2 Formalization of Feasible Mathematics

We will now formally define the theory for feasible mathematics characterized by the three informal postulates in the previous section. The theory, named PV in Cook's paper [Coo75], provides a way to define feasibly constructive functions and perform mathematics about them.

2.1 Cook's Theory PV

We will first define Cook's equational theory PV [Coo75]. Intuitively, PV is defined using Cobham's recursive-theoretic characterization of polynomial-time functions [Cob65]. One reason to choose this particular construction of polynomial-time functions is that it is *machine-independent*, i.e., does not depend on any concrete machine model, which makes it easier to work on and (probably) more general.

That is, the theory only deals with equations "$u = v$".

Concretely, we will define:

- *Feasibly constructible functions* are constructed according to Cobham's rules. Moreover, Cobham's characterization allows introducing a function in a recursion manner using the rule of *limited recursion on notation*.
- *Definition axioms* are the Cobham's rules to define the functions.

Explained below :)

Moreover, PV allows a *restricted version* of structural induction rule. Recall in the standard structural induction proofs, we need to prove that if an equation $f(x) = g(x)$, then $f(2x) = g(2x)$ and $f(2x+1) = g(2x+1)$. However, this requires a proper notion of "conditional proofs" that, in Cook's definition, is not natively supported. Instead, Cook [Coo75] introduced a weaker version of induction that is closely related to the Cobham's characterization, and showed that the standard induction proofs can be translated back to the weaker version of induction rule.

2.1.1 Cobham's Characterization of FP

Before proceeding with the formal definition, we fix the notation: ε denotes the empty string, $|a|$ denotes the bit-length of $a \in \{0,1\}^*$, FP is the set of functions $f(x_1, \ldots, x_k) : (\{0,1\}^*)^k \to \{0,1\}^*$ (for a constant $k \in \mathbb{N}$) that are computable by polynomial-time algorithms (i.e., in $\mathsf{poly}(|x_1|, \ldots, |x_n|)$ time). We use $\vec{x} = (x_1, \ldots, x_k)$ to denote a vector of variables for some $k \in \mathbb{N}$.

We first define some simple functions:

- $c(x) := \varepsilon$;
- $\circ(x, y)$ means concatenation of (the binary representation of) x and y;
- $s_i(x) := x \circ i$, $i \in \{0,1\}$;
- $\#(x, y)$ means concatenation of the binary representation of x for $|y|$ times;
- $\mathsf{TR}(x)$ removes the rightmost bit of x;
- $\pi_i(x_1, \ldots, x_k) := x_i$ for $i \in [k]$.

We may use $\circ$ and $\#$ as infix operators $x \circ y$ and $x \# y$, where $\#$ has higher precedence, and denote $\mathsf{TR}(x)$ by $\lfloor x/2 \rfloor$.

In particular, the identity function is $\pi_1(x) := x$.

We introduce two Cobham rules for defining new functions from existing functions: the composition rule and the rule of limited recursion on notation.

- (*Composition*): From $h(x_1, \ldots, x_k)$ and $g_1(\vec{y}), g_2(\vec{y}), \ldots, g_k(\vec{y})$, we can introduce $f(\vec{y}) := h(g_1(\vec{y}), g_2(\vec{y}), \ldots, g_k(\vec{y}))$, where $k \in \mathbb{N}$.

- (*Limited Recursion on Notation*): From $g(\vec{x})$, $h_i(\vec{x}, y, z)$ $(i \in \{0,1\})$, and $k(\vec{x}, y)$, we can introduce $f(\vec{x}, y)$ as:

$$f(\vec{x}, 0) := g(\vec{x}) \qquad f(\vec{x}, s_i(y)) := h_i(\vec{x}, y, f(\vec{x}, y)),\ i \in \{0,1\}; \tag{7}$$

  provided that $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$.

The inequality $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$ ensures the recursion will not generate strings of super-polynomial bit-length.

Let $\mathcal{F}$ be the least class of functions that contains the base functions and is closed under these two rules. It is clear that $\mathcal{F} \subseteq \mathsf{FP}$, and indeed, Cobham [Cob65] proved:

Theorem 2.1 (Cobham [Cob65])**.** $\mathcal{F} = \mathsf{FP}$.

We provide an illustrative examples:

Example 2.1. The *if-then-else* function $\mathsf{ITE}(y, u, v)$ is defined as

$$\mathsf{ITE}(y, u, v) := \begin{cases} u & y = 1; \\ v & y = 0; \\ * & \text{otherwise}; \end{cases}$$

where $*$ is a placeholder meaning that we don't care about the value. To define this function, we use the rule of limited recursion on notation. Let $g(u, v) := \varepsilon$, $h_0(u, v, y, z) := v$, $h_1(u, v, y, z) := u$, and $k(u, v, y) := u \circ v$. Then let $f(y, u, v)$ as:

$$f(u, v, y) := g(u, v); \tag{8}$$
$$f(u, v, s_i(y)) := h_i(u, v, y, f(u, v, y))\ \ i \in \{0,1\}. \tag{9}$$

It can be verified that $|f(u, v, y)| \leq |k(u, v, y)|$, as $|u| \leq |u \circ v|$ and $|v| \leq |u \circ v|$. Let

$$\mathsf{ITE}(y, u, v) := f(\pi_2(y, u, v), \pi_3(y, u, v), \pi_1(y, u, v)) = f(u, v, y)$$

is defined as the composition of f and π_2, π_3, π_1.

We can always permute variables wlog using π and composition :)

To prove Theorem 2.1, we need to show that the (clocked) universal Turing machine, i.e., the function $\mathcal{U}(M, x, t)$ that simulates the Turing machine M on the input x for $|t|$ steps. To see that this is sufficient, we can check that if $M(x)$ runs in time $O(|x|^c)$ for a constant c, then

$$\mathcal{U}(M, x, \overbrace{x \# x \# \ldots \# x}^{c+1 \text{ times}}) = M(x)$$

when the input length is sufficient large.

Note that one can always hard-code the output when the input length is small using the ITE function.

Remark 2.1. We note that our definition of Cobham's family $\mathcal{F}$ is different from the original definition [Cob65] and the standard expositions in literature, where $\mathcal{F}$ is defined as a class of functions over $\mathbb{N}$ rather than $\{0,1\}^*$. There is essentially no difference between these two formulations; we choose the formalization using Boolean strings rather than $\mathbb{N}$ because it appears to be more intuitive from a computational perspective as we don't need to worry about the encoding of natural numbers.

2.1.2 Definition of PV

The Cobham rules are pretty simple and clean except for the presence of the "upper bound function" $k(\vec{x}, y)$ in the rule of limited recursion on notation.

This causes a technical issue in defining feasible mathematics Cobham rules. The rule of limited recursion on notation requires the inequality $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$ to be "true". In feasible definition, however, it is not enough to have the inequality be true. According to the postulate of feasible functions (see Postulate 1), we require that the construction of functions needs to "clearly demonstrate" the feasibility of the function, and the feasibility of f obtained from limited recursion on notation using (g, h_0, h_1, k) is clearly demonstrated only if the inequality $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$ is *clearly demonstrated*.

Intuition of the definition. Cook [Coo75] resolves the issue with a natural idea — inductively define PV *functions* and PV *proofs* simultaneous. Concretely:

- Base functions in Cobham's class $\mathcal{F}$ are order-0 PV functions and their definition axioms are order-0 PV proofs.
- One can introduce an order-$(i+1)$ function f by limited recursion on notation if the inequality $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$ admits an order-i PV proof.

  and thus is clearly demonstrated
- An order-i PV proof can use the definition axioms of order-i PV functions as well as the structural induction over the definition of order-i PV functions.

Finally, *feasible functions* are formally defined as PV functions of finite order, and *feasible proofs* are formally defined as PV proofs of finite order.

Before giving the formal definition, we make a slight technical change to the rule of limited recursion on notation. Instead of proving the inequality $|f(\vec{x}, y)| \leq |k(\vec{x}, y)|$, we require to prove two inequalities $|h_i(\vec{x}, y, z)| \leq |z \circ k_i(\vec{x}, y)|$ for $i \in \{0, 1\}$. It is not hard to show that the updated rule is at most as strong as the original one:

Lemma 2.2. *Let $\mathcal{F}'$ be the class obtained by replacing the inequalities $h_i(\vec{x}, y) \leq k(\vec{x}, y)$ in the rule of limited induction on notation in $\mathcal{F}$ to $|h_i(\vec{x}, y, z)| \leq |z \circ k_i(\vec{x}, y)|$ for $i \in \{0, 1\}$. Then $\mathcal{F}' \subseteq \mathcal{F}$.*

Proof Sketch. We first show that $\mathcal{F}' \subseteq \mathcal{F}$ by structural induction over the Cobham rules (with the updated limited recursion on notation rule). We need to prove that if f is introduced from $g, h_1, h_2 \in \mathcal{F}$ by Equation (7) such that $|h_i(\vec{x}, y, z)| \leq |z \circ k_i(\vec{x}, y, z)|$ for some $k_i \in \mathcal{F}$, $i \in \{0, 1\}$, then $f \in \mathcal{F}$. Indeed, we will prove $f \in \mathsf{FP}$ and apply Cobham's theorem (see Theorem 2.1). This follows from the fact that

We use $\mathsf{BIT}(x, j)$ to denote the j-th bit of x, and $y_{>j}$ be the suffix of y of length $|y| - j$.

$$\begin{aligned} |f(\vec{x}, s_i(y))| &\leq |h_i(\vec{x}, y, f(\vec{x}, y))| \\ &\leq |f(\vec{x}, y)| + |k_i(\vec{x}, y)| && \text{(unfolding } f \text{ recursively)} \\ &\leq |g(\vec{x})| + \sum_{0 \leq j < |s_i(y)|} |k_{\mathsf{BIT}(s_i(y), j+1)}(\vec{x}, s_i(y)_{>j}))| \leq \mathsf{poly}(|\vec{x}|, |y|) \end{aligned}$$

and that $g, h_0, h_1 \in \mathcal{F} = \mathsf{FP}$. (Notice that the the computation of f only requires calling g, h_0, h_1 on inputs of $\mathsf{poly}(|\vec{x}|, |y|)$ length.) □

Indeed, a white-box inspection of the proof of Cobham's theorem shows that $\mathcal{F}' = \mathcal{F}$. We will postpone this to the proof of Theorem 2.6.

Moreover, since we will use the comparison of the bit-length of numbers, we will need to introduce the *iterative trimming function* $\mathsf{ITR}(x, y)$ that removes the leftmost $|y|$ bits of x as the base function, and formalize $|x| \leq |y|$ by the equation $\mathsf{ITR}(x, y) = \varepsilon$.

Definition of PV. Now we present the formal definition of equational theory PV. We first define the base case:

- ε be a constant symbol.
- $s_0(x), s_1(x), \circ(x, y), \#(x, y), \mathsf{TR}(x), \mathsf{ITR}(x, y)$ are function symbols of order 0.
- A *term* of order i is defined by compositions of order-i functions, the constant symbol, and variables, e.g., $\mathsf{ITR}(s_0(s_1(x)), s_1(y))$ is a term of order 0.
- An *equation* of order i is of form $s = t$, where s, t are terms of order i.

where x, y are variables

The definition axioms of the function symbols are proofs of order 0:

$$x \circ \varepsilon = x, \quad x \circ s_i(y) = s_i(x \circ y) \qquad i \in \{0, 1\} \qquad (10)$$

$$x \# \varepsilon = \varepsilon, \quad x \# s_i(y) = x \circ (x \# y) \qquad i \in \{0, 1\} \qquad (11)$$

$$\mathsf{TR}(\varepsilon) = \varepsilon, \quad \mathsf{TR}(s_i(x)) = x \qquad i \in \{0, 1\} \qquad (12)$$

$$\mathsf{ITR}(x, \varepsilon) = x, \quad \mathsf{ITR}(x, s_i(y)) = \mathsf{TR}(\mathsf{ITR}(x, y)) \qquad i \in \{0, 1\} \qquad (13)$$

The intended meaning of the constant symbol ε is the *empty string*. The meaning of a variable is an arbitrary Boolean string. The intended meaning of a provable equation $s = t$ is that $s(\vec{x}) = t(\vec{x})$, where $\vec{x} = (x_1, \ldots, x_n)$ are variables in the terms s and t, s and t are interpreted as functions over $(\{0, 1\}^*)^n$. We also note that s_0 and s_1 do not have definition axiom; the intended meaning of $s_i(x)$ is to append a bit $i \in \{0, 1\}$ to the right of a string x. They are used as the encoding of Boolean strings, e.g., 0011 is encoded as $s_1(s_1(s_0(s_0(\varepsilon))))$.

We assume that one can choose variables from an countably infinite set of alphabets to form terms and introducing new functions (see the function introduction rules below).

Remark 2.2. We note that similar to our formalization of Cobham's class (see Remark 2.1), our treatment of the intended meaning of PV is slightly different from that of Cook's original definition [Coo75], in which he interpreted functions in PV as functions in $\mathbb{N}$. To deal with the encoding of natural numbers, Cook's definition uses the *dyadic notation* of natural numbers, and the functions s_0, s_1 are replaced by $s_1(x) = 2x + 1$ and $s_2(x) = 2x + 2$.

I feel slightly more comfortable working with $\{0, 1\}$ instead of $\{1, 2\}$. What about you?

Function introduction rules. For every $i \geq 1$, a function of order i can be introduced according to one of the following two rules.

- (*Composition*). If t is an order-$(i-1)$ term with variables $\vec{x}$, $f_t^{(0)}$ can be introduced as an order-i function with the definition axiom $f_t^{(0)}(\vec{x}) = t$.

The feasibility of $f_t^{(0)}$ is clearly demonstrated by Cobham's rule of composition.

- (*Recursion*). If $g(\vec{x})$, $h_0(\vec{x}, y, z)$, $h_1(\vec{x}, y, z)$, $k_0(\vec{x}, y)$, $k_1(\vec{x}, y)$ are order-$(i-1)$ functions, and there are order-$(i-1)$ proofs π_i of the equation:

$$\mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon \qquad (14)$$

Informally, eq. (14) means that: $|h_i(\vec{x}, y, z)| \leq |z \circ k_i(\vec{x}, y)|$

for $i \in \{0,1\}$, then $f_\Pi^{(1)}$ (where $\Pi := (g, h_0, h_1, k_0, k_1, \pi_0, \pi_1)$) may be introduced as an order-i function with the definition axioms:

$$f_\Pi^{(1)}(\vec{x}, 0) = g(\vec{x}) \tag{15}$$
$$f_\Pi^{(1)}(\vec{x}, s_i(y)) = h_i(\vec{x}, y, f_\Pi^{(1)}(\vec{x}, y)) \;\; i \in \{0,1\} \tag{16}$$

The feasibility of $f_\Pi^{(1)}$ is clearly demonstrated by Cobham's rule of limited recursion on notation (as modified in Lemma 2.2).

Moreover, any function of order $i-1$ is also an order-i function.

Deduction rules. For every $i \geq 1$, an order-i proof of an order-i equation $s = t$ is a sequence of order-i equations $(e_1, e_2, \ldots, e_\ell)$ for some $\ell \in \mathbb{N}$, where $e_\ell =$ "$s = t$" and the equations are introduced one by one following the rules below:

- (*Logical Rules*). The logical rules for equations follow:
  - (L0): One may introduce $s = s$ for any term s.
  - (L1): If $s = t$ has been introduced, one may introduce $t = s$.
  - (L2): If $s = t$, $t = u$ have been introduced, one may introduce $s = u$.
  - (L3): If $s_1 = t_1, \ldots, s_n = t_n$ has been introduced and $f(x_1, \ldots, x_n)$ is an order-i function symbol with n variables, one may introduce the equation $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$.
  - (L4): If $s = t$ has been introduced, v is an order-i term, and x is an variable, one may introduce $s[x/v] = t[x/v]$, where $s[x/v]$ denote the term obtained from s by substituting all occurrences of x by v.

Exercise: (L0) follows from other rules.

- (*Definition Axioms*). A definition axiom of an order-i function may be introduced without premise.
- (*Structural Induction*). If $g(\vec{x})$, $h_0(\vec{x}, y, z)$, $h_1(\vec{x}, y, z)$ are order i functions, and $f_1(\vec{x}, y)$, $f_2(\vec{x}, y)$ are two functions satisfying that equations

$$f_1(\vec{x}, y) = g(\vec{x}), \quad f_1(\vec{x}, s_i(y)) = h(\vec{x}, y, f_1(\vec{x}, y)), \; i \in \{0,1\} \tag{17}$$
$$f_2(\vec{x}, y) = g(\vec{x}), \quad f_2(\vec{x}, s_i(y)) = h(\vec{x}, y, f_2(\vec{x}, y)), \; i \in \{0,1\} \tag{18}$$

  have all been introduced, then one may introduce $f_1(\vec{x}, y) = f_2(\vec{x}, y)$.

It is tedious to the extent that authors usually omit it in research papers.

This completes the formal definition of PV.

We will use the standard notation $\mathsf{PV} \vdash s = t$ to denote that there is a (finite-order) PV proof of the equation $s = t$. The equational theory PV is defined as the set of equations with (finite-order) PV proofs.

Remark 2.3. Here we explain the intended meaning of variables in PV equations. Variables in an equation $s = t$ should be considered to be *universally quantified*. If $s(\vec{x}) = t(\vec{x})$ is a PV-provable equation, the intended meaning is that for every $\vec{x}$, $s(\vec{x}) = t(\vec{x})$; or equivalently, the *functions* $\vec{x} \mapsto s(\vec{x})$ and $\vec{x} \mapsto t(\vec{x})$ are functionally equivalent. Variables with the same name in an equation are considered to be identical, while variables in different equations should be considered independent even if they share the same name.

Of course, you can force variables in different equations to have different names to avoid any confusion...

2.2 Warmup: Basic Properties of Concatenation

As a warm up, we first prove a couple of basic properties about the concatenation function $\circ$. We will prove that ε is the left identity of concatenation (i.e. $\varepsilon \circ x = x$) and that concatenation is associative (i.e. $(x \circ y) \circ z = x \circ (y \circ z)$).

Note that ε is the right identity of concatenation by the definition axiom.

Proposition 2.3. $\mathsf{PV} \vdash \varepsilon \circ x = x$.

Proof. We define two PV functions $f_1(x) := \varepsilon \circ x$ and $f_2(x) := x$ by the composition rule using terms $\varepsilon \circ x$ and x, respectively. We will prove in PV that $f_1(x) = f_2(x)$, which immediately implies $\varepsilon \circ x = x$ by (L2) transitivity and the definition axioms of f_1 and f_2.

Intuitively, the proof follows from a structural induction on x. We first verify the base case, i.e., $x = \varepsilon$:

$$\begin{aligned}
&\mathsf{PV} \vdash f_1(\varepsilon) = \varepsilon \circ \varepsilon && \text{((L4) substitution } x/\varepsilon \text{ to the definition axiom)}\\
&\mathsf{PV} \vdash x \circ \varepsilon = x && \text{(Axiom eq. (10))}\\
&\mathsf{PV} \vdash \varepsilon \circ \varepsilon = \varepsilon && \text{((L4) substitution } x/\varepsilon \text{ to the equation above)}\\
&\mathsf{PV} \vdash f_1(\varepsilon) = \varepsilon && \text{((L2) transitivity)}
\end{aligned}$$

Similarly, $f_2(\varepsilon) = \varepsilon$. Therefore, let $g = \varepsilon$ be a function with no variable, then $f_j(\varepsilon) = g$ ($j \in \{1, 2\}$) is provable in PV.

Then, we consider the induction step, i.e., x is of form $s_i(x)$ for $i \in \{0, 1\}$. Notice that for every $i \in \{0, 1\}$:

$$\begin{aligned}
&\mathsf{PV} \vdash f_1(s_i(x)) = \varepsilon \circ s_i(x) && \text{((L4) substitution } x/s_i(x) \text{ to the definition axiom)}\\
&\mathsf{PV} \vdash \varepsilon \circ s_i(x) = s_i(\varepsilon \circ x) && \text{((L4) substitution } x/\varepsilon, y/x \text{ to Axiom eq. (10))}\\
(*)\ &\mathsf{PV} \vdash \varepsilon \circ x = f_1(x) && \text{((L1) symmetricity to the definition axiom)}\\
(\star)\ &\mathsf{PV} \vdash s_i(x) = s_i(x) && \text{((L0) reflexivity)}\\
&\mathsf{PV} \vdash s_i(\varepsilon \circ x) = s_i(f_1(x)) && \text{((L3) substitution with } (\star) \text{ and } (*))\\
&\mathsf{PV} \vdash f_1(s_i(x)) = s_i(f_1(x)) && \text{((L2) transitivity)}
\end{aligned}$$

Similarly, one can prove that $f_2(s_i(x)) = s_i(f_2(x))$. Therefore, let $h_i(x, z) = s_i(z)$, we can prove that for $j \in \{1, 2\}$ and $i \in \{0, 1\}$:

$$\mathsf{PV} \vdash f_j(s_i(x)) = h_i(x, f_j(x)).$$

Then, by the rule of structural induction, we can prove that $f_1(x) = f_2(x)$ as they are both inductively defined from (g, h_0, h_1). □

Proposition 2.4. $\mathsf{PV} \vdash (x \circ y) \circ z = x \circ (y \circ z)$.

Proof. We define two functions $f_1(x, y, z) := (x \circ y) \circ z$ and $f_2(x, y, z) := x \circ (y \circ z)$, and it suffices to prove that $f_1(x, y, z) = f_2(x, y, z)$. The strategy is to use the induction rule. Notice that

$$\begin{aligned}
&\mathsf{PV} \vdash f_1(x, y, \varepsilon) = (x \circ y) \circ \varepsilon && \text{((L4) substitution } z/\varepsilon \text{ to the definition axiom)}\\
&\mathsf{PV} \vdash x \circ \varepsilon = x && \text{(Definition axiom of } \circ)\\
&\mathsf{PV} \vdash (x \circ y) \circ \varepsilon = x \circ y && \text{((L4) substitution } x/(x \circ y) \text{ to equation above)}\\
&\mathsf{PV} \vdash f_1(x, y, \varepsilon) = x \circ y && \text{((L2) transitivity)}
\end{aligned}$$

Similarly, we can (with slightly more steps) prove that $\mathsf{PV} \vdash f_2(x, y, \varepsilon) = x \circ y$.

Also notice that for every $i \in \{0, 1\}$,

$$\begin{aligned}
&\mathsf{PV} \vdash f_1(x, y, s_i(z)) = (x \circ y) \circ s_i(z) && \text{((L4) substitution to the definition axiom)}\\
&\mathsf{PV} \vdash x \circ s_i(y') = s_i(x \circ y') && \text{(Definition axiom of } \circ)\\
&\mathsf{PV} \vdash (x \circ y) \circ s_i(z) = s_i((x \circ y) \circ z) && \text{((L4) substitution } x/(x \circ y) \text{ and } y'/z)\\
&\mathsf{PV} \vdash (x \circ y) \circ z = f_1(x, y, z) && \text{((L1) symmetricity to the definition axiom)}\\
&\mathsf{PV} \vdash s_i((x \circ y) \circ z) = s_i(f_1(x, y, z)) && \text{((L3) using above and function } s_i)\\
&\mathsf{PV} \vdash f_1(x, y, s_i(z)) = s_i(f_1(x, y, z)) && \text{((L2) transitivity, twice)}
\end{aligned}$$

Similarly, we can (again with slightly more steps) prove that $\mathsf{PV} \vdash f_2(x, y, s_i(z)) = s_i(f_2(x, y, z))$. Therefore, by the induction rule using $g(x, y) := x \circ y$ and $h_i(x, y, z) = s_i(z)$, we can prove that $f_1(x, y, z) = f_2(x, y, z)$. □

The proof strategy of these two propositions are similar. Take the latter one as an example. To prove the equation $(x \circ y) \circ z = x \circ (y \circ z)$, we are actually proving that both functions (on the LHS and RHS of the equation) are identical to the following recursively defined function

$$f(x, y, \varepsilon) = x \circ y; \quad f(x, y, s_i(z)) = s_i(f(x, y, z)).$$

The function f is not formally introduced, but it is instructive to think of the proof as induction over f.

Note that this particular recursive construction of f is feasible, since it is constructed from an induction on the variable z that takes at most $|z|$ rounds, and $|f(x, y, z)| \leq |f_1(x, y, z)|$ for a function $f_1(x, y, z) := (x \circ y) \circ z$ that is known to be feasible.

f_1 is feasible as $\circ$ is feasible, and the composition of feasible functions is feasible.

Moreover, the proofs that both sides of the original equation are identical to $f(x, y, z)$ are feasible, as they only require the definition axioms of $\circ$. Therefore, we can prove by structural induction over z that both sides of the equation are identical to $f(x, y, z)$, concluding that the equation is true. Throughout the proof, we are only utilizing definition axioms of feasibly constructive functions and the structural induction rule in PV.

2.3 Thesis of Feasible Mathematics

In Cook's paper [Coo75], a *Verifiability Thesis* is proposed to conjecture that the theory PV characterizes the informal concept of *polynomially verifiability*, which we have not explored carefully. Here, we propose an alternate *Thesis of Feasible Mathematics*: PV *is* the theory that characterizes feasible mathematics explained by the three postulates, or more concretely:

> A mathematical statement is provable in feasible mathematics (informally defined by the three postulates) if and only if there is a straightforward formalization of the statement in PV that is provable in PV.

Just like the Church-Turing Thesis and Cook's Verifiability Thesis, the Thesis of Feasible Mathematics is not a precise mathematical statement and cannot be proved. In particular, people may disagree with whether a statement is provable in feasible mathematics, and the standard for "straightforward formalization" is quite subjective. Nevertheless, we will try to explain why it is reasonable to believe that PV suffices for feasible mathematicians.

PV proofs $\Rightarrow$ feasible proofs. We will need to show that the function symbols in PV are feasibly constructive functions, and PV proofs are proofs in the informal notion of feasible mathematics. Since PV functions and proofs are inductively defined simultaneous, we will prove by induction on k that PV functions of order k are feasibly constructive and PV proofs of order k are feasible proofs.

We first consider the base case $k = 0$. Order-0 functions of PV are base functions, which are simple function that clearly demonstrates their feasibility, and thus should be considered as feasibly constructive functions by Postulate 1. Similarly, order-0 PV proofs are simply the definition axioms of order-0 functions that supports their feasibility, and thus are considered feasible by Postulate 2.

We then consider the induction case. Suppose that order-k PV functions are feasibly constructive and order-k PV proofs are feasible. Recall that order-$(k+1)$ PV functions are constructed either by composition or (limited) recursion from order-k PV functions. The composition of feasible function is certainly feasible and clearly demonstrates their feasibility through the construction. The mathematical facts supporting the clear demonstration, or its definition axiom, is allowed to be used as axioms in feasible mathematics.

Now we explain why functions introduced by the limited recursion rule is feasibly constructive. Recall that from g, h_0, h_1, k_0, k_1 we can introduce a function through limited recursion if there are order-k PV proofs of

$$\mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon \quad (i \in \{0, 1\}).$$

By the induction hypothesis, we know that this proof is feasible, given which by the reasoning in Lemma 2.2 it is quite clear that the function f defined by limited recursion from g, h_0, h_1 is feasible. Moreover, the facts that enables the clear demonstration are the equations:

$$\begin{aligned}
&f(\vec{x}, 0) = g(\vec{x}) \\
&f(\vec{x}, s_i(y)) = h_i(\vec{x}, y, f(\vec{x}, y)) \quad (i \in \{0, 1\}) \\
&\mathsf{ITR}(h_i(\vec{x}, y, z), z \circ k_i(\vec{x}, y)) = \varepsilon \quad (i \in \{0, 1\})
\end{aligned}$$

could be included as axioms. Indeed, the first two equations are included as axioms in order-$(k+1)$ PV proofs, and the last equation admits an order-k PV proof (and thus there is no need to include it as an axiom).

Finally, we show that order-$(k+1)$ PV proofs are also provable in feasible mathematics. The definition axioms are valid by Postulate 2. We now show that the structural induction rule is feasibly provable.

Suppose that f_1, f_2, g, h_0, h_1 are order-$(k+1)$ PV functions and thus are feasibly constructive, and the equations

$$\begin{aligned}
f_1(\vec{x}, y) = g(\vec{x}), \quad f_1(\vec{x}, s_i(y)) = h(\vec{x}, y, f_1(\vec{x}, y)), \ i \in \{0, 1\} \\
f_2(\vec{x}, y) = g(\vec{x}), \quad f_2(\vec{x}, s_i(y)) = h(\vec{x}, y, f_2(\vec{x}, y)), \ i \in \{0, 1\}
\end{aligned}$$

are already feasibly provable, where free variables are considered to be universally quantified. We will need to prove feasibly that $\forall \vec{x}\ \forall y\ f_1(\vec{x}, y) = f_2(\vec{x}, y)$. Fix any $\vec{x}$. Note that given any y, the property $f_1(\vec{x}, y) = f_2(\vec{x}, y)$ is feasibly provable, and this we are allowed to prove the equation by a structural induction on y:

- (*Base Case*). $f_1(\vec{x}, \varepsilon) = f_2(\vec{x}, \varepsilon)$ is feasibly provable since both of then are feasibly provably equal to $g(\vec{x})$;

- (*Induction Case*). Recall that by the assumption, it is feasibly provable that $f_j(\vec{x}, s_i(x)) = h(\vec{x}, y, f_j(\vec{x}, y))$ for $j \in \{1, 2\}$ and $i \in \{0, 1\}$. By the induction hypothesis, we know that $f_1(\vec{x}, y) = f_2(\vec{x}, y)$, and thus

$$f_1(\vec{x}, s_i(y)) = h(\vec{x}, y, f_1(\vec{x}, y)) = h(\vec{x}, y, f_2(\vec{x}, y)) = f_2(\vec{x}, s_i(y))$$

  can be feasibly proved.

This induction proof is feasible by Postulate 3. Therefore, $f_1(\vec{x}, y) = f_2(\vec{x}, y)$ is also feasibly provable.

Feasible proofs $\Rightarrow$ PV proofs. The other direction is more tricky as feasible proofs are not formally defined.

For the Church-Turing Thesis, a justification that Turing machine suffices to capture the informal notion of computation is that it simulates seemingly stronger models, including multi-tape Turing machines, Turing machines with two-dimension tapes, Church's λ-calculus, and recursive functions. Similarly, ZFC set theory is generally accepted as a suitable foundation of mathematics because people demonstrates natural formalization of many branches of mathematics in ZFC set theory.

To justify the belief that feasibly provable statements can be formalized and proved in PV, we will need to show that:

1. Any feasibly constructive function that we can imagine is definable in PV in a natural sense. That is, there is a "compiler" from functions that demonstrates their feasibility (possibly written in a reasonable "high-level" programming language) to functions in PV.
2. Any feasibly provable statements that we can imagine is provable in PV in a natural sense. That is, there is a "compiler" from feasible proofs (possibly written in a reasonable "high-level" formal system) to functions in PV.

Note that both properties above are highly non-trivial. For instance, the function introduction rules in PV only allow a restricted version of recursion, and the functions in PV must deal with "raw data" in the form of binary strings rather than in a structured form. Similarly, the deduction rules in PV only allow a special version of induction rather than the general induction rule as discussed in Postulate 3. Nevertheless, just as elaborate modern computer systems can be built from a rather small number of CPU instructions, we will show that the function introduction rules and deduction rules suffice to build a rich system for feasibly constructive functions and feasible mathematics.

2.4 Standard Model of PV

As we demonstrated in the last subsection, Cobham's class $\mathcal{F}, \mathcal{F}'$ and the fact that $\mathcal{F}' \subseteq \mathcal{F} = \mathsf{FP}$ provide an informal explanation that the definition of PV "satisfies" the three postulates of feasible mathematics. Of course we will not be able to formally "prove" this, as the three postulates themselves are intended to describe an informal concept of feasibly constructive proofs.

Nevertheless, we can at least try to formally prove a couple of facts showing that PV does not contradict with the postulates in an obvious sense. We will check that all PV functions are indeed feasible (so that it is not too strong), and all feasible functions are definable in PV (so that it is not too weak).

Standard model of PV. We first formally define the *standard model* (or the *standard interpretation*) of PV, i.e., the "ground-truth" that PV tries to depict. We will first define what does it mean by the model of an equational theory.

Skip this part if you are a *real* feasible mathematician as you may refuse the believe the existence of the ground-truth.

Formally, an (equational) theory $\mathcal{T}$ is a set of equations that are considered true. For instance, the theory PV is defined as the set of equations $s = t$ that admits an order-k proof for some $k \in \mathbb{N}$.

Let $\mathcal{L} = (\Sigma, \mathcal{F})$ be the language of an (equational) theory $\mathcal{T}$, where Σ is the set of constant symbols and $\mathcal{F}$ is the set of function symbols. For PV, we have $\Sigma = \{\varepsilon\}$ and $\mathcal{F}$ be the set of all (finite order) PV functions.

There is only one predicate "=" that is more or less embedded in the logic, so we don't include it in the language.

A *model* or *structure* is defined as a tuple $\mathcal{M} = (\mathcal{U}, \mathcal{I}^{\mathsf{c}}, \mathcal{I}^{\mathsf{f}})$ where:

- $\mathcal{U}$ is the *universe*, i.e., the set of possible values.
- $\mathcal{I}^{\mathsf{c}} : c \mapsto u$ maps every $c \in \Sigma$ to an element $u \in \mathcal{U}$.
- $\mathcal{I}^{\mathsf{f}} : f \mapsto F$ maps every function symbol $f \in \mathcal{F}$ with k variables to a function $F : \mathcal{U}^k \to \mathcal{U}$ over the universe, for every $k \in \mathbb{N}$.

We call $u \in \mathcal{U}$ and $F : \mathcal{U}^k \to \mathcal{U}$ above the *interpretation* of $c \in \Sigma$ and $f \in \mathcal{F}$, respectively. We may denote the interpretation of $c \in \Sigma$ (resp. $f \in \mathcal{F}$) by $c^{\mathcal{M}}$ or $c^{\mathcal{I}^{\mathsf{c}}}$ (resp. $f^{\mathcal{M}}$ or $f^{\mathcal{I}^{\mathsf{f}}}$) for simplicity. Moreover, the mappings $\mathcal{I}^{\mathsf{c}}$ and $\mathcal{I}^{\mathsf{f}}$ naturally induce an interpretation of *terms*, i.e., a mapping from any term t in the language of $\mathcal{L}$ with $k \in \mathbb{N}$ free variables to a function $F : \mathcal{U}^k \to \mathcal{U}$, as follows. Recall that A term is a composition of function symbols and constant symbols, we replace the constant and function symbols by their interpretation in the universe $\mathcal{U}$ according to $\mathcal{I}^{\mathsf{c}}$ and $\mathcal{I}^{\mathsf{f}}$, respectively, and compose the functions and elements in the same way that the term is constructed. We will denote the induced interpretation of a term t as $t^{\mathcal{M}}$ or $t^{\mathcal{I}^{\mathsf{c}}, \mathcal{I}^{\mathsf{f}}}$ for simplicity.

Formally, one can define the mapping inductively over the structure of a term.

We say that an equation $s = t$ *satisfies* a model $\mathcal{M}$, denoted by $\mathcal{M} \vDash s = t$, if $s^{\mathcal{M}}$ is the same function as $t^{\mathcal{M}}$; a theory $\mathcal{T}$ is said to satisfy a model $\mathcal{M}$ if every equation in $\mathcal{T}$ satisfies $\mathcal{M}$. In particular, the theory of the model $\mathcal{M}$, denoted by the set of all equations satisfying $\mathcal{M}$, is the maximum theory that satisfies $\mathcal{M}$.

The standard theory of PV, denoted by $\mathbb{M}$, is defined as the tuple $(\{0,1\}^*, \mathcal{I}^{\mathsf{c}}, \mathcal{I}^{\mathsf{f}})$, where $\mathcal{I}^{\mathsf{c}}$ maps the only constant symbol ε to $\varepsilon \in \{0,1\}^*$, and $\mathcal{I}^{\mathsf{f}}$ maps the PV function symbols according to their definitions. Formally, we define the mapping $\mathcal{I}^{\mathsf{f}}_k$ from order-k PV functions to functions over $\{0,1\}^*$ inductively over $k \in \mathbb{N}$, and define $\mathcal{I}^{\mathsf{f}} = \bigcup_{k \in \mathbb{N}} \mathcal{I}^{\mathsf{f}}_k$. The mapping $\mathcal{I}^{\mathsf{f}}_k$ is defined as:

$\mathbb{M}$ is used to distinguish it from $\mathbb{N}$, i.e., the standard interpretation as natural numbers rather than strings.

- For $k = 0$, the base functions as functions satisfying their definition axioms, see Equation (10) to (13). That is, $\circ$ is the function that concatenates two strings, $\#(x, y)$ concatenates the string x with itself for $|y|$ times, $\mathsf{TR}(x)$ removes the leftmost bit of x, and $\mathsf{ITR}(x, y)$ removes the $|y|$ leftmost bits of x.
- For $k \geq 1$ consider two cases:
  - If $f = f^{(0)}_t$ is introduced by the composition rule, where t is an order-$(i-1)$ term, f is interpreted as the function $t^{\mathcal{I}^{\mathsf{c}}, \mathcal{I}^{\mathsf{f}}_{k-1}}$, i.e., the interpretation of the term t based on $\mathcal{I}^{\mathsf{f}}_{k-1}$ as given in the induction hypothesis.
  - If $f = f^{(1)}_{\Pi}$ is introduced by the recursion rule, where Π is the tuple $(g, h_0, h_1, k_0, k_1, \pi_0, \pi_1)$, we define the interpretation of f as the unique function satisfying the definition axioms Equation (15) and (16) with g, h_0, h_1 replaced by their interpretation in $\mathcal{I}^{\mathsf{f}}_{k-1}$.

PV functions in the standard model. Now we show that PV functions are polynomial-time computable on the standard model, as required by the postulate of feasible functions (see Postulate 1).

Lemma 2.5. *The following two properties hold.*

- *For every* PV *function f, its standard interpretation $f^{\mathbb{M}} \in \mathcal{F}'$.*
- PV *satisfies its standard model.*

Proof Sketch. We prove by simultaneous induction on k that (1) every k-order PV function is interpreted as a polynomial-time computable function by $\mathcal{I}^{\mathsf{f}}_k$ and (2) every order-k provable equation $e : s = t$ satisfies that $s^{\mathcal{I}^{\mathsf{c}}_k, \mathcal{I}^{\mathsf{f}}_k}$ and $t^{\mathcal{I}^{\mathsf{c}}_k, \mathcal{I}^{\mathsf{f}}_k}$ are the same function. The base case for $k = 0$ is obvious. For $k \geq 1$, we can see that:

- For any function $f = f^{(0)}_t$ introduced by the composition rule, we know by the induction hypothesis that all order-$(k-1)$ functions are interpreted as functions in $\mathcal{F}'$ (by $\mathcal{I}^{\mathsf{f}}_k$), and thus all order-$(k-1)$ terms are interpreted as functions in $\mathcal{F}'$. It follows by the definition of $\mathcal{I}^{\mathsf{f}}_k$ that $f^{\mathcal{I}^{\mathsf{f}}_k} \in \mathcal{F}'$.

- For any function $f = f^{(1)}_{\Pi}$ introduced by the recursion rule, where the tuple $\Pi = (g, h_0, h_1, k_0, k_1, \pi_0, \pi_1)$, π_0, π_1 are order-$(k-1)$ proofs of Equation (14), and $g, h_0, h_1, k_0, k_1 \in \mathcal{F}'$. By the definition of Equation (14), its standard interpretation, and the induction hypothesis (2), we know that over the standard model $\mathbb{M}$,
$$|h^{\mathbb{M}}_i(\vec{x}, y, z)| \leq |z \circ k^{\mathbb{M}}_i(\vec{x}, y)| \quad i \in \{0, 1\}$$
and thus $f^{\mathbb{M}} \in \mathcal{F}'$.

- For every order-k equation $e : s = t$ that admits an order-k proof π, by induction over the length of π and a case study on the deduction rules, we can show that $s^{\mathbb{M}} = t^{\mathbb{M}}$.

This completes the proof. □

By Lemma 2.2 and Theorem 2.1, we know that every (finite-order) PV function is polynomial-time computable. On the other hand, we will prove later that by a white-box inspection of the proof of Theorem 2.1 that:

Theorem 2.6 ([Coo75])**.** *For every function $F \in \mathsf{FP}$, there is a (finite-order) function symbol f such that $f^{\mathbb{M}} = F$.*

This indicates that the formalize of PV is not syntactically too weak in the sense that it defines every polynomial-time computable function.

We note, however, that there might be two PV functions f_1, f_2 that are functionally equivalent in the standard model, but $f_1 = f_2$ is not PV provable — because the functional equivalence is not provable by feasible mathematics. This is expected, as PV (and feasible mathematics in general) only allows to use the definition axioms and the structural induction over their constructions, while the functional equivalence of two feasibly constructible functions may only be provable using more advanced mathematics. Here is an example:

Also note that the combination of Lemma 2.5 and Theorem 2.6 proves that $\mathcal{F}' \subseteq \mathsf{FP} = \mathcal{F}$, and thus by Lemma 2.2 $\mathcal{F}' = \mathcal{F} = \mathsf{FP}$.

Example 2.2. Let Prime be the function that $\mathsf{Prime}(x) = [x \text{ is a prime number}]$. By the celebrated AKS algorithm [AKS04] we know that $\mathsf{Prime} \in \mathsf{FP}$, and thus by Theorem 2.6 there is a function symbol f such that $f^{\mathbb{M}} = \mathsf{Prime}$. Then:

- Let AKS be the Turing machine implementing the AKS algorithm running in time cn^c, $\mathcal{U}(M, x, T)$ be the PV function that simulates the Turing machine $M(x)$ for $|T|$ steps. The equation $\mathcal{U}(\mathsf{AKS}, x, 1^{c|x|^c}) = f(x)$ may not be provable in PV, although it is a true equation in $\mathbb{M}$.
- Let $\mathsf{Div}, \mathsf{Le}, \mathsf{Imp}, \mathsf{And}, \mathsf{Not}$ be the PV functions: $\mathsf{Div}(x, y) := [x \mid y]$, $\mathsf{Le}(x, y) := [x < y]$, $\mathsf{Imp}(x, y) := [x \leq y]$, $\mathsf{And}(x, y) := x \times y$, $\mathsf{Not}(x) = 1 - x$. The equation

  $$\mathsf{Imp}(f(x), \mathsf{Not}(\mathsf{And}(\mathsf{And}(\mathsf{Div}(y, x), \mathsf{Le}(1, y)), \mathsf{Le}(y, x)))) = 1$$

  which formalizes the statement "$f(x) = 1$ implies y is not a non-trivial divisor of x" is a true statement, but may not be provable in PV.

$[p]$ is the function that returns 0/1 if the proposition p is false/true.