A New Structural Induction Scheme for Proving Properties of Mutually Recursive Concepts

Peiya Liu
Siemens Research and Technology Laboratories
105 College Road East
Princeton, NJ 08540
Tel:609-7343349

Ruey-Juin Chang
Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712
CS.CHANG@UTEXAS-20

ABSTRACT

Structural induction schemes have been used for mechanically proving properties of self-recursive concepts in previous research. However, based on those schemes, it becomes very difficult to automatically generate the right induction hypotheses whenever the conjectures are involved with mutually recursive concepts. This paper will show that the difficulties come mainly from the weak induction schemes provided in the past, and a strong induction scheme is needed for the mutually defined concepts. Furthermore, a generalized induction principle is provided to smoothly integrate both schemes. Thus, in this mechanical induction, hypotheses are generated by mixing strong induction schemes with weak inductions schemes. While the weak induction schemes are suggested by self-recursive concepts, the strong induction schemes are suggested by mutually recursive concepts.

I. Introduction

There are currently two good ways of programming based on formal logic, namely: (1) programs based on recursive functions, such as LISP, and (2) programs based on non-negative recursive relations, such as PROLOG. Structural induction schemes have been provided for proving properties of self-recursive functions [Bourbaki 68] [Burstall 69] [Brotz 74] [Boyer 75] [Aubin 76] [Cartwright 76] [Boyer 79] and of self-recursive relations [Clark 77] [Brown&Liu 85] [Brown 86] [Liu 86]. Both schemes are applicable to recursively defined data objects such as *natural numbers, lists, and trees* [Hoare 75] [Boyer 79]. However, it is hard to apply these schemes to mutually recursive functions and relations. In this paper, a new structural induction scheme is introduced for proving properties of mutually recursive concepts. In addition, we show that the new scheme can be smoothly integrated with the old scheme in a generalized structural induction principle.

II. Self-Recursive and Mutually Recursive Concepts

Before formally stating the recursive concepts, some definitions are necessary. S is a *term* if it is a variable, a sequence of a function symbol of n arguments followed by n terms, or a sequence of a universal quantifier ALL or existential quantifier EX of two arguments followed by a variable and a term. The *scope* of a quantifier occurring in the term is the subterm to which the quantifier applies. For example, the scope of the quantifier ALL in the term (ALL X(FOO X Y)) is (FOO X Y). A variable is *free* in the term if at least one occurrence of it is not within the scope of a quantifier employing the variable. A term t *governs* an occurrence of term s if either there is a subterm (IF t p q) and the occurrence of s is in p, or there is a subterm (IF t' p q) and the occurrence of s in q, where t is (NOT t'). A term is *f-free* if the symbol f does not occur in the term as a function symbol. (ALL_LIST (x_1 ... x_n) p) is an abbreviation for (ALL x_1(ALL x_2(... (ALL x_n p)))), (EX_LIST(x_1 ... x_n) p) for (EX x_1(EX x_2(... (EX x_n p)))), and (ALL_EX (x_1 ... x_n) p) for a sequence of n mixed quantifiers over p, its negated form (EX_ALL (x_1 ... x_n)(NOT p)). NIL is considered to be false and T denotes true. The symbols EQUAL and IF are two primitive operators. Informally speaking, if X is NIL, then (IF X Y Z) is equal to Z, and if X is not NIL, then (IF X Y Z) is equal to Y. The logic operators AND, IMPLIES, OR, and NOT can also be represented by IF formulae.

The recursive concepts are formally defined as follows.

(DEFQ
(EQUAL(f_1 x_1 ... x_k $x_{1,k+1}$ ... x_{1,n_1}) $body_1$)
(EQUAL(f_2 x_1 ... x_k $x_{2,k+1}$ ... x_{2,n_2}) $body_2$)
...
(EQUAL(f_n x_1 ... x_k $x_{n,k+1}$ ... x_{n,n_n}) $body_n$)), where

(A) f_1 ... f_n are new function symbols of n_1 ... n_n arguments, respectively, and $1 \le k \le n_i$ for $1 \le i \le n$;

(B) x_1 ... x_k, $x_{i,k+1}$, ..., x_{i,n_i} for $1 \le i \le n$ are distinct variables;

(C) $body_i$ for $1 \le i \le n$ is a term and only mentions free variables in x_1, ..., x_k, $x_{i,k+1}$, ..., x_{i,n_i};

(D) there is a well-founded relation r and a measure function m of k arguments; and

(E) for each occurrence of a subterm of form (f_j y_1 ... y_k, $y_{j,k+1}$, $y_{j,k+2}$, ... , y_{j,n_j}), $1 \le j \le n$ in the $body_i$, $1 \le i \le n$, it is a theorem that:

```
(ALL__LIST (x1 ... xk x1,k+1 ... x1,n1)
      (ALL__LIST (z1 ... zs)
            (IMPLIES(AND t1 ... tp)
         (r (m y1 ... yk) (m x1 ... xk))))),
```

where t1 ... tp are f_t-free governing terms in the $body_i$ for $1 \le t \le n$, and z_1 ... z_s are the governing variables which are free variables, excluding variables x_1 ... x_k $x_{i,k+1}$ ... x_{i,n_i}, in the governing terms or subterm (f_j y_1 ... y_k, $y_{j,k+1}$, $y_{j,k+2}$, ... , y_{j,n_j}).

The definition principle is to describe that n axioms constitute a recursive definition of some concept. N axioms of the form: (f_1 x_1 ... x_k $x_{1,k+1}$ ... x_{1,n_1})= $body_1$, (f_2 x_1 ... x_k $x_{2,k+1}$ ... x_{2,n_2})= $body_2$, ..., (f_n x_1 ... x_k $x_{n,k+1}$ ... x_{n,n_n})= $body_n$ can be shown to be recursive if, according to the same measure m, the complexity of the arguments of every occurrence of f_1, ..., f_n in any $body_i$, assuming the hypotheses governing the occurrences in the $body_i$, is less than the complexity of x_1 ... x_n. The purpose of requirement (E) in the definition principle is to make recursive concepts terminate, and further, to avoid an inconsistency problem. Note that if n=1, then the principle of definition defines a self-recursive concept.

Example Q0:

```
(DEFQ
(EQUAL(EVAL L ENVRN)
      (IF(LISTP L)
         (APPLY.SUBR (CAR L)
                          (EVAL.LIST(CDR L) ENVRN))
         L))
(EQUAL(EVAL.LIST L ENVRN)
       (IF(LISTP L)
          (CONS(EVAL(CAR L) ENVRN)
               (EVAL.LIST(CDR L) ENVRN))
          NIL)))
```

In example Q0 above, mutually recursive concepts will be admitted by the following instantiation of our definition principle.

(A) f_1 is the function symbol EVAL; f_2 is the function symbol EVAL.LIST.

(B) x_1 is L, x_2 is ENVRN, k is 2, n is 2, n_1 is 2, and n_2 is 2.

(C) $body_1$ is the term (IF(LISTP L) (APPLY.SUBR (CAR L) (EVAL.LIST(CDR L) ENVRN)) L), and $body_2$ is the term (IF(LISTP L) (CONS(EVAL(CAR L) ENVRN) (EVAL.LIST(CDR L) ENVRN)) NIL).

(D) r is PLESSP and m is (LENGTH1 L ENVRN), where (LENGTH1 L ENVRN) is defined to be (LENGTH L). LENGTH is a primitive function for counting the elements in L.

(E) The following theorems are required in the definition principle:

- For an occurrence (EVAL.LIST(CDR L) ENVRN) in the body of function EVAL, the governing term is (LISTP L). It is a theorem that (ALL L(IMPLIES(LISTP L)(PLESSP(LENGTH(CDR L)) (LENGTH L)))).
- For an occurrence (EVAL.LIST(CDR L)ENVRN) in the body of function EVAL.LIST, the governing term is (LISTP L). It is a theorem that (ALL L(IMPLIES(LISTP L)(PLESSP(LENGTH(CDR L))(LENGTH L)))).
- For an occurrence (EVAL(CAR L)ENVRN) in the body of function (EVAL.LIST L ENVRN), the governing term is (LISTP L). It is a theorem that (ALL L(IMPLIES(LISTP L)(PLESSP(LENGTH(CAR L))(LENGTH L)))).

Suppose that (PART L C L1 L2) is true if L1 is a list of elements of L less than C, and L2 is a list of the rest of L. For example, suppose C is 6 and L is a list (2 6 3 9 10), then L1 is (2 3) and L2 is (6 9 10). The quick sort concept could be defined as follows.

Example Q1:

```
(DEFQ
(EQUAL(QSORT.R Z W1 W2)
  (IF(LISTP Z)
     (EX X(EX Y(IF(PART(CDR Z)(CAR Z)  X  Y)
            (EX V(IF(QSORT.R X W1 (CONS(CAR Z) V))
                           (QSORT.R Y  V W2)
                            NIL))
                          NIL)))
       (EQUAL W1 W2))))
```

In the predicate (QSORT.R Z W1 W2), Z is an input list and the output is the difference list of W1 and W2, which is an ordered list Z. The QSORT.R could be added to the system because (ALL Z(ALL W1(ALL W2(ALL X (ALL Y(ALL V(IMPLIES(AND(LISTP Z)(PART(CDR Z)(CAR Z) X Y))(PLESSP(LENGTH1 X W1 (CONS(CAR Z)) V)(LENGTH1 Z W1 W2)))))))))) and (ALL Z(ALL W1(ALL W2(ALL X (ALL Y(ALL V(IMPLIES(AND(LISTP Z)(PART(CDR Z)(CAR Z) X Y))(PLESSP(LENGTH1 Y V W2) V)(LENGTH1 Z W1 W2)))))))))) hold.

III. A Generalized Structural Induction Principle

A. Why Strong Induction Schemes are Needed

Essentially mechanical induction reasoning works because the similarity could be contrived between the structures of the recursive definition functions and of the induction schemes. The structures of recursive functions serve as templates for automatically generating the suitable induction hypotheses to prove a conjecture involved with those recursive functions. However, there is often no structure similarity between mutually recursive functions and weak induction schemes provided in previous research. The finite number of hypotheses are needed to be specified *explicitly* in the weak induction schemes. Using these weak induction schemes often results in the generation of useless induction hypotheses for the conjecture involved with mutually recursive functions. A strong induction form will be shown to be needed and can be generated from the structures of mutually recursive concepts. In the strong induction schemes, the finite number of hypotheses are *implicitly* described by particular recursive concepts.

An example will illustrate the problem in using weak induction schemes for hypothesis generation. Suppose we try to prove the conjecture (ALL L(EQUAL L(FOO L))), where the mutually recursive functions are defined as follows.

```
(DEFQ
(EQUAL(FOO L)
      (IF(LISTP L)
         (CONS(CAR L)(FOOLIST(CDR L)))
         L))
(EQUAL(FOOLIST L)
      (IF(LISTP L)
         (CONS(FOO(CAR L))(FOOLIST(CDR L)))
         L)))
```

Let (p L) be the term (EQUAL L(FOO L)). In the weak induction schemes, the instantiated terms of (p L) as induction hypotheses are required to be explicitly described. Thus, the induction hypothesis, based on the weak induction scheme and the structure of (FOO L), could be (AND(LISTP L)(p (CDR L))), (AND(LISTP L)(p (CAR L))), or (AND(LISTP L)(AND(p (CAR L))(p(CDR L)))), where (p (CAR L)) is the term (EQUAL (CAR L)(FOO (CAR L))), and (p (CDR L)) is the term (EQUAL (CDR L)(FOO (CDR L))). However, if we open the term (p L) in the proof by induction, it will not look like its counterpart in the hypotheses, and the hypotheses will be useless.

Even if we change these functions into a self-recursive function with an extra argument S as follows, our problem still exists.

```
(DEFQ
(EQUAL(FOOS L S)
    (IF(EQUAL S 0)
        (IF(LISTP L)
           (CONS (CAR L)(FOOS(CDR L) 1))
            L)
         (IF(EQUAL S 1)
            (IF(LISTP L)
               (CONS(FOOS(CAR L) 0)(FOOS(CDR L) 1))
                L)
             NIL))))
```

An induction scheme, following the weak induction principle, for the conjecture (ALL L(EQUAL L(FOOS L 0))) will be generated as (ALL L(IMPLIES(AND(LISTP L)(AND(p (CDR L))(p (CAR L)))) (p L))), where (p L) is the term (EQUAL L(FOOS L 0)). In the base case, we need to prove (ALL L(IMPLIES(NOT(LISTP L))(p L))). However, if we open the term (FOOS L 0) in the conclusion (EQUAL L(FOOS L 0)), it will also not look like its counterpart in the hypothesis. Often this type of redefined self-recursive functions is hard to suggest right induction hypotheses in weak induction schemes, due to its unnatural recursion characteristics and certain sensitive switch arguments irrelevant to the measured arguments on which functions recurse. In this example, what is really needed in the hypothesis is the term (AND(LISTP L) (FOOLIST-IND (CDR L))), where (FOOLIST-IND L) and (FOO-IND L) are mutually defined as

```
(EQUAL(FOO-IND L)
      (IF(LISTP L)
         (FOOLIST-IND (CDR L))
          T)), and

(EQUAL(FOOLIST-IND L)
      (IF(LISTP L)
         (AND (p (CAR L)) (FOOLIST-IND (CDR L)))
         T)).
```

Intuitively, the term (AND (LISTP L)(FOOLIST-IND (CDR L))) is actually ANDing the terms (LISTP L), (p (CAR (CDR L))), ..., (p (CADDDD...R L)) together by recursively opening up the term (FOOLIST-IND (CDR L)). Thus, this hypothesis implicitly represents a series of instantiated conjectures and this induction form is actually a strong induction scheme. More importantly, there is an obvious structural similarity between (FOO L) & (FOOLIST L) and (FOO-IND L) & (FOOLIST-IND L). Later on, we will give detailed descriptions of automatically constructing the terms (FOOLIST-IND L) and (FOO-IND L) from mutually recursive concepts. (FOOLIST-IND (CDR L)) is obtained from the body of (FOO-IND L) since (FOO L) appears in the conjecture, and the corresponding term (FOO-IND L) suggests the possible induction hypotheses from the recursive structure of its body.

B. A Comparison Between Weak and Strong Induction Schemes

In the induction step of the weak induction scheme, we show that if X has the desired property *at* an arbitrarily given point, then it also has the property at the next higher point. Suppose X is a pair, then it can be constructed by applying CONS to two previously constructed objects, namely, (CAR X) and (CDR X). Thus, in the weak induction scheme, we prove that a certain property (P X) holds for all X by considering two cases. In the first case, called the base case, we prove that (P X) holds for all nonpair objects X. In the second case, called the induction step, we assume that X is a pair and that (P (CAR X)) and (P (CDR X)) hold, and prove that (P X) holds.

On the other hand, in the strong induction scheme, we prove that a certain property (P X) holds for all X by considering two cases. In the first case, called the base case, we show that (P X) holds for all nonpair objects X. However, in the induction step, we assume that X is a pair and that (P (CAR X)), (P (CADR X)), ..., and (P(CADDD ... R X)) hold, and prove that (P X) holds. In other words, the induction step shows that if X has the desired property *up to* an arbitrarily given point, then it also has the property at the next higher point. For the *convenience of mechanical induction*, this series of hypotheses is represented as a recursive concept (Q X) defined to be (IF(LISTP X)(AND(P(CAR X))(Q(CDR X))) T). In the FOO example, we represent the hypothesis as (AND(LISTP X) (P*2 (CDR X))), where (P*2 X) is defined to be (IF(LISTP X)(AND(p (CAR X))(P*2(CDR X))) T). In the next section, we will show that the hypothesis can automatically be generated from mutually recursive functions by examining their structures.

C. Hypothesis Terms

Intuitively, hypothesis terms are those terms allowable to be instantiated as hypotheses in the strong induction schemes. These terms are quite powerful. They can implicitly represent a series of induction hypotheses in mechanical induction proof about the properties of mutually recursive concepts. A formal definition of hypothesis terms is described as follows. A subterm is a *call of f in the term s* if the subterm beginning with the function symbol f occurs in the term s. (P_1 x_1 ... x_n x_{n+1} ... x_t), ..., (P_d x_1 ... x_n x_{n+1} ... x_t) are the *hypothesis terms* of f_1, ..., f_d *with* P_0 *replacing* f_1, ..., f_j $1 \le j \le d$, if

1. f_1, ..., f_d are the following mutually recursive functions based on a well-founded relation R and a measure function M of n arguments,

   (EQUAL(f_1 x_1 ... x_n x_{n+1} ... x_t) $body_1$),
   (EQUAL(f_2 x_1 ... x_n x_{n+1} ... x_t) $body_2$),
   ...,
   (EQUAL(f_d x_1 ... x_n x_{n+1} ... x_t) $body_d$);

2. (P_0 x_1 ... x_n, x_{n+1} ... x_t) is a term; and

3. (P_1 x_1 ... x_n x_{n+1} ... x_t), ..., (P_d x_1 ... x_n x_{n+1} ... x_t) are obtained in the following way.

   (EQUAL(P_1 x_1 ... x_n x_{n+1} ... x_t) $body'_1$),
   (EQUAL(P_2 x_1 ... x_n x_{n+1} ... x_t) $body'_2$),
   ...,
   (EQUAL(P_d x_1 ... x_n x_{n+1} ... x_t) $body'_d$),

   where $body'_i$=(HT $body_i$) for $1 \le i \le d$ and HT is recursively defined as follows:

   a. Suppose the term s has the form (ALL_EX(z) v), then (HT s)=(ALL_EX(z)(HT v)).

   b. Suppose the term s has the form (IF c u v). Then (HT s)=(IF c (HT u)(HT v)) if the term c is f_i-free, $1 \le i \le d$; (HT s)=(IF (HT c)(HT u)(HT v)) otherwise.

   c. Suppose the term s is f_i-free, $1 \le i \le d$, then (HT s)=T.

   d. Suppose s' is a term obtained by replacing every occurrence of f_k (for $1 \le k \le j$) as a function symbol in the term s with the symbol P_0, and by replacing every occurrence of f_k (for $j < k \le d$) as a function symbol in the term s with the symbol P_k. Then (HT s)=(AND *all calls of P_i for $0 \le i \le d$ in the term s'*), if there is more than one call of P_i, or (HT s)= *a call of P_i for $0 \le i \le d$ in the term s'*, if only one call exists.

Example Q2: To find out the hypothesis terms of FOO and FOOLIST with P_0 replacing FOO.

```
(EQUAL(P1 L) (IF(LISTP L)
                 (P2(CDR L))
                 T))
(EQUAL(P2 L) (IF(LISTP L)
                 (AND(P0(CAR L))(P2(CDR L)))
                 T))
```

From the bodies of FOO and FOOLIST, the hypothesis terms (P_1 L) and (P_2 L) are constructed as above. Symbol T comes from step 3(c). The term (AND(P_0(CAR L))(P_2(CDR L))) in the body of (P_2 L) is obtained by step 3(d) from the term (CONS(FOO(CAR L))(FOOLIST(CDR L))) in the body of (FOOLIST L). In step 3(d), s' is (CONS(P_0(CAR L))(P_2(CDR L))), and (HT s)=(AND *all calls of* P_i *for* $0 \le i \le d$ *in the term s'*)=(AND(P_0(CAR L))(P_2(CDR L))).

A formal description of the generalized induction principle is contained in Appendix I. The key point in the generalized induction principle is to allow hypothesis terms, in addition to (P_0 x_1 ... x_t), to be instantiated in the induction hypothesis. This extension will make the strong induction forms possible in the hypotheses. The soundness proof of this principle was shown in [Liu 86]. The principle extends the weak induction schemes [Boyer 79] [Brown&Liu 85] to include the strong one. While strong induction schemes are shown to have a close relationship to mutually recursive concepts, weak induction schemes are related to the self-recursive concepts. In the next section, we focus on strong induction schemes and interactions between strong and weak schemes. For the pure weak induction schemes, we refer the readers to prior work [Boyer 79] [Brown&Liu 85] [Brown 86] [Liu 86].

D. Illustrations of Mixing Induction Hypotheses

Once each induction scheme is suggested by any term in the conjecture, we begin to heuristically combine the individual schemes to synthesize the best one for the conjecture. Smooth interactions between induction schemes suggested by self-recursive and mutually recursive concepts are shown below in the synthesis of the final induction scheme.

Suppose that we try to prove the conjecture (ALL L(EQUAL (FOOLIST L)(FOO L))). Note that it contains two mutually recursive concepts. Let (P_0 L) be (EQUAL(FOOLIST L)(FOO L)). (P_1 L) and (P_2 L) are the hypothesis terms of (FOO L), (FOOLIST L) with P_0 replacing FOO, FOOLIST.

```
(EQUAL(P1 L)(IF(LISTP L)
               (P0(CDR L))
               T))

(EQUAL(P2 L)(IF(LISTP L)
               (AND (P0 (CAR L))(P0 (CDR L)))
               T))
```

Therefore, the induction scheme suggested by (FOO L) is: (ALL L(IMPLIES(AND(LISTP L)(P_0 (CDR L)))(P_0 L))), and the scheme suggested by (FOOLIST L) is: (ALL L(IMPLIES(AND(LISTP L)(AND(P_0(CAR L)) (P_0(CDR L))))(P_0L))). An interesting thing is shown in this case. Two mutually recursive concepts are supposed to suggest the strong induction schemes. However, since both concepts appear in the conjecture, the strong schemes are collapsed into the weak induction schemes. By merging these two induction hypotheses, we provide one induction step and one base case to cover all the relevant recursive aspects as follows.

```
Base case:      (ALL L(IMPLIES(NOT(LISTP L))(P0L)))
Induction step: (ALL L(IMPLIES(AND(LISTP L)
                                  (AND(P0 (CAR L))
                                      (P0 (CDR L))))
                                  (P0 L)))
```

In the second example, there are self-recursive and mutually recursive concepts in the conjecture (ALL L(EQUAL(FOO L)(COPY L))), where (COPY L) is defined as (IF(LISTP L) (CONS (COPY(CAR L)) (COPY(CDR L))) L). Let (P_0 L) be (EQUAL(FOO L)(COPY L)). Thus, the weak induction scheme suggested by the function (COPY L) is: (ALL L(IMPLIES(AND(LISTP L)(AND (P_0 (CAR L))(P_0 (CDR L)))) (P_0 L))), and the strong induction scheme for the function (FOO L) is: (ALL L(IMPLIES(AND(LISTP L)(P_2 (CDR L))) (P_0 L))), where

```
(EQUAL(P1 L) (IF(LISTP L)
                (P2(CDR L))
                T))
(EQUAL(P2 L) (IF(LISTP L)
                (AND(P0(CAR L))(P2(CDR L)))
                T))
```

The final induction scheme is obtained by mixing one strong induction scheme and one weak induction scheme as follows.

```
Base case: (ALL L(IMPLIES(NOT(LISTP L))(P0 L)))
Induction step: (ALL L(IMPLIES(AND(LISTP L)
                                  (AND(P2(CDR L))
                                  (AND(P0(CAR L))
                                      (P0(CDR L)))))
                                  (P0L)))
```

Once the above induction hypotheses are set up, the rest of the proofs will become straightforward. In the research [Boyer 79] [Liu 86], many heuristics are provided to manipulate the induction schemes and formulate the best one.

IV. Conclusions

A generalized induction principle is provided for the conjectures involved with both self-recursive and mutually recursive concepts. Mechanical induction under the principle could be used as a proof strategy for a theorem prover or logic program interpreter. Two results are shown in this paper for proving properties of recursive concepts: (1) mutually recursive concepts need to suggest strong induction hypotheses, and (2) the relationship between the strong induction scheme and the weak induction scheme in mechanical structural induction.

Appendix I: A Formal Description of the Induction Principle

Suppose:

(A) p_0 is the term (p*0 x_1 ... x_n x_{n+1} ... x_t) with t distinct free variables, $1 \le n \le t$;

(B) r is a well-founded relation;

(C) m is a measure function of n arguments;

(D) (p*1 x_1 ... x_n x_{n+1} ... x_t), ..., (p*d x_1 ... x_n x_{n+1} ... x_t) are hypothesis terms of any given mutually recursive functions based on r and m with p*0 replacing a subset of {p*1, ..., p*d};

(E) b_1, ..., b_k are non-negative integers;

(F) for each i $1 \le i \le k$, variables $z_{i,1}$, ..., z_{i,b_i} are distinct and different from x_1, ..., x_n, x_{n+1}, ..., x_t;

(G) q_1, ..., q_k are terms;

(H) h_1, ..., h_k are positive integers; and

(I) for $1 \le i \le k$ and $1 \le j \le h_i$, $s_{i,j}$ is a substitution and it is a theorem that

(ALL__LIST (x_1 ... x_t)(ALL__LIST($z_{i,1}$ ... z_{i,b_i})
(IMPLIES q_i
(r(m x_1 ... x_n)/$s_{i,j}$(m x_1 ... x_n))))).

Then
(A). (ALL__LIST (x_1 ... x_t) p_0) is a theorem if

for the base case,
(ALL__LIST(x_1 ... x_t)
  (IMPLIES(AND(NOT(ALL__EX$_1$($z_{1,1}$ ... z_{1,b_1})q_1))
      ...
      (NOT(ALL__EX$_k$($z_{k,1}$ ... z_{k,b_k})q_k)))[1]
      p_0))
is a theorem and

for each $1 \leq i \leq k$ induction step,
(ALL__LIST(x_1 ... x_t)
  (IMPLIES (ALL__EX$_i$($z_{i,1}$ ... z_{i,b_i})
      (AND q_i $p^{i,1}/s_{i,1}$ ... $p^{i,h_i}/s_{i,h_i}$))[2]
      p_0))
is a theorem.

(E). (EX__LIST (x_1 ... x_t) p_0) is a theorem if

for the base case,
(EX__LIST(x_1 ... x_t)
  (AND (AND(NOT(ALL__EX$_1$($z_{1,1}$ ... z_{1,b_1})q_1))
      ...
      (NOT(ALL__EX$_k$($z_{k,1}$ ... z_{k,b_k})q_k)))
      p_0))
is a theorem or

for some $1 \leq i \leq k$ induction step,
(EX__LIST(x_1 ... x_t)
  (AND (ALL__EX$_i$($z_{i,1}$ ... z_{i,b_i})
      (AND q_i (NOT $p^{i,1}$)/$s_{i,1}$ ... (NOT p^{i,h_i})/s_{i,h_i}))
      p_0))
is a theorem.

We now illustrate an application of this induction principle to prove the conjecture (ALL L(EQUAL L(FOO L))). The induction is obtained by the following instantiation of this principle. p_0 is the term (p*0 L) defined as (EQUAL L(FOO L)); (p*1 L) and (p*2 L) are the hypothesis terms of FOO and FOOLIST with p*0 replacing FOO; r is a well-founded relation PLESSP; m is LENGTH; n is 1; t is 1; k is 1; b_1 is 0; x_1 is L; q_1 is the term (LISTP L); h_1 is 1; $s_{1,1}$ is {<L, (CDR L)>}; and one theorem required by (I) is: (ALL L(IMPLIES (LISTP L)(PLESSP(LENGTH (CDR L))(LENGTH L)))). Thus, the base case and the induction step produced by this induction principle are (ALL L (IMPLIES (NOT (LISTP L)) (p*0 L))) and (ALL L (IMPLIES (AND (LISTP L) (p*2(CDR L))) (p*0 L))). The soundness of (A) and (E) in this induction principle has been proved [Liu 86]. The proof needs two important properties that hypothesis terms preserve: (1) They satisfy the function definition principle based on the same R and M, since governing conditions remain unchanged after translation, and (2) Let <X1 ... Xt> be a t-tuple in the domain of D^t. If (P_0 Y1 ... Yt) is not false for every t-tuple <Y1 ... Yt> in the domain of D^t that is RM-smaller than <X1 ... Xt>, then (P_i Y1 ... Yt) $1 \leq i \leq d$ should not be false for such t-tuples. *RM* is the well-founded relation defined on n-tuples by (RM <Z1 ... Zn><Y1 ... Yn>)=(R (M Z1 ... Zn)(M Y1 ... Yn)).

[1] ALL_EX$_1$, ..., ALL_EX$_k$ could be any sequence of mixed quantifiers.

[2] $p^{i,1}$, ..., p^{i,h_i} are chosen from any member of {(p*0 x_1 ... x_n x_{n+1} ... x_t), (p*1 x_1 ... x_n x_{n+1} ... x_t), ..., (p*d x_1 ... x_n x_{n+1} ... x_t)}.

References

[Aubin 76] Aubin, R.
Mechanizing Structural Induction, Ph.D. Thesis.
The University of Edingburgh , 1976.

[Bourbaki 68] Bourbaki, N.
Elements of Mathematics Theory of Sets.
Addison-Wesley, Reading, 1968.

[Boyer 75] Boyer, R.S., and J S. Moore.
Proving Theorems about LISP Functions.
Journal of ACM 22(1), 1975.

[Boyer 79] Boyer, R.S., and J S. Moore.
A Computational Logic.
New York, Academic Press, 1979.

[Brotz 74] Brotz, D.
Proving Theorems by Structural Induction, Ph.D. Thesis.
Stanford University , 1974.

[Brown 86] Brown, F. M.
An Experimental Logic Based on the Fundamental Deduction Principle.
AI Journal 30(2), 1986.

[Brown&Liu 85] Brown, F. and P. Liu.
A Logic Programming and Verification System for Recursive Quantificational Logic.
Proceedings of IJCAI-85, Los Angeles , 1985.

[Burstall 69] Burstall, R.
Proving Properties of Programs by Structural Induction.
Computer Journal 12(1), 1969.

[Cartwright 76] Cartwright, R.
A Practical Formal Semantic Definition and Verification System for Typed LISP, Ph.D. Thesis.
Stanford University , 1976.

[Clark 77] Clark, K. L. and S-A Tarnlaund.
A First Order Theory of Data and Programs.
IFIP 77, North Holland , 1977.

[Hoare 75] Hoare, C.A.R.
Recursive Data Structures.
International Journal of Computer and Information Sciences 4(2), 1975.

[Liu 86] Liu, P.
A Logic-based Programming System, Ph.D. Thesis.
Department of Computer Sciences, The University of Texas at Austin , 1986.