

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* $\{\rightarrow, \wedge, \vee, \neg\}$ 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:

- An *atomic formula* of PV-PL is either \perp (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 \rightarrow for implication. This is without loss of generality, as we can define other propositional logic connectives from $\{\perp, \rightarrow\}$, for instance:

We will denote PV-PL formulas using Greek letters.

$$\neg\varphi := \varphi \rightarrow \perp; \quad \varphi \vee \psi := \neg\varphi \rightarrow \psi; \quad \varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi).$$

- 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:

$$\overline{\Gamma, \alpha \vdash \alpha}.$$

- (\rightarrow_i) . This rule is used to introduce an implication connective to the conclusion, formally:

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi}.$$

- (\rightarrow_e) . This rule is used to eliminate an implication connective in the conclusion, formally:

$$\frac{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$$

- (\perp_e) . This rule is used to eliminate a contradiction in the conclusion, which captures the standard “proof by contradiction” tactic:

$$\frac{\Gamma, \neg\varphi \vdash \perp}{\Gamma \vdash \varphi},$$

where $\neg\varphi$ is a shorthand of $\varphi \rightarrow \perp$.

Exercise: Define and prove the Explosion Rule from (\perp_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 symmetricity 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 $\Gamma \vdash e$.
- $(D\varepsilon)$. 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, \dots, x_n are variables, $n \in \mathbb{N}$. Then we have the following rule:

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

Note that the quantification over j and i_1, \dots, 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 $i_1, \dots, i_n \in \{0, 1\}$, 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 \perp_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 \perp 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, \dots, 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, \dots, x_n . That is, for every $i_1, \dots, i_n \in \{0, 1\}$, 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

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

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:

$$\frac{\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi} (\rightarrow_i) \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi} (\rightarrow_e)$$

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 ($=_j$) to prove that

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

Therefore by the rule of cut it suffices to prove that

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

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

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

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

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

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

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

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:

$$\vdash \text{EQ}(y_1, y_2) = 1 \rightarrow y_1 = y_2,$$

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

To prove this, we first apply the introduction rule of \rightarrow to move the assumption into the antecedent, and therefore it suffices to prove $\text{EQ}(y_1, y_2) = 1 \vdash y_1 = y_2$. Now we perform an induction on y_1 and y_2 simultaneously using (Ind₂). This introduces the following subgoals:

- (Base Case 1). $\text{EQ}(\varepsilon, y_2) = 1 \vdash \varepsilon = y_2$. 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). $\text{EQ}(\varepsilon, \varepsilon) = 1 \vdash \varepsilon = \varepsilon$. 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 $\text{EQ}(\varepsilon, s_i(x)) = 1 \vdash \varepsilon = s_i(x)$. Recall that by the definition axiom of EQ we have that $\text{PV} \vdash \text{EQ}(\varepsilon, s_i(x)) = 0$, and therefore by the (PV) rule as well as weakening we can obtain that

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

By the rule of cut, it suffices to prove that

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

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

$$\text{EQ}(\varepsilon, s_i(x)) = 1, \text{EQ}(\varepsilon, s_i(x)) = 0 \vdash \perp. \quad (94)$$

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

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

$$\begin{array}{ccc} \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi} (\wedge_e^1) & \frac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \psi} (\wedge_e^2) & \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} (\wedge_i) \\ \frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \vee \psi} (\vee_i^1) & \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \vee \psi} (\vee_i^2) & \frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \alpha \quad \Gamma, \psi \vdash \alpha}{\Gamma \vdash \beta} (\vee_e) \end{array}$$

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. 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 $\text{PV} \vdash e$.*

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 $t_c^{(1)}, \dots, t_c^{(\ell)}$, $\ell \in \mathbb{N}$, we define

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

as

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

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

We use $\text{RHS}[e]$ and $\text{LHS}[e]$ to represent the RHS and LHS of the conditional 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 $[\Gamma \vdash \varphi]_{\text{PV}}$, as a PV equation defined as follows:

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

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

where $[\alpha]_{\text{PV}}$ 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

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

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

$$\begin{aligned} [\perp]_{\text{PV}} &:= 0, \\ [t = s]_{\text{PV}} &:= “\text{EQ}(t, s)”. \end{aligned}$$

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

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

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

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

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

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

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

Indeed, we will prove that

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

If this is provable, we can derive Equation (95) by the substitution $x/[\psi_1]_{PV}$, $y/[\psi_2]_{PV}$ 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 IsNotEps and ITE . The details are omitted. \square

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

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 $PV \vdash [\Gamma \vdash \varphi]_{PV}$.*

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

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

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

Moreover, it suffices to prove that:

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

To see this, we can treat $[\beta_n]_{\text{PV}} \Rightarrow \dots \Rightarrow [\beta_1]_{\text{PV}} \Rightarrow [\varphi]_{\text{PV}} = 1$ as the equation $s = t$ and $[\alpha]_{\text{PV}}$ as the term t' , and that $\text{PV} \vdash \text{IsNotEps}([\alpha]_{\text{PV}}) = 1$ 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. $\text{PV} \vdash \text{IsNotEps}(z) \Rightarrow \text{EQ}(x, y) \Rightarrow z \Rightarrow x = y$.

It remains to prove the proposition.

Proposition 4.8. *PV proves that*

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

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

Proof of Proposition 4.7. We can prove this by a case study on z using Theorem 3.5. In case for z/ε , $\text{IsNotEps}(\varepsilon) = 0$ 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 $\text{IsNotEps}(s_i(z)) = 1$, and thus by Proposition 4.8

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

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

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

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

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

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

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 $\text{PV} \vdash [\Gamma, \alpha, \alpha \vdash \varphi]_{\text{PV}}$, then $\text{PV} \vdash [\Gamma, \alpha \vdash \varphi]_{\text{PV}}$.*

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 $\text{PV} \vdash \text{IsNotEps}(t_c) = 1$ and $\text{PV} \vdash t_c \Rightarrow t_1 = t_2$, then $\text{PV} \vdash t_c \Rightarrow t_c \Rightarrow t_1 = t_2$.*

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 $[e]_{\text{EQ}}$ to denote the term $\text{EQ}(\text{LHS}[e], \text{RHS}[e])$. We define the conditional equation $e \Rightarrow s = t$ as the equation $[e]_{\text{EQ}} \Rightarrow s = t$. Similarly, the equation

$$“(e_1 \Rightarrow e_2) \Rightarrow s = t” \equiv “[[e_1]_{\text{EQ}} \Rightarrow \text{LHS}[e_2] = \text{RHS}[e_2]]_{\text{EQ}} \Rightarrow s = t”.$$

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. $\text{PV} \vdash \text{IsNotEps}(z) \Rightarrow (x = y) \Rightarrow z \Rightarrow x = y$.

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 $\text{ITE}(s_1(z), y, x) = x$. It then suffices to prove that

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

which is the equation

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

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

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

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 $\text{PV} \vdash [\Gamma, \alpha, \beta, \Delta \vdash \varphi]_{\text{PV}}$, then $\text{PV} \vdash [\Gamma, \beta, \alpha, \Delta \vdash \varphi]_{\text{PV}}$.*

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

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

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

This can be easily derived from the following proposition:

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

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

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 $\text{IsNotEps}(z_2) = 0$ and thus by the Explosion Rule (see Proposition 3.11),

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

We can further add a dummy condition $\text{IsNotEps}(z_1)$ 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

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

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

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

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

then

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

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

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

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) , $(D\varepsilon)$, 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 , $PV \vdash [\Gamma \vdash e]_{PV}$.*

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

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

Proof. It suffices to prove that $PV \vdash [\vdash \varepsilon \neq s_i(x)]_{PV}$ and apply the admissibility of weakening. Notice that $[\vdash \varepsilon \neq s_i(x)]_{PV} \equiv \text{“EQ}(\varepsilon, s_i(x)) \Rightarrow 0 = 1\text{”}$. 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). \square

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

Proof. It suffices to prove that $PV \vdash [\vdash s_0(x) \neq s_1(y)]_{PV}$ and apply the admissibility of weakening. Notice that $[\vdash s_0(x) \neq s_1(y)]_{PV} \equiv \text{“EQ}(s_0(x), s_1(y)) \Rightarrow 0 = 1\text{”}$. 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). \square

4.5.2 Logical Axioms for Equality

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

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

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

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

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

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). \square

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

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

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

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:

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

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. \square

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

Proof Sketch. The “moreover” part follows straightforwardly, so we will only prove the first part. To show that $\text{PV} \vdash \text{IsNotEps}(x) \Rightarrow (x \Rightarrow y = z) \Rightarrow x \Rightarrow \text{EQ}(y, z) = 1$, 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 $\text{EQ}(y, z) \Rightarrow \text{EQ}(y, z) = 1$, which can be proved using the fact that

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

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

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

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

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

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

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

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

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

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

Auxiliary functions. Let $EQL(x, y)$ 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

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

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

$$\begin{aligned} Suf(x, \varepsilon) &:= \varepsilon, & Suf(\varepsilon, y) &:= \varepsilon \\ Suf(s_i(x), s_j(y)) &:= s_i(Suf(x, y)). \end{aligned}$$

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

Proposition 4.25. *PV proves the following equations:*

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

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 $EQL(s_0(y), s_1(y)) = 1$ and Modus Ponens (see Proposition 3.10) to prove that $Suf(x, s_0(y)) = Suf(x, s_1(y))$. The last bullet can be proved by an induction on y and using the second and the third bullets to prove that $Suf(x, x \circ \varepsilon) = x$ and $Suf(x, x \circ s_i(y)) = Suf(x, x \circ y)$. \square

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

A generalized equation. We will show that PV proves:

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

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

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

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:

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

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

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

Also, notice that

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

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 $\text{LastBit}(\text{ITR}(x, z)) = \text{LastBit}(\text{ITR}(y, z))$. Formally, let z' be a fresh variable, and we will prove that

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

To see that this suffices, we can substitute

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

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 $\text{LastBit}(z') \neq \text{LastBit}(z'')$, both sides of the consequence of the conditional equation evaluates to $f_t(x, \vec{w})$. Now we consider the cases that $\text{LastBit}(z') = \text{LastBit}(z'')$, for instance, with the substitution $z'/s_0(z')$ and $z''/s_0(z'')$. In such case, the LHS of the equation evaluates to

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

while the RHS evaluates to

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

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. \square

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 $\{\rightarrow, \perp\}$.

4.6.1 Assumption and Substitution

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

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

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

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

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

Proof Sketch. It suffices to prove that substitution commutes with the PV translation, i.e., $[\Gamma[z/t] \vdash \alpha[z/t]]_{\text{PV}} \equiv [\Gamma \vdash \alpha]_{\text{PV}}[z/t]$, 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 $\Gamma = (\beta_1, \dots, \beta_n)$. Notice that

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

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

4.6.2 Rules of Implication

Next, we show that the introduction (\rightarrow_i) and elimination (\rightarrow_e) rules of implication are PV admissible.

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

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

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

$$\text{PV} \vdash [\beta_n]_{\text{PV}} \Rightarrow \dots \Rightarrow [\beta_1]_{\text{PV}} \Rightarrow [\varphi]_{\text{PV}} \Rightarrow [\psi]_{\text{PV}} = 1. \quad (99)$$

and we need to prove that

$$\text{PV} \vdash [\beta_n]_{\text{PV}} \Rightarrow \dots \Rightarrow [\beta_1]_{\text{PV}} \Rightarrow \text{ITE}([\varphi]_{\text{PV}}, [\psi]_{\text{PV}}, 1) = 1. \quad (100)$$

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

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

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

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

where the antecedent (and thus its $[\cdot]_{\text{EQ}}$ 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:

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

To prove this, we perform 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

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

we will show that PV proves

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

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

$$\begin{aligned} & \text{IsNotEps}(z_{n+1}) \Rightarrow \dots \Rightarrow \text{IsNotEps}(z_1) \Rightarrow \\ & (z_{n+1} \Rightarrow \dots \Rightarrow z_1 \Rightarrow 1 \Rightarrow 0 = 1) \Rightarrow z_{n+1} \Rightarrow \dots \Rightarrow z_1 \Rightarrow 0 = 1. \end{aligned} \quad (101)$$

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

$$\begin{aligned} & \text{IsNotEps}(z_n) \Rightarrow \dots \Rightarrow \text{IsNotEps}(z_1) \Rightarrow \\ & (z_n \Rightarrow \dots \Rightarrow z_1 \Rightarrow 1 \Rightarrow 0 = 1) \Rightarrow z_n \Rightarrow \dots \Rightarrow z_1 \Rightarrow 0 = 1, \end{aligned} \quad (102)$$

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

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

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

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

To see that this suffices, notice that we can substitute $x/[\varphi]_{\text{PV}}$, $y/[\psi]_{\text{PV}}$, 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). \square

4.6.3 Rules of Contradiction

The last logical rule we need to consider is the elimination rule of \perp , i.e., “proof by contradiction”.

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

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

$$\text{PV} \vdash [\Gamma \vdash \neg\neg\alpha \rightarrow \alpha]_{\text{PV}}.$$

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

$$\frac{\frac{}{\Gamma \vdash \neg\neg\alpha \rightarrow \alpha} \quad \frac{\Gamma, \neg\alpha \vdash \perp}{\Gamma \vdash \neg\neg\alpha} (\rightarrow_i)}{\Gamma \vdash \alpha} (\rightarrow_e)$$

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

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

Furthermore, it suffices to prove the more general version:

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

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.

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 (Ind₂); nevertheless, the admissibility of (Ind_n) clearly follows from the same technique.

Lemma 4.32. *The structural induction rule (Ind₂) is admissible. That is, for any terms t_1, t_2 , Γ , and variables x_1, x_2 , suppose that PV proves*

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

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

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

Proof. By the admissibility of the permutation rule, we can change the second bullet to $[t_1 = t_2, \Gamma \vdash t_1[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)] = t_2[x_1/s_{i_1}(x_1), x_2/s_{i_2}(x_2)]]_{\text{PV}}$. 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 $\Gamma = (\beta_1, \dots, \beta_n)$. 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, \dots, u_n , if PV proves

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

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

Base case. We first consider the case $n = 1$. The case for $\Gamma = \emptyset$ 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 ,

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

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

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

We will prove the alternative equation:

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

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

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 $h_{i_1 i_2}(x, y, z) = \text{ITE}(z, 1, 0)$ 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:

- $\text{ITE}(u, \text{EQ}(f_1(\varepsilon, \varepsilon), f_2(\varepsilon, \varepsilon)), 1) = g_{00} = 1$,
- $\text{ITE}(u, \text{EQ}(f_1(\varepsilon, s_j(x_2)), f_2(\varepsilon, s_j(x_2))), 1) = g_{01}(s_j(x_2)) = 1$,
- $\text{ITE}(u, \text{EQ}(f_1(s_j(x_1), \varepsilon), f_2(s_j(x_1), \varepsilon)), 1) = g_{10}(s_j(x_1)) = 1$,

from the assumption. Therefore, it suffices to prove that

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

We will first prove a more general result:

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

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

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

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

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

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 $\text{PV} \vdash \text{IsNotEps}(t_c) = 1$ and $\text{PV} \vdash \text{IsNotEps}(t'_c) = 1$. Then $\text{PV} \vdash t_c \Rightarrow t'_c \Rightarrow t_1 = t_2$ if and only if $\text{PV} \vdash \text{And}(t_c, t'_c) \Rightarrow t_1 = t_2$.*

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

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

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

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

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 $\text{And}(t_c, t'_c) \Rightarrow t_1 = t_2$ and completes the proof. \square

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

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

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 $(\hat{=}/_)$:

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

(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 \perp or a PV equation $u = v$. Note that the case for $\varphi \equiv \perp$ 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 $\beta \rightarrow \alpha$ and $\varphi' \equiv \beta' \rightarrow \alpha'$. 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 $(\hat{=}/_)$ with φ being the substitution formula, i.e.,

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

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

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

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

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

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

$$\frac{\frac{\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta \vdash \beta \rightarrow \alpha}{\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta \vdash \beta} (A) \quad \frac{\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta \vdash \beta}{\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta \vdash \alpha} (A)}{\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta \vdash \alpha} (\rightarrow_e)$$

(Subgoal 2.2): $\Gamma, x = y, \beta \rightarrow \alpha, \beta', \beta, \alpha \vdash \alpha'$. 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. \square

Comprehension of PV translation. Recall that the PV translation could translate a formula φ in PV-PL into a PV term $[\varphi]_{\text{PV}}$. 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 ($C^{-1}\text{PV}$):

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

Lemma 4.37. *Both (CPV) and ($C^{-1}\text{PV}$) 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 $\varphi = \perp$ is trivial. Suppose that φ is of form $u = v$, then (CPV) follows from the rewrite rule as well as $\text{PV} \vdash \text{EQ}(x, x) = 1$, while ($C^{-1}\text{PV}$) follows from the correctness of EQ as demonstrated in Example 4.2.

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

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

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

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

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

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

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

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

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 $\Gamma \vdash [\varphi]_{\mathbf{PV}} = 1$ from the premises. By the induction rule (Ind₁) in **PV-PL**, it suffices to prove that

- $\Gamma \vdash [\varphi]_{\mathbf{PV}}[z/\varepsilon] = 1$
- $\Gamma, [\varphi]_{\mathbf{PV}} = 1 \vdash [\varphi]_{\mathbf{PV}}[z/s_i(z)] = 1$ 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

- $\Gamma \vdash [\varphi[z/\varepsilon]]_{\mathbf{PV}} = 1,$
- $\Gamma, [\varphi]_{\mathbf{PV}} = 1 \vdash [\varphi[z/s_i(z)]]_{\mathbf{PV}} = 1$ 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: **Append**(ℓ, x) (also denoted by $\ell :: x$) appends a string x to the end of a list ℓ , **Head**(ℓ), **Tail**(ℓ) and proves in PV that

$$\text{Head}(\varepsilon) = \varepsilon, \quad \text{Head}(\ell :: x) = x \quad (107)$$

$$\text{Tail}(\varepsilon) = \varepsilon, \quad \text{Tail}(\ell :: x) = \ell \quad (108)$$

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:

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

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

Indeed the construction is similar to tuples. We define:

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

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

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

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

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

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

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. $\varepsilon :: \varepsilon$).

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

$$\text{PV} \vdash \text{GenList}(v, \varepsilon) = \varepsilon \quad (111)$$

$$\text{PV} \vdash \text{GenList}(v, s_i(y)) = \text{GenList}(v, y) :: v \quad (112)$$

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 $k_i(v, y) = 11 \circ \text{PEnc}(v)$, we can verify that

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

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

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 $\text{GenList}(v, y) := f(v, y)$, it will satisfy Equation (111) and (112).

We also note that by composing GenList and $\#$, we can initiate a list of arbitrary polynomial length. For instance, $\text{GenList}(v, y \# y \# y)$ 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*

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

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

$$f(\vec{x}, \varepsilon) = g(\vec{x}) \quad (113)$$

$$f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f(\vec{x}, \ell)) \quad (114)$$

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

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

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

$$\text{PV} \vdash \text{IsList}'(\varepsilon) = 1 \quad (115)$$

$$\text{PV} \vdash \text{IsList}'(\ell :: u) = \text{And}(\text{Or}(\text{IsEps}(\ell), \text{EQ}(\ell, \text{Tail}(\ell) :: \text{Head}(\ell))), \text{IsList}'(\ell)). \quad (116)$$

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

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

Proof Sketch. Both assertion can be proved by simple unfolding. \square

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

$$\text{PV} \vdash \text{Len}(\varepsilon) = 0 \quad (117)$$

$$\text{PV} \vdash \text{Len}(\ell :: u) = s_0(\text{Len}(\ell)) \quad (118)$$

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

$$\text{PV} \vdash \text{Concat}(\ell_1, \varepsilon) = \ell_1 \quad (119)$$

$$\text{PV} \vdash \text{Concat}(\ell_1, \ell_2 :: u) = \text{Concat}(\ell_1, \ell_2) :: u \quad (120)$$

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

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

$$\text{PV} \vdash \text{Map}_f(\varepsilon) = \varepsilon$$

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

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

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, $\text{Find}_f(\ell)$ will satisfy that

$$\text{PV} \vdash \text{Find}_f(\varepsilon) = \varepsilon$$

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

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:*

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

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:*

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

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

$$\text{IsList}(\ell_3) = 1 \vdash (\ell_1 ++ \ell_2) ++ \ell_3 = \ell_1 ++ (\ell_2 ++ \ell_3).$$

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

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

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

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

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

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

$$(\ell_1 ++ \ell_2) ++ (\ell_3 :: u) = \ell_1 ++ (\ell_2 ++ (\ell_3 :: u))$$

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 $\text{Lookup}(\gamma, x)$ and $\text{Update}(\gamma, x, y)$ as follows:

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

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

Intuitively, $\text{Lookup}(\gamma, x)$ finds the value corresponding to the key x by searching for the rightmost (i.e. latest) pair of form (x, \cdot) , and $\text{Update}(\gamma, x, y)$ 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:

$$\Gamma \vdash \text{Lookup}(\varepsilon, x) = \varepsilon; \quad (123)$$

$$\Gamma, \text{Lookup}(\gamma, x) = y, x' \neq x \vdash \text{Lookup}(\text{Update}(\gamma, x', y'), x) = y; \quad (124)$$

$$\Gamma \vdash \text{Lookup}(\text{Update}(\gamma, x, y), x) = y. \quad (125)$$

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 $\text{Lookup}(\gamma :: \text{MakePair}(x', y')) = y$, and by further unfolding **Lookup** and Find_f , it suffices to prove from $\Gamma, \text{Lookup}(\gamma, x) = y, x' \neq x$ that

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

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

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

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 $\text{EQ}(x', x) = 0$, and therefore by the cut rule, it suffices to prove the equation above from the antecedent $\Gamma, \text{Lookup}(\gamma, x) = y, x' \neq x, \text{EQ}(x', x) = 0$. Using the substitution/generalization rule, it suffices to prove

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

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

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

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

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

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

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

Proposition 5.8. PV-PL proves the following assertions:

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

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:*

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

The proof of Theorem 5.5 is rather straightforward. Intuitively, we will prove by induction on the length of ℓ using the (Ind₁) rule in PV-PL, where the length is calculated by the function $\text{Len}(\ell)$ 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

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

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

To prove the assertion above, we apply (Ind₁) to the variable, which requires proofs of the following three subgoals:

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

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

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

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

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

which leads to a contradiction by (D ε). 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 $\ell = \text{Tail}(\ell) :: \text{Head}(\ell)$, so that $\text{Len}(\ell) = s_0(\text{Len}(\text{Tail}(\ell))) \neq s_1(z)$. Therefore we can prove \perp from the assumptions, and subsequently prove $t_1 = t_2$ by the explosion rule, which is a special case of (\perp_e).

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

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

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

which is left as an exercise.

Recall that by the premises we know that

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

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

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

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

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

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*

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

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 $y' := \text{ITR}(\ell, y)$ times. Note that the function $\text{ITRL}(\ell, y)$ 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 $\text{ITRL}(\ell, \ell) = \varepsilon$.

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

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

where $\hat{\ell} := \text{ITRL}(\ell, \text{ITR}(\ell, s_i(y)))$. 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 $\text{ITRL}(\ell, y)$ be the function defined as

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

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

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

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

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

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

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

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

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

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

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

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

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

The first equation follows from $\text{ITR}(x, x) = \varepsilon$, 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

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

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

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

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

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

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

$$\text{ITRL}(\ell :: u, s_0(s_0(y))) = \text{Tail}(\text{ITRL}(\ell :: u, s_0(y)))$$

and

$$\text{ITRL}(\ell, s_0(y)) = \text{Tail}(\text{ITRL}(\ell, y)).$$

Since $\text{ITRL}(\ell :: u, s_0(y)) = \text{ITRL}(\ell, y)$ by the assumption hypothesis, we can prove the induction case by applying the PV-PL axioms of equality. \square

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

$$f'(\vec{x}, \ell :: u, \text{TR}(\ell :: u)) = f'(\vec{x}, \ell, \ell)$$

where the RHS is $f(\vec{x}, \ell)$. We will then show that

$$f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(\ell :: u))),$$

which, together with the fact that $f'(\vec{x}, \ell :: u, \text{TR}(\ell :: u)) = f(\vec{x}, \ell)$, concludes the proof of Equation (114).

Proposition 5.10. PV proves that

$$f'(\vec{x}, \ell :: u, y) = \begin{cases} g(\vec{x}) & \text{Or}(\text{IsEps}(\hat{\ell}), \text{IsEps}(y)) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(y))) & \text{otherwise} \end{cases}$$

where $\hat{\ell} := \text{ITRL}(\ell :: u, \text{ITR}(\ell :: u, y))$.

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. \square

Proposition 5.11. $\text{PV} \vdash f(\vec{x}, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(\ell :: u)))$.

Proof Sketch. Note that it is easy to prove that $\text{PV} \vdash \text{IsNotEps}(\ell :: u) = 1$ by unfolding the definition of **Append**. Moreover, let $\hat{\ell} = \text{ITRL}(\ell :: u, \text{ITR}(\ell :: u, \ell :: u))$, we know that $\hat{\ell} = \ell :: u$ and thus we can prove that $\text{IsEps}(\hat{\ell}) = 0$. By Proposition 5.10 with $y/\ell :: u$ and unfolding ITE's (used to implement the case analysis), we can conclude that

$$f'(\vec{x}, \ell :: u, \ell :: u) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(\ell :: u))),$$

where the LHS is exactly $f(\vec{x}, \ell :: u)$ by the definition axiom of f . \square

Proof of Equation (126). We will prove Equation (126) by performing an induction on $\text{ITR}(\ell, y)$. Concretely, let y' be a fresh variable, we will prove the equation where y is substituted by $\text{ITR}(\ell, y')$ by induction on y' .

Note that this will actually lead to a proof of

$$f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, y')))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, y'))) \quad (127)$$

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 $\text{EQL}(x, y)$ 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:*

- $\text{IsNotEps}(\text{ITR}(x, y)) \vdash \text{EQL}(\text{ITR}(x, \text{ITR}(x, y)), y) = 1$.
- $\text{EQL}(x, y) \vdash \text{ITRL}(\ell, x) = \text{ITRL}(\ell, y)$.
- $\text{EQL}(x, y) \vdash f'(\vec{z}, \ell, x) = f'(\vec{z}, \ell, y)$.

Proof Sketch. All the equations can be proved by applying induction on x and y using the induction rule of PV-PL. \square

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). $\text{PV} \vdash f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\ell))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \ell))$.

Proof Sketch. Since $\text{PV} \vdash \text{ITR}(\ell, \ell) = \varepsilon$, 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 $\text{PV} \vdash \text{IsEps}(\text{ITR}(\ell :: u, s_0(\ell))) = 0$ by unfolding $\ell :: u$ and proving basic properties about ITR. By Proposition 5.10, we know that

$$\begin{aligned} & f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\ell))) \\ &= \begin{cases} g(\vec{x}) & \text{Or}(\text{IsEps}(\hat{\ell}), \text{IsEps}(\text{ITR}(\ell :: u, s_0(\ell)))) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(\text{ITR}(\ell :: u, s_0(\ell)))) & \text{otherwise} \end{cases} \end{aligned}$$

where $\hat{\ell} := \text{ITRL}(\ell :: u, \text{ITR}(\ell :: u, \text{TR}(\text{ITR}(\ell :: u, s_0(\ell)))))$. Note that

$$\text{EQL}(\text{ITR}(\ell :: u, \text{ITR}(\ell :: u, s_0(\ell))), s_0(\ell)) = 1,$$

and the proof is left as an exercise. By $\text{EQL}(x, y) = 1 \Rightarrow \text{ITRL}(\ell, x) = \text{ITRL}(\ell, y)$ we can further prove that

$$\hat{\ell} = \text{ITRL}(\ell :: u, s_0(\ell)),$$

which is further equal to $\text{ITRL}(\ell, \ell) = \varepsilon$. Therefore by unfolding the equation from Proposition 5.10 we have $f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\ell))) = g(\vec{x})$. \square

Hint: Using Proposition 5.12

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 $\hat{\ell} = \text{ITRL}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))$ as an assumption (aka. in the antecedent). Note that by unfolding ITRL we can prove in PV that $\hat{\ell} = \text{ITRL}(\ell, \text{ITR}(\ell, s_i(y')))$.

Introduction of fresh variable can be done by (V), (A), and (Cut).

We will perform a case analysis on whether $\text{IsEps}(\hat{\ell}) = 1$. Concretely, let Γ be the set of antecedents including:

- $\hat{\ell} = \text{ITRL}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))$
- $\hat{\ell} = \text{ITRL}(\ell, \text{ITR}(\ell, s_i(y')))$
- $f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y'))))$ (i.e. Equation (127)).

We will prove the following two lemmas:

Lemma 5.14. PV-PL proves $\Gamma, \text{IsEps}(\hat{\ell}) = 1 \vdash$

$$f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))).$$

Lemma 5.15. PV-PL proves $\Gamma, \text{IsEps}(\hat{\ell}) = 0 \vdash$

$$f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))).$$

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

$$\begin{aligned} & f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) \\ &= \begin{cases} g(\vec{x}) & \text{Or}(\text{IsEps}(\hat{\ell}'), \text{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(y''))) & \text{otherwise} \end{cases} \end{aligned}$$

where

$$\begin{aligned} y'' &:= \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y')))), \\ \hat{\ell}' &:= \text{ITRL}(\ell :: u, \text{ITR}(\ell :: u, y'')). \end{aligned}$$

Note that we can prove $\text{EQL}(\text{ITR}(\ell :: u, y''), s_0(\text{ITR}(\ell, s_i(y')))) = 1$ (details omitted), so that by Proposition 5.12:

$$\hat{\ell}' = \text{ITRL}(\ell :: u, \text{ITR}(\ell :: u, y'')) = \text{ITRL}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y')))) = \hat{\ell}.$$

Since $\text{IsEps}(\hat{\ell}) = 1$ is available in the antecedent, we can apply the rewrite rule ($\hat{=}$), unfold Or and ITE, and conclude that

$$f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) = g(\vec{x}).$$

(RHS). Again, by Proposition 5.10, we know that

$$\begin{aligned} & f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))) \\ &= \begin{cases} g(x) & \text{Or}(\text{IsEps}(\hat{\ell}'), \text{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell, \text{TR}(y''))) & \text{otherwise} \end{cases} \end{aligned}$$

where

$$\begin{aligned} y'' &:= \text{ITR}(\ell, \text{ITR}(\ell, s_i(y'))), \\ \hat{\ell}' &:= \text{ITRL}(\ell, \text{ITR}(\ell, y'')). \end{aligned}$$

Note that we can prove $\text{EQL}(\text{ITR}(\ell, y''), \text{ITR}(\ell, s_i(y'))) = 1$ (details omitted), so that by Proposition [5.12](#):

$$\hat{\ell}' = \text{ITRL}(\ell, \text{ITR}(\ell, y'')) = \text{ITRL}(\ell, \text{ITR}(\ell, s_i(y'))) = \hat{\ell}.$$

Since $\text{IsEps}(\hat{\ell}) = 1$ is available in the antecedent, we can apply the rewrite rule ($\hat{=}$), unfold Or and ITE, and conclude that

$$f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))) = g(\vec{x}).$$

This completes the proof. \square

Proof of Lemma [5.15](#). Similar to the proof of Lemma [5.14](#), we can apply Proposition [5.10](#) to show that:

$$\begin{aligned} &f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) \\ &= \begin{cases} g(\vec{x}) & \text{Or}(\text{IsEps}(\hat{\ell}'), \text{IsEps}(y'')) \\ h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(y''))) & \text{otherwise} \end{cases} \end{aligned}$$

where it is provable that $\hat{\ell}' = \hat{\ell}$, and $y'' := \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))$. It can be proved that $\text{IsEps}(y'') = 0$ as $\text{IsEps}(\text{ITR}(\ell :: u, s_0(\ell))) = 0$. Therefore, by the assumption that $\text{IsEps}(\hat{\ell}) = 0$, we can conclude by applying the rewrite rule and unfolding that:

$$\begin{aligned} &f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))) \\ &= h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \text{TR}(\text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y')))))). \end{aligned} \quad (128)$$

Similarly, we can also prove that

$$\begin{aligned} &f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))) \\ &= h(\vec{x}, u, \ell, f'(\vec{x}, \ell, \text{TR}(\text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))))). \end{aligned} \quad (129)$$

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

$$\begin{aligned} &f'(\vec{x}, \ell :: u, \text{TR}(\text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y')))))) \\ &= f'(\vec{x}, \ell, \text{TR}(\text{ITR}(\ell, \text{ITR}(\ell, s_i(y'))))). \end{aligned} \quad (130)$$

Finally, notice that PV proves:

$$\begin{aligned} &\text{EQL}(\text{TR}(\text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, s_i(y'))))), \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, y')))) = 1, \\ &\text{EQL}(\text{TR}(\text{ITR}(\ell, \text{ITR}(\ell, s_i(y')))), \text{ITR}(\ell, \text{ITR}(\ell, y'))) = 1. \end{aligned}$$

(The detail is omitted.) We can therefore modify our goal Equation [\(130\)](#) using Proposition [5.12](#) and the rewrite rule to:

$$f'(\vec{x}, \ell :: u, \text{ITR}(\ell :: u, s_0(\text{ITR}(\ell, y')))) = f'(\vec{x}, \ell, \text{ITR}(\ell, \text{ITR}(\ell, y'))),$$

which is available as the third bullet of the antecedent (see the definition of Γ above). \square

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:

$$\begin{aligned} \text{Cmd} := & \mathbf{skip} \mid \mathbf{let } x := \text{Exp} \mid \text{Cmd}; \text{Cmd} \\ & \mid \mathbf{if } \text{Exp} \mathbf{ then } \text{Cmd} \mathbf{ else } \text{Cmd} \\ & \mid \mathbf{for } x := \text{Exp}; x \neq \varepsilon; x := \text{TR}(x) \mathbf{ do } \text{Cmd} \end{aligned}$$

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 $\text{Suf}(y, \omega)$, 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 $c \in \text{Cmd}$ 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:

$$\overline{\{\varphi\} \text{ skip } \{\varphi\}}$$

- Assignment axiom scheme:

$$\overline{\{\varphi[x/e]\} \text{ let } x := e \{\varphi\}}$$

- Rule of composition:

$$\frac{\{\varphi_1\} c_1 \{\psi\} \quad \{\psi\} c_2 \{\varphi\}}{\{\varphi_1\} c_1; c_2 \{\varphi\}}$$

- Rule of if-then-else:

$$\frac{\{\varphi_1 \wedge \text{LastBit}(e) = 1\} c_1 \{\varphi_2\} \quad \{\varphi_1 \wedge \text{LastBit}(e) \neq 1\} c_0 \{\varphi_2\}}{\{\varphi_1\} \text{ if } e \text{ then } c_1 \text{ else } c_0 \{\varphi_2\}}$$

$\varphi \wedge \varphi_2$ is indeed defined using connectives $\{\rightarrow, \neg\}$ in PV-PL.

- Rule of for-loop: For $i \in \{0, 1\}$, we have

$$\frac{\{\varphi[x/s_i(x)]\} c \{\varphi\}}{\{\varphi[x/e]\} \text{ for } x := e; x \neq \varepsilon; x := \text{TR}(x) \text{ do } c \{\varphi[x/\varepsilon]\}}$$

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 behavior 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:

```

LenLTR := let z := ℓ;
          let y := ε;
          for x := ℓ; x ≠ ε; x := TR(x) do
            if IsEps(z) then skip else let y := s0(y); let z := Tail(z)

```

We can formalize its correctness as the Hoare tuple

$$\{\text{IsList}(\ell) = 1\} \text{LenITR} \{y = \text{Len}(\ell)\}, \quad (131)$$

where Len is the function defined by Equation (117) and (118).

It can be proved using the loop invariant:

$$\varphi := \bigwedge \{\text{IsList}(z) = 1, \text{Len}(z) \circ y = \text{Len}(\ell), \text{ITR}(x, \ell) = \varepsilon, z = \text{ITRL}(\ell, \text{ITR}(\ell, x))\},$$

where $\text{ITRL}(\ell, y)$ is the function defined as

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

More formally, let LenITR_3 be the for-loop in LenITR , we will prove the Hoare tuple $\{\varphi[x/\ell]\} \text{LenITR}_3 \{\varphi[x/\varepsilon]\}$. It is left as an exercise that Equation (131) can be derived from this Hoare tuple.

To prove $\{\varphi[x/\ell]\} \text{LenITR}_3 \{\varphi[x/\varepsilon]\}$, 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

$$\{\varphi[x/s_i(x)]\} \text{ if IsEps}(z) \text{ then skip else let } y := s_0(y); \text{ let } z := \text{Tail}(z) \{\varphi\}.$$

Subsequently, we can apply the *rule of if-then-else*, so that it suffices to prove the following two subgoals:

- (Subgoal 1). $\{\varphi[x/s_i(x)] \wedge \text{LastBit}(\text{IsEps}(z)) = 1\} \text{ skip } \{\varphi\}$
- (Subgoal 2). $\{\varphi[x/s_i(x)] \wedge \text{LastBit}(\text{IsEps}(z)) \neq 1\} \text{ let } y := s_0(y); \text{ let } z := \text{Tail}(z) \{\varphi\}.$

To prove (Subgoal 1), it suffices to prove in PV-PL that

$$\varphi[x/s_i(x)] \wedge \text{LastBit}(\text{IsEps}(z)) = 1 \vdash \varphi,$$

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

$$\varphi[x/s_i(x)] \wedge \text{LastBit}(\text{IsEps}(z)) \neq 1 \vdash \varphi[z/\text{Tail}(z)][y/s_0(y)],$$

apply the *consequence rule* to weaken the precondition to $\varphi[z/\text{Tail}(z)][y/s_0(y)]$, and subsequently apply the *rule of composition* (with $\psi := \varphi[z/\text{Tail}(z)]$) 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 $\text{Suf}(e, \omega)$. 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) $\text{IMP}(\text{PV})$ 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 $\text{IMP}(\text{PV})$ program. We define its PV translation, denoted also by $[P]_{\text{PV}}$, 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, \dots, x_k , we will use a k -tuple π of length k to store the context, i.e., $\text{UnwindTuple}_i(\pi)$ stores the value of x_k . For simplicity, we will denote $\text{UnwindTuple}_i(\pi)$ simply by π_i .
- (*Expressions*). For an expression e in P , where the variables are from x_1, \dots, x_k , we define its PV translation with respect to the context π , denoted by $[e]_{\text{PV}}^{\pi, \omega}$, as $\text{Suf}(e[x_i/\pi_i \ \forall i \in [k]], \omega)$. We may omit ω in the superscript in case that there is no ambiguity.
- (*Empty statement*). We define $[\text{skip}]_{\text{PV}}(\omega, \pi) := \pi$, i.e., the identity function.
- (*Assignment*). We define $[\text{let } x_i := e]_{\text{PV}}(\omega, \pi)$ be the function that outputs the tuple $\text{MakeTuple}(\pi_1, \dots, \pi_{i-1}, [e]_{\text{PV}}^{\pi, \omega}, \pi_{i+1}, \dots, \pi_k)$.
- (*Composition*). We define $[c_1; c_2]_{\text{PV}}(\omega, \pi) := [c_2]_{\text{PV}}([c_1]_{\text{PV}}(\pi))$, i.e., the PV translation commutes with composition of $\text{IMP}(\text{PV})$ programs.
- (*If-then-else*). We define the PV translation of if-then-else statement as

$$[\text{if } e \text{ then } c_1 \text{ else } c_0]_{\text{PV}}(\pi) := \text{ITE}_0^{(\varepsilon)}([e]_{\text{PV}}^{\pi, \omega}, [c_1]_{\text{PV}}(\pi), [c_2]_{\text{PV}}(\pi)),$$

where $\text{ITE}_0^{(\varepsilon)}$ is the “dirty” if-then-else function.

- (*For Loop*). We define $[\text{for } x_i := e; x_i \neq \varepsilon; x_i := \text{TR}(x_i) \text{ do } c]_{\text{PV}}$ as the function $f(\omega, \pi)$ constructed below:

- Let $f_{\text{iter}}(\omega, \pi) := \text{MakeTuple}(\pi'_1, \dots, \pi'_{i-1}, \text{TR}(\pi'_i), \pi'_{i+1}, \dots, \pi'_k)$, where $\pi' = [c]_{\text{PV}}(\omega, \pi)$.
- Let $g(\omega, \pi) = \pi$, $h_i(\omega, \pi, y, z) = \pi'$, $k_i(\omega, \pi, y, z) = \pi''$, $i \in \{0, 1\}$, where

$$\begin{aligned} \pi' &:= \text{MakeTuple}^{(k)}(\text{Suf}(f_{\text{iter}}(\omega, z)_1, \omega), \dots, \text{Suf}(f_{\text{iter}}(\omega, z)_k, \omega)) \\ \pi'' &:= \text{MakeTuple}^{(k)}(\omega, \omega, \dots, \omega) \end{aligned}$$

It can be proved in PV that $\text{ITR}(h_i(\omega, \pi, y, z), z \circ k_i(\omega, \pi, y, z)) = \varepsilon$ 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

$$\begin{aligned} f_{\text{loop}}(\omega, \pi, \varepsilon) &= g(\omega, \pi) \\ f_{\text{loop}}(\omega, \pi, s_i(y)) &= h_i(\omega, \pi, y, f_{\text{loop}}(\omega, \pi, y)) \end{aligned}$$

- Let $f(\omega, \pi) = f_{\text{loop}}(\omega, \text{MakeTuple}(\pi_1, \dots, \pi_{i-1}, [e]_{\text{PV}}^{\omega, \pi}, \pi_{i+1}, \dots, \pi_k), [e]_{\text{PV}}^{\omega, \pi})$.

6.1.4 Translating of Hoare Proofs to PV Proofs

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.

Hopefully feasible mathematicians will be satisfied with writing even less formal (pseudo) proofs after developing this result :)

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, \dots, 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 $\pi_i (= \text{UnwindTuple}_i^{(k)}(\pi))$. Then a Hoare tuple $\{\varphi_1\} c \{\varphi_2\}$ is formalized as the PV-PL assertion

$$\varphi_1 \vdash \varphi_2^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))},$$

which is denoted as $[\{\varphi_1\} c \{\varphi_2\}]_{\text{PV-PL}}$.

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*

$$\varphi_1 \vdash \varphi_2^{[P]_{\text{PV}}(\omega, \text{MakeTuple}(x_1, \dots, x_k))}.$$

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:*

$$\frac{\varphi_1 \vdash \varphi'_1 \quad \varphi'_2 \vdash \varphi_2 \quad \varphi'_1 \vdash \varphi_2'^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))}}{\varphi_1 \vdash \varphi_2^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))}}$$

Proof. Note that from the premises, it suffices to derive that

$$\varphi_2'^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))} \vdash \varphi_2^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))}$$

and apply the transitivity of PV-PL assertion. Recall that $\varphi_2'^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))}$ (resp. $\varphi_2^{[c]_{\text{PV}}(\text{MakeTuple}(x_1, \dots, x_k))}$) is obtained from φ'_2 (resp. φ_2) by substituting

$$x_i / \text{UnwindTuple}_i(\text{MakeTuple}(x_1, \dots, x_k))$$

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. \square

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:*

$$\overline{\varphi[x_i / \text{Suf}(e, \omega)]} \vdash \varphi^{[\text{let } x_i := e]_{\text{PV}}(\omega, \text{MakeTuple}(x_1, \dots, x_k))}$$

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

$$\begin{aligned} & [\text{let } x_i := e]_{\text{PV}}(\omega, \text{MakeTuple}(x_1, \dots, x_k)) \\ &= \text{MakeTuple}(x_1, \dots, x_{i-1}, \text{Suf}(e, \omega), x_{i+1}, \dots, x_k). \end{aligned}$$

Moreover, notice that PV also proves

$$\varphi^{\text{MakeTuple}(x_1, \dots, x_{i-1}, \text{Suf}([e], \omega), x_{i+1}, \dots, x_k)} = \varphi[x_i / \text{Suf}(e, \omega)].$$

Therefore the axiom scheme is provable by rewriting the consequence to $\varphi[x_i / \text{Suf}(e, \omega)]$ (using Lemma 4.36) and then apply the assumption axiom of PV-PL. \square

Lemma 6.4. *Let c be an IMP(PV) program and P be the program **for** $x_i := e$; $x_i \neq \varepsilon$; $x_i := \text{TR}(x_i)$ **do** c . The rule of for-loop, formalized as follows, is admissible in PV-PL:*

$$\frac{\varphi[x_i / s_i(x_i)] \vdash \varphi^{[c]_{\text{PV}}(\omega, \text{MakeTuple}(x_1, \dots, x_k))}}{\varphi[x_i / \text{Suf}(e, \omega)] \vdash \varphi[x_i / \varepsilon]^{[P]_{\text{PV}}(\omega, \text{MakeTuple}(x_1, \dots, x_k))}}$$

Proof. TKTK \square

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, Jiayu 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 Krajíček. 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.