3 Programming in PV

Although the formal definition of PV has been quite tedious, the "programming functionality" or "instruction set" provided by PV is still quite week. For instance, it is not immediately clear how Theorem 2.6 can be proved. Moreover, it is not clear whether PV can write classical algorithms, such as quick sort, in a straightforward way and proves their correctness.

The purpose of this section and subsequent section is to build a "computer" or "programming language" with *feasibly provable functionality*: It would be somewhat familiar to you if you have ever tried to design a CPU or programming language (especially when you have experience in writing formal proofs, such as Coq or Lean), except that our "underlying platform" is neither (say) LLVM nor (say) RISC-5, but Cook's theory PV (i.e. modified Cobham's class $\mathcal{F}'$). For an analogy in mathematics, the purpose of this section is similar to the first several lectures of an axiomatic set theory course that shows how to encode standard mathematical objects such as natural numbers in ZF or ZFC.

Glad to see you again, *real* feasible mathematicians.

3.1 Control and Logic in PV

In this subsection, we will define necessary tools and prove meta-theorems about control and (propositional) logic PV.

3.1.1 If-Then-Else Function

We first show how to implement the *if-then-else* function $\mathsf{ITE}(y,u,v)$ discussed in Example 2.1. Concretely, we will show that it is possible to define a PV-function $\mathsf{ITE}(y,u,v)$ such that the followings are provable in PV:

$$\mathsf{ITE}(s_0(x),u,v)=v,\quad \mathsf{ITE}(s_1(x),u,v)=u \tag{19}$$

We will encode `False` with 0 (i.e. $s_0(\varepsilon)$) and `True` with 1 (i.e. $s_1(\varepsilon)$). Alternatively, one may encode them using (say) ε and 0, but I guess $\{0,1\}$ is the most natural choice.

Indeed, we define ITE to be slightly more general: it will choose to output u or v based on the rightmost bit of y.

Similar to Example 2.1, we will use the limited recursion rule. Let $g(u,v):=\varepsilon$, $h_0(u,v,y,z):=v$, $h_1(u,v,y,z):=u$, $k_0(u,v,y):=v$, and $k_1(u,v,y)=u$. These three functions are of order-1 introduced by the rule of composition. We need to verify that for $i\in\{0,1\}$, there is a PV proof π_i for

$$\mathsf{ITR}(h_i(u,v,y,z),z\circ k_i(u,v,y))=0, \tag{20}$$

define $f^{(1)}_{\Pi}(u,v,y)$ from $\Pi:=(g,h_0,h_1,k_0,k_1,\pi_0,\pi_1)$, and then it is easy to see that Equation (19) follows if we define $\mathsf{ITE}(y,u,v)=f^{(1)}_{\Pi}(u,v,y)$.

It remains to prove Equation (20). Indeed, both of the cases for $i\in\{0,1\}$ can be derived from the following equation using the definition axioms of h_i, k_i and the substitution rules:

Proposition 3.1. $\mathsf{PV}\vdash\mathsf{ITR}(x,z\circ x)=\varepsilon$.

Informally, the proof will be done by an induction on x. Let $f_1(z,x):=\mathsf{ITR}(x,z\circ x)$ and $f_2(z,x)=\varepsilon$. We first consider the base case x/ε. It is easy to see that $\mathsf{PV}\vdash f_2(z,\varepsilon)=\varepsilon$ by (L4) substitution, and therefore we only need to prove:

Proposition 3.2. $\mathsf{PV} \vdash f_1(z, \varepsilon) = \varepsilon$.

Proof. We need to prove in PV that $\mathsf{ITR}(\varepsilon, z \circ \varepsilon) = \varepsilon$. Notice that $z \circ \varepsilon = z$, and thus by applying (L3) on the function $w \mapsto \mathsf{ITR}(\varepsilon, w \circ \varepsilon)$ we know that $\mathsf{ITR}(\varepsilon, z \circ \varepsilon) = \mathsf{ITR}(\varepsilon, z)$, and thus (by transitivity) we only need to prove in PV that

$$\mathsf{ITR}(\varepsilon, x) = \varepsilon. \tag{21}$$

It can be proved using the rule of structural induction. Concretely, we will define functions $f_3(x) = \mathsf{ITR}(\varepsilon, x)$ and $f_4(x) = \varepsilon$, and prove that both of them can be constructed by recursion from $g = \varepsilon$, $h_i(x, z) = \mathsf{TR}(z)$. The details are omitted and left as an exercise. □

Now we consider and case that the variable x is of form $s_i(x)$. Note that $\mathsf{PV} \vdash f_1(z, s_i(x)) = \mathsf{ITR}(s_i(x), z \circ s_i(x))$ by (L4) substitution, and we need to see how $f_1(z, s_i(x))$ can be constructed as a function of z, x, and $f_1(z, x)$. Notice that by the definition axiom of ITR and (L4) substitution we know that

$$\mathsf{PV} \vdash \mathsf{ITR}(s_i(x), z \circ s_i(x)) = \mathsf{TR}(\mathsf{ITR}(s_i(x), z \circ x)).$$

By thinking about its interpretation in the standard model $\mathbb{M}$, it is easy to see that

$$\mathsf{TR}(\mathsf{ITR}(s_i(x), z \circ x)) = \mathsf{ITR}(x, z \circ x) \tag{22}$$

and the RHS of the equation is indeed $f_1(z, x)$ — completing our goal of constructing $f_1(z, s_i(x))$ as a function of z, x and $f_1(z, x)$. Therefore it suffices to prove that:

Proposition 3.3. $\mathsf{PV} \vdash \mathsf{TR}(\mathsf{ITR}(s_i(x), z \circ x)) = \mathsf{ITR}(x, z \circ x)$.

To prove this, we need the fact that ITR and TR commutes:

Proposition 3.4. $\mathsf{PV} \vdash \mathsf{TR}(\mathsf{ITR}(x, y)) = \mathsf{ITR}(\mathsf{TR}(x), y)$.

Proof Sketch. We will prove by an induction on y that both sides of the equation are identical to the feasibly constructive function recursively defined from $g'(x) = \mathsf{TR}(x)$ and $h'_i(x, y, z) = \mathsf{TR}(z)$.

To see this for the LHS, notice that $\mathsf{PV} \vdash \mathsf{TR}(\mathsf{ITR}(x, \varepsilon)) = \mathsf{TR}(x)$ by unfolding ITR, and that for $i \in \{0, 1\}$,

$$\mathsf{PV} \vdash \mathsf{TR}(\mathsf{ITR}(x, s_i(y))) = \mathsf{TR}(\mathsf{TR}(\mathsf{ITR}(x, y))).$$

by unfolding ITR.

To see this for the RHS, notice that $\mathsf{ITR}(\mathsf{TR}(x), \varepsilon) = \mathsf{TR}(x)$ by unfolding ITR, and that for $i \in \{0, 1\}$,

$$\mathsf{PV} \vdash \mathsf{ITR}(\mathsf{TR}(x), s_i(y)) = \mathsf{TR}(\mathsf{ITR}(\mathsf{TR}(x), y))$$

by unfolding ITR. Therefore, we can prove the original equation by the structural induction rule in PV using $g'(x) = \mathsf{TR}(x)$ and $h'_i(x, y, z) = \mathsf{TR}(z)$. □

Proof of Proposition 3.3. By Proposition 3.4 with the substitutions $x/s_i(x)$ and $y/z \circ x$, we know that the LHS of the equation is equal to $\mathsf{ITR}(\mathsf{TR}(s_i(x)), z \circ x)$. Note that $\mathsf{PV} \vdash \mathsf{TR}(s_i(x)) = x$ by the definition equation of TR, therefore we can unfold TR and further prove that $\mathsf{ITR}(\mathsf{TR}(s_i(x)), z \circ x) = \mathsf{ITR}(x, z \circ x)$, which is exactly the RHS of the equation. □

Therefore, by applying Proposition 3.3, we notice that for $i \in \{0,1\}$,

$$\mathsf{PV} \vdash f_1(z, s_i(x)) = h_i(z, x, f_1(z,x))$$

for $h_i(z,x,z') := z'$. Also, notice that for $i \in \{0,1\}$

$$\mathsf{PV} \vdash f_2(z, s_i(x)) = h_i(z, x, f_2(z,x))$$

as both sides of the equation directly evaluates to ε. Therefore, by the induction rule, we can prove that $f_1(z,x) = f_2(z,x)$, which leads to the proof of Proposition 3.1.

Finally, I gave up the plan of always using z as the last variable while writing the function h_i; but z' is not that bad, I guess.

Finally, one can easily verify Equation (19) are provable in PV by the definition axioms

$$f_{\Pi}^{(1)}(u,v,s_i(y)) = h_i(u,v,y,f_{\Pi}^{(1)}(u,v,y)) \quad (i \in \{0,1\})$$

of $f_{\Pi}^{(1)}$ and the definition axiom $\mathsf{ITE}(y,u,v) = f_{\Pi}^{(1)}(u,v,y)$ of ITE. Therefore, we will safely refer ITE as the function defined here and use Equation (19) while writing PV proofs.

We note that an additional benefit of our definition is that PV proves

$$\mathsf{ITE}(\varepsilon, u, v) = \varepsilon. \tag{23}$$

We can treat ε as "error" or "undefined", which will be useful at some point.

The function LastBit. As an immediate application, we can simply define the function $\mathsf{LastBit}(x)$ as $\mathsf{LastBit}(x) = \mathsf{ITE}(x,1,0)$. It can be easily proved from the PV proofs of Equation (19) that

$$\mathsf{PV} \vdash \mathsf{LastBit}(\varepsilon) = \varepsilon; \quad \mathsf{PV} \vdash \mathsf{LastBit}(s_i(x)) = i \tag{24}$$

for $i \in \{0,1\}$.

IsEps and IsNotEps. Similar to ITE, we can define functions that determine whether a string is ε or not. Let $\mathsf{IsEps}(x)$ and $\mathsf{IsNotEps}(x)$ be defined as follows:

$$\mathsf{IsEps}(\varepsilon) = 1; \quad \mathsf{IsEps}(s_i(x)) = 0 \quad (i \in \{0,1\}) \tag{25}$$

$$\mathsf{IsNotEps}(\varepsilon) = 0; \quad \mathsf{IsNotEps}(s_i(x)) = 1 \quad (i \in \{0,1\}) \tag{26}$$

Notice that: $\mathsf{IsEps}(x)$ can be defined using limited recursion from $g = 1$, $h_i(x,z) = 0$, and $k_i(x) = 0$; and $\mathsf{IsNotEps}(x)$ can be defined from $g = 0$, $h_i(x,z) = 1$, and $k_i(x) = 1$. The equations $\mathsf{ITR}(h_i(x,z), z \circ k_i(x)) = 0$ can be proved in PV using Proposition 3.1 by unfolding k_i, h_i and applying (L4) substitution to the variable x in Proposition 3.1.

3.1.2 Proof by Case Study

We now introduce a meta-theorem that implements the idea of proof by case study. Let $s(\vec{x},y) = t(\vec{x},y)$ be an equation. A standard way to prove the equation is to consider whether the last bit of y is 0, 1, or $y = \varepsilon$. Formally:

Theorem 3.5. *Let $s(\vec{x},y) = t(\vec{x},y)$ be an equation. If*

- $\mathsf{PV} \vdash s(\vec{x},\varepsilon) = t(\vec{x},\varepsilon)$;

- $\mathsf{PV} \vdash s(\vec{x}, s_i(y)) = t(\vec{x}, s_i(y))$ *for* $i \in \{0,1\}$*;*

then $\mathsf{PV} \vdash s(\vec{x}, y) = t(\vec{x}, y)$.

Proof. We define two functions $f_s(\vec{x}, y) = s(\vec{x}, y)$ and $f_t(\vec{x}, y) = t(\vec{x}, y)$ by the rule of composition. We will prove that both f_s and f_t are identical to the function inductively defined by some $g(\vec{x}), h_0(\vec{x}, y, z), h_1(\vec{x}, y, z)$. Indeed, we will simply choose

$$\begin{aligned} g(\vec{x}) &:= f_s(\vec{x}, \varepsilon) \\ h_i(\vec{x}, y, z) &:= f_s(\vec{x}, s_i(y)) \quad (i \in \{0,1\}); \end{aligned}$$

that is, we define g, h_0, h_1 based on the LHS of the equation to prove. We need to prove that:

$$\begin{aligned} &\mathsf{PV} \vdash f_t(\vec{x}, \varepsilon) = g(\vec{x}) \\ &\mathsf{PV} \vdash f_t(\vec{x}, s_i(y)) = h_i(\vec{x}, y, z) \quad (i \in \{0,1\}). \end{aligned}$$

Indeed, by unfolding the definition of g and h_i and applying (L2) transitivity, these three equations are exactly the equations in the assumption. This completes the proof. □

Case on ITE. A typical application of the theorem is to perform case study on the condition in the if-then-else function, summarized as the following meta-theorem:

Theorem 3.6. *Let* $f_c(\vec{x}), f_0(\vec{x}), f_1(\vec{x}), g(\vec{x}, y), f_0'(\vec{x}), f_1'(\vec{x}), g'(\vec{x}, y)$ *are* PV *functions. Suppose that*

- $\mathsf{PV} \vdash g(\vec{x}, \varepsilon) = g'(\vec{x}, \varepsilon)$
- $\mathsf{PV} \vdash g(\vec{x}, f_i(\vec{x})) = g'(\vec{x}, f_i'(\vec{x}))$ *for* $i \in \{0,1\}$*;*

then $\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}(f_c(\vec{x}), f_1(\vec{x}), f_0(\vec{x}))) = g'(\vec{x}, \mathsf{ITE}(f_c(\vec{x}), f_1'(\vec{x}), f_2'(\vec{x})))$.

Proof Sketch. We can prove a stronger result that

$$\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}(z, f_1(\vec{x}), f_0(\vec{x}))) = g'(\vec{x}, \mathsf{ITE}(z, f_1'(\vec{x}), f_0'(\vec{x})))$$

and the theorem follows by performing (L4) substitution $z/f_c(\vec{x})$ to the equation. To prove this stronger result, we perform case study on the variable z using Theorem 3.5, and the three cases (after unfolding ITE) will be the three cases in the assumption. For instance, consider the case that $z = \varepsilon$ in Theorem 3.5, we need to prove that

$$\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}(\varepsilon, f_1(\vec{x}), f_0(\vec{x}))) = g'(\vec{x}, \mathsf{ITE}(\varepsilon, f_1'(\vec{x}), f_0'(\vec{x}))). \tag{27}$$

Notice that $\mathsf{PV} \vdash \mathsf{ITE}(\varepsilon, x, y) = \varepsilon$ by the definition axiom of ITE Equation (23). Then, by applying (L4) substitution $x/f_1(\vec{x})$ and subsequently $y/f_0(\vec{x})$, we have

This "unfolding" trick will be used multiple times; we will simply say "unfolding" later on.

$$\begin{aligned} &\mathsf{PV} \vdash \mathsf{ITE}(\varepsilon, f_1(\vec{x}), f_0(\vec{x})) = \varepsilon \\ &\mathsf{PV} \vdash \mathsf{ITE}(\varepsilon, f_1'(\vec{x}), f_0'(\vec{x})) = \varepsilon. \end{aligned}$$

We now apply (L3) to the equations above with the PV function $y \mapsto g(\vec{x}, y)$ to obtain

$$\begin{aligned} &\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}(\varepsilon, f_1(\vec{x}), f_0(\vec{x}))) = g(\vec{x}, \varepsilon) \\ &\mathsf{PV} \vdash g'(\vec{x}, \mathsf{ITE}(\varepsilon, f_1'(\vec{x}), f_0'(\vec{x}))) = g(\vec{x}, \varepsilon) \end{aligned}$$

Recall that $\mathsf{PV} \vdash g(\vec{x}, \varepsilon) = g'(\vec{x}, \varepsilon)$ holds by the assumption. We can therefore obtain Equation (27) by applying (L2) transitivity. The other cases can be proved accordingly using similar approach. □

Remark 3.1. We further note that the same proof technique leads to a more general result: We can perform case analysis even if there are multiple ITE's in g and g' with the same condition $f_c(\vec{x})$.

Suppose we have sequences of PV functions $\vec{f_1}(\vec{x}), \vec{f_0}(\vec{x}), \vec{f'_1}(\vec{x}), \vec{f'_0}(\vec{x})$ each of which has say $k \in \mathbb{N}$ functions, e.g., $\vec{f_1}(\vec{x}) = (f_1^{(1)}(\vec{x}), \ldots, f_1^{(k)}(\vec{x}))$, and both g and g' takes $\vec{x}$ and k additional variables. Suppose that we define $f_{-1}^{(j)}(\vec{x}), f_{-1}^{(j)}(\vec{x})$ as ε for simplicity of the notation. If PV proves for each $\vec{j} \in \{0, 1, -1\}^k$ that

$$g(\vec{x}, f_{j_1}^{(1)}(\vec{x}), \ldots, f_{j_k}^{(k)}(\vec{x})) = g'(\vec{x}, f'^{(1)}_{j_1}(\vec{x}), \ldots, f'^{(k)}_{j_k}(\vec{x}))),$$

Then PV also proves that

$$\begin{aligned} &g(\vec{x}, \mathsf{ITE}(f_c(\vec{x}), f_1^{(1)}(\vec{x}), f_0^{(1)}(\vec{x})), \ldots, \mathsf{ITE}(f_c(\vec{x}), f_1^{(k)}(\vec{x}), f_0^{(k)}(\vec{x}))) \\ = &\, g'(\vec{x}, \mathsf{ITE}(f_c(\vec{x}), f'^{(1)}_1(\vec{x}), f'^{(1)}_0(\vec{x})), \ldots, \mathsf{ITE}(f_c(\vec{x}), f'^{(k)}_1(\vec{x}), f'^{(k)}_0(\vec{x}))), \end{aligned}$$

This also holds for the case study on "dirty" ITE that will be discussed below.

Case on "dirty" ITE. We will occasionally need an alternative version of Theorem 3.6: When it can be feasibly proved that $f_c(\vec{x})$ is not ε, or not of form $s_i(y)$ for some $i \in \{0, 1\}$, we will be able to remove one of the three assumptions of Theorem 3.6. To formalize this method, we will define a "dirty" ITE, denoted by $\mathsf{ITE}_j^{(\varepsilon)}(x, u_1, u_0)$ as follows:

Why do I know this? There is no magic: I stuck at Section 3.2.1 and found this necessary.

$$\mathsf{ITE}_j^{(\varepsilon)}(\varepsilon, u_1, u_0) = u_j \quad \mathsf{ITE}^{(\varepsilon)}(s_i(x), u_1, u_0) = u_i \quad (i \in \{0, 1\}). \tag{28}$$

The formal definition of the function is similar to the definition of ITE and is left as an exercise. Informally, this function is called "dirty" ITE as it does not "correctly" deal with the case for $x = \varepsilon$. Then we have

Theorem 3.7. *Let $f_c(\vec{x}), f_0(\vec{x}), f_1(\vec{x}), g(\vec{x}, y), f'_0(\vec{x}), f'_1(\vec{x}), g'(\vec{x}, y)$ are PV functions. Suppose that*

- $\mathsf{PV} \vdash g(\vec{x}, f_i(\vec{x})) = g'(\vec{x}, f'_i(\vec{x}))$ *for* $i \in \{0, 1\}$;

then $\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}_j^{(\varepsilon)}(f_c(\vec{x}), f_1(\vec{x}), f_0(\vec{x}))) = \mathsf{PV} \vdash g'(\vec{x}, \mathsf{ITE}_j^{(\varepsilon)}(f_c(\vec{x}), f'_1(\vec{x}), f'_0(\vec{x})))$ *for* $j \in \{0, 1\}$.

The proof is similar to Theorem 3.6 and is left as an exercise. As an corollary:

Corollary 3.8. *Let $f_c(\vec{x}), f_0(\vec{x}), f_1(\vec{x}), g(\vec{x}, y), f'_0(\vec{x}), f'_1(\vec{x}), g'(\vec{x}, y)$ are PV functions, $j \in \{0, 1\}$. Suppose that*

- $\mathsf{PV} \vdash \mathsf{ITE}_j^{(\varepsilon)}(f_c(\vec{x}), u_1, u_0) = \mathsf{ITE}(f_c(\vec{x}), u_1, u_0)$;
- $\mathsf{PV} \vdash g(\vec{x}, f_i(\vec{x})) = g'(\vec{x}, f'_i(\vec{x}))$ *for* $i \in \{0, 1\}$;

then $\mathsf{PV} \vdash g(\vec{x}, \mathsf{ITE}_j^{(\varepsilon)}(f_c(\vec{x}), f_1(\vec{x}), f_0(\vec{x}))) = g'(\vec{x}, \mathsf{ITE}_j^{(\varepsilon)}(f_c(\vec{x}, f'_1(\vec{x}), f'_0(\vec{x}))))$.

Remark 3.2. Similarly, we can define "dirty" ITE that does not correctly deal with the case where x is of form $s_i(y)$ for some $i \in \{0, 1\}$, and prove a meta-theorem similar to Theorem 3.7.

3.1.3 Propositional Logic: And, Or, Not

An important application of the ITE function is to simulate propositional logic, i.e., implementing the connectives And, Or, Not. They are naturally defined as

$$\mathsf{And}(x,y) := \mathsf{ITE}(x, \mathsf{ITE}(y,1,0), \mathsf{ITE}(y,0,0)) \tag{29}$$
$$\mathsf{Or}(x,y) := \mathsf{ITE}(x, \mathsf{ITE}(y,1,1), \mathsf{ITE}(y,1,0)) \tag{30}$$
$$\mathsf{Not}(x) := \mathsf{ITE}(x,0,1) \tag{31}$$

by compositions of ITE.

One may ask why we write $\mathsf{ITE}(y,0,0)$ instead of 0 in the definition equation of And, and $\mathsf{ITE}(y,1,1)$ instead of 1 in the definition equation of Or. This is, again, a trick in system design. It prevents *short-circuit evaluation*, i.e., it ensures that if one of the input is ε, the output of the function is always ε. Therefore, we can use ε to encode "undefined" or "error" when it is useful.

Therefore, we can embed an arbitrary propositional formula with constants and variables into a term in PV. We will show a powerful meta-theorem showing that PV is sound and complete as a propositional proof system (PPS for short) using this embedding.

Theorem 3.9 (PV as a PPS)**.** *Let φ_1, φ_2 be propositional formulas consisting of variables and constants (i.e. `True` or `False`), and $p_{\varphi_1}, p_{\varphi_2}$ be the* PV*-terms by replacing propositional connectives with* {And, Or, Not} *and replacing `True` and `False` with $s_1(t)$ and $s_0(t')$ for arbitrary terms t and t' (not necessarily the same for all replacements). Assume that there is no variable that appears in exactly one of φ_1 and φ_2. Then: $\varphi_1 \equiv \varphi_2$ if and only if $\mathsf{PV} \vdash \mathsf{LastBit}(p_{\varphi_1}) = \mathsf{LastBit}(p_{\varphi_2})$.*

The proof is not feasibly constructive as it involves the standard model $\mathbb{M}$.

Proof. One side is easy: If $p_{\varphi_1} = p_{\varphi_2}$ admits a PV proof, the equation holds in the standard model $\mathbb{M}$. By the interpretation of ITE in $\mathbb{M}$, we can see that it immediately implies that for all assignments to variables $\vec{x} \in \{\mathtt{True}, \mathtt{False}\}^*$, $\varphi_1(\vec{x}) = \varphi_2(\vec{x})$, which implies that $\varphi_1 \equiv \varphi_2$. (In the rest of the proof, we identify 0 and `False`, as well as 1 and `True`.)

Now we prove the other side. Let $k \in \mathbb{N}$ be the number of variables involved in φ_1, φ_2. We prove that there is a PV-proof of $p_{\varphi_1} = p_{\varphi_2}$ for every φ_1, φ_2 such that $\varphi_1 \equiv \varphi_2$ by induction on k. (Note that this induction is not made within PV, but in our meta-theory.) The case for $k = 0$ is simple: By referring to the axioms of ITE (see Equation (19)) and using the logical rules for equation (as well as using the meta-theorem Theorem 3.6), we can prove in PV for some $b \in \{0,1\}$ that $\mathsf{LastBit}(p_{\varphi_1}) = b$ and $\mathsf{LastBit}(p_{\varphi_2}) = b$, i.e., we can evaluate both sides of the equation. Thus by (L1) symmetricity and (L2) transitivity, we can prove that $p_{\varphi_1} = p_{\varphi_2}$.

Now, assume that it is true for all φ_1, φ_2 consist of at most k variables, we want to prove the case for $k+1$. Let $\varphi_1'(\vec{x}, y)$ and $\varphi_2'(\vec{x}, y)$ be formulas consisting of at most $k+1$ variables and every variable in $\{\vec{x}, y\}$ appears in both φ_1' and φ_2', where $|\vec{x}| = k$. Assume that $\varphi_1'(\vec{x}, y) \equiv \varphi_2'(\vec{x}, y)$. Let $p_{\varphi_1'}(\vec{x}, y)$ and $p_{\varphi_2'}(\vec{x}, y)$ be the terms embedding φ_1' and φ_2' into PV as mentioned above, respectively. Notice that for $i \in \{0,1\}$, $j \in \{1,2\}$, and a fresh variable w, we will have that $p_{\varphi_j'}(\vec{x}, s_i(w))$ is a valid translation of the formula $\varphi_j'(\vec{x}, y/i)$ obtained by replacing occurrences of y in φ_j' to i as mentioned above, i.e., by

To make everything super formal, we can prove by induction on the formation of the formulas.

- replacing propositional connectives with {And, Or, Not},

- replacing `True` and `False` with $s_1(t)$ and $s_0(t')$ for arbitrary terms t, t'.

Notice that as $\varphi_1' \equiv \varphi_2'$, we know that $\varphi_1'(\vec{x}, y/i) \equiv \varphi_2'(\vec{x}, y/i)$ for $i \in \{0,1\}$. Therefore, by the induction hypothesis, there are PV-proofs of

$$\mathsf{LastBit}(p_{\varphi_1'}(\vec{x}, s_i(w))) = \mathsf{LastBit}(p_{\varphi_2'}(\vec{x}, s_i(w))) \tag{32}$$

for $i \in \{0,1\}$. Also, we can prove (by Equation (23) and Equation (24)) that

$$\mathsf{LastBit}(p_{\varphi_1'}(\vec{x}, \varepsilon)) = \mathsf{LastBit}(p_{\varphi_2'}(\vec{x}, \varepsilon)); \tag{33}$$

indeed, PV proves that both sides of the equation are equal to ε (recall that y appears in both φ_1' and φ_2'). We can then prove

$$\mathsf{LastBit}(p_{\varphi_1'}(\vec{x}, y)) = \mathsf{LastBit}(p_{\varphi_2'}(\vec{x}, y)) \tag{34}$$

by applying Theorem 3.5 (i.e. proof by case study) on y. □

Obviously, I'm being a bit more sloppy while writing proofs in meta-theory compared to writing proofs in PV. Hopefully there is no confusion for readers.

Remark 3.3. We note that the reason to put $\mathsf{LastBit}(\cdot)$ outside of both p_{φ_1} and p_{φ_2} is to deal with the case that p_{φ_i} is not wrapped by $\mathsf{And}, \mathsf{Or}, \mathsf{Not}$. The assumption thta there is no variable that appears in exactly one of φ_1 and φ_2 is necessary because it can be the case that the variable is ε in PV so that one side is ε and another side is 0 or 1.

3.1.4 Conditional Equations

Another application of the if-then-else function is to implement *conditional equations*. Let t_c be a term and t_1, t_2 be terms. (As we will not deal with variables in this subsection, we hide all $\vec{x}$ in terms.) The conditional equation $t_c \Rightarrow t_1 = t_2$ is defined as the PV equation

$$\mathsf{ITE}(t_c, t_2, t_1) = t_1.$$

We call t_c the *antecedent* of the conditional equation, while $t_1 = t_2$ is called the *consequence* of the conditional equation.

Intuitively, this equation is true if $\mathsf{LastBit}(t_c) = 0$ or $t_1 = t_2$ and is false if $\mathsf{LastBit}(t_c) = 1$ and $t_1 \neq t_2$. The case for $t_c = \varepsilon$ is considered a non-defined behavior.

It is not hard to see that:

Proposition 3.10 (Modus Ponens)**.** *If* PV *proves* $t_c \Rightarrow t_1 = t_2$ *and* $t_c = 1$*, then* $\mathsf{PV} \vdash t_1 = t_2$.

Proof. If $\mathsf{PV} \vdash t_c = 1$, we know that $\mathsf{ITE}(t_c, y, x) = x$ by unfolding ITE. Therefore, $\mathsf{PV} \vdash \mathsf{ITE}(t_c, t_2, t_1) = t_2$ by substitution, and thus $t_1 = t_2$ by transitivity (as $\mathsf{PV} \vdash \mathsf{ITE}(t_c, t_2, t_1) = t_1$ by the assumption). □

Proposition 3.11 (Explosion Rule)**.** PV *proves that* $s_0(z) \Rightarrow x = y$.

Proof. By unfolding the definition, we need to prove that $\mathsf{ITE}(s_0(z), y, x) = x$, where the LHS of the equation PV-provably evaluates to x. □

Conditioning is an important tool in PV, and we will see later on that with conditional equations we can make it possible for PV to encode *equations*, which is the key to develop useful meta-theorem in the next section. Here we provide a simple example on how conditioning could help:

Theorem 3.12. *Let t be a* PV*-term. Suppose that* PV *proves* $\mathsf{IsEps}(t) = 1$*, then* PV *also proves* $t = \varepsilon$*.*

Proof. We first prove that $\mathsf{IsEps}(x) \Rightarrow x = \varepsilon$. This can be proved by a simple case study using Theorem 3.5. Suppose that $x = \varepsilon$ the conditional equation PV-provably evaluates to $\varepsilon = \varepsilon$, which is true by (L0) reflexivity. Otherwise, $\mathsf{PV} \vdash \mathsf{IsEps}(s_i(x)) = 0$ and by the explosion rule, PV also proves the equation.

By (L3) substitution, we can then prove that $\mathsf{IsEps}(t) \Rightarrow t = \varepsilon$, and thus by Modus Ponens, we can conclude that $\mathsf{PV} \vdash t = \varepsilon$. □

3.2 Basic Data Structures in PV

In this subsection, we move on to design basic data structures for the programming language of feasible mathematicians. We will design *pairs* and *tuples* that are critical to implement algorithms in PV in a natural sense.

3.2.1 Pairs

We first show how to encode pairs in PV. To support making and unwinding pairs, we need to implement three PV functions: $\mathsf{MakePair}(x, y)$ intended to make a pair $\tau = (x, y)$, $\mathsf{Left}(\tau) = x$, and $\mathsf{Right}(\tau) = y$. We also need to ensure that the properties of pairs are PV-provable:

$$\mathsf{Left}(\mathsf{MakePair}(x, y)) = x, \quad \mathsf{Right}(\mathsf{MakePair}(x, y)) = y. \tag{35}$$

Description of encoding. A standard trick of encoding a pair (x, y) using a string is to encode y with the alphabet $\{00, 10\}$ and use 11 as a comma which separates x and y.

We define a couple of functions in turn. Let $\mathsf{PLen}(x)$ be the function that outputs 1 (i.e. $s_1(\varepsilon)$) if $|x|$ is odd and 0 (i.e. $s_0(\varepsilon)$) if $|x|$ is even. This can be easily defined by limited recursion as

$$\mathsf{PLen}(\varepsilon) := 0 \tag{36}$$
$$\mathsf{PLen}(s_i(x)) := \mathsf{Not}(\mathsf{PLen}(x)) \quad (i \in \{0, 1\}) \tag{37}$$

We define $\mathsf{PEnc}(x)$, $\mathsf{PDec}(y)$ be the encoding and decoding functions from the alphabet $\{0, 1\}$ to the alphabet $\{00, 10\}$, as follows:

Wrapping LastBit outside of the condition for ITE helps to apply Theorem 3.9.

$$\mathsf{PEnc}(\varepsilon) := \varepsilon \tag{38}$$
$$\mathsf{PEnc}(s_i(x)) := s_0(s_i(\mathsf{PEnc}(x))) \quad (i \in \{0, 1\}) \tag{39}$$
$$\mathsf{PDec}(\varepsilon) := \varepsilon \tag{40}$$
$$\mathsf{PDec}(s_i(x)) := \mathsf{ITE}(\mathsf{LastBit}(\mathsf{PLen}(x)), s_i(\mathsf{PDec}(x)), \mathsf{PDec}(x)) \tag{41}$$

To formally define these functions in PV, we need to apply the limited recursion rule using some g, h_0, h_1, k_0, k_1. We take PDec as an example:

$$g := \varepsilon$$
$$h_i(x, z) := \mathsf{ITE}(\mathsf{LastBit}(\mathsf{PLen}(x)), s_i(z), z) \quad (i \in \{0, 1\})$$
$$k_i(x) := s_i(\varepsilon) \quad (i \in \{0, 1\})$$

and we need to prove that:

Proposition 3.13. $\mathsf{PV} \vdash \mathsf{ITR}(h_i(x,z), z \circ k_i(x))$ *for* $i \in \{0,1\}$.

Proof Sketch. The idea is to perform case analysis on the condition $\mathsf{LastBit}(\mathsf{PLen}(x))$ using Theorem 3.6. It suffices to prove in PV that $\mathsf{ITR}(\varepsilon, z \circ \varepsilon) = 0$ and $\mathsf{ITR}(s_i(z), z \circ i) = 0$ for $i \in \{0,1\}$. The former equation follows from Proposition 3.1 by applying (L4) substitution z/ε, while the later equation (for $i \in \{0,1\}$) follows from $\mathsf{PV} \vdash \mathsf{ITR}(x,x) = 0$, which can be prove following the proof of Proposition 3.1. The details are omitted and left as exercise. □

It can be verified that:

Proposition 3.14. $\mathsf{PV} \vdash \mathsf{PLen}(\mathsf{PEnc}(x)) = 0$.

Proof Sketch. We prove by induction on x. More formally, let $f_1(x) := \mathsf{PLen}(\mathsf{PEnc}(x))$ and $f_2(x) = 0$, it can be verified that both $f_1(x)$ and $f_2(x)$ are identical to the function recursively defined by $g = 0$ and $h_i(x,z) := z$ ($i \in \{0,1\}$). □

Proposition 3.15. $\mathsf{PV} \vdash \mathsf{PDec}(\mathsf{PEnc}(x)) = x$.

Proof Sketch. We prove by induction on x. More formally, let $f_1(x) := \mathsf{PDec}(\mathsf{PEnc}(x))$ and $f_2(x) = x$, it can be verified that both $f_1(x)$ and $f_2(x)$ are identical to the function recursively defined by $g = \varepsilon$ and $h_i(x,z) = s_i(z)$. To see how to prove

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

notice that (by applying definition axioms) the LHS is PV provably equal to:

$$\begin{aligned}
&\mathsf{ITE}(\mathsf{LastBit}(\mathsf{PLen}(s_0(s_i(\mathsf{PEnc}(x))))), s_0(s_i(\mathsf{PDec}(\mathsf{PEnc}(x)))), \mathsf{PDec}(s_0(s_i((\mathsf{PEnc}(x)))))) \\
(\mathsf{PV} \vdash) =&\mathsf{ITE}(\mathsf{LastBit}(\mathsf{Not}(\mathsf{Not}(\mathsf{PLen}(\mathsf{PEnc}(x)))))), s_0(s_i(\mathsf{PDec}(\mathsf{PEnc}(x)))), \mathsf{PDec}(s_0(s_i(\mathsf{PEnc}(x))))) \\
(\mathsf{PV} \vdash) =&\mathsf{ITE}(\mathsf{LastBit}(\mathsf{PLen}(\mathsf{PEnc}(x))), s_0(s_i(f(x))), \mathsf{PDec}(s_0(s_i(\mathsf{PEnc}(x))))) \\
(\mathsf{PV} \vdash) =&\mathsf{ITE}(0, s_0(s_i(f(x))), \mathsf{PDec}(s_0(s_i(\mathsf{PEnc}(x))))) \qquad \text{(Proposition 3.14)} \\
(\mathsf{PV} \vdash) =&\mathsf{PDec}(s_0(s_i(\mathsf{PEnc}(x))))
\end{aligned}$$

Note that the second equation applied Theorem 3.9 with the propositional formula $\neg\neg x = x$. Again, after a sequence of tedious but straightforward unfolding, the last line is PV-provably equal to $s_i(f(x))$, which completes the proof. □

Description of pairing and unpacking functions. Then, we can simply define $\mathsf{MakePair}(x,y)$ as

$$\mathsf{MakePair}(x,y) := x \circ 11 \circ \mathsf{PEnc}(y). \tag{42}$$

Recall that $11 = s_1(s_1(\varepsilon))$

Then we define the functions $\mathsf{Left}(\tau)$ and $\mathsf{Right}(\tau)$. Notice that if we can define $\mathsf{Right}(\tau)$, $\mathsf{Left}(\tau)$ can be simply defined as

$$\mathsf{Left}(\tau) := \mathsf{TR}(\mathsf{TR}(\mathsf{ITR}(\tau, \mathsf{PEnc}(\mathsf{Right}(\tau))))). \tag{43}$$

Therefore, we now focus on defining $\mathsf{Right}(\tau)$. The idea is to read the encoding of y bit by bit until touching the comma "11", and then decode the string using PDec.

We first define a function $\mathsf{RightRaw}(\tau)$ that reads a suffix of τ before "11":

$$\mathsf{RightRaw}(\varepsilon) := \varepsilon \tag{44}$$

$$\mathsf{RightRaw}(s_0(\tau)) := s_0(\mathsf{RightRaw}(\tau)) \tag{45}$$

$$\mathsf{RightRaw}(s_1(\tau)) := \mathsf{ITE}(\tau, 1, s_1(\mathsf{RightRaw}(\tau))) \tag{46}$$

The formal definition of RightRaw in PV should be clear and left as an exercise. Note that there could be two cases: $\mathsf{RightRaw}(\mathsf{MakePair}(x, y))$ may be equal to $1 \circ \mathsf{PEnc}(y)$ if the leftmost bit of y is 0 or $y = \varepsilon$, and is equal to $\mathsf{PEnc}(y)$ otherwise. We need to define a function that removes the leftmost 1 when the string is of odd length.

Recall that $\mathsf{IsEps}(x)$ is the PV function that determines whether $x = \varepsilon$. Let RT and CleanLeft be defined as follows

$$\mathsf{RT}(\varepsilon) := \varepsilon \tag{47}$$
$$\mathsf{RT}(s_i(x)) := \mathsf{ITE}(\mathsf{IsEps}(x), \varepsilon, s_i(\mathsf{RT}(x))) \quad (i \in \{0, 1\}) \tag{48}$$
$$\mathsf{CleanLeft}(x) := \mathsf{ITE}(\mathsf{LastBit}(\mathsf{PLen}(x)), \mathsf{RT}(x), x) \tag{49}$$

then it can be verified that:

$$\mathsf{PV} \vdash \mathsf{CleanLeft}(\mathsf{PEnc}(y)) = \mathsf{PEnc}(y) \tag{50}$$
$$\mathsf{PV} \vdash \mathsf{CleanLeft}(1 \circ \mathsf{PEnc}(y)) = \mathsf{PEnc}(y) \tag{51}$$

(Recall that PV proves $\mathsf{PLen}(\mathsf{PEnc}(y)) = 0$ by Proposition 3.14, and $\mathsf{PLen}(1 \circ \mathsf{PEnc}(y)) = 1$ by essentially the same proof.)

Finally, we define:

$$\mathsf{Right}(\tau) := \mathsf{PDec}(\mathsf{CleanLeft}(\mathsf{RightRaw}(\tau))). \tag{52}$$

Proof of Correctness. We will sketch the proof that PV proves Equation (35):

Here we hide tons of unfolding; I may try to write a Lean/Coq proof at some point as these unfolding can be done by calling "simp", I think.

Lemma 3.16. PV *proves the following equations:*

- $\mathsf{Left}(\mathsf{MakePair}(x, y)) = x$;
- $\mathsf{Right}(\mathsf{MakePair}(x, y)) = y$.

Proof Sketch. (*Correctness of* Right). We start by proving the second equation. We first need to prove that:

$$\mathsf{RightRaw}(\mathsf{MakePair}(x, y)) = \mathsf{ITE}_0^{(\varepsilon)}(\mathsf{FirstBit}(y), \mathsf{PEnc}(y), 1 \circ \mathsf{PEnc}(y)) \tag{53}$$

where $\mathsf{FirstBit}(y) := \mathsf{ITR}(y, \mathsf{TR}(y))$, $\mathsf{ITE}_0^{(\varepsilon)}(c, x, y)$ outputs y if $c = \varepsilon$ or $\mathsf{LastBit}(c) = 1$, and outputs x if $\mathsf{LastBit}(c) = 0$. This equation formally describes the fact that

$$\mathsf{RightRaw}(\mathsf{MakePair}(x, y))$$

is $1 \circ \mathsf{PEnc}(y)$ if $y = \varepsilon$ or the leftmost bit of y is 0, and is $\mathsf{PEnc}(y)$ otherwise. Equation (53) can be proved by induction on y that both sides of the equation is identical to the function recursively defined by $g(x) = 1$ and

$$h_0(x, y, z) = \mathsf{ITE}(\mathsf{IsEps}(y), 100, s_0(s_0(z))) \tag{54}$$
$$h_1(x, y, z) = \mathsf{ITE}(\mathsf{IsEps}(y), 10, s_0(s_1(z))) \tag{55}$$

for $i \in \{0, 1\}$. Note that

- To prove that the LHS of Equation (53) is equal to the function recursively defined by g, h_0, h_1, notice that PV proves

$$\mathsf{RightRaw}(\mathsf{MakePair}(x, s_i(y))) = s_0(\mathsf{RightRaw}(s_i(\mathsf{MakePair}(x, y))))$$

by unfolding for $i \in \{0,1\}$. Then we prove by case study on y that

Here we hide tons of unfolding.

$$\mathsf{PV} \vdash s_0(\mathsf{RightRaw}(s_0(\mathsf{MakePair}(x,y)))) = \mathsf{ITE}(\mathsf{IsEps}(y), 100, s_0(s_0(z)))$$
$$\mathsf{PV} \vdash s_0(\mathsf{RightRaw}(s_1(\mathsf{MakePair}(x,y)))) = \mathsf{ITE}(\mathsf{IsEps}(y), 10, s_0(s_0(z)))$$

by Theorem 3.5. Notice that the RHS of the equations above are exactly $h_0(x,y,z)$ and $h_1(x,y,z)$.

- To prove that the RHS of Equation (53) is equal to the function recursively defined by g, h_0, h_1, the hard part is to prove that PV proves

$$\begin{aligned}&\mathsf{ITE}_0^{(\varepsilon)}(\mathsf{FirstBit}(s_i(y)), \mathsf{PEnc}(s_i(y)), 1 \circ \mathsf{PEnc}(s_i(y)))\\ =&h_i(x, y, \mathsf{ITE}_0^{(\varepsilon)}(\mathsf{FirstBit}(y), \mathsf{PEnc}(y), 1 \circ \mathsf{PEnc}(y)))\end{aligned}$$

for $i \in \{0,1\}$. We can prove this by case study on y using Theorem 3.5, the fact that $\mathsf{PV} \vdash \mathsf{FirstBit}(s_j(s_i(y))) = \mathsf{FirstBit}(s_i(y))$, and Theorem 3.7.

The details of the proofs above are omitted.

Now we move on to prove that $\mathsf{Right}(\mathsf{MakePair}(x,y)) = y$. Notice that PV proves the following sequence of calculation:

$$\begin{aligned}&\mathsf{Right}(\mathsf{MakePair}(x,y))\\ =\ &\mathsf{PDec}(\mathsf{CleanLeft}(\mathsf{RightRaw}(\mathsf{MakePair}(x,y)))) && \text{(Unfolding Right)}\\ =\ &\mathsf{PDec}(\mathsf{CleanLeft}(\mathsf{ITE}_0^{(\varepsilon)}(\mathsf{FirstBit}(y), \mathsf{PEnc}(y), 1 \circ \mathsf{PEnc}(y)))). && (56)\end{aligned}$$

The last equation follows from Equation (53).

To further simplify Equation (56), we will perform a case study on "dirty" ITE using Theorem 3.7. We need to prove that

Recall that I said Theorem 3.7 will be useful :)

$$\mathsf{PDec}(\mathsf{CleanLeft}(\mathsf{PEnc}(y))) = y \quad (57)$$
$$\mathsf{PDec}(\mathsf{CleanLeft}(1 \circ \mathsf{PEnc}(y))) = y \quad (58)$$

By Equation (50) and (51), both of them are implied by $\mathsf{PDec}(\mathsf{PEnc}(y)) = y$, which is given by Proposition 3.15.

(*Correctness of* Left). We will prove that $\mathsf{PV} \vdash \mathsf{Left}(\mathsf{MakePair}(x,y)) = x$. By the definition axiom of Left, it suffices to show that PV proves:

$$\mathsf{TR}(\mathsf{TR}(\mathsf{ITR}(\mathsf{MakePair}(x,y), \mathsf{PEnc}(\mathsf{Right}(\mathsf{MakePair}(x,y)))))) = x.$$

By the correctness of Right, we know that $\mathsf{Right}(\mathsf{MakePair}(x,y)) = y$ and thus we need to prove in PV that:

$$\mathsf{TR}(\mathsf{TR}(\mathsf{ITR}(x \circ 11 \circ \mathsf{PEnc}(y), \mathsf{PEnc}(y)))) = x.$$

Note that by an induction on y, it can be proved that:

Proposition 3.17. $\mathsf{PV} \vdash \mathsf{ITR}(x \circ y, y) = x$.

Thus, it suffices to prove in PV that $\mathsf{TR}(\mathsf{TR}(x \circ 11)) = x$, which can be proved by unfolding TR's. □

3.2.2 Tuples

Now we define tuples. This will (to some extent) relieve us from forgetting adding `\vec` to a sequence of variables: If we have tuples, it usually does not really matter whether we prove some meta-theorem for one or many variables, as one can encode a tuple using a single variable.

To support making and unwinding tuples, we need to implement the following functions for every $k \in \mathbb{N}$ and $i \in [k]$:

- $\mathsf{MakeTuple}^{(k)}(x_1, \ldots, x_k)$ intended to construct a tuple $\pi = (x_1, \ldots, x_k)$;
- $\mathsf{UnwindTuple}_i^{(k)}(\pi) = x_i$ takes the i-th element.

We use the standard construction: a $(k+1)$-tuple $(x_1, \ldots, x_k)$ is defined as a pair of x_1 and $(x_2, \ldots, x_k)$. Formally, we define by induction on k that for $k \geq 3$:

$$\mathsf{MakeTuple}^{(k)}(x_1, x_2, \ldots, x_k) = \mathsf{MakePair}(x_1, \mathsf{MakeTuple}^{(k-1)}(x_2, \ldots, x_k)) \tag{59}$$

$$\mathsf{UnwindTuple}_1^{(k)}(\pi) = \mathsf{Left}(\pi) \tag{60}$$

$$\mathsf{UnwindTuple}_i^{(k)}(\pi) = \mathsf{UnwindTuple}^{(k-1)}(\mathsf{Right}(\pi)) \quad (2 \leq i \leq k) \tag{61}$$

and for $k = 2$, $\mathsf{MakeTuple}^{(2)} = \mathsf{MakePair}$, $\mathsf{UnwindTuple}_1^{(k)} = \mathsf{Left}$, $\mathsf{UnwindTuple}_2^{(k)} = \mathsf{Right}$. We can prove that:

Lemma 3.18. *For every $k \geq 2$ and $i \in [k]$, PV proves that:*

$$\mathsf{UnwindTuple}_i^{(k)}(\mathsf{MakeTuple}^{(k)}(x_1, \ldots, x_k)) = x_i.$$

Proof Sketch. We prove by induction on k and applying Lemma 3.16. Notice that this induction is in meta-theory instead of PV. □

For simplicity, we will ignore the superscript (k) if there is no ambiguity. We will simply denote the tuple $\mathsf{MakeTuple}(x_1, \ldots, x_k)$ as $\pi = (x_1, \ldots, x_k)$, and the function $\mathsf{UnwindTuple}_i(\pi)$ as π_i.

3.3 Extensions on Recursion and Induction

In this section, we will propose a few extensions on the (limited) recursion rule for introducing PV functions, as well as the (structural) induction rule on the extensions of recursion.

3.3.1 Recursion on Multiple Variables

We first consider a version of recursion on multiple variables. As an motivating example, we will consider how to define a function $\mathsf{EQ}(x, y)$ that outputs 1 if and only if $x = y$, and output 0 otherwise. Intuitively we will define the function by considering the last bit of x and y:

$$\mathsf{EQ}(\varepsilon, \varepsilon) := 1 \tag{62}$$

$$\mathsf{EQ}(\varepsilon, s_i(y)) := 0 \quad (i \in \{0,1\}) \tag{63}$$

$$\mathsf{EQ}(s_i(x), \varepsilon) := 0 \quad (i \in \{0,1\}) \tag{64}$$

$$\mathsf{EQ}(s_i(x), s_j(y)) := \mathsf{EQ}(x, y) \quad (i, j \in \{0,1\}, i = j) \tag{65}$$

$$\mathsf{EQ}(s_i(x), s_j(y)) := 0 \quad (i, j \in \{0,1\}, i \neq j) \tag{66}$$

Recursion on multiple variables. Generalizing this example, we would like to have the following meta-theorem:

Theorem 3.19 ([Coo75]). *Let $g_{00}(\vec{x}), g_{01}(\vec{x}, y), g_{10}(\vec{x}, y)$ be* PV *functions, $h_\alpha(\vec{x}, y_1, y_2, z)$ and $k_\alpha(\vec{x}, y_1, y_2)$ for $\alpha \in \{0,1\}^2$ be* PV *functions. If*

$$\mathsf{PV} \vdash \mathsf{ITR}(h_\alpha(\vec{x}, y_1, y_2, z), z \circ k_\alpha(\vec{x}, y_1, y_2)) = 0$$

for every $\alpha \in \{0,1\}^2$, then there is a PV *function $f(\vec{x}, y_1, y_2)$ such that the following equations are provable in* PV*:*

- $f(\vec{x}, \varepsilon, \varepsilon) = g_{00}(\vec{x})$;
- $f(\vec{x}, \varepsilon, s_i(y)) = g_{01}(\vec{x}, s_i(y))$ *for* $i \in \{0,1\}$;
- $f(\vec{x}, s_i(y), \varepsilon) = g_{10}(\vec{x}, s_i(y))$ *for* $i \in \{0,1\}$;
- $f(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2)) = h_{i_1 i_2}(\vec{x}, y_1, y_2, f(\vec{x}, y_1, y_2))$ *for* $i_1, i_2 \in \{0,1\}$.

Here we write $s_i(y)$ instead of y because otherwise we will need to ensure that $g_{01}(\vec{x}, \varepsilon) = g_{10}(\vec{x}, \varepsilon)$.

Similarly, we can propose a further generalization of the theorem where we perform simultaneous recursion on more than two variables, which can be proved similar to this meta-theorem.

To prove Theorem 3.19, we need to implement recursion on two variables using the limited recursion rule in PV that only allows recursion on a single variable. The idea is to introduce two auxiliary variables w_1, w_2 and define a function $f'(\vec{x}, y_1, y_2, w_1, w_2)$ that is supposed to satisfy that

$$f'(\vec{x}, y_1, y_2, w_1, w_2) = f(\vec{x}, \mathsf{ITR}(y_1, \mathsf{ITR}(w_1, w_2)), \mathsf{ITR}(y_2, \mathsf{ITR}(w_1, w_2)))$$

and define f' by recursion on the variable w_2. Finally, we will define

$$f(\vec{x}, y_1, y_2) := f'(\vec{x}, y_1, y_2, y_1 \circ y_2, y_1 \circ y_2)$$

and verify that all equations in Theorem 3.19 can be proved in PV.

Proof Sketch of Theorem 3.19. We will define a function $f'(\vec{x}, y_1, y_2, w_1, w_2)$ as discussed above. That is, we will use the limited recursion rule on the auxiliary variable w_2 using the functions g, h_0, h_1, where $g(\vec{x}, y_1, y_2, w_1)$

$$= \begin{cases} \varepsilon & \mathsf{Not}(\mathsf{Or}(\mathsf{IsEps}(\mathsf{ITR}(y_1, w_1)), \mathsf{IsEps}(\mathsf{ITR}(y_2, w_1)))) \\ g_{00}(\vec{x}) & \mathsf{And}(\mathsf{IsEps}(\mathsf{ITR}(y_1, w_1)), \mathsf{IsEps}(\mathsf{ITR}(y_2, w_1))) \\ g_{01}(\vec{x}, \mathsf{ITR}(y_2, w_1)) & \mathsf{And}(\mathsf{IsEps}(\mathsf{ITR}(y_1, w_1)), \mathsf{Not}(\mathsf{IsEps}(\mathsf{ITR}(y_2, w_1)))) \\ g_{10}(\vec{x}, \mathsf{ITR}(y_1, w_1)) & \mathsf{And}(\mathsf{Not}(\mathsf{IsEps}(\mathsf{ITR}(y_1, w_1))), \mathsf{IsEps}(\mathsf{ITR}(y_2, w_1))) \end{cases} \tag{67}$$

and for $i \in \{0,1\}$, $h_i(\vec{x}, y_1, y_2, w_1, w_2, z)$

$$= \begin{cases} g(\vec{x}, \hat{y}_1, \hat{y}_2, \varepsilon) & \mathsf{Or}(\mathsf{IsEps}(\hat{y}_1), \mathsf{IsEps}(\hat{y}_2)) \\ h_{i_1 i_2}(\vec{x}, \mathsf{TR}(\hat{y}_1), \mathsf{TR}(\hat{y}_2), z) & (\mathsf{LastBit}(\hat{y}_1), \mathsf{LastBit}(\hat{y}_2)) = (i_1, i_2) \end{cases} \tag{68}$$

where $\hat{y}_j := \mathsf{ITR}(y_j, \mathsf{ITR}(w_1, s_i(w_2)))$ for $j \in \{1,2\}$. Note that the case study in defining g, h_0, and h_1 can be done by the if-then-else function. We will define the function $k_i(\vec{x}, y_1, y_2, w_1, w_2)$ as

$$= \begin{cases} g(\vec{x}, \hat{y}_1, \hat{y}_2, \varepsilon) & \mathsf{Or}(\mathsf{IsEps}(\hat{y}_1), \mathsf{IsEps}(\hat{y}_2)) \\ k_{i_1 i_2}(\vec{x}, \mathsf{TR}(\hat{y}_1), \mathsf{TR}(\hat{y}_2)) & (\mathsf{LastBit}(\hat{y}_1), \mathsf{LastBit}(\hat{y}_2)) = (i_1, i_2) \end{cases} \tag{69}$$

Inequality for limited recursion. We need to verify that (g, h_0, h_1, k_0, k_1) satisfy that

$$\mathsf{PV} \vdash \mathsf{ITR}(h_i(\vec{x}, y_1, y_2, w_1, w_2, z), k_i(\vec{x}, y_1, y_2, w_1, w_2)) = 0$$

for $i \in \{0, 1\}$ to use the limited recursion rule. This is implied by the assumption, the fact that $\mathsf{PV} \vdash \mathsf{ITR}(x, x) = 0$, and the case study technique on ITE (see Theorem 3.6 and Remark 3.1).

Correctness. As mentioned above, we will define

$$f(\vec{x}, y_1, y_2) := f'(\vec{x}, y_1, y_2, y_1 \circ y_2, y_1 \circ y_2),$$

and it remains to verify that the equations in the theorem statement are provable in PV. We sketch the proof of each equation:

- To see that $f(\vec{x}, \varepsilon, \varepsilon) = g_{00}(\vec{x})$, notice that $f(\vec{x}, \varepsilon, \varepsilon)$ is equal to $g(\vec{x}, \varepsilon, \varepsilon, \varepsilon)$. By the definition of g, we will obtain (after unfolding $\mathsf{ITR}, \mathsf{IsEps}$ and And) that $g(\vec{x}, \varepsilon, \varepsilon, \varepsilon) = g_{00}(\vec{x})$.
- To see that $f(\vec{x}, \varepsilon, s_j(y)) = g_{01}(\vec{x}, s_j(y))$, notice that $f(\vec{x}, \varepsilon, s_j(y))$ is equal to

  $$f'(\vec{x}, \varepsilon, s_j(y), \varepsilon \circ s_j(y), \varepsilon \circ s_j(y)) = f'(\vec{x}, \varepsilon, s_j(y), s_j(y), s_j(y)).$$

  Since $\hat{y}_1 = \mathsf{ITR}(\varepsilon, s_j(y), s_j(y)) = \varepsilon$, we know (by unfolding ITE in the definition of h_j) that this is PV-provably equal to

  $$g(\vec{x}, \varepsilon, \mathsf{ITR}(s_j(y), \mathsf{ITR}(s_j(y), s_j(y))), \varepsilon) = g(\vec{x}, \varepsilon, s_j(y), \varepsilon)$$

  (The equation follows from that $\mathsf{ITR}(x, x) = \varepsilon$ and $\mathsf{ITR}(y, \varepsilon) = y$ are provable in PV.) By unfolding g and ITE's (in the definition of g), we can prove in PV that $g(\vec{x}, \varepsilon, s_j(y), \varepsilon) = g_{01}(\vec{x}, s_j(y))$.
- The case for $f(\vec{x}, s_j(y), \varepsilon) = g_{10}(\vec{x}, s_j(y))$ is similar to the case above.
- To see that $f(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2)) = h_{i_1 i_2}(\vec{x}, y_1, y_2, f(\vec{x}, y_1, y_2))$ for $i_1, i_2 \in \{0, 1\}$, notice that PV proves

  $$\begin{aligned}
  & f(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2)) && (70)\\
  = {} & f'(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2), s_{i_1}(y_1) \circ s_{i_2}(y_2), s_{i_1}(y_1) \circ s_{i_2}(y_2)) && (71)\\
  = {} & f'(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2)) && (72)\\
  = {} & h_{i_2}(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_1}(y_1) \circ y_2, z) && (73)
  \end{aligned}$$

  where

  $$\begin{aligned}
  z := {} & f'(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_1}(y_1) \circ y_2) && (74)\\
  = {} & f'(\vec{x}, y_1, y_2, y_1 \circ y_2, y_1 \circ y_2) && (75)\\
  = {} & f(\vec{x}, y_1, y_2). && (76)
  \end{aligned}$$

  Here, Equation (75) is non-trivial, and will be deferred to the end of the proof. Also, notice that

  $$\begin{aligned}
  \hat{y}_1 = \mathsf{ITR}(s_{i_1}(y_1), \mathsf{ITR}(s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2))) = s_{i_1}(y_1) && (77)\\
  \hat{y}_2 = \mathsf{ITR}(s_{i_2}(y_2), \mathsf{ITR}(s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_2}(s_{i_1}(y_1) \circ y_2))) = s_{i_2}(y_2) && (78)
  \end{aligned}$$

  By unfolding h_{i_2} we can see that

  $$(73) = h_{i_1 i_2}(\vec{x}, y_1, y_2, z), \tag{79}$$

  which (together with Equation (76)) obtains the equation we need.

It remains to prove Equation (75). Indeed, we will prove a more general equation (which derives Equation (75) by z_1/y_1 and z_2/y_2).

Proposition 3.20. PV *proves the following equation*

$$f'(\vec{x}, s_{i_1}(z_1), s_{i_2}(z_2), s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_1}(y_1) \circ y_2) = f'(\vec{x}, z_1, z_2, y_1 \circ y_2, y_1 \circ y_2).$$

Proof Sketch. Let $f_1(\vec{x}, z_1, z_2, y_1, y_2)$ and $f_2(\vec{x}, z_1, z_2, y_1, y_2)$ be the LHS and RHS of the equation, respectively. Notice that when y_1 and y_2 are both substituted by ε, the equation is provable in PV by simply unfolding f'. We then prove that

$$f_1(\vec{x}, z_1, z_2, y_1, \varepsilon) = f_2(\vec{x}, z_1, z_2, y_1, \varepsilon). \tag{80}$$

To see this, notice that both sides of the equation are identical to the function recursively defined on y_1 using some PV functions g', h'_0, h'_1. Specifically, the base case is given by that $f_1(\vec{x}, z_1, z_2, \varepsilon, \varepsilon) = f_2(\vec{x}, z_1, z_2, \varepsilon, \varepsilon)$, and the induction step requires proving identities on ITR, such as $\mathsf{ITR}(z, \mathsf{ITR}(s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_1}(y_1) \circ y_2)) = \mathsf{TR}(z)$.

Hopefully you are convinced that all these identities are PV provable...

Subsequently, we prove that

$$f_1(\vec{x}, z_1, z_2, y_1, y_2) = f_2(\vec{x}, z_1, z_2, y_1, y_2)$$

by showing that both sides of the equation are identical to the function recursively defined on y_2 using some PV functions g'', h''_0, h''_1. This time, the base case is given by Equation (80), while the induction case also requires proving identities on ITR. □

This completes the proof. □

3.3.2 Induction on Multiple Variables

After defining the function EQ, we will need to prove some properties about it. For instance, we may need to prove that

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) = \mathsf{EQ}(y, x). \tag{81}$$

Since EQ is defined using Theorem 3.19 instead of the standard recursion rule in PV, we will need a structural induction rule for functions defined by recursion on multiple variables. Formally, we will need to prove that

Theorem 3.21. *Let $f_1(\vec{x}, y_1, y_2), f_2(\vec{x}, y_1, y_2)$ be two* PV *functions. Let $g_{00}(\vec{x})$, $g_{01}(\vec{x}, y)$, $g_{10}(\vec{x}, y)$, and $h_\alpha(\vec{x}, y_1, y_2, z)$, $\alpha \in \{0,1\}^2$ be* PV *functions. Suppose that for $j \in \{1, 2\}$,* PV *proves that*

- $f_j(\vec{x}, \varepsilon, \varepsilon) = g_{00}(\vec{x})$;
- $f_j(\vec{x}, \varepsilon, s_j(y)) = g_{01}(\vec{x}, s_j(y))$ *for* $j \in \{0, 1\}$;
- $f_j(\vec{x}, s_j(y), \varepsilon) = g_{10}(\vec{x}, s_j(y))$ *for* $j \in \{0, 1\}$;
- $f_j(\vec{x}, s_{i_1}(y_1), s_{i_2}(y_2)) = h_{i_1 i_2}(\vec{x}, y_1, y_2, f_j(\vec{x}, y_1, y_2))$ *for* $i_1, i_2 \in \{0, 1\}$.

Then $\mathsf{PV} \vdash f_1(\vec{x}, y_1, y_2) = f_2(\vec{x}, y_1, y_2)$.

To see that the theorem suffices to prove Equation (81), notice that for $f_1(x, y) := \mathsf{EQ}(x, y)$ and $f_2(x, y) = \mathsf{EQ}(y, x)$, both f_1 and f_2 PV-provably satisfy the precondition in Theorem 3.21 with $g_{00} = 1$, $g_{01} = 0$, $g_{10} = 0$, and

See Equation (62) to (66)

$$h_{ij}(x, y, z) = \begin{cases} z & i = j \\ 0 & i \neq j \end{cases},$$

therefore by Theorem 3.21 we know that $\mathsf{PV} \vdash f_1(x, y) = f_2(x, y)$.

Proof Sketch of Theorem 3.21. The proof idea of Theorem 3.21 closely follows the proof of Theorem 3.19. For each $j \in \{1, 2\}$, we define

$$f'_j(\vec{x}, y_1, y_2, w_1, w_2) := f_j(\vec{x}, \mathsf{ITR}(y_1, \mathsf{ITR}(w_1, w_2)), \mathsf{ITR}(y_2, \mathsf{ITR}(w_1, w_2))). \tag{82}$$

We will prove that

$$\mathsf{PV} \vdash f'_0(\vec{x}, y_1, y_2, y_1 \circ y_2, w_2) = f'_1(\vec{x}, y_1, y_2, y_1 \circ y_2, w_2) \tag{83}$$

and thus the theorem follows by substituting w_2 with $y_1 \circ y_2$.

To prove Equation (83), we will use the induction rule in PV on the variable w_2. Concretely, we define $g'(\vec{x}, y_1, y_2) = g_{00}(\vec{x})$ and for $i \in \{0, 1\}$, $h'_i(\vec{x}, y_1, y_2, w_2, z)$

$$= \begin{cases} g_{00}(\vec{x}) & \mathsf{And}(\mathsf{IsEps}(\mathsf{ITR}(y_1, \mathsf{ITR}(w_1, s_i(w_2)))), \\ & \quad \mathsf{IsEps}(\mathsf{ITR}(y_2, \mathsf{ITR}(w_1, s_i(w_2))))) = 1 \\ g_{01}(\vec{x}, \mathsf{ITR}(y_2, \mathsf{ITR}(w_1, s_i(w_2)))) & \mathsf{IsEps}(\mathsf{ITR}(y_1, \mathsf{ITR}(w_1, s_i(w_2)))) = 1 \\ g_{10}(\vec{x}, \mathsf{ITR}(y_1, \mathsf{ITR}(w_1, s_i(w_2)))) & \mathsf{IsEps}(\mathsf{ITR}(y_2, \mathsf{ITR}(w_1, s_i(w_2)))) = 1 \\ h_{i_1 i_2}(\vec{x}, \mathsf{TR}(\hat{y}_1), \mathsf{TR}(\hat{y}_2), z) & (\mathsf{LastBit}(\hat{y}_1), \mathsf{LastBit}(\hat{y}_2)) = (i_1, i_2) \end{cases}$$

where $w_1 := y_1 \circ y_2$, $\hat{y}_j := \mathsf{ITR}(y_j, w_1, s_i(w_2))$ for $j \in \{1, 2\}$. It can be verified (similar to the proof of Theorem 3.21) that both LHS and RHS of Equation (83) are identical to the function recursively defined by $g'(\vec{x}, y_1, y_2), h_0(\vec{x}, y_1, y_2, w_2, z), h_1(\vec{x}, y_1, y_2, w_2, z)$ inductively on the variable w_2. □

Remark 3.4. We note that although we only consider the case for two variables, the same proof idea generalizes to induction on k variables for all $k \in \mathbb{N}$.

3.3.3 Application: Basic Arithmetic

Another important application of recursion and induction on multiple variables (i.e. Theorem 3.19 and 3.21) is to implement basic arithmetic operations of natural numbers such as addition and multiplication.

Cheers! Feasible mathematicians finally start to study primary school mathematics.

Encoding of natural numbers. Recall that natural numbers are not "first-class citizens" in the theory of PV — only binary strings have native support in PV. We will need to specify an encoding of natural numbers with binary strings.

Following [Coo75], we will use *dyadic encoding* where the leftmost bit is the most significant bit. Formally, let $[x]_{\mathbb{N}}$ be the number encoded by the string x, we define

$$[\varepsilon]_{\mathbb{N}} := 0, \ [s_0(x)]_{\mathbb{N}} := 2x + 2, \ [s_1(x)]_{\mathbb{N}} := 2x + 1.$$

This encoding method ensures a bijection between natural numbers and binary strings so that we will no longer need to verify whether a string encodes a valid number. For simplicity, we will identify $s_2(x)$ with $s_0(x)$; we also denote $s_2(x)$ by $x2$ and $s_1(x)$ by $x1$.

A downside is that ε means both "error" and 0 :(

Addition. To define addition $\mathsf{Add}(x, y)$ (i.e. $[\mathsf{Add}(x, y)]_{\mathbb{N}} = [x]_{\mathbb{N}} + [y]_{\mathbb{N}}$), we perform the standard algorithm on the dyadic encoding using Theorem 3.19:

$$\mathsf{Add}(x, \varepsilon) = \mathsf{Add}(\varepsilon, x) := x \tag{84}$$
$$\mathsf{Add}(x2, y2) := s_2(\mathsf{Succ}(\mathsf{Add}(x, y))) \tag{85}$$
$$\mathsf{Add}(x2, y1) = \mathsf{Add}(x1, y2) := s_1(\mathsf{Succ}(\mathsf{Add}(x, y))) \tag{86}$$
$$\mathsf{Add}(x1, y1) := s_2(\mathsf{Add}(x, y)) \tag{87}$$

where $\mathsf{Succ}(x)$ is the successor function $x \mapsto x + 1$, whose definition is left as an exercise. Specifically, Add is defined from $g_{00} = \varepsilon, g_{01}(y) = g_{10}(y) = y$, $h_{00}(x, y, z) = s_2(\mathsf{Succ}(z))$, $h_{01}(x, y, z) = h_{10}(x, y, z) = s_1(\mathsf{Succ}(z))$, and $h_{11}(x, y, z) = s_2(z)$.

Note that to apply Theorem 3.19, we will also need to define functions k_α for $\alpha \in \{0, 1\}$ and proves in PV that

$$\mathsf{ITR}(h_\alpha(x, y, z), z \circ k_\alpha(x, y)) = 0.$$

Indeed, it is not hard to verify that it suffices to define $k_{00}(x, y) = k_{01}(x, y) = k_{10}(x, y) = k_{11}(x, y) = 11$. (To see this, we will need that $\mathsf{PV} \vdash \mathsf{ITR}(\mathsf{Succ}(x), s_0(x)) = 0$, which is easy if Succ is defined properly.)

Proposition 3.22. *The following equations are provable in* PV.

- $\mathsf{Add}(x, y) = \mathsf{Add}(y, x)$
- $\mathsf{Add}(x, 1) = \mathsf{Succ}(x)$
- $\mathsf{Add}(\mathsf{Add}(x, y), z) = \mathsf{Add}(x, \mathsf{Add}(y, z))$

The first equation can be proved similar to the proof of $\mathsf{EQ}(x, y) = \mathsf{EQ}(y, x)$. The second can be proved by unfolding Add using the definition axioms.

To prove the last equation, we will need to apply induction on three variables (see Theorem 3.21 and Remark 3.4). Specifically, let $g_{000} = 0$, $g_{001}(x, y) = g_{010}(x, y) = g_{100}(x, y) = \mathsf{Add}(x, y)$, $g_{011}(x) = g_{101}(x) = g_{110}(x) = x$, and some $h_\alpha(x, y, z, w)$ for $\alpha \in \{0, 1\}^3$, both LHS and RHS of the equation are PV-provably equal to the function recursively defined from these g_α and h_α. For instance, for $\alpha = 111$, $h_{111}(x, y, z, w) := s_1(\mathsf{Succ}(w))$ and we can prove in PV that

$$\mathsf{Add}(\mathsf{Add}(x1, y1), z1) = h_{111}(x, y, z, \mathsf{Add}(\mathsf{Add}(x, y), z))$$
$$\mathsf{Add}(x1, \mathsf{Add}(y1, z1)) = h_{111}(x, y, z, \mathsf{Add}(x, \mathsf{Add}(y, z))$$

by unfolding Add and h_{111}. Note that other basic properties of addition can also be proved following similar approach.

From now, we will slightly abuse the notation to write $\mathsf{Add}(x, y)$ as $x + y$.

Multiplication. Similarly, we can define multiplication $\mathsf{Mul}(x, y)$ of two numbers. Let $2 \cdot x := \mathsf{Add}(x, x)$ and $4 \cdot x := \mathsf{Add}(2 \cdot x, 2 \cdot x)$. We define:

$$\mathsf{Mul}(x, \varepsilon) = \mathsf{Mul}(\varepsilon, x) = \varepsilon \tag{88}$$
$$\mathsf{Mul}(x1, y1) := 4 \cdot \mathsf{Mul}(x, y) + 2 \cdot x + 2 \cdot y + 1 \tag{89}$$
$$\mathsf{Mul}(x2, y1) := 4 \cdot \mathsf{Mul}(x, y) + 2 \cdot x + 4 \cdot y + 2 \tag{90}$$
$$\mathsf{Mul}(x1, y2) := 4 \cdot \mathsf{Mul}(x, y) + 4 \cdot x + 2 \cdot y + 2 \tag{91}$$
$$\mathsf{Mul}(x2, y2) := 4 \cdot \mathsf{Mul}(x, y) + 4 \cdot x + 4 \cdot y + 4 \tag{92}$$

More formally, Mul is recursively defined using Theorem 3.19 from $g_{00} = g_{01}(x) = g_{10}(x) = \varepsilon$, $h_{11}(x,y,z) = 4\cdot z + 2\cdot x + 2\cdot y + 1$, $h_{01}(x,y,z) = 4\cdot z + 2\cdot x + 4\cdot y + 2$, $h_{10}(x,y,z) = 4\cdot z + 4\cdot x + 2\cdot y + 2$, and $h_{00}(x,y,z) = 4\cdot z + 4\cdot x + 4\cdot y + 4$. We will need to ensure that for every $\alpha \in \{0,1\}^2$, there is a function k_α such that

$$\mathsf{PV} \vdash \mathsf{ITR}(h_\alpha(x,y,z), z \circ k_\alpha(x,y)) = \varepsilon.$$

Indeed, we can define $k_\alpha(x,y) := 1111 \circ (x \circ 1111) \circ (y \circ 1111)$. To see that this suffices, we will need to prove that

Proposition 3.23. PV *proves the following equations:*

- $\mathsf{ITR}(\mathsf{Add}(x,y), x \circ y) = 0$
- $\mathsf{ITR}(2\cdot x, x \circ 1111) = 0$
- $\mathsf{ITR}(4\cdot x, x \circ 1111) = 0$

The first equation can be proved by induction on x and y using Theorem 3.21. The second and third equations can be proved by an induction on x using the induction rule of PV. From now, we will slightly abuse the notation to denote $\mathsf{Mul}(x,y)$ as $x\cdot y$ or xy.

Similar to the case for addition, basic properties of multiplication can be proved by induction on one or multiple variables. For instance:

Proposition 3.24. PV *proves the following equations:*

- $xy = yx$
- $(xy)z = x(yz)$
- $x(y+z) = xy + xz$

We sketch the proof of the third equation for completeness. We will prove the equation by induction on x, y, z (see Theorem 3.21 and Remark 3.4). The cases that at least one of x, y, z is ε are easy as we can simply unfold Mul and Add. It suffices to show that there are $h_\alpha(x,y,z,w)$ for $\alpha \in \{0,1\}^3$ such that both LHS and RHS of the equation satisfies the recursive relation specified by h_α. For instance, letting $\alpha = 111$, we will have

$$\begin{aligned} x1\cdot(y1 + z1) &= x1\cdot((y+z)2) \\ &= 4\cdot x(y+z) + 4\cdot x + 2\cdot(y+z) + 2; \\ x1\cdot y1 + x1\cdot z1 &= 4\cdot xy + 2\cdot x + 2\cdot y + 1 + 4\cdot xz + 2\cdot x + 2\cdot z + 1 \\ &= 4\cdot(xy+xz) + 4\cdot x + 2\cdot(y+z) + 2; \end{aligned}$$

where the last equation follows from the commutativity and associativity of addition. Therefore, it suffices to define $h_{111}(x,y,z,w) = 4\cdot z + 2\cdot x + 2\cdot(y+z) + 2$.

3.4 The Function EQ and Equality

An important application of the induction principle on multiple variables is to prove that the function EQ we defined before is indeed the characteristic function of the equality relation in PV. We will use the notion of conditional equation introduced in Section 3.1.4.

Lemma 3.25. $\mathsf{PV} \vdash \mathsf{EQ}(y_1, y_2) \Rightarrow y_1 = y_2$.

Proof. Unfolding the definition of conditional equations, we need to prove that $\mathsf{PV} \vdash \mathsf{ITE}(\mathsf{EQ}(y_1, y_2), y_2, y_1) = y_1$. We prove this by applying Theorem 3.21 on the variables y_1 and y_2. Let $f_1(x, y) = \mathsf{ITE}(\mathsf{EQ}(x, y), y, x)$ and $f_2(x, y) = x$. We define $g_{00} = 1$, $g_{10}(y) = y$, $g_{01}(y) = \varepsilon$, $h_{01}(y_1, y_2, z) = s_0(y_1)$, $h_{10}(y_1, y_2) = s_1(y_1)$, and $h_{ii}(y_1, y_2, z) = s_i(z)$ for $i \in \{0, 1\}$. Then both f_1 and f_2 are identical to the function recursively defined from $g_{i_1 i_2}$ and $h_{i_1 i_2}$.

We verify this for the LHS of the equation. The base cases (i.e. at least one of y_j is ε) are straightforward, so we will only consider recursion case. Fix any $i_1, i_2 \in \{0, 1\}$, we need to prove that

$$\begin{aligned} &\mathsf{ITE}(\mathsf{EQ}(s_{i_1}(y_1), s_{i_2}(y_2)), s_{i_2}(y_2), s_{i_1}(y_1)) \\ &= h_{i_1 i_2}(y_1, y_2, \mathsf{ITE}(\mathsf{EQ}(y_1, y_2), y_2, y_1)). \end{aligned} \tag{93}$$

We prove this by a case study on whether $i_1 = i_2$. (Note that the case study happens in meta-theory.)

Suppose that $i_1 = i_2$, then by the definition equations of EQ we know that $\mathsf{EQ}(s_{i_1}(y_1), s_{i_2}(y_2)) = \mathsf{EQ}(y_1, y_2)$. Therefore, the LHS of Equation (93) is PV-provably equal to

$$\mathsf{ITE}(\mathsf{EQ}(y_1, y_2), s_{i_2}(y_2), s_{i_1}(y_1))$$

which is further PV-provably equal to

$$s_{i_1}(\mathsf{ITE}(\mathsf{EQ}(y_1, y_2)), y_2, y_1) = h_{i_1 i_2}(y_1, y_2, \mathsf{ITE}(\mathsf{EQ}(y_1, y_2), y_2, y_1)).$$

(Here, we use the property that $i_1 = i_2$.)

Suppose that $i_1 \neq i_2$, then by the definition equations of EQ we know that $\mathsf{EQ}(s_{i_1}(y_1), s_{i_2}(y_2)) = 0$, and hence by unfolding ITE we know that the LHS of Equation (93) is PV-provably equal to $s_{i_1}(y_1) = h_{i_1 i_2}(y_1, y_2, z)$ for any z. □

Theorem 3.26. $\mathsf{PV} \vdash \mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$ *if and only if* $\mathsf{PV} \vdash s(\vec{x}) = t(\vec{x})$.

Proof. The ($\Leftarrow$) side is straightforward and we will only prove the ($\Rightarrow$) side. Suppose that $\mathsf{PV} \vdash \mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$. By Lemma 3.25 and (L44) substitution we have

$$\mathsf{PV} \vdash \mathsf{EQ}(s(\vec{x}), t(\vec{x})) \Rightarrow s(\vec{x}) = t(\vec{x}).$$

Then $\mathsf{PV} \vdash s(\vec{x}) = t(\vec{x})$ by Modus Ponens (see Proposition 3.10). □

Remark 3.5. One remark to this theorem is that the ($\Rightarrow$) direction is proved in a *black-box* manner: The transformation from a proof of $\mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$ to a proof of $s(\vec{x}) = t(\vec{x})$ does not read the proof; the only information needed is the proof is correct and the last line is $\mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$.

4 Basic Proof Theory of PV

Recall that in the previous section, we have already proved several meta-theorems that serves as an abstraction of a particular proof tactic. However, the meta-theorems are still far from informal mathematical reasoning.

In this section, we start to develop more tools for *reasoning* in PV. We will develop a natural proof system that is close to human reasoning, and in particular, is close to the informal notion of feasible mathematics. This will be an important step towards constructing more advanced algorithms and data structures as the results developed in this section will relieve us from writing long and tedious PV proofs.

4.1 A Predicate Logic and its Proof System

We first define a proof system PV-PL that, intuitively, extends PV by allowing *logical connectives* $\{\to, \wedge, \vee, \neg\}$ and *conditional proofs*. However, we will (at this time) refrain from including *quantifiers* in our theory, as the interpretation of quantifiers (in particular, the existential quantifier) in feasible mathematics is not very clear at this moment.

Syntax. We define the syntax of the system PV-PL as follows:

We will denote PV-PL formulas using Greek letters.

- An *atomic formula* of PV-PL is either $\bot$ (denoting contradiction) or an equation in the language of PV. A *formula* of PV-PL is either an atomic formula, or a composition of atomic formulas using the logical connective $\to$ for implication. This is without loss of generality, as we can define other propositional logic connectives from $\{\bot, \to\}$, for instance:

$$\neg\varphi := \varphi \to \bot; \quad \varphi \vee \psi := \neg\varphi \to \psi; \quad \varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi).$$

- A *assertion* of PV-PL is of form $\Gamma \vdash \varphi$, where Γ is a finite sequence of formulas and φ is a formula. Γ is called the *antecedents*, and φ is called the *consequence*.
- We say that x is a *variable* of an atomic formula is x is a variable of the PV equation. Similarly, x is a variable of a formula if it is a variable of any atomic formula inside it, and is called a variable of an assertion if it is a variable of any formula of the assertion.
- A *deduction rule* of PV-PL is written of form

$$\frac{\Gamma_1 \vdash \varphi_1 \quad \dots \quad \Gamma_k \vdash \varphi_k}{\Gamma \vdash \varphi},$$

  where the assertions above the line are called *premises* and the assertion below the line is called the *conclusion*. An *axiom* of PV-PL is a deduction rule with no premise.
- A *proof* of a PV-PL assertion $\Gamma \vdash \varphi$ is a tree, where each leaf is an axiom, each internal node is a deduction rule, and the root is $\Gamma \vdash \varphi$. A PV-PL assertion is said to be *provable* if there is a proof of it.

Interpretation of assertions. Suppose that $\vec{x}$ contains all variables occurred in $\Gamma \vdash \varphi$, the intuitive interpretation of the assertion is that for every $\vec{n}$ running over the universe, if $\alpha[\vec{x}/\vec{n}]$ is true for all $\alpha \in \Gamma$, then $\varphi[\vec{x}/\vec{n}]$ is also true. That is:

- Suppose x appears in both Γ and φ, they are considered as the occurrences of the same variable, instead of different variables.
- Variables in an assertion are considered to be universally quantified.

We can define models (and the standard model) and interpretations of assertions similar to the definition of models and interpretations of PV, which is left as an exercise.

Deduction rules. There are three groups of deduction rules. Firstly, we have *structural rules* that deal with the antecedents.

- (W). The weakening rule is used to introduce a dummy condition in the antecedent of an assertion:
$$\frac{\Gamma \vdash \varphi}{\Gamma, \alpha \vdash \varphi}.$$
- (C). The contraction rule removes redundant conditions in the antecedent:
$$\frac{\Gamma, \alpha, \alpha \vdash \varphi}{\Gamma, \alpha \vdash \varphi}$$
- (P). The permutation rule allows an arbitrary permutation of conditions in the antecedent:
$$\frac{\Gamma, \alpha, \beta, \Delta \vdash \varphi}{\Gamma, \beta, \alpha, \Delta \vdash \varphi}.$$

Note that the contraction and permutation rules essentially makes the antecedent a set rather than a sequence. Some authors directly define the antecedent as a set so that both rules are not necessary, while others also define the antecedent as a multiset so that the permutation rule can be removed.

We have the following logical rules that deals with assumptions, logical connectives, equations, as well as variables in the formulas.

- (A). This axiom is used to apply an assumption:
$$\frac{}{\Gamma, \alpha \vdash \alpha}.$$
- ($\to_i$). This rule is used to introduce an implication connective to the conclusion, formally:
$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}.$$
- ($\to_e$). This rule is used to eliminate an implication connective in the conclusion, formally:
$$\frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$
- ($\bot_e$). This rule is used to eliminate a contradiction in the conclusion, which captures the standard "proof by contradiction" tactic:
$$\frac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi},$$
where $\neg\varphi$ is a shorthand of $\varphi \to \bot$.

Exercise: Define and prove the Explosion Rule from ($\bot_e$) and (W).

- (V). This is the substitution rule for variables. Let x is a variable and t be a term that does not contain x. Then for any formula φ,

$$\frac{\Gamma \vdash \varphi}{\Gamma[x/t] \vdash \varphi[x/t]},$$

  where $\alpha[x/t]$ denotes the formula obtained by substituting all occurrences of x in α with t, and $\Gamma[x/t] := \{\alpha \in \Gamma \mid \alpha[x/t]\}$.
- $(=_r)$. This is the reflexivity axiom of equality: $\Gamma \vdash x = x$.
- $(=_s)$. This is the symmtricity axiom of equality: $\Gamma, x = y \vdash y = x$.
- $(=_t)$. This is the transitivity axiom of equality: $\Gamma, x = y, y = z \vdash x = z$.
- $(=_/)$. This is the substitution axiom of equality: $\Gamma, x = y \vdash t[z/x] = t[z/y]$.

We stress that the substitution rule (V) for variables is valid because the variables in an PV-PL assertion is considered *universally* quantified. This rule essentially means that if we can prove that for all x, $\Gamma \vdash \alpha$ is true, we can substitute x in both the antecedent and the conclusion to any term t. In particular, the conclusion of the deduction rule $\Gamma[x/t] \vdash \alpha[x/t]$ could be weaker than $\Gamma \vdash \alpha$.

Finally, we have non-logical axioms that allows us to include equations in PV as well as some useful facts. Concretely:

- (PV). This axiom is used to apply a PV-provable equation. Concretely, for every PV provable equation e, we have $\Gamma \vdash e$.
- (Dε). This axiom distinguishes the empty string ε and any non-empty string. For $i \in \{0, 1\}$, we have $\Gamma \vdash \varepsilon \neq s_i(x)$, where for all terms t_1 and t_2, $t_1 \neq t_2$ is a shorthand of $\neg(t_1 = t_2)$.
- (Di). This axiom distinguishes 0 and 1: $\Gamma \vdash s_0(x) \neq s_1(y)$.

Recall that the main reason that PV only includes a restricted version of structural induction rather than the standard version is that it is not clear how to define *conditional proofs* in PV. As we have already introduced the notion of assertions, we can now formulate the standard structural induction scheme as:

- (Ind_n). Suppose that t_1, t_2 are terms and $x_1, \cdots, x_n$ are variables, $n \in \mathbb{N}$. Then we have the following rule:

$$\frac{\begin{array}{c}\forall j \in [n] : \ \Gamma \vdash t_1[x_j/\varepsilon] = t_2[x_j/\varepsilon]; \quad \forall i_1 \ldots, i_n \in \{0,1\} : \\ \Gamma, t_1 = t_2 \vdash t_1[x_1/s_{i_1}(x_1), x_n/s_{i_n}(x_n)] = t_2[x_1/s_{i_1}(x_1), \ldots, x_n/s_{i_n}(x_n)]\end{array}}{\Gamma \vdash t_1 = t_2}.$$

  Note that the quantification over j and $i_1, \ldots, i_n$ is running in the meta-theory, instead of the proof system. That is, if we have all $2^n + n$ assumptions above the line, where $j \in [n]$ and $i_1, \ldots, i_n \in \{0, 1\}$, we can prove the conclusion below the line.

We note that (Ind_n) can be used to implement the method of case study (see Theorem 3.5) by simply ignoring the induction hypothesis $t_1 = t_2$, where "ignoring the induction hypothesis" can be formally done by the weakening rule (W).

Interpretation of the deduction rules. One may read a natural deduction proof from the root to the bottom as a goal-directed proof search. We use the elimination rule of $\bot_e$ (i.e. proof by contradiction) as an example. To prove that $\Gamma \vdash \alpha$, we can assume, towards a contradiction, that α is false and prove a contradiction, i.e., deduce $\bot$ from $\Gamma, \neg\alpha$.

Similarly, the induction rule (Ind_n) can be interpreted as follows. Suppose that we want to prove $\Gamma \vdash t_1 = t_2$. It suffices to prove that

- We can prove the identity if one of $x_1, \ldots, x_n$ is an empty string. That is, for every $j \in [n]$, we can prove $t_1 = t_2$ from Γ if we substitute x_j/ε.
- From Γ and $t_1 = t_2$, we can prove the identity $t_1 = t_2$ if we append a bit to each of $x_1, \ldots, x_n$. That is, for every $i_1, \ldots, i_n \in \{0, 1\}$, we can deduce $t_1[x_j/s_{i_j}(x_j)\ (\forall j)] = t_2[x_j/s_{i_j}(x_j)\ (\forall j)]$ from $t_1 = t_2$ and Γ.

This exactly captures the informal version of structural induction as we discussed in the first section.

The substitution rule (V) for variables, in the interpretation, is the tactic of proof by *generalization*. Suppose that we want to prove a fact $\varphi[x/t]$ from $\Gamma[x/t]$, the substitution rule (V) suggests that we can prove a more general form φ from Γ, where we pick a *fresh* variable x to replace some occurrences of the term t. Note that it is said to be more general as from $\Gamma \vdash \varphi$ we can deduce $\Gamma[x/t'] \vdash \varphi[x/t']$ for *any* term t' instead of just for $t = t'$.

Note that since the goal-directed proof search interpretation is closer to the standard writing style of mathematical reasoning, we will mostly write PV-PL proofs in this way. Nevertheless, it should be straightforward to translate such proofs into a proof tree in PV-PL.

The rule of cut. One important proof tactic in informal mathematics is to first propose a *lemma*, prove that the lemma suffices to derive the conclusion, and then prove the lemma. This is called the rule of *cut*, formally denoted as

$$(\mathrm{Cut}): \quad \frac{\Gamma \vdash \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}$$

where $\Gamma \vdash \psi$ is the final goal, and φ is the "lemma", or called the *cut formula*. Indeed, the rule of cut is *admissible* in the sense that it can be implemented by the axioms and rules in PV-PL. Concretely, we can use the following proof tree:

$$\frac{\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\,(\to_i) \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\,(\to_e)$$

to simulate the rule of cut.

4.2 Warmup: Reasoning in PV-PL

We first provide two examples to demonstrate the power of the system PV-PL. The first example considers a basic property of strings: If $s_i(x) = s_i(y)$, then $x = y$. The second example considers a form of the correctness of EQ (see Lemma 3.25).

Example 4.1. We will show that $s_i(x) = s_i(y) \vdash x = y$ for $i \in \{0, 1\}$. To see this, we first apply the substitution rule of equality ($=_/$) to prove that

$$s_i(x) = s_i(y) \vdash \mathsf{TR}(s_i(x)) = \mathsf{TR}(s_i(y)).$$

Therefore by the rule of cut it suffices to prove that

$$s_i(x) = s_i(y), \mathsf{TR}(s_i(x)) = \mathsf{TR}(s_i(y)) \vdash x = y.$$

Note that we can prove by the definition axiom of TR (using the rule (PV)) and weakening (to introduce dummy conditions) we can prove that

$$s_i(x) = s_i(y), \mathsf{TR}(s_i(x)) = \mathsf{TR}(s_i(y)) \vdash x = \mathsf{TR}(s_i(x)),$$

and therefore by the rule of cut it suffices to prove that

$$s_i(x) = s_i(y), \mathsf{TR}(s_i(x)) = \mathsf{TR}(s_i(y)), x = \mathsf{TR}(s_i(x)) \vdash x = y.$$

We can introduce $\mathsf{TR}(s_i(y)) = y$ in the antecedent using the rule of cut as above, and it suffices to prove that

$$s_i(x) = s_i(y), \mathsf{TR}(s_i(x)) = \mathsf{TR}(s_i(y)), x = \mathsf{TR}(s_i(x)), \mathsf{TR}(s_i(y)) = y \vdash x = y.$$

Then by the substitution (generalization) rule (V), it suffices to prove

$$s_i(x) = s_i(y), z_1 = z_3, z_2 = z_1, z_3 = z_4 \vdash z_2 = z_4.$$

The rest of the proof is simply applying transitivity of equality and the rule of cut, which is left as an exercise.

Example 4.2. As an example to show the power of the induction rule in PV-PL, we consider the correctness of equality (see Lemma 3.25), that is:

$$\vdash \mathsf{EQ}(y_1, y_2) = 1 \to y_1 = y_2,$$

where the conditional equations are replaced by the logical connective $\to$ that is natively supported in PV-PL.

To prove this, we first apply the introduction rule of $\to$ to move the assumption into the antecedent, and therefore it suffices to prove $\mathsf{EQ}(y_1, y_2) = 1 \vdash y_1 = y_2$. Now we perform an induction on y_1 and y_2 simultaneously using (Ind_2). This introduces the following subgoals:

- (Base Case 1). $\mathsf{EQ}(\varepsilon, y_2) = 1 \vdash \varepsilon = y_2$. To prove this, we apply a case study on y_2 using the induction rule and the weakening rule, which introduces the following sub-goals:
  - (Base Case 1.*i*). $\mathsf{EQ}(\varepsilon, \varepsilon) = 1 \vdash \varepsilon = \varepsilon$. This follows from a weakening (that removes the assumption), a generalization using (V) with $t = \varepsilon$, and applying the rule of reflexivity.
  - (Base Case 1.*ii*). Let $i \in \{0, 1\}$, we need to prove $\mathsf{EQ}(\varepsilon, s_i(x)) = 1 \vdash \varepsilon = s_i(x)$. Recall that by the definition axiom of EQ we have that $\mathsf{PV} \vdash \mathsf{EQ}(\varepsilon, s_i(x)) = 0$, and therefore by the ($\mathsf{PV}$) rule as well as weakening we can obtain that

$$\mathsf{EQ}(\varepsilon, s_i(x)) = 1 \vdash \mathsf{EQ}(\varepsilon, s_i(x)) = 0.$$

By the rule of cut, it suffices to prove that

$$\mathsf{EQ}(\varepsilon, s_i(x)) = 1, \mathsf{EQ}(\varepsilon, s_i(x)) = 0 \vdash \varepsilon = s_i(x).$$

By the explosion rule (which can be simulated by the elimination rule of $\bot$ and weakening), it suffices to prove that

$$\mathsf{EQ}(\varepsilon, s_i(x)) = 1, \mathsf{EQ}(\varepsilon, s_i(x)) = 0 \vdash \bot. \tag{94}$$

From the antecedent we can deduce that $0 = \mathsf{EQ}(\varepsilon, s_i(x))$ (by symmetricity) and subsequently $0 = 1$ (by transitivity). Also, from the axiom (Di) we know that $0 \neq 1$, i.e., $\vdash 0 = 1 \to \bot$. Therefore, by applying the elimination rule of $\to$ we can conclude Equation (94), which concludes the sub-goal.

- (Base Case 2). $\mathsf{EQ}(y_1, \varepsilon) = 1 \vdash y_1 = \varepsilon$. The proof is almost identical to the proof of the previous case, and is therefore omitted.
- (Base Case 3). $\mathsf{EQ}(s_i(y_1), s_j(y_2)) = 1, y_1 = y_2 \vdash s_i(y_1) = s_j(y_2)$, where $i = j$. This can be proved by weakening (to remove the first assumption) and subsequently applying the substitution rule of equality ($=_/$).
- (Base Case 4). $\mathsf{EQ}(s_i(y_1), s_j(y_2)) = 1, y_1 = y_2 \vdash s_i(y_1) = s_j(y_2)$, where $i \neq j$. In this case, we know from the definition rule that $\mathsf{EQ}(s_i(y_1), s_j(y_2)) = 0$. The rest of the proof is similar to that of the Base Case 1, which is left as an exercise.

Through these examples, one can notice that an informal proof almost directly translates to a natural deduction form of proof in PV-PL. In the rest of the note, we may simply write informal proofs in case that it translates to a PV-PL proof straightforwardly.

Remark 4.1. One can verify that the PV-PL rules for logical connectives are complete for propositional logic in the sense that if φ can be derived from Γ only through propositional logic deduction, then $\Gamma \vdash \varphi$ is PV-PL provable. Moreover, this is also true for connectives defined from $\{\to, \bot\}$ such as $\wedge$ and $\vee$.

We say that a rule is *admissible* in PV-PL if it can be implemented by PV-PL proof trees. In particular, the following rules for $\wedge$ and $\vee$ can be implemented by PV-PL proof trees.

$$\frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi}(\wedge_e^1) \quad \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi}(\wedge_e^2) \quad \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}(\wedge_i)$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi}(\vee_i^1) \quad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi}(\vee_i^2) \quad \frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \alpha \quad \Gamma, \psi \vdash \alpha}{\Gamma \vdash \beta}(\vee_e)$$

4.3 Translation Theorem

Now we are ready to formulate and prove a theorem that connects PV-PL and PV. By the (PV) rule in PV-PL we know that a provable equation in PV is also provable in PV-PL, and therefore PV-PL is an extension of PV-PL. Moreover, we can prove that the extension does not introduce any provable equation that is not provable in PV. Formally:

Theorem 4.1. *Let e be an equation in* PV. *Then $\vdash e$ is provable in* PV-PL *if and only if* PV $\vdash e$.

Such an extension T' of a theory T that does not introduce new provable theorems that can be formulated in the language of theory T is called a *conservative extension*. In particular, the theorem suggests that PV-PL is a conservative extension of the theory PV.

An extension of conditional equations. We also note that since $t \Rightarrow e$ itself is an equation, we can also talk about a "conditional conditional equation" $t_1 \Rightarrow (t_2 \Rightarrow e)$, or even with more $(\Rightarrow)$'s. We may remove the parentheses and assume that $\Rightarrow$ is right-associative.

We slightly extend the notation. For a sequence of terms $t_c^{(1)}, \ldots, t_c^{(\ell)}$, $\ell \in \mathbb{N}$, we define

$$t_c^{(1)}, \ldots, t_c^{(\ell)} \Rightarrow t_1 = t_2$$

as

$$t_c^{(1)} \Rightarrow \ldots \Rightarrow t_c^{(\ell)} \Rightarrow t_1 = t_2,$$

which intuitively means that the conjunction of all conditions $t_c^{(1)}, \ldots, t_c^{(\ell)}$ imply $t_1 = t_2$.

We use $\mathsf{RHS}[e]$ and $\mathsf{LHS}[e]$ to represent the RHS and LHS of the conditiona equation e. Note that they are not PV functions but meta-functions (in the meta-theory) dealing with formulas. We use $e_1 \equiv e_2$ to denote that the terms (or equations) e_1 and e_2 are the same terms (or equations).

Nevertheless, they should be feasible assuming reasonable encoding of PV equations.

The PV Translation. Indeed, Theorem 4.1 is a consequence of an even stronger theorem called the *translation theorem*. Let $\Gamma \vdash \varphi$ be an assertion, we define the PV translation of it, denoted by $[\Gamma \vdash \varphi]_{\mathsf{PV}}$, as a PV equation defined as follows:

- (*Translation of Assertion*): Let $\Gamma = (\alpha_1, \ldots, \alpha_n)$, $n \in \mathbb{N}$. We define

  $$[\Gamma \vdash \varphi]_{\mathsf{PV}} := \text{“}[\alpha_n]_{\mathsf{PV}} \Rightarrow [\alpha_{n-1}]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\alpha_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} = 1\text{”},$$

  where $[\alpha]_{\mathsf{PV}}$ is the PV translation of formulas that will be defined later, and $t_c \Rightarrow t_1 = t_2$ denotes conditional equation. In particular, if $n \in \mathbb{N}$, we have

  $$[\Gamma \vdash \varphi]_{\mathsf{PV}} := [\varphi]_{\mathsf{PV}} = 1.$$

- (*Translation of Atomic Formula*). We define

  $$[\bot]_{\mathsf{PV}} := 0,$$
  $$[t = s]_{\mathsf{PV}} := \text{“}\mathsf{EQ}(t, s)\text{”}.$$

- (*Translation of the Connective*). We define

  $$[\varphi \to \psi]_{\mathsf{PV}} := \mathsf{ITE}([\varphi]_{\mathsf{PV}}, [\psi]_{\mathsf{PV}}, 1).$$

Proposition 4.2. *Let φ be a formula. Then* $\mathsf{PV} \vdash \mathsf{IsNotEps}([\varphi]_{\mathsf{PV}}) = 1$.

Proof Sketch. We perform an induction (in the meta-theory) on the outermost connective of φ. If φ is an atomic formula, then either $\varphi = \bot$ or φ is a PV equation. In the first case it suffices to prove that $\mathsf{PV} \vdash \mathsf{IsNotEps}(0) = 1$, which follows from a simple unfolding of IsNotEps, while in the latter case it suffices to prove that

$\mathsf{PV} \vdash \mathsf{IsNotEps}(\mathsf{EQ}(x, y)) = 1$, which follows from a simultaneous induction on x and y using Theorem 3.21.

Suppose that $\varphi := \psi_1 \to \psi_2$. From the induction hypothesis, we know that $\mathsf{PV} \vdash \mathsf{IsNotEps}([\psi_1]_{\mathsf{PV}}) = 1$ and $\mathsf{IsNotEps}([\psi_2]_{\mathsf{PV}}) = 1$. We will need to prove that

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(\mathsf{ITE}([\psi_1]_{\mathsf{PV}}, [\psi_2]_{\mathsf{PV}}, 1)) = 1. \tag{95}$$

Indeed, we will prove that

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow \mathsf{IsNotEps}(y) \Rightarrow \mathsf{IsNotEps}(\mathsf{ITE}(x, y, 1)) = 1. \tag{96}$$

If this is provable, we can derive Equation (95) by the substitution $x/[\psi_1]_{\mathsf{PV}}$, $y/[\psi_2]_{\mathsf{PV}}$ and applying Modus Ponens (see Proposition 3.10).

It remains to prove Equation (96). We perform a case study on x and y using Theorem 3.5, where all 9 cases can be proved by simple unfolding of $\mathsf{IsNotEps}$ and ITE. The details are omitted. □

Proposition 4.3. *Let φ be a formula. Then $\mathsf{PV} \vdash \mathsf{TR}([\varphi]_{\mathsf{PV}}) = \varepsilon$.*

The proof is similar to the proposition above. The detail is omitted.

The translation theorem claims that the PV translation of an assertion preserves its provability. Formally:

Theorem 4.4 (Translation Theorem)**.** *Let $\Gamma \vdash \varphi$ be a* PV-PL *assertion. Then $\Gamma \vdash \varphi$ is provable in* PV-PL *if and only if $\mathsf{PV} \vdash [\Gamma \vdash \varphi]_{\mathsf{PV}}$.*

To see that the translation theorem implies Theorem 4.1, notice that if we take $\Gamma = \varnothing$ and $\varphi := s = t$, we will have that $[\Gamma \vdash \varphi]_{\mathsf{PV}} = \text{“}\mathsf{EQ}(s, t) = 1$. Therefore, suppose that $\vdash s = t$ is provable in PV-PL, we have that $\mathsf{PV} \vdash \mathsf{EQ}(s, t) = 1$, which implies that $\mathsf{PV} \vdash s = t$ by Lemma 3.25.

Road map to the proof of the translation theorem. The proof of the translation theorem is highly constructive. Indeed, we will show that each deduction rule in PV-PL is PV-*admissible*, in the sense that if the PV translation of all premises of the rule are PV-provable, then the PV translation of the conclusion is also PV-provable; this implies the proof of translation theorem by a structural induction on the PV-PL proof tree.

In the rest of this subsection, we will prove the admissibility of all PV-PL rules, which concludes the translation theorem.

4.4 Admissibility of Structural Rules

We prove the admissibility of the structural rules, including the weakening, contraction, and permutation rule that deals with the antecedents.

4.4.1 Admissibility of Weakening

Lemma 4.5. *The weakening rule is admissible. That is, for any assertion $\Gamma \vdash \varphi$, suppose that $\mathsf{PV} \vdash [\Gamma \vdash \varphi]_{\mathsf{PV}}$, then $\mathsf{PV} \vdash [\Gamma, \alpha \vdash \varphi]_{\mathsf{PV}}$.*

Let $\Gamma = (\beta_1, \ldots, \beta_n)$, $n \in \mathbb{N}$. By the definition of the PV translation, we know that it suffices to prove that

$$\frac{\mathsf{PV} \vdash [\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} = 1}{\mathsf{PV} \vdash [\alpha]_{\mathsf{PV}} \Rightarrow [\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} = 1}.$$

Moreover, it suffices to prove that:

Proposition 4.6. *Let t_c, t_1, t_2 be terms. If $\mathsf{PV} \vdash t_1 = t_2$ and $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c) = 1$, $\mathsf{PV} \vdash t_c \Rightarrow t_1 = t_2$.*

To see this, we can treat $[\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} = 1$ as the equation $s = t$ and $[\alpha]_{\mathsf{PV}}$ as the term t', and that $\mathsf{PV} \vdash \mathsf{IsNotEps}([\alpha]_{\mathsf{PV}}) = 1$ by Proposition 4.2.

Furthermore, the following proposition implies the fact above by the substitution z/t', x/s, y/t, and Modus Ponens (see Proposition 3.10).

Proposition 4.7. $\mathsf{PV} \vdash \mathsf{IsNotEps}(z) \Rightarrow \mathsf{EQ}(x, y) \Rightarrow z \Rightarrow x = y$.

It remains to prove the proposition.

Proposition 4.8. PV *proves that*

- $\mathsf{LHS}[s_1(z) \Rightarrow x = y] = \mathsf{LHS}[x = y]$
- $\mathsf{RHS}[s_1(z) \Rightarrow x = y] = \mathsf{RHS}[x = y]$

Proof Sketch. Both equations can be proved by a straightforward unfolding of ITE in the definition of conditional equations. □

Proof of Proposition 4.7. We can prove this by a case study on z using Theorem 3.5. In case for z/ε, $\mathsf{IsNotEps}(\varepsilon) = 0$ and thus we prove the equation by the Explosion Rule (see Proposition 3.11). Otherwise, for $i \in \{0, 1\}$, we need to prove the equation after the substitution $z/s_i(z)$. In either case, we know that $\mathsf{IsNotEps}(s_i(z)) = 1$, and thus by Proposition 4.8

$$\begin{aligned}
&\mathsf{PV} \vdash \mathsf{LHS}[\mathsf{IsNotEps}(s_i(z)) \Rightarrow e] = \mathsf{LHS}[e], \\
&\mathsf{PV} \vdash \mathsf{RHS}[\mathsf{IsNotEps}(s_i(z)) \Rightarrow e] = \mathsf{RHS}[e],
\end{aligned}$$

where e is defined as the equation "$\mathsf{EQ}(x, y) \Rightarrow s_i(z) \Rightarrow x = y$". Subsequently, it suffices to prove $\mathsf{LHS}[e] = \mathsf{RHS}[e]$, i.e., the equation

$$e \equiv \text{“}\mathsf{EQ}(x, y) \Rightarrow s_i(z) \Rightarrow x = y\text{”}.$$

For the case that $i = 0$, notice that the conditional equation $s_0(z) \Rightarrow x = y$ is provable by the Explosion Rule (see Proposition 3.11), and thus by Proposition 4.8 we know that e is also provable.

For the case that $i = 1$, let $e' := s_1(z) \Rightarrow x = y$, notice that

$$\begin{aligned}
e &\equiv \text{“}\mathsf{LHS}[e] = \mathsf{RHS}[e]\text{”} \\
&\equiv \text{“}\mathsf{ITE}(\mathsf{EQ}(x, y), \mathsf{RHS}[e'], \mathsf{LHS}[e']) = \mathsf{LHS}[e']\text{”} \\
&\equiv \text{“}\mathsf{ITE}(\mathsf{EQ}(x, y), x, \mathsf{ITE}(s_1(z), y, x)) = \mathsf{ITE}(s_1(z), y, x)\text{”}.
\end{aligned}$$

By unfolding ITE, it suffices to prove that $\mathsf{ITE}(\mathsf{EQ}(x, y), x, y) = y$, i.e., $\mathsf{EQ}(x, y) \Rightarrow y = x$. This can be proved following the proof of the correctness of EQ, i.e., $\mathsf{EQ}(x, y) \Rightarrow x = y$ (see Lemma 3.25). □

4.4.2 Admissibility of Contraction

Lemma 4.9. *The contraction rule is admissible. That is, for any assertion* $\Gamma, \alpha \vdash \varphi$*, suppose that* $\mathsf{PV} \vdash [\Gamma, \alpha, \alpha \vdash \varphi]_{\mathsf{PV}}$*, then* $\mathsf{PV} \vdash [\Gamma, \alpha \vdash \varphi]_{\mathsf{PV}}$*.*

Similar to the case for weakening, we know by unfolding the definition of the PV translation that it suffices to prove the following meta-theorem of PV:

Proposition 4.10. *Let* t_c, t_1, t_2 *be terms. Suppose that* $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c) = 1$ *and* $\mathsf{PV} \vdash t_c \Rightarrow t_1 = t_2$*, then* $\mathsf{PV} \vdash t_c \Rightarrow t_c \Rightarrow t_1 = t_2$*.*

To prove the proposition, we need to further extend the syntactic of conditional equations to allow equations (as well as conditional equations) to appear in the antecedent of a conditional equation, rather than only in the conclusion of a conditional equation. Let e be an equation, we use $[e]_{\mathsf{EQ}}$ to denote the term $\mathsf{EQ}(\mathsf{LHS}[e], \mathsf{RHS}[e])$. We define the conditional equation $e \Rightarrow s = t$ as the equation $[e]_{\mathsf{EQ}} \Rightarrow s = t$. Similarly, the equation

$$\text{“}(e_1 \Rightarrow e_2) \Rightarrow s = t\text{”} \equiv \text{“}[[e_1]_{\mathsf{EQ}} \Rightarrow \mathsf{LHS}[e_2] = \mathsf{RHS}[e_2]]_{\mathsf{EQ}} \Rightarrow s = t\text{''}.$$

We can therefore allow the formulation of an arbitrary composition of equations and the condition symbol $\Rightarrow$.

By substitution and Modus Ponens (see Proposition 3.10), it is easy to verify that the following proposition implies Proposition 4.10.

Proposition 4.11. $\mathsf{PV} \vdash \mathsf{IsNotEps}(z) \Rightarrow (x = y) \Rightarrow z \Rightarrow x = y$*.*

Proof Sketch. The proof is done by a case study on z using Theorem 3.5. We will only demonstrate the case for $z/s_1(z)$, while other two cases z/ε and $z/s_0(z)$ are left as exercise.

By the definition of conditional equation, we know that $s_1(z) \Rightarrow x = y$ is the equation $\mathsf{ITE}(s_1(z), y, x) = x$. It then suffices to prove that

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) \Rightarrow \mathsf{ITE}(s_1(z), y, x) = x,$$

which is the equation

$$\mathsf{PV} \vdash \mathsf{ITE}(\mathsf{EQ}(x, y), x, \mathsf{ITE}(s_1(z), y, x)) = \mathsf{ITE}(s_1(z), y, x).$$

(Recall that we can add a dummy condition $\mathsf{IsNotEps}(z)$ by Proposition 4.6.)

Note that $\mathsf{PV} \vdash \mathsf{ITE}(s_1(z), y, x) = y$. Therefore, it suffices to prove that $\mathsf{PV} \vdash \mathsf{ITE}(\mathsf{EQ}(x, y), x, y) = y$, which is exactly $\mathsf{PV} \vdash \mathsf{EQ}(x, y) \Rightarrow y = x$. This can be proved similar to the proof of Lemma 3.25. □

4.4.3 Admissibility of Permutation

Lemma 4.12. *The permutation rule is admissible. That is, for any assertion* $\Gamma, \alpha, \beta, \Delta \vdash \varphi$*, suppose that* $\mathsf{PV} \vdash [\Gamma, \alpha, \beta, \Delta \vdash \varphi]_{\mathsf{PV}}$*, then* $\mathsf{PV} \vdash [\Gamma, \beta, \alpha, \Delta \vdash \varphi]_{\mathsf{PV}}$*.*

Again, to prove the admissibility of permutation, it suffices to prove the following meta-theorem of PV:

Proposition 4.13. *Let* $s_1, \ldots, s_n, t_c^1, t_c^2, t_1, t_2$ *be terms,* $n \in \mathbb{N}$. *Suppose that* $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c^i) = 1$ *for* $i \in \{0, 1\}$, *and that* $\mathsf{PV} \vdash s_n \Rightarrow \ldots \Rightarrow s_1 \Rightarrow t_c^1 \Rightarrow t_c^2 \Rightarrow t_1 = t_2$, *then*

$$\mathsf{PV} \vdash s_n \Rightarrow \ldots \Rightarrow s_1 \Rightarrow t_c^2 \Rightarrow t_c^1 \Rightarrow t_1 = t_2.$$

This can be easily derived from the following proposition:

Proposition 4.14. *Let* $n \in \mathbb{N}$. *Then* PV *proves*

$$\begin{aligned}&\mathsf{IsNotEps}(z_1) \Rightarrow \mathsf{IsNotEps}(z_2) \Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow z_1 \Rightarrow z_2 \Rightarrow x = y)\\ &\quad \Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow z_2 \Rightarrow z_1 \Rightarrow x = y).\end{aligned}$$

We will prove this by a case study on z using Theorem 3.5. The cases for z_1/ε or z_2/ε are easy. For instance, if we substitute z_2/ε, we have that $\mathsf{IsNotEps}(z_2) = 0$ and thus by the Explosion Rule (see Proposition 3.11),

$$\begin{aligned}\mathsf{PV} \vdash \mathsf{IsNotEps}(\varepsilon) &\Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow z_1 \Rightarrow \varepsilon \Rightarrow x = y)\\ &\Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow \varepsilon \Rightarrow z_1 \Rightarrow x = y).\end{aligned}$$

We can further add a dummy condition $\mathsf{IsNotEps}(z_1)$ by Proposition 4.6.

Now we consider the case for $z_1/s_i(z_1)$ and $z_2/s_j(z_2)$. We will prove that

$$\begin{aligned}\mathsf{PV} \vdash (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_i(z_1) \Rightarrow s_j(z_2) \Rightarrow x = y)\\ \Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_j(z_2) \Rightarrow s_i(z_1) \Rightarrow x = y).\end{aligned}$$

Again, if $i = 0$ or $j = 0$, we can complete the proof by the Explosion Rule (see Proposition 3.11) and adding dummy conditions using Proposition 4.6. Therefore, it suffices to deal with the case for $i = j = 1$.

This will be proved by an induction on n in the meta-theory. Suppose that $n = 0$, we will need to prove that:

Proposition 4.15. $\mathsf{PV} \vdash (s_1(z_1) \Rightarrow s_1(z_2) \Rightarrow x = y) \Rightarrow s_1(z_1) \Rightarrow s_1(z_2) \Rightarrow x = y$.

By unfolding all the definitions, this equation will essentially reduce to the correctness of equality. The details are omitted.

Now we consider induction case. That is, if

$$\begin{aligned}\mathsf{PV} \vdash (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_i(z_1) \Rightarrow s_j(z_2) \Rightarrow x = y)\\ \Rightarrow (w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_j(z_2) \Rightarrow s_i(z_1) \Rightarrow x = y),\end{aligned}$$

then

$$\begin{aligned}\mathsf{PV} \vdash (w_{n+1} \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_i(z_1) \Rightarrow s_j(z_2) \Rightarrow x = y)\\ \Rightarrow (w_{n+1} \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_j(z_2) \Rightarrow s_i(z_1) \Rightarrow x = y),\end{aligned}$$

With the substitution $x/[w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_i(z_1) \Rightarrow s_j(z_2) \Rightarrow x = y]_{\mathsf{EQ}}$, $y_1/\mathsf{LHS}[w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_j(z_2) \Rightarrow s_i(z_1) \Rightarrow x = y]$, $y_2/\mathsf{RHS}[w_n \Rightarrow \ldots \Rightarrow w_1 \Rightarrow s_j(z_2) \Rightarrow s_i(z_1) \Rightarrow x = y]$, w/w_{n+1} and Modus Ponens (see Proposition 3.10), it suffices to prove the "transitivity" of $\Rightarrow$, formalized as:

Proposition 4.16. $\mathsf{PV} \vdash (x \Rightarrow y_1 = y_2) \Rightarrow (w \Rightarrow x) \Rightarrow w \Rightarrow y_1 = y_2$.

This can be proved by applying a case study on w and x using Theorem 3.5, and using the correctness of EQ (see Lemma 3.25). The detail is omitted and left as an exercise.

4.5 Admissibility of Axioms for Equality

Now we consider the admissibility of axioms for equality, including the logical axioms $(=_r)$, $(=_s)$, $(=_t)$, and $(=_/)$, as well as the non-logical axioms (PV), (Dε), and (Di).

4.5.1 Non-logical Axioms about Equality

The non-logical axioms are relatively straightforward, so we quickly scan over them.

Lemma 4.17. *The* (PV) *axiom is admissible. That is, for any* Γ *and any* PV *provable equation* e, $\mathsf{PV} \vdash [\Gamma \vdash e]_{\mathsf{PV}}$.

Proof. Let $e \equiv t_1 = t_2$. It suffices to prove that $\mathsf{PV} \vdash [\vdash e]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that $[\vdash e]_{\mathsf{PV}} \equiv$ "$\mathsf{EQ}(t_1, t_2) = 1$". Since $\mathsf{PV} \vdash t_1 = t_2$, we know by the correctness of EQ (see Lemma 3.25) that $\mathsf{EQ}(t_1, t_2) = 1$. □

Lemma 4.18. *The* (Dε) *axiom is admissible. That is, for any* Γ *and* $i \in \{0, 1\}$, $\mathsf{PV} \vdash [\Gamma \vdash \varepsilon \neq s_i(x)]_{\mathsf{PV}}$.

Proof. It suffices to prove that $\mathsf{PV} \vdash [\vdash \varepsilon \neq s_i(x)]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that $[\vdash \varepsilon \neq s_i(x)]_{\mathsf{PV}} \equiv$ "$\mathsf{EQ}(\varepsilon, s_i(x)) \Rightarrow 0 = 1$". By the definition axioms of EQ we know that this conditional equation PV-provably evaluates to $0 \Rightarrow 0 = 1$, which is provable in PV by the Explosion Rule (see Proposition 3.11). □

Lemma 4.19. *The* (Di) *axiom is admissible. That is, for any* Γ, $\mathsf{PV} \vdash [\Gamma \vdash s_0(x) \neq s_1(y)]$.

Proof. It suffices to prove that $\mathsf{PV} \vdash [\vdash s_0(x) \neq s_1(y)]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that $[\vdash s_0(x) \neq s_1(y)]_{\mathsf{PV}} \equiv$ "$\mathsf{EQ}(s_0(x), s_1(y)) \Rightarrow 0 = 1$". By the definition axioms of EQ we know that this conditional equation PV-provably evaluates to $0 \Rightarrow 0 = 1$, which is provable in PV by the Explosion Rule (see Proposition 3.11). □

4.5.2 Logical Axioms for Equality

Lemma 4.20. *The reflexivity rule* $(=_r)$ *is admissible. That is, for any* Γ, $\mathsf{PV} \vdash [\Gamma \vdash x = x]_{\mathsf{PV}}$.

Proof. It suffices to prove that $\mathsf{PV} \vdash [\vdash x = x]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that $[\vdash x = x]_{\mathsf{PV}} \equiv$ "$\mathsf{EQ}(x, x) = 1$", which is provable in PV by a simple induction on x. □

Lemma 4.21. *The symmetricity rule* $(=_s)$ *is admissible. That is, for any* Γ, $\mathsf{PV} \vdash [\Gamma, x = y \vdash y = x]_{\mathsf{PV}}$.

Proof Sketch. It suffices to prove that $\mathsf{PV} \vdash [x = y \vdash y = x]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that $[x = y \vdash y = x]_{\mathsf{PV}} \equiv$ "$\mathsf{EQ}(x, y) \Rightarrow \mathsf{EQ}(y, x) = 1$", or equivalently:

$$\mathsf{ITE}(\mathsf{EQ}(x, y), 1, \mathsf{EQ}(y, x)) = \mathsf{EQ}(y, x).$$

This can be proved by simultaneous induction on x, y using Theorem 3.21, which is similar to the proof of the correctness of EQ (see Lemma 3.25). □

Lemma 4.22. *The transitivity rule* $(=_t)$ *is admissible. That is, for any* Γ, $\mathsf{PV} \vdash [\Gamma, x = y, y = z \vdash x = z]_{\mathsf{PV}}$.

Proof Sketch. It suffices to prove that $\mathsf{PV} \vdash [x = y, y = z \vdash x = z]_{\mathsf{PV}}$ and apply the admissibility of weakening. Notice that

$$
\begin{aligned}
&[x = y, y = z \vdash x = z]_{\mathsf{PV}} \\
\equiv\ &\text{“}\mathsf{EQ}(y, z) \Rightarrow \mathsf{EQ}(x, y) \Rightarrow \mathsf{EQ}(x, z) = 1\text{”} \\
\equiv\ &\text{“}\mathsf{EQ}(y, z) \Rightarrow \mathsf{ITE}(\mathsf{EQ}(x, y), 1, \mathsf{EQ}(x, z)) = \mathsf{EQ}(x, z)\text{”} \\
\equiv\ &\text{“}\mathsf{ITE}(\mathsf{EQ}(y, z), \mathsf{EQ}(x, z), \mathsf{ITE}(\mathsf{EQ}(x, y), 1, \mathsf{EQ}(x, z))) = \mathsf{ITE}(\mathsf{EQ}(x, y), 1, \mathsf{EQ}(x, z))\text{”}.
\end{aligned}
$$

To see that this is provable in PV, we will perform induction simultaneously on x, y, and z using Theorem 3.21 (see Remark 3.4). Specifically, we will show that both sides of the equation are identical to the function $f(x, y, z)$ recursively defined by the equations:

$$
\begin{gathered}
f(\varepsilon, y, z) := \mathsf{ITE}(\mathsf{EQ}(\varepsilon, y), 1, \mathsf{EQ}(\varepsilon, z)), \quad f(x, \varepsilon, z) := \mathsf{ITE}(\mathsf{EQ}(x, \varepsilon), 1, \mathsf{EQ}(x, z)), \\
f(x, y, \varepsilon) := \mathsf{ITE}(\mathsf{EQ}(x, y), 1, \mathsf{EQ}(x, \varepsilon)) \\
f(s_i(x), s_j(y), s_k(z)) := \begin{cases} f(x, y, z) & i = j = k \\ \mathsf{ITE}(\mathsf{EQ}(x, y), 1, 0) & i = j \wedge i \neq k \\ \mathsf{EQ}(x, z) & i = k \wedge i \neq j \\ 0 & j = k \wedge i \neq j \end{cases}
\end{gathered}
$$

Note that for cases where one of x, y, z is substituted by ε, we need to further perform an induction on the remaining variables. The details are omitted and left as an exercise. □

Before proving the admissibility of the substitution rule for equality, we first prove a technical lemma showing the correctness of EQ in conditional equations.

Proposition 4.23. $\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow (x \Rightarrow y = z) \Rightarrow x \Rightarrow \mathsf{EQ}(y, z) = 1$. *Moreover, if for any terms* t_c, t_1, t_2, *if* $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c)$ *and* $\mathsf{PV} \vdash t_c \Rightarrow t_1 = t_2$, *then* $\mathsf{PV} \vdash t_c \Rightarrow \mathsf{EQ}(t_1, t_2) = 1$.

Proof Sketch. The “moreover” part follows straightforwardly, so we will only prove the first part. To show that $\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow (x \Rightarrow y = z) \Rightarrow x \Rightarrow \mathsf{EQ}(y, z) = 1$, we perform a case study on z using Theorem 3.5. The case for z/ε can be proved using the Explosion Rule (see Proposition 3.11), and the case for $z/s_0(z)$ can be proved using the Explosion Rule by adding dummy conditions using Proposition 4.6.

For the case $z/s_1(z)$, the conditional equation PV-provably evaluates to $\mathsf{EQ}(y, z) \Rightarrow \mathsf{EQ}(y, z) = 1$, which can be proved using the fact that

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow \mathsf{TR}(x) = \varepsilon \Rightarrow x \Rightarrow x = 1$$

(which can be proved by a simple case study) and that $\mathsf{PV} \vdash \mathsf{IsNotEps}(\mathsf{EQ}(y, z)) = 1$ and $\mathsf{PV} \vdash \mathsf{TR}(\mathsf{EQ}(y, z)) = \varepsilon$ (both of which can be proved by simultaneous induction on y and z. □

Lemma 4.24. *The substitution rule* $(=_/)$ *for equality is admissible. That is, for any* Γ, $\mathsf{PV} \vdash [\Gamma, x = y \vdash t[z/x] = t[z/y]]_{\mathsf{PV}}$.

Proof Sketch. Again, it suffices to prove that $\mathsf{PV} \vdash [x = y \vdash t[z/x] = t[z/y]]_{\mathsf{PV}}$, or equivalently:

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) \Rightarrow \mathsf{EQ}(t[z/x], t[z/y]) = 1.$$

Indeed, we will prove a stronger result (by Proposition 4.23) that

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) \Rightarrow t[z/x] = t[z/y].$$

Suppose that $z, \vec{w}$ are the variables occurred in t, and $f_t(z, \vec{w}) = t$ is the function defined from t using the composition rule in PV. It suffices to prove that

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) \Rightarrow f_t(x, \vec{w}) = f_t(y, \vec{w}),$$

or equivalently $\mathsf{PV} \vdash \mathsf{ITE}(\mathsf{EQ}(x, y), f_t(y, \vec{w}), f_t(x, \vec{w})) = f_t(x, \vec{w})$.

Auxiliary functions. Let $\mathsf{EQL}(x, y)$ be the function that outputs 1 (resp. 0) if and only if x and y are of the same length. It can be defined, for instance, by simultaneous recursion on x and y, provided that it proves

$$\mathsf{EQL}(\varepsilon, s_i(y)) = \mathsf{EQL}(s_j(x), \varepsilon) = 0, \quad \mathsf{EQL}(s_i(x), s_j(y)) = \mathsf{EQL}(x, y).$$

Let $\mathsf{Suf}(x, y)$ be the function that outputs the suffix of x of length $|y|$, that is:

$$\mathsf{Suf}(x, \varepsilon) := \varepsilon, \quad \mathsf{Suf}(\varepsilon, y) := \varepsilon$$
$$\mathsf{Suf}(s_i(x), s_j(y)) := s_i(\mathsf{Suf}(x, y)).$$

It can be proved by an induction on x that $\mathsf{Suf}(x, y \circ x) = x$. Moreover, we can prove the following properties of Suf:

Proposition 4.25. PV *proves the following equations:*

- $\mathsf{EQL}(y, z) \Rightarrow \mathsf{Suf}(x, y) = \mathsf{Suf}(x, z)$.
- $\mathsf{Suf}(x, y \circ x) = x$.
- $\mathsf{Suf}(x, s_i(y)) = \mathsf{LastBit}(\mathsf{ITR}(x, y)) \circ \mathsf{Suf}(x, y)$, $i \in \{0, 1\}$.
- $\mathsf{Suf}(x, x \circ y) = x$.

Proof Sketch. The first bullet can be proved by simultaneous induction on y and z using Theorem 3.21. The second bullet can be proved by an induction on x. The third bullet can be proved by simultaneous induction on x and y using Theorem 3.21, where we would need the first bullet as well as $\mathsf{EQL}(s_0(y), s_1(y)) = 1$ and Modus Ponens (see Proposition 3.10) to prove that $\mathsf{Suf}(x, s_0(y)) = \mathsf{Suf}(x, s_1(y))$. The last bullet can be proved by an induction on y and using the second and the third bullets to prove that $\mathsf{Suf}(x, x \circ \varepsilon) = x$ and $\mathsf{Suf}(x, x \circ s_i(y)) = \mathsf{Suf}(x, x \circ y)$. □

Hopefully I'm not making any stupid mistake here :)

A generalized equation. We will show that PV proves:

$$\mathsf{ITE}(\mathsf{EQ}(\mathsf{Suf}(x, z), \mathsf{Suf}(y, z)), f_t(\mathsf{ITR}(x, z) \circ \mathsf{Suf}(y, z), \vec{w}), f_t(x, \vec{w})) = f_t(x, \vec{w}). \quad (97)$$

To see that this suffices, we can substitute $z/x \circ y$, so that the LHS of the equation is PV-provably equal to $\mathsf{ITE}(\mathsf{EQ}(x, y), f_t(y, \vec{w}), f_t(x, \vec{w}))$ by Proposition 4.25.

It remains to prove Equation (97). We will perform an induction on the auxiliary variable z using the induction rule in PV. Concretely, we will show that both sides of the equation are identical to the function $g(z, x, y, \vec{w})$ recursively defined as

$$g(\varepsilon, x, y, \vec{w}) = f_t(x, \vec{w}),$$
$$g(s_i(z), x, y, \vec{w}) = \begin{cases} f_t(x, \vec{w}) & \mathsf{EQ}(\mathsf{LastBit}(\mathsf{ITR}(x, z)), \mathsf{LastBit}(\mathsf{ITR}(y, z))) = 1; \\ g(z, x, y, \vec{w}) & \text{otherwise.} \end{cases}$$

The RHS of Equation (97) is clearly identical to g.

Useful properties. Now we show that this is also true for the LHS, starting from proving a few properties that will help simplify Equation (97) with $z/s_i(z)$. Notice that by Proposition 4.25:

$$\begin{aligned} & \mathsf{EQ}(\mathsf{Suf}(x, s_i(z)), \mathsf{Suf}(y, s_i(z))) \\ = \ & \mathsf{EQ}(\mathsf{LastBit}(\mathsf{ITR}(x, z)) \circ \mathsf{Suf}(x, z), \mathsf{LastBit}(\mathsf{ITR}(y, z)) \circ \mathsf{Suf}(y, z)). \end{aligned} \tag{98}$$

Moreover, by simultaneous induction on x and y using Theorem 3.21, we can prove that $\mathsf{EQ}(\mathsf{LastBit}(z) \circ x, \mathsf{LastBit}(w) \circ y) = \mathsf{And}(\mathsf{EQ}(\mathsf{LastBit}(z), \mathsf{LastBit}(w)), \mathsf{EQ}(x, y))$, which implies that

$$(98) = \mathsf{And}(\mathsf{EQ}(\mathsf{LastBit}(\mathsf{ITR}(x, z)), \mathsf{LastBit}(\mathsf{ITR}(y, z))), \mathsf{EQ}(\mathsf{Suf}(x, z), \mathsf{Suf}(y, z))).$$

Also, notice that

$$\begin{aligned} & \mathsf{ITR}(x, s_i(z)) \circ \mathsf{Suf}(y, s_i(z)) \\ = \ & \mathsf{ITR}(x, s_i(z)) \circ (\mathsf{LastBit}(\mathsf{ITR}(y, z)) \circ \mathsf{Suf}(y, z)). && \text{(Proposition 4.25)} \\ = \ & \mathsf{ITR}(x, s_i(z)) \circ \mathsf{LastBit}(\mathsf{ITR}(y, z)) \circ \mathsf{Suf}(y, z) \end{aligned}$$

Proof of Equation (97). Note that it is clear that the LHS of Equation (97) with the substitution z/ε is identical to $g(\varepsilon, x, y, \vec{w})$ by unfolding. We will now simplify Equation (97) with $z/s_i(z)$. The strategy is to perform a case study on whether $\mathsf{LastBit}(\mathsf{ITR}(x, z)) = \mathsf{LastBit}(\mathsf{ITR}(y, z))$. Formally, let z' be a fresh variable, and we will prove that

$$\begin{cases} f_t(\mathsf{TR}(z') \circ \mathsf{LastBit}(z'') \circ \mathsf{Suf}(y, z), \vec{w}) & \mathsf{And}(\mathsf{EQ}(\mathsf{LastBit}(z'), \mathsf{LastBit}(z'')), \\ & \quad \mathsf{EQ}(\mathsf{Suf}(x, z), \mathsf{Suf}(y, z))) = 1 \\ f_t(x, \vec{w}) & \text{otherwise} \end{cases}$$
$$= \begin{cases} \mathsf{ITE}(\mathsf{EQ}(\mathsf{Suf}(x, z), \mathsf{Suf}(y, z)), f_t(z' \circ \mathsf{Suf}(y, z), \vec{w}), f_t(x, \vec{w})) & \mathsf{EQ}(\mathsf{LastBit}(z'), \mathsf{LastBit}(z'')) = 1; \\ f_t(x, \vec{w}) & \text{otherwise;} \end{cases}$$
$$\equiv \begin{cases} \mathsf{LHS}[(97)] & \mathsf{EQ}(\mathsf{LastBit}(z'), \mathsf{LastBit}(z'')) = 1; \\ f_t(x, \vec{w}) & \text{otherwise;} \end{cases}$$

To see that this suffices, we can substitute

$$z'/\mathsf{ITR}(x, z), \quad z''/\mathsf{ITR}(y, z)$$

so that according to the properties we proved above, the equation shows exactly that the LHS of Equation (97) is identical to the function g recursively defined as above.

Finally, we prove the equation above by a case study on z' and z'' using Theorem 3.5. In all cases that $\mathsf{LastBit}(z') \neq \mathsf{LastBit}(z'')$, both sides of the consequence of the conditional equation evaluates to $f_t(x, \vec{w})$. Now we consider the cases that $\mathsf{LastBit}(z') = \mathsf{LastBit}(z'')$, for instance, with the substitution $z'/s_0(z')$ and $z''/s_0(z'')$. In such case, the LHS of the equation evaluates to

$$\begin{cases} f_t(z' \circ 0 \circ \mathsf{Suf}(y,z), \vec{w}) & \mathsf{EQ}(\mathsf{Suf}(x,z), \mathsf{Suf}(y,z))) = 1; \\ f_t(x, \vec{w}) & \text{otherwise}, \end{cases}$$

while the RHS evaluates to

$$\begin{aligned} &\mathsf{ITE}(\mathsf{EQ}(\mathsf{Suf}(x,z), \mathsf{Suf}(y,z)), f_t(s_0(z') \circ \mathsf{Suf}(y,z), \vec{w}), f_t(x, \vec{w})) \\ = &\begin{cases} f_t(s_0(z') \circ \mathsf{Suf}(y,z), \vec{w}) & \mathsf{EQ}(\mathsf{Suf}(x,z), \mathsf{Suf}(y,z))) = 1; \\ f_t(x, \vec{w}) & \text{otherwise}. \end{cases} \end{aligned}$$

Since $z' \circ 0 = s_0(z')$, we can prove that they are identical by a case study on ITE using Theorem 3.6. This completes the proof. □

4.6 Admissibility of Logical Rules

Now we will prove the admissibility of logical rules, including the assumption axiom (V), the substitution rule (V), and the introduction and elimination rules for logical connectives $\{\to, \bot\}$.

4.6.1 Assumption and Substitution

Lemma 4.26. *The assumption rule* (A) *is admissible. That is, for any* Γ *and formula* α, $\mathsf{PV} \vdash [\Gamma, \alpha \vdash \alpha]_{\mathsf{PV}}$.

Proof Sketch. Again, it suffices to prove the admissibility of $\alpha \vdash \alpha$, i.e., $\mathsf{PV} \vdash [\alpha \vdash \alpha]_{\mathsf{PV}}$. Notice that $[\alpha \vdash \alpha]_{\mathsf{PV}} \equiv$ "$[\alpha]_{\mathsf{PV}} \Rightarrow [\alpha]_{\mathsf{PV}} = 1$", which is provable in PV by the fact that

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow (\mathsf{TR}(x) = \varepsilon) \Rightarrow x \Rightarrow x = 1$$

and that $\mathsf{PV} \vdash \mathsf{IsNotEps}([\alpha]_{\mathsf{PV}})$ (see Proposition 4.2) and $\mathsf{PV} \vdash \mathsf{TR}([\alpha]_{\mathsf{PV}}) = \varepsilon$ (see Proposition 4.3) using Modus Ponens (see Proposition 3.10). □

Lemma 4.27. *The substitution rule* (V) *is admissible. That is, for any assertion* $\Gamma \vdash \alpha$ *such that* $\mathsf{PV} \vdash [\Gamma \vdash \alpha]_{\mathsf{PV}}$, *we have* $\mathsf{PV} \vdash [\Gamma[z/t] \vdash \alpha[z/t]]_{\mathsf{PV}}$ *for any term* t.

Proof Sketch. It suffices to prove that substitution commutes with the PV translation, i.e., $[\Gamma[z/t] \vdash \alpha[z/t]]_{\mathsf{PV}} \equiv [\Gamma \vdash \alpha]_{\mathsf{PV}}[z/t]$, so that the lemma follows directly from the substitution rule (L4) in PV.

We first prove that substitution commutes with conditional equations, i.e., $(s_c \Rightarrow s_1 = s_2)[z/t] \equiv s_c[z/t] \Rightarrow s_1[z/t] = s_2[z/t]$, which follows directly from the definition of conditional equations. Let $\Gamma = (\beta_1, \ldots, \beta_n)$. Notice that

$$\begin{aligned} &[\Gamma \vdash \alpha]_{\mathsf{PV}}[z/t] \\ \equiv\ &[\beta_n]_{\mathsf{PV}}[z/t] \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}}[z/t] \Rightarrow [\alpha]_{\mathsf{PV}}[z/t] = 1 \\ \equiv\ &[\Gamma[z/t] \vdash \alpha[z/t]]_{\mathsf{PV}}, \end{aligned}$$

where the first equation follows from applying the fact that substitution commutes with conditional equations for n times, and second equation froms from the definition of the translation. □

4.6.2 Rules of Implication

Next, we show that the introduction ($\to_i$) and elimination ($\to_e$) rules of implication are PV admissible.

Lemma 4.28. *The introduction rule of implication ($\to_i$) is admissible. That is, for any Γ and formulas φ, ψ, suppose that $\mathsf{PV} \vdash [\Gamma, \varphi \vdash \psi]_{\mathsf{PV}}$, then $\mathsf{PV} \vdash [\Gamma \vdash \varphi \to \psi]_{\mathsf{PV}}$.*

Proof Sketch. By the admissibility of the permutation rule, it suffices to prove that $\mathsf{PV} \vdash [\varphi, \Gamma \vdash \psi]_{\mathsf{PV}}$ implies that $\mathsf{PV} \vdash [\Gamma \vdash \varphi \to \psi]_{\mathsf{PV}}$.

Let $\Gamma = (\beta_1, \ldots, \beta_n)$, $n \in \mathbb{N}$. By the assumption and the definition of PV translation, we have that

$$\mathsf{PV} \vdash [\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} \Rightarrow [\psi]_{\mathsf{PV}} = 1. \tag{99}$$

and we need to prove that

$$\mathsf{PV} \vdash [\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow \mathsf{ITE}([\varphi]_{\mathsf{PV}}, [\psi]_{\mathsf{PV}}, 1) = 1. \tag{100}$$

Proposition 4.29. *Let $n \in \mathbb{N}$. Then PV proves*

$$\begin{aligned}
&\mathsf{IsNotEps}(x) \Rightarrow \mathsf{IsNotEps}(y) \Rightarrow \mathsf{TR}(x) = \varepsilon \Rightarrow \mathsf{TR}(y) = \varepsilon \Rightarrow \\
&\mathsf{IsNotEps}(z_n) \Rightarrow \ldots \Rightarrow \mathsf{IsNotEps}(z_1) \Rightarrow \\
&(z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow x \Rightarrow y = 1) \Rightarrow z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow \mathsf{ITE}(x, y, 1) = 1.
\end{aligned}$$

To see that this proposition suffices, notice that with the substitution $x/[\varphi]_{\mathsf{PV}}$, $y/[\psi]_{\mathsf{PV}}$, $z_i/[\beta_i]_{\mathsf{PV}}$ for $i \in [n]$ and Proposition 4.2 and 4.3, we obtain that

$$\begin{aligned}
&([\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow [\varphi]_{\mathsf{PV}} \Rightarrow [\psi]_{\mathsf{PV}} = 1) \\
&\quad \Rightarrow [\beta_n]_{\mathsf{PV}} \Rightarrow \ldots \Rightarrow [\beta_1]_{\mathsf{PV}} \Rightarrow \mathsf{ITE}([\varphi]_{\mathsf{PV}}, [\psi]_{\mathsf{PV}}, 1) = 1,
\end{aligned}$$

where the antecedent (and thus its $[\cdot]_{\mathsf{EQ}}$ translation) is provable in PV. Therefore, by Modus Ponens (see Proposition 3.10), we can obtain Equation (100).

Proof of Proposition 4.29. It remains to prove Proposition 4.29. Similar to the proof of Proposition 4.14, we will prove the conditional equation by an induction on n (in the meta-theory).

Consider the case for $n = 0$. That is:

$$\begin{aligned}
&\mathsf{IsNotEps}(x) \Rightarrow \mathsf{IsNotEps}(y) \Rightarrow \mathsf{TR}(x) = \varepsilon \Rightarrow \mathsf{TR}(y) = \varepsilon \Rightarrow \\
&x \Rightarrow y = 1 \Rightarrow \mathsf{ITE}(x, y, 1) = 1.
\end{aligned}$$

To prove this, we performs a case analysis on x and y using Theorem 3.5, where all cases are straightforward.

It remains to consider the induction case. That is, assuming that PV proves

$$\begin{aligned}&\mathsf{IsNotEps}(x) \Rightarrow \mathsf{IsNotEps}(y) \Rightarrow \mathsf{TR}(x) = \varepsilon \Rightarrow \mathsf{TR}(y) = \varepsilon \Rightarrow \\ &\mathsf{IsNotEps}(z_n) \Rightarrow \ldots \Rightarrow \mathsf{IsNotEps}(z_1) \Rightarrow \\ &(z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow x \Rightarrow y = 1) \Rightarrow z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow \mathsf{ITE}(x, y, 1) = 1,\end{aligned}$$

we will show that PV proves

$$\begin{aligned}&\mathsf{IsNotEps}(x) \Rightarrow \mathsf{IsNotEps}(y) \Rightarrow \mathsf{TR}(x) = \varepsilon \Rightarrow \mathsf{TR}(y) = \varepsilon \Rightarrow \\ &\mathsf{IsNotEps}(z_{n+1}) \Rightarrow \ldots \Rightarrow \mathsf{IsNotEps}(z_1) \Rightarrow \\ &(z_{n+1} \Rightarrow \ldots \Rightarrow z_1 \Rightarrow x \Rightarrow y = 1) \Rightarrow z_{n+1} \Rightarrow \ldots \Rightarrow z_1 \Rightarrow \mathsf{ITE}(x, y, 1) = 1,\end{aligned}$$

We perform a case analysis on x and y twice (i.e. we consider what is the last and the second last bits of x and y). All cases are trivial except for the case $x/1$ and $y/0$, in which it suffices to prove that

$$\begin{aligned}&\mathsf{IsNotEps}(z_{n+1}) \Rightarrow \ldots \Rightarrow \mathsf{IsNotEps}(z_1) \Rightarrow \\ &(z_{n+1} \Rightarrow \ldots \Rightarrow z_1 \Rightarrow 1 \Rightarrow 0 = 1) \Rightarrow z_{n+1} \Rightarrow \ldots \Rightarrow z_1 \Rightarrow 0 = 1. \qquad (101)\end{aligned}$$

We perform a case study on z_{n+1}. The only non-trivial case is that $z_{n+1}/s_1(z_{n+1})$,

Let e be the conditional equation $z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow 0 = 1$. With the substitution $z_{n+1}/s_1(z_{n+1})$, Equation (101) PV-provably evaluates to

$$\begin{aligned}&\mathsf{IsNotEps}(z_n) \Rightarrow \ldots \Rightarrow \mathsf{IsNotEps}(z_1) \Rightarrow \\ &(z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow 1 \Rightarrow 0 = 1) \Rightarrow z_n \Rightarrow \ldots \Rightarrow z_1 \Rightarrow 0 = 1, \qquad (102)\end{aligned}$$

which follows from the assumption (with the same substitution $x/1$ and $y/0$). This completes the proof. □

Lemma 4.30. *The elimination rule of implication ($\to_e$) is admissible. That is, for any Γ and formula φ, ψ, suppose that $\mathsf{PV} \vdash [\Gamma \vdash \varphi \to \psi]_{\mathsf{PV}}$ and $\mathsf{PV} \vdash [\Gamma \vdash \varphi]$, we have that $\mathsf{PV} \vdash [\Gamma \vdash \psi]$.*

Proof Sketch. We can deal with the antecedent Γ similar to the proof of the admissibility of $(\to_i)$; for simplicity of presentation, we will only demonstrate the admissibility in case that $\Gamma = \varnothing$. By the definition of the PV translation, we will need to derive $\mathsf{PV} \vdash [\psi]_{\mathsf{PV}}$ from $\mathsf{PV} \vdash \mathsf{ITE}([\varphi]_{\mathsf{PV}}, [\psi]_{\mathsf{PV}}, 1) = 1$ and $\mathsf{PV} \vdash [\varphi]_{\mathsf{PV}} = 1$. Therefore, it suffices to prove that:

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(y) \Rightarrow (\mathsf{TR}(y) = \varepsilon) \Rightarrow (x = 1) \Rightarrow (\mathsf{ITE}(x, y, 1) = 1) \Rightarrow y = 1. \quad (103)$$

To see that this suffices, notice that we can substitute $x/[\varphi]_{\mathsf{PV}}$, $y/[\psi]_{\mathsf{PV}}$, and apply Modus Ponens (see Proposition 3.10).

To prove Equation (103), we perform a case analysis on x and y using Theorem 3.5, and all cases can be proved directly by unfolding with the Explosion Rule (see Proposition 3.11) and the trick of adding dummy conditions (see Proposition 4.6). □

4.6.3 Rules of Contradiction

The last logical rule we need to consider is the elimination rule of $\bot$, i.e., "proof by contradiction".

Lemma 4.31. *The elimination rule* $(\bot_e)$ *of* $\bot$ *is admissible. That is, for any* Γ *and any formula* α*, suppose that* $\mathsf{PV} \vdash [\Gamma, \neg\alpha \vdash \bot]_{\mathsf{PV}}$*, then* $\mathsf{PV} \vdash [\Gamma \vdash \alpha]_{\mathsf{PV}}$*.*

Proof Sketch. We will prove that the axiom of double negation elimination is admissible in PV, that is:

$$\mathsf{PV} \vdash [\Gamma \vdash \neg\neg\alpha \to \alpha]_{\mathsf{PV}}.$$

To see that this suffices, notice that the elimination rule $(\bot_e)$ can be simulated by following proof tree using double negation elimination using the introduction and elimination rules of $\to$ that have already proved to be admissible:

$$\dfrac{\dfrac{}{\Gamma \vdash \neg\neg\alpha \to \alpha} \quad \dfrac{\Gamma, \neg\alpha \vdash \bot}{\Gamma \vdash \neg\neg\alpha}\,(\to_i)}{\Gamma \vdash \alpha}\,(\to_e)$$

Moreover, by the admissibility of weakening (W), it suffices to prove that admissibility of the case $\Gamma = \varnothing$. That is,

$$\begin{aligned} &\mathsf{PV} \vdash [\vdash \neg\neg\alpha \to \alpha]_{\mathsf{PV}} \\ \iff &\mathsf{PV} \vdash \mathsf{ITE}(\mathsf{ITE}(\mathsf{ITE}([\alpha]_{\mathsf{PV}}, 0, 1), 0, 1), [\alpha]_{\mathsf{PV}}, 1) = 1. \end{aligned}$$

Furthermore, it suffices to prove the more general version:

$$\mathsf{PV} \vdash \mathsf{IsNotEps}(x) \Rightarrow \mathsf{ITE}(\mathsf{ITE}(\mathsf{ITE}(x, 0, 1), 0, 1), x, 1) = 1,$$

which can be proved by a case study on x using Theorem 3.5. □

4.7 Admissibility of Structural Induction

Finally, we prove that the structural induction rule is admissible in PV.

I guess there is no need to "recall", dear fellow feasible mathematicians :)

Recall that in the definition of PV, we only provide a restricted version of structural induction in the sense that if two functions are both identical to a recursively defined feasible function, we can conclude that the two functions are identical. The induction rule in PV-PL, however, captures the generic structural induction principle as discussed in Postulate 3. Therefore, the feasibility of the induction rule in PV-PL shows that the restricted structural induction rule in PV suffices to capture the informal notion of structural induction.

Remark 4.2. For simplicity, we will only prove the feasibility of (Ind_2); nevertheless, the admissibility of (Ind_n) clearly follows from the same technique.

Lemma 4.32. *The structural induction rule* (Ind_2) *is admissible. That is, for any terms* t_1, t_2*,* Γ*, and variables* x_1, x_2*, suppose that* PV *proves*

- $[\Gamma \vdash t_1[x_j/\varepsilon] = t_2[x_j/\varepsilon]]_{\mathsf{PV}}$*, where* $j \in \{1, 2\}$*.*
- $[\Gamma, t_1 = t_2 \vdash t_1[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)] = t_2[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)]]_{\mathsf{PV}}$*, where* $i_1, i_2 \in \{0, 1\}$*.*

Then $\mathsf{PV} \vdash [\Gamma \vdash t_1 = t_2]_{\mathsf{PV}}$*.*

Proof. By the admissibility of the permutation rule, we can change the second bullet to $[t_1 = t_2, \Gamma \vdash t_1[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)] = t_2[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)]]_{\mathsf{PV}}$. Moreover, we can (without loss of generality) replace t_1, t_2 by the PV functions symbols f_1, f_2 defined by $f_i(x_1, x_2, \vec{w}) = t_i$, $i \in \{1, 2\}$, where $\vec{w}$ denote other variables in t_1, t_2. For simplicity, we will ignore $\vec{w}$ in the proof and write $f_i(x_1, x_2)$.

Let $\Gamma = (\beta_1, \ldots, \beta_n)$. We will prove the admissibility in PV by an induction on n (in the meta-theory). Concretely, we will prove that for any PV terms $u_1, \ldots, u_n$, if PV proves

- $\mathsf{IsNotEps}(u_i) = 1$ for any $i \in [n]$,
- $u_n \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(\varepsilon, x_2), f_2(\varepsilon, x_2)) = 1$,
- $u_n \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, \varepsilon), f_2(x_1, \varepsilon)) = 1$,
- $u_n \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) \Rightarrow \mathsf{EQ}(f_1(s_{i_1}(x_1), s_{i_2}(x_2)), f_2(s_{i_1}(x_1), s_{i_2}(x_2))) = 1$, for any $i_1, i_2 \in \{0, 1\}$,

Then PV proves $u_n \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) = 1$. (Note that by instantiating $u_i := [\beta_i]_{\mathsf{PV}}$ for $i \in [n]$ we can derive the lemma.)

Base case. We first consider the case $n = 1$. The case for $\Gamma = \varnothing$ can be derived from this case, which is left as an exercise. Note that we need to prove that if PV proves for any PV term u,

Hint: Introduce a provable equation, e.g., $\varepsilon = \varepsilon$ in the antecedent using cut.

- $\mathsf{IsNotEps}(u) = 1$,
- $u \Rightarrow \mathsf{EQ}(f_1(\varepsilon, x_2), f_2(\varepsilon, x_2)) = 1$,
- $u \Rightarrow \mathsf{EQ}(f_1(x_1, \varepsilon), f_2(x_1, \varepsilon)) = 1$,
- $u \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) \Rightarrow \mathsf{EQ}(f_1(s_{i_1}(x_1), s_{i_2}(x_2)), f_2(s_{i_1}(x_1), s_{i_2}(x_2))) = 1$, for any $i_1, i_2 \in \{0, 1\}$,

then $\mathsf{PV} \vdash u \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) = 1$. That is:

$$\mathsf{ITE}(u, 1, \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2))) = \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)). \tag{104}$$

We will prove the alternative equation:

$$\mathsf{ITE}(u, \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)), 1) = 1 \tag{105}$$

in PV, and it is left as an exercise to prove that this suffices.

Hint: Prove a more general conditional equation and apply Modus Ponens.

We will prove the equation by simultaneous induction on x_1 and x_2 using Theorem 3.21. Let $g_{00} = g_{01}(x) = g_{10}(y) = 1$, and $h_{i_1 i_2}(x, y, z) = \mathsf{ITE}(z, 1, 0)$ for $i_1, i_2 \in \{0, 1\}$. It is easy to see that the the RHS of $(\star)$ is identical to the function defined recursively from $g_{j_1 j_2}$ and $h_{i_1 i_2}$. Moreover, we can easily see that PV proves:

- $\mathsf{ITE}(u, \mathsf{EQ}(f_1(\varepsilon, \varepsilon), f_2(\varepsilon, \varepsilon)), 1) = g_{00} = 1$,
- $\mathsf{ITE}(u, \mathsf{EQ}(f_1(\varepsilon, s_j(x_2)), f_2(\varepsilon, s_j(x_2))), 1) = g_{01}(s_j(x_2)) = 1$,
- $\mathsf{ITE}(u, \mathsf{EQ}(f_1(s_j(x_1), \varepsilon), f_2(s_j(x_1), \varepsilon)), 1) = g_{01}(s_j(x_1)) = 1$,

from the assumption. Therefore, it suffices to prove that

$$\begin{aligned} &\mathsf{ITE}(u, \mathsf{EQ}(f_1(s_{i_1}(x_1), s_{i_2}(x_2)), f_2(s_{i_1}(x_1), s_{i_2}(x_2))), 1) \\ = \; &\mathsf{ITE}(\mathsf{ITE}(u, \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)), 1), 1, 0). \end{aligned} \tag{106}$$

We will first prove a more general result:

Proposition 4.33. $\mathsf{PV} \vdash (x \Rightarrow y \Rightarrow z = 1) \Rightarrow \mathsf{ITE}(x, z, 1) = \mathsf{ITE}(\mathsf{ITE}(x, y, 1), 1, 0)$.

This can be proved by a simple case study on x, y, and z using Theorem 3.5. Finally, with the substitution

$$\begin{aligned}&x/u\\&y/\mathsf{EQ}(f_1(x_1,x_2),f_2(x_1,x_2))\\&z/\mathsf{EQ}(f_1(s_{i_1}(x_1),s_{i_2}(x_2)),f_2(s_{i_1}(x_1),s_{i_2}(x_2)))\end{aligned}$$

we can see that the antecedent of Proposition 4.33 is exactly the second bullet of the assumption, and the consequence is exactly Equation (106) that we need to prove. This concludes the base case by Modus Ponens (see Proposition 3.10).

Induction case. Our strategy is to "merge" u_{n+1} and u_n in the antecedents of the conditional equation so that it reduces to the case for $|\Gamma| = n$. Concretely, we will prove the following meta-theorem of PV:

Proposition 4.34. *Let t_1, t_2, t_c, t_c' be terms such that $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c) = 1$ and $\mathsf{PV} \vdash \mathsf{IsNotEps}(t_c') = 1$. Then $\mathsf{PV} \vdash t_c \Rightarrow t_c' \Rightarrow t_1 = t_2$ if and only if $\mathsf{PV} \vdash \mathsf{And}(t_c, t_c') \Rightarrow t_1 = t_2$.*

To see that this suffices, notice that by applying the meta-theorem on both the premise and the conclusion of the rule, it suffices to derive from

$$\begin{aligned}&\mathsf{PV} \vdash \mathsf{IsNotEps}(u_i) = 1 \quad (i \in [n-1])\\&\mathsf{PV} \vdash \mathsf{IsNotEps}(\mathsf{And}(u_{n+1}, u_n)) = 1\\&\mathsf{PV} \vdash \mathsf{And}(u_{n+1}, u_n) \Rightarrow u_{n-1} \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(\varepsilon, x_2), f_2(\varepsilon, x_2)) = 1\\&\mathsf{PV} \vdash \mathsf{And}(u_{n+1}, u_n) \Rightarrow u_{n-1} \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, \varepsilon), f_2(x_1, \varepsilon)) = 1\\&\mathsf{PV} \vdash \mathsf{And}(u_{n+1}, u_n) \Rightarrow u_{n-1} \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) \Rightarrow\\&\qquad \mathsf{EQ}(f_1(s_{i_1}(x_1), s_{i_2}(x_2)), f_2(s_{i_1}(x_1), s_{i_2}(x_2))) = 1 \quad (i_1, i_2 \in \{0,1\})\end{aligned}$$

that $\mathsf{PV} \vdash \mathsf{And}(u_{n+1}, u_n) \Rightarrow u_{n-1} \Rightarrow \ldots \Rightarrow u_1 \Rightarrow \mathsf{EQ}(f_1(x_1, x_2), f_2(x_1, x_2)) = 1$, which is true by the induction hypothesis and the fact that if $\mathsf{PV} \vdash \mathsf{IsNotEps}(u) = 1$ and $\mathsf{PV} \vdash \mathsf{IsNotEps}(u') = 1$, then $\mathsf{PV} \vdash \mathsf{IsNotEps}(\mathsf{And}(u, u')) = 1$.

It remains to prove Proposition 4.34. We will only prove the ($\Rightarrow$) directly, and the converse can be proved using the same approach. Consider the following more general version:

Proposition 4.35. $\mathsf{PV} \vdash \mathsf{IsNotEps}(z_1) \Rightarrow \mathsf{IsNotEps}(z_2) \Rightarrow (z_1 \Rightarrow z_2 \Rightarrow x = y) \Rightarrow \mathsf{And}(z_1, z_2) \Rightarrow x = y$.

This can be proved by a case study on z_1 and z_2 using Theorem 3.5. The cases for z_1/ε, z_2/ε, $z_1/s_0(z_1)$, and $z_2/s_0(z_2)$ can be proved by the Explosion Rule (see Proposition 3.11). For the case $z_1/s_1(z_1)$ and $z_2/s_1(z_2)$, we can unfold all the definitions and apply the correctness of EQ (see Lemma 3.25).

To see that this implies the ($\Rightarrow$) direction of Proposition 4.34, we can substitute z_1/t_c, z_2/t_c', x/t_1, y/t_2. We can remove the outermost three antecedents using the assumption and Modus Ponens (see Proposition 3.10), which derives $\mathsf{And}(t_c, t_c') \Rightarrow t_1 = t_2$ and completes the proof. □

4.8 Extensions of Rules

Now we show a few extensions of the rules in PV-PL that may be helpful in formalizing informal mathematical proofs.

Rewrite of formulas. We will also consider a stronger form of $(=_/)$, denoted by $(\hat{=}_/)$, together with its converse $(\hat{=}_/^{-1})$. Both of the rules will be referred as "rewrite rules" as it can be used to rewrite a formula based on an equation in the antecedent.

Let $\Gamma, x = y \vdash \varphi$ be an assertion. Suppose that φ' is obtained from φ by replacing *some* occurrences of x to y, then the axioms $(\hat{=}_/)$ and its converse $(\hat{=}_/^{-1})$ are defined as

$$(\hat{=}_/): \quad \frac{}{\Gamma, x = y, \varphi \vdash \varphi'}; \quad (\hat{=}_/^{-1}): \quad \frac{}{\Gamma, x = y, \varphi' \vdash \varphi}.$$

We call φ the substitution formula of the rule.

Note that $(\hat{=}_/)$ and its converse simulate each other by the cut rule and the symmetricity of equality; the following proof tree simulates $(\hat{=}_/^{-1})$ using $(\hat{=}_/)$:

$$\dfrac{\dfrac{}{\Gamma, x = y \vdash y = x}\,(=_s) \quad \dfrac{}{\Gamma, x = y, y = x, \varphi' \vdash \varphi}\,(\hat{=}_/)}{\Gamma, x = y, \varphi' \vdash \varphi}\,(\text{Cut})$$

(All structural rules are omitted.)

Lemma 4.36. *$(\hat{=}_/)$ and $(\hat{=}_/^{-1})$ are admissible in* PV-PL.

Proof Sketch. We perform an induction on the depth of the substitution formula φ.

Base case. Suppose that φ is atomic, i.e., it is either $\bot$ or a PV equation $u = v$. Note that the case for $\varphi \equiv \bot$ is trivial by the assumption rule (A). Consider the case for $\varphi \equiv$ "$u = v$" and $\varphi' \equiv$ "$u' = v'$", and it suffices to prove the admissibility of $(\hat{=}_/)$ as $(\hat{=}_/^{-1})$ can be simulated by $(\hat{=}_/)$ with substitution formula of the same depth. Let z be a fresh variable and $\hat{u}$ and $\hat{v}$ be the terms satisfying that

$$\hat{u}[z/y] = u', \hat{u}[z/x] = u, \ \hat{v}[z/y] = v', \hat{v}[z/x] = v.$$

Notice that we can prove $\Gamma, x = y, u = v \vdash u = u'$ by the proof tree (where all structural rules are omitted):

$$\frac{}{\Gamma, x = y, u = v \vdash \hat{u}[z/x] = \hat{u}[z/y]}\,(=_/)$$

Similarly, we can prove that $\Gamma, x = y, u = v \vdash v = v'$. Therefore by the cut rule and the symmetricity and transitivity of equality, we can prove $\Gamma, x = y, u = v \vdash u' = v'$.

Induction case. Suppose that φ is of form $\beta \to \alpha$ and $\varphi' \equiv \beta' \to \alpha'$. By the induction hypothesis, we know that both $(\hat{=}_/)$ and $(\hat{=}_/^{-1})$ are admissible if the substitution formula is α, α', β or β'. It suffices to prove the admissibility of $(\hat{=}_s)$ with φ being the substitution formula, i.e.,

$$\Gamma, x = y, \beta \to \alpha \vdash \beta' \to \alpha'.$$

By the introduction rule of implication, it suffices to prove that $\Gamma, x = y, \beta \to \alpha, \beta' \vdash \alpha'$. We then apply the cut rule with β being the cut formula, so that it generates two subgoals to resolve:

(*Subgoal* 1): $\Gamma, x = y, \beta \to \alpha, \beta' \vdash \beta$. This is provable by the induction hypothesis, as we can apply $(\hat{=}_/^{-1})$ with β' being the substitution formula.

(*Subgoal* 2): $\Gamma, x = y, \beta \to \alpha, \beta', \beta \vdash \alpha'$. We apply the cut rule again with α being the cut formula, so that it further generates two subgoals:

(*Subgoal 2.1*): $\Gamma, x = y, \beta \to \alpha, \beta', \beta \vdash \alpha$. Notice that this can be proved by Modus Ponens on two assumptions, which can be implemented by the elimination rule of $\to$:

This is usually called the `apply` tactic in proof assistants.

$$\dfrac{\dfrac{}{\Gamma, x = y, \beta \to \alpha, \beta', \beta \vdash \beta \to \alpha}\,(\mathrm{A}) \quad \dfrac{}{\Gamma, x = y, \beta \to \alpha, \beta', \beta \vdash \beta}\,(\mathrm{A})}{\Gamma, x = y, \beta \to \alpha, \beta', \beta \vdash \alpha}\,(\to_e)$$

(*Subgoal 2.2*): $\Gamma, x = y, \beta \to \alpha, \beta', \beta, \alpha \vdash \alpha'$. This is provable by the induction hypothesis, as we can apply $(\hat{=}_/)$ with α being the substitution formula.

This completes all subgoals and leads to the lemma. □

Comprehension of PV translation. Recall that the PV translation could translate a formula φ in PV-PL into a PV term $[\varphi]_{\mathsf{PV}}$. We now show that we can prove the equivalence of a formula and its PV translation in PV-PL, formalized as the following comprehension axiom (CPV) and its converse (C^{-1}PV):

$$(\mathrm{CPV}):\ \frac{}{\Gamma, \varphi \vdash [\varphi]_{\mathsf{PV}} = 1};\quad (\mathrm{C}^{-1}\mathrm{PV}):\ \frac{}{\Gamma, [\varphi]_{\mathsf{PV}} = 1 \vdash \varphi}.$$

Lemma 4.37. *Both* (CPV) *and* (C^{-1}PV) *are admissible in* PV-PL.

Proof Sketch. We prove this by induction on the depth of the formula φ in the comprehension axiom. The case for $\varphi = \bot$ is trivial. Suppose that φ is of form $u = v$, then (CPV) follows from the rewrite rule as well as $\mathsf{PV} \vdash \mathsf{EQ}(x, x) = 1$, while ($\mathrm{C}^{-1}$PV) follows from the correctness of EQ as demonstrated in Example 4.2.

Now we consider the case that φ is of form $\beta \to \alpha$, so that it suffices to prove in PV-PL that:

- $\Gamma, \beta \to \alpha \vdash \mathsf{ITE}([\beta]_{\mathsf{PV}}, [\alpha]_{\mathsf{PV}}, 1) = 1$,
- $\Gamma, \mathsf{ITE}([\beta]_{\mathsf{PV}}, [\alpha]_{\mathsf{PV}}, 1) = 1 \vdash \beta \to \alpha$.

The proofs of these two assertions are similar, so we only explains the first bullet.

By the cut rule, we first prove that $\Gamma, \beta \to \alpha \vdash [\beta]_{\mathsf{PV}} = 1 \to [\alpha]_{\mathsf{PV}} = 1$, which reduces to $\Gamma, \beta \to \alpha, [\beta]_{\mathsf{PV}} = 1 \vdash [\alpha]_{\mathsf{PV}} = 1$. By the induction hypothesis, we know that $\Gamma, \beta \to \alpha, [\beta]_{\mathsf{PV}} = 1 \vdash \beta$ and subsequently

$$\Gamma, \beta \to \alpha, [\beta]_{\mathsf{PV}} = 1 \vdash \alpha.$$

Using the cut rule it suffices to prove that $\Gamma, \beta \to \alpha, \alpha, [\beta]_{\mathsf{PV}} = 1 \vdash [\alpha]_{\mathsf{PV}}$, which again follows from the induction hypothesis. □

Induction on formulas. One primary application of the comprehension axioms is to generalize the induction principle (Ind_n) from equations to formulas. Suppose that φ is a formula and z is a variable, we have

$$(\mathrm{Ind}_1^{\varphi}):\ \frac{\Gamma \vdash \varphi[z/\varepsilon] \quad \Gamma, \varphi \vdash \varphi[z/s_0(z)] \quad \Gamma, \varphi \vdash \varphi[z/s_1(z)]}{\Gamma \vdash \varphi}$$

Lemma 4.38. (Ind_1^{φ}) *is admissible in* PV-PL.

Proof Sketch. By the cut rule and the comprehension axiom, it suffices to prove that $\Gamma \vdash [\varphi]_{\mathsf{PV}} = 1$ from the premises. By the induction rule (Ind_1) in PV-PL, it suffices to prove that

- $\Gamma \vdash [\varphi]_{\mathsf{PV}}[z/\varepsilon] = 1$
- $\Gamma, [\varphi]_{\mathsf{PV}} = 1 \vdash [\varphi]_{\mathsf{PV}}[z/s_i(z)] = 1$ for $i \in \{0, 1\}$.

It is easy to see that substitution commutes with the PV translation, and therefore it is equivalent to prove that

- $\Gamma \vdash [\varphi[z/\varepsilon]]_{\mathsf{PV}} = 1$,
- $\Gamma, [\varphi]_{\mathsf{PV}} = 1 \vdash [\varphi[z/s_i(z)]]_{\mathsf{PV}} = 1$ for $i \in \{0, 1\}$,

which can be proved from the premises using the comprehension axioms. □

Remark 4.3. Though we only wrote down the induction principle with one variable for simplicity of presentation, the same proof clearly generalizes to the case for multiple variables.

5 "Advanced" Data Structures

With the stronger reasoning system PV-PL and the translation theorem (see Theorem 4.4), we continue the investigation of the programming language built from Cook's theory PV. Recall that we have already demonstrated the definition of pairs, tuples, and natural numbers (in dyadic notation).

In this section, we will develop (slightly) more advanced data structure such as lists and maps with PV (or PV-PL) provable correctness. The constructions will serve as an evidence towards the Feasible Mathematics Thesis, as these data structures will significantly extend our programming ability (as well as the ability of reasoning about programs).

Throughout this section, we will write "pseudo-proofs", i.e., goal-targeted mathematical proofs in natural language that could be directly translated to a proof tree in PV-PL, and subsequently a proof in PV by the translation theorem.

5.1 Lists

Next we implement *lists*, i.e., a sequence of unbounded length. List is arguably the most important data structure in *functional programming* and will greatly simplify the formalization of algorithms in PV.

5.1.1 Definition of Lists

We first implement basic functions to form and unfold a list: $\mathsf{Append}(\ell, x)$ (also denoted by $\ell :: x$) appends a string x to the end of a list ℓ, $\mathsf{Head}(\ell)$, $\mathsf{Tail}(\ell)$ and proves in PV that

Here we encode the empty list using ε; notice that the empty list ε is different from a single-element list consisting of ε (i.e. $\varepsilon :: \varepsilon$).

$$\mathsf{Head}(\varepsilon) = \varepsilon, \qquad \mathsf{Head}(\ell :: x) = x \tag{107}$$

$$\mathsf{Tail}(\varepsilon) = \varepsilon, \qquad \mathsf{Tail}(\ell :: x) = \ell \tag{108}$$

Also, we will prove that the description length of a list does not grow super-polynomially so that we can create a list of polynomial length. Formally:

$$\mathsf{PV} \vdash \mathsf{ITR}(\ell :: x, \ell \circ 11 \circ \mathsf{PEnc}(x)) = \varepsilon. \tag{109}$$

(We will later see how this equation will be useful.)

Indeed the construction is similar to tuples. We define:

$$\mathsf{Append}(\ell, x) := \mathsf{MakePair}(\ell, x),\ \mathsf{Head}(\ell) = \mathsf{Right}(\ell),\ \mathsf{Tail}(\ell) = \mathsf{Left}(\ell).$$

The proof of Equation (107) and (108) naturally follows from Lemma 3.16. Therefore, it suffices to verify Equation (109):

Lemma 5.1. $\mathsf{PV} \vdash \mathsf{ITR}(\ell :: x, \ell \circ (11 \circ \mathsf{PEnc}(x))) = \varepsilon$.

Proof. By unfolding the definition of Append (i.e. $\ell :: x$), subsequently unfolding $\mathsf{MakePair}$, and applying the associativity of concatenation (see Proposition 2.4), it suffices to prove that

$$\mathsf{PV} \vdash \mathsf{ITR}(\ell \circ 11 \circ \mathsf{PEnc}(x), \ell \circ 11 \circ \mathsf{PEnc}(x)) = \varepsilon. \tag{110}$$

This follows from $\mathsf{PV} \vdash \mathsf{ITR}(x, x) = \varepsilon$ using (L4) substitution, and $\mathsf{ITR}(x, x) = \varepsilon$ can be proved in PV using Proposition 3.1 and Proposition 2.3. □

Generating a list. As an example, we will define a function $\mathsf{GenList}(v, y)$ that generates a list of length $|y|$ consists of v. Formally, it will satisfy that

$$\mathsf{PV} \vdash \mathsf{GenList}(v, \varepsilon) = \varepsilon \tag{111}$$
$$\mathsf{PV} \vdash \mathsf{GenList}(v, s_i(y)) = \mathsf{GenList}(v, y) :: v \tag{112}$$

Quite naturally, we will define the function using limited recursion on the variable y. Let $g(v) = \varepsilon$, $h_i(v, y, z) = z :: v$, and $k_i(v, y) = 11 \circ \mathsf{PEnc}(v)$, we can verify that

Proposition 5.2. $\mathsf{PV} \vdash \mathsf{ITR}(h_i(v, y, z), z \circ k_i(v, y)) = \varepsilon$ *for* $i \in \{0, 1\}$.

Proof. Unfolding h_i, k_i, we will notice that this is exactly Lemma 5.1. □

Therefore, we can define a function f from limited recursion rule using the functions (g, h_0, h_1, k_0, k_1). It is easy to verify that if we define $\mathsf{GenList}(v, y) := f(v, y)$, it will satisfy Equation (111) and (112).

We also note that by composing $\mathsf{GenList}$ and $\#$, we can initiate a list of arbitrary polynomial length. For instance, $\mathsf{GenList}(v, y \# y \# y)$ will generate a list of length $|y|^3$ consisting of v.

5.1.2 Recursion on Lists

Now we state a meta-theorem that allows to recursively define functions over lists. Using this tool, we will be able to define useful functions over lists, e.g., the length of a list and concatenation of lists.

Theorem 5.3 (Recursion on Lists)**.** *Let* $g(\vec{x})$, $h(\vec{x}, u, \ell, z)$, $k(\vec{x}, u, \ell)$ *be* PV *functions satisfying that*

$$\mathsf{PV} \vdash \mathsf{ITR}(h(\vec{x}, u, \ell, z), z \circ k(\vec{x}, u, \ell)) = 0.$$

Then there is a PV *function* $f(\vec{x}, \ell)$ *such that* PV *proves:*

$$f(\vec{x}, \varepsilon) = g(\vec{x}) \tag{113}$$
$$f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f(\vec{x}, \ell)) \tag{114}$$

We defer the proof of the theorem to the end of this section as it is quite technical. Here we will first define some functions on lists using this meta-theorem. Note that the existence of the length upper bound k in all the examples below is obvious, and is left as an exercise.

Type-checking. We can define a function using Theorem 5.3 that performs type-checking on lists. Let $g = 1$ and

$$h(u, \ell, z) = \mathsf{And}(\mathsf{EQ}(\ell, \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)), z),$$

we can apply Theorem 5.3 to define a function $\mathsf{IsList}'(\ell)$ such that

$$\mathsf{PV} \vdash \mathsf{IsList}'(\varepsilon) = 1 \tag{115}$$
$$\mathsf{PV} \vdash \mathsf{IsList}'(\ell :: u) = \mathsf{And}(\mathsf{Or}(\mathsf{IsEps}(\ell), \mathsf{EQ}(\ell, \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell))), \mathsf{IsList}'(\ell)). \tag{116}$$

We can define $\mathsf{IsList}(\ell) = \mathsf{And}(\mathsf{Or}(\mathsf{IsEps}(\ell), \mathsf{EQ}(\ell, \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell))), \mathsf{IsList}'(\ell))$.

Proposition 5.4. PV-PL *proves* $\vdash \mathsf{IsList}(\varepsilon) = 1$ *and* $\mathsf{IsList}(\ell) = 1 \vdash \mathsf{IsList}(\ell :: u) = 1$.

Proof Sketch. Both assertion can be proved by simple unfolding. □

Length. Similarly, we can define a function using Theorem 5.3 that outputs the length of a list in unary. Concretely:

$$\mathsf{PV} \vdash \mathsf{Len}(\varepsilon) = 0 \tag{117}$$
$$\mathsf{PV} \vdash \mathsf{Len}(\ell :: u) = s_0(\mathsf{Len}(\ell)) \tag{118}$$

Concatenation. We can also define the concatenation of two lists ℓ_1 and ℓ_2 by recursion on ℓ_2. Concretely:

$$\mathsf{PV} \vdash \mathsf{Concat}(\ell_1, \varepsilon) = \ell_1 \tag{119}$$
$$\mathsf{PV} \vdash \mathsf{Concat}(\ell_1, \ell_2 :: u) = \mathsf{Concat}(\ell_1, \ell_2) :: u \tag{120}$$

For simplicity, we will use $\ell_1 \mathbin{+\!\!+} \ell_2$ to denote $\mathsf{Concat}(\ell_1, \ell_2)$.

Map. We can define a *functional* Map_f that produces a function given any PV function symbol $f(x, \vec{w})$. Intuitively, $\mathsf{Map}_f(\ell)$ will apply the function f to each element in ℓ and return the new list. That is:

Functional should be considered as a notation in the meta-theory, rather than a symbol in PV.

$$\mathsf{PV} \vdash \mathsf{Map}_f(\varepsilon) = \varepsilon$$
$$\mathsf{PV} \vdash \mathsf{Map}_f(\ell :: u) = \mathsf{Map}_f(\ell) :: f(u, \vec{w})$$

Find. Another functional that will be particularly important in our development of other data structures is to find the rightmost element in a list satisfying a given property. Let $f(x, \vec{w})$ be a PV function symbol, $\mathsf{Find}_f(\ell)$ will satisfy that

$$\mathsf{PV} \vdash \mathsf{Find}_f(\varepsilon) = \varepsilon$$
$$\mathsf{PV} \vdash \mathsf{Find}_f(\ell :: u) = \mathsf{ITE}(f(u, \vec{w}), u, \mathsf{Find}_f(\ell))$$

This is certainly not a complete list of functions we can define using Theorem 5.3. The fun puzzle of defining other functions on lists is left to the readers.

5.1.3 Induction on Lists

We will then prove an induction principle on lists that as an analogy of the induction rule (Ind_n) in PV-PL. For simplicity, we will only consider induction on one list:

Theorem 5.5 (Induciton on Lists)**.** *Let u be a variable with no occurrences in Γ, t_1, t_2, and ℓ be a variable with no occurrences in Γ. The following rule is admissible in* PV-PL*:*

$$\frac{\Gamma \vdash t_1[\ell/\varepsilon] = t_2[\ell/\varepsilon] \quad \Gamma, \mathsf{IsList}(\ell) = 1, t_1 = t_2 \vdash t_1[\ell/\ell :: u] = t_2[\ell/\ell :: u]}{\Gamma, \mathsf{IsList}(\ell) = 1 \vdash t_1 = t_2}$$

The proof of the theorem is deferred to Section 5.2.

Note that a stronger version of induction is admissible, namely induction on an arbitrary PV-PL formula:

Corollary 5.6. *Let u be a variable with no occurrences in Γ, t_1, t_2, and ℓ be a variable with no occurrences in Γ. The following induction rule for lists is admissible in* PV-PL*:*

$$\frac{\Gamma \vdash \varphi[\ell/\varepsilon] \quad \Gamma, \mathsf{IsList}(\ell), \varphi \vdash \varphi[\ell/\ell :: u]}{\Gamma, \mathsf{IsList}(\ell) \vdash \varphi}$$

The proof utilizes the admissibility of the comprehension axioms (see Lemma 4.37) and is similar to that of Lemma 4.38, which is left as an exercise.

Example 5.1. We now show how to prove the transitivity of concatenation of lists using Corollary 5.6. Our goal is to prove in PV-PL that

$$\mathsf{IsList}(\ell_3) = 1 \vdash (\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} \ell_3 = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \ell_3).$$

We prove this by induction on ℓ_3 using Corollary 5.6, which generalizes two subgoals as follows.

(*Subgoal 1*). $\vdash (\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} \varepsilon = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \varepsilon)$. This can be easily proved by the definition equation of $\mathbin{+\!\!+}$.

(*Subgoal 2*). We need to prove that

$$\begin{aligned} &\mathsf{IsList}(\ell_3) = 1, (\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} \ell_3 = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \ell_3) \\ \vdash\ &(\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} (\ell_3 :: u) = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} (\ell_3 :: u)) \end{aligned}$$

By the definition equations of $\mathbin{+\!\!+}$ (see Equation (119) and (120)), we can prove that the LHS of the consequence is equal to $((\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} \ell_3) :: u$, which is subsequently equal to $(\ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \ell_3)) :: u$ by rewriting using the assumption that $(\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} \ell_3 = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \ell_3)$ (see Lemma 4.36).

Similarly, we can prove by the definition equations of $\mathbin{+\!\!+}$ that the RHS of the consequence is equal to $\ell_1 \mathbin{+\!\!+} ((\ell_2 \mathbin{+\!\!+} \ell_3) :: u)$, and is subsequently equal to $(\ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} \ell_3)) :: u$. Therefore, we can prove

$$(\ell_1 \mathbin{+\!\!+} \ell_2) \mathbin{+\!\!+} (\ell_3 :: u) = \ell_1 \mathbin{+\!\!+} (\ell_2 \mathbin{+\!\!+} (\ell_3 :: u))$$

by the symmetricity and transitivity of equality. This completes the proof.

5.1.4 Dictionaries

Building on lists, we will implement *dictionaries*, i.e., the data structure maintaining key-value relation that can be updated. In particular, we can use an dictionary to simulate an array of length ℓ by fixing the keys to be $[\ell]$.

Definition of dictionaries. We will implement dictionary using a list containing a list of pairs maintaining key-value relation. Concretely, we define the functions $\mathsf{Lookup}(\gamma, x)$ and $\mathsf{Update}(\gamma, x, y)$ as follows:

$$\mathsf{Lookup}(\gamma, x) := \mathsf{Right}(\mathsf{Find}_f(\gamma)), \quad f(\tau, x) = \mathsf{EQ}(\mathsf{Left}(\tau), x); \tag{121}$$

$$\mathsf{Update}(\gamma, x, y) := \gamma :: \mathsf{MakePair}(x, y). \tag{122}$$

Intuitively, $\mathsf{Lookup}(\gamma, x)$ finds the value corresponding to the key x by searching for the rightmost (i.e. latest) pair of form $(x, \cdot)$, and $\mathsf{Update}(\gamma, x, y)$ adds a new key-value relation (x, y) to the dictionary γ. Note that we encode an empty dictionary using an empty list ε.

Correctness. We will prove the correctness of dictionaries formalized by the following PV-PL assertions:

$$\Gamma \vdash \mathsf{Lookup}(\varepsilon, x) = \varepsilon; \tag{123}$$
$$\Gamma, \mathsf{Lookup}(\gamma, x) = y, x' \neq x \vdash \mathsf{Lookup}(\mathsf{Update}(\gamma, x', y'), x) = y; \tag{124}$$
$$\Gamma \vdash \mathsf{Lookup}(\mathsf{Update}(\gamma, x, y), x) = y. \tag{125}$$

All these three assertions can be proved in PV-PL by simply unfolding all the functions; here, we will demonstrate the proof of Equation (124), and the rest of the proof is left as an exercise.

Proposition 5.7. PV-PL *proves Equation* (124).

Proof. By unfolding Update, the consequence of the assertion can be changed to $\mathsf{Lookup}(\gamma :: \mathsf{MakePair}(x', y')) = y$, and by further unfolding Lookup and Find_f, it suffices to prove from $\Gamma, \mathsf{Lookup}(\gamma, x) = y, x' \neq x$ that

$$\mathsf{Right}(\mathsf{ITE}(\mathsf{EQ}(\mathsf{Left}(\mathsf{MakePair}(x', y')), x), \mathsf{MakePair}(x', y'), \mathsf{Find}_f(\gamma)) = y,$$

where $f(\tau, x) := \mathsf{EQ}(\mathsf{Left}(\tau), x)$. By applying the correctness of Left and MakePair (using the cut rule), it suffices to prove (from the same antecedent) that

$$\mathsf{Right}(\mathsf{ITE}(\mathsf{EQ}(x', x), \mathsf{MakePair}(x', y'), \mathsf{Find}_f(\gamma)) = y.$$

Note that by the correctness of EQ (see, e.g., Example 4.2) in PV-PL and the property $x' \neq x$ in the antecedent, we can conclude that $\mathsf{EQ}(x', x) = 0$, and therefore by the cut rule, it suffices to prove the equation above from the antecedent $\Gamma, \mathsf{Lookup}(\gamma, x) = y, x' \neq x, \mathsf{EQ}(x', x) = 0$. Using the substitution/generalization rule, it suffices to prove

$$\mathsf{Right}(\mathsf{ITE}(z, \mathsf{MakePair}(x', y'), \mathsf{Find}_f(\gamma)) = y$$

from $\Gamma, \mathsf{Lookup}(\gamma, x) = y, x' \neq x, z = 0$, and subsequently we can change the consequence to

$$\mathsf{Right}(\mathsf{ITE}(0, \mathsf{MakePair}(x', y'), \mathsf{Find}_f(\gamma)) = y$$

using the rewriting rule (see Lemma 4.36). Therefore, by unfolding ITE, it suffices to prove that $\mathsf{Right}(\mathsf{Find}_f(\gamma)) = y$, which, by the definition equation of Lookup (see Equation (121)) is provably identical to $\mathsf{Lookup}(\gamma, x) = y$. This is available in the assumption and thus concludes the proof. □

Type-checking. Similar to lists, we can define a function $\mathsf{IsDict}(\gamma)$ that verifies whether a string correctly encodes a dictionary. Concretely, given any string γ, our algorithm first checks whether γ is a list using $\mathsf{IsList}(\gamma)$. Let $f(x)$ be defined as

$$f(x) := \mathsf{Not}(\mathsf{EQ}(\mathsf{MakePair}(\mathsf{Left}(x), \mathsf{Right}(x)), x)),$$

our algorithm further checks whether $\mathsf{IsEps}(\mathsf{Find}_f(\gamma)) = 1$. The algorithm accepts if it passes both checks.

Proposition 5.8. PV-PL *proves the following assertions:*

- $\vdash \mathsf{IsDict}(\varepsilon) = 1$.
- $\mathsf{IsDict}(\gamma) = 1 \vdash \mathsf{IsDict}(\mathsf{Update}(\gamma, x, v)) = 1$.

5.2 Proof of the Admissibility of Induction on Lists

We first recall the formal statement of the induction rule on lists:

Theorem 5.5 (Induciton on Lists)**.** *Let u be a variable with no occurrences in Γ, t_1, t_2, and ℓ be a variable with no occurrences in Γ. The following rule is admissible in* PV-PL*:*

$$\frac{\Gamma \vdash t_1[\ell/\varepsilon] = t_2[\ell/\varepsilon] \quad \Gamma, \mathsf{IsList}(\ell) = 1, t_1 = t_2 \vdash t_1[\ell/\ell :: u] = t_2[\ell/\ell :: u]}{\Gamma, \mathsf{IsList}(\ell) = 1 \vdash t_1 = t_2}$$

The proof of Theorem 5.5 is rather straightforward. Intuitively, we will prove by induction on the length of ℓ using the (Ind_1) rule in PV-PL, where the length is calculated by the function $\mathsf{Len}(\ell)$ as defined by Equation (117) and (118). Here we only sketch the proof.

Proof Sketch of Theorem 5.5. We first observe that it suffices to prove from the premises that

$$\Gamma, \mathsf{IsList}(\ell) = 1 \vdash \mathsf{Len}(\ell) = z \to t_1 = t_2$$

for a fresh variable z, as if this is possible, we can substitute $z/\mathsf{Len}(\ell)$ using the substitution rule and subsequently remove the dummy condition $\mathsf{Len}(\ell) = \mathsf{Len}(\ell)$ as it is provable by the reflexivity of equality.

To prove the assertion above, we apply (Ind_1) to the variable, which requires proofs of the following three subgoals:

- (*Subgoal 1*). $\Gamma, \mathsf{IsList}(\ell) = 1 \vdash \mathsf{Len}(\ell) = \varepsilon \to t_1 = t_2$;
- (*Subgoal 2*). $\Gamma, \mathsf{IsList}(\ell) = 1, \mathsf{Len}(\ell) = z \to t_1 = t_2 \vdash \mathsf{Len}(\ell) = s_0(z) \to t_1 = t_2$;
- (*Subgoal 3*). $\Gamma, \mathsf{IsList}(\ell) = 1, \mathsf{Len}(\ell) = z \to t_1 = t_2 \vdash \mathsf{Len}(\ell) = s_1(z) \to t_1 = t_2$.

(*Subgoal 1*) is relatively easy. We first apply $(\to_i)$ to introduce $\mathsf{Len}(\ell) = \varepsilon$ as an assumption. By unfolding the definition of $\mathsf{IsList}(\ell)$, we can prove $\Gamma, \mathsf{IsList}(\ell) = 1 \vdash \ell \neq \varepsilon \to \ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$ (details omitted), and therefore by the cut rule we can add the consequence as an assumption. Next, we prove that

$$\Gamma, \mathsf{IsList}(\ell) = 1, \mathsf{Len}(\ell) = \varepsilon, \mathsf{IsList}(\ell) = 1, \neg \ell = \varepsilon \to \ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell) \vdash \ell = \varepsilon.$$

To see this, we prove towards a contradiction that $\ell \neq \varepsilon$ (i.e. applying $(\bot_e)$). Since we have both $\ell \neq \varepsilon$ and $\ell \neq \varepsilon \to \ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$ as assumptions, we can derive that $\ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$. By rewriting rule and the assumption that $\mathsf{Len}(\ell) = \varepsilon$, we know that

$$\mathsf{Len}(\mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)) = s_0(\mathsf{Tail}(\ell)) = \varepsilon,$$

which leads to a contradiction by $(\mathrm{D}\varepsilon)$. Finally, with $\ell = \varepsilon$ and the premise that $\Gamma \vdash t_1[\ell/\varepsilon] = t_2[\ell/\varepsilon]$, we can prove $t_1 = t_2$ as $t_i = t_i[\ell/\varepsilon]$ for each $i \in \{1, 2\}$ by rewriting.

(*Subgoal 3*) is also straightforward. Similar to the proof of (*Subgoal 1*), we can see that $\ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$, so that $\mathsf{Len}(\ell) = s_0(\mathsf{Len}(\mathsf{Tail}(\ell))) \neq s_1(z)$. Therefore we can prove $\bot$ from the assumptions, and subsequently prove $t_1 = t_2$ by the explosion rule, which is a special case of $(\bot_e)$.

It remains to prove (*Subgoal 2*). We first introduce $\mathsf{Len}(\ell) = s_0(z)$ as an assumption. Similar to the proof of (*Subgoal 1*), we can prove that $\ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$. As $\mathsf{IsList}(\ell) = 1$, we can also prove that $\mathsf{IsList}(\mathsf{Tail}(\ell)) = 1$. Moreover, by $\mathsf{Len}(\ell) = s_0(z)$,

we can prove that $\mathsf{Len}(\mathsf{Tail}(\ell)) = z$. Applying the cut rule we can add all provable equations above as assumptions. It then follows (from the assumptions) that:

$$t_1[\ell/\mathsf{Tail}(\ell)] = t_2[\ell/\mathsf{Tail}(\ell)],$$

which is left as an exercise.

Hint: Pick a fresh variable in place of $\mathsf{Tail}(\ell)$ and use substitution rule.

Recall that by the premises we know that

$$\Gamma, \mathsf{IsList}(\ell) = 1, t_1 = t_2 \vdash t_1[\ell/\ell :: u] = t_2[\ell/\ell :: u].$$

Pick a fresh variable x. We can apply the substitution rule for three times with $u/\mathsf{Head}(x)$, $\ell/\mathsf{Tail}(x)$, and subsequently x/ℓ, so that we have

$$\begin{aligned}&\Gamma, \mathsf{IsList}(\mathsf{Tail}(\ell)) = 1, t_1[\ell/\mathsf{Tail}(\ell)] = t_2[\ell/\mathsf{Tail}(\ell)]\\ \vdash\ & t_1[\ell/\mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)] = t_2[\ell/\mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)].\end{aligned}$$

Notice that all formulas in the antecedents have already been available as assumptions, and therefore we can prove the consequence $t_1[\ell/\mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)] = t_2[\ell/\mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)]$. This further leads to the proof of $t_1 = t_2$ since $\ell = \mathsf{Tail}(\ell) :: \mathsf{Head}(\ell)$ is available as an assumption. □

5.3 Proof of the Recursion on Lists Meta-theorem

Recall the statement of Theorem 5.3:

Theorem 5.3 (Recursion on Lists)**.** *Let $g(\vec{x})$, $h(\vec{x}, u, \ell, z)$, $k(\vec{x}, u, \ell)$ be* PV *functions satisfying that*

$$\mathsf{PV} \vdash \mathsf{ITR}(h(\vec{x}, u, \ell, z), z \circ k(\vec{x}, u, \ell)) = 0.$$

Then there is a PV *function $f(\vec{x}, \ell)$ such that* PV *proves:*

$$f(\vec{x}, \varepsilon) = g(\vec{x}) \tag{113}$$
$$f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f(\vec{x}, \ell)) \tag{114}$$

The intuition of our proof is to define a function $f'(\vec{x}, \ell, y)$ that outputs $f(\vec{x}, \hat{\ell}_y)$, where $\hat{\ell}_y$ denotes the list obtained by iteratively removing the last (i.e. rightmost) element in ℓ for $y' := \mathsf{ITR}(\ell, y)$ times. Note that the function $\mathsf{ITRL}(\ell, y)$ that outputs the list obtained by iteratively removing the last element in ℓ for y times can be defined naturally by recursion on y, and we can prove that $\mathsf{ITRL}(\ell, \ell) = \varepsilon$.

Subsequently, $f'(\vec{x}, \ell, y)$ can also be defined by recursion on y. Note that when $y = \varepsilon$, we have that $\ell_y = \varepsilon$ and thus $f'(\vec{x}, \ell, \varepsilon) = f(\vec{x}, \varepsilon)$. Moreover, observe that

$$f'(\vec{x}, \ell, s_i(y)) = \begin{cases} f(\vec{x}, \varepsilon) & \mathsf{IsEps}(\hat{\ell}) \\ h(\vec{x}, \mathsf{Head}(\hat{\ell}), \mathsf{Tail}(\hat{\ell}), f'(\vec{x}, \ell, y)) & \text{otherwise} \end{cases}$$

where $\hat{\ell} := \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y)))$. Finally, we will define $f(\vec{x}, \ell) := f'(\vec{x}, \ell, \ell)$ and prove that Equation (113) and (114) hold.

Step 1: Defining the function ITRL. Let $\mathsf{ITRL}(\ell, y)$ be the function defined as

$$\mathsf{ITRL}(\ell, \varepsilon) := \ell, \quad \mathsf{ITRL}(\ell, s_i(y)) := \mathsf{Tail}(\mathsf{ITRL}(\ell, y)) \quad (i \in \{0, 1\}).$$

We need to prove that $\mathsf{ITRL}(\ell, \ell) = \varepsilon$; indeed, we will need to first prove that $\mathsf{IsEps}(\mathsf{ITR}(\mathsf{ITRL}(\ell, y), \mathsf{ITR}(\ell, y))) = 1$. To see that this suffices, notice that if we substitute y/ℓ to the equation, we have

$$\mathsf{IsEps}(\mathsf{ITR}(\mathsf{ITRL}(\ell, \ell), \mathsf{ITR}(\ell, \ell))) = 1,$$

where the LHS of the equation is PV-provably equal to

$$\mathsf{IsEps}(\mathsf{ITRL}(\ell, \ell)) = 1$$

by unfolding ITR and using the fact that $\mathsf{ITR}(\ell, \ell) = \varepsilon$ (which has been proved). This immediate implies $\mathsf{ITRL}(\ell, \ell) = \varepsilon$ by Theorem 3.12.

Step 2: Defining the function f'. We now define the function $f'(\vec{x}, \ell, y)$ that (intuitively) outputs $f(\vec{x}, \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, y)))$.

Let $g'(\vec{x}, \ell) = g(\vec{x})$. For $i \in \{0, 1\}$, we define

$$h'_i(\vec{x}, \ell, y, z) := \begin{cases} g(\vec{x}) & \mathsf{IsEps}(\hat{\ell}) \\ h(\vec{x}, \mathsf{Head}(\hat{\ell}), \mathsf{Tail}(\hat{\ell}), z) & \text{otherwise} \end{cases}$$

where $\hat{\ell} := \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y)))$, and

$$k'_i(\vec{x}, \ell, y) := \begin{cases} g(\vec{x}) & \mathsf{IsEps}(\hat{\ell}) \\ k(\vec{x}, \mathsf{Head}(\hat{\ell}), \mathsf{Tail}(\hat{\ell}), z) & \text{otherwise} \end{cases}$$

To recursively define a function from g', h'_i, k'_i, we need to show that it is provable in PV that $\mathsf{ITR}(h'_i(\vec{x}, \ell, y, z), z \circ k'_i(\vec{x}, \ell, y)) = \varepsilon$ for $i \in \{0, 1\}$. Notice that both h'_i and k'_i are defined with a case study on the condition $\mathsf{IsEps}(\hat{\ell})$. Therefore, by case study on ITE (see Theorem 3.6 and Remark 3.1), it suffices to prove that

- $\mathsf{ITR}(g(\vec{x}), g(\vec{x})) = \varepsilon$
- $\mathsf{ITR}(h(\vec{x}, \mathsf{Head}(\hat{\ell}), \mathsf{Tail}(\hat{\ell}), z), z \circ k(\vec{x}, \mathsf{Head}(\hat{\ell}), \mathsf{Tail}(\hat{\ell}), z)) = \varepsilon$

The first equation follows from $\mathsf{ITR}(x, x) = \varepsilon$, and the second equation follows from the assumption.

We can then define a function $f'(\vec{x}, \ell, y)$ recursively from g', h'_i, k'_i in PV, and we define $f(\vec{x}, \ell) := f'(\vec{x}, \ell, \ell)$.

Step 3: Verifying the properties. Now it suffices to verify the properties in Equation (113) and (114) for the function f we defined above. To see that Equation (113), notice that PV proves

$$f(\vec{x}, \varepsilon) = f'(\vec{x}, \varepsilon, \varepsilon) = g(\vec{x})$$

by unfolding the definitions.

The proof of Equation (114), however, is much more complicated. We will prove that

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(y))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, y)). \tag{126}$$

Recall that $f'(\vec{x}, \ell, y)$ is intended to be $f(\vec{x}, \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, y)))$, and when $|y| \leq |\ell|$, Equation (126) is intended to be

$$f(\vec{x}, \mathsf{ITRL}(\ell :: u, s_0(y))) = f(\vec{x}, \mathsf{ITRL}(\ell, y)),$$

which should be true since $\mathsf{ITRL}(\ell :: u, s_0(y)) = \mathsf{ITRL}(\ell, y)$.

Proposition 5.9. $\mathsf{PV} \vdash \mathsf{ITRL}(\ell :: u, s_0(y)) = \mathsf{ITRL}(\ell, y)$.

Proof. This can be proved by an induction on y. The equation trivially holds when $y = \varepsilon$ by unfolding the LHS. Suppose that the equation holds for y, notice that

$$\mathsf{ITRL}(\ell :: u, s_0(s_0(y))) = \mathsf{Tail}(\mathsf{ITRL}(\ell :: u, s_0(y)))$$

and

$$\mathsf{ITRL}(\ell, s_0(y)) = \mathsf{Tail}(\mathsf{ITRL}(\ell, y)).$$

Since $\mathsf{ITRL}(\ell :: u, s_0(y)) = \mathsf{ITRL}(\ell, y)$ by the assumption hypothesis, we can prove the induction case by applying the PV-PL axioms of equality. □

Proof of Equation (114). We first show that Equation (126) suffices to prove Equation (114). Suppose that Equation (126) is true, by substituting y/ε and unfolding ITR, we can obtain that

$$f'(\vec{x}, \ell :: u, \mathsf{TR}(\ell :: u)) = f'(\vec{x}, \ell, \ell)$$

where the RHS is $f(\vec{x}, \ell)$. We will then show that

$$f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\ell :: u))),$$

which, together with the fact that $f'(\vec{x}, \ell :: u, \mathsf{TR}(\ell :: u)) = f(\vec{x}, \ell)$, concludes the proof of Equation (114).

Proposition 5.10. PV *proves that*

$$f'(\vec{x}, \ell :: u, y) = \begin{cases} g(\vec{x}) & \mathsf{Or}(\mathsf{IsEps}(\hat{\ell}), \mathsf{IsEps}(y)) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(y))) & \textit{otherwise} \end{cases}$$

where $\hat{\ell} := \mathsf{ITRL}(\ell :: u, \mathsf{ITR}(\ell :: u, y))$.

Proof Sketch. We perform case study on y using the induction rule of PV-PL. This equation holds for all three cases y/ε, $y/s_0(y)$, and $y/s_1(y)$ by simple unfolding. □

Proposition 5.11. $\mathsf{PV} \vdash f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\ell :: u)))$.

Proof Sketch. Note that it is easy to prove that $\mathsf{PV} \vdash \mathsf{IsNotEps}(\ell :: u) = 1$ by unfolding the definition of Append. Moreover, let $\hat{\ell} = \mathsf{ITRL}(\ell :: u, \mathsf{ITR}(\ell :: u, \ell :: u))$, we know that $\hat{\ell} = \ell :: u$ and thus we can prove that $\mathsf{IsEps}(\hat{\ell}) = 0$. By Proposition 5.10 with $y/\ell :: u$ and unfolding ITE's (used to implement the case analysis), we can conclude that

$$f'(\vec{x}, \ell :: u, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\ell :: u))),$$

where the LHS is exactly $f(\vec{x}, \ell :: u)$ by the definition axiom of f. □

Proof of Equation (126). We will prove Equation (126) by performing an induction on $\mathsf{ITR}(\ell, y)$. Concretely, let y' be a fresh variable, we will prove the equation where y is substituted by $\mathsf{ITR}(\ell, y')$ by induction on y'.

Note that this will actually lead to a proof of

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, y')))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, y'))) \tag{127}$$

that is not known to imply Equation (126). Nevertheless, since what we actually need to do is to prove the original property Equation (114) of the function f by substituting y/ε, the equation above also suffices with the substitution y'/ℓ instead.

Recall that $\mathsf{EQL}(x, y)$ outputs 1 (resp. 0) if and only if $|x| = |y|$ (resp. $|x| \neq |y|$). It turns out:

Proposition 5.12. PV-PL *proves the following assertions:*

- $\mathsf{IsNotEps}(\mathsf{ITR}(x, y)) \vdash \mathsf{EQL}(\mathsf{ITR}(x, \mathsf{ITR}(x, y)), y) = 1$.
- $\mathsf{EQL}(x, y) \vdash \mathsf{ITRL}(\ell, x) = \mathsf{ITRL}(\ell, y)$.
- $\mathsf{EQL}(x, y) \vdash f'(\vec{z}, \ell, x) = f'(\vec{z}, \ell, y)$.

Proof Sketch. All the equations can be proved by applying induction on x and y using the induction rule of PV-PL. □

We perform an induction on y' to prove Equation (127), in which we will need to resolve the following three subgoals:

(*Subgoal* 1). $\vdash$ Equation (127) with substitution y'/ε.
(*Subgoal* 2). Equation (127) $\vdash$ Equation (127) with substitution $y'/s_0(y')$
(*Subgoal* 3). Equation (127) $\vdash$ Equation (127) with substitution $y'/s_1(y')$

We will deal with the three cases in the following lemmas.

Lemma 5.13 (*Subgoal* 1)**.** $\mathsf{PV} \vdash f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\ell))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \ell))$.

Proof Sketch. Since $\mathsf{PV} \vdash \mathsf{ITR}(\ell, \ell) = \varepsilon$, the RHS of the equation is PV-provably equal to $f'(\vec{x}, \ell, \varepsilon)$, and by the definition axiom of f' it is further PV-provably equal to $g(\vec{x})$.

As for the LHS, notice that $\mathsf{PV} \vdash \mathsf{IsEps}(\mathsf{ITR}(\ell :: u, s_0(\ell))) = 0$ by unfolding $\ell :: u$ and proving basic properties about ITR. By Proposition 5.10, we know that

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\ell))) = \begin{cases} g(\vec{x}) & \mathsf{Or}(\mathsf{IsEps}(\hat{\ell}), \mathsf{IsEps}(\mathsf{ITR}(\ell :: u, s_0(\ell)))) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\ell))))) & \text{otherwise} \end{cases}$$

where $\hat{\ell} := \mathsf{ITRL}(\ell :: u, \mathsf{ITR}(\ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\ell)))))$. Note that

$$\mathsf{EQL}(\mathsf{ITR}(\ell :: u, \mathsf{ITR}(\ell :: u, s_0(\ell))), s_0(\ell)) = 1,$$

and the proof is left as an exercise. By $\mathsf{EQL}(x, y) = 1 \Rightarrow \mathsf{ITRL}(\ell, x) = \mathsf{ITRL}(\ell, y)$ we can further prove that

Hint: Using Proposition 5.12

$$\hat{\ell} = \mathsf{ITRL}(\ell :: u, s_0(\ell)),$$

which is further equal to $\mathsf{ITRL}(\ell, \ell) = \varepsilon$. Therefore by unfolding the equation from Proposition 5.10 we have $f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\ell))) = g(\vec{x})$. □

Next, we prove (*Subgoal* 2) and (*Subgoal* 3). Fix any $i \in \{0,1\}$. Let $\hat{\ell}$ be a fresh variable, so that we can introduce $\hat{\ell} = \mathsf{ITRL}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))$ as an assumption (aka. in the antecedent). Note that by unfolding ITRL we can prove in PV that $\hat{\ell} = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y')))$.

Introduction of fresh variable can be done by (V), (A), and (Cut).

We will perform a case analysis on whether $\mathsf{IsEps}(\hat{\ell}) = 1$. Concretely, let Γ be the set of antecedents including:

- $\hat{\ell} = \mathsf{ITRL}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))$
- $\hat{\ell} = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y')))$
- $f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, y')))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, y')))$ (i.e. Equation (127)).

We will prove the following two lemmas:

Lemma 5.14. PV-PL *proves* $\Gamma, \mathsf{IsEps}(\hat{\ell}) = 1 \vdash$

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))).$$

Lemma 5.15. PV-PL *proves* $\Gamma, \mathsf{IsEps}(\hat{\ell}) = 0 \vdash$

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))).$$

These two lemma suffices to prove that $\Gamma \vdash$ Equation (127) (with the substitution $y'/s_i(y')$ by a case study on $\hat{\ell}$, which subsequently completes the proof of all subgoals.

Proof Sketch of Lemma 5.14. Indeed, we will prove that the LHS and the RHS of the equation are both identical to $g(\vec{x})$.

(LHS). Note that by Proposition 5.10, we know that

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))))$$
$$= \begin{cases} g(\vec{x}) & \mathsf{Or}(\mathsf{IsEps}(\hat{\ell}'), \mathsf{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(y''))) & \text{otherwise} \end{cases}$$

where

$$y'' := \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))),$$
$$\hat{\ell}' := \mathsf{ITRL}(\ell :: u, \mathsf{ITR}(\ell :: u, y'')).$$

Note that we can prove $\mathsf{EQL}(\mathsf{ITR}(\ell :: u, y''), s_0(\mathsf{ITR}(\ell, s_i(y')))) = 1$ (details omitted), so that by Proposition 5.12:

$$\hat{\ell}' = \mathsf{ITRL}(\ell :: u, \mathsf{ITR}(\ell :: u, y'')) = \mathsf{ITRL}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))) = \hat{\ell}.$$

Since $\mathsf{IsEps}(\hat{\ell}) = 1$ is available in the antecedent, we can apply the rewrite rule $(\hat{=}/)$, unfold Or and ITE, and conclude that

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))) = g(\vec{x}).$$

(RHS). Again, by Proposition 5.10, we know that

$$f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y'))))$$
$$= \begin{cases} g(x) & \mathsf{Or}(\mathsf{IsEps}(\hat{\ell}'), \mathsf{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell, \mathsf{TR}(y''))) & \text{otherwise} \end{cases}$$

where

$$y'' := \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y'))),$$
$$\hat{\ell}' := \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, y'')).$$

Note that we can prove $\mathsf{EQL}(\mathsf{ITR}(\ell, y''), \mathsf{ITR}(\ell, s_i(y'))) = 1$ (details omitted), so that by Proposition 5.12:

$$\hat{\ell}' = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, y'')) = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y'))) = \hat{\ell}.$$

Since $\mathsf{IsEps}(\hat{\ell}) = 1$ is available in the antecedent, we can apply the rewrite rule $(\hat{=}_/)$, unfold Or and ITE, and conclude that

$$f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))) = g(\vec{x}).$$

This completes the proof. □

Proof of Lemma 5.15. Similar to the proof of Lemma 5.14, we can apply Proposition 5.10 to show that:

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))))$$
$$= \begin{cases} g(\vec{x}) & \mathsf{Or}(\mathsf{IsEps}(\hat{\ell}'), \mathsf{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(y''))) & \text{otherwise} \end{cases}$$

where it is provable that $\hat{\ell}' = \hat{\ell}$, and $y'' := \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))$. It can be proved that $\mathsf{IsEps}(y'') = 0$ as $\mathsf{IsEps}(\mathsf{ITR}(\ell :: u, s_0(\ell))) = 0$. Therefore, by the assumption that $\mathsf{IsEps}(\hat{\ell}) = 0$, we can conclude by applying the rewrite rule and unfolding that:

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))))$$
$$= h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))))). \tag{128}$$

Similarly, we can also prove that

$$f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y'))))$$
$$= h(\vec{x}, u, \ell, f'(\vec{x}, \ell, \mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))))). \tag{129}$$

Notice that the LHS of Equation (128) and (129) are the LHS and RHS of the equation in the lemma, respectively. Therefore, it suffices to prove that

$$f'(\vec{x}, \ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))))$$
$$= f'(\vec{x}, \ell, \mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y'))))). \tag{130}$$

Finally, notice that PV proves:

$$\mathsf{EQL}(\mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))), \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, y')))) = 1,$$
$$\mathsf{EQL}(\mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))), \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, y'))) = 1.$$

(The detail is omitted.) We can therefore modify our goal Equation (130) using Proposition 5.12 and the rewrite rule to:

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, y')))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, y'))),$$

which is available as the third bullet of the antecedent (see the definition of Γ above). □

6 Feasible Mathematics Thesis, Revisited

In this section, we give additional evidence to the Feasible Mathematics Thesis by showing that PV is strong enough to simulate natural computation models, including a simple imperative programming language IMP(PV) and Turing machines.

Building on these results, we provide a proof of Gödel's incompleteness theorem for PV, which shows that PV cannot prove the consistency of itself.

6.1 Simulation of Imperative Programming Languages

We now demonstrate how to simulate ordinary imperative programming languages using functions in PV and reason about programs.

After all, most programmers rely on imperative programming languages in their minds, I guess...

6.1.1 Syntax of IMP(PV)

We introduce a simple untyped imperative programming language IMP(PV) that extends PV functions with variables and control statements. We start by defining the syntax of IMP(PV).

- (*Expressions*). An expression is an arbitrary PV term.
- (*Commands*). Commands in IMP(PV) is defined by the following BNF:

$$\begin{aligned}\mathsf{Cmd} := {} & \textbf{skip} \mid \textbf{let } x := \mathsf{Exp} \mid \mathsf{Cmd}; \mathsf{Cmd} \\ & \mid \textbf{if } \mathsf{Exp} \textbf{ then } \mathsf{Cmd} \textbf{ else } \mathsf{Cmd} \\ & \mid \textbf{for } x := \mathsf{Exp};\ x \neq \varepsilon;\ x := \mathsf{TR}(x) \textbf{ do } \mathsf{Cmd}\end{aligned}$$

  where x denotes an arbitrary variable name.

For simplicity, we will assume that all variables are global variables (i.e. structural statements do not create name spaces). The semantic of an IMP(PV) in the standard model should be clear.

To ensure that the program terminates in polynomial time, we further introduce two restrictions to IMP(PV) programs:

- (Read-only restriction). The variable x used as the index variable of a for-loop statement is read-only within the loop, i.e., it neither be assigned using **let** statement within the loop, nor be the index variable of an inner for-loop statement.
- (Length restriction). Let P be an IMP(PV) program satisfying the read-only restriction, and ω be a fresh variable, an *ω-length-restricted* program is of form (P, ω), which intuitively means that all variables and expressions are subject to the length upper bound ω. An expression that evaluates to a string y longer than ω will be rounded to $\mathsf{Suf}(y, \omega)$, i.e., the rightmost $|\omega|$-bits of y. (Nevertheless, we allow the intermediate computation within an expression to temporarily exceed the length restriction, as it will not cause any trouble.)

Arguably, these two restrictions should usually not trouble programmers: The read-only restriction is essentially enforcing a good programming habit, and the length restriction also occurs in real-world programming languages due to the word length of built-in data types.

I guess...?

6.1.2 Formal Semantic via Hoare Logic

To define the semantic of IMP(PV) as well as reason about programs, we introduce a Hoare logic for IMP(PV).

Let φ_1, φ_2 be PV-PL formulas and $c \in \mathsf{Cmd}$ be a command, a *Hoare tuple* is of the form $\{\varphi_1\}\ c\ \{\varphi_2\}$ indicating (intuitively) that if φ_1 is true before the execution of c, φ_2 is true after the execution of c. The formula φ_1 is called the *precondition*, and φ_2 is called the *postcondition*.

The semantic of IMP(PV) is given by the following axioms and rules:

- Consequence rule: Suppose that PV-PL proves $\varphi_1 \vdash \varphi_1'$ and $\varphi_2' \vdash \varphi_2$, then
$$\frac{\{\varphi_1'\}\ c\ \{\varphi_2'\}}{\{\varphi_1\}\ c\ \{\varphi_2\}}$$
This rule is used to weaken the precondition and strengthen the postcondition.
- Empty statement axiom scheme:
$$\frac{}{\{\varphi\}\ \textbf{skip}\ \{\varphi\}}$$
- Assignment axiom scheme:
$$\frac{}{\{\varphi[x/e]\}\ \textbf{let}\ x := e\ \{\varphi\}}$$
- Rule of composition:
$$\frac{\{\varphi_1\}\ c_1\ \{\psi\} \qquad \{\psi\}\ c_2\ \{\varphi\}}{\{\varphi_1\}\ c_1; c_2\ \{\varphi_2\}}$$
- Rule of if-then-else:
$$\frac{\{\varphi_1 \wedge \mathsf{LastBit}(e) = 1\}\ c_1\ \{\varphi_2\} \qquad \{\varphi_1 \wedge \mathsf{LastBit}(e) \neq 1\}\ c_0\ \{\varphi_2\}}{\{\varphi_1\}\ \textbf{if}\ e\ \textbf{then}\ c_1\ \textbf{else}\ c_0\ \{\varphi_2\}}$$
$\varphi \wedge \varphi_2$ is indeed defined using connectives $\{\rightarrow, \neg\}$ in PV-PL.
- Rule of for-loop: For $i \in \{0, 1\}$, we have
$$\frac{\{\varphi[x/s_i(x)]\}\ c\ \{\varphi\}}{\{\varphi[x/e]\}\ \textbf{for}\ x := e;\ x \neq \varepsilon;\ x := \mathsf{TR}(x)\ \textbf{do}\ c\ \{\varphi[x/\varepsilon]\}}$$
This rule allows to reason about the for-loop using a loop invariant φ that is maintained throughout the loop.

We say that a Hoare tuple $\{\varphi_1\}\ c\ \{\varphi_2\}$ is provable if there is a proof tree using the axioms and rules above concluding $\{\varphi_1\}\ c\ \{\varphi_2\}$.

The rules of Hoare logic provide an intuitive reasoning framework on the bahavior of IMP(PV) programs. The interpretation of the rules should be clear. We will use the following example to show the application of the Hoare rules.

Example 6.1. Consider the following IMP(PV) program for calculating the length of a list by iteratively removing the rightmost element:

$$\begin{aligned}\mathsf{LenITR} := \ & \textbf{let}\ z := \ell; \\ & \textbf{let}\ y := \varepsilon; \\ & \textbf{for}\ x := \ell;\ x \neq \varepsilon;\ x := \mathsf{TR}(x)\ \textbf{do} \\ & \ \textbf{if}\ \mathsf{IsEps}(z)\ \textbf{then skip else let}\ y := s_0(y);\ \textbf{let}\ z := \mathsf{Tail}(z)\end{aligned}$$

We can formalize its correctness as the Hoare tuple

$$\{\mathsf{IsList}(\ell) = 1\}\ \mathsf{LenITR}\ \{y = \mathsf{Len}(\ell)\}, \tag{131}$$

where Len is the function defined by Equation (117) and (118).

It can be proved using the loop invariant:

$$\varphi := \bigwedge \{\mathsf{IsList}(z) = 1, \mathsf{Len}(z) \circ y = \mathsf{Len}(\ell), \mathsf{ITR}(x, \ell) = \varepsilon, z = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, x))\},$$

where $\mathsf{ITRL}(\ell, y)$ is the function defined as

$$\mathsf{ITRL}(\ell, \varepsilon) := \ell, \quad \mathsf{ITRL}(\ell, s_i(y)) := \mathsf{Tail}(\mathsf{ITRL}(\ell, y)) \quad (i \in \{0, 1\}).$$

More formally, let LenITR_3 be the for-loop in LenITR, we will prove the Hoare tuple $\{\varphi[x/\ell]\}\ \mathsf{LenITR}_3\ \{\varphi[x/\varepsilon]\}$. It is left as an exercise that Equation (131) can be derived from this Hoare tuple.

To prove $\{\varphi[x/\ell]\}\ \mathsf{LenITR}_3\ \{\varphi[x/\varepsilon]\}$, notice that LenITR_3 is a for-loop, and therefore we can apply the *rule of for-loop* so that it suffices to prove that

$$\{\varphi[x/s_i(x)]\}\ \textbf{if}\ \mathsf{IsEps}(z)\ \textbf{then skip else let}\ y := s_0(y);\ \textbf{let}\ z := \mathsf{Tail}(z)\ \{\varphi\}.$$

Subsequently, we can apply the *rule of if-then-else*, so that it suffices to prove the following two subgoals:

- (*Subgoal 1*). $\{\varphi[x/s_i(x)] \wedge \mathsf{LastBit}(\mathsf{IsEps}(z)) = 1\}\ \textbf{skip}\ \{\varphi\}$
- (*Subgoal 2*). $\{\varphi[x/s_i(x)] \wedge \mathsf{LastBit}(\mathsf{IsEps}(z)) \neq 1\}\ \textbf{let}\ y := s_0(y);\ \textbf{let}\ z := \mathsf{Tail}(z)\ \{\varphi\}$.

To prove (*Subgoal 1*), it suffices to prove in PV-PL that

$$\varphi[x/s_i(x)] \wedge \mathsf{LastBit}(\mathsf{IsEps}(z)) = 1 \vdash \varphi,$$

apply the *consequence rule*, and subsequently apply the *empty statement axiom*. The PV-PL proof of the assertion above is easy and left as an exercise.

To prove (*Subgoal 2*), it suffices to prove in PV-PL that

$$\varphi[x/s_i(x)] \wedge \mathsf{LastBit}(\mathsf{IsEps}(z)) \neq 1 \vdash \varphi[z/\mathsf{Tail}(z)][y/s_0(y)],$$

apply the *consequence rule* to weaken the precondition to $\varphi[z/\mathsf{Tail}(z)][y/s_0(y)]$, and subsequently apply the *rule of composition* (with $\psi := \varphi[z/\mathsf{Tail}(z)]$) and the *assignment axiom* in both branches generated by the *rule of composition*. The proof of the assertion above is tedious but straightforward.

Similarly, we can define the Hoare logic of ω-length-restricted PV programs, where the *assignment axiom scheme* and the *rule of for-loop* are modified by replacing all occurrences of the expression e in the preconditions and postconditions to $\mathsf{Suf}(e, \omega)$. We will denote a Hoare tuple for ω-length-restricted programs by $\{\varphi_1\}\ (P, \omega)\ \{\varphi_2\}$.

It is left as an exercise that for the example above, we can pick an appropriate length upper bound ω such that we can still prove the correctness of the (ω-length-restricted) program.

6.1.3 Compiling Programs to PV Functions

Next, we show that it is possible to compile any (length-restricted) IMP(PV) program back to a PV function in a way that preserves the semantic of the function computed by the program.

Let (P, ω) be an ω-length-restricted IMP(PV) program. We define its PV translation, denoted also by $[P]_{\mathsf{PV}}$, as a PV function $f_P(\omega, \pi)$ follows.

- (*Context storage*). Suppose that $k \in \mathbb{N}$ variables are used in P named $x_1, \ldots, x_k$, we will use a k-tuple π of length k to store the context, i.e., $\mathsf{UnwindTuple}_i(\pi)$ stores the value of x_k. For simplicity, we will denote $\mathsf{UnwindTuple}_i(\pi)$ simply by π_i.

- (*Expressions*). For an expression e in P, where the variables are from $x_1, \ldots, x_k$, we define its PV translation with respect to the context π, denoted by $[e]^{\pi,\omega}_{\mathsf{PV}}$, as $\mathsf{Suf}(e[x_i/\pi_i \ \forall i \in [k]], \omega)$. We may omit ω in the superscript in case that there is no ambiguity.

- (*Empty statement*). We define $[\textbf{skip}]_{\mathsf{PV}}(\omega, \pi) := \pi$, i.e., the identity function.

- (*Assignment*). We define $[\textbf{let } x_i := e]_{\mathsf{PV}}(\omega, \pi)$ be the function that outputs the tuple $\mathsf{MakeTuple}(\pi_1, \ldots, \pi_{i-1}, [e]^{\pi}_{\mathsf{PV}}, \pi_{i+1}, \ldots, \pi_k)$.

- (*Composition*). We define $[c_1;\ c_2]_{\mathsf{PV}}(\omega, \pi) := [c_2]_{\mathsf{PV}}([c_1]_{\mathsf{PV}}(\pi))$, i.e., the PV translation commutes with composition of IMP(PV) programs.

- (*If-then-else*). We define the PV translation of if-then-else statement as

$$[\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_0]_{\mathsf{PV}}(\pi) := \mathsf{ITE}^{(\varepsilon)}_0([e]^{\pi}_{\mathsf{PV}}, [c_1]_{\mathsf{PV}}(\pi), [c_2]_{\mathsf{PV}}(\pi)),$$

  where $\mathsf{ITE}^{(\varepsilon)}_0$ is the "dirty" if-then-else function.

- (*For Loop*). We define $[\textbf{for } x_i := e;\ x_i \neq \varepsilon;\ x_i := \mathsf{TR}(x_i) \textbf{ do } c]_{\mathsf{PV}}$ as the function $f(\omega, \pi)$ constructed below:

  - Let $f_{\mathsf{iter}}(\omega, \pi) := \mathsf{MakeTuple}(\pi'_1, \ldots, \pi'_{i-1}, \mathsf{TR}(\pi'_i), \pi'_{i+1}, \ldots, \pi'_k)$, where $\pi' = [c]_{\mathsf{PV}}(\omega, \pi)$.
  - Let $g(\omega, \pi) = \pi$, $h_i(\omega, \pi, y, z) = \pi'$, $k_i(\omega, \pi, y, z) = \pi''$, $i \in \{0, 1\}$, where

$$\pi' := \mathsf{MakeTuple}^{(k)}(\mathsf{Suf}(f_{\mathsf{iter}}(\omega, z)_1, \omega), \ldots, \mathsf{Suf}(f_{\mathsf{iter}}(\omega, z)_k, \omega))$$
$$\pi'' := \mathsf{MakeTuple}^{(k)}(\omega, \omega, \ldots, \omega)$$

    It can be proved in PV that $\mathsf{ITR}(h_i(\omega, \pi, y, z), z \circ k_i(\omega, \pi, y, z)) = \varepsilon$ by the properties of MakeTuple and Suf, which is left as an exercise. Therefore, by the recursion rule of PV, there is a function f_{loop} satisfying that

$$f_{\mathsf{loop}}(\omega, \pi, \varepsilon) = g(\omega, \pi)$$
$$f_{\mathsf{loop}}(\omega, \pi, s_i(y)) = h_i(\omega, \pi, y, f_{\mathsf{loop}}(\omega, \pi, y))$$

  - Let $f(\omega, \pi) = f_{\mathsf{loop}}(\omega, \mathsf{MakeTuple}(\pi_1, \ldots, \pi_{i-1}, [e]^{\omega,\pi}_{\mathsf{PV}}, \pi_{i+1}, \ldots, \pi_k), [e]^{\omega,\pi}_{\mathsf{PV}})$.

6.1.4 Translating of Hoare Proofs to PV Proofs

Hopefully feasible mathematicians will be satisfied with writing even less formal (pseudo) proofs after developing this result :)

It should be easy to see that in the standard model, the semantic of an ω-length-restricted IMP(PV) program is identical to its PV translation. In addition, we will now prove that Hoare proofs can also be translated back into PV proofs — providing a strong and intuitive tool to reason about IMP(PV) programs.

Translation of Hoare tuples. We first demonstrate the formalization of Hoare tuples as PV-PL assertions. Let φ be a formula with variables $x_1, \ldots, x_k$, we define the translation of φ in the context π, denoted as φ^π as the formula obtained by replacing each x_i with $\pi_i (= \mathsf{UnwindTuple}_i^{(k)}(\pi))$. Then a Hoare tuple $\{\varphi_1\}\ c\ \{\varphi_2\}$ is formalized as the PV-PL assertion

$$\varphi_1 \vdash \varphi_2^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))},$$

which is denoted as $[\{\varphi_1\}\ c\ \{\varphi_2\}]_{\mathsf{PV\text{-}PL}}$.

Theorem 6.1. *Let (P, ω) be a length-restricted* IMP(PV) *program. Suppose that there is a Hoare proof of the tuple $\{\varphi_1\}\ (P, \omega)\ \{\varphi_2\}$, there is a* PV-PL *proof the the assertion*

$$\varphi_1 \vdash \varphi_2^{[P]_{\mathsf{PV}}(\omega,\mathsf{MakeTuple}(x_1,\ldots,x_k))}.$$

Admissibility of Hoare Rules. With the translation of Hoare tuples to PV-PL assertion, we can translate Hoare rules and axioms into PV-PL rules and axioms by translating both premises and the conclusion into PV-PL assertions. Since Hoare logic is also formulated as a natural deduction type system, it remains to prove that all Hoare rules (under this translation) are admissible in PV-PL.

We start by showing that the *consequence rule* is admissible.

Lemma 6.2. *The consequence rule is admissible in* PV-PL*:*

$$\frac{\varphi_1 \vdash \varphi_1' \quad \varphi_2' \vdash \varphi_2 \quad \varphi_1' \vdash \varphi_2'^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))}}{\varphi_1 \vdash \varphi_2^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))}}$$

Proof. Note that from the premises, it suffices to derive that

$$\varphi_2'^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))} \vdash \varphi_2^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))}$$

and apply the transitivity of PV-PL assertion. Recall that $\varphi_2'^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))}$ (resp. $\varphi_2^{[c]_{\mathsf{PV}}(\mathsf{MakeTuple}(x_1,\ldots,x_k))}$) is obtained from φ_2' (resp. φ_2) by substituting

$$x_i / \mathsf{UnwindTuple}_i(\mathsf{MakeTuple}(x_1, \ldots, x_k))$$

for each $i \in [k]$. Therefore, this assertion can be proved from the premise $\varphi_2' \vdash \varphi_2$ directly using the substitution rule (V) of PV-PL. □

The proofs for the admissibility of other rules are similar, so we will only demonstrate the proof for the *assignment axiom scheme* and *rule of for-loop*.

Lemma 6.3. *The assignment axiom scheme is admissible in* PV-PL*:*

$$\overline{\varphi[x_i/\mathsf{Suf}(e,\omega)] \vdash \varphi^{[\textbf{\textit{let}}\ x_i:=e]_{\mathsf{PV}}(\omega,\mathsf{MakeTuple}(x_1,\ldots,x_k))}}$$

Proof. By the definition of the PV translation of length-restricted IMP(PV) programs and the correctness of tuples (see Equation (59) to (61)), we know that PV proves

$$\begin{aligned}&[\textbf{let } x_i := e]_{\mathsf{PV}}(\omega, \mathsf{MakeTuple}(x_1, \ldots, x_k))\\ =\ &\mathsf{MakeTuple}(x_1, \ldots, x_{i-1}, \mathsf{Suf}(e, \omega), x_{i+1}, \ldots, x_k).\end{aligned}$$

Moreover, notice that PV also proves

$$\varphi^{\mathsf{MakeTuple}(x_1,\ldots,x_{i-1},\mathsf{Suf}([e],\omega),x_{i+1},\ldots,x_k)} = \varphi[x_i/\mathsf{Suf}(e,\omega)].$$

Therefore the axiom scheme is provable by rewriting the consequence to $\varphi[x_i/\mathsf{Suf}(e,\omega)]$ (using Lemma 4.36) and then apply the assumption axiom of PV-PL. □

Lemma 6.4. *Let c be an* IMP(PV) *program and P be the program* ***for*** $x_i := e;\ x_i \neq \varepsilon;\ x_i := \mathsf{TR}(x_i)$ ***do*** c. *The rule of for-loop, formalized as follows, is admissible in* PV-PL*:*

$$\frac{\varphi[x_i/s_i(x_i)] \vdash \varphi^{[c]_{\mathsf{PV}}(\omega,\mathsf{MakeTuple}(x_1,\ldots,x_k))}}{\varphi[x_i/\mathsf{Suf}(e,\omega)] \vdash \varphi[x_i/\varepsilon]^{[P]_{\mathsf{PV}}(\omega,\mathsf{MakeTuple}(x_1,\ldots,x_k))}}$$

Proof. TKTK □

6.2 Simulation of Turing Machines

6.3 Gödel's Incompleteness Theorem

7 Connection to Propositional Proof Complexity

In this section, we will try to address the question we posted at very beginning.

Recall that feasible mathematics is defined as an interpretation of constructive mathematics (i.e. the BHK interpretation), where the "effective procedures" are interpreted by feasible functions, i.e., polynomial-time computable functions. However, the meaning of "proofs" is not specified: We believe that the notion of proof should be at most as strong as NP, it is unclear which specific NP proof system should we consider in terms of feasible mathematics.

Indeed, we will see that the notion of "proofs" will be defined by a translation, usually called Cook's translation, from PV proofs to a propositional proof system called *extended Frege* of bounded length. Moreover, we will show that there is a tight connection between extended Frege proofs and PV proofs, which is similar to the connection between circuits and uniform algorithms.

References

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole principle. *Comb.*, 14(4):417–433, 1994.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the p=?np question. *SIAM Journal on computing*, 4(4):431–442, 1975.

[Bis67] Errett Bishop. Foundations of constructive analysis. 1967.

[BN21] Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 233–350. IOS Press, 2021.

[BPI22] Douglas Bridges, Erik Palmgren, and Hajime Ishihara. Constructive Mathematics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

[CHO+22] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. *J. ACM*, 69(4):25:1–25:49, 2022.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1990–1999. ACM, 2024.

[CLO24] Lijie Chen, Jiatu Li, and Igor Carboni Oliveira. Reverse mathematics of complexity lower bounds. *Electron. Colloquium Comput. Complex.*, pages TR24–060, 2024.

[Cob65] Alan Cobham. The intrinsic computational difficulty of functions. 1965.

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In William C. Rounds, Nancy Martin, Jack W. Carlyle, and Michael A. Harrison, editors, *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, May 5-7, 1975, Albuquerque, New Mexico, USA*, pages 83–97. ACM, 1975.

[Coo90] Stephen A Cook. *Computational complexity of higher type functions*. American Mathematical Society, 1990.

[Dij68] Edsger W. Dijkstra. Letters to the editor: go to statement considered harmful. *Commun. ACM*, 11(3):147–148, 1968.

[Dij72] Edsger W Dijkstra. The humble programmer. *Communications of the ACM*, 15(10):859–866, 1972.

[JJ22] Abhishek Jain and Zhengzhong Jin. Indistinguishability obfuscation via mathematical proofs of equivalence. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1023–1034. IEEE, 2022.

[JKLV24] Zhengzhong Jin, Yael Kalai, Alex Lombardi, and Vinod Vaikuntanathan. Snargs under LWE via propositional proofs. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1750–1757. ACM, 2024.

[Kha22] Erfan Khaniki. Nisan-wigderson generators in proof complexity: New lower bounds. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 17:1–17:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 44:1–44:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Kor21] Oliver Korten. The hardest explicit construction. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 433–444. IEEE, 2021.

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 1(170):123–140, 2001.

[Kra10] Jan Krajícek. A form of feasible interpolation for constant depth frege systems. *J. Symb. Log.*, 75(2):774–784, 2010.

[Kra19] Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 2000–2007. ACM, 2024.

[Oli24] Igor Carboni Oliveira. Meta-mathematics of computational complexity theory. 2024.

[Pic15] Ján Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. *Log. Methods Comput. Sci.*, 11(2), 2015.

[PS21] Ján Pich and Rahul Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. In *Symposium on Theory of Computing* (STOC), pages 223–233, 2021.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.