A Unifying View of Structural Induction and Computation Induction for Logic Programs

Laurent Fribourg
L.I.E.N.S.
URA 1327 du C.N.R.S.
45 rue d'Ulm, 75005 Paris France
fribourg@dmi.ens.fr

Hans Olsén
Dept. of Information Science
Linköping University
S-581 83 Linköping Sweden
hanol@ida.liu.se

Abstract

In the framework of Extended Prolog Execution [K&S 86] a rule of Structural Induction is presented. For the induction step the new rule takes advantage of a Prolog program synthesized through Proof-Extraction techniques, and rests on extensive use of Negation as Failure Inference for exploiting the information contained in the extracted program. We describe how to synthesize Prolog procedures when applying the inference rule and show that they are guaranteed to preserve partial correctness and termination. A comparison is made between the new rule and the rules of Restricted Structural Induction [Fri 90] and Computation Induction [Cla 79], and we note that by an uncomplicated preprocessing of the goal formula and the program the new rule subsumes the two latter rules.

1 Introduction

Consider the predicate $p(X)$ defined by the Prolog program P, where X is a vector of variables. Suppose we want to prove that some relation $Q(X)$ holds between the values of the computation. This is expressed by $\forall X Q(X) \Leftarrow p(X)$, and is usually proved by applying some form of induction.

One of the problems with inductive reasoning is that the size of the formulas to be proved grows rapidly with the number of applications of induction. In this paper a rule of Structural Induction is presented, which splits the induction step into two formulas. One of these two formulas corresponds to the only-if part of the Clark completion of the program [Cla 78]. The inference rule takes advantage of a Prolog program synthesized through Proof-Extraction techniques, and rests on extensive use of Negation as Failure Inference (NFI), as defined in [K&S 86], for exploiting the information contained in the extracted program. For this reason the rule is called *NFI-driven Induction.*

Since we are interested in proving formulas of the general form $\forall X \exists Y G^{+}(X,Y) \Leftarrow G^{-}(X)$, and the rule of NFI-driven Induction presupposes that programs can be extracted from such proofs, we need a framework in wich we can manipulate such formulas and synthesize the associated programs. Extended Prolog Execution [K&S 86] provides the proof system needed, and programs can be synthesized during such proofs using a method of Proof-Extraction proposed by Fribourg [Fri 90].

The plan is as follows. In section 2 we review some formal notions of Extended Prolog Execution and of classical induction rules. In section 3 we introduce the rule of NFI-driven Induction. We compare this rule with Structural Induction (section 4) and with Computation Induction (section 5). We then indicate a preprocessing of goal formulas and programs in order to enhance NFI-driven Induction (section 6) and conclude with final remarks.

2 Background

In this section a subset of the Extended Execution of Prolog [K&S 86] is informally reviewed and we briefly explain the main concepts of Proof-Extraction. Because of limited space we will not explain how to synthesize programs during proofs using the rules mentioned in this section. The reader should consult [Fri 90] for details. Familiarity with notions such as atoms, predicates, substitutions, most general unifiers (mgu) and definite clause programs is assumed.

2.1 Extended Execution of Prolog

We consider sentences G of the form $\forall X \exists Y G^{+}(X,Y) \Leftarrow G^{-}(X)$ where $G^{+}(X,Y)$ and $G^{-}(X)$ are conjunctions of atoms, and X and Y are vectors of variables. No variable of Y is allowed to occur in $G^{-}(X)$. The formula $G' : G^{+}(X,?Y) \Leftarrow G^{-}(X)$ obtained from G by replacing the existentially quantified variables Y by $?Y$ and dropping the quantifiers, is called an *implicative goal* where $G^{-}(X)$ is the *hypothesis* and $G^{+}(X,?Y)$ is the *conclusion*. The variables of X are called *free* and those of $?Y$ are called *undecided*. We use lower case greek letters to denote substitutions. $G'\sigma$ denotes the formula that results from applying the substitution σ to the formula G'. A substitution instantiating only undecided variables is called a *deciding substitution*. A goal, the conclusion of which is empty, is said to be a *trivial goal*. A goal which is a conjunction of the form $G^{+}(X,?Y)$ is an implicative goal with an empty hypothesis.

The predicates appearing in a goal are defined by a definite clause program and we prove the goal in a backward fashion by applying the rules of Extended Execution. Given a program P defining the predicates occurring in a goal G, an inference rule *generates* or *yields* a set S of goals $H1, \ldots, Hn$ such that the conjunction of the goals of S implies G in the least Herbrand model of P.

With every goal $G : G^{+}(X,?Y) \Leftarrow G^{-}(X)$ we associate a Prolog atom, $io_G(X,Y)$, called an *io-atom*, often abbreviated as $IO_G(Y)$ or simply as IO_G. The io-atom is intended to have the same meaning, or define the same relation between the variables as the goal G, and is defined by a definite clause program, the *io-program*, synthesized during the proof of G. With each rule of inference there is associated a rule of construction for a set of definite clauses, an *io-procedure*, to add to the synthesized program when the rule is applied. For example, if the inference rule applied to G yields the goals $H1, \ldots, Hn$, the associated io-procedure will define IO_G in terms of $IO_{H1}, \ldots, IO_{Hn}$. The intention that the io-atom should have the same meaning as the goal is formalized by the notions of *partial correctness* and *termination*. Given a goal $G : G^+(X, ?Y) \Leftarrow G^-(X)$ the io-program Q defining the associated io-atom $io_G(X, Y)$ is said to be partially correct w.r.t. G iff $\forall XY\, G^+(X, Y) \Leftarrow G^-(X) \wedge io_G(X, Y)$ and terminating w.r.t. G iff $\forall X \exists Y\, io_G(X, Y) \Leftarrow G^-(X)$. In case G is a trivial goal we take the io-procedure defining IO_G to be the clause $IO_G(X) \Leftarrow G^-(X)$ which obviously is terminating and partially correct. The io-procedures associated with applications of inference rules preserve partial correctness and termination, therefore programs synthesized using this scheme are guaranteed to possess these properties.

In this paper we make use of three of the inference rules of Extended Execution applied to the set of implicative goals: DCI, Simplification and NFI. The rules of Restricted Structural Induction and Computation Induction are reviewed in subsections 2.2 and 2.3 respectively.

DCI is an extension of SLD-resolution to implicative goals. By this rule, an atom of the conclusion of a goal that unifies with the head of a program clause via a deciding mgu, may be replaced by the body of the program clause. All new variables introduced are treated as undecided.

Example 2.1 Let the predicate $*(x, y, z)$ (whose intended meaning is that z equals x times y) be defined by the program

*(0,y,0).
*(succ(x),y,z) ⇐ *(x,y,w) ∧ +(y,w,z).

Consider the goal

G: *(succ(x),y,?z) ⇐ *(x,y,v).

DCI generates the new goal

H: *(x,y,?w) ∧ +(y,?w,?z) ⇐ *(x,y,v).

By the rule of Simplification an atom of the conclusion of a goal that unifies with an atom of the hypothesis via a deciding mgu, may be deleted. Optionally the atom of the hypothesis may be deleted as well.

Example 2.2 Consider the goal

G: *(x,y,?w) ∧ +(y,?w,?z) ⇐ *(x,y,v).

Simplification via the substitution $\{v/w\}$ yields

H: +(y,v,?z).

Given the atom A occurring in the hypothesis of a goal G, for every program clause C_i whose head unifies with A the rule of NFI will generate a new goal H_i where the atom A is replaced by the body of the clause C_i. All new variables introduced are treated as free.

Example 2.3 Consider the goal

$$\text{G:}\quad +(\text{x},?\text{v},\text{z}) \Leftarrow *(\text{succ}(\text{x}),\text{y},\text{z}).$$

NFI yields the new goal

$$\text{H:}\quad +(\text{y},?\text{v},\text{z}) \Leftarrow *(\text{x},\text{y},\text{w}) \wedge +(\text{y},\text{w},\text{z}).$$

2.2 Restricted Structural Induction

The general rule of Structural Induction was introduced by Burstall (in the framework of functional programming) [Bur 69]. When applied to implicative goals, the rule introduces a second level of implication connectives and thus goes beyond our class of implicative goals. In order to stay within the framework of implicative goals, we consider a restricted form of the rule. Rules of Structural Induction presumes that the variables are typed, and there has to be one instance of the rule for every datatype. We present the rule of Restricted Structural Induction for the case the induction variable is of type natural number.

Definition 2.1 Consider the goal $G : G^{+}(x, ?Y) \Leftarrow G^{-}$ where x is a free variable of type natural number, not occurring in G^{-}, and Y is a vector of undecided variables. *Restricted Structural Induction* w.r.t. x yields the new goals

$$\text{H1:}\quad \text{G}^{+}(0,?\text{Y}) \Leftarrow \text{G}^{-}.$$
$$\text{H2:}\quad \text{G}^{+}(\text{succ}(\text{w}),?\text{Y}) \Leftarrow \text{G}^{+}(\text{w},\text{Y}') \wedge \text{G}^{-}.$$

where w is a new free variable of type natural number and Y' is a vector of free variables that is a renaming of Y.
The free variables other than x are all frozen and treated as constants. (This why the rule is said to be "restricted".)

Example 2.4 Let the predicate $+(x, y, z)$ (whose intended meaning is that z equals x plus y) be defined by the program

$$+(0,\text{y},\text{y}).$$
$$+(\text{succ}(\text{x}),\text{y},\text{succ}(\text{z})) \Leftarrow +(\text{x},\text{y},\text{z}).$$

Consider the goal expressing the termination of +

$$\text{G:}\quad +(\text{x},\text{y},?\text{z}).$$

Restricted Structural Induction w.r.t. x yields the new goals

$$\text{H1:}\quad +(0,\text{y},?\text{z}).$$
$$\text{H2:}\quad +(\text{succ}(\text{x}),\text{y},?\text{z}) \Leftarrow +(\text{x},\text{y},\text{z}_1).$$

2.3 Computation Induction

The rule of Computation Induction has been defined by Clark for logic programs [Cla 79]. Roughly speaking, when applied to a goal of the form $G^+ \Leftarrow H$ where H is an atom defined by a program P, it generates the induction scheme obtained by replacing all occurrences of H in P by the corresponding instances of G^+. Formally we have:

Definition 2.2 Let P be a definite clause program. Consider the goal G : $G^+(?Y) \Leftarrow H$ where H is an atom of the form $h(T(X))$. h is a predicate symbol, $T(X)$ a vector of terms and X a vector of variables. Let $D1, \ldots, Dn$ be all clauses

D1: $\mathrm{H}_1 \Leftarrow \mathrm{H}_{11} \wedge \ldots \wedge \mathrm{H}_{1m_1} \wedge \mathrm{R}_1$.
...
Dn: $\mathrm{H}_n \Leftarrow \mathrm{H}_{n1} \wedge \ldots \wedge \mathrm{H}_{nm_n} \wedge \mathrm{R}_n$.

in P the heads of which unify with H by a mgu $\sigma_i \cup \tau_i, 1 \leq i \leq n$, such that $H\sigma_i \equiv H_i\tau_i$. Let R_i be a possibly empty conjunction of atoms containing no instance of the predicate h. It may be that $m_i = 0$. Then, if all instantiated atoms $H_{ij}\tau_i, 1 \leq j \leq m_i$, are instances of the atom H by a substitution θ_{ij} so that $H_{ij}\tau_i \equiv H\theta_{ij}$, *Computation Induction* applies and yields the new goals

I1: $\mathrm{G}^+(?\mathrm{Y})\sigma_1 \Leftarrow \mathrm{G}^+(\mathrm{Y}_{11})\theta_{11} \wedge \ldots \wedge \mathrm{G}^+(\mathrm{Y}_{1m_1})\theta_{1m_1} \wedge \mathrm{R}_1\tau_1$.
...
In: $\mathrm{G}^+(?\mathrm{Y})\sigma_n \Leftarrow \mathrm{G}^+(\mathrm{Y}_{n1})\theta_{n1} \wedge \ldots \wedge \mathrm{G}^+(\mathrm{Y}_{nm_n})\theta_{nm_n} \wedge \mathrm{R}_n\tau_n$.

where Y_{ij} are distinct renamings of Y as new vectors of free variables.

Example 2.5 Consider again the goal expressing commutativity of +

G: +(y,x,z) $\Leftarrow$ +(x,y,z).

Computation Induction yields

H1: +(y,0,y).
H2: +(y,succ(x_0),succ(z_0)) $\Leftarrow$ +(y,x_0,z_0).

3 NFI-driven Induction

We have pointed out that general Structural Induction does not preserve the class of implicative goals. On the other hand, Restricted Structural Induction preserves this class but does not allow the induction variable to occur in the hypothesis of the goal. We now introduce a form of Structural Induction, called "NFI-driven Induction", that at once preserves the class of implicative goals and allows induction variables to occur in goal hypotheses. The inference rule is derived by examining the formulas resulting from general Structural Induction. We consider the case when the induction variable is of type natural number, and following this example it is straight forward to apply the line of reasoning to any datastructure.

Consider the sentence

$$\text{S:}\quad \forall x,z\ \exists y\ C(x,z,y) \Leftarrow H(x,z)$$

A proof of S by general structural induction w.r.t. x (where x is a natural number) will result in the induction step formula

$$\begin{aligned}\text{I:}\quad \forall x\ (&\forall z\ \exists y\ C(succ(x),z,y) \Leftarrow H(succ(x),z)) \Leftarrow \\ &(\forall z\ \exists y\ C(x,z,y) \Leftarrow H(x,z))\end{aligned}$$

Transformed to prenex form, I becomes

$$\begin{aligned}\text{I}':\quad \forall x,z\ \exists y,z'\ \forall y'\ &(C(succ(x),z,y) \Leftarrow H(succ(x),z)) \Leftarrow \\ &(C(x,z',y') \Leftarrow H(x,z'))\end{aligned}$$

A natural attempt to prove I' would be to assume $H(succ(x),z)$, prove $H(x,z')$, apply the induction hypothesis to get $C(x,z',y')$ and finally prove $C(succ(x),z,y)$. Formally, this is achieved by proving the sentences

$$\text{A:}\quad \forall x,z\ \exists z'\ H(x,z') \Leftarrow H(succ(x),z).$$

$$\text{B:}\quad \forall x,z,z',y'\ \exists y\ C(succ(x),z,y) \Leftarrow C(x,z',y') \wedge Q(x,z,z')$$

where $Q(x,z,z')$ satisfies the conditions

$$\text{(Term):}\quad \forall x,z\ \exists z'\ Q(x,z,z') \Leftarrow H(succ(x),z)$$

$$\text{(Corr):}\quad \forall x,z,z'\ H(x,z') \Leftarrow Q(x,z,z') \wedge H(succ(x),z)$$

It is straightforward to verify that these conditions are sufficient and, indeed, necessary for A and B to guarantee that I' holds. Now we may transform the sentences A and B to the implicative goals

$$\text{A}':\quad H(x,?z') \Leftarrow H(succ(x),z)$$

$$\text{B}':\quad C(succ(x),z,?y) \Leftarrow C(x,z'',y') \wedge Q(x,z,z'')$$

Letting the subformula $Q(x,z,z')$ be the io-atom $io_{A'}(x,z,z')$ associated with the goal A', the conditions $(Term)$ and $(Corr)$ express termination and partial correctness of $io_{A'}(x,z,z')$ w.r.t. A'. The goal A' states the existence of a recursion scheme for $H(x,z)$ according to the structure of x. We will say that such a goal is an *assumption of structural recursion* of H w.r.t. x. Actually, A' is an induction axiom stating that every vector (x,z) of variables such that $H(x,z)$ holds, is either of the form $(0,z)$ or of the form $(succ(x'),z)$ where $io_{A'}(x',z,z')$ and $H(x',z')$ holds. Thus, $IO_{A'}$ induces a well-founded ordering relation $<$ on the data so that $(x',z') < (succ(x'),z)$ iff $io_{A'}(x',z,z')$.

Typically, a proof of B' starts with some applications of NFI to the atom $IO_{A'}$ in order to communicate the substitutions done during the proof of the assumption of structural recursion of H. This is why we call the induction rule "NFI-driven Induction". The proof of the assumption of structural recursion also usually starts by applying NFI. We present the rule of NFI-driven Induction for the case when the induction variable is of type natural number.

Definition 3.1 Let G be the goal $G^+(x, Z, ?Y) \Leftarrow G^-(x, Z)$ where x is a free variable of type natural number. Then *NFI-driven Induction* w.r.t. x yields the new goals

$$\begin{array}{ll} \text{H1:} & \mathrm{G}^+(0,\mathrm{Z},?\mathrm{Y}) \Leftarrow \mathrm{G}^-(0,\mathrm{Z}). \\ \text{H2:} & \mathrm{G}^-(\mathrm{w},?\mathrm{Z}') \Leftarrow \mathrm{G}^-(\mathrm{succ}(\mathrm{w}),\mathrm{Z}). \\ \text{H3:} & \mathrm{G}^+(\mathrm{succ}(\mathrm{w}),\mathrm{Z},?\mathrm{Y}) \Leftarrow \mathrm{G}^+(\mathrm{w},\mathrm{Z}'',\mathrm{Y}') \wedge \mathrm{IO}_{H2}(\mathrm{w},\mathrm{Z},\mathrm{Z}''). \end{array}$$

where $IO_{H2}(w, Z, Z')$ is the io-atom associated with the goal $H2$ and w is a new free variable of type natural number, Z' and Z'' are renamings of Z as new vectors of variables, and Y' is a renaming of Y as a new vector of free variables.

The io-procedure defining the io-atom $IO_G(x, Z, Y)$ associated with the goal G is shown in appendix A.

Example 3.1 Consider once more the goal expressing commutativity of +.

$$\text{G:} \quad +(\mathrm{y,x,z}) \Leftarrow +(\mathrm{x,y,z}).$$

By NFI-driven Induction w.r.t. x, we get

$$\begin{array}{ll} \text{H1:} & +(\mathrm{y},0,\mathrm{z}) \Leftarrow +(0,\mathrm{y},\mathrm{z}). \\ \text{H2:} & +(\mathrm{x},?\mathrm{y}_0,?\mathrm{z}_0) \Leftarrow +(\mathrm{succ}(\mathrm{x}),\mathrm{y},\mathrm{z}). \\ \text{H3:} & +(\mathrm{y},\mathrm{succ}(\mathrm{x}),\mathrm{z}) \Leftarrow +(\mathrm{y}_0,\mathrm{x},\mathrm{z}_0) \wedge \mathrm{io}_{H2}(\mathrm{x},\mathrm{y},\mathrm{z},\mathrm{y}_0,\mathrm{z}_0). \end{array}$$

NFI applied to $H1$ by $\{y/z\}$ yields

$$\text{I1:} \quad +(\mathrm{y},0,\mathrm{y}).$$

This may be proved by induction w.r.t. y, but we interrupt the proof of this branch. NFI applied to $H2$ by $\{succ(z_1)/z\}$ yields

$$\text{I2:} \quad +(\mathrm{x},?\mathrm{y}_0,?\mathrm{z}_0) \Leftarrow +(\mathrm{x},\mathrm{y},\mathrm{z}_1).$$

Simplification on $I2$ by $\{y/y_0, z_1/z_0\}$ yields the trivial goal. The io-program defining $IO_{H2}(x, y, z, z_0, y_0)$ becomes

$$\begin{array}{l} \mathrm{io}_{H2}(\mathrm{x},\mathrm{y},\mathrm{succ}(\mathrm{z}_1),\mathrm{z}_0,\mathrm{y}_0) \Leftarrow \mathrm{io}_{I2}(\mathrm{x},\mathrm{y},\mathrm{z}_1,\mathrm{y}_0,\mathrm{z}_0). \\ \mathrm{io}_{I2}(\mathrm{x},\mathrm{y},\mathrm{z}_1,\mathrm{y},\mathrm{z}_1). \end{array}$$

Two applications of NFI to $H3$ w.r.t. IO_{H2} by $\{succ(z_1)/z, y/y_0,\ z_1/z_0\}$ yields

$$\text{I3:} \quad +(\mathrm{y},\mathrm{succ}(\mathrm{x}),\mathrm{succ}(\mathrm{z}_1)) \Leftarrow +(\mathrm{y},\mathrm{x},\mathrm{z}_1).$$

NFI-driven Induction w.r.t. y yields

$$\begin{array}{ll} \text{J3:} & +(0,\mathrm{succ}(\mathrm{x}),\mathrm{succ}(\mathrm{z}_1)) \Leftarrow +(0,\mathrm{x},\mathrm{z}_1). \\ \text{J4:} & +(\mathrm{y},?\mathrm{x}_0,?\mathrm{z}_2) \Leftarrow +(\mathrm{succ}(\mathrm{y}),\mathrm{x},\mathrm{z}_1). \\ \text{J5:} & +(\mathrm{succ}(\mathrm{y}),\mathrm{succ}(\mathrm{x}),\mathrm{succ}(\mathrm{z}_1)) \Leftarrow +(\mathrm{y},\mathrm{succ}(\mathrm{x}_0),\mathrm{succ}(\mathrm{z}_2)) \wedge \\ & \qquad \mathrm{io}_{J4}(\mathrm{x},\mathrm{y},\mathrm{z}_1,\mathrm{x}_0,\mathrm{z}_2). \end{array}$$

The goal $J3$ is proved by an application of NFI followed by an application of Simplification. NFI on $J4$ by $\{succ(z_3)/z_1\}$ yields

K4: $+(y,?x_0,?z_2) \Leftarrow +(y,x,z_3)$.

Simplification by $\{x/x_0, z_3/z_2\}$ yields the trivial goal. The io-atom $io_{J4}(x, y, z_1, x_0, z_2)$ is defined by the io-program

$io_{J4}(x,y,succ(z_3),x_0,z_2) \Leftarrow io_{K4}(x,y,z_3,x_0,z_2)$.
$io_{K4}(x,y,z_3,x,z_3)$.

Consider the goal $J5$. Two applications of NFI w.r.t. the atom IO_{J4} by $\{succ(z_3)/z_1\}$ followed by $\{x/x_0, z_3/z_2\}$ yields

K5: $+(succ(y),succ(x),succ(succ(z_3))) \Leftarrow +(y,succ(x),succ(z_3))$.

DCI followed by Simplification yields the trivial goal.

4 NFI-driven Induction vs. Restricted Structural Induction

Restricted Structural Induction is a special case of NFI-driven Induction since when the induction variable does not occur in the hypothesis of the goal, the assumption of structural recursion generated by NFI-driven Induction is always true and can be proved by at most some applications of Simplification. The following example shows that NFI-driven Induction is actually a *proper* extension of the rule of Restricted Structural Induction.

Example 4.1 Commutativity of $+$ expressed as $+(x, y, z) \Leftarrow +(y, x, z)$ cannot be proved by Restricted Structural Induction since all free variables occur in the hypothesis. It does not solve the problem reformulating the goal as $+(x, y, ?z) \wedge +(y,x,?z)$. Consider the goal

G: $+(x,y,?z) \wedge +(y,x,?z)$.

Restricted Structural Induction w.r.t. x yields

H1: $+(0,y,?z) \wedge +(y,0,?z)$.
H2: $+(succ(x),y,?z) \wedge +(y,succ(x),?z) \Leftarrow +(x,y,z_0) \wedge +(y,x,z_0)$.

DCI on $H2$ w.r.t. the atom $+(succ(x), y, ?z)$ by $\{succ(z_1)/z\}$ yields

I2: $+(x,y,?z_1) \wedge +(y,succ(x),succ(?z_1)) \Leftarrow +(x,y,z_0) \wedge +(y,x,z_0)$.

Simplification by $\{z_0/z_1\}$ yields

J2: $+(y,succ(x),succ(z_0)) \Leftarrow +(y,x,z_0)$.

While true, the goal $J2$ would have to be proved by induction on y, but y occurs in the hypothesis so Restricted Structural Induction does not apply. However, the goal $J2$ can be proved by NFI-driven Induction (see example 3.1).

5 NFI-driven Induction vs. Computation Induction

Though introduced as a generalization of Restricted Structural Induction, NFI-driven Induction sometimes behaves as Computation Induction. Consider the goal $G: \quad G^{+}(x, Z, ?Y) \Leftarrow h(x, Z)$ where h is recursively defined as

h(0,Z) $\Leftarrow$ B(Z).
h(succ(x),Z) $\Leftarrow$ h(x,Z$'$) $\wedge$ R(x,Z,Z$'$).

NFI-driven Induction and Computation Induction yields essentially the same goals and io-programs (compare examples 2.5 and 3.1 where the goals $H1$ and $H2$ of example 2.5 are identical (up to renaming of variables) to the goals $I1$ and $I3$ of example 3.1).

The similarity between Computation Induction and NFI-driven Induction stems from the coincidence between the recursion scheme of h and the recursive structure of the datum x. In such a case the assumption of structural recursion of $h(x, Z)$ trivially holds. In general, when the recursion scheme of h differs from the recursion scheme of x, the rule of NFI-driven Induction w.r.t. x behaves differently from Computation Induction. Actually, in such cases NFI-driven Induction often fails, as illustrated by the following example.

Example 5.1 Let the predicates $odd(x)$ and $even(x)$ be defined by the program P

odd(succ(0)).
odd(succ(succ(x))) $\Leftarrow$ odd(x).
even(0).
even(succ(succ(x))) $\Leftarrow$ even(x).

Consider the goal

G: odd(succ(x)) $\Leftarrow$ even(x).

Computation Induction yields

H1: odd(succ(0)).
H2: odd(succ(succ(succ(x)))) $\Leftarrow$ odd(succ(x)).

DCI on $H1$ yields the trivial goal and DCI on $H2$ yields

I2: odd(succ(x)) $\Leftarrow$ odd(succ(x)).

Simplification yields the trivial goal. Let us see what happens when we try to prove the goal G by NFI-driven Induction. NFI-driven Induction on G yields

H1: odd(succ(0)) $\Leftarrow$ even(0).
H2: even(x) $\Leftarrow$ even(succ(x)).
H3: odd(succ(succ(x))) $\Leftarrow$ odd(succ(x)) $\wedge$ io$_{H2}$(x).

The goal $H2$ corresponds to the assumption of structural recursion of $even(x)$ w.r.t. x. This assumption is false here and NFI-driven Induction fails.

In order to make NFI-driven Induction behave as Computation Induction (and succeed) even in case h is not recursively defined according to the structure of x, we need a preprocessing of the goal G that will be explained in the next section.

6 NFI-driven Induction As a Simulation of Computation Induction

Structural Induction is in a way a special case of Computation Induction [Bur 69]. Since any data structure can be defined by a program, every proof by Structural Induction can be turned into a proof by Computation Induction (for example, natural numbers can be defined by the clauses $nat(0)$ and $nat(succ(x)) \Leftarrow nat(x)$, see [K&S 86]). In this section we take the opposite point of view to the effect that Computation Induction is a special case of Structural Induction. Instead of saying that every datatype is defined by a program, we say that every program defines a datatype. Consider the program P defining the atom $h(x)$

$$\begin{array}{l}
\mathrm{h(x)} \Leftarrow \mathrm{B_1(x)}.\\
\ldots\\
\mathrm{h(x)} \Leftarrow \mathrm{B}_n\mathrm{(x)}.\\
\mathrm{h(x)} \Leftarrow \mathrm{h(x_1)} \wedge \ldots \wedge \mathrm{h(x}_{k_1}) \wedge \mathrm{R_1(x,x_1,} \ldots, \mathrm{x}_{k_1}).\\
\ldots\\
\mathrm{h(x)} \Leftarrow \mathrm{h(x_1)} \wedge \ldots \wedge \mathrm{h(x}_{k_m}) \wedge \mathrm{R}_m\mathrm{(x,x_1,} \ldots, \mathrm{x}_{k_m}).
\end{array}$$

where $B_1, \ldots, B_n, R_1, \ldots, R_m$ are conjunctions of atoms not containing any occurrences of h or any atom defined (even indirectly) in terms of h. We introduce the datatype $comp_h$ associated with the atom h, consisting of n constant symbols $b_1, \ldots, b_n$ and m functors $c_1, \ldots, c_m$ of arity $k_i, 1 \leq i \leq m$, where b_j and c_i are new symbols. We extend the predicate $h(x)$ to the associated predicate $h'(x^h, x)$ defined by the program $P' \cup P$ where P' is

$$\begin{array}{ll}
\mathrm{h'(b_1,x)} \Leftarrow \mathrm{B_1(x)}. & \\
\ldots & \\
\mathrm{h'(b}_n\mathrm{,x)} \Leftarrow \mathrm{B}_n\mathrm{(x)}. & \\
\mathrm{h'(c_1(x^h{}_1,} \ldots, \mathrm{x^h{}_{k_1}),x)} \Leftarrow & \mathrm{h'(x^h{}_1,x_1)} \wedge \ldots \wedge \mathrm{h'(x^h{}_{k_1},x_{k_1})} \wedge \\
& \mathrm{R_1(x,x_1,} \ldots, \mathrm{x}_{k_1}).\\
\ldots & \\
\mathrm{h'(c}_m\mathrm{(x^h{}_1,} \ldots, \mathrm{x^h{}_{k_m}),x)} \Leftarrow & \mathrm{h'(x^h{}_1,x_1)} \wedge \ldots \wedge \mathrm{h'(x^h{}_{k_m},x_{k_m})} \wedge \\
& \mathrm{R}_m\mathrm{(x,x_1,} \ldots, \mathrm{x}_{k_m}).
\end{array}$$

It is clear that for any x, $h(x)$ is true in the least Herbrand model of P iff $h'(x^h, x)$ is true for some x^h in the least Herbrand model of $P' \cup P$. Thus, any goal $G : G^+(x) \Leftarrow G^-(x) \wedge h(x)$ is true in the least Herbrand model of P iff the goal $G' : G^+(x) \Leftarrow G^-(x) \wedge h'(x^h, x)$ is true in the least Herbrand model of $P' \cup P$. This is a generalisation of the program transformation process described in [E&McA 88]. A proof by Structural Induction w.r.t. the variable x^h will be identical to a proof by induction w.r.t. the depth of recursion. In particular, NFI-driven Induction w.r.t. x^h will generate goals and io-programs that are equivalent to those generated by Computation Induction.

Example 6.1 Consider the goal $G : odd(succ(x)) \Leftarrow even(x)$ in example 5.1. The atoms $even(x)$ and $odd(x)$ are defined by the program P

odd(succ(0)).
odd(succ(succ(x))) $\Leftarrow$ odd(x).
even(0).
even(succ(succ(x))) $\Leftarrow$ even(x).

The extended program becomes $P' \cup P$ where P' defines the predicate $even'(x^{even}, x)$

even′(b,0).
even′(c(x^{even}),succ(succ(x))) $\Leftarrow$ even′(x^{even},x).

Consider the goal G'

G′: odd(succ(x)) $\Leftarrow$ even′(x^{even},x).

NFI-driven Induction w.r.t. x^{even} yields the new goals

H1: odd(succ(x)) $\Leftarrow$ even′(b,x).
H2: even′(x^{even},?x$_0$) $\Leftarrow$ even′(c(x^{even}),x).
H3: odd(succ(x)) $\Leftarrow$ odd(succ(x$_0$)) $\wedge$ io$_{H2}$(x^{even},x,x$_0$).

The goal $H1$ is easily proved by NFI followed by DCI. NFI applied to $H2$ by $\{succ(succ(x_1))/x\}$ yields

I2: even′(x^{even},?x$_0$) $\Leftarrow$ even′(x^{even},x$_1$).

Simplification by $\{x_1/x_0\}$ yields the trivial goal. The associated io-atom $io_{H2}(x^{even}, x, x_0)$ is defined by the io-program

io$_{H2}$(x^{even},succ(succ(x$_1$)),x$_1$).

NFI applied to $H3$ w.r.t. to the atom IO_{H2} by $\{succ(succ(x_1))/x, x_1/x_0\}$ yields

I3: odd(succ(succ(succ(x$_1$)) $\Leftarrow$ odd(succ(x$_1$)).

DCI followed by simplification yields the trivial goal.

We see that by extending the predicate with a recursion variable, NFI-driven Induction subsumes Computation Induction.

7 Final Remarks

We have introduced the rule of NFI-driven Induction, a new method for inductive reasoning that merges the principles of Structural Induction and Computation Induction by assuming that the recursion is made w.r.t. the structure of the data. That this is not a strong assumption has been shown by interpreting a recursive program as an inductive definition of a data structure over which the recursion is implicitly made. NFI-driven Induction is a generalization of the rule of Restricted Structural Induction, and we have noted that by extending of the recursively defined predicates with a recursion variable NFI-driven Induction subsumes Computation Induction.

Though the NFI-driven Induction rule does not formally require that the hypothesis of a goal must be atomic, it often fails if it is not. The goal $H2$ in example 5.1 succeeds since the recursive calls of the two atoms are independent, and in the example in appendix C the goal $J6$ succeeds because the recursions, though not independent, happen to coincide. In general the computations of the atoms often interfere in such a way that the assumption of structural recursion is not true. For example $(x*y)*z = (x*z)*y$, "right commutativity of times" [Fri 91], expressed as the goal $G : *(x, y.u) \wedge *(u, z, w) \Leftarrow *(x, z, v) \wedge *(v, y, w)$ is not provable by NFI-driven Induction since the number of recursive calls of the atom $*(v, y, w)$ is a function of the number of recursive calls of $*(x, z, v)$ in a non trivial way.

We should not expect to find a general solution to this problem, but NFI-driven Induction may perhaps be refined to deal with commonly appearing cases such as functional composition and tupling (actually, the assumption of structural recursion is always provable when the predicates are compatible for composition or tupling [Bou 91]).

Appendix A: Proof of Correctness

Consider the goal G in the definition of NFI-driven Induction, (definition 3.1). The io-atom $IO_G(x, Z, Y)$ associated with the goal G is defined by

$$\mathrm{IO}_G(0,\mathrm{Z},\mathrm{Y}) \Leftarrow \mathrm{IO}_{H1}(\mathrm{Z},\mathrm{Y}).$$
$$\mathrm{IO}_G(\mathrm{succ(w)},\mathrm{Z},\mathrm{Y}) \Leftarrow \mathrm{IO}_{H2}(\mathrm{w},\mathrm{Z},\mathrm{Z}'') \wedge \mathrm{IO}_G(\mathrm{w},\mathrm{Z}'',\mathrm{Y}') \wedge \mathrm{IO}_{H3}(\mathrm{w},\mathrm{Z},\mathrm{Y}',\mathrm{Z}'',\mathrm{Y}).$$

theorem 1 The io-procedure associated with the rule of NFI-driven Induction preserves partial correctness and termination.

Proof of partial correctness: We have to show that if

(Corr1): $\forall \mathrm{Z},\mathrm{Y}\ \mathrm{G}^+(0,\mathrm{Z},\mathrm{Y}) \Leftarrow \mathrm{G}^-(0,\mathrm{Z}) \wedge \mathrm{IO}_{H1}(\mathrm{Z},\mathrm{Y})$

(Corr2): $\forall \mathrm{x},\mathrm{Z}\ \mathrm{Z}'\ \mathrm{G}^-(\mathrm{x},\mathrm{Z}') \Leftarrow \mathrm{G}^-(\mathrm{succ(x)},\mathrm{Z}) \wedge \mathrm{IO}_{H2}(\mathrm{x},\mathrm{Z},\mathrm{Z}')$

(Corr3): $\forall \mathrm{x},\mathrm{Y},\mathrm{Y}'\ \mathrm{Z}\ \mathrm{Z}''\ \mathrm{G}^+(\mathrm{succ(x)},\mathrm{Z}'',\mathrm{Y}) \Leftarrow \mathrm{G}^+(\mathrm{x},\mathrm{Z}'',\mathrm{Y}') \wedge \mathrm{IO}_{H2}(\mathrm{x},\mathrm{Z},\mathrm{Z}'') \wedge \mathrm{IO}_{H3}(\mathrm{x},\mathrm{Z},\mathrm{Y}',\mathrm{Z}'',\mathrm{Y})$

hold, then

$$\text{(Corrx):} \quad \forall \text{x,Y,Z } \text{G}^+(\text{x,Z,Y}) \Leftarrow \text{G}^-(\text{x,Z}) \wedge \text{IO}_G(\text{x,Z,Y})$$

holds. Suppose $(Corr1)$, $(Corr2)$ and $(Corr3)$ hold. We prove $(Corrx)$ by induction w.r.t. x.

Base case $(x = 0)$. Assume $G^-(0, Z)$ and $IO_G(0, Z, Y)$. By the only-if part of the io-procedure $IO_G(0, Z, Y)$ yields $IO_{H1}(Z, Y)$, which together with $(Corr1)$ yields $G^+(0, Z, Y)$. So the base case holds.

Induction step $(x = succ(w))$. Assume

$$\text{(ind. hyp.):} \quad \forall \text{Z,Y } \text{G}^+(\text{w,Z,Y}) \Leftarrow \text{G}^-(\text{w,Z}) \wedge \text{IO}_G(\text{w,Z,Y})$$

Suppose $G^-(succ(w), Z)$ and $IO_G(succ(w), Z, Y)$ hold. By the only-if part of the io-procedure, $IO_G(succ(w), Z, Y)$ gives us $IO_G(w, Z'', Y')$, $IO_{H2}(w, Z, Z'')$ and $IO_{H3}(w, Z, Y', Z'', Y)$, for some Y' and Z''. By $(Corr2)$, $G^-(succ(w), Z)$ and $IO_{H2}(w, Z, Z'')$ we get $G^-(w, Z'')$. Now, by the induction hypothesis, $G^-(w, Z'')$ and $IO_G(w, Z'', Y')$ give us $G^+(w, Z'', Y')$. And finally, $(Corr3)$, $G^+(w, Z'', Y')$, $IO_{H2}(w, Z, Z'')$ and $IO_{H3}(w, Z, Y', Z'', Y)$ yield $G^+(succ(w), Z'', Y)$. This concludes the proof of partial correctness.

Proof of termination: We have to show that if

$$\text{(Term1):} \quad \forall \text{Z } \exists \text{Y } \text{IO}_{H1}(\text{Z,Y}) \Leftarrow \text{G}^-(0,\text{Z})$$
$$\text{(Term2):} \quad \forall \text{x,Z } \exists \text{Z}' \text{ IO}_{H2}(\text{x,Z,Z}') \Leftarrow \text{G}^-(\text{succ(x),Z})$$
$$\text{(Term3):} \quad \forall \text{x,Y}' \text{Z,Z}'' \exists \text{Y } \text{IO}_{H3}(\text{x,Z,Y}',\text{Z}'',\text{Y}) \Leftarrow \text{G}^+(\text{x,Z}'',\text{Y}') \wedge \text{IO}_{H2}(\text{x,Z,Z}'')$$

hold, then

$$\text{(Termx):} \quad \forall \text{x,Z } \exists \text{Y } \text{IO}_G(\text{x,Z,Y}) \Leftarrow \text{G}^-(\text{x,Z})$$

holds. Suppose $(Term1)$, $(Term2)$ and $(Term3)$ holds. We prove $(Termx)$ by induction w.r.t. x.

Base case $(x = 0)$. Assume $G^-(0, Z)$. By $(Term1)$ $IO_{H1}(Z, Y)$ holds for some Y. By the io-procedure we get $IO_G(0, Z, Y)$, so the base case holds.

Induction step $(x = succ(w))$. Assume

$$\text{(ind. hyp.):} \quad \forall \text{Z } \exists \text{Y } \text{IO}_G(\text{w,Z,Y}) \Leftarrow \text{G}^-(\text{w,Z})$$

Suppose $G^-(succ(w), Z)$ holds. By $(Term2)$ we get $IO_{H2}(w, Z, Z')$ for some Z'. $G^-(succ(w), Z)$, $IO_{H2}(w, Z, Z')$ and $(Corr2)$ yield $G^-(w, Z')$ which by the induction hypothesis gives us $IO_G(w, Z', Y)$ for some Y. From $(Corrx)$, $IO_G(w, Z', Y)$ and $G^-(w, Z')$ we get $G^+(w, Z', Y)$. Now $IO_{H2}(p, w, Z, Z')$, $G^+(w, Z', Y)$ and $(Term3)$ give us, for some Y', $IO_{H3}(w, Z, Y, Z', Y')$. Then finally by the io-procedure we have $IO_G(succ(w), Z, Y')$. This concludes the proof of termination.

Appendix B: Commutativity of Times

Let the predicates $*(x, y, z)$ and $+(x, y, z)$ be defined by the program P

$+(0,y,y).$
$+(succ(x),y,succ(z)) \Leftarrow +(x,y,z).$
$*(0,y,0).$
$*(succ(x),y,z) \Leftarrow *(x,y,w) \wedge +(y,w,z).$

Consider the goal G expressing the commutativity of times

G: $*(x,y,z) \Leftarrow *(y,x,z).$

NFI-driven Induction w.r.t. y yields

H1: $*(x,0,z) \Leftarrow *(0,x,z).$
H2: $*(y,?x_0,?z_0) \Leftarrow *(succ(y),x,z).$
H3: $*(x,succ(y),z) \Leftarrow *(x_0,y,z_0) \wedge io_{H2}(y,x,z,x_0,z_0).$

By some applications of NFI and Simplification these three goals yields the two new goals

I1: $*(x,0,0).$
I3: $*(x,succ(y),z) \Leftarrow *(x,y,w_0) \wedge +(x,w_0,z).$

NFI-driven Induction applied to $I1$ yields the new goals

J1: $*(0,0,0).$
J4: $*(succ(x),0,0) \Leftarrow *(x,0,0).$

which are easily proved. NFI-driven Induction w.r.t. x applied to $I3$ yields

J5: $*(0,succ(y),z) \Leftarrow *(0,y,w_0) \wedge +(0,w_0,z).$
J6: $*(x,?y_1,?w_1) \wedge +(x,?w_1,?z_1) \Leftarrow *(succ(x),y,w_0) \wedge$
$+(succ(x),w_0,z).$
J7: $*(succ(x),succ(y),z) \Leftarrow *(x,succ(y_1),z_1) \wedge$
$io_{J6}(x,y,z,w_0,y_1,z_1,w_1).$

$J5$ is proved by two applications of NFI and one application of DCI. Now, the assumption $J6$ will have to be proved by induction and this will reappear in the proof of the goal $J7$, so let us look at this part of the proof in more detail. Two applications of NFI applied to the goal $J6$ by $\{succ(z_2)/z\}$ yields

K6: $*(x,?y_1,?w_1) \wedge +(x,?w_1,?z_1) \Leftarrow *(x,y,w_2) \wedge +(y,w_2,w_0) \wedge$
$+(x,w_0,z_2).$

Simplification by $\{y/y_1, w_2/w_1\}$ yields

L6: $+(x,w_2,?z_1) \Leftarrow +(y,w_2,w_0) \wedge +(x,w_0,z_2).$

NFI-driven induction w.r.t. x yields

M8: $+(0,w_2,?z_1) \Leftarrow +(y,w_2,w_0) \wedge +(0,w_0,z_2).$
M9: $+(?y_3,?w_3,?w_4) \wedge +(x,?w_4,?z_3) \Leftarrow +(y,w_2,w_0) \wedge$
$+(succ(x),w_0,z_2).$
M10: $+(succ(x),w_2,?z_1) \Leftarrow +(x,w_3,z_4) \wedge$
$io_{M9}(x,y,z_2,w_0,w_2,w_3,w_4,z_3).$

DCI on $M8$ yields the trivial goal. NFI on $M9$ by $\{succ(z_5)/z_2\}$ yields

N9: $+(?y_3,?w_3,?w_4) \wedge +(x,?w_4,?z_3) \Leftarrow +(y,w_2,w_0) \wedge +(x,w_0,z_5)$.

Two applications of Simplification by $\{y/y_3, w_2/w_3, w_0/w_4, z_5/z_3\}$ will yield the trivial goal. Three applications of NFI on $M10$ w.r.t. the atom IO_{M9} yields the goal

N10: $+(succ(x),w_2,?z_1) \Leftarrow +(x,w_2,z_4)$.

Applying DCI by $\{succ(z_6)/z_1\}$ and Simplification by $\{z_4/z_6\}$ we get the trivial goal. The atom $io_{J6}(x, y, z, w_0, y_1, z_1, w_1)$ is defined by

$io_{J6}(x,y,succ(z_2),w_0,y,z_1,w_2) \Leftarrow io_{L6}(x,w_0,w_2,y,z_2,z_1)$.

$io_{L6}(0,w_0,w_2,y,w_0,w_2) \Leftarrow +(y,w_2,w_0)$.
$io_{L6}(succ(x),w_0,w_2,y,succ(z_5),succ(z_4)) \Leftarrow io_{L6}(x,w_0,w_2,y,z_5,z_4)$.

Consider the goal $J7$. NFI by $\{succ(z_2)/z\}$ yields

K7: $*(succ(x),succ(y),succ(z_2)) \Leftarrow *(x,succ(y),z_1) \wedge$
$io_{L6}(x,w_0,w_2,y,z_2,z_1)$.

Two applications of DCI followed by Simplification yields

L7: $+(y,z_1,z_2) \Leftarrow io_{L6}(x,w_0,w_2,y,z_2,z_1)$.

Now we apply NFI-driven Induction w.r.t. x

M11: $+(y,z_1,z_2) \Leftarrow io_{L6}(0,w_0,w_2,y,z_2,z_1)$.
M12: $io_{L6}(x,?w_7,?w_8,?y_7,?z_8,?z_7) \Leftarrow io_{L6}(succ(x),w_0,w_2,y,z_2,z_1)$.
M13: $+(y,z_1,z_2) \Leftarrow +(y_7,z_7,z_8) \wedge$
$io_{M12}(x,w_0,w_2,y,z_1,z_2, w_7,w_8,y_7,z_7,z_8)$.

NFI on $M11$ yields

N11: $+(y,z_1,z_2) \Leftarrow +(y,z_1,z_2)$.

Simplification yields the trivial goal. NFI applied to $M12$ by $\{succ(z_5)/z_2, succ(z_4)/z_1\}$ yields

N12: $io_{L6}(x,?w_7,?w_8,?y_7,?z_8,?z_7) \Leftarrow io_{L6}(x,w_0,w_2,y,z_5,z_4)$.

Simplification by $\{w_0/w_7, w_2/w_8, y/y_7, z_4/z_7, z_5/z_8\}$ will result in the trivial goal. Now, NFI on $M13$ w.r.t. IO_{M12} yields

N13: $+(y,succ(z_4),succ(z_5)) \Leftarrow +(y,z_4,z_5)$.

N13 is easily proved by one application of NFI-driven Induction (see example 3.1).

References

[Bou 91] Bouverot A., *Extracting and Transforming Logic Programs.* Technical Report LIENS-91-4, Laboratoire d'Informatique de l'Ecole Normale Supérieure, Paris, 1991.

[Bur 69] Burstall R. M., *Proving Properties of Programs by Structural Induction.* Comput. J., vol. 12, February 1969, pp. 41-48.

[Cla 78] Clark K. L., *Negation As Failure.* In Logic and Databases, H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 1978, pp. 293-322.

[Cla 79] Clark K. L., *Predicate Logic as a Computational Formalism.* Research Monograph 79/59, TOC, Imperial College, 1979.

[E&McA 88] Elkan C. and McAllester D., em Automated Inductive Reasoning about Logic Programs. Proc. 5th Intl. Conf. and Symp. on Logic Programming, 1988.

[Fri 90] Fribourg L., *Extracting Logic Programs from Proofs that use Extended Prolog Execution and Induction.* Proc. 7th Intl. Conf. on Logic Programming, Jerusalem, 1990, pp. 685-699.

[Fri 91] Fribourg L., *Automatic Generation of Simplification Lemmas for Inductive Proofs.* Proc. Intl. Symp. on Logic Programming, San Diego, 1991, pp. 103-116.

[K&S 86] Kanamori T. and Seki H., *Verification of Prolog Programs Using an Extension of Execution.* In (Shapiro E., ed.) 3rd International Conference on Logic Programming, Lecture Notes in Computer Science 225, 1986, pp. 475-489.

[M&V 72] Manna Z. and Vuillemin J., *Fixpoint Approach to the Theory of Computation.* Commun. Ass. Comput. Mach., vol. 15, no. 7, 1972, pp. 528-536

[R&Y 76] Reynolds C. and Yeh R. T., *Induction as the Basis for Program Verification.* In IEEE Transactions on Software Enginering, vol. SE-2, no. 4, December 1976.

[Rog 67] Rogers H.R., *Theory of Recursive Functions and Effective Computability.* N.Y., McGraw-Hill. th IEEE Symp. on Logic Programming, San Fransisco, 1967, pp. 215-223.