On effective undecidability and Post's problem

Bas Westerbaan
bas@westerbaan.name

July 11, 2012

1 Introduction

A subset $A \subseteq \mathbb{N}$ of the natural numbers is called **decidable**, if its characteristic function $\mathbb{1}_A$, defined by

$$\mathbb{1}_A(n) = \begin{cases} 1 & n \in A \\ 0 & n \notin A, \end{cases}$$

is computable. **Computable**, **recursive** and **effective** are synonyms.

A subset $A \subseteq \mathbb{N}$ is called **recursively enumerable** if it is finite or there exists a computable $a\colon \mathbb{N} \to \mathbb{N}$ that enumerates it. That is:

$$A = \{a(0),\, a(1),\, \ldots\}.$$

Similarly **computably enumerable** and **effectively enumerable** are synonyms. Every decidable set is recursively enumerable, but there are a lot of recursively enumerable sets that are not decidable.

An important example of such a set is $\mathcal{K}$.[1]. Let $\varphi^1_0,\, \varphi^1_1,\, \ldots$ be a standard enumeration of all computable partial functions from $\mathbb{N}$ to $\mathbb{N}$ and let $\varphi^1_{37}(n){\downarrow}$ denote that φ^1_{37} is defined on n. Then we define

$$\mathcal{K} = \{e;\ e \in \mathbb{N};\ \varphi^1_e(e){\downarrow}\}.$$

We can find a recursive $r\colon \mathbb{N} \to \mathbb{N}$ such that

$$\varphi^1_{r(e)}(x){\downarrow} \iff \varphi^1_e(x) = 0.$$

Then

$$r(e) \in \mathcal{K} \iff \varphi^1_{r(e)}(r(e)){\downarrow} \iff \varphi^1_e(r(e)) = 0$$

Hence $\varphi^1_e(r(e)) \neq \mathbb{1}_{\mathcal{K}}(r(e))$, for any e. Thus no φ^1_e can compute $\mathcal{K}$.

Two famously undecidable sets are

$$\begin{aligned} \textsc{pa} &= \{\#\psi;\ \psi \in L_{\mathrm{N}};\ \mathrm{PA} \vdash \psi\} \\ \textsc{n} &= \{\#\psi;\ \psi \in L_{\mathrm{N}};\ \langle \mathbb{N}, +, \cdot, 0, 1, \leq \rangle \vDash \psi\}, \end{aligned}$$

where L_{N} is the set of first-order formulas about the natural numbers and $\#\psi$ is the Gödelcode of ψ. Thus PA is the set of natural numbers coding a sentence that is provable from the axioms of Peano and N is the (bigger) set of natural numbers coding a sentence that is true about the natural numbers.

The undecidability of N is a celebrated result by Alan Turing[Tur36][2] and a definitive blow to Gottfried Leibniz' and David Hilbert's dream that (mathematical) truth might be established mechanically.

The set $\mathcal{K}$ is as undecidable as PA in the following sense. If there were an oracle which would answer every query of the form "is $n \in \mathcal{K}$?", then we can give an algorithm using this oracle to compute PA. And vice versa.

The set N, however, is "more undecidable" than PA. That is: even if we allow an algorithm to use an oracle for PA, it still cannot compute N. It is not even recursively enumerable.

This leads to the study of degrees of undecidability. In 1944 Emil Post asked the following question:

Problem 1 (Post [Pos44])**.** Is there a recursively enumerable set that is undecidable but strictly "less undecidable" than $\mathcal{K}$?

[1] $\mathcal{K}$ appears (implicitely) in the work of Gödel [Göd31], Turing [Tur36] and Kleene [Kle36]

[2] It was also discovered independently by Alonzo Church.[Chu36]

This problem was solved 14 years later by Albert Muchnik[Muc58] (and independently a year later by Richard Friedberg[Fri57]) using a method, now called the *priority method*, which has become a main tool in recursion theory.

Figure 1: Emil Post

The set they constructed is, in a sense, artificial. One would hope for a set that emerges more naturally, such as PA.

One uncanny aspect of the set constructed by Friedberg and Muchnik, say X, is that we only have an indirect method to prove that X differs from any given decidable set D. The method suggests a number n_1 that might be an example of a number that is in D, but is not in X. When n_1 nevertheless becomes a member of X, the method yields another number n_2, which again might be an example of a number that is in $D - X$. When n_2 appears in X, the method yields a n_3 which might be in $D - X$. Et cetera. This, however, does not go on indefinitely. There is a computable $f\colon \mathbb{N} \to \mathbb{N}$ such that for the eth decidable set, the method requires less than $f(e)$ tries.

In Post's search for a solution and also in later work of Martin and others it appears that every set that is in some reasonable sense 'effectively undecidable' ends up being as undecidable as $\mathcal{K}$.

In this thesis, we will look at some known and some previously uninvestigated notions of effective undecidability. We try to discover how far we can stretch effective undecidability in the hope to get a more tractable solution to Post's problem, than that of Friedberg and Muchnik.

Contents

2 Basic recursion theory

In this section, we will briefly review the basics of recursion theory. The reader familiar with the matter, is suggested to skip to Subsection 2.7.

2.1 Models of computation and the Church-Turing Thesis

In the introduction, we informally talked about computable functions. Can we define these formally? There have been dozens of wildly varying suggestions for formalization of the notion of computation. All suggestions so far are proven equivalent.[3] The Church-Turing Thesis asserts that the notion of computation is captured by any of those.

We will briefly review three formalizations of computation: μ-recursive, λ-definable and Turing-computable.

2.1.1 μ-recursive functions

Let $\mathcal{F}_k$ be the set of partial functions from $\mathbb{N}^k$ to $\mathbb{N}$ and $\mathcal{F} = \bigcup_{k\in\mathbb{N}} \mathcal{F}_{k+1}$. The set of μ-recursive functions is a subset of $\mathcal{F}$ defined inductively as follows:[Vel87]

1. Zero function. $0 \mapsto 0$ is μ-recursive.
2. Projections. For any k and $i \in \{1, \dots, k\}$ the function $(n_1, \dots, n_k) \mapsto n_i$ is μ-recursive.
3. Successor. The function $n \mapsto n+1$ is μ-recursive.
4. Composition. For any k, l and μ-recursive k-ary h and μ-recursive l-ary $g_1, \dots, g_k$ the function
$$(n_1, \dots, n_l) \mapsto h(g_1(n_1, \dots, n_l), \dots, g_k(n_1, \dots, n_l))$$
is also μ-recursive.
5. Primitive recursion. Given any μ-recursive $(k+2)$-ary function g, the $(k+1)$-ary function f such that
$$\begin{aligned} f(0, x_1, \dots, x_k) &= g(0, 0, x_1, \dots, x_k) \\ f(n+1, x_1, \dots, x_k) &= g(n+1, f(n, x_1, \dots, x_k), x_1, \dots, x_k) \end{aligned}$$
is also μ-recursive.

There is one rule left: minimization. The partial functions defined by only the first 5 rules are in fact total and called the primitive recursive functions. In the early 1900s it was believed that all total computable functions can be defined by primitive recursion.[Döt91] In 1923, Wilhelm Ackermann published a counterexample[4]. To capture all computable functions, Kleene suggested the final rule for μ-recursive functions:

[3]See Theorem I7.12 of [Odi87]

[4]This is Ackermann's function. It is total and computable, but not primitive recursive.
$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

6. Minimisation. Given any μ-recursive $(k+1)$-ary function g, the k-ary function f such that

$$\begin{aligned} f(x_1,\ldots,x_k) = z \iff & f(z,x_1,\ldots,x_k) = 0 \text{ and} \\ & \forall n < z \exists y \neq 0 [f(n,x_1,\ldots,x_k) = y] \end{aligned}$$

is also μ-recursive. In general f is partial.

Although the definition of μ-recursive functions is elegant, it is not intuitively clear that they are exactly the computable functions: we are a priori not sure that there is no counterexample like that Ackermann for the primitive recursive functions. However, rest assured: the other two equivalent formalizations that follow are rather more convincing in capturing the notion of computability.

2.1.2 Lambda Calculus

Another formalization is the Lambda Calculus.[BG00] Alonzo Church proved in 1936 that there is no λ-definable function that decides whether two given λ-terms are β-equal.[Chu36] These two notions will be defined later on. If λ-definable is the same as computable, this proves that there are classes of mathematical statements that cannot be solved algorithmically.

Given a function f and an argument x, creating $f(x)$ is called *(function) application*. Given some expression with a free variable (for instance $y^2 - \int_0^1 x^2 \,\mathrm{d}x$) creating a function in that free variable (in our example: $f(y) = y^2 - \int_0^1 x^2 \,\mathrm{d}x$) is called *(function) abstraction*.

Usually, when one considers some kind of functions, their arguments are of a different type than the functions themselves. Linear functions act on vectors, functors act on structures, et cetera.

The λ-calculus is a language to describe functions that act on themselves and (because that on itself would not be very interesting) allow (function) abstractions.

The *words* in the calculus are the *lambda terms*.

- x is a variable symbol.
- If v is a variable symbol, then v' is a variable symbol.
- If v is a variable symbol, then v is a lambda term.
- If M and N are lambda terms, then the word (MN) is a lambda term. This corresponds to function application: M applied to N.
- If v is a variable symbol and M is a lambda term, then the word $(\lambda v.\, M)$ is a lambda term. This corresponds to function abstraction: $v \mapsto M$.

Thus $(\lambda x.\,(xx))$ corresponds to the function that applies its argument to itself. By convention, we will write $x, y, z, \ldots$ for variables; leave out redundant parenthesis; let application bind stronger than abstraction; associate application to the left and abstraction to the right *and* write $\lambda a_1 \cdots a_n.\, M$ as shorthand for $\lambda a_1. \cdots \lambda\, a_n.M$. If two terms are equal up to renaming of variables that respects binding of λ, we consider them equal. Example:

$$\begin{aligned}\lambda xy.\, y(xx) &= (\lambda x.\, (\lambda x'.\, (x'(xx))))\\ &= \lambda yx.\, x(yy)\\ &\neq \lambda yx.\, y(xx).\end{aligned}$$

To capture the intention to model function, we study the following reduction:

$$(\lambda v.\, M)N \to_\beta M[v := N].$$

Here $M[v := N]$ is understood to mean "M with all unbound occurances of v are replaced by N" if M and N do not share variables. If M and N do share variables, we can always find equivalent terms by renaming variables such that this is not the case. Also if for some lambda term M there is a subterm N such that $N \to_\beta N'$ then $M \to_\beta M'$, where M' is M with that subterm N replaced by N'. We write $M \to_\beta^* N$ for the reflexive transitive closure of $\to_\beta$ and $=_\beta$ for the symmetric transitive closure. Example:

$$\begin{aligned}&(\lambda x.\, xx)((\lambda xyz.\, x(yz))x(\lambda x.\, xx))\\ &\to_\beta (\lambda xyz.\, x(yz))x(\lambda x.\, xx)((\lambda xyz.\, x(yz))x(\lambda x.\, xx))\\ &\to_\beta x((\lambda x.\, xx)((\lambda xyz.\, x(yz))x(\lambda x.\, xx)))\\ &\to_\beta x((\lambda xyz.\, x(yz))x(\lambda x.\, xx)((\lambda xyz.\, x(yz))x(\lambda x.\, xx)))\\ &\leftarrow_\beta x((\lambda x.\, xx)((\lambda xyz.\, x(yz))x(\lambda x.\, xx))).\end{aligned}$$

We can elegantly represent the natural numbers and their operation in the lambda calculus using so-called Church-numerals: $\underline{0} = \lambda so.\, o$ and $\underline{n+1} = \lambda so.\, \underline{n}s(so)$. Thus $\underline{3} =_\beta \lambda so.\, s(s(so))$. Let $\oplus = \lambda nmso.\, ns(mso)$ and $\otimes = \lambda nmso.\, n(\lambda c.\, msc)o$. It is not hard to verify that $\underline{n+m} =_\beta \oplus\underline{nm}$ and $\underline{n \cdot m} =_\beta \otimes\underline{nm}$ for all $n, m \in \mathbb{N}$.

A partial function $f\colon \mathbb{N} \to \mathbb{N}$ is called λ-definable[Chu33][Kle35] if there is a lambda term M_f such that for all n and m:

$$f(n) = m \iff M_f\underline{n} =_\beta \underline{m}.$$

2.1.3 Turing Machines

The λ-calculus has found many applications especially in theoretical computer-science. Also there are many programming languages modelled on the calculus. These are called *functional languages*. However, at the time (and still on the present day), many consider λ-calculus to be a weird exercise.

A year after Chuch, Alan Turing independently proved that mathematical truth cannot be established algorithmically. To do this, he introduced Turing Machines. When the famous logician Kurt Gödel heard of Turing's work, he promptly stated that Turing finally convincingly defined computability with his machines. Computer processors and imperative programming languages are based on the Turing Machine and are vastly more popular than their functional counterparts.

A Turing Machine is an idealized computer. It consists of a tape and a read/write-head on that tape. The tape extends infinitely in both directions and consists out of blocks. Each block can hold the value 0 or 1. Thus, the state of the tape can be modeled by a map $\mathbb{Z} \to \{0,1\}$. The read/write-head holds a program. A program consists of a finite number of states. For each state s it is specified which action to take when the head is on a block with a 0 and when the head is on a block with a 1. An action consists of either:

1. writing a 0 or a 1; then moving the head one block to the left or to the right *and* then changing the state of the machine *or*
2. stopping the machine.

To *run the machine on input* n for $n \in \mathbb{N}$ we start with a tape consisting of n times a 1 right of the starting position of the head and for the remainders only 0s. Then we execute the program, step by step. The machine might halt. This is the case if after a finite number of steps, the program instructs to stop the machine. If the tape consists of m times a 1 right of the ending position of the head and for the rest 0s, we call m the *output*.

A partial function $f\colon \mathbb{N} \to \mathbb{N}$ is Turing-computable if there is a program such that for every n where f is defined, the Turing Machine loaded with that program, halts on input n with output $f(n)$ and for every n where f is undefined, the Turing Machine loaded with that program does not halt.

2.2 Computable partial functions and sets

Pick a preferred model of computation. Programs (for Turing Machines), λ-terms (in de λ-Calculus) and μ-recursive functions can all be enumerated effectively. Let φ_e denote the eth computable partial function from $\mathbb{N}^*$ to $\mathbb{N}$, where $\mathbb{N}^*$ are the finite lists of natural numbers. We define $\varphi_e^k = \varphi_e \restriction \mathbb{N}^k \to \mathbb{N}$.

It is in general undecidable whether $\varphi_e(x)\downarrow$. However, if we limit ourselves to s steps of computation and define:

$$\varphi_{e,s}(x) = \begin{cases} \varphi_e(x) & \varphi_e(x) \text{ requires at most } s \text{ steps of computation} \\ \uparrow & \varphi_e(x) \text{ did not stop after } s \text{ steps of computation.} \end{cases}$$

Then $\varphi_{e,s}(x)\downarrow$ is decidable.

A set $A \subseteq \mathbb{N}$ is called computable if its characteristic function is computable. That is: there is an e such that $\mathbb{1}_A(x) = \varphi_e^1(x)$.

2.2.1 Basic results from Recursion Theory

In this thesis we will assume some familiarity with the basic results of Recursion Theory, such as the following.[Odi87]

1. Lists of natural numbers are effectively encodable with natural numbers. Thus there exists a computable bijection $\langle\ \rangle : \mathbb{N}^* \to \mathbb{N}, (x_1, \dots, x_m) \mapsto \langle x_1, \dots, x_m \rangle$.
2. Arguments can be 'hardcoded' in the program. That is: for all $n, m \in \mathbb{N}$, there is a computable $s_m^n\colon \mathbb{N}^{m+1} \to \mathbb{N}$, such that for all $x_1, \dots, x_{m+n} \in \mathbb{N}$ we have:

   $$\varphi_{s_m^n(e,x_1,\dots,x_m)}(x_{m+1}, \dots, x_{m+n}) \simeq \varphi_e(x_1, \dots, x_m, x_{m+1}, \dots, x_{m+n}).$$

   This result is called **Kleene's s_{mn}-Theorem**.[Kle38]

3. Computation is computable. There are computable UM and UM′ such that:[Tur37][Kle38]

$$\begin{aligned} \mathrm{UM}(p,x) &\simeq \varphi_p(x) \qquad && (p,x\in\mathbb{N}) \\ \mathrm{UM}'(p,x,s). &\simeq \varphi_{p,s}(x) \qquad && (p,x,s\in\mathbb{N}) \end{aligned}$$

2.3 Recursively enumerable sets

Definition 2. Let $\mathcal{W}_e = \{n;\, n\in\mathbb{N};\, \varphi_e(n)\downarrow\} = \operatorname{dom}\varphi^1_e$. $\mathcal{W}_e$ is the **eth recursively enumerable set**. Let $\mathcal{W}_{e,s} = \{n;\, n\in\mathbb{N};\, \varphi^1_{e,s}(n)\downarrow\}$.

Proposition 3 (Kleene [Kle36])**.** *The following are equivalent*

1. *$A = \mathcal{W}_e = \operatorname{dom}\varphi^1_e$ for some e.*
2. *$A = \operatorname{Im}\varphi^1_e$ for some e.*
3. *A has a computable enumeration. That is: there is an e such that:*
   - *if A is finite, then $\varphi^1_e(n)\uparrow$ for $n \geq \#A$ and $A = \{\varphi^1_e(0),\ldots,\varphi^1_e(n-1)\}$.*
   - *if A is infinite, then φ^1_e is total, injective and $A = \{\varphi^1_e(0),\varphi^1_e(1),\ldots\}$.*

Proof. **1 ⇒ 2** Suppose $A = \mathcal{W}_e$ for some $e\in\mathbb{N}$. There is a e' such that

$$\varphi^1_{e'}(x) = \begin{cases} x & \varphi^1_e(x)\downarrow \\ \uparrow & \varphi^1_e(x)\uparrow. \end{cases}$$

Then $x\in\operatorname{Im}\varphi^1_{e'} \Leftrightarrow x\in\operatorname{dom}\varphi^1_e \Leftrightarrow x\in A$.

2 ⇒ 3 Suppose $A = \operatorname{Im}\varphi^1_e$ for some e. Consider the computable f given by:

```
function $f(n)$
    set $A \leftarrow \emptyset$
    set $s \leftarrow 0$
    for $m$ in $\{0,\ldots,s\}$ do
        if $\varphi^s_e(m)\downarrow$ and $\varphi_e(m) \notin A$ then
            append $\varphi_e(m)$ to $A$
            if $\#A = n+1$ then
                return $\varphi_e(m)$
```

The function f returns x if and only if $x = \varphi_{e,s}(m)$ for some $s,m\in\mathbb{N}$. Thus $x\in A$. Furthermore, it is injective. Finally, suppose $x\in A$. Then $\varphi_{e,t}(m) = x$ for some $t,m\in\mathbb{N}$. For some n, in the execution of $f(n)$, the variable s will become bigger than t and m. $f(n)$ might return another number y if $\varphi_{e,s}(m')\downarrow$ for a $m' < m$. However, there are only finitely many m' for which this can happen and for some $n' > n$ we will have $f(n') = x$. Thus f is the desired enumeration.

3 ⇒ 1 Suppose φ_e is a computable enumeration of A. There is a $e'\in\mathbb{N}$ such that $\varphi_{e'}(n) = \mu t[\varphi_e(t) = n]$. Then $\varphi_{e'}(n)\downarrow$ if and only if φ_e enumerates n and thus if and only if $n\in A$. □

Proposition 4 (Post [Pos43], Kleene [Kle36] and Mostowski [Mos47])**.** *Given an $A \subseteq \mathbb{N}$. If both A and $\overline{A}$ are recursively enumerable, then A is decidable.*

Proof. Let $f\colon \mathbb{N}^3 \to \mathbb{N}$ be given by the following algorithm.

```
function $f(e, e', x)$
    set $s \leftarrow 0$
    loop
        if $x \in \mathcal{W}_{e,s}$ then
            return 1
        if $x \in \mathcal{W}_{e',s}$ then
            return 0
        set $s \leftarrow s + 1$
```

There is a computable $p\colon \mathbb{N}^2 \to \mathbb{N}$ such that

$$\varphi_{p(e,e')}(x) = f(e, e', x).$$

Given $e, e' \in \mathbb{N}$ such that $\mathcal{W}_e \cap \mathcal{W}_{e'} = \emptyset$, then

$$\begin{aligned} \varphi_{p(e,e')}(x) = 1 &\iff x \in \mathcal{W}_e \\ \varphi_{p(e,e')}(x) = 0 &\iff x \in \mathcal{W}_{e'} \\ \varphi_{p(e,e')}(x)\uparrow &\iff x \notin \mathcal{W}_e \text{ and } x \notin \mathcal{W}_{e'}. \end{aligned}$$

Thus in partical, if $\mathcal{W}_e = A$ and $\mathcal{W}_{e'} = \overline{A}$, then $\mathbb{1}_A = \varphi^1_{p(e,e')}$. □

2.4 Oracles and Turing degrees

2.4.1 Oracles

Given a $A \subseteq \mathbb{N}$, let φ^A_e denote the eth "computable" partial function $\mathbb{N}^* \to \mathbb{N}$ that is allowed as extra computational step to ask questions about membership of A. One says that the "computable function" may consult an oracle about A. In the case of Turing Machines, one could add an extra tape with on that tape the characteristic function of A.[Tur45] It is convenient and harmless to assume that $\varphi^{\emptyset}_e = \varphi_e$ for all e.

We define $\varphi^{k,A}_e = \varphi^A_e \restriction \mathbb{N}^k \to \mathbb{N}$ and $\mathcal{W}^A_e = \{n;\, n \in \mathbb{N};\, \varphi^{1,A}_e(n)\downarrow\}$.

2.4.2 Turing reductions

If $\mathbb{1}_B = \varphi^{1,A}_e$ for some e, one might say:

- B is recursive in A.
- B is computable with knowledge of A.
- B is **Turing reducible** to A — in symbols: $B \leq_{\mathrm{T}} A$.

$\leq_{\mathrm{T}}$ is transitive and reflexive. Thus we can study $\mathfrak{P}(\mathbb{N})/\leq_{\mathrm{T}}$, the subsets of the natural numbers modulo Turing reducibility. That is: the equivalence classes of the equivalence relation

$$A \equiv_{\mathrm{T}} B \quad\iff\quad A \leq_{\mathrm{T}} B \quad\text{and}\quad B \leq_{\mathrm{T}} A.$$

These equivalence classes are called the **Turing degrees**.[Pos48]

Given a computable set A, then $\mathbb{1}_A = \varphi_e = \varphi_e^{\emptyset}$. Conversely, suppose $\mathbb{1}_A = \varphi_e^{1,B}$ for some A and a computable B, then we can modify e to replace every query to its oracle with an appriopriate computation and thus $\mathbb{1}_A = \varphi_{e'}^{1}$ for some e'. Thus the Turing degree of $\emptyset$ is the set of computable sets.

The Turing degree of $\mathcal{K}$ is strictly above the degree of $\emptyset$. A degree is called enumerable, if there is a recursively enumerable A in that degree. There are no enumerable degrees above $\mathcal{K}$:

Proposition 5 (Post [Pos44])**.** *Given $A \subseteq \mathbb{N}$. If A is recursively enumerable, then $A \leq_{\mathrm{T}} \mathcal{K}$.*

Proof. Find e such that $A = \mathcal{W}_e$. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a total computable function such that $\varphi_{h(x)}(y) = \varphi_e(x)$ for all $e, x, y \in \mathbb{N}$. Then:

$$h(x) \in \mathcal{K} \iff \varphi_{h(x)}(h(x))\downarrow \iff \varphi_e(x)\downarrow \iff x \in \mathcal{W}_e.$$

To complete the proof, let $\varphi_a^{\mathcal{K}}(x) = \mathbb{1}_{\{x;\, h(x)\in\mathcal{K}\}} = \mathbb{1}_A$. □

This allows us to state Post's problem more succinctly:

Problem 6 (Post [Pos44])**.** Is there a Turing degree between that of $\emptyset$ and $\mathcal{K}$?

2.5 Other reductions

The Turing reductions we have seen so far, are of a special kind:

Definition 7 (Post [Pos44])**.** A is **m-reducible to** B (in symbols: $A \leq_{\mathrm{m}} B$) if there is a total computable f such that for all e, we have $e \in A \iff f(e) \in B$.

2.6 Fixed-Point theorem and second recursion theorem

2.6.1 Second recursion theorem

The second recursion theorem states that we can write algorithms that use their own codenumber. More precisely:

Theorem 8 (Kleene [Kle38])**.** *There is a total recursive f such that for all e we have*

$$\varphi_{f(e)}(x) \simeq \varphi_e(f(e), x). \qquad (e, x \in \mathbb{N})$$

Proof. Let S be a total recursive function such that $\varphi_{S(x,y)}(z) \simeq \varphi_x(y, z)$; g such that $\varphi_{g(e)}(z, x) = \varphi_e(S(z, z), x)$ and $f(e) = S(g(e), g(e))$. Then

$$\begin{aligned}\varphi_e(f(e), x) &\simeq \varphi_e(S(g(e), g(e)), x) \\ &\simeq \varphi_{g(e)}(g(e), x) \\ &\simeq \varphi_{S(g(e),g(e))}(x) \simeq \varphi_{f(e)}(x).\end{aligned}$$

□

It is easy and useful to strengthen the theorem.

Corollary 9. *For any $n, m \in \mathbb{N}$, there is a total function $f_m\colon \mathbb{N}^{m+1} \to \mathbb{N}$ such that for all $e, x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathbb{N}$:*

$$\varphi_{f_m(e,y_1,\ldots,y_k)}(x_1, \ldots, x_n) \simeq \varphi_e(f_m(e, y_1, \ldots, y_m), x_1, \ldots, x_n).$$

Proof. Given $n, m \in \mathbb{N}$. There exist computable h_1, h_2 and f_m such that for all $x_1, \dots, x_n, y_1, \dots, y_m, p, e \in \mathbb{N}$, we have

$$\begin{aligned} \varphi_{h_1(e)}(x_1, \dots, x_n) &\simeq \varphi_e(\langle x_1, \dots, x_n \rangle) \\ \varphi_{h_2(e,y_1,\dots,y_m)}(p, \langle x_1, \dots, x_n \rangle) &\simeq \varphi_e(h_1(p), x_1, \dots, x_n, y_1, \dots, y_m) \\ f_m(e, y_1, \dots, y_m) &= h_1(f(h_2(e, y_1, \dots, y_m))). \end{aligned}$$

And thus:

$$\begin{aligned} &\varphi_{f_m(e,y_1,\dots,y_m)}(x_1, \dots, x_n) \\ &\quad \simeq \varphi_{h_1(f(h_2(e,y_1,\dots,y_m)))}(x_1, \dots, x_n) \\ &\quad \simeq \varphi_{f(h_2(e,y_1,\dots,y_m))}(\langle x_1, \dots, x_n \rangle) \\ &\quad \simeq \varphi_{h_2(e,y_1,\dots,y_m)}(f(h_2(e, y_1, \dots, y_m)), \langle x_1, \dots, x_n \rangle) \\ &\quad \simeq \varphi_e(h_1(f(h_2(e, y_1, \dots, y_m))), x_1, \dots, x_n, y_1, \dots, y_m), \end{aligned}$$

as desired. □

On first sight, the proof of the second recursion theorem seems magical. We will study a closely related theorem, called the Fixed-Point theorem, which is very similar to a theorem with the same name in the λ-Calculus.

2.6.2 Fixed-Point theorem

Theorem 10 (Kleene [Kle36], Turing [Tur37], Curry [Cur42] and Rosenbloom [Ros50])**.** *There is a λ-term $\mathcal{Y}$ such that for all λ-terms A*

$$A(\mathcal{Y}A) =_\beta \mathcal{Y}A.$$

Proof. Define ω, C and $\mathcal{Y}$ as follows

$$\omega = \lambda x.\, xx \qquad C = \lambda xyz.\, x(yz) \qquad \mathcal{Y} = \lambda x.\, (Cx\omega)(Cx\omega).$$

Then:

$$\begin{aligned} \mathcal{Y}A &=_\beta (CA\omega)(CA\omega) \\ &=_\beta A(\omega(CA\omega)) \\ &=_\beta A((CA\omega)(CA\omega)) \\ &=_\beta A(\mathcal{Y}A). \end{aligned}$$

□

Theorem 11 (Kleene [Kle38])**.** *There is a total recursive f such that for all e such that φ_e is total we have*

$$\varphi_{f(e)} \simeq \varphi_{\varphi_e(f(e))}.$$

Proof. Define ω, C and f such that

$$\varphi_\omega(x) \simeq \varphi_x(x) \qquad \varphi_{\varphi_{C(x,y)}(z)} \simeq \varphi_{\varphi_x(\varphi_y(z))} \qquad f(e) \simeq \varphi_{C(e,\omega)}(C(e,\omega))$$

and C is computable and $\varphi_{C(x,y)}$ is total. Then f is total and

$$\begin{aligned}\varphi_{f(e)} &\simeq \varphi_{\varphi_{C(e,\omega)}(C(e,\omega))} \\ &\simeq \varphi_{\varphi_e(\varphi_\omega(C(e,\omega)))} \\ &\simeq \varphi_{\varphi_e(\varphi_{C(e,\omega)}(C(e,\omega)))} \\ &\simeq \varphi_{\varphi_e(f(e))},\end{aligned}$$

as desired. □

There is a short proof of the Fixed-Point theorem with the assumption of the second recursion theorem. And vice versa. As an example, we will prove a strong version of the Fixed-Point theorem that we will use often.

Theorem 12. *For every $n \in \mathbb{N}$ there is a $F\colon \mathbb{N} \to \mathbb{N}$ such that for all e, if φ_e is total, then for all $z_1, \ldots, z_n \in \mathbb{N}$ we have:*

$$\varphi_{\varphi_{F(e)}(z_1,\ldots,z_n)} \simeq \varphi_{\varphi_e(\varphi_{F(e)}(z_1,\ldots,z_n),z_1,\ldots,z_n)}.$$

Proof. There is an $a \in \mathbb{N}$ and a computable $F\colon \mathbb{N} \to \mathbb{N}$ such that

$$\begin{aligned}\varphi_a(p,x,z_1,\ldots,z_n) &\simeq \varphi_{\varphi_e(p,z_1,\ldots,z_n)}(x) \\ \varphi_{F(e)}(z_1,\ldots,z_n) &= f_{n+1}(a,e,z_1,\ldots,z_n),\end{aligned}$$

where f_{n+1} is from the strong second recursion theorem (Corollary 9). Thus:

$$\begin{aligned}\varphi_{\varphi_{F(e)}(z_1,\ldots,z_n)}(x) &\simeq \varphi_{f_{n+1}(a,e,z_1,\ldots,z_n)}(x) \\ &\simeq \varphi_a(f_{n+1}(a,e,z_1,\ldots,z_n),x,e,z_1,\ldots,z_n) \\ &\simeq \varphi_{\varphi_e(f_{n+1}(a,e,z_1,\ldots,z_n),z_1,\ldots,z_n)}(x) \\ &\simeq \varphi_{\varphi_e(\varphi_{F(e)}(z_1,\ldots,z_n),z_1,\ldots,z_n)}(x).\end{aligned}$$ □

2.7 Some conventions

1. Given a set $A \subseteq \mathbb{N}$. Where it does not confuse, we write $A(n)$ for $\mathbb{1}_A(n)$.
2. Given a subset $A \subseteq \mathbb{N}$, we write $\overline{A}$ for $\mathbb{N} - A$.
3. Consider the sets $\mathfrak{P}_{\text{fin}}(\mathbb{N}), \mathbb{N}^2, \ldots$ and their standard bijections with the natural numbers. We call a map between any of these sets and/or $\mathbb{N}$ computable if the lifted map between the natural numbers is computable.

3 Varieties of effective undecidability

3.1 Creative sets

Post proved: if both A and $\overline{A}$ are recursively enumerable, then A is recursive. Thus: if A is recursively enumerable but undecidable, every attempt to enumerate the complement must go awry.

Consider $\mathcal{K}$. Let $\mathcal{W}_e \subseteq \overline{\mathcal{K}}$. Then $e \notin \mathcal{K}$, because if $e \in \mathcal{K}$, then $e \in \mathcal{W}_e \subseteq \overline{\mathcal{K}}$, which is absurd. Thus also $e \notin \mathcal{W}_e$. We know for all e:

$$\text{if} \quad \mathcal{W}_e \subseteq \overline{\mathcal{K}} \quad \text{then} \quad e \in \overline{\mathcal{K}} - \mathcal{W}_e.$$

Thus every attempt to enumerate the complement of $\mathcal{K}$ forgets some element we can effectively determine beforehand. This has inspired the following definition.

Definition 13 (Post [Pos44] and Myhill [Myh55])**.** A recursively enumerable set A is called **creative** if there exists a (total) computable $f \colon \mathbb{N} \to \mathbb{N}$ such that for all e

$$\text{if} \quad \mathcal{W}_e \subseteq \overline{A} \quad \text{then} \quad f(e) \in \overline{A} - \mathcal{W}_e.$$

Proposition 14. *The complement of a creative set contains an infinite recursively enumerable subset.*

Proof. Suppose A is creative via f. Let w be the computable map $\mathfrak{P}_{\text{fin}}(\mathbb{N}) \to \mathbb{N}$ such that $\mathcal{W}_{w(A)} = A$ for all $A \in \mathfrak{P}_{\text{fin}}(\mathbb{N})$ and

$$B_0 = \emptyset \qquad B_{n+1} = B_n \cup \{f(w(B_n))\} \qquad B = \bigcup_n B_n.$$

Then B is a recursively enumerable, infinite subset of $\overline{A}$. □

Creative sets are certainly not a solution to Post's problem:

Theorem 15 (Myhill [Myh55])**.** *A set is creative if and only if it is m-complete.*

Proof. Suppose A is m-complete via f; that is: $e \in \mathcal{K} \Leftrightarrow f(e) \in A$. Let h be such that $z \in \mathcal{W}_{h(e)} \Leftrightarrow f(z) \in \mathcal{W}_e$. Suppose $\mathcal{W}_e \subseteq \overline{A}$. Observe that:

1. If $\mathcal{W}_e \subseteq \overline{A}$, then $\mathcal{W}_{h(e)} \subseteq \overline{\mathcal{K}}$, since $z \in \mathcal{W}_{h(e)} \Leftrightarrow f(z) \in \mathcal{W}_e \subseteq \overline{A} \Rightarrow z \in \overline{\mathcal{K}}$.
2. If $\mathcal{W}_e \subseteq \overline{\mathcal{K}}$ then $e \in \overline{\mathcal{K}} - \mathcal{W}_e$, for $e \in \mathcal{K}$ implies $e \in \mathcal{W}_e$ and therefore $\mathcal{W}_e \not\subseteq \overline{\mathcal{K}}$.

And thus

$$\mathcal{W}_e \subseteq \overline{A} \Rightarrow \mathcal{W}_{h(e)} \subseteq \overline{\mathcal{K}} \Rightarrow h(e) \in \overline{K} - \mathcal{W}_{h(e)} \Rightarrow f(h(e)) \in \overline{A} - \mathcal{W}_e,$$

which makes A creative via $f \circ h$.

Conversely, suppose A is creative via f. The strong Fixed-Point theorem (Theorem 12) ensures there is a computable $g \colon \mathbb{N} \to \mathbb{N}$ such that

$$\mathcal{W}_{g(z)} = \begin{cases} \{f(g(z))\} & \text{if } z \in \mathcal{K} \\ \emptyset & \text{otherwise.} \end{cases}$$

Then A is m-complete via $f \circ g$:

- Suppose $z \in \mathcal{K}$. Then $\mathcal{W}_{g(z)} = \{f(g(z))\}$. We must have $f(g(z)) \in A$, since otherwise $\mathcal{W}_{g(z)} \subseteq \overline{A}$ and thus $f(g(z)) \in \overline{A} - \mathcal{W}_{g(z)}$, which is absurd.
- Conversely, suppose $z \in \overline{\mathcal{K}}$. Then $\mathcal{W}_{g(z)} = \emptyset$. Hence $\mathcal{W}_{g(z)} \subseteq \overline{A}$. And consequently $f(g(z)) \in \overline{A} - \mathcal{W}_{g(z)} = \overline{A}$. □

Later on we will look at weaker varieties of creativity. First we turn our attention at a different class of undecidable sets.

3.2 Simple sets

Recall that decidable sets have recursively enumerable complements and that the complement of a creative set is not recursively enumerable, but contains an infinite recursively enumerable subset.

There are recursively enumerable sets that are undecidable because their infinite complement does not contain an infinite recursively enumerable set. These are called simple.

Definition 16 (Post [Pos44]). A recursively enumerable set A is called **simple** if $\overline{A}$ is infinite but does not contain an infinite recursively enumerable subset.

Post invented and investigated simple sets and sets with even stronger similar properties (hypersimple and hyperhypersimple sets) as possible solutions to his problem. Most simple sets that are easy to construct tend to be T-complete, as we will show at the end of this subsection Post did look in the right direction, for the solution of Friedberg, Muchnik and others after them are often *simple*.

Example 17 (Post [Pos44]). To create a simple set S we must ensure that:

1. $\overline{S}$ is infinite and
2. for every e, if $\mathcal{W}_e$ is infinite, then $\mathcal{W}_e \cap S \neq \emptyset$.

We dovetail the computation of all recursively enumerable sets. For every e, we will put the first $x > 2e$, for which we compute $x \in \mathcal{W}_e$, in S. To wit:

$$S = \{x;\ \exists s, e[x > 2e; x \in \mathcal{W}_{e,s} \text{ and } \nexists y > 2e[x \neq y \text{ and } y \in \mathcal{W}_{e,s}]]\}.$$

Simple sets are quite abundant: there is one in every recursive enumerable degree. It will take some work and new notions to show this. However, this work will proof useful later on.

3.2.1 Retraceable sets

Definition 18 (Dekker and Myhill [DM58]). A principal enumeration a of a set A (that is: $A = \{a_0 < a_1 < \ldots\}$ is called **retraceable** if there is a (total) computable $f\colon \mathbb{N} \to \mathbb{N}$ such that

$$f(a_0) = a_0 \qquad f(a_{n+1}) = a_n.$$

Note that in general the enumeration a is not computable.

Definition 19. A set A is called **immune** if it is infinite, but does not contain an infinite recursively enumerable subset.

Remark 20. A set A is simple if and only if it is recursively enumerable and its complement is immune.

Proposition 21 (Dekker and Myhill [DM58])**.** *A retraceable set is either recursive or immune.*

Proof. Suppose A is retraceable via f and not immune. Then there is an infinite recursively enumerable $B \subseteq A$. We want to prove that A is recursive. That is, we want a decision method whether $x \in A$. Find a $b \in B$ such that $x < b$. There is one, since B is infinite. Then calculate $f^1(b), f^2(b), \dots, f^b(b)$. Note that these are all the elements of A smaller than b. Thus x is among them if and only if $x \in A$. □

Corollary 22. *An undecidable recursively enumerable set, which has a retraceable complement, is simple.*

3.2.2 Deficiency sets

Let A be a undecidable recursively enumerable set enumerated without repetition by a computable $a\colon \mathbb{N} \to A$. Note that if a is ascending ($a(0) < a(1) < \dots$), then A is decidable.

Definition 23 (Dekker and Myhill [DM58])**.** The **deficiency set** of a recursively enumerable set A with respect to a computable enumeration without repetitions $a\colon \mathbb{N} \to A$ is

$$\mathcal{D}_a = \{s;\, \exists t > s[a(t) < a(s)]\}.$$

Remark 24. The elements of $\overline{\mathcal{D}_a}$ are called the **true stages** of the enumeration. They are interesting because:

$$x \in A \iff x \in \{a(0), \dots, a(s_0)\} \quad \text{where} \quad s_0 = \mu s \in \overline{\mathcal{D}_a}[a(s) > x].$$

This directly leads to the following proposition.

Proposition 25 (Dekker and Myhill [DM58])**.** *For all enumerations $a\colon \mathbb{N} \to A$ without repetitions of A, we have*

1. $A \equiv_{\mathrm{T}} \mathcal{D}_a$
2. $\overline{\mathcal{D}_a}$ *is retraceable and thus (by corollary 22) $\mathcal{D}_a$ is simple.*

Proof. $A \leq_{\mathrm{T}} \mathcal{D}_a$ by the preceding remark. Conversely, to decide whether $s \in \mathcal{D}_a$, given A we check whether

$$A \cap \{0, \dots, a(s)\} = \{a(0), \dots, a(s)\}.$$

This is precisely the case when s is a true stage and thus $s \notin \mathcal{D}_a$.

To show that $\overline{\mathcal{D}_a}$ is retraceable, we have to compute the element preceding a given $s \in \overline{\mathcal{D}_a}$ (if it exists): we pick the largest $t < s$ such that

$$\{a(0), \dots, a(s)\} \cap \{0, \dots, a(t)\} = \{a(0), \dots, a(t)\}.$$

If there is no such t, then there is no predecessor. □

Corollary 26. *There is a simple set in every undecidable recursively enumerable degree.*

3.2.3 Effectively simple sets and Arslanov's criterium

Although there are simple sets that fail to be T-complete, any simple set for which we know it is simple in an effective way *is* T-complete.

Definition 27 (Smullyan [Smu64])**.** A recursively enumerable set A is called **effectively simple** if there is a computable $f\colon \mathbb{N} \to \mathbb{N}$ such that

$$\text{if} \quad \mathcal{W}_e \subseteq \overline{A} \quad \text{then} \quad |\mathcal{W}_e| \leq f(e).$$

Example 28. Post's simple set (example 17) is effectively simple via $e \mapsto 2e+1$.

We will first need to study the subtle completeness criterium of Arslanov, which can be understood as a generalization of the Fixed Point Theorem of Recursion Theory to any function of incomplete degree.

Theorem 29 (Martin [Mar66], Lachlan [Lac68] and Arslanov [Ars81])**.** *A recursively enumerable set A is T-complete if and only if there exists a function $f \leq_{\mathrm{T}} A$ without fixed points. That is: for all e, $\mathcal{W}_e \neq \mathcal{W}_{f(e)}$.*

Proof. Suppose A is T-complete. Then we can compute $\{e;\, 0 \notin \mathcal{W}_e\}$ in A. Define f such that

$$\mathcal{W}_{f(e)} = \begin{cases} \mathbb{N} & 0 \notin \mathcal{W}_e \\ \emptyset & \text{otherwise.} \end{cases}$$

Then $\mathcal{W}_{f(e)}$ and $\mathcal{W}_e$ differ on 0 for every e. Thus f is fixed-point free.

Conversely, suppose $f \leq_{\mathrm{T}} A$ and f has no fixed points. Let e be such that $f = \varphi_e^{1,A}$. For every s, the function $\varphi_{e,s}^{1,A_s}$ is an approximation of f, but because it has fixed-points it differs from f.

Let s be the modulus of $\mathcal{K}$. That is:

$$s_x = \begin{cases} \mu s[x \in \mathcal{K}_s] & x \in \mathcal{K} \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $s \leq_{\mathrm{T}} A$ then $\mathcal{K} \leq_{\mathrm{T}} A$.

Find a g using the Fixed Point Theorem such that

$$\mathcal{W}_{g(x)} = \begin{cases} \mathcal{W}_{\varphi_{e,s_x}^{1,A_{s_x}}(g(x))} & x \in \mathcal{K} \\ \emptyset & \text{otherwise.} \end{cases}$$

Since both f and g are total, we can compute $\psi \leq_{\mathrm{T}} A$ such that

$$f(g(x)) = \varphi_{e,\psi_x}^{1,A_{\psi_x}}(g(x)).$$

If $x \in \mathcal{K}$ then $g(x)$ is a fixed-point of $\varphi_{e,s_x}^{1,A_{s_x}}$ and thus $s_x < \psi_x$. If $x \notin \mathcal{K}$, then $s_x = 0 < \psi_x$. Consequently $x \in \mathcal{K} \Leftrightarrow x \in \mathcal{K}_{\psi_x}$. Hence $\mathcal{K} \leq_{\mathrm{T}} A$. □

Proposition 30 (Martin [Mar66])**.** *Every effectively simple set is T-complete.*

Proof. Let A be effectively simple via f. Define $g \leq_{\mathrm{T}} A$ such that

$$\mathcal{W}_{g(e)} = \{\text{the first } f(e)+1 \text{ elements of } \overline{A}\}.$$

Then g cannot have fixed-points: if $\mathcal{W}_e = \mathcal{W}_{g(e)}$ then $\mathcal{W}_e \subseteq \overline{A}$, however $|\mathcal{W}_e| = f(e)+1$, which is absurd. Thus A is T-complete by the criterium of Arslanov. □

3.3 Weakly and strongly effectively undecidable sets

3.3.1 Plain w.e.u. and creative sets

In the definitions of simple and creative sets we effectively thwart properties of decidable sets. This is an indirect approach. One could wonder what the status is of the following direct approach.

Definition 31. A recursively enumerable set A is called **(weakly) effectively undecidable** (**w.e.u.**) if there exists a computable $f\colon \mathbb{N} \to \mathbb{N}$ such that for all e

$$\text{if} \quad \varphi_e(f(e))\downarrow \quad \text{then} \quad \varphi_e(f(e)) \neq A(f(e)).$$

The usual argument that $\mathcal{K}$ is undecidable also shows that $\mathcal{K}$ is w.e.u. Actually, with a slight modification it shows that every m-complete set is w.e.u.

Proposition 32. *Every m-complete set is w.e.u.*

Proof. Suppose A is recursively enumerable and m-complete via f. That is: $e \in \mathcal{K} \Leftrightarrow f(e) \in A$. Let h be the computable function such that

$$\varphi_{h(e)}(x) = \begin{cases} 0 & \varphi_e(f(x)) = 0 \\ \uparrow & \text{otherwise.} \end{cases}$$

Suppose $\varphi_e(f(h(e))) = A(f(h(e)))$. Then

$$f(h(e)) \in A \Leftrightarrow \varphi_e(f(h(e))) \neq 0 \Leftrightarrow \varphi_{h(e)}(h(e))\uparrow \Leftrightarrow h(e) \notin \mathcal{K} \Leftrightarrow f(h(e)) \notin A,$$

which is absurd. Thus if $\varphi_e(f(h(e))\downarrow$, then $\varphi_e(f(h(e)) \neq A(f(h(e)))$. This shows A is w.e.u. via $f \circ h$. □

However, the w.e.u. sets are not a new class.

Proposition 33 (Veldman)**.** *Every w.e.u. set is creative.*

Proof. Suppose A is w.e.u. witnessed by f. Let $g\colon \mathbb{N} \to \mathbb{N}$ be such that for all e, if $\mathcal{W}_e \subseteq \overline{A}$ then

$$\varphi_{g(e)}(x) = \begin{cases} 0 & x \in \mathcal{W}_e \\ 1 & x \in A \\ \uparrow & \text{otherwise.} \end{cases}$$

- Suppose $\varphi_{g(e)}(f(g(e))) = 0$. Then $f(g(e)) \in \mathcal{W}_e$ by definition of g and thus $f(g(e)) \notin A$. However by w.e.u. $\varphi_{g(e)}(f(g(e))) = 0 \neq 1 = A(f(g(e)))$. Contradiction.

- Suppose $\varphi_{g(e)}(f(g(e))) = 1$. Then $f(g(e)) \in A$ by definition of g. However by w.e.u. $\varphi_{g(e)}(f(g(e))) = 1 \neq 0 = A(f(g(e)))$. Contradiction.

Apparently, as it is the only remaining possibility: $\varphi_{g(e)}(f(g(e)))\uparrow$. And thus by definition of g: $f(g(e)) \notin \mathcal{W}_e$ and $f(g(e)) \notin A$. Hence $f(g(e)) \in \overline{A} - \mathcal{W}_e$. This shows A is creative via $f \circ g$. □

3.3.2 D-w.e.u. and $\mathcal{W}$-w.e.u. sets

The set constructed by Muchnik and Friedberg is not w.e.u.. We can effectively determine at which point the set might differ from a given decidable set, but we might be mistaken a recursively bounded amount of times. More precisely:

Definition 34. A recursively enumerable set A is called **$\mathcal{W}$-w.e.u.**, if there is a computable map $f\colon \mathbb{N} \to \mathbb{N}^2$ such that for all e

1. $|\mathcal{W}_{f(e)_1}| \leq f(e)_2$ and
2. If $\forall z \in \mathcal{W}_{f(e)_1}[\varphi_e(z)\downarrow]$, then $\exists z \in \mathcal{W}_{f(e)_1}[\varphi_e(z) \neq A(z)]$.

Before we will investigate this class of sets, we will restrict our attention to an easier subclass:

Definition 35. A recursively enumerable set A is called **D-w.e.u.**, if there is a computable map $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ such that for all e

$$\text{If} \quad \forall z \in f(e)[\varphi_e(z)\downarrow] \quad \text{then} \quad \exists z \in f(e)[\varphi_e(z) \neq A(z)].$$

One wonders whether there is a solution to Posts problem that is D-w.e.u. This is not the case:

Theorem 36. *Suppose a is a recursive enumeration without repetitions of a D-w.e.u. set A, then the deficiency set $\mathcal{D}_a$ is effectively simple. Hence every D-w.e.u. set is T-complete.*

Proof. Let A be D-w.e.u. via f; α be such that $A = \mathcal{W}_\alpha$ and g be the computable function such that

$$z \in \mathcal{W}_{g(e)} \Leftrightarrow \exists y \in \mathcal{W}_e[a(y) > z \text{ and } z \notin \{a(0), \ldots, a(y)\}].$$

Furthermore, let p be a function (see Proposition 4) such that for all $e_1, e_2 \in \mathbb{N}$, if $\mathcal{W}_{e_1} \cap \mathcal{W}_{e_2} = \emptyset$, then

$$\begin{aligned} \varphi_{p(e_1,e_2)}(z) = 1 &\Leftrightarrow z \in \mathcal{W}_{e_1} \\ \varphi_{p(e_1,e_2)}(z) = 0 &\Leftrightarrow z \in \mathcal{W}_{e_2} \\ \varphi_{p(e_1,e_2)}(z)\uparrow &\Leftrightarrow z \notin \mathcal{W}_{e_1} \cup \mathcal{W}_{e_2}. \end{aligned}$$

Suppose $\mathcal{W}_e \subseteq \overline{\mathcal{D}_a}$. Note that then $\mathcal{W}_{g(e)} \subseteq \overline{A}$. Let $\xi = p(\alpha, g(e))$.

Assume $\varphi_\xi(z)\downarrow$ for each $z \in f(\xi)$. Then there must be a $z \in f(\xi)$ such that $\varphi_\xi(z) \neq A(z)$. Imagine $\varphi_\xi(z) = 1$. Then by definition of p and α, we have $z \in A$. However, by definition of f also $z \notin A$. Contradiction. Apparently $\varphi_\xi(z) = 0$. Thus $f(z) \in A$, but then again by definition of p we should have $\varphi_\xi(z) = 1$. Another contradiction.

Thus there is a $z' \in f(\xi)$ such that $\varphi_\xi(z')$ diverges. Hence $z' \notin A$ and $z' \notin \mathcal{W}_{g(e)}$. Thus for all $y \in \mathcal{W}_e$, we have $a(y) < \max f(\xi)$ and consequently $|\mathcal{W}_e| < \max f(\xi)$. □

Corollary 37 (Smullyan [Smu64])**.** *The deficiency set of $\mathcal{K}$ is effectively simple.*

We proved that creative sets, m-complete sets and w.e.u. sets coincide. Can we show that D-w.e.u. coincides with another notion of completeness and creativity using similar proofs? It turns out we cannot use the same proofs for D-w.e.u. sets in general. However, we can do it for a special subclass.

3.3.3 D-s.e.u., d-complete and quasicreative sets

Definition 38. A recursively enumerable set A is called **D-strongly effectively undecidable** (**D-s.e.u.**) via $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ if

1. A is D-w.e.u. via f — that is: for all e there is a $z \in f(e)$ such that $A(z) \neq \varphi_e(z)$ and furthermore
2. $f(e) \subseteq \overline{A}$ for all e.

Definition 39 (Shoenfield [Sho57])**.** A recursively enumerable set $A \subseteq \mathbb{N}$ is called **quasicreative** if there is a computable function $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ such that for all e:

$$\text{if} \quad \mathcal{W}_e \subseteq \overline{A} \quad \text{then} \quad f(e) \subseteq \overline{A} \quad \text{and} \quad \exists z \in f(e)[z \in \overline{A} - \mathcal{W}_e].$$

Definition 40 (Jockusch [Joc66])**.** A recursively enumerable set A is called **disjunctive complete** (**d-complete**) if there exists a computable $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ such that

$$e \in \mathcal{K} \iff \exists z \in f(e)[z \in A].$$

In [Sho57] Shoenfield showed that d-completeness and quasicreativeness coincide. Furthermore:

Proposition 41. *For a recursively enumerable A, the following are equivalent:*

1. *A is D-s.e.u.*
2. *A is quasicreative.*
3. *A is d-complete.*

Proof. **D-s.e.u. $\Rightarrow$ quasicreative** Suppose A is D-s.e.u. via f. Let $g\colon \mathbb{N} \to \mathbb{N}$ be such that for all e, if $\mathcal{W}_e \subseteq \overline{A}$ then

$$\varphi_{g(e)}(x) = \begin{cases} 0 & x \in \mathcal{W}_e \\ 1 & x \in A \\ \uparrow & \text{otherwise.} \end{cases}$$

We will prove that A is quasicreative via $f \circ g$. Note that for all e we have $f(e) \subseteq \overline{A}$. Assume that $\forall z \in f(e)[\varphi_{g(e)}(z)\downarrow]$. Then there must be a $z \in f(e)$ such that $\varphi_{g(e)}(z) \neq A(z)$, but both

$$\varphi_{g(e)}(z) = 1 \Rightarrow z \in A \Rightarrow \varphi_{g(e)}(z) \neq 1$$
$$\varphi_{g(e)}(z) = 0 \Rightarrow z \in \mathcal{W}_e \Rightarrow z \notin A \Rightarrow \varphi_{g(e)}(z) \neq 0$$

are absurd. Thus there is a $z \in f(e)$ such that $\varphi_{g(e)}(z)\uparrow$. Then $z \in \overline{A} - \mathcal{W}_e$ and we are done.

quasicreative $\Rightarrow$ d-complete, Shoenfield [Sho57] Suppose A is quasicreative witnessed by f. Let g be the computable function such that

$$\mathcal{W}_{g(e)} = \begin{cases} f(g(e)) & e \in \mathcal{K} \\ \emptyset & \text{otherwise.} \end{cases}$$

Then A is d-complete via $f \circ g$:

- If $e \in \mathcal{K}$ then $\mathcal{W}_{g(e)} = f(g(e))$. Also $\exists z \in f(g(e))[z \in A]$, for otherwise $\mathcal{W}_{g(e)} \subseteq \overline{A}$ and then there is a $z \in f(g(e))$ such that $z \in \overline{A} - \mathcal{W}_{g(e)}$, which is absurd since $f(g(e)) = \mathcal{W}_{g(e)}$.
- Conversely, if $e \in \overline{\mathcal{K}}$ then $\mathcal{W}_{g(e)} = \emptyset$ and $\mathcal{W}_{g(e)} \subseteq \overline{A}$ and thus there is a $z \in f(g(e))$ such that $z \in \overline{A} - \mathcal{W}_{g(e)} = \overline{A}$.

d-complete $\Rightarrow$ D-s.e.u. Suppose A is d-complete witnessed by f. Let h be the computable function such that

$$\varphi_{h(e)}(x) = \begin{cases} 0 & \forall z \in f(x)[\varphi_e(z) = 0] \\ \uparrow & \text{otherwise.} \end{cases}$$

Suppose $\varphi_e(z) = A(z)$ for all $z \in f(h(e))$. Then

$$\begin{aligned} \exists z \in f(h(e))[z \in A] &\Leftrightarrow \exists z \in f(h(e))[\varphi_e(z) \neq 0] \\ &\Leftrightarrow \varphi_{h(e)}(h(e))\uparrow \\ &\Leftrightarrow h(e) \notin \mathcal{K} \\ &\Leftrightarrow \forall z \in f(h(e))[z \notin A], \end{aligned}$$

which is absurd. Thus if for every $z \in f(h(e))$ we know $\varphi_e(z)\downarrow$, then there is a $z \in f(h(e))$ such that $\varphi_e(z) \neq A(z)$ as desired. □

Proposition 42. *The complement of a quasicreative set contains an infinite recursively enumerable subset.*

Proof. Similar to the proof of proposition 14. □

Proposition 43 (Shoenfield [Sho57])**.** *There is a quasicreative set that is not creative.*

We will use a different and finer construction than Shoenfield's to not only show that the notions d-complete and m-complete differ, but also to show that several intermediate distinct completeness notions exist.

3.3.4 n-d-complete sets

Definition 44. For a $n \in \mathbb{N}$, the recursively enumerable set A is called *n-d-complete* via $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ if

1. A is d-complete witnessed by f — that is: for all e
$$e \in \mathcal{K} \iff \exists z \in f(e)[z \in A].$$
2. $|f(e)| \leq n$ for all e.

This completeness notion has a corresponding reduction:

Definition 45. For a $n \in \mathbb{N}$ and recursively enumerable sets A and B, we say A *n-d-reduces* to B (in symbols: $A \leq_{n\text{-}d} B$) via $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ if for all $e \in \mathbb{N}$

1. $|f(e)| \leq n$.
2. $e \in A \iff \exists z \in f(e)[z \in B]$.

Remark 46. 1. This reduction corresponds to the completeness: A is n-d-complete if and only if every recursively enumerable B n-d-reduces to A.

2. However, $\leq_{n\text{-}d}$ is not transitive. See Corollary 48.

Proposition 47. *For all $i < j$ there is a set A that is j-d-complete and not i-d-complete.*

Proof. We will define two recursively enumerable sets A and B such that for all $e \in \mathbb{N}$ the following hold:

P_e**:** $e \in \mathcal{K} \iff je \in A \vee je + 1 \in A \vee \ldots \vee je + j - 1 \in A$.

N_e**:** There is a p_e such that if $\varphi_e(\langle e, p_e \rangle)\downarrow$ and $|D_{\varphi_e(\langle e,p_e \rangle)}| \leq i$ then either

- $\langle e, p_e \rangle \in B$ but $\forall z \in D_{\varphi_e(\langle e,p_e \rangle)}[z \notin A]$ or
- $\langle e, p_e \rangle \notin B$ but $\exists z \in D_{\varphi_e(\langle e,p_e \rangle)}[z \in A]$.

If P_e holds for all $e \in \mathbb{N}$, A is j-d-complete. If N_e holds, then B does not i-d-reduce to A via φ_e. Thus if N_e holds for all $e \in \mathbb{N}$, the set A is not i-d-complete.

At some stage of the construction of A and B, we notice might that $e \in \mathcal{K}$. To meet P_e, we will have to put one of $\{je, \ldots, je + j - 1\}$ in A. However, for every $n \in \{0, \ldots, j - 1\}$, there might be several e' such that $N_{e'}$ currently holds because $je + n \notin A$. For those $N_{e'}$ that are 'injured' because $je + n$ is put in A, we will have to find a new $p_{e'}$, which might involve new elements we wish to keep out of A.

To ensure that eventually all requirements are met, we will allow a requirement to be injured only a finite number of times. To that end, we use the so-called priority method. The requirement N_{e_1} is considered of higher priority than N_{e_2} if $e_1 < e_2$. We will only allow a requirement to be injured in favor of a requirement of higher priority. Thus, when we need to choose $je + n \in \{je, \ldots, je+j-1\}$ to put in A, we consider $C_{je+n} = \{e'; N_{e'} \text{ depends on } je+n \notin A\}$. The priority by which we keep $je + n$ out of A will be the highest priority of C_{je+n}. That is $\min C_{je+n}$. We will put the $je + n$ in A with the lowest priority.

Because $i < j$, a requirement $N_{e'}$ cannot depend on all $\{je, \ldots, je + j - 1\}$ staying out of A. Thus when $e \in \mathcal{K}$ and $N_{e'}$ has the highest priority among the C_{je+n} it will not be injured. We say that the requirements that are injured, are injured in favor of e'. Note that a requirement will only be injured in favor of a requirement of higher priority.

We will describe a program, that generates A and B. The program uses the following variables.

- C_e : A list of z such that 'φ_z is not a i-d-reduction from B to A' depends on $e \notin A$
- p_e : The number of times 'φ_e is not a i-d-reduction from B to A' has injured.

Although there is an infinite amount of variables and the program will perform operations on an infinite amount of them, we can represent this with a finite structure: we do not store the values of each of the variables, but the operations performed. For instance, to represent the variables p_e, we use a finite list of pairs. Each pair $\langle v, e\rangle$ means 'we set all variables p_i to v when $\varphi_e(i) = 1$'. To get the current value of p_i, we simply replay the operations.

The program first initializes its variables and a list of conditions to watch for: the watchlist. Initially, the program watches for $e \in \mathcal{K}$ and $\varphi(\langle e, 0\rangle)\downarrow$ for all $e \in \mathbb{N}$. After initialization, the program starts to compute on the conditions in the watchlist in parallel. When a condition is found to hold, the specified action is executed and the condition is removed from the watchlist.

Without further ado: the program to generate A and B.

```
initially
    add $e \in \mathcal{K}$ for all $e \in \mathbb{N}$ to the watchlist
    add $\varphi_e(\langle e, 0\rangle)\downarrow$ for all $e \in \mathbb{N}$ to the watchlist
    set $p_e \leftarrow 0$ for all $e \in \mathbb{N}$
    set $C_e \leftarrow \langle\rangle$ for all $e \in \mathbb{N}$

when $e \in \mathcal{K}$
    if there is a $0 \le n < j$ such that $C_{je+n} = \langle\rangle$ then
        with that $n$
        put $je + n$ in $A$
    else
        find $n$ such that $\min C_{je+n}$ is maximal in $\{\min C_{je+m}; 0 \le m < j\}$
        put $je + n$ in $A$
        for $z$ in $C_{je+n}$ do
            set $p_z \leftarrow p_z + 1$
            add $\varphi_z(\langle z, p_z\rangle)\downarrow$ to the watchlist
            remove $z$ from $C_a$ for all $a \in \mathbb{N}$

when $\varphi_e(\langle e, p_e\rangle)\downarrow$
    set $P \leftarrow D_{\varphi_e(\langle e,p_e\rangle)}$
    if $|P| \le i$ and $\forall z \in P[z \notin A]$ then
        put $\langle e, p_e\rangle$ in $B$
        for $z \in P$ do
            append $e$ to $C_z$
```

We say that at a stage in the construction, the requirement N_e has settled down if after that moment there will not be a requirement injured in favor of e.

All requirements N_e will settle. We will prove this by induction. Suppose all $e' < e$ will settle.

1. Because all requirements of higher priority will settle, there will be a stage after which e will not be injured anymore. Thus p_e will not change anymore.

2. If for the final p_e it happens to be that $\varphi_e(\langle e, p_e\rangle)\uparrow$, the requirement N_e will not pick elements which it wants to stay out of A. And thus no other requirement will be injured anymore in favor of N_e.

   In the other case, there will be a later stage when it has been discovered that $\varphi_e(\langle e, p_e\rangle)\downarrow$ and acted upon. That is, the program might have put $e \in C_z$ for several z. After that moment, e will not be put in C_z anymore for any z.

3. In the latter case, there is a later stage such that for all e' and $0 \leq n < j$ such that $e \in C_{je'+n}$, either:

   (a) $e' \notin \mathcal{K}$ or

   (b) $e' \in \mathcal{K}$, which has been discovered and acted upon by the program. This might involve injuring another requirement in favor of e.

   After this stage, no requirement will be injured anymore in favor of e. Thus e settles down. ☐

Corollary 48. *$\leq_{n\text{-}d}$ is not transitive.*

Proof. By the previous, we know there is a recursively enumerable $A \subseteq \mathbb{N}$ such that A is $2n$-d-complete, but not n-d-complete. In particular, there is a computable $f\colon \mathbb{N} \to \mathbb{N}^{2n}$ such that

$$x \in \mathcal{K} \iff \exists i \in \{1, \ldots, 2n\}[f(x)_i \in A].$$

And now consider the $B \subseteq \mathbb{N}$ such that

$$\begin{aligned} 2x \in B &\iff \exists i \in \{1, \ldots, n\}[f(x)_i \in A] \\ 2x+1 \in B &\iff \exists i \in \{n+1, \ldots, 2n\}[f(x)_i \in A]. \end{aligned}$$

Then $\mathcal{K} \leq_{n\text{-}d} B \leq_{n\text{-}d} A$, but $\mathcal{K} \not\leq_{n\text{-}d} A$, for otherwise A would be n-d-complete. ☐

Proposition 49. *There is a set A that is d-complete, but not n-d-complete for all n.*

Proof. We will use a variant of the program of the previous proof to generate recursively enumerable A and B such that for all $e, i \in \mathbb{N}$ the following holds

P_e: $e \in \mathcal{K} \iff \langle e, 0\rangle \in A \vee \ldots \vee \langle e, e\rangle \in A$.

N_e^i: There is a p_e^i such that if $\varphi_e(\langle e, p_e^i\rangle)\downarrow$ and $|D_{\varphi_e(\langle e, p_e^i\rangle)}| \leq i$ then either

- $\langle e, p_e^i\rangle \in B$ but $\forall z \in D_{\varphi_e(\langle e, p_e^i\rangle)}[z \notin A]$ or
- $\langle e, p_e^i\rangle \notin B$ but $\exists z \in D_{\varphi_e(\langle e, p_e^i\rangle)}[z \in A]$.

The P_e combined assert that A is d-complete. The requirement N_e^i asserts φ_e is not a i-d-reduction from B to A. Note that N_e^i implies N_e^j for $j < i$, but the converse does not hold.

Pick a computable bijection $\pi\colon \mathbb{N}^2 \to \mathbb{N}$. The requirement N_e^i has higher priority than $N_{e'}^{i'}$ (in symbols: $N_e^i \succ N_{e'}^{i'}$) if $\pi(i, e) < \pi(i', e')$.

```
initially
    add $e \in \mathcal{K}$ for all $e \in \mathbb{N}$ to the watchlist
    add $\varphi_e(\langle e, i, 0\rangle)\downarrow$ for all $e, i \in \mathbb{N}$ to the watchlist
    set $p_e^i \leftarrow 0$ for all $e \in \mathbb{N}$
    set $C_e \leftarrow \langle\rangle$ for all $e \in \mathbb{N}$

when $e \in \mathcal{K}$
    if there is a $0 \leq n \leq e$ such that $C_{\langle e,n\rangle} = \langle\rangle$ then
        with that $n$
        **put** $\langle e,n\rangle$ **in** $A$
    **else**
        **find** $n$ **such that** $\min C_{\langle e,n\rangle}$ is maximal in $\{\min C_{\langle e,m\rangle}; 0 \le m \le e\}$
        **put** $\langle e,n\rangle$ **in** $A$
        **for** $\langle z,i\rangle$ **in** $C'_{\langle e,m\rangle}$ **do**
            **set** $p^i_z \leftarrow p^i_z + 1$
            **add** $\varphi_z(\left\langle z,i,p^i_z\right\rangle)\downarrow$ **to the watchlist**
            **remove** $\langle z,i\rangle$ **from** $C'_a$ **for all** $a \in \mathbb{N}$
            **remove** $\pi(z,i)$ **from** $C_a$ **for all** $a \in \mathbb{N}$

**when** $\varphi_e(\left\langle e,i,p^i_e\right\rangle)\downarrow$
    **set** $P \leftarrow D_{\varphi_e(\langle e,i,p^i_e\rangle)}$
    **if** $\forall z \in P[z \notin A]$ **and** $|P| \le i$ **then**
        **put** $\left\langle e,i,p^i_e\right\rangle$ **in** $B$
        **for** $z \in P$ **do**
            **append** $\langle e,i\rangle$ **to** $C'_z$
            **if** $\neg\exists e'\forall 0 \le n \le e'[\langle e',n\rangle \in P]$ **then**
                **append** $\pi(e,i)$ **to** $C_z$
```

The d-completeness is obvious from the construction. The i-d-incompleteness is not. Again, we say N^i_e settles at a stage if from that moment on no other requirements will be injured in favor of N^i_e. Suppose all $N^{i'}_{e'} \succ N^i_e$ eventualy settle.

1. Because all requirements of higher priority will settle, there will be a stage after which N^i_e will not be injured anymore in favor of another requirement.

   However, it may happen that N^i_e depends on all $\{\langle e',0\rangle, \ldots, \langle e',e'\rangle\}$ staying out of A for some e'. If it is discovered that $e' \in \mathcal{K}$, the requirement N^i_e will be injured.

   This may happen several times, but it will stop because of the following. To be injured in this way, requires an e' such that at least

   - $e' \in \mathcal{K}$, but this must not have been discovered and acted upon at the moment $\varphi_e(\left\langle e,i,p^i_e\right\rangle)\downarrow$ is handled.
   - $e' \le i$, because otherwise $D_{\varphi_e(\langle e,i,p^i_e\rangle)}$ cannot have less than or equal i elements and contain all $\{\langle e',0\rangle, \ldots, \langle e',e'\rangle\}$.

   Thus there are only a finite amount of oppurtunities for this kind of injury. Hence, eventually, N^i_e will not be injured anymore. Thus p_e will not change anymore.

The rest of the argument is the same as for the previous program. □

3.3.5 D-w.e.u. and hyper-simple sets

Recall that D-s.e.u. sets are quasicreative; that quasicreative sets contain an infinite recursively enumerable subset in their complement and thus are not simple. We will construct a simple D-w.e.u. set and thus show that D-w.e.u. and D-s.e.u. are different.

Proposition 50. *There is a simple D-w.e.u. set.*

Proof. We will modify the construction of Post's simple set (see example 17). Let X_e be a computable partition of $\mathbb{N}$ with $|X_e| = e+2$. For instance, let $X_e = \{\frac{1}{2}(e+2)(e-1)-1, \ldots, \frac{1}{2}(e+3)(e+2)-2\}$. For every e

1. Add the first $x \in \mathcal{W}_e$ outside $X_0 \cup \ldots \cup X_e$ to A.

2. Add the first $x \in X_e$ with $\varphi_e(x) = 0$ to A.

Clearly, A is recursively enumerable. Furthermore $|X_e \cap \overline{A}| \geq 1$. By this and (1), A is simple. Let e be given. If there is an $x \in X_e$ with $\varphi_e(x) = 0$ then $\mathbb{1}_A$ will differ from φ_e on the first such x by (2). In the other case there is no $x \in X_e$ with $\varphi_e(x) = 0$. Let $x \in X_e$ be such that $x \notin A$. Then $\varphi_e(x)$ diverges or differs in value with $\mathbb{1}_A$. Thus A is D-w.e.u. via $e \mapsto X_e$. □

Although there is no recursively enumerable subset of the complement of a D-w.e.u. set A, we can recursively enumerate an infinite list of disjoint finite sets, each of which intersects with the complement of our D-w.e.u. set.

Definition 51 (Post [Pos44]).

1. A computable $h\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ such that if $n \neq m$ then $h(n) \cap h(m) =$ is called a **disjoint strong array**.

2. A set A **intersects with a disjoint strong array** h if for every n there is a $z \in h(n)$ such that $z \in A$.

Proposition 52. *If A is D-w.e.u., then there is a disjoint strong array intersecting $\overline{A}$.*

Proof. Suppose A is D-w.e.u. via f. Define $\overline{w}$ to be a computable function such that for all finite sets Y:

$$\varphi_{\overline{w}(Y)}(y) = \begin{cases} 0 & y \in Y \\ 1 & y \notin Y. \end{cases}$$

Then $f(\overline{w}(\emptyset))$ intersects $\overline{A}$. Thus our first try could be:

$$X_0 = f(\overline{w}(\emptyset)) \qquad X_{n+1} = f(\overline{w}(\bigcup_{i \leq n} X_i)).$$

This, however, does not work for it could be possible that $X_1 = X_0 - \overline{A}$. Secondly, X_i is in general not disjoint.

The following algorithm does generate a intersecting disjoint strong array:

```
set $G \leftarrow \emptyset$                    ▷ current approximation of $A$
set $Y \leftarrow \emptyset$                    ▷ already yielded
loop
    set $B \leftarrow f(\overline{w}(G))$       ▷ get the next batch
    if $B \cap Y \neq \emptyset$ then
        set $G \leftarrow G - B$
    else
        yield $B$
        set $Y \leftarrow Y \cup B$
        set $G \leftarrow G \cup B$
```

Note that in every iteration of the loop, by definition of f, there is a $t \in B$ such that $t \in A \cup G$ or $t \notin A \cup G$. If $t \in A \cup G$, then$B \cap G \neq \emptyset$. Note that $G \subseteq Y$ and thus if $Y \cap G = \emptyset$, then with the previous $B \cap \overline{A} \neq \emptyset$ and thus we yield it.

Otherwise we remove the elements of B from G. Since G is finite, the algorithm cannot be stuck in this case. Finally, since we check whether $B \cap Y = \emptyset$ the yielded sets are disjoint. □

Definition 53. 1. A set A is called **hyper-immune** if it is infinite, but there is no disjoint strong array intersection A.

2. A set A is called **hyper-simple** if it is recursively enumerable and has hyper-immune complement.

Thus D-s.e.u. sets cannot be simple and D-w.e.u. sets cannot be hyper-simple.

In his 1944 article Post started his search for an intermediate set by first considering simple sets, then hyper-simple sets and then even so called hyperhyper-simple sets.

3.3.6 D-w.e.u. and wtt-complete sets

We cannot generalize the reasoning of the proof of proposition 41 to w.e.u. sets. However, approaching the problem 'from above' by specialing T-completeness and the Arslanov's criterium is fruitful.

Definition 54 (Friedberg and Rogers [FR59])**.** A recursively enumerable set A is called ***wtt*-complete** if there is an e and a computable $f\colon \mathbb{N} \to \mathbb{N}$ such that $\mathbb{1}_{\mathcal{K}} = \varphi_e^{1,A}$ and the computation of $\varphi_e^{1,A}(x)$ requires only queries to the oracle for elements below $f(x)$.

Proposition 55 (Arslanov [Ars81])**.** *A recursively enumerable set $A \subseteq \mathbb{N}$ is wtt-complete if and only if there is a function $f \leq_{\mathrm{wtt}} A$ without fixed-points.*

Proof. Follow the original proof (theorem 29) and note that with the stronger assumptions we can replace T-reduction with wtt-reductions. □

Theorem 56. *A set is D-w.e.u. if and only if it is wtt-complete.*

Proof. Suppose A is D-w.e.u. witnessed by f. Define $g \leq_{\mathrm{wtt}} A$ such that

$$\varphi_{g(e)}(x) = \begin{cases} 1 & x \in f(e) \text{ and } x \in A \\ 0 & x \in f(e) \text{ and } x \notin A \\ \uparrow & \text{otherwise.} \end{cases}$$

Imagine g has a fixed-point e, then for all $z \in f(e)$ we know $\varphi_e(z) = \varphi_{g(e)}(z) = A(z)$. However, by D-w.e.u., there must be a z such that $\varphi_e(z) \neq A(z)$. Contradiction. Thus g is fixed-point free and consequently by Arslanov's criteria for wtt-completeness, A is wtt-complete.

Conversely, suppose A is wtt-complete. Then there is an a and a $f\colon \mathbb{N} \to \mathbb{N}$ such that $\mathcal{K} = \varphi_a^{1,A}$ and for all x the computation of $\varphi_a^A(x)$ only requires elements less than $f(x)$ of A. Let e be given. There is a computable $g\colon \mathbb{N} \to \mathbb{N}$

such that $\varphi_a^{\varphi_e} \simeq \varphi_{g(e)}$. There is a computable $k\colon \mathbb{N} \to \mathbb{N}$ such that $\mathcal{K}(k(e)) \not\simeq \varphi_e(k(e))$. Thus

$$\varphi_a^A(k(g(e))) \simeq \mathcal{K}(k(g(e))) \not\simeq \varphi_{g(e)}(k(g(e))) \simeq \varphi_a^{\varphi_e}(k(g(e))).$$

The computation of $\varphi_a^A(k(g(e)))$ only requires elements less than $f(k(g(e)))$ from A, thus φ_e and A must differ on the first $f(k(g(e)))$ places. That is A is D-w.e.u. witnessed by $x \mapsto \{0, 1, \ldots, f(k(g(x)))\}$. $\square$

3.3.7 *wtt*-complete and weakly quasicreative sets

We saw d-completeness corresponds to quasicreativeness. There is a weakening of quasicreativeness to which *wtt*-completeness corresponds.

Definition 57 (Kanovich [Kan70])**.** A recursively enumerable set A is called **weakly quasicreative** if there exists a computable $f\colon \mathbb{N} \to \mathfrak{P}_{\text{fin}}(\mathbb{N})$ such that for all e, if $\mathcal{W}_e \subseteq \overline{A}$ then there is a $z \in f(e)$ such that $z \in \overline{A} - \mathcal{W}_e$.

Note that the difference with definition of quasicreative is the absence of the requirement that $f(e) \subseteq \overline{A}$ for all $e \in \mathbb{N}$.

Theorem 58 (Kanovich [Kan70])**.** *A set is weakly quasicreative if and only if it is wtt-complete.*

Proof. Suppose A is weakly quasicreative witnessed by f. Define $g \leq_{\text{wtt}} A$ such that $\mathcal{W}_{g(e)} = f(e) \cap \overline{A}$. Imagine g has a fixed-point e. Then $\mathcal{W}_e = \mathcal{W}_{g(e)} = \overline{A} \cap f(e)$, but there must be a $z \in f(e)$ such that $z \in \overline{A} - \mathcal{W}_e = \emptyset$. Contradiction. Thus g has no fixed-points and consequently A is *wtt*-complete by Arslanov's criterium.

Conversely, suppose A is *wtt*-complete. Then by Theorem 56 it is D-w.e.u. via some g. Let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function such that

$$\varphi_{h(e)} = \begin{cases} 0 & x \in \mathcal{W}_e \\ 1 & x \in A \\ \uparrow & \text{otherwise.} \end{cases}$$

Suppose $\mathcal{W}_e \subseteq \overline{A}$. There is a $z \in g(h(e))$ such that $\varphi_{h(e)}(z) \not\simeq A(z)$. If $z \in A$ then $\varphi_{h(e)}(z) = 1 = A(z)$ and also if $z \in \mathcal{W}_e$, then $\varphi_{h(e)}(z) = 0 = A(z)$. Thus $z \in \mathcal{W}_e - \overline{A}$, which shows A is weakly quasicreative via $g \circ h$. $\square$

4 Solutions to Post's problem

We have considered various notions of effective undecidability and have proven that all of them imply T-completeness. That is, except for $\mathcal{W}$-w.e.u, but every undecidable set is $\mathcal{W}$-w.e.u.

A solution to Post's problem is a reursively enumerable set A such that $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} \mathcal{K}$. We will review various constructions of such sets and try to discover how far we can stretch our effective knowledge of their undecidability.

4.1 Two recursively enumerable sets of incomparable degree

Theorem 59 (Friedberg [Fri57]). *There exist recursively enumerable sets A and B such that $A \not\leq_{\mathrm{T}} B$ and $B \not\leq_{\mathrm{T}} A$.*

Since for any X and all computable C, we know $C \leq_{\mathrm{T}} X$, both A and B are undecidable. Furthermore, since for all complete K and recursively enumerable X we know $X \leq T$, both A and B are not T-complete. Thus both A and B are solutions to Post's problem.

4.1.1 Sketch of the construction

For every e we will try to find a_e and b_e for which we can ensure

1. $\varphi_e^B(a_e)\uparrow$ or $\varphi_e^B(a_e) \neq A(a_e)$ and
2. $\varphi_e^A(b_e)\uparrow$ or $\varphi_e^A(b_e) \neq B(b_e)$.

For instance, when we discover that $\varphi_{37}^{B_s}(a_{37} = 0$, we will put a_{37} into A. Here B_s is B as far as it has been defined at the current stage in the construction. The computation of $\varphi_{37}^{B_s}$ depends on a finite part of B. We must make sure that B_s does not differ with B on that part, for otherwise the $\varphi_{37}^{B_s}$ might not equal φ_{37}^{B}.

There might be a b_e in that finite part of B for which we have yet to handle $\varphi_e^{A_s}(b_e)\downarrow$. It might be the case that we would like to put $b_e \in B$, but by doing that would alter the result of φ_{37}^{B}. We could change b_e to a new value outside of the finite part of B on which $\varphi_{37}^{B_s}$ depends. However, if we always do this, we might move b_e an infinite amount of times.

To solve this, we will use priorities: ($x \prec y$ means y has higher priority than x)

$$a_0 \succ b_0 \succ a_1 \succ b_1 \succ \dots$$

If b_e has lower priority than b_{37}, then we will move b_e. Otherwise, we will move b_{37}.

4.1.2 Algorithm

In the construciton we will use the same event-based setup we used in Proposition 47. We will use the following variables:

1. a_e: The element for which we try to ensure $\varphi_e^B(a_e) \not\simeq A(a_e)$.
2. C_e^a: 1 when we handled $\varphi_e^B(a_e)$ and 0 otherwise.

3. U_e^a: the initial number of elements of A on which $\varphi_e^A(b_e)$ depends if $C_e^b = 1$ and -1 otherwise.

b_e, C_e^b and U_e^a are used similarly with the rôles of a/A and b/B swapped. If we write A or B in the algorithm, we refer to A and B as far as they have been defined at that moment.

```
initially
    add ‘ $\varphi_e^A(b_e)\downarrow$ and $C_e^b = 0$ ’ for all $e \in \mathbb{N}$ to the watchlist
    add ‘ $\varphi_e^B(a_e)\downarrow$ and $C_e^a = 0$ ’ for all $e \in \mathbb{N}$ to the watchlist
    set $a_e \leftarrow e$ for all $e \in \mathbb{N}$
    set $b_e \leftarrow e$ for all $e \in \mathbb{N}$
    set $U_e^a \leftarrow -1$ for all $e \in \mathbb{N}$
    set $U_e^b \leftarrow -1$ for all $e \in \mathbb{N}$
    set $C_e^a \leftarrow 0$ for all $e \in \mathbb{N}$
    set $C_e^b \leftarrow 0$ for all $e \in \mathbb{N}$

when $\varphi_e^A(b_e)\downarrow$ and $C_e^b = 0$
    set $C_e^b \leftarrow 1$.
    find $u$ such that $\varphi_e^A(b_e)$ depends on at most the first $u$ elements of $A$.
    set $U_e^a \leftarrow u$.
    for $e' > e$ such that $a_{e'} \leq u$ and $C_{e'}^a = 0$ do
        set $a \leftarrow \mu a[A(a) = 0$ and $\max_x U_x^a \leq a$ and $\forall e''[a_{e''} \neq a]]$.
        remove ‘ $\varphi_{e'}^B(a_{e'})\downarrow$ and $C_{e'}^a = 0$ ’ from the watchlist
        set $a_{e'} \leftarrow a$
        add ‘ $\varphi_{e'}^B(a_{e'})\downarrow$ and $C_{e'}^a = 0$ ’ to the watchlist
    for $e' > e$ such that $U_{e'}^b \geq b_e$ and $C_{e'}^a = 1$ do
        set $C_{e'}^a \leftarrow 0$
        set $U_{e'}^b \leftarrow -1$
        set $b \leftarrow \mu a[A(a) = 0$ and $\max_x U_x^a \leq a$ and $\forall e''[a_{e''} \neq a]]$.
        set $a_{e'} \leftarrow a$
        add ‘ $\varphi_{e'}^B(a_{e'})\downarrow$ and $C_{e'}^a = 0$ ’ to the watchlist

when $\varphi_e^B(a_e)\downarrow$ and $C_e^a = 0$
    Similarly, with $A/a$ swapped with $B/b$.
```

4.1.3 Analysis of effective undecidability

a_0 is never moved. b_0 is only moved in favor of a_0. Thus at most once. a_1 is only moved in favor of b_0. b_0 will cause at most two moves of a_1, thus a_1 is moved at most twice. Et cetera.

In general, let M_{a_e} denote the number of times a_e is moved. Then

$$M_{a_e} = \sum_{b_{e'} \succ a_e} M_{b_{e'}} + 1 \qquad M_{b_e} = \sum_{a_{e'} \succ b_e} M_{a_{e'}} + 1.$$

Proposition 60. *$M_{a_n} = F_{2(n+1)} - 1$ and $M_{b_n} = F_{2(n+1)+1} - 1$, where F_n is* *the nth Fibonacci number. That is:*

$$F_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F_{n-2} + F_{n-1} & n > 1. \end{cases}$$

Proof. By induction.

Base

$$\begin{aligned} M_{a_0} &= 0 = F_2 - 1 = F_{2(0+1)} - 1 \\ M_{b_0} &= M_{a_0} + 1 = 1 = F_3 - 1 = F_{2(0+1)+1} - 1. \end{aligned}$$

Step Suppose $M_{a_n} = F_{2(n+1)} - 1$ and $M_{b_n} = F_{2(n+1)+1} - 1$. Then

$$\begin{aligned} M_{a_{n+1}} &= M_{a_n} + M_{b_n} + 1 \\ &= F_{2(n+1)} - 1 + F_{2(n+1)+1} - 1 + 1 \\ &= F_{2((n+1)+1)} - 1 \\ M_{b_{n+1}} &= M_{b_n} + M_{a_{n+1}} + 1 \\ &= F_{2(n+1)+1} - 1 + F_{2(n+2)} - 1 + 1 \\ &= F_{2(n+2)+1} - 1. \end{aligned}$$ □

For the final a_e we know $\varphi_e^B(a_e) \neq A(e)$. If e does not use the oracle, then $\varphi_e(a_e) \neq A(a_e)$.

4.2 A n^2-$\mathcal{W}$-w.e.u solution

Definition 61. A recursively enumerable set is called f-$\mathcal{W}$-w.e.u if there is a computable $g\colon \mathbb{N} \to \mathbb{N}$ such that for all e

1. $|\mathcal{W}_{g(e)}| \leq f(e)$
2. If $\forall z \in \mathcal{W}_{g(e)}[\varphi_e(z)\downarrow]$, then $\exists z \in \mathcal{W}_{g(e)}[\varphi_e(z) \neq A(z)]$.

Thus the set A constructed in Subsection 4.1 is $(F_{2(n+1)} - 1)$-$\mathcal{W}$-w.e.u. We can modify the construction a bit to get a sharper result. In the original construction the priorityorder of the requirements is:

$$a_0 \succ b_0 \succ a_1 \succ b_1 \succ \ldots$$

Given $A_0, A_1, \ldots \in \mathbb{N}$. If we rearrange the priorities as follows.

$$\begin{aligned} a_0 \succ \ldots \succ a_{A_0-1} \succ b_0 \succ a_{A_0} \succ \ldots \succ a_{A_0+A_1-1} \succ b_1 \\ \succ a_{A_0+A_1} \succ \ldots \end{aligned}$$

Then $M_{a_e} = M_{a_{e+1}} = \cdots = M_{a_{e+A_e-1}}$. Define

$$\alpha_n = M_{a_{A_0+\cdots+A_{n-1}}} \quad \text{and} \quad \beta_n = M_{b_n}.$$

It is easy to verify that for $n > 0$:

$$\begin{aligned} \alpha_0 &= 0 & \beta_0 &= A_0 \\ \alpha_n &= \alpha_{n-1} + \beta_{n-1} + 1 & \beta_n &= \beta_{n-1} + A_n(\alpha_n + 1). \end{aligned}$$

And thus

$$\begin{aligned}\alpha_n &= \alpha_{n-1} + 1 + \sum_{i=0}^{n-1} A_i(\alpha_i + 1) \\ &\leq \alpha_{n-1} + 1 + \sum_{i=0}^{n-1} A_1(\alpha_{n-1} + 1) \\ &= (A_0 + \ldots + A_{n-1} + 1)(\alpha_{n-1} + 1).\end{aligned}$$

And consequently:

Proposition 62. *Given any order-preserving computable $f\colon \mathbb{N} \to \mathbb{N}$ such that for every $N \in \mathbb{N}$, there exists a $M \in \mathbb{N}$ such that*

$$f(M + N) \geq (N + M + 1)(f(N) + 1).$$

There is an $A \subseteq \mathbb{N}$ that is f-$\mathcal{W}$-w.e.u and $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} \mathcal{K}$.

Proof. It is sufficient to find $A_0, A_1, \ldots \in \mathbb{N}$ such that for any n we have

$$f(A_0 + \cdots + A_{n-1}) \geq \alpha_n.$$

By assumption on f here is a N such that

$$f(N + 0) \geq (N + 1)(f(0) + 1)$$

and thus

$$\begin{aligned}&\geq (N + 1) \\ &\geq (N + 1)(\alpha_0 + 1) \\ &\geq \alpha_1.\end{aligned}$$

Pick this N as A_0.

Suppose we already found suitable $A_0, \ldots, A_{n-1} \in \mathbb{N}$. Then by assumption on f there is a N such that:

$$\begin{aligned}&f(A_1 + \cdots + A_{n-1} + N) \\ &\qquad \geq (A_1 + \cdots + A_{n-1} + N + 1)(f(A_1 + \cdots + A_{n-1}) + 1)\end{aligned}$$

and thus

$$\begin{aligned}&\geq (A_1 + \cdots + A_{n-1} + N + 1)(\alpha_n + 1) \\ &\geq \alpha_{n+1}.\end{aligned}$$

Thus use N for A_n. □

Corollary 63. *There is a n^2-$\mathcal{W}$-w.e.u solution to Post's problem. That is: there is a recursively enumerable A such that A is n^2-$\mathcal{W}$-w.e.u and $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} \mathcal{K}$.*

5 Conclusion

First we investigated some new direct notions of effective undecidability. They turned out to be equivalent to previously investigated notions of completeness and creativity.[5]

completeness	creativity	effective undecidability
wtt-complete	weakly quasicreative	D-w.e.u.
d-complete	quasicreative	D-s.e.u.
	$\vdots$	
n-d-complete	(n-creative)	(n-D-s.e.u.)
	$\vdots$	
2-d-complete	(2-creative)	(2-D-s.e.u.)
m-complete	creative	w.e.u.

This leads to the following question.

Open Problem 64. Is there a notion of effective undecidability that is equivalent to T-completeness?

Then we investigated the effective undecidability of the existing constructions of sets that are solutions to Post's problem. We saw we could find a n^2-$\mathcal{W}$-w.e.u solution. Can we do better?

Open Problem 65. Is there a recursively enumerable A such that $\emptyset <_{\mathrm{T}} A <_{\mathrm{T}} \mathcal{K}$ and A is f-$\mathcal{W}$-w.e.u. for a bounded computable $f\colon \mathbb{N} \to \mathbb{N}$?

[5]The notions in paranthesis were not covered in this thesis.

A Bibliography and index

References

[Ars81] M.M. Arslanov, *On some generalizations of the fixed-point theorem*, Sov. Math. **228** (1981), 9–16.

[BG00] H.P. Barendregt and Herman Geuvers, *Introduction to lambda calculus*, 2000.

[Chu33] A. Church, *A set of postulates for the foundation of logic (second paper)*, Ann. Math. **34** (1933), 839–864.

[Chu36] ______, *An unsolvable problem of elementary number theory*, Am. J. of Math. **58** (1936), 345–363.

[Cur42] H.B. Curry, *The inconsistency of certain formal logics*, J. Symb. Log. **7** (1942), 115–117.

[DM58] J.C.E. Dekker and J. Myhill, *Retraceable sets*, Can. J. Math. **10** (1958), 357–373.

[Döt91] G. Dötzel, *A function to end all functions*, Algorithm: Recreational Programming **2** (1991), no. 4, 16–17.

[FR59] R.M. Friedberg and R. Rogers, *Reducibilities and completeness for sets of integers*, Zeit. Math. log. Grund. Math. **5** (1959), 117–125.

[Fri57] R.M. Friedberg, *Two recursively enumerable sets of incomparable degree of unsolvability*, Proc. Nat. Acad. Sci. **43** (1957), 236–238.

[Göd31] K. Gödel, *über formal unentscheidbare sätze der principia mathematica und verwandter systeme i*, Monasch. Math. Phys. **38** (1931), 173–198.

[Joc66] C.G. Jockusch, *Reducibilities in recursive function theories*, Ph.D. thesis, M.I.T., 1966.

[Kan70] M.I. Kanovich, *On the decision complexity of r.e. sets*, Dokl. Acad. Nauk **192** (1970), 721–723.

[Kle35] S.K. Kleene, *A theory of positive integers in formal logic*, Am. J. Math. **57** (1935), 153–173.

[Kle36] ______, *General recursive functions on natural numbers*, Math. Ann. **112** (1936), 727–742.

[Kle38] ______, *On notation for ordinal numbers*, J. Symb. Log. **3** (1938), 150–155.

[Lac68] A.H. Lachlan, *Complete recursively enumerable sets*, Proc. Am. Math. Soc. **19** (1968), 99–102.

[Mar66] D.A. Martin, *Completeness, the recursion theorem and effectively simple sets*, Proc. Am. Math. Soc. **17** (1966), 838–842.

[Mos47] A. Mostowski, *On definable sets of positive integers*, Fund. Math. **34** (1947), 81–112.

[Muc58] A.A. Muchnik, *Solution of post's reduction problem and of certain other problems in the theory of algorithms*, Trud. Mosk. Math. Obsc., **7** (1958), 391–401.

[Myh55] J. Myhill, *Creative sets*, Zeit. Math. Log. Grund. Math. **1** (1955), 97–108.

[Odi87] P. Odifreddi, *Classical recursion theory*, Studies in logic and the foundations of mathematics, no. v. 1, Elservier Science Publisher B.V., 1987.

[Pos43] E.L. Post, *Formal reductions of the general combinatorial decision problem*, Am. J. Math. **65** (1943), 197–215.

[Pos44] ______, *Recursively enumerable sets of positive integers and their decision problems*, Bull. Am. Math. Soc. **50** (1944), no. 5, 284–316.

[Pos48] ______, *Degrees of recursive unsolvability*, Bull. Am. Math. Soc. **54** (1948), 641–642.

[Ros50] Rosenbloom, *The elements of mathematical logic*, Dover Press, 1950.

[Sho57] J.R. Shoenfield, *Quasicreative sets*, Proc. Am. Math. Soc. **8** (1957), 964–967.

[Smu64] R.M. Smullyan, *Effectively simple sets*, Proc. Am. Math. Soc. **15** (1964), 893–894.

[Tur36] A.M. Turing, *On computable numbers with an application to the entscheidungsproblem*, Proc. London Math. Soc. **42** (1936), 230–265.

[Tur37] ______, *The p-function in λk-conversion*, J. Symb. Log. **2** (1937), 164.

[Tur45] ______, *Systems of logic based on ordinals*, Proc. Lond. Math. Soc. **45** (1945), 161–228.

[Vel87] W. Veldman, *Dictaat berekenbaarheid en bewijsbaarheid*, 1987.

Index