7.2 Operations on Lists

7.2.1 Introduction. In this section we are concerned with the problem of specification and verification of various useful simple operations over lists. We will show how the algorithms can be implemented by using structural recursion and how their specification properties can be proved by the corresponding induction principles.

7.2.2 Map. The operation $\mathit{Map}_f(x)$ applies an unary function f to each element of the list x:

$$\vdash_{\mathrm{PA}} L\,\mathit{Map}_f(x) = L(x) \tag{1}$$

$$\vdash_{\mathrm{PA}} i < L(x) \to \mathit{Map}_f(x)[i] = f(x[i]) \tag{2}$$

The mapping is defined by structural list recursion as a p.r. function in f by

$\mathit{Map}_f(0) = 0$
$\mathit{Map}_f\langle v, w\rangle = \big\langle f(v), \mathit{Map}_f(w)\big\rangle.$

Verification. (1): By a straightforward structural list induction.
(2): By structural induction on the list x as $\forall i(2)$. In the base case there is nothing to prove. In the induction step, when $x = \langle v, w\rangle$ for some v, w, take any i s.t. $i < L\,\langle v, w\rangle = L(w) + 1$ and consider two cases. If $i = 0$ then

$$\mathit{Map}_f\langle v, w\rangle[0] = \big\langle f(v), \mathit{Map}_f(w)\big\rangle[0] = f(v) = f(\langle v, w\rangle[0]).$$

If $i = j + 1$ for some j then $j < L(w)$ and we thus obtain

$$\begin{aligned}\mathit{Map}_f\langle v, w\rangle[j+1] = \big\langle f(v), \mathit{Map}_f(w)\big\rangle[j+1] = \mathit{Map}_f(w)[j] \overset{\mathrm{IH}}{=}\\ = f(w[j]) = f(\langle v, w\rangle[j+1]).\end{aligned}$$

Note that the induction hypothesis is applied with j in place of i. □

7.2.3 Take and drop. The function $\mathit{Take}(n, x)$ yields the initial segment of a list x of the length n provided $n \le L(x)$. The function satisfies

$$\vdash_{\mathrm{PA}} n \le L(x) \to L\,\mathit{Take}(n, x) = n \tag{1}$$

$$\vdash_{\mathrm{PA}} n \le L(x) \to \exists y\, x = \mathit{Take}(n, x) \oplus y \tag{2}$$

and it is defined by primitive recursion on n with substitution in parameter as primitive recursive by

$\mathit{Take}(0, x) = 0$
$\mathit{Take}(n+1, \langle v, w\rangle) = \big\langle v, \mathit{Take}(n, w)\big\rangle.$

Note the default $\mathit{Take}(n+1, 0) = 0$.

The function $\mathit{Drop}(n,x)$ removes the initial segment of a list x of the length n provided $n \le L(x)$. The function satisfies

$$\vdash_{\mathrm{PA}} n \le L(x) \to L\,\mathit{Drop}(n,x) = L(x) \dot{-} n \tag{3}$$
$$\vdash_{\mathrm{PA}} n \le L(x) \to \exists y\, x = y \oplus \mathit{Drop}(n,x) \tag{4}$$

and it is defined by primitive recursion on n with substitution in parameter as primitive recursive by

$$\begin{aligned}&\mathit{Drop}(0,x) = x\\ &\mathit{Drop}(n+1,\langle v,w\rangle) = \mathit{Drop}(n,w).\end{aligned}$$

Note the default $\mathit{Drop}(n+1,0) = 0$.

Usually we intend to apply both operations $\mathit{Take}(n,x)$ and $\mathit{Drop}(n,x)$ only in cases when $n \le L(x)$. We can take the following properties as alternative programs for computing the functions in such cases:

```
⊢PA n ≤ L(x) → Take(n,x) = case
                              n = 0 ⇒ 0
                              n = m + 1 ⇒ let x = ⟨v,w⟩ in ⟨v, Take(n,w)⟩
                           end.

⊢PA n ≤ L(x) → Drop(n,x) = case
                              n = 0 ⇒ x
                              n = m + 1 ⇒ let x = ⟨v,w⟩ in Drop(n,w)
                           end.
```

Note that both programs share the same condition of regularity

$$\vdash_{\mathrm{PA}} n \le L(x) \land n = m+1 \land x = \langle v,w\rangle \to w < x \land m \le L(w)$$

which is trivially satisfied.

Verification. (1): This is proved by induction on n as $\forall x(1)$. The base case is obvious. In the induction step take any x such that $n+1 \le L(x)$. Then $x = \langle v,w\rangle$ for some v,w, where $n \le L(w)$. We obtain

$$L\,\mathit{Take}(n+1,\langle v,w\rangle) = L\,\big\langle v, \mathit{Take}(n,w)\big\rangle = L\,\mathit{Take}(n,w) + 1 \overset{\mathrm{IH}}{=} n+1.$$

Note that the induction hypothesis is applied with w in place of x.

(2): By induction on n as $\forall x(2)$. In the base case it suffices to take $y := x$ since $\mathit{Take}(0,x) \oplus x = 0 \oplus x = x$. In the induction step assume $n+1 \le L(x)$. Then $x = \langle v,w\rangle$ for some v,w. Since $n \le L(w)$ we get from IH applied with w in place of x that $w = \mathit{Take}(n,w) \oplus y$ for some y. We then have

$$\langle v,w\rangle = \langle v, \mathit{Take}(n,w) \oplus y\rangle = \langle v, \mathit{Take}(n,w)\rangle \oplus y = \mathit{Take}(n+1,\langle v,w\rangle) \oplus y.$$

The remaining properties (3) and (4) are proved similarly. □

7.2.4 Interval. The binary function $[m \mathinner{..} n)$ returns the list of numbers from m to $n-1$ if $m < n$; the list is empty if $m \geq n$. The function satisfies

$$\vdash_{\mathrm{PA}} L\,[m \mathinner{..} n) = n \dot{-} m \tag{1}$$

$$\vdash_{\mathrm{PA}} i + m < n \to [m \mathinner{..} n)[i] = m + i \tag{2}$$

and it is defined by recursion with measure $n \dot{-} m$ as a p.r. function by

$$\begin{aligned}&[m \mathinner{..} n) = 0 \leftarrow m \geq n\\ &[m \mathinner{..} n) = \bigl\langle m, [m+1 \mathinner{..} n)\bigr\rangle \leftarrow m < n.\end{aligned}$$

Note that this is an example of function definition by backward recursion.

We usually intend to apply the operation $[m \mathinner{..} n)$ only in cases when $m \leq n$. For that we can take the following property as an alternative (conditional) program for computing the function:

$$\begin{aligned}\vdash_{\mathrm{PA}} m \leq n \to [m \mathinner{..} n) = &\ \textbf{case}\\ &\quad m = n \Rightarrow 0\\ &\quad m \neq n \Rightarrow \bigl\langle m, [m+1 \mathinner{..} n)\bigr\rangle\\ &\ \textbf{end}\end{aligned}$$

Its condition of regularity

$$\vdash_{\mathrm{PA}} m \leq n \wedge m \neq n \to n \dot{-} (m+1) < n \dot{-} m \wedge m + 1 \leq n$$

is trivially satisfied. Note that the program does not terminate for $m > n$.

Verification. (1): By induction with measure $n \dot{-} m$. Take any m, n and consider two cases. If $m \geq n$ then $L\,[m \mathinner{..} n) = L(0) = 0 = n \dot{-} m$. If $m < n$ then $m < m + 1 \leq n$ and we obtain

$$L\,[m \mathinner{..} n) = L\bigl\langle m, [m+1 \mathinner{..} n)\bigr\rangle = L\,[m+1 \mathinner{..} n) + 1 \overset{\mathrm{IH}}{=} n \dot{-} (m+1) + 1 = n \dot{-} m.$$

(2): This is proved by induction with measure $n \dot{-} m$ as $\forall i(2)$. Take any m, n, i such that $i + m < n$ and consider two cases. If $i = 0$ then we have

$$[m \mathinner{..} n)[0] = \bigl\langle m, [m+1 \mathinner{..} n)\bigr\rangle[0] = m = m + 0.$$

If $i = j + 1$ for some j then $j + (m+1) = j + 1 + m < n$ and thus

$$\begin{aligned}[m \mathinner{..} n)[j+1] = \bigl\langle m, [m+1 \mathinner{..} n)\bigr\rangle[j+1] = [m+1 \mathinner{..} n)[j] \overset{\mathrm{IH}}{=}\\ = m + 1 + j = m + (j+1).\end{aligned}$$

Note that the induction hypothesis is applied with j in place of i. $\square$

7.2.5 Filter. Let $A(x)$ be arbitrary but fixed unary predicate. The function $\mathit{Filter}_A(x)$ removes all elements from a list which do not satisfy the predicate. The function satisfies

$$\vdash_{\mathrm{PA}} a \mathrel{\varepsilon} \mathit{Filter}_A(x) \leftrightarrow a \mathrel{\varepsilon} x \land A(a) \tag{1}$$

and it is defined by structural list recursion as a p.r. predicate in A:

$$\begin{aligned}
&\mathit{Filter}_A(0) = 0\\
&\mathit{Filter}_A\langle v, w\rangle = \big\langle v, \mathit{Filter}_A(w)\big\rangle \leftarrow A(v)\\
&\mathit{Filter}_A\langle v, w\rangle = \mathit{Filter}_A(w) \leftarrow \neg A(v).
\end{aligned}$$

Verification. Property (1) is proved by structural induction on the list x. The base case is obvious. In the induction step, when $x = \big\langle v, w\big\rangle$ for some v, w, we consider two cases. If $A(v)$ then we have

$$a \mathrel{\varepsilon} \mathit{Filter}_A\langle v, w\rangle \Leftrightarrow a \mathrel{\varepsilon} \big\langle v, \mathit{Filter}_A(w)\big\rangle \Leftrightarrow a = v \lor a \mathrel{\varepsilon} \mathit{Filter}_A(w) \overset{\mathrm{IH}}{\Leftrightarrow}$$
$$a = v \lor a \mathrel{\varepsilon} w \land A(a) \overset{(*)}{\Leftrightarrow} (a = v \lor a \mathrel{\varepsilon} w) \land A(a) \Leftrightarrow a \mathrel{\varepsilon} \langle v, w\rangle \land A(a).$$

The equivalence marked by $(*)$ is by case analysis on whether or not $a = v$. The case when $A(v)$ does not hold is similar. □

7.2.6 Removal of duplicates from lists. The function $\mathit{Nodoubles}(x)$ removes duplicates from a list. The function satisfies

$$\vdash_{\mathrm{PA}} a \mathrel{\varepsilon} \mathit{Nodoubles}(x) \leftrightarrow a \mathrel{\varepsilon} x \tag{1}$$
$$\vdash_{\mathrm{PA}} a \mathrel{\varepsilon} \mathit{Nodoubles}(x) \rightarrow \#_a \mathit{Nodoubles}(x) = 1 \tag{2}$$

and it is defined by structural list recursion as a p.r. function:

$$\begin{aligned}
&\mathit{Nodoubles}(0) = 0\\
&\mathit{Nodoubles}\langle v, w\rangle = \mathit{Nodoubles}(w) \leftarrow v \mathrel{\varepsilon} w\\
&\mathit{Nodoubles}\langle v, w\rangle = \big\langle v, \mathit{Nodoubles}(w)\big\rangle \leftarrow v \mathrel{\not\varepsilon} w.
\end{aligned}$$

Verification. (1): By a straightforward structural list induction.

(2): By structural induction on the list x. In the base case there is nothing to prove. In the induction step, when $x = \langle v, w\rangle$ for some v, w, assume $a \mathrel{\varepsilon} \mathit{Nodoubles}\langle v, w\rangle$ and consider two cases. If $v \mathrel{\varepsilon} w$ then, by definition, $a \mathrel{\varepsilon} \mathit{Nodoubles}(w)$. We obtain

$$\#_a \mathit{Nodoubles}\langle v, w\rangle = \#_a \mathit{Nodoubles}(w) \overset{\mathrm{IH}}{=} 1.$$

If $v \mathrel{\not\varepsilon} w$ then by definition either $a = v$ or $a \mathrel{\varepsilon} \mathit{Nodoubles}(w)$, and also

$$\#_a \mathit{Nodoubles}\langle v, w\rangle = (a =_* v) + \#_a \mathit{Nodoubles}(w). \tag{$\dagger_1$}$$

Now consider two subcases. If $a = v$ then $v \mathrel{\not\varepsilon} \mathit{Nodoubles}(w)$ by (1) and thus

$$\#_v \mathit{Nodoubles}\langle v, w\rangle \overset{(\dagger_1)}{=} (v =_* v) + \#_v \mathit{Nodoubles}(w) \overset{7.2.8(3)}{=} 1 + 0 = 1.$$

If $a \neq v$ then it must be $a \mathrel{\varepsilon} \mathit{Nodoubles}(w)$ and therefore

$$\#_a Nodoubles\,\langle v,w\rangle \overset{(\dagger_1)}{=} 0 + \#_a Nodoubles(w) \overset{\text{IH}}{=} 0 + 1 = 1. \qquad \square$$

7.2.7 List minimum. The function $Minl(x)$ yields the minimal element of a non-empty list. The function satisfies

$$\vdash_{\text{PA}} x \neq 0 \rightarrow Minl(x)\ \varepsilon\ x \tag{1}$$

$$\vdash_{\text{PA}} x \neq 0 \land a\ \varepsilon\ x \rightarrow Minl(x) \leq a \tag{2}$$

and it is defined by list recursion as a p.r. function:

$Minl\,\langle v,0\rangle = v$
$Minl\,\langle v,w\rangle = \min(v, Minl(w)) \leftarrow w \neq 0.$

Note the default $Minl(0) = 0$.

We usually intend to apply the operation $Minl$(x) only in cases when input lists are non-empty. For that we can take the following property as an alternative (conditional) program for computing list minimum:

```
⊢PA  $x \neq 0 \rightarrow Minl(x) =$ **let** $x = \langle v,w\rangle$ **in**
                              **case**
                                $w = 0 \Rightarrow v$
                                $w \neq 0 \Rightarrow \min(v, Minl(w))$
                              **end**.
```

Its condition of regularity

$$\vdash_{\text{PA}} x \neq 0 \land x = \langle v,w\rangle \land w \neq 0 \rightarrow w < x \land w \neq 0$$

is trivially satisfied.

Verification. (1),(2): By a straightforward structural list induction on x. $\square$

7.2.8 Multiplicity. The binary function $\#_a\,(x)$ counts the number of occurrences of the element a in the list x. This is called the multiplicity of a in x. The function satisfies

$$\vdash_{\text{PA}} \#_a\langle b,0\rangle = (a =_* b) \tag{1}$$

$$\vdash_{\text{PA}} \#_a\,(x \oplus y) = \#_a\,(x) + \#_a\,(y) \tag{2}$$

and it is defined by structural list recursion as a p.r. function:

$\#_a\,(0) = 0$
$\#_a\langle v,w\rangle = (a =_* v) + \#_a\,(w).$

We have also

$$\vdash_{\text{PA}} a\ \varepsilon\ x \leftrightarrow \#_a\,(x) \neq 0. \tag{3}$$

$$\vdash_{\text{PA}} x = 0 \leftrightarrow \forall a\, \#_a\,(x) = 0. \tag{4}$$

Verification. (1): Directly from definition.

(2): By structural induction on the list x. The base case is obvious. The induction step follows from

$$\begin{aligned}\#_a\,(\langle v,w\rangle \oplus y) = \#_a\langle v, w\oplus y\rangle = (a =_* v) + \#_a\,(w\oplus y) \overset{\mathrm{IH}}{=}\\ = (a =_* v) + \#_a\,(w) + \#_a\,(y) = \#_a\langle v,w\rangle + \#_a\,(y)\,.\end{aligned}$$

(3): By a straightforward structural induction on the list x.

(4): By a simple case analysis on whether or not the list x is empty. □

7.2.9 Permutations. We wish to introduce into PA the binary predicate $x \sim y$ holding if the list x is a permutation of the list y. For example:

$$\langle 1,2,3,0\rangle \quad \langle 2,1,3,0\rangle \quad \langle 2,3,1,0\rangle \quad \langle 1,3,2,0\rangle \quad \langle 3,1,2,0\rangle \quad \langle 3,2,1,0\rangle$$

are all permutations of the three-element list $\langle 1,2,3,0\rangle$. The standard mathematical definition uses a second-order concept (bijections over finite sets) which is not expressible directly in first-order arithmetic. Our definition of the predicate in PA is based on the following simple observation:

two lists are permutations precisely when every number has the same multiplicity in either list.

Thus we can define the predicate explicitly by

$$x \sim y \leftrightarrow \forall a\, \#_a\,(x) = \#_a\,(y)\,.$$

Note that from 7.2.8(3) we get

$$\vdash_{\mathrm{PA}}\ x \sim y \leftrightarrow \forall a\big(a \,\varepsilon\, x \to \#_a\,(x) = \#_a\,(y)\big) \land \forall a\big(a \,\varepsilon\, y \to \#_a\,(x) = \#_a\,(y)\big).$$

Consequently, the predicate $x \sim y$ is primitive recursive.

7.2.10 Basic properties of permutations. First note the predicate $x \sim y$ constitutes an equivalence relation which is reflexive, symmetric and transitive. This is expressed in that order by

$$\vdash_{\mathrm{PA}}\ x \sim x \tag{1}$$
$$\vdash_{\mathrm{PA}}\ x \sim y \to y \sim x \tag{2}$$
$$\vdash_{\mathrm{PA}}\ x \sim y \land y \sim z \to x \sim z. \tag{3}$$

Congruence properties of permutations are expressed by

$$\vdash_{\mathrm{PA}}\ x \sim y \to \langle a, x\rangle \sim \langle a, y\rangle \tag{4}$$
$$\vdash_{\mathrm{PA}}\ x \sim y \to L(x) = L(y) \tag{5}$$

$$\vdash_{\mathrm{PA}}\ x_1 \sim y_1 \wedge x_2 \sim y_2 \rightarrow x_1 \oplus x_2 \sim y_1 \oplus y_2 \tag{6}$$
$$\vdash_{\mathrm{PA}}\ x \sim y \wedge a \mathrel{\varepsilon} x \rightarrow a \mathrel{\varepsilon} y. \tag{7}$$

There is one cancellation law, namely:

$$\vdash_{\mathrm{PA}}\ x_1 \oplus \langle a, x_2\rangle \sim y_1 \oplus \langle a, y_2\rangle \leftrightarrow x_1 \oplus x_2 \sim y_1 \oplus y_2. \tag{8}$$

Finally, we have also the following recurrent properties of permutations:

$$\vdash_{\mathrm{PA}}\ x \sim 0 \leftrightarrow x = 0 \tag{9}$$
$$\vdash_{\mathrm{PA}}\ x \sim \langle v, w\rangle \leftrightarrow \exists z_1 \exists z_2 (x = z_1 \oplus \langle v, z_2\rangle \wedge w \sim z_1 \oplus z_2). \tag{10}$$

In the sequel we will use these properties without explicitly referring to them.

Proof. Properties (1)–(3) hold trivially. Property (4) follows directly from the definition. Properties (6)–(9) follow from the properties of the multiplicity function (see Par. 7.2.8).

(10): In the direction ($\rightarrow$) assume $x \sim \langle v, w\rangle$. Then $v \mathrel{\varepsilon} x$ by (7) and thus, by 7.1.13(4), we have $x = z_1 \oplus \langle v, z_2\rangle$ for some z_1, z_2. Now it suffices to apply (8) to get $w \sim z_1 \oplus z_2$. The reverse direction ($\leftarrow$) follows from (8).

(5): This is proved as $\forall y(5)$ by structural induction on the list x. The base case is straightforward. In the induction step, when $x = \langle v, w\rangle$ for some v, w, take any y such that $\langle v, w\rangle \sim y$. By (10), there are lists z_1, z_2 such that $y = z_1 \oplus \langle v, z_2\rangle$ and $w \sim z_1 \oplus z_2$. We then obtain

$$\begin{aligned} L\langle v, w\rangle = L(w) + 1 &\overset{\mathrm{IH}}{=} L(z_1 \oplus z_2) + 1 = L(z_1) + L(z_2) + 1 = \\ &= L(z_1) + L\langle v, z_2\rangle = L(z_1 \oplus \langle v, z_2\rangle). \end{aligned}$$

Note that the induction hypothesis is applied with $z_1 \oplus z_2$ in place of y. □