Zbornik radova Prirodno-matematičkog fakulteta-Univerzitet u Novom Sadu
knjiga 12 (1982)
Review of Research Faculty of Science-University of Novi Sad,Volume 12(1982)

SOME PROOFS BY STRUCTURAL INDUCTION ON LISTS

Ivan Stojmenović and Vojislav Stojković
Prirodno-matematički fakultet. Institut za matematiku
21000 Novi Sad,ul. dr Ilije Djuričića br.4, Jugoslavija

ABSTRACT

In the paper, the sets Ω and ω are defined in the following way:

a) Ω is the set of the lists of the form:

$[e_1,e_2,\ldots,e_n,\ldots]$, $n \to \infty$

Each element of the list can be:

an atom, a finite list, an element of the set Ω and an element of the set ω .

b)ω is the set of lists which satisfy one of the following conditions:

- they are the elements of the set Ω
- some of its sublist is the element of the set Ω
- they contain an infinite number of sublist.

The functions length and len are defined in the following way:

length [ ] = 0

length (a:x) = 1 + length x

len [ ] = 0

len a = 0; atom a

len (a:x) = 1 + len a + len x .

The main results are:

length $x \to \infty$ $\Leftrightarrow$ $x \in \Omega$ and

len $x \to \infty$ $\Leftrightarrow$ $x \in \omega$.

The functions lenght(len) is used for the examination: the value of the function, whose arguments are the elements of the set $\Omega(\omega)$, is the element of the set $\Omega(\omega)$.

LISTS

Lists are ordered collection of arbitrary many elements which can be accessed by position. The elements of lists may be of any type including numbers, strings (atoms) or other lists (sublists). Mixed type of elements and duplicating are permitted too.

The finite lists is a list which contains a finite number of elements. The finite list is written in this way:

$[e_1,e_2,\ldots,e_n]$, where n is the number of elements.

The Ω-list is a lists which contains an infinite number of elements. The Ω-list is written in this way:

$$[e_1,e_2,\ldots,e_n] \quad , \quad n \to \infty \ .$$

The set of all Ω-lists is denoted by Ω ([2,3,4]). The ω-list is a list which contains an infinite number of atoms and/or an infinite number of sublists.

The set of all ω-lists is denote by ω .

The list which contains no elements(an empty lists) is written in this way:

[] .

SOME ELEMENTARY FUNCTIONS ON LISTS

Lists may be the arguments and results of functions. The basic construction operator on lists is ":". It makes a list one unit longer by prefixing an element to it. For example:

1:[2,3]

has the value [1,2,3] .

The "cons" function may be defined as follows:

cons a x = a:x

where:

a is an atom or a list,

x is a list.

The formal parameters need not be simple variables but can also the expressions built out of the variables and constants using the ":" operator.

This use of expressions reduces the need for explicit guards, and makes the definitions of functions, by the system of equations, more readable.

A way for decomposing a list is by using functions that yield the head or the tail of a given list.

The selector functions on the lists are the "hd" and "tl" functions.

The "hd" function yield the first element of the list. It may be defined as follows:

hd(a:x) = a

For example:

hd([1,2,3]) = hd(1:[2,3]) = 1.

The "tl" function yields the list which remains when the head element of the argument list is removed. It may be defined as follows:

tl(a:x) = x

For example:

tl([1,2,3]) = tl(1:[2,3]) = [2,3] .

Notice the differences between the results of the functions: "hd" yeilds an element of the same type as the first element of the list whereas "tl" yields a list.

Both functions have a restricted domain in that they may be only applied to non-empty lists. Thus:

hd [ ] and

tl [ ]

are both undefined.

Let us consider some basic functions.

The "lenght" function yields the number of elements in a list. The "length" function may be defined as follows:

length[ ] = 0 ... length.1

length(a:x)=1+length x length.2

For example:

length([1,2,3]) = 3 .

The length of a list is the number of elements (including duplicates) in the list to which the function is applied.

An (infix) cocnatenation operator can be used to create a list which contains all the elements of its first operand, followed by all the elements of the second operand. The concatenat - on operator "++" may be defined as follows:

```
[]++y = y                                  ... ++.1
(a:x)++ y = a : (x++y)                     ... ++.2
```

For example:

[1,2] ++ [3,4] = [1,2,3,4] .

The "reverse" function yields the list whose elements follow the reverse order in respect to the argument list. The "reverse" function may be defined as follows:

```
reverse [] = []                            ... reverse.1
reverse a:x = (reverse x)++ [a]            ... reverse.2
```

For example:

reverse [1,2,3,4] = [4,3,2,1] .

The "sort" function, on lists whose elements are numeric atoms, yields the list whose elements do not follow in decreasing order.

The "sort" function may be defined as follows:

```
sort [] = []                               ... sort.1
sort(a:x) = insert a (sort x)              ... sort.2
insert a []= [a]                                ... insert.1
insert a (b:x) = if a ≤ b then a:b:x
                           else b: insert a x ... insert.2
```

In |6| a wrong definition of the "sort" function is given. The algorithm used is a particular method of sorting, usually called "insert-sort".

STRUCTURAL INDUCTION

To make formal proofs about list-like objects, which can be described by abstract rules, requires the use of structural induction. This is a technique proposed by R.M. Burstall (|1|). for dealing with trees of unbounded size.

The principle of structural induction for lists can be stated as follows. To prove that all lists have the property P, we show:

(a) P([]) and

(b) P(a:x) under the assumption P(x).

To say it differently,

(a) Prove the required property for the basic elements (empty list)

(b1) assume the required property as an induction hypothesis for elements of the non-basic elements (the non empty list)

(b2) prove the required property for objects which contain previous elements.

Of course, this can be explained by an induction to the length of the list.

Let us prove by the structural induction the following lemma:

LEMMA 1. *A function* length x *is defined for every list* x .

P r o o f by induction on x

case [ ]

length [ ] = 0 by length.1

case (a:x)

length(a:x) = 1+length x by length.2 .

Since length x is ex hypothesi defined, then length(a:x) is defined too, and the lemma is proved.

Analogously, we can prove, by induction on x, that the functions cons a x, reverse x, insert a x and sort x are defined for every list x.

SOME PROPERTIES OF THE FUNCTION LENGTH

LEMMA 2. $\text{length}[a_1,\ldots,a_n] = n \quad (n \geq 1)$.

P r o o f.

$$
\begin{aligned}
\text{length}[a_1,\ldots,a_n] &= \text{length}(a_1:[a_2,\ldots,a_n]) && \text{by :}\\
&= 1+\text{length}[a_2,\ldots,a_n] && \text{by length.2}\\
&= 1+\text{length}(a_2:[a_3,\ldots,a_n]) && \text{by :}\\
&= 1+1+\text{length}[a_3,\ldots,a_n] && \text{by length.2}\\
&\ \ \ldots\\
&= n-1+\text{length}[a_n]\\
&= n-1+\text{length}(a_n:[\,]) && \text{by :}\\
&= n-1+1+\text{length}\ [\,] && \text{by length.2}\\
&= n+0 && \text{by length.1}\\
&= n.
\end{aligned}
$$

LEMMA 3. length $x \geq 0$.

Proof by induction on x

case $[\,]$

length $[\,] = 0$ by length.1

case (a:x)

length (a:x) = 1+length x by length.2

$\geq 1+0$ ex hypothesi

≥ 0

as required.

THEOREM 1. length $x \to \infty$ *iff* $x \in \Omega$.

Proof. Let $x = [a_1, a_2, \ldots, a_n]$ be an Ω-list. Then:

length x = length $[a_1, a_2, \ldots, a_n, \ldots]$

= length $(a_1 : [a_2, \ldots, a_n, \ldots]$ by

= 1+length $([a_2, \ldots, a_n, \ldots])$ by length.2

. . .

$= n+\text{length}\,[a_{n+1}, \ldots,] \to \infty$.

Let length $x \to \infty$. By lemma 2 x can not be a finite list, then $x \in \Omega$.

Notice that, according to Theorem 1, the function length characterizes all the Ω-lists, i.e. lists which contain an infinite number of elements.

LEMMA 4. length x++y = length x + length y .

Proof by induction on x.

case

length ($[\,]$++y) = length y by ++.1

= 0+length y

= length $[\,]$+length y by length.1

case (a:x)

length ((a:x)++y) = length (a:(x++y)) by ++.2

= 1+length (x++y) by length.2

=1+length x + length y ex hypothesi

= length (a:x) + length y by length.2

as required.

LEMMA 5. $\text{length}(x{+}{+}y) \geq \text{length}\ x$,
$\text{length}(x{+}{+}y) \geq \text{length}\ y$.

P r o o f. Immediately follows from Lemma 3 and Lemma 4.

STRUCTURAL INDUCTION AND Ω-LISTS

We can formulate the following induction principle: to prove that a property P holds for all Ω-lists it is enough to prove:

(1) $P(\Omega)$, and

(2) $P(a{:}x)$ under the assumption $P(x)$.

This is an extension of the idea of structural induction, originally suggested in |6|, and is calles the partial object induction.

THEOREMA 2. $x{+}{+}y = x$ *for any* Ω*-list* x *(and arbitrary* y *).*

P r o o f by a partial object induction on x:

case Ω

$\Omega{+}{+}y = \Omega$

Let us prove that the concatenation of an Ω-list x and arbitrary list y is an Ω -list. By Theorem 1, it suffices to prove that $\text{length}(\Omega{+}{+}y) \to \infty$.

Let $x = [a_1, a_2, \ldots]$ be an Ω-list, i.e. $x \in \Omega$.

$$\begin{aligned}
\text{length}(x{+}{+}y) &= \text{length}([a_1, a_2, \ldots]\ {+}{+}y) \\
&= \text{length}((a_1 : [a_2, \ldots]){+}{+}y) && \text{by:} \\
&= \text{length}(\ a_1 : ([a_2, \ldots]{+}{+}y)) && \text{by ++.2} \\
&= 1{+}\text{length}([a_2, \ldots]{+}{+}y) && \text{by length.2} \\
&= \ldots \\
&= n{+}\text{length}([a_{n+1}, \ldots]{+}{+}y) \to \infty
\end{aligned}$$

case $(a{:}x)$

$$\begin{aligned}
(a{:}x){+}{+}y &= a{:}(x{+}{+}y) && \text{by ++.2} \\
&= a{:}x && \text{ex hypothesi}
\end{aligned}$$

as required.

COROLLARY 1. *The concatenation of the two lists is a finite list iff both lists are finite lists.*

P r o o f. Follows from Lemma 5 and Theorem 2.

SOME PROPERTIES OF Ω-LISTS

By applying the function length it is possible to find out whether the value of a function, one of whose arguments is an Ω-list, is an Ω-list itself.

THEOREM 3. length(reverse x) = length x.

P r o o f by induction on x:

case [ ]

length(reverse[ ]) = length [ ] by reverse.1

case (a:x)

length(reverse(a:x)) = length(reverse x++[a]) by reverse.2
= length(reverse x)+length[a] by Lemma 4
= lengthx+length(a:[ ]) ex hypothesi;by:
= lengthx+1+length [ ] by length.2
= lengthx+1+0 by length.1
= lengthx+1
= length(a:x) by length.2

as required.

COROLLARY 2. reverse Ω = Ω.

Let x∈Ω . If reverse x is a finite list then the value of the length (reverse x) is a finite number, which is impossible by Theorem 3.

LEMMA 6. length(insert a x) = 1 + length x .

P r o o f by induction on x

case [ ]

length(insert a [ ]) = length[a] by insert.1
= length (a:[ ]) by:
= 1+length [ ] by length.1

case (b:x)

I let a ≤ b

length(insert a(b:x)) = length(a:b:x) by insert.2
= 1+length(b:x) by length.2

II Let a > b

length(insert a(b:x)) = length(b:insert a x) by insert.2
= 1+length(insert a x) by length.2
= 1+1+length x ex hypothesi
= 1+length(b:x) by length.2
as required.

THEOREM 4. length(sort x) = length x .

P r o o f by induction on x

case [ ]

length(sort[ ]) = length [ ] by sort.1

case (a:x)

length(sort(a:x)) = length(insert a(sort x)) by sort.2
= 1+length(sort x) by Lemma 6
= 1+length x ex hypothesi
= length(a:x) by length.2
as required.

COROLLARY 3. Sort Ω = Ω .

The list obtained by sorting an Ω-list is also an Ω - list.

ω- LISTS

A sublist is a non-empty tail of a non-empty

a) start list or

b) sublist.

For example:

the list [1[2[3[4]]]] has theree sublists:

[2[3[4]]]

[3[4]] and

[4]

The start list and empty list are not sublists.

The number of sublists of a list is equal to the num-

ber of [, in the notation of the list, minus one.

An ω-list is the list whose notation is infinite, i.e. the list itself or some of its sublists has an infinite number of elements, or the number of sublists is infinite.

Obvious, Ω is a subset of ω .

For example:

$u = [\,[a_1,\ldots,a_n,\ldots]\,]$,

$v = v{:}[a] = [[[\ \ldots\ a]a]a]$

are ω-lists, i.e. they are elements of the set ω . v is a circular list and satisfies the following equations:

hd v = v and tl v = [a].

Lists v is an example of ω-lists , which can be implemented. In the general case the infinite lists can not be impleemented. For lists u and v holds:

length u = 1, and

length v = 2.

This shows that the function "length" can not characterize ω-lists.

Let us define the function len x in the following original way:

len [ ] = 0 ... len.1

len a = 0; atom a ... len.2

len (a:x)=1+len a + len x ... len.3

The function atom x is a basic function. The value of the function atom x is "true" iff x is not a list, otherwise it is "false".

LEMMA 7. $\mathrm{len}[a_1,\ldots,a_n] = n$ if $a_1,\ldots,a_n$ are atoms.

P r o o f by induction on n

case n = 0

len [ ] = 0 by len.1

case n > 0

$\mathrm{len}[a_1,\ldots,a_n] = \mathrm{len}(a_1{:}[a_2,\ldots,a_n])$ by:

$= 1+\mathrm{len}\ a_1+\mathrm{len}[a_2,\ldots,a_n]$ by len.3

$= 1+0+\mathrm{len}[a_2,\ldots,a_n]$ by len.2

$= 1+0+n-1$ ex hypothesi

$= n$

as required.

LEMMA 8. *The value of the function* len x *is equal to the sum of the number of atoms of the list* x *and the number of sublists of the list* x .

P r o o f by induction on the number of sublists of list x

case n=0

Since there is no sublist, the proof immediately follows by Lemma 7.

case n > 0

Let $x = [a_1,\ldots,a_k]$

$$\begin{aligned} \text{len } x &= \text{len } [a_1,\ldots,a_k] \\ &= \text{len}(a_1:[a_2,\ldots,a_k]) \quad \text{by:} \\ &= 1+\text{len } a_1+\text{len}[a_2,\ldots,a_k] \qquad \text{by len.3} \\ &= \ldots \\ &= (1+\text{len } a_1)+(1+\text{len } a_2)+\ldots+(1+\text{len } a_k) \ . \end{aligned}$$

If a_i $(1\leq i\leq k)$ is an atom then the value of the expression $1+\text{len } a_i$ is equal to 1.

If a_i $(1\leq i\leq k)$ is a sublist then a unity in the expression $1+\text{len } a_i$ corresponds the sublist a_i $(1 \leq i \leq k)$. The number of sublists of the list a_i $(1 \leq i \leq k)$ is less than n, so len a_i is equal to the total number of atoms and sublists of a_i (ex hypothesi). Since a_i was an arbitrary sublist, Lemma 8 holds.

THEOREM 5. len $x \to \infty$ *iff* $x \in \omega$.

P r o o f. Let len $x \to \infty$. If the number of sublists of list x is infinite then $x \in \omega$ by definition of ω. If the number of sublists of a list is finite then by Lemma 8 there is a sublist which is an Ω-list, so $x \in \omega$.

Let $x \in \omega$. Then the number of sublists or the number of elements of some sublist is infinite. In both cases by Lemma 8 len $x \to \infty$.

Between functions length and len there exists a simple relation.

THEOREM 6. length $x \leq$ len x.

P r o o f by induction on x

case [ ]

length[ ]=len [ ] = 0 by length.1 and len.1

case (a:x)

length(a:x) = 1+length x by length.2

$\leq$ 1+len x ex hypothesi

$\leq$ 1+len x + len a

= len (a:x) by len.3

as required.

THEOREM 7. $x \in \Omega \Rightarrow x \in \omega$.

P r o o f directly follows by Theorem 6, 5 and 1.

SOME PROPERTIES OF ω-LISTS

Let us see for example function hd x. Let x = a:y .

LEMMA 9. len x > len(hd x)

P r o o f. len x = len (a:y)

= 1+len a + len y by len.3

> len a

= len(hd x).

LEMMA 10. hd $x \in \omega \Rightarrow x \in \omega$.

P r o o f is obvious. What can we conclude about Ω ?

Let $a_1 \notin \Omega$ and $\underline{u} = [[a_1,\ldots,a_n,\ldots]]$. Then:

hd u = hd$[[a_1,\ldots,a_n,\ldots]]$

= $[a_1,\ldots,a_n,\ldots] \in \Omega$

hd hd u = hd$[a_1,\ldots,a_n,\ldots]$

= $a_1 \notin \Omega$.

Let $\underline{w} = [[a_1,\ldots,]b_1,\ldots]$. Then:

$\underline{w} \in \Omega$

hd w = hd$[[a_1,\ldots],b_1,\ldots]$

= $[a_1,\ldots] \in \Omega$.

Consequently, all four cases, depending on whether the argument and the value of the function belong to Ω , are possible.

Let us see the function cons x y. It makes it easier to prove the following:

LEMMA 11. length(cons x y) = 1+length y ;
len(cons x y) = 1+len x + len y.

Consequently, function cons x y is an element of $\Omega(\omega)$ iff the least one of the arguments x and y is an element $\Omega(\omega)$.

LEMMA 12. len(x++y) = len x + len y.

P r o o f by induction on x

case [ ]

len ([ ]++y) = len y by ++.1
= len[ ] + len y by len.1

case (a:x)

len((a:x)++y) = len(a:(x++y)) by ++.2
= 1+len a + len(x++y) by len.3
= 1+len a + len x + len y by Lemma 12
= len(a:x) + len y by len.3
as required.

THEOREM 8. ω ++y = ω

P r o o f. Let $x \in \omega$. If $x \in \Omega$ then $x++y \in \Omega$ and $x++y \in \omega$. Otherwise $x = [a_1,\ldots,a_n]$ and len $x \to \infty$
By Lemma 12 follows:

len(x++y) = len x + len y $\to \infty$

then $x++y \in \omega$.

Remark that from $x \in \omega$, it is not possible to prove by structural induction that x++y = x.

LEMMA 13. len(reverse x) = len x

P r o o f by induction on x

case [ ]

len(reverse[ ]) = len [ ] by reverse.1

case (a:x)

len(reverse a:x) = len(reverse x++[a]) by reverse.2
= len(reverse x)+ len[a] by Lemma 12
= len x + len[a] ex hypothesi
= len(a:x)
as required.

THEOREM 9. reverse ω = ω .

P r o o f immediately follows from Lemma 13 and Theorema 5.

LEMMA 14. len(insert a x) = 1 +len a +len x

P r o o f by induction on x

case [ ]

len(insert a[ ]) = len[a] by insert.1
= len(a:[ ]) by:
= 1+len a + len[ ] by len.3

case (b:x)

I a ≤ b

len(insert a b:x) = len(a:b:x) by insert.2
= 1+len a + len(b:x) by len.3

II a > b

len(insert a b:x) = len(b:insert a x) by insert.3
= 1+len b + len(insert a x) by len.3
≡ 1+len b+1+len a + len x ex hypothesi
= 1+len a + len(b:x) by len.3
as required.

LEMMA 15. len(sort x) = len x

P r o o f by induction on x

case [ ]

len(sort[ ]) = len[ ] by sort.1

case (a:x)

len(sort a:x) = len(insert a(sort x)) by sort.2
= 1+len a + len(sort x) by Lemma 14
= 1+len a + len x ex hypothesi

= len(a:x)

as required.

THEOREM 10. sort(ω) = ω

P r o o f immediately follows from Lemma 15 and Theorem 5.

REFERENCES

|1| Burstall,R.M., *Proving Properties of Programs by Structural Induction, Computer Journal vol. 12, No 1, (February 1969), pp. 41-47.*

|2| Henderson,P., *Functional Programing, Application and Implementation, Prentice-Hall International, London, 1980.*

|3| Darlington,J.(ed), *Functional Programing and its Applications, Cambridge University Press, Cambridge, 1982.*

|4| Jones,C., *Software Development: A Rigorous Approach, Prentice-Hall International, London, 1980.*

|5| Manna,Z., *Mathematical Theory of Computation, Mc Graw-Hill Book Company, New York, 1974.*

|6| Turner,D.A., *Functional Programing and Proofs of Program Correctness, in: Néel,D.(ed), Tools and Notions for Program Construction, Cambridge University Press, Cambridge, 1982.*

Received by the editors May 30,1983.

REZIME

DOKAZIVANJE RAZNIH OSOBINA LISTI POMOĆU STRUKTURNE INDUKCIJE

U radu su definisani skupovi Ω i ω .

a) Ω je skup listi oblika

$[e_1, e_2, \ldots, e_n, \ldots]_{n \to \infty}$.

Svaki element e_n može biti:

- atom
- konačna lista

- element skupa Ω
- element skupa ω

b) ω je skup listi koje ispunjavaju jedan od sledećih uslova:

- pripadaju skupu Ω
- neka njena podlista pripada skupu Ω
- sadrže beskonačno mnogo podlista.

Definisane su funkcije length i len na sledeći način:

length [] = 0
len (a:x) = 1+ length x
len [] = 0
len a = 0; atom a
len (a:x) = 1 + len a + len x

Dokazano je da važi:

length $x \to \infty$ $\Longrightarrow x \in \Omega$ i
len $x \to \infty$ $\Longleftrightarrow x \in \omega$.

Funkcije length i len se koriste za ispitivanje da li je vrednost funkcije čiji su elementi argumenti skupa Ω odnosno ω , element skupa Ω odnosno ω .