sequence of bits, can be given as input to a program.
The receiving program may or may not respect the “right” interpretation.
Any program that takes a single input parameter can be passed itself (or rather, its own encoding) as its input.

Halting Problem

Definition

$$H(P, I) = \begin{cases} \text{"halt"} & \text{if } P(I) \text{ halts} \\ \text{"loop"} & \text{if } P(I) \text{ does not halt} \end{cases}$$

Halting Problem

Definition

$$H(P, I) = \begin{cases} \text{"halt"} & \text{if } P(I) \text{ halts} \\ \text{"loop"} & \text{if } P(I) \text{ does not halt} \end{cases}$$

Definition

FSOC Assume H exists. Construct D

$$D(P) = \begin{cases} \text{loop forever} & \text{if } H(D, P) \text{ halts} \\ \text{return} & \text{if } H(D, P) \text{ does not halt} \end{cases}$$

Halting Problem

Definition

$$H(P, I) = \begin{cases} \text{"halt"} & \text{if } P(I) \text{ halts} \\ \text{"loop"} & \text{if } P(I) \text{ does not halt} \end{cases}$$

Definition

FSOC Assume H exists. Construct D

$$D(P) = \begin{cases} \text{loop forever} & \text{if } H(D, P) \text{ halts} \\ \text{return} & \text{if } H(D, P) \text{ does not halt} \end{cases}$$

What does $H(D, D)$ return?