2 Set Theory and Complexity

For many areas of mathematics a long process can usually be traced in which ideas evolve until an ultimate flash of inspiration, often by a number of mathematicians almost simultaneously, produces a discovery of major importance. Set theory however is rather different. Its basis is the creation of one person, Georg Cantor. In his seminal 1874 paper Cantor observes different kinds of infinity. Before this, orders of infinity did not exist but all infinite collections were considered "the same size". Cantor showed that this is not true. In particular, he showed that the cardinality of the set of real numbers is larger than the cardinality of the set of natural numbers. We will repeat his proof here.

2.1 Infinite sets

Recall that the *cardinality* of a set is the number of members it contains. When S is an infinite set, $|S|$ will be an infinite number. The cardinality of some infinite set is called a *transfinite number* or *transfinite cardinal*.

Two sets can be put into *one-to-one correspondence* if and only if their members can be paired off such that each member of the first set has exactly one counterpart in the second set, and each member of the second set has exactly one counterpart in the first set, or formally, there is a bijective mapping from one to another. If sets A and B can be put into one-to-one correspondence, then we say $A \simeq B$.

We say that the cardinality of a set A is *at least as large* as the cardinality of set B, or $|A| \geq |B|$, if and only if some subset of A and the whole of B can be put into one-to-one correspondence. Or equivalently, $|A| \geq |B|$ if and only if there is a surjective mapping from A to B. A and B are defined to have the *same cardinality*, or $|A| = |B|$, if and only if $|A| \geq |B|$ and $|B| \geq |A|$. It is not difficult to check that with these definitions "=" forms an equivalence relation and "$\geq$" forms an order. Furthermore, it is easy to prove the following theorem.

Theorem 2.1 *Let A and B be arbitrary sets. If $A \simeq B$ then $|A| = |B|$.*

The most important sets are

- $\mathbb{N} = \{1, 2, 3, 4, \ldots\}$: set of *natural numbers*
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$: set of *integers*
- $\mathbb{Q} = \{x/y : x \in \mathbb{Z} \text{ and } y \in \mathbb{N}\}$: set of *rational numbers*
- $\mathbb{R}$: set of *real numbers*

Given a set of numbers S, S_+ denotes the set of all positive numbers (i.e. > 0) in S, and S_- denotes the set of all negative numbers (i.e. < 0) in S.

It is tempting to conclude that

$$|\mathbb{N}| < |\mathbb{Z}| < |\mathbb{Q}| < |\mathbb{R}| .$$

However, we will show that this is not true.

Theorem 2.2 $|\mathbb{N}| = |\mathbb{Z}|$.

Proof. In order to show that $|\mathbb{N}| \geq |\mathbb{Z}|$ consider the mapping $f : \mathbb{N} \to \mathbb{Z}$ with $f(x) = (-1)^x \cdot \lfloor x/2 \rfloor$. f is surjective, because for every $y \in \mathbb{Z}$, $x = 2 \cdot |y| + (|y| - y)/(2|y|)$ (if $y \neq 0$ and otherwise $x = 1$) fulfills $f(x) = y$. Using the mapping $g : \mathbb{Z} \to \mathbb{N}$ with $g(x) = 1 + |x|$, it is straightforward to see that also $|\mathbb{Z}| \geq |\mathbb{N}|$. Thus, we must have $|\mathbb{N}| = |\mathbb{Z}|$. □

Theorem 2.3 $|\mathbb{Z}| = |\mathbb{Q}|$.

Proof. Obviously, $\mathbb{Q} \simeq \mathbb{Z} \times \mathbb{N}$. Using the mapping $f : \mathbb{Z} \times \mathbb{N} \to \mathbb{Z}$ with $f(x, y) = x$, it follows that $|\mathbb{Q}| \geq |\mathbb{Z}|$. Hence, it remains to show that $|\mathbb{Z}| \geq |\mathbb{Q}|$. The following table illustrates Cantor's method to obtain this inequality.

		Numerators				
		1	2	3	4	
	1	1/1	2/1	3/1	4/1	...
Denomi-	2	1/2	2/2	3/2	4/2	...
nators	3	1/3	2/3	3/3	4/3	...
	4	1/4	2/4	3/4	4/4	...
			...			

Obviously, every positive rational number can be found in this table. We can enumerate all these numbers by moving diagonally through the table: 1/1, 2/1, 1/2, 3/1, 2/2, 1/3, ... Assigning 1/1 to the value 1, 2/1 to the value 2, 1/2 to the value 3, and so on, we obtain a surjective mapping g from $\mathbb{Z}_+$ to $\mathbb{Q}_+$. Using the same strategy for the mapping from $\mathbb{Z}_-$ to $\mathbb{Q}_-$ and mapping 0 to 0 we arrive at a surjective mapping from $\mathbb{Z}$ to $\mathbb{Q}$. Hence, $|\mathbb{Z}| = |\mathbb{Q}|$. □

Since equality is transitive, Theorems 2.2 and 2.3 immediately yield the following result.

Corollary 2.4 $|\mathbb{N}| = |\mathbb{Q}|$.

A generalization of the enumeration technique in the proof of Theorem 2.3 also yields the following result.

Theorem 2.5 *For every $k \in \mathbb{N}$, $|\mathbb{N}| = |\mathbb{N}^k|$.*

In the light of these results, it might be tempting to believe now that all infinite sets are of the same cardinality. However, the next two theorems show that this is not true.

Theorem 2.6 $|\mathbb{R}| = |2^{\mathbb{N}}|$.

Proof. First we show that $|\mathbb{R}| \geq |2^{\mathbb{N}}|$. Given a number $x \in \mathbb{R}$, let $b(x)$ be the binary representation of $|x| - \lfloor |x| \rfloor$, i.e. $|x| - \lfloor |x| \rfloor = \sum_{i \geq 1} b_i(x)/2^i$. Furthermore, given a (possibly infinite) binary string w, let $S_w \in 2^{\mathbb{N}}$ be the set that contains exactly those natural numbers i with $w_i = 1$ (w_i represents the ith digit in w). Because all sets $S \in 2^{\mathbb{N}}$ have a unique binary string representing it and every binary string can be represented as a unique real number (by changing the basis from 2 to 10), $f : \mathbb{R} \to 2^{\mathbb{N}}$ with $f(x) = S_{b(x)}$ is a surjective mapping.

It remains to show that $|2^{\mathbb{N}}| \geq |\mathbb{R}|$. In order to obtain a surjective mapping from $2^{\mathbb{N}}$ to $\mathbb{R}$, we fist transform a set $S \in 2^{\mathbb{N}}$ into a binary string $w(S)$ in a way that the ith digit of $w(S)$ is 1 if and only if $i \in S$. Next, we transform a binary string w into a real number x_w in a way that w_1 encodes the sign of x, all w_i with even i encode the value of x left from the decimal point, and all w_i with odd $i > 1$ encode the value of x right from the decimal point. Since these transformations ensure that every real number can be represented as a unique binary string and every binary string can be represented as a unique subset of $\mathbb{N}$, $g(S) = x_{w(S)}$ is a mapping that is surjective on $\mathbb{R}$. □

Theorem 2.7 $|\mathbb{N}| < |2^{\mathbb{N}}|$.

Proof. We give a negative proof and assume that there is a surjective mapping from $\mathbb{N}$ to $2^{\mathbb{N}}$. In this case, fix any of these mappings. Let this hypothetical mapping be represented by the leftmost column of the following table (that is, number i is mapped to set S_i).

Is the number in the set?	natural numbers				
	1	2	3	4	
S_1	**yes**	no	yes	no	...
S_2	no	**yes**	no	yes	...
S_3	yes	yes	**no**	no	...
			...		

Since every row i of "yeses" and "noes" uniquely determines set S_i, we can read any row of "yeses" and "noes" as a code for a particular set of natural numbers.

Now look at the diagonal formed by the "yeses" and "noes" in bold face. Turn any "yes" along the diagonal into a "no" and vice versa. The resulting infinite string is demonstrably different from every string listed on the table, because for every row i it differs from the string in this row at least at position i. But this means that the set represented by the negation of the diagonal has not been assigned to any of the natural numbers. This contradicts our assumption that we exhaustively listed all the sets of natural numbers. Therefore, our assumption that the selected mapping is surjective is false, and there is no surjective mapping from $\mathbb{N}$ to $2^{\mathbb{N}}$. Hence, $|\mathbb{N}| < |2^{\mathbb{N}}|$. □

The method used in the proof above is called *diagonalization* and has been used in many other contexts to prove important results. We will come back to it later.

Combining Theorem 2.6 and Theorem 2.7, we obtain together with the previous theorems that

$$|\mathbb{N}| = |\mathbb{Z}| = |\mathbb{Q}| < |\mathbb{R}| .$$

Moreover, Theorems 2.5 and 2.7 imply that for every $k \in \mathbb{N}$, $|\mathbb{N}^k| < |\bigcup_{i \geq 0} \mathbb{N}^i|$.

One may ask whether there is a set S with $|\mathbb{N}| < |S| < |\mathbb{R}|$. This was actually the first problem on David Hilbert's famous 1900 list of important unsolved problems in mathematics. The *continuum hypothesis* asserts that there is no such set S. In 1938, Gödel showed that the continuum hypothesis cannot be disproved from the axioms of Zermelo-Fraenkel (ZF) set theory (the closest thing we have to "standard" set theory). Finally, in 1963 Paul Cohen showed that the continuum hypothesis also cannot be proved from the ZF axioms. Thus, the continuum hypothesis is *undecidable* in ZF.

One can generalize Theorem 2.7 to the following result (this will be an assignment).

Theorem 2.8 *For every set A it holds that $|A| < |2^A|$.*

Theorem 2.8 implies there is an infinite series of transfinite cardinals. These cardinals are denoted by the hebrew letter aleph with subscripts numbered 0,1,2,.... $|\mathbb{N}|$ is defined as $Aleph_0$ and $|\mathbb{R}|$ is defined as $Aleph_1$. $|\mathbb{N}|$ was chosen to represent the "smallest" infinite set, because it can be shown that every infinite set has a cardinality that is at least as large as $|\mathbb{N}|$.

2.2 Paradoxes in set theory

Cantor continued to publish a six part treatise on set theory from the years 1879 to 1884. This work appears in *Mathematische Annalen* and it was a brave move by the editor to publish the work despite a growing opposition to Cantor's ideas. The leading figure in the opposition was Kronecker who was an extremely influential figure in the world of mathematics at that time. However, even Cantor had his problems with his new theory. He is believed to be the first that discovered paradoxes in the set theory, although the first documented paradox is from 1897, published by Cesare Burali-Forti. In 1899 Cantor discovered a different paradox that arises from the set of all sets. What is the cardinality of the set of all sets? Clearly, it must be the greatest possible cardinal. Yet, as we showed in the previous section, the cardinal of the set of all subsets of a set always has a greater cardinal than the set itself. It began to look as if the criticism of Kronecker might be at least partially right, since the extension of the set concept too far seemed to be producing the paradoxes. The "ultimate" paradox was found by Russel in 1902 (and independently by Zermelo). It simply uses the set

$$A = \{X : X \text{ is not a member of } X\} .$$

Russel then asked: *Is A an element of A?* Both the assumption that A is a member of A and A is not a member of A lead to a contradiction. The set construction itself appears to give a paradox. So what went wrong?

First of all, when defining A, we implicitly assumed that A exists. Hence, the paradox simply represents a proof via contradiction that A cannot exist! However, there seems to be no problem with the complement of A, namely

$$B = \{X : X \text{ is a member of } X\} .$$

So is set theory not closed under complement? To solve the problem, let us specify the domain of the X, for instance

$$B = \{X \in \mathbb{N} : X \text{ is a member of } X\} .$$

In this case, obviously $B = \emptyset$. Hence,

$$A = \{X \in \mathbb{N} : X \text{ is not a member of } X\} = \mathbb{N} .$$

Now, check the paradox again:

- $A \in A \overset{def.}{\Rightarrow} A \notin A$: this argument is not valid any more, because A is not a member of A (resp. $\mathbb{N} \notin \mathbb{N}$).
- $A \notin A \overset{def.}{\Rightarrow} A \in A$: $A \notin A$ is true, but since $A \notin \mathbb{N}$, it cannot be concluded from this that $A \in A$ (notice that only the elements $X \in \mathbb{N}$ can be members of A!).

Thus, once we specify the domain of the X, we can nicely resolve the paradox. It also reveals that A can never be part of the domain from which the X are chosen (replacing $\mathbb{N}$ by any other domain yields the same result as for $\mathbb{N}$). Hence, if X can be an arbitrary set, then A cannot exist!

Some mathematicians did not like the fact of being able to define sets that cannot exist. This caused on the one hand the attempt to axiomatize set theory in a way that prevents the occurance of paradoxes and on the other hand the rejection of proofs via contradiction by a group of mathematicians (which are called the "constructivists"). The attempt to axiomatize set theory actually failed (and it is known today why it had to fail). Although paradoxes may seem strange when confronted with them for the first time, they are actually not unique to set theory. Define, for instance, x to be the largest natural number. Obviously, x cannot exist, because $x + 1$ is always a number that is larger than x. Thus, we can also define non-existing objects in number theory.

2.3 Consequences for complexity theory

Before we use our knowledge in set theory to prove results in the area of complexity, we start with some basic properties of recursive and recursively enumerable languages.

Properties of recursive and recursively enumerable languages

We start with two lemmas.

Lemma 2.9 *The complement of a recursive language is recursive.*

Proof. Suppose that L is recursive. Then there is a Turing machine M that decides it. Reversing the answers of M, we arrive at a Turing machine that decides $\bar{L}$. Hence, $\bar{L}$ is also recursive. □

Lemma 2.10 *If a language L and its complement $\bar{L}$ are both recursively enumerable, then L (and also $\bar{L}$) is recursive.*

Proof. Let M_1 be the Turing machine that accepts L and M_2 be the Turing machine that accepts $\bar{L}$. We construct out of M_1 and M_2 a Turing machine M that decides L in the following way.

Given an input x, M simulates simultaneously M_1 started on x and M_2 started on x. If M_1 accepts x, M accepts x and halts. If M_2 accepts x, M halts without accepting x.

Obviously, x must be either in L or in $\bar{L}$. That is, the only two cases that can happen are that x is accepted by M_1 and not by M_2, or that x is accepted by M_2 and not by M_1. Thus, M

will always either accept or reject, but will never do both. Furthermore, in both cases M halts. Since M accepts L, M therefore decides L. □

The proof of this theorem uses a concept we have not explicitly shown previously: that a Turing machine can simulate another Turing machine. We just give an intuition here why this is true. Obviously, it is easy to write an algorithm that simulates any given Turing machine: given the encoding of a Turing machine, the algorithm creates a data structure for its finite state control and the tape and then simulates the Turing machine step by step. Since every algorithm (based on known computational models) can be simulated by a Turing machine, there must also be a Turing machine that can simulate any given Turing machine (for more detailed information, please consult the book by Hopcroft and Ullman).

Let L and $\bar{L}$ be a pair of complementary languages. Then the two previous lemmas imply that either

1. both L and $\bar{L}$ are recursive,
2. neither L nor $\bar{L}$ are recursively enumerable, or
3. one of L and $\bar{L}$ is recursively enumerable but not recursive; the other is not recursively enumerable.

Obviously, languages exist for case 1. The other two cases will be studied below.

We conclude the section about basic properties with the following lemma.

Lemma 2.11 *The union of two recursive languages is recursive.*

Proof. Let L_1 and L_2 be recursive languages. Furthermore, let M_1 be the TM that decides L_1 and M_2 be the TM that decides L_2. We construct out of M_1 and M_2 a Turing machine M for $L_1 \cup L_2$ in the following way:

Given some input x, M first simulates M_1 started on x. If M_1 accepts, also M accepts. Otherwise, M simulates M_2 started on x. If M_2 accepts, also M accepts. Otherwise M halts without accepting.

Going through all cases, it can be seen that M accepts exactly those inputs x with $x \in L_1 \cup L_2$. Furthermore, since M_1 and M_2 halt for all inputs, also M halts for all inputs. Hence, M decides $L_1 \cup L_2$. □

Similar results can be shown for the intersection and other basic set operations. Thus, recursive languages are closed under basic set operations. (Assignment: show that there are basic set operations (negation!) for which this does NOT holds for r.e. languages!)

The limits of computability

In the following, let $\mathcal{L}$ be the set of all languages, $\mathcal{L}_{re}$ be the set of all recursively enumerable languages, and $\mathcal{L}_r$ be the set of all recursive languages. A fundamental question has been, whether $\mathcal{L}_{re} = \mathcal{L}$ or not. Using our results from set theory, it will turn out to be quite easy to answer.

We start with fixing a representation for $\mathcal{L}$. Recall that every decision problem can be represented as a language $L \subseteq \{0,1\}^*$ (L contains all binary encodings of the inputs that have the answer “yes”). Hence, the set of all possible decision problems can be represented as $2^{\{0,1\}^*}$. Using a suitable binary encoding strategy, every Turing machine can be uniquely represented as a finite binary string. In the following, we denote the binary encoding of a Turing machine M by $\langle M \rangle$. Let $\mathcal{M} \subseteq \{0,1\}^*$ denote the set of all binary strings that represent encodings of Turing machines. Then the mapping $f : \mathcal{M} \to \mathcal{L}_{re}$ with $f(\langle M \rangle) = L(M)$ is surjective. Hence, $|\mathcal{M}| \geq |\mathcal{L}_{re}|$. Furthermore, $|\{0,1\}^*| \geq |\mathcal{M}|$, because a subset of $\{0,1\}^*$ can certainly be brought into one-to-one correspondance with $\mathcal{M}$ (recall the original definition of “$\geq$”). Now observe that according to Theorem 2.8 we have $|2^{\{0,1\}^*}| > |\{0,1\}^*|$. Combining all the inequalities, we obtain $|2^{\{0,1\}^*}| > |\mathcal{L}_{re}|$ and therefore $|\mathcal{L}| > |\mathcal{L}_{re}|$. What we actually showed by this is that there cannot be a surjective mapping from $\mathcal{L}_{re}$ to $\mathcal{L}$ (because otherwise we could construct a surjective mapping from $\{0,1\}^*$ to $2^{\{0,1\}^*}$, contradicting Theorem 2.8). Hence, the set of recursively enumerable languages cannot cover the set of all possible languages. Or in other words:

Theorem 2.12 *There is a decision problem that does not have a Turing machine that accepts it.*

Hence, there are non-TM computable decision problems resp. non-recursively enumerable languages. The recursively enumerable languages are therefore a proper subset of the set of all languages.

Next we study the relationship between recursive and recursively enumerable languages. Consider the following language:

$$L_d = \{\langle M \rangle : \ M \text{ started with } \langle M \rangle \text{ does not accept}\} .$$

Note that L_d exists and is a language (otherwise the next theorem would be trivial!). Similar to Russel's paradox in set theory, we can show the following result.

Lemma 2.13 *L_d is not recursive.*

Proof. Suppose on the contrary that L_d is recursive. Then there would be a Turing machine M that decides L_d. What would M do when started with itself? We distinguish between two cases.

1. M accepts $\langle M \rangle$: then, according to the definition of L_d, M does not accept $\langle M \rangle$.

2. M does not accept $\langle M \rangle$: then, according to the definition of L_d, M accepts $\langle M \rangle$.

Since for both cases we arrive at a contradiction, our assumption that there is a Turing machine that decides L_d must be false. □

Combining Lemma 2.13 and Lemma 2.9, we obtain the following result.

Corollary 2.14 *$\bar{L}_d$ is not recursive.*

Note that $\bar{L}_d$ is defined as

$$\begin{aligned}\bar{L}_d &= \{0,1\}^* - L_d \\ &= \{x \in \{0,1\}^* : \ x \text{ is not an encoding of a TM}\} \cup \\ &\quad\ \{\langle M\rangle : \ M \text{ started with } \langle M\rangle \text{ accepts}\} .\end{aligned}$$

We prove now the following theorem.

Theorem 2.15 *$\bar{L}_d$ is recursively enumerable.*

Proof. The Turing machine M that accepts $\bar{L}_d$ works as follows. First, it checks whether the input x is an encoding of a Turing machine. If not, M accepts. Otherwise, assume that x is the encoding of Turing machine M'. Then M simulates M' started with x. If M' accepts, then also M accepts. Obviously, M accepts exactly those inputs x with $x \in \bar{L}_d$. Hence, $\bar{L}_d$ is recursively enumerable. $\square$

Corollary 2.14 and Theorem 2.15 imply that there is a language that is recursively enumerable but not recursive. Hence, the set of recursive languages is a proper subset of the set of recursively enumerable languages. Furthermore, combining Corollary 2.14 and Theorem 2.15 with Lemma 2.10, we obtain that L_d is not only non-recursive, but also non-recursively enumerable.

To summarize the results we obtained so far, we have

$$\mathcal{L}_r \subset \mathcal{L}_{re} \subset \mathcal{L} .$$

Furthermore, we explicitly constructed a language that is non-recursive but recursively enumerable and a language that is non-recursively enumerable. Are there also languages L where both L and $\bar{L}$ are non-recursively enumerable? The next theorem shows that the answer is "yes".

Theorem 2.16 *There is a language L for which both L and $\bar{L}$ are not recursively enumerable.*

Proof. In order to prove the theorem, we use Cantor's diagonalization method. Recall that the set of all languages is $2^{\{0,1\}^*}$ and that the set of all binary encodings of Turing machines is equal to $\mathcal{M} \subseteq \{0,1\}^*$. Now, consider the two sets $\mathcal{M} \times \{0,1\}$ and $\mathcal{L}_{rec} = \{L \in 2^{\{0,1\}^*} : \ L \in \mathcal{L}_{re}$ or $\bar{L} \in \mathcal{L}_{re}\}$. In words, $\mathcal{L}_{rec}$ contains the set of all languages L for which either L or $\bar{L}$ is recursively enumerable. Obviously, the mapping $f : \mathcal{M} \times \{0,1\} \rightarrow \mathcal{L}_{rec}$ with $f(\langle M\rangle c) = L(M)$ if $c = 0$ and $\overline{L(M)}$ otherwise is surjective. Hence, $|\mathcal{M} \times \{0,1\}| \geq |\mathcal{L}_{rec}|$. Furthermore, $|\{0,1\}^*| \geq |\mathcal{M} \times \{0,1\}|$, because a subset of $\{0,1\}^*$ can obviously brought into one-to-one correspondance with $\mathcal{M} \times \{0,1\}$. Since $|2^{\{0,1\}^*}| > |\{0,1\}^*|$, we conclude that $|2^{\{0,1\}^*}| > |\mathcal{L}_{rec}|$. In other words, there can be no surjective mapping from $\mathcal{L}_{rec}$ to $\mathcal{L}$. Hence, there must be languages L for which neither L nor $\bar{L}$ are recursively enumerable. $\square$

Oracle computations

One is tempted to ask what would happen if a certain undecidable problem were decidable? Could we then compute everything? To answer the question we must be careful. If we start out by assuming that some undecidable problem is decidable, we have a contradictory set of assumptions and may conclude anything. We avoid this problem by defining a Turing machine with an oracle.

Let O be a language, i.e. $O \subseteq \{0,1\}^*$. A *Turing machine with oracle* O is a Turing machine with three special states: $q_?$, q_y, and q_n. The state $q_?$ is used to ask whether a string is in the set O. When the Turing machine enters state $q_?$ it requests an answer to the question: "Is the string of nonblank symbols to the right of the tape head in O?" The answer is supplied by having the state of the Turing machine change on the next move to one of the two states q_y or q_n, depending on whether the answer is yes or no. The computation continues normally until the next time $q_?$ is entered, when the "oracle" answers another question.

Observe that if O is a recursive set, then the oracle can be simulated by another Turing machine, and the set accepted by the Turing machine with oracle O is always recursively enumerable. On the other hand, if O is not a recursive set and an oracle is available to supply the correct answer, then the Turing machine with oracle O may be able to accept a set that is not recursively enumerable. We denote a Turing machine M with oracle O be M^O. A set L is called *recursively enumerable with respect to* O if $L = L(M^O)$ for some Turing machine M. A set L is *recursive with respect to* O if $L = L(M^O)$ for some Turing machine M^O that always halts. Two oracles are *equivalent* if each is recursive in the other.

Given some oracle O, let $\mathcal{L}_{re}^O$ be the set of all languages that are recursively enumerable with respect to O. Then we can show the following theorem.

Theorem 2.17 *For every oracle* O, $\mathcal{L}_{re}^O \subset \mathcal{L}$.

Proof. Obviously, $\mathcal{L}_{re}^O \subseteq \mathcal{L}$. Thus, it remains to prove that there is a language that is not in $\mathcal{L}_{re}^O$.

Consider some fixed oracle O. Let us extend the alphabet for the encoding of a Turing machine from $\{0,1\}$ to $\{0,1,q_?,q_y,q_n\}$. This alphabet is used in a way that every Turing machine M is encoded in the standard way up to the states $q_?$, q_y, and q_n, for which the new symbols are used. Let $\mathcal{M}^O \subseteq \{0,1,q_?,q_y,q_n\}^*$ be the set of all possible encodings of Turing machines using or not using oracle O. Obviously, the function $f : \mathcal{M}^O \to \mathcal{L}_{re}^O$ with $f(\langle M^O \rangle) = L(M^O)$ is surjective. Hence, $|\mathcal{M}^O| \geq |\mathcal{L}_{re}^O|$. Furthermore, since $\mathcal{M}^O \subseteq \{0,1,q_?,q_y,q_n\}^*$, we have $|\{0,1,q_?,q_y,q_n\}^*| \geq |\mathcal{M}^O|$. Since $\{0,1,q_?,q_y,q_n\}^* \simeq \{000,001,100,101,110\}^* \subseteq \{0,1\}^*$, it also holds that $|\{0,1\}^*| \geq |\{0,1,q_?,q_y,q_n\}^*|$. Because $|2^{\{0,1\}^*}| > |\{0,1\}^*|$, it follows that $|2^{\{0,1\}^*}| > |\mathcal{L}_{re}^O|$, and therefore there must be languages that are not in $\mathcal{L}_{re}^O$. □

Theorem 2.17 also holds for multiple oracles. Hence, any finite number of oracles is still not enough to ensure that all languages can be accepted.

2.4 References

- P. Suber. A crash course in the mathematics of infinite sets. In the *St. John's Review* XLIV, 2, pp. 35-59, 1998. See also http://www.earlham.edu/~peters/writing/infapp.htm.
- J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, Reading, 1979.