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:

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.

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

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 $f_\Pi^{(1)}(u, v, y)$ 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 $\mathsf{ITE}(y, u, v) = f_\Pi^{(1)}(u, v, y)$.

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 $f_{-1}^{(j)}(\vec{x}), f'^{(j)}_{-1}(\vec{x})$ 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 $\mathsf{And}, \mathsf{Or}, \mathsf{Not}$. 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 $\{\mathsf{And}, \mathsf{Or}, \mathsf{Not}\}$ 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 $\mathsf{PV} \vdash \mathsf{LastBit}(p_{\varphi_1}) = \mathsf{LastBit}(p_{\varphi_2})$.*

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`.)

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

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 $\{\mathsf{And}, \mathsf{Or}, \mathsf{Not}\}$,

- 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

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.

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

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. □

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 $\mathsf{LastBit}$ 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

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

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

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":

$$
\begin{aligned}
\mathsf{RightRaw}(\varepsilon) &:= \varepsilon &(44)\\
\mathsf{RightRaw}(s_0(\tau)) &:= s_0(\mathsf{RightRaw}(\tau)) &(45)\\
\mathsf{RightRaw}(s_1(\tau)) &:= \mathsf{ITE}(\tau, 1, s_1(\mathsf{RightRaw}(\tau))) &(46)
\end{aligned}
$$

The formal definition of $\mathsf{RightRaw}$ 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 $\mathsf{CleanLeft}$ 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:

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

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 :)

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

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 $i \in [k]$:

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

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

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

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

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

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:*

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

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

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

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

  (Equation (75) can be proved in PV by unfolding the definition of f and applying identities for ITR, such as $\mathsf{ITR}(s_{i_2}(s_{i_1}(y_1) \circ y_2), s_{i_1}(y_1) \circ y_2) = \mathsf{ITR}(s_i(y_1 \circ y_2), y_1 \circ y_2)$ and $\mathsf{ITR}(s_{i_1}(y_1), \mathsf{ITR}(s_i(y_1 \circ y_2), y_1 \circ y_2)) = y_1$, $\mathsf{ITR}(x, x) = \varepsilon$.) Also, notice that

  Hopefully you are convinced that these equations are all provable in PV.

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

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.

This completes the proof. $\square$

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

$$\mathsf{PV} \vdash \mathsf{EQ}(x, y) = \mathsf{EQ}(y, x). \tag{80}$$

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 3.20. *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 (80), 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 3.20 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 3.20 we know that $\mathsf{PV} \vdash f_1(x, y) = f_2(x, y)$.

Proof Sketch of Theorem 3.20. The proof idea of Theorem 3.20 closely follows the proof of Theorem 3.19. For each $j \in \{1, 2\}$, we define

$$f'_j(\vec{x}, y_1, y_2, w_1, w_2) := f_j(\vec{x}, \mathsf{ITR}(y_1, \mathsf{ITR}(w_1, w_2)), \mathsf{ITR}(y_2, \mathsf{ITR}(w_1, w_2))). \tag{81}$$

We will prove that

$$\mathsf{PV} \vdash f'_0(\vec{x}, y_1, y_2, y_1 \circ y_2, w_2) = f'_1(\vec{x}, y_1, y_2, y_1 \circ y_2, w_2) \tag{82}$$

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

To prove Equation (82), 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 3.20) that both LHS and RHS of Equation (82) 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 3.20) 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:

$$\mathsf{Add}(x, \varepsilon) = \mathsf{Add}(\varepsilon, x) := x \tag{83}$$
$$\mathsf{Add}(x2, y2) := s_2(\mathsf{Succ}(\mathsf{Add}(x, y))) \tag{84}$$
$$\mathsf{Add}(x2, y1) = \mathsf{Add}(x1, y2) := s_1(\mathsf{Succ}(\mathsf{Add}(x, y))) \tag{85}$$
$$\mathsf{Add}(x1, y1) := s_2(\mathsf{Add}(x, y)) \tag{86}$$

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 3.21. *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 3.20 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

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

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:

$$\mathsf{Mul}(x,\varepsilon) = \mathsf{Mul}(\varepsilon,x) = \varepsilon \tag{87}$$
$$\mathsf{Mul}(x1,y1) := 4\cdot\mathsf{Mul}(x,y) + 2\cdot x + 2\cdot y + 1 \tag{88}$$
$$\mathsf{Mul}(x2,y1) := 4\cdot\mathsf{Mul}(x,y) + 2\cdot x + 4\cdot y + 2 \tag{89}$$
$$\mathsf{Mul}(x1,y2) := 4\cdot\mathsf{Mul}(x,y) + 4\cdot x + 2\cdot y + 2 \tag{90}$$
$$\mathsf{Mul}(x2,y2) := 4\cdot\mathsf{Mul}(x,y) + 4\cdot x + 4\cdot y + 4 \tag{91}$$

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 3.22. 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 3.20. 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 3.23. 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 3.20 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 3.24. $\mathsf{PV} \vdash \mathsf{EQ}(y_1, y_2) \Rightarrow y_1 = y_2$.

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 3.20 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

$$\begin{aligned}
&\mathsf{ITE}(\mathsf{EQ}(s_{i_1}(y_1), s_{i_2}(y_2)), s_{i_2}(y_2), s_{i_1}(y_1)) \\
&= h_{i_1 i_2}(y_1, y_2, \mathsf{ITE}(\mathsf{EQ}(y_1,y_2), y_2, y_1)). \qquad (92)
\end{aligned}$$

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 (92) 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 (92) is PV-provably equal to $s_{i_1}(y_1) = h_{i_1 i_2}(y_1, y_2, z)$ for any z. □

Theorem 3.25. $\mathsf{PV} \vdash \mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$ *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 3.24 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 $\mathsf{EQ}(s(\vec{x}), t(\vec{x})) = 1$.