Witnessability of Undecidable Problems

Jaehui Hwang, Jungin Rhee, Gyeongwon Jeong

November 2023

1 Introduction

This paper defines a subset of undecidable problems called "witnessable" problems and demonstrates that many problems in the field of programming languages fall within this category. Informally, a problem is considered witnessable if there exists a computable function that takes a decidable approximation as an input and returns a witness input demonstrating its imprecision.

The report consists of two parts: a summary of the paper (Sections 3, 4) and additional studies and analyses we conducted (Sections 5, 6, 7). In Sections 5 through 7, we studied the relationship between witnessable problems and the Rice's theorem, constructed witness functions for the diagonal halting problem and the problems from Rice's theorem, and tried to improve the iterative witness algorithm presented in the paper using our devised witness function constructions.

2 How We Studied

We held weekly meetings to discuss each part of the paper and to conduct additional analysis. We studied the proofs of the three main theorems in the paper together, and divided the remaining sections into three parts to study individually and shared what each of us studied.

- Gyeongwon studied the algorithm that iteratively finds imprecision witnesses.
- Jaehui studied the existence of non-witnessability and its set-theoretic cardinality.
- Jungin studied the case study section that applies the theory of witnessability to problems in programming language theory.

After understanding the theory and the case study paper presents, we tried to (1) fill in some missing holes in the paper that it only mentions briefly (Section 5, 6) and (2) improve the algorithm paper presents (Section 7).

3 Preliminaries

We call $P \triangle Q = (P - Q) \cup (Q - P)$ as the symmetric difference of P and Q.

Definition 1 (Partial Computable Functions)**.** A function $f : \mathbb{N}^k \to \mathbb{N}$ is called *k-ary partial computable* if partial functions are computable by Turing machines, lambda calculus, or any equivalent models of computation.

Definition 2 (Definedness)**.** The domain of a partial computable function ϕ is the set of inputs on which ϕ is defined : $\{x : \phi(x) \uparrow\}$. If ϕ is undefined(or divergent) on the input x, we write as $\phi(x) \downarrow$. If ϕ is defined over all natural numbers, then we say ϕ is *total computable*.

Definition 3 (Numberings)**.** We define partial computable function ϕ as a natural number, $i \in \mathbb{N}$. In this case, we use the notation ϕ_i instead of ϕ, and the index in the numbering is analogous to a "program(code)" implementing the corresponding partial computable function. Note that there exists infinitely many natural numbers corresponding to ϕ.

Definition 4 (Computational Problems)**.** A (computational) problem is formulated as a subset of $\mathbb{N}$. For example, the diagonal halting problem $K = \{i : \phi_i(i) \downarrow\}$ is the set of natural numbers representing programs that halt on themselves.

Definition 5 (Decidable Approximation)**.** Given an undecidable problem P, any decidable problem $Q \subseteq \mathbb{N}$ is called a *decidable approximation* of P. In addition, if $Q \subsetneq P$, then Q is called a decidable under-approximation of P. Also, any element in $P \triangle Q$ is called an *imprecision witness* for Q.

Definition 6 (Many-to-one Reduction)**.** Given two problems $A, B \subseteq \mathbb{N}$ one says A is *many-one reducible* to B and writes

$$A \leq_m B,$$

if there exists a total computable function g with $x \in A$ if and only if $g(x) \in B$.

4 Witnessable Problems

Definition 7 (Witnessability)**.** We say an undecidable problem $P \in \mathcal{P}(\mathbb{N})$ is *witnessable* if and only if there exists a partial computable function w_P, such that for any decidable approximation $Q \subset \mathbb{N}$ and any natural number q such that $Q = \{x : \phi_q(x) = 1\}$, $w_P(q)$ is defined and $w_P(q) \in P \triangle Q$. The partial computable function w_P is called a *witness function* of P.

The paper introduces following three main theorems about witnessability:

Theorem 1. *The diagonal halting problem $K = \{i : \phi_i(i) \downarrow\}$ is witnessable.*

Theorem 2. *If P is witnessable, then its complement P^C is also witnessable.*

Theorem 3. *If P_1 is witnessable and P_1 is many-one reducible to P_2, P_2 is also shown to be witnessable.*

The paper also suggest the following theorem which states that we can compute n imprecision witnesses for each approximation of each witnessable problem for arbitrarily large n:

Theorem 4 (Iterative Witness)**.** *If a problem P is witnessable and Q is a decidable approximation of P, then there is a 2-ary total computable function t such that for any $q \in \mathbb{N}$ such that $Q = \{x : \phi_q(x) = 1\}$, $\{t(q,0), t(q,1), \cdots\}$ is an infinite list of different imprecision witnesses for Q.*

From the proof of this theorem, we can derive an algorithm to find n distinct imprecision witnesses for Q as follows.

Algorithm 1

Input: a program $q_0 \in \mathbb{N}$ such that $Q = \{x : \phi_{q_0}(x) = 1\}$
for $i = 1$ *to* n **do**
  $t_i := w_p(q_{i-1})$
  $q_i := \lambda x.(\textbf{if } x = t_i \textbf{ then } 1 - \phi_{q_i}(x) \textbf{ else } \phi_{q_i}(x))$
return $t_1, \cdots, t_n$

Also, the paper claims that there exists uncountably many witnessable undecidable problems, and so as non-witnessable undecidable problems.

Theorem 5. *There are $2^{\aleph_0}$ witnessable undecidable problems, and non-witnessable undecidable problems.*

5 Witnessability and Rice's Theorem

The paper mentions (without concrete proof) that common undecidable problems encountered in programming language theory are witnessable. These common undecidable problems refer to all non-trivial semantic properties of programs mentioned in Rice's theorem. We studied Rice's theorem to understand this sentence and completed the proof omitted in the paper.

Definition 8. Let $\mathbf{P}^{(1)}$ denote the collection of all unary partial computable functions, i.e. $\mathbf{P}^{(1)} := \{\phi_i : i \in \mathbb{N}\}$. A *property* is a subset of $\mathbf{P}^{(1)}$. A property F is said to be *trivial* if $F = \emptyset$ or $F = \mathbf{P}^{(1)}$.

Theorem 6. *For every property $F \subseteq \mathbf{P}^{(1)}$, define a decision problem $D_F := \{i \in \mathbb{N} : \phi_i \in F\}$. Then D_F is decidable if and only if F is trivial.*

Proof. This theorem is an immediate consequence of the following **Lemma 1**. □

Lemma 1. For any non-trivial property $F \subset \mathbf{P}^{(1)}$, $K \leq_m D_F$.

Proof. WLOG, assume that *no-halt* $\notin F$ where *no-halt* being the partial computable function that is undefined everywhere. Since F is non-trivial, there exists $a \in \mathbb{N}$ such that $\phi_a \in F$. Now for any $i \in \mathbb{N}$, define $g(i)$ as a natural number representing the following algorithm $T(x)$:

$$T \text{ first simulates } \phi_i(i);\ \text{then } T \text{ simulates } \phi_a(x) \text{ and returns its result.}$$

If $i \in K$, since $\phi_i(i)$ terminates, $\phi_{g(i)} = \phi_a \in F$, thus $g(i) \in D_F$. If $i \notin K$, then since $\phi_i(i)$ not terminates, $\phi_{g(i)} = \textit{no-halt} \notin F$, thus $g(i) \notin D_F$. Therefore, g is a many-one reduction from K to D_F. □

Using this lemma, we can also prove the witnessability of D_F.

Theorem 7. *For any non-trivial property $F \subset \mathbf{P}^{(1)}$, D_F is witnessable.*

Proof. Because $K \leq_m D_F$, by the **Theorem 3**, D_F is witnessable. □

6 Constructing Witness Functions

One drawback of the definition of witnessable is that even if a problem is proved to be witnessable, it doesn't provide information about what the witness function for that problem might be. Thus, we explored whether we could explicitly construct witness functions for several witnessable problems. From now on, we will regard a program $p \in \mathbb{N}$ as a string (which is equivalent to a base-n integer, where n is the number of all distinct characters) using the syntax of the language learned in class, and assume ϕ_p is a partial computable function representing the semantic of p.

6.1 Witness Function of the Diagonal Halting Problem

We first construct the witness function of the diagonal halting problem K, by following the proof of **Theorem 1** which demonstrates that K is witnessable. The witness function w_K we constructed is as follows:

$$w_K(q) := \lambda x.\,(\textbf{if } \phi_q(x) = 0 \textbf{ then } 0 \textbf{ else } (\textbf{while true skip}))$$

for some decidable approximation $Q := \{x : \phi_q(x) = 1\}$. The following theorem shows that our w_K is indeed a witness function of K.

Theorem 8. *For any $q \in \mathbb{N}$ such that $Q = \{x : \phi_q(x) = 1\}$ is decidable, $w_K(q) \in K \triangle Q$.*

Proof. Suppose $w_K(q) \in K$. Then $\phi_{w_K(q)}(w_K(q)) \downarrow$, so the program

$$\textbf{if } \phi_q(w_K(q)) = 0 \textbf{ then } 0 \textbf{ else } (\textbf{while true skip})$$

terminates. To do so we should have $\phi_q(w_K(q)) = 0$, which implies $w_K(q) \notin Q$.
Suppose, on the other hand, $w_K(q) \notin K$. Then $\phi_{w_K(q)}(w_K(q)) \uparrow$, so the program above runs forever. To do so we should have $\phi_q(w_K(q)) = 1$, which implies $w_K(q) \in Q$. Therefore, $w_K(q) \in K \triangle Q$ in both cases. □

6.2 Witness Functions from Rice's Theorem

Furthermore, we construct the witness function of D_F for any non-trivial property F as mentioned in Rice's theorem. Assuming *no-halt* $\notin F$ and $\phi_a \in F$ for some $a \in \mathbb{N}$, the witness function w_F we constructed is as follows:

$$w_F(q) := \textbf{let } g \equiv \lambda x.\,(\lambda i.\,(\phi_i(i);\, \phi_b(x))) \textbf{ in } g(\lambda x.\,(\textbf{if } \phi_q(g(x)) = 0 \textbf{ then } 0 \textbf{ else } (\textbf{while true skip})))$$

for some decidable approximation $Q := \{x : \phi_q(x) = 1\}$. For readability, we represent the function application syntax $e_1\, e_2$ as $e_1(e_2)$.

Theorem 9. *For any $q \in \mathbb{N}$ such that $Q = \{x : \phi_q(x) = 1\}$ is decidable, $w_F(q) \in D_F \triangle Q$.*

Proof. Suppose $w_F(q) \in D_F$. Notice that g is a function that represents a many-one reduction from K to D_F(refer to the proof of **Lemma 1**). Thus by the definition of many-one reduction, the program

$$\textbf{let } g \equiv \lambda x.\,(\lambda i.\,(\phi_i(i);\, \phi_b(x))) \textbf{ in } \lambda x.\,(\textbf{if } \phi_q(g(x)) = 0 \textbf{ then } 0 \textbf{ else } (\textbf{while true skip}))$$

is in K, and this implies

$$\phi_q\big(\textbf{let } g \equiv \lambda x.\,(\lambda i.\,(\phi_i(i);\, \phi_b(x))) \textbf{ in } g(\lambda x.\,(\textbf{if } \phi_q(g(x)) = 0 \textbf{ then } 0 \textbf{ else } (\textbf{while true skip})))\big) = 0.$$

This exactly means $\phi_q(w_F(q)) = 0$, which implies $w_F(q) \notin Q$. Similarly, we can prove $w_F(q) \notin D_F$ implies $w_F(q) \in Q$. □

7 Parallelized Iterative Witness Algorithm

We explored methods to improve **Algorithm 1** in Section 4 and devised a parallelized version of this algorithm. Let $P \subseteq \mathbb{N}$ be a witnessable problem that we discussed in Section 6 and Q be its decidable approximation. We suggest following parallel algorithms to compute n impricision witnesses of Q with an input $q_0 \in \mathbb{N}$ such that $Q = \{x : \phi_{q_0}(x) = 1\}$:

Algorithm 2

Input: q_0
for $i = 1$ *to* $\lfloor n/2 \rfloor$ **do**
  $t_i := w_P(q_{i-1})$
  $q_i := \lambda x.\,(\textbf{if } x = t_i \textbf{ then } 1 - \phi_{q_{i-1}}(x) \textbf{ else } \phi_{q_{i-1}}(x))$
return $t_1, \cdots, t_{\lfloor n/2 \rfloor}$

Algorithm 3

Input: q_0
for $i = 1$ *to* $\lfloor n/2 \rfloor$ **do**
  $q_i := \lambda x.\,(\textbf{if } x = w_P(q_{i-1}) \textbf{ then } 1 - \phi_{q_{i-1}}(x) \textbf{ else } \phi_{q_{i-1}}(x))$
for $i = \lfloor n/2 \rfloor + 1$ *to* n **do**
  $t_i := w_P(q_{i-1})$
  $q_i := \lambda x.\,(\textbf{if } x = t_i \textbf{ then } 1 - \phi_{q_{i-1}}(x) \textbf{ else } \phi_{q_{i-1}}(x))$
return $t_{\lfloor n/2 \rfloor + 1}, \cdots, t_n$

At first glance, **Algorithm 3** seems to be doing more work compared to **Algorithm 2**, but both algorithms have the same time complexity in terms of the number of calls to the witness function w_P. This parallel algorithm assumed that we have only two computation cores, but this idea can be extended for k cores in such a way that the number of times the witness function needs to be called on each core becomes n/k. This can be advantageous when n is relatively large compared to the complexity of w_P, especially when there is a substantial number of cores available for computation. The inspiration behind this parallelization lies in the ability to define the code for program $q_{\lfloor n/2 \rfloor}$ without knowing $t_1, \cdots, t_{\lfloor n/2 \rfloor}$. This is possible because we defined the witness function explicitly in Section 6, because we need explicit string representation of w_P while calculating q_i in the first loop of **Algorithm 3**.

8 References

Shuo Ding and Qirun Zhang. Witnessability of undecidable problems. *Proc. ACM Program. Lang.*, 7(POPL), jan 2023.