CS 4510 Automata and Complexity 3/07/2024

Homework 5: Undecidability

Vidit D. Pokharna *Due: 3/15/2024*

You should submit a typeset or *neatly* written pdf on Gradescope. The grading TA should not have to struggle to read what you've written; if your handwriting is hard to decipher, you will be asked to typeset your future assignments. Five bonus points if you use LaTeX, and our template. You may collaborate with other students, but any written work should be your own.

1. (6 points) Prove that $\{\langle M\rangle \mid 1001 \in L(M)\}$ is undecidable using the method of reduction.

   Assume to the contrary that the language $\{\langle M\rangle \mid 1001 \in L(M)\}$, which we can call L_{1001}, is decidable. This means there exists a Turing machine D that decides L_{1001}.

   We can construct a new Turing machine M' for any given Turing machine M and string w. Machine M' works as follows: for any input x, it ignores x and simulates M on w. If M accepts w, M' writes 1001 on its tape and then halts. If M does not accept w, M' loops indefinitely or rejects.

   Since we assumed L_{1001} is decidable, we could use D to decide whether 1001 is in the language recognized by M'. In other words, we can run D on $\langle M'\rangle$. If D says yes, that means M accepts w. If D says no, that means M does not accept w.

   This construction leads to a contradiction because it implies that we can decide A_{TM} by using D. We could construct M' from any M and w, and use D to decide if M accepts w. But since A_{TM} is undecidable, there cannot exist such a machine D.

   Therefore, the initial assumption that L_{1001} is decidable must be false. Hence, the language $\{\langle M\rangle \mid 1001 \in L(M)\}$ is undecidable by reduction.

2. (6 points) Prove that $HALTALL = \{\langle M \rangle \mid M$ halts on ALL inputs $\}$ is undecidable using the method of reduction.

   Assume to the contrary that $HALTALL$ is decidable. This means there exists a Turing machine D that decides $HALTALL$.

   Consider an arbitrary Turing machine M and an arbitrary string w. We construct a new Turing machine M_w as follows:

   - For any input x, M_w first checks if $x = w$.
   - If $x = w$, then M_w simulates M on input w.
   - If $x \neq w$, M_w automatically halts.

   This construction is crucial because M_w is designed to halt on all inputs except possibly for w. Now, we can use the hypothetical decider D to decide whether $\langle M_w \rangle$ belongs to $HALTALL$.

   If D determines that M_w halts on all inputs, it implies that M_w also halts on input w, and therefore M halts on input w. Conversely, if D determines that M_w does not halt on all inputs, the only input that could potentially cause M_w not to halt is w. Therefore, in this case, M does not halt on w.

   By constructing M_w and using D, we have created a way to decide whether an arbitrary Turing machine M halts on an arbitrary input w, effectively solving $HALT$, which is known to be undecidable.

   This leads to a contradiction because we assumed $HALTALL$ is decidable, but our construction implies that its decidability would lead to the decidability of the Halting Problem. As the Halting Problem is undecidable, our initial assumption that $HALTALL$ is decidable must therefore be false.

   Hence, we conclude that $HALTALL$ is undecidable by reduction.

3. (8 points) Consider the function

$$f(x) = \begin{cases} 3x+1 & x \text{ is odd} \\ x/2 & x \text{ is even} \end{cases} \tag{1}$$

We use the notation $f^i(x)$ to represent repeated application of f to x an i number of times. For example, $f^3(x) = f(f(f(x)))$. The Collatz Conjecture states

$$\forall x \exists i [f^i(x) = 1]$$

with $x, i \in \mathbb{N}_{\geq 1}$. That is, repeated application of this function to any number will eventually reach 1. For example, if $x = 17$, we get the sequence 17,52,26,13,40,20,10,5,16,8,4,2,1 The conjecture remains a massive open problem in mathematics. It has been computationally verified for the first few million numbers, but has not been proven for all numbers. Suppose you were given a decider for $HALTALL$ called H. Describe a method in which you could prove that the conjecture is either true or false.

For every natural number x, construct a specific Turing machine M_x that embodies the computation of the repeated applications of the Collatz function f starting from x. This Turing machine would work as follows:

- Initialize with x on the tape
- Apply the Collatz function f repeatedly to the number on the tape
- If the number reaches 1, halt
- If the number enters a loop that does not include 1 (hypothetically, since no such loop has been found), continue indefinitely (though according to the conjecture, this should never happen)

Since M_x is designed to only work on a single input (it starts with x and follows the Collatz sequence from there), "halting on all inputs" in this context effectively means it halts on its single intended path. Use H to decide if M_x halts for its specific sequence.

- If H decides that M_x halts, this corresponds to the sequence eventually reaching 1. Since x was arbitrary, this gives evidence in favor of the Collatz Conjecture for this particular x.
- If H ever found a machine M_x that does not halt, this would mean there exists some starting number x for which the Collatz sequence does not reach 1, thus disproving the conjecture.

Ideally, we'd use H to systematically check every natural number x. In practice, this is infinite and not feasible, but theoretically, this approach could test the Collatz Conjecture for any given range of natural numbers.

If for every tested x, H determines that M_x halts, and thus the Collatz sequence reaches 1, this supports (but due to practical limitations, does not conclusively prove) the truth of the Collatz Conjecture. Conversely, finding just one x for which M_x does not halt would disprove the conjecture.

4. (8 points) Prove (by diagonalization) there exists a language which is intuitively computable, but not context-free.

   We must first note there are countably infinite context-free grammars. This is because a context-free grammar can be represented as a finite string over a finite alphabet (including nonterminal and terminal symbols, as well as production rules), and the set of all finite strings over a finite alphabet is countable.

   There are also countably infinite Turing machines, since a Turing machine can be encoded as a finite string over a finite alphabet. This set includes all possible decidable (intuitively computable) languages since each decidable language can be recognized by at least one Turing machine.

   Since there are countably infinite Turing machines, there are countably infinite decidable languages. However, not all of these decidable languages are context-free. To prove this, assume to the contrary that every decidable language is context-free. Since there are only countably infinite context-free grammars, this would imply a one-to-one correspondence between context-free grammars and decidable languages.

   Construct a diagonal language to show a contradiction. Let's assume we list all context-free grammars in some order as $G_1, G_2, G_3, \ldots$ and let their corresponding languages be $L(G_1), L(G_2), L(G_3), \ldots$.

   Define a new language L_{diag} such that a string w_i (the i-th string in some canonical ordering of all strings) is in L_{diag} if and only if $w_i \notin L(G_i)$. Essentially, L_{diag} includes w_i if and only if w_i is not in the language defined by the i-th context-free grammar in our list.

   This language L_{diag} is decidable by a Turing machine that, given a string w_i, finds the i-th grammar G_i, generates the language $L(G_i)$ up to the length of w_i (since context-free languages can be parsed in polynomial time), and decides whether w_i is in $L(G_i)$. If w_i is not in $L(G_i)$, the Turing machine accepts w_i; otherwise, it rejects.

   The language L_{diag} is, by construction, decidable since we've outlined a Turing machine that decides it. However, L_{diag} cannot be context-free. If it were, it would be represented by some context-free grammar G_j in our list. But by the construction of L_{diag}, we have that w_j is in L_{diag} if and only if w_j is not in $L(G_j)$, which is a contradiction.

   Therefore, there must exist a decidable language that is not context-free, disproving the initial assumption that all decidable languages are context-free by a contradiction diagonalization argument.

5. (8 points) Let C be any kind of computational model, mechanical process, or decision procedure which satisfies the fathomability criterion. Suppose further that it cannot define machines which loop infinitely. Every machine definable in C terminates on any input. You could perhaps consider C as a programming language which only defines bounded loops. Prove (by diagonalization) that C is not Turing-complete.

   Assume to the contrary that C is Turing-complete. This means, by definition, that C can simulate any computation that a Turing machine can perform. Given that every machine definable in C is required to terminate on any input, C inherently cannot define or simulate machines which loop infinitely.

   Consider a hypothetical Turing machine TM_d that operates as follows: for each natural number i, TM_d simulates the action of the i-th computational model C_i defined within C (since C satisfies the fathomability criterion, such an enumeration is possible). For its input i, if C_i halts, TM_d will continue to loop, and vice versa, if C_i loops, TM_d will halt.

   Since C is assumed to be Turing-complete, it should be able to define a computational model that behaves exactly like TM_d. However, this leads to a contradiction: there cannot exist such a model within C because TM_d is designed to perform the opposite action of every machine defined within C when given its own index as input.

   This contradiction demonstrates that no machine within C can have the same behavior as TM_d, because TM_d effectively contradicts the operational outcome of each possible machine in C when run on its index. The contradiction arose under the assumption that C is Turing-complete. Hence, we must conclude that our original assumption is false. Therefore, C, a computational model that cannot define machines which loop infinitely and that satisfies the fathomability criterion, is not Turing-complete.

6. (4 points) Read the Library of Babel by Jorge Luis Borges. Let us suppose that ZFC is a set of axioms to model our natural law, and let LB be a set of axioms to model the world of the Library of Babel. Let T be any statement. Prove or disprove:

$$ZFC \vdash T \iff LB \vdash T$$

In ZFC, theorems and propositions follow from a coherent and consistent set of axioms. A statement T being derivable within ZFC ($ZFC \vdash T$) signifies that T holds true under the logical structure and axioms of ZFC. This implies a strict adherence to mathematical logic and deductive reasoning.

The concept of $LB \vdash T$ is paradoxical because LB, as described by Borges, does not follow a logical structure; it contains all possible texts. Therefore, while it is true that within the library there exists a sequence of symbols that could represent the proof of T, there also exists a sequence representing the negation of T, as well as every other conceivable statement.

The statement $ZFC \vdash T \iff LB \vdash T$ combines two fundamentally distinct concepts: logical proof in a formal system (ZFC) and mere textual existence in an infinite library (LB). In ZFC, T being provable means it logically follows from foundational mathematical principles. In LB, the provability of T merely means that somewhere within an infinite area, there exists a text that coincidentally matches the form of a proof of T, but without regard to logical consistency or relevance.

Therefore, the equivalence $ZFC \vdash T \iff LB \vdash T$ does not hold. In ZFC, T is proven through a sequence of logical deductions from the axioms. In LB, while there might exist a string of characters that resembles a proof of T, this occurrence is random and lacks the methodical, reasoned derivation required in formal mathematics. Thus, ZFC and LB operate under completely different paradigms, leading to the equivalence being invalid.