5 "Advanced" Data Structures

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

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

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

5.1 Lists

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

5.1.1 Definition of Lists

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

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

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

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

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

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

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

Indeed the construction is similar to tuples. We define:

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

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

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

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

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

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

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

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

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

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

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

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

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

5.1.2 Recursion on Lists

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\mathsf{PV} \vdash \mathsf{Find}_f(\varepsilon) = \varepsilon$$
$$\mathsf{PV} \vdash \mathsf{Find}_f(\ell :: u) = \mathsf{ITE}(f(u, \vec{w}), s_0(u), \mathsf{Find}_f(\ell))$$

(Note that we output $s_0(u)$ instead of u as otherwise we cannot distinguish between the case that $\mathsf{Find}_f(\ell)$ does not find the element, or find the element $\varepsilon \in \ell$ satisfying that $f(\varepsilon, \vec{w})$.)

In particular, we can define membership query of an element in a list as $\mathsf{Find}_x(\ell) := \mathsf{IsNotEps}(\mathsf{Find}_{f_x}(\ell))$ with $f_x(u) = \mathsf{EQ}(x, u)$. For simplicity, we write $[x \in \ell]$ as the abbreviation of $\mathsf{Find}_x(\ell) = 1$. It is left as an exercise that PV-PL proves:

$$\vdash \ \neg[x \in \varepsilon]$$
$$\vdash \ [x \in \ell :: x]$$
$$[x \in \ell] \vdash [x \in \ell :: u]$$

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

5.1.3 Induction on Lists

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

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

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

The proof of the theorem is deferred to Section 5.3.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.1.4 Dictionaries

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

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

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

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

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

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

$$\Gamma \vdash \mathsf{Lookup}(\varepsilon, x) = \varepsilon; \tag{123}$$

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

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

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

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

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

$$\mathsf{Right}(\mathsf{TR}(\mathsf{ITE}(\mathsf{EQ}(\mathsf{Left}(\mathsf{MakePair}(x', y')), x), s_0(\mathsf{MakePair}(x', y')), \mathsf{Find}_f(\gamma))) = y,$$

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

$$\mathsf{Right}(\mathsf{TR}(\mathsf{ITE}(\mathsf{EQ}(x', x), s_0(\mathsf{MakePair}(x', y')), \mathsf{Find}_f(\gamma)))) = y.$$

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

$$\mathsf{Right}(\mathsf{TR}(\mathsf{ITE}(z, s_0(\mathsf{MakePair}(x', y')), \mathsf{Find}_f(\gamma)))) = y$$

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

$$\mathsf{Right}(\mathsf{TR}(\mathsf{ITE}(0, s_0(\mathsf{MakePair}(x', y')), \mathsf{Find}_f(\gamma)))) = y$$

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

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

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

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

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

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

5.2 Proof of the Recursion on Lists Meta-theorem

Recall the statement of Theorem 5.3:

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

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

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

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

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

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

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

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

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

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

We need to prove that $\mathsf{ITRL}(\ell, \ell) = \varepsilon$. Indeed, we will prove a stronger result in PV-PL:

$$\vdash \mathsf{ITR}(\mathsf{ITRL}(\ell, x), \mathsf{ITR}(\ell, x)) = \varepsilon,$$

which suffices as we can substitute x/ℓ.

The assertion above can be proved by an induction on x using the (Ind_1) rule. The base case is straightforward. In the induction case, we need prove that

$$\mathsf{ITR}(\mathsf{ITRL}(\ell, x), \mathsf{ITR}(\ell, x)) = \varepsilon \vdash \mathsf{ITR}(\mathsf{ITRL}(\ell, s_i(x)), \mathsf{ITR}(\ell, s_i(x))) = \varepsilon,$$

Note that the consequence is PV-provably equivalent to

$$\mathsf{ITR}(\mathsf{Tail}(\mathsf{ITRL}(\ell, x)), \mathsf{TR}(\mathsf{ITR}(\ell, x))) = \varepsilon.$$

Therefore, it suffices (by the substitution/generalization rule) to prove that $\mathsf{ITR}(x, y) = \varepsilon \vdash \mathsf{ITR}(\mathsf{Tail}(x), \mathsf{TR}(y)) = \varepsilon$. This can be derived from basic properties of ITR and $\vdash \mathsf{ITR}(\mathsf{Tail}(x), \mathsf{TR}(x)) = \varepsilon$, which subsequently follows from a tedious but straightforward induction on x.

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

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

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

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

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

To recursively define a function from g', h'_i, k'_i, we need to show that it is provable in PV that $\mathsf{ITR}(h'_i(\vec{x}, \ell, y, z), z \circ k'_i(\vec{x}, \ell, y)) = \varepsilon$ for $i \in \{0, 1\}$. Notice that both h'_i and k'_i are defined with a case study on the condition $\mathsf{IsEps}(\hat{\ell})$. Therefore, by case study on ITE (see Theorem 3.6 and Remark 3.2), it suffices to prove that

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

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

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

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

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

by unfolding the definitions.

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

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

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

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

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

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

Proof. This can be proved by an induction on y. The equation trivially holds when $y = \varepsilon$ by unfolding the LHS. Suppose that the equation holds for y, notice that

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

and

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

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

Proof of Equation (114). We first show that Equation (126) suffices to prove Equation (114). Suppose that Equation (126) is true, by substituting y/ε and unfolding ITR, we can obtain that

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

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

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

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

Proposition 5.10. PV *proves that*

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

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

Proof Sketch. We perform case study on y using the induction rule of PV-PL. This equation holds for all three cases y/ε, $y/s_0(y)$, and $y/s_1(y)$ by simple unfolding. □

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

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

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

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

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

Note that this will actually lead to a proof of

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

that is not known to imply Equation (126). Nevertheless, since what we actually need to do is to prove the original property Equation (114) of the function f by substituting y/ε, the equation above also suffices with the substitution y'/ℓ instead.

Recall that $\mathsf{EQL}(x, y)$ outputs 1 (resp. 0) if and only if $|x| = |y|$ (resp. $|x| \neq |y|$). It turns out:

Proposition 5.12. PV-PL *proves the following assertions:*

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

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

We perform an induction on y' to prove Equation (127), in which we will need to resolve the following three subgoals:

(*Subgoal* 1). $\vdash$ Equation (127) with substitution y'/ε.
(*Subgoal* 2). Equation (127) $\vdash$ Equation (127) with substitution $y'/s_0(y')$
(*Subgoal* 3). Equation (127) $\vdash$ Equation (127) with substitution $y'/s_1(y')$

We will deal with the three cases in the following lemmas.

Lemma 5.13 (*Subgoal* 1)**.** $\mathsf{PV} \vdash f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\ell))) = f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \ell))$.

Proof Sketch. Since $\mathsf{PV} \vdash \mathsf{ITR}(\ell, \ell) = \varepsilon$, the RHS of the equation is PV-provably equal to $f'(\vec{x}, \ell, \varepsilon)$, and by the definition axiom of f' it is further PV-provably equal to $g(\vec{x})$.

As for the LHS, notice that $\mathsf{PV} \vdash \mathsf{IsEps}(\mathsf{ITR}(\ell :: u, s_0(\ell))) = 0$ by unfolding $\ell :: u$ and proving basic properties about ITR. By Proposition 5.10, we know that

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

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

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

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

Hint: Using Proposition 5.12

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

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

Next, we prove (*Subgoal* 2) and (*Subgoal* 3). Fix any $i \in \{0,1\}$. Let $\hat{\ell}$ be a fresh variable, so that we can introduce $\hat{\ell} = \mathsf{ITRL}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))$ as an assumption (aka. in the antecedent). Note that by unfolding ITRL we can prove in PV that $\hat{\ell} = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, s_i(y')))$.

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

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

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

We will prove the following two lemmas:

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

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

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

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

These two lemma suffices to prove that $\Gamma \vdash$ Equation (127) (with the substitution $y'/s_i(y')$ by a case study on $\hat{\ell}$, which subsequently completes the proof of all subgoals.

Proof Sketch of Lemma 5.14. Indeed, we will prove that the LHS and the RHS of the equation are both identical to $g(\vec{x})$.

(LHS). Note that by Proposition 5.10, we know that

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

where

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

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

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

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

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

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

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

where

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

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

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

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

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

This completes the proof. □

Proof of Lemma 5.15. Similar to the proof of Lemma 5.14, we can apply Proposition 5.10 to show that:

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

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

$$f'(\vec{x}, \ell :: u, \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))))). \tag{128}$$

Similarly, we can also prove that

$$f'(\vec{x}, \ell, \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))) = h(\vec{x}, u, \ell, f'(\vec{x}, \ell, \mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))))). \tag{129}$$

Notice that the LHS of Equation (128) and (129) are the LHS and RHS of the equation in the lemma, respectively. Therefore, it suffices to prove that

$$f'(\vec{x}, \ell :: u, \mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y')))))) = f'(\vec{x}, \ell, \mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y')))))). \tag{130}$$

Finally, notice that PV proves:

$$\mathsf{EQL}(\mathsf{TR}(\mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, s_i(y'))))), \mathsf{ITR}(\ell :: u, s_0(\mathsf{ITR}(\ell, y')))) = 1,$$
$$\mathsf{EQL}(\mathsf{TR}(\mathsf{ITR}(\ell, \mathsf{ITR}(\ell, s_i(y'))))), \mathsf{ITR}(\ell, \mathsf{ITR}(\ell, y'))) = 1.$$

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

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

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

5.3 Proof of the Admissibility of Induction on Lists

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

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

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

Intuitively, we will prove the theorem by applying an induction on the length of the list ℓ using the (Ind_1) rule of PV-PL, where the base case and the induction case will be resolved by the premises of the rule. The idea is quite similar to the proof of Theorem 5.3; we will prove by induction on a fresh variable y that $t_1 = t_2$ after the substitution

$$y/\mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, y)).$$

That is, $t_1 = t_2$ holds for every prefix of the list ℓ, and we prove the fact by induction on the list of the prefix.

We will prove Theorem 5.5 in two steps: We first define the partial list function $\mathsf{PrfList}(\ell, x) = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, x))$ and proves its properties, and then perform the induction proof.

Step 1: Properties of Partial Lists. Recall that $\mathsf{ITRL}(\ell, x)$ iteratively remove the outermost element in the list ℓ for $|x|$ times, and $\mathsf{Len}(\ell)$ computes the length of the list ℓ. We can define a function $\mathsf{PrfList}(\ell, x) = \mathsf{ITRL}(\ell, \mathsf{ITR}(\ell, x))$.

Proposition 5.16. *The following assertions are provable in* PV-PL*:*

- $\mathsf{IsList}[\ell] \vdash \mathsf{PrfList}(\ell, \varepsilon) = \varepsilon$*;*
- $\mathsf{IsList}[\ell] \vdash \mathsf{IsList}[\mathsf{PrfList}(\ell, x)]$
- $\mathsf{IsList}[\ell] \vdash \mathsf{PrfList}(\ell, x) = \mathsf{PrfList}(\ell, s_i(x)) \vee \mathsf{PrfList}(\ell, x) = \mathsf{Tail}(\mathsf{PrfList}(\ell, s_i(x)))$.

The disjunction is defined in terms of $\to$ and $\bot$. Note that standard inference rules are admissible, see Remark 4.1.

Proof Sketch. The first bullet is equivalent to $\mathsf{ITRL}(\ell, \ell) = \varepsilon$, which has been proved in Step 1 of the proof of Theorem 5.3 (see Section 5.2).

The second bullet can be proved by an induction on y on the stronger statement $\mathsf{IsList}[\ell] \vdash \mathsf{IsList}[\mathsf{ITRL}(\ell, y)]$. The base case is trivial as $\mathsf{ITRL}(\ell, \varepsilon) = \varepsilon$, while the induction case follows from the definition equation of ITRL and IsList.

For the last bullet, we first prove a case study on whether $\mathsf{ITR}(x, \ell) = \varepsilon$. If it is not the case, we have that $\mathsf{ITR}(\ell, x) = \mathsf{ITR}(\ell, s_i(x)) = \varepsilon$ and the left side of the disjunction holds by the reflexivity of equality. Otherwise, we know that

$$\mathsf{PV} \vdash \mathsf{EQL}(\mathsf{ITR}(\ell, x), s_0(\mathsf{ITR}(\ell, s_i(x)))) = 1.$$

and it suffices to prove for a fresh variable z that

$$\mathsf{ITRL}(\ell, z) = \mathsf{ITRL}(\ell, s_0(z)) \vee \mathsf{ITRL}(\ell, s_0(z)) = \mathsf{Tail}(\mathsf{ITRL}(\ell, z))$$

by the second bullet of Proposition 5.12. The right side of the disjunction holds by the definition equation of ITRL. □

Step 2: Induction. Let y be a fresh variable. We will prove a stronger statement that

$$\Gamma, \mathsf{IsList}[\ell] \vdash t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]. \tag{131}$$

It is easy to verify that this suffices as we can make the substitution $y/\mathsf{Len}(\ell)$ so that the consequence is PV-PL provably equivalent to $t_1 = t_2$.

We prove Equation (131) by applying (Ind_1) on the variable y. It generates three subgoals:

- (*Subgoal 1*): $\Gamma, \mathsf{IsList}[\ell] \vdash t_1[\ell/\mathsf{PrfList}(\ell, \varepsilon)] = t_2[\ell/\mathsf{PrfList}(\ell, \varepsilon)]$.
- (*Subgoal 2*): From $\Gamma, \mathsf{IsList}[\ell], t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]$ deduce that $t_1[\ell/\mathsf{PrfList}(\ell, s_0(y))] = t_2[\ell/\mathsf{PrfList}(\ell, s_0(y))]$.
- (*Subgoal 3*): From $\Gamma, \mathsf{IsList}[\ell], t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]$ deduce that $t_1[\ell/\mathsf{PrfList}(\ell, s_1(y))] = t_2[\ell/\mathsf{PrfList}(\ell, s_1(y))]$.

We first prove (*Subgoal 1*). By the first bullet of Proposition 5.16, we know that $\mathsf{PrfList}(\ell, \varepsilon) = \varepsilon$, and thus it suffices to prove $t_1[\ell/\varepsilon] = t_2[\ell/\varepsilon]$, which follows from the first premise of the rule.

It remains to prove the (*Subgoal 2*) and (*Subgoal 3*). Let $i \in \{0, 1\}$, we need to prove that

$$t_1[\ell/\mathsf{PrfList}(\ell, s_i(y))] = t_2[\ell/\mathsf{PrfList}(\ell, s_i(y))] \tag{132}$$

from $\Gamma, \mathsf{IsList}[\ell], t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]$. By the last bullet of Proposition 5.16, we can perform a case study on whether

$$\mathsf{PrfList}(\ell, x) = \mathsf{PrfList}(\ell, s_i(x)) \text{ or } \mathsf{PrfList}(\ell, x) = \mathsf{Tail}(\mathsf{PrfList}(\ell, s_i(x))).$$

In the former case, we can prove Equation (132) as the equation is equivalent to $t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]$, which is available in the antecedent.

In the latter case, we first prove that $\mathsf{IsList}[\mathsf{PrfList}(\ell, y), \mathsf{PrfList}(\ell, s_i(y))]$ by the second bullet of Proposition 5.16. Then it follows that

$$\begin{aligned} &\mathsf{PrfList}(\ell, s_i(x)) \\ =\ &\mathsf{Tail}(\mathsf{PrfList}(\ell, s_i(x))) :: \mathsf{Head}(\mathsf{PrfList}(\ell, s_i(x))) \\ =\ &\mathsf{PrfList}(\ell, x) :: \mathsf{Head}(\mathsf{PrfList}(\ell, s_i(x))), \end{aligned}$$

where the first equality follows from the definition equation of $\mathsf{IsList}[\mathsf{PrfList}(\ell, s_i(x))]$, and the second equality follows from that $\mathsf{PrfList}(\ell, x) = \mathsf{Tail}(\mathsf{PrfList}(\ell, s_i(x)))$.

By the cut rule, we can add this equation into the antecedent, and thus it suffices to prove from

$$\Gamma, \mathsf{IsList}[\mathsf{PrfList}(\ell, y)], t_1[\ell/\mathsf{PrfList}(\ell, y)] = t_2[\ell/\mathsf{PrfList}(\ell, y)]$$

that

$$\begin{aligned} &t_1[\ell/\mathsf{PrfList}(\ell, x) :: \mathsf{Head}(\mathsf{PrfList}(\ell, s_i(x)))] \\ =\ &t_2[\ell/\mathsf{PrfList}(\ell, x) :: \mathsf{Head}(\mathsf{PrfList}(\ell, s_i(x)))]. \end{aligned}$$

Pick fresh variable $\hat{\ell}$ and u. By the substitution/generalization rule of PV-PL, it then suffices to prove that:

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

This can be proved from the second premise with the substitution $\ell/\hat{\ell}$, which completes the proof.