UNDECIDABILITY AND THE STRUCTURE OF THE TURING DEGREES

PHILIP ADAMS

Abstract. This paper explores the structure and properties of the Turing degrees, or degrees of undecidability. We introduce the Turing machine, an abstract model of computation, in order to develop the concepts of undecidability and Turing reduction. We demonstrate the technique of proof by reduction through a series of examples of undecidable problems related to context-free grammars. We then employ reducibility to consider a partial ordering on the set of Turing degrees, $\mathcal{D}$. Finally, we prove a variety of theorems related to the structure of $\mathcal{D}$.

Contents

1. Introduction

Historical overview of the field, motivation for study [1] [20] [19] [11]

1.1. Decision Problems.

2. Turing Machines

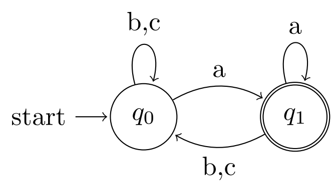
Upon consideration of different types of problems, it quickly becomes clear that some are much more difficult than others. For example, when considering words over the alphabet $\{a, b, c\}$, it seems much easier to determine whether a word ends in an a than whether a word is a palindrome. It seems even harder still to determine

Date: August 11, 2018.

whether the word is of the form $a^n b^n c^n$. In order to formalize these notions of difficulty, we need to build abstract models of computation, and then test their ability to decide such questions. We observe that a machine that can solve the first problem only needs to "remember" the same amount of information no matter how long the input is: the last letter of the word. In contrast, the amount of memory that the second and third problems require is dependent on the size of the input, because the first half of the palindrome or n, respectively, can be arbitrarily large. The difference between the second and third problems is more subtle: the second problem can be solved moving only one direction in memory, while the third problem requires the ability to "look back" in memory.

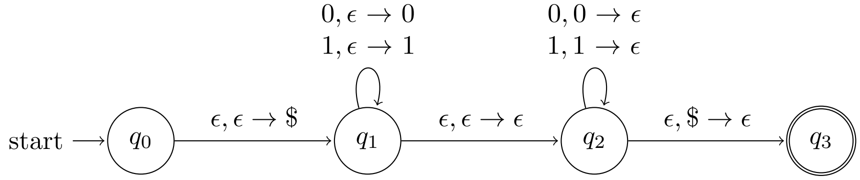
Based on these differences, we can begin to construct different models of computation that have just enough power to solve each problem. The first problem can be solved by a *Finite State Automaton (FSA)*, a collection of a finite number of states, a transition function, and a set of accepting states. The FSA accepts the input if the sequential application of each element to the input finishes in an accepting state, and rejects otherwise. Finite State Automata are often represented by charts as in Figure 1, a representation of an FSA that decides the first problem.

FIGURE 1. An FSA that determines whether a string ends in 'a'.



The second problem can be solved by a machine called a *Pushdown Automaton (PDA)*, which is essentially a finite state automaton with the addition of a stack. Rather than transforming the current state and the input into a new state, as with a finite state automata, the transition function of a PDA also considers the stack, which it can pop and push from. Within the collection of PDAs, there is another division between *Deterministic Pushdown Automata (DPDAs)* and *Nondeterministic Pushdown Automata*. For DPDAs, the transition function only outputs one move, rather than a set of moves. Nondeterministic PDAs are able to recognize more languages than DPDAs. Figure 2 is a graphical representation of a PDA that decides palindromes. In this representation, ϵ is an empty input, and is a special symbol that denotes the bottom of the stack. The moves are represented by a double of input and stack operation.

FIGURE 2. A Pushdown Automaton that decides palindromes over the alphabet a, b



Finally, we arrive at the Turing Machine (TM), devised by Alan Turing in his paper *On Computable Numbers, With an Application To the Entscheidungsproblem* [22]. Turing Machines offer an unrestricted model of computation, Discussion of TMs

3. Undecidability

Problem 3.1 (The Halting Problem)**.** *[22] [18]*

Theorem 3.2 (Rice's Theorem)**.** *[10]*

3.1. Incompleteness.

Theorem 3.3 (Gödel's Incompleteness Theorem)**.** *[7]*

3.2. **Reducibility.** While it is possible to prove that a problem is undecidable directly, as in problem 3.1, it is often more convenient to prove undecidability through comparison to problems which are already known to be undecidable. This comparison takes place through the technique of Turing reduction.

Definition 3.4. Let A and B be decision problems. We say that A is *Turing reducible* to B and write $A \leq_T B$ if, given a machine D_B that decides B, it is possible to construct a machine D_A that decides A.

So, we have that if A is reducible to B, then A can be no harder than B, because any solution to B also leads to a solution of B. So, if we have some problem B that we would like to prove is undecidable, we can do so by showing that some problem A which is already known to be undecidable is reducible to B. Since A cannot be harder than B, it follows that B must also be undecidable. Alternatively, if we would like to show that some problem A is decidable, it is sufficient to show that it is reducible to some problem B that is known to be decidable. [18] [12]

[9]

4. Context Free Grammars

Definition 4.1. A Context Free Grammar is...

Definition 4.2. A Pushdown Automata is...

Theorem 4.3 (Equivalence of CFGs and Pushdown Automata)**.** *[6]*

Theorem 4.4 (Closure Properties)**.** *[18] Union, Concat, Star, intersection with regular, substitution*

Problem 4.5 (Emptiness)**.** *Let G be a context-free grammar. Is $L(G) = \varnothing$?*

Problem 4.6 (Finite)**.** *Let G be a context-free grammar. Is $L(G)$ finite?*

Problem 4.7 (Regular Containment)**.** *Take some context-free grammar G and some regular language R. Is $L(G) \subseteq R$?*

Proof of Decidability. We have that $L(G) \subseteq R$ if and only if $\overline{L(G)} \cup R = \Sigma^*$. So, it follows that $L(G)$ is contained by R if and only iff $\overline{\overline{L(G)} \cup R} = \varnothing$. By DeMorgan's laws, we have that this is equivalent to the statement $L(G) \cap \overline{R} = \varnothing$. Regular languages are closed under complement and context-free languages are closed under intersection with a regular language, so it follows that $L(G) \cap \overline{R}$ is a context free language. So the problem of containment by a regular language is reducible to Problem 4.5, the problem of emptiness, which is decidable. So, the problem of containment by a regular language is decidable. ∎

4.1. **Undecidable Problems.**

Problem 4.8 (The Post Correspondence Problem)**.** *Consider some alphabet Σ and two finite lists of words over Σ denoted $A = a_1, \ldots, a_n$ and $B = b_1, \ldots, b_n$. Then, does there exist some sequence of indices (i_k), for $1 \leq k \leq K$ and for $K \geq 1$, and with $1 \leq i_k \leq n$ for all k such that*

$$a_{i_1} \ldots a_{i_K} = b_{i_1} \ldots b_{i_K}?$$

Solution. This problem is shown to be undecidable through a reduction to the halting problem. The proof is technical, and involves encoding the computation history of a Turing machine on an input in such a way that the encoding satisfies the PCP if and only if the Turing machine accepts the input. It is described in detail by Sipser in his book *Introduction to the Theory of Computation* [18]. □

The Post correspondence problem is a useful tool, because it allows us to demonstrate the undecidability of problems without doing the complex reasoning about Turing machines that the halting problem requires. We will now show the undecidability of a variety of problems about context-free grammars through reduction to the Post correspondence problem.

Problem 4.9 (Disjointness)**.** *Let P, Q be context-free grammars. Is $L(P) \cap L(Q) = \varnothing$?*

Proof of Undecidability. Consider the Post correspondence problem for two lists of words A, B. We construct two context free grammars from these lists as follows:

$$\begin{array}{ll}
G_A \to a_1 1 & G_B \to a_1 1 \\
\vdots & \vdots \\
G_A \to a_n n & G_B \to a_n n \\
G_A \to a_1 G_A 1 & G_B \to a_1 G_B 1 \\
\vdots & \vdots \\
G_A \to a_n G_A n & G_B \to a_n G_B n
\end{array}$$

Then, we can observe that for some string s to exist in both $L(G_A)$ and $L(G_B)$, it must be a solution to the Post correspondence for A, B. So, if $L(G_A) \cap L(G_B) = \varnothing$, then there are no solutions to the Post correspondence for A, B. So, it follows that the Post correspondence problem is reducible to the problem of the disjointness of context-free grammars, so since the Post correspondence problem is undecidable, it follows that the disjointness problem is undecidable. □

Problem 4.10 (Universality)**.** *Let G be some context-free grammar over an alphabet Σ. Then, is*

$$L(G) = \Sigma^*?$$

Proof of Undecidability.

□

Definition 4.11. An ambiguous grammar is...

Problem 4.12 (Ambiguity)**.** *Let G be a context-free grammar. Is G ambiguous?*

Proof of Undecidability. Consider the Post correspondence problem for two lists of words A, B. Construct their corresponding grammars G_A, G_B. Now, consider the grammar

$$G \to G_A | G_B.$$

It follows that the ambiguity of G implies that a solution to the Post correspondence problem for A, B exists, so the Post correspondence problem reduces to the ambiguity problem, so the ambiguity problem is undecidable. □

[4] [5]

Problem 4.13. *Let G be a context free grammars. Is $L(G)$ regular?*

Proof of Undecidability. Note that Σ^* is regular, so if $L(G)$ is not regular then it follows that $L(G) \neq \Sigma^*$. Additionally, universality is decidable within regular languages, because they are closed under complement and emptiness is decidable even within context-free grammars. So, we have that Problem 4.10 reduces to the regularity problem. It follows that the regularity problem is undecidable. □

Problem 4.14 (Equality)**.** *Let G_1, G_2 be context free grammars. Is $L(G_1) = L(G_2)$?*

Proof of Undecidability. Note that Σ^* is regular, so it is also a context-free. So, we have that Problem 4.10 reduces to the equality problem. It follows that the equality problem is undecidable. □

Problem 4.15 (Inclusion)**.** *Let G_1, G_2 be context free grammars. Is $L(G_1) \subseteq L(G_2)$?*

Proof of Undecidability. As before, note that Σ^* is regular, so it is also a context-free. Additionally, observe that for any grammar G, $L(G) \subseteq \Sigma^*$. So, we have that Problem 4.10 reduces to the inclusion problem, since $\Sigma^* \subseteq L(G) \implies \Sigma^* = L(G)$. It follows that the inclusion problem is undecidable. □

4.1.1. *Problems decidable for Deterministic CFLs.* [3]

Definition 4.16. A deterministic context-free language is a language accepted by a deterministic pushdown automata.

Theorem 4.17 (Closure Properties)**.** *If L is a deterministic context-free language, then $\overline{L}$ is a deterministic context-free language.*

Problem 4.18. *Let L be a deterministic context-free language. Is $L = \Sigma^*$?*

The decidability of the following problem was an open problem in the field of computability theory from 1965, when it was introduced by Ginsburg and Greibach [3]. In 1997, when it was solved by Géraud Sénizergues [15]. A sketch of the proof follows.

Problem 4.19 (The Equivalence Problem for Deterministic CFGs)**.** *Let L_1, L_2 be deterministic context-free languages. Is $L_1 = L_2$?*

5. Turing Degrees

In Subsection 3.2, we introduced the concept of Turing reducibility, denoted by the symbol $\leq_T$. This symbol suggests some type of ordering over the set of decision problems. In this section, we introduce that partial ordering on the set of decision problems and examine its properties.

Definition 5.1. Let A and B be decision problems. We say that A and B are *mutually reducibile* and write $=_T$ if $A \leq_T B$ and $B \leq_T A$.

Fact 5.2. The relation $=_T$ is an equivalence relation.

We call the equivalence classes produced by this relation *Turing degrees* or *degrees of unsolvability*, a concept first introduced in 1944 by Emil Post [12]. We write the set of all such degrees as $\mathcal{D}$.

Lemma 5.3. *The relation $\leq_T$ is a partial ordering of $\mathcal{D}$.*

Definition 5.4. Computably Enumerable

Definition 5.5. Completeness

5.1. Properties and Structure.

Lemma 5.6. *The set $\mathcal{D}$ is of cardinality $2^{\aleph_0}$.*

Lemma 5.7. *Every Turing degree contains countably infinite elements.*

While it is simple to explicitly state problems lying in the lower degrees, in order to more easily study the general structure of $\mathcal{D}$, we introduce a new operator, *jump*, that increments Turing degrees. Much of the behavior of this operator was shown in a join paper by Kleene and Post [8].

Definition 5.8. The jump operator

Problem 5.9 (Post's Problem)**.** *[12] [2]*

Lemma 5.10. *D does not form a lattice [8]*

Theorem 5.11. *D forms an upper semi-lattice [8]*

Theorem 5.12. *Existence of minimal degrees [21] [16]*

Theorem 5.13. *The c.e. degrees are dense [14]*

Homogeneity problems
Theory of D Undecidable
Theory of D equivalent to to second-order arithmetic [17]

5.2. Turing Degrees of Problems related to CFGs. [13]

Acknowledgments

It is a pleasure to thank my mentor, Ronno Das, for supervising this project and providing valuable feedback and advice. I would also like to thank Peter May for organizing this REU, and Daniil Rudenko for running the Apprentice Program.

References

[1] Klaus Ambos-Spies and Peter A. Fejer. *Degrees of Unsolvability.* 2006. URL: https://www.cs.umb.edu/~fejer/articles/History_of_Degrees.pdf.

[2] Richard M. Friedberg. "Two recursively enumerable sets of incomparable degrees of unsolvability (solution of Post's problem, 1944)". In: *Proceedings of the National Academy of Sciences* 43.2 (1957), pp. 236–238. ISSN: 0027-8424. DOI: 10.1073/pnas.43.2.236. eprint: http://www.pnas.org/content/43/2/236.full.pdf. URL: http://www.pnas.org/content/43/2/236.

[3] Seymour Ginsburg and Sheila A. Greibach. "Deterministic context free languages". In: *6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965).* Oct. 1965, pp. 203–220. DOI: 10.1109/FOCS.1965.7.

[4] Sheila A. Greibach. "The unsolvability of the recognition of linear context-free languages". In: *J. ACM* 13.4 (Oct. 1966), pp. 582–587. ISSN: 0004-5411. DOI: 10.1145/321356.321365. URL: http://doi.acm.org/10.1145/321356.321365.

[5] J. E. Hopcroft. "On the equivalence and containment problems for context-free languages". In: *Mathematical systems theory* 3.2 (June 1969), pp. 119–124. ISSN: 1433-0490. DOI: 10.1007/BF01746517. URL: https://doi.org/10.1007/BF01746517.

[6] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* 3rd ed. Pearson/Addison Wesley, 2007. ISBN: 0321455363.

[7] S. C. Kleene. "Recursive predicates and quantifiers". In: *Transactions of the American Mathematical Society* 53.1 (1943), pp. 41–41. DOI: 10.1090/s0002-9947-1943-0007371-8. URL: https://doi.org/10.1090/s0002-9947-1943-0007371-8.

[8] S. C. Kleene and Emil L. Post. "The Upper Semi-Lattice of Degrees of Recursive Unsolvability". In: *The Annals of Mathematics* 59.3 (1954), p. 379. DOI: 10.2307/1969708. URL: https://doi.org/10.2307/1969708.

[9] Stephen Cole Kleene. *Introduction to metamathematics.* 1952.

[10] Dexter C. Kozen. *Automata and Computability.* Springer, 1999. ISBN: 9780387949079. URL: http://www.worldcat.org/oclc/247385967?referer=xid.

[11] Manuel Lerman. *Degrees of Unsolvability.* Cambridge University Press, 2016. ISBN: 9781107168138.

[12] Emil L. Post. "Recursively enumerable sets of positive integers and their decision problems". In: *Bulletin of the American Mathematical Society* 50 (1944), pp. 284–316. ISSN: 1088-9485. DOI: https://doi.org/10.1090/S0002-9904-1944-08111-1.

[13] Ann Reedy and Walter J. Savitch. "The Turing degree of the inherent ambiguity problem for context-free languages". In: *Theoretical Computer Science* 1.1 (1975), pp. 77–91. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(75)90013-4. URL: http://www.sciencedirect.com/science/article/pii/0304397575900134.

[14] Gerald E. Sacks. "The recursively enumerable degrees are dense". In: *Annals of Mathematics* 80.2 (1964), pp. 300–312. ISSN: 0003486X. URL: http://www.jstor.org/stable/1970393.

[15] Géraud Sénizergues. "The equivalence problem for deterministic pushdown automata is decidable". In: *Automata, Languages and Programming.* Ed.

by Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 671–681. ISBN: 978-3-540-69194-5.

[16] J. R. Shoenfield. "A Theorem on Minimal Degrees". In: *The Journal of Symbolic Logic* 31.04 (1966), pp. 539–544. DOI: 10.2307/2269688. URL: https://doi.org/10.2307/2269688.

[17] Stephen G. Simpson. "First-order theory of the degrees of recursive unsolvability". In: *Annals of Mathematics* 105.1 (1977), pp. 121–139. ISSN: 0003486X. URL: http://www.jstor.org/stable/1971028.

[18] Michael Sipser. *Introduction to the Theory of Computation*. 3rd ed. Cengage Learning, 2013. ISBN: 1133187811.

[19] Robert I Soare. "The history and concept of computability". In: *Handbook of computability theory* 140 (1999), pp. 3–36.

[20] Robert I. Soare. "History of Computability". In: *Turing Computability*. Theory and Applications of Computability. Springer Berlin Heidelberg, 2016. Chap. 17, pp. 227–249. DOI: 10.1007/978-3-642-31933-4. URL: https://doi.org/10.1007/978-3-642-31933-4.

[21] Clifford Spector. "On Degrees of Recursive Unsolvability". In: *The Annals of Mathematics* 64.3 (1956), p. 581. DOI: 10.2307/1969604. URL: https://doi.org/10.2307/1969604.

[22] A. M. Turing. "On Computable Numbers, With an Application To the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* s2-42.1 (1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230. URL: https://doi.org/10.1112/plms/s2-42.1.230.