Transforming Inductive Definitions

Maurizio Proietti
IASI-CNR
Viale Manzoni 30, I-00185 Roma, Italy
proietti@iasi.rm.cnr.it

Alberto Pettorossi
DISP, University of Roma Tor Vergata,
Via di Tor Vergata, I-00133 Roma, Italy
adp@iasi.rm.cnr.it

Abstract

The main goal of this paper is to provide a common foundation to the theories of correctness of program transformations for a large variety of programming languages.

We consider the notion of *rule set* and the notion of *inductive set* which is defined by a rule set. We also consider a class of transformations of rule sets, called *rule replacements*, which replace an old rule set by a new rule set. These replacements can be viewed as generalizations of the most commonly used transformations, such as folding and unfolding.

We study two methods for proving the correctness of rule replacements, that is, for showing that the old rule set and the new rule set define the same inductive set. These methods are: (i) the Unique Fixpoint Method, based on the well-foundedness property of the new rule set, and (ii) the Improvement Method, based on the fact that the premises of the old rule set are replaced by premises which have 'smaller' proofs w.r.t. a suitable well-founded relation. Our Unique Fixpoint and Improvement Methods generalize many methods described in the literature which deal with transformation rules for functional and logic programming languages. Our methods have also the advantages of: (i) being parametric w.r.t. the well-founded relation which is actually used, and (ii) being applicable to rules with finite or infinite sets of premises.

1 Introduction

A large number of program transformation rules, which we will also call *transformations*, for brevity, have been proposed in the literature (see, for instance, [4, 13] and [9] for a survey). These transformations may vary according to the programming language considered and the semantics to be preserved. For instance, program transformations have been proposed for functional languages with eager or lazy semantics, for logic languages with least Herbrand model or computed answer semantics, for constraint and concurrent languages. For each language, the correctness of these transformations is usually proved from scratch. The main goal of this paper is to provide a common foundation to many different transformations proposed for the different programming languages.

A methodology which is often used for defining the semantics of programming languages, is based on the concept of *inductive definitions* presented by means of *rule sets* (see, for instance, [1] for a survey of some theoretical results on inductive definitions, and [16] for their application to the definition of the operational semantics of some imperative, functional, and concurrent programming languages). A rule is pair of the form (X/y), also written as $\frac{X}{y}$, where X is a set of *premises* and y is a *conclusion*.

For instance, in the case of a simple recursive equation language with (i) integer constants n, (ii) variables x, (iii) binary operators **op** (such as $+$ and $\times$), (iv) the **if-then-else** construct, (iv) function applications, and (v) function declarations, we may define the operational semantics of call-by-value by the infinite set R of rules which are derived, as we will indicate below, from the following five schematic rules (here and in what follows we feel free to omit the curly brackets when writing a *set* of premises) [16]:

$$(r1)\quad \frac{}{n \to n} \qquad\qquad (r2)\quad \frac{t_1 \to n_1 \qquad t_2 \to n_2}{t_1 \ \mathbf{op}\ t_2 \ \to\ n} \ \text{ if } n = (n_1 \ op\ n_2)$$

$$(r3)\quad \frac{t_0 \to 0 \qquad\quad t_1 \to n_1}{\mathbf{if}\ t_0\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2 \ \to\ n_1} \qquad (r4)\quad \frac{t_0 \to n_0 \qquad\quad t_2 \to n_2}{\mathbf{if}\ t_0\ \mathbf{then}\ t_1\ \mathbf{else}\ t_2 \ \to\ n_2} \ \text{ if } n_0 \neq 0$$

$$(r5)\quad \frac{t_1 \to n_1 \quad \ldots \quad t_k \to n_k \quad d[n_1/x_1,\ldots,n_k/x_k] \to n}{f(t_1,\ldots,t_k) \ \to\ n}$$

where: (i) f is assumed to be defined by the declaration $f(x_1,\ldots,x_k) = d$, (ii) $d[n/x]$ denotes the substitution of x by n in the expression d, and (iii) op is the semantic operator corresponding to the syntactic operator **op**.

Every rule in R is obtained as an instance of a schematic rule by replacing the metavariables $n, n_0, \ldots, t_0, t_1, \ldots,$ by suitable elements of the set $\mathbf{N}$ of integers and the set **CTerms** of closed terms so that the given constraints (such as $n_0 \neq 0$ in rule $r4$) are satisfied.

Thus, the rule $\frac{3+1 \ \to\ 4 \quad 2 \ \to\ 2}{(3+1)+2 \ \to\ 6}$ is in R because it is an instance of the schematic rule $r2$.

The set R of rules defines the binary relation $\to$ as a subset of $\mathbf{CTerms} \times \mathbf{N}$ which is said to be the *inductive set* defined by R. The relation $\to$ is constructed as follows:
(i) take the following operator T_R from the powerset of $\mathbf{CTerms}\times\mathbf{N}$ to the powerset of $\mathbf{CTerms} \times \mathbf{N}$:

$$T_R(A) = \{(t \to n) \in \mathbf{CTerms} \times \mathbf{N} \mid \text{there exists } \frac{X}{(t \to n)} \in R \text{ and } X \subseteq A\}$$

and then
(ii) take the relation $\to$ to be the least fixpoint of T_R (recall that T_R is a monotonic operator on the complete lattice given by the powerset of $\mathbf{CTerms} \times \mathbf{N}$ ordered by $\subseteq$).

The methodology for specifying the semantics of programs based on inductive definitions, generalizes the technique for specifying the semantics of a definite logic program P which consists in taking the least fixpoint of the immediate consequence operator T_P [15]. Indeed, in the case a definite logic program P the set of rule instances is the set $Ground(P)$ of all ground instances of the clauses of P, and the operator $T_{Ground(P)}$ is equal to T_P.

Similarly, by means of rule sets one may define other semantics for logic programs such as the *computed answer* and the *finite failure* semantics (see, for instance, [7]).

In order to study the correctness of program transformations in a way that is independent of the specific programming language, in this paper we assume that the semantics of the language in use is defined by means of a set of rules whose premises and conclusions belong to a suitable domain.

Our approach relies on the fact that the transformation of a program P into a program Q induces a transformation on the rule set, say $R(P)$, that defines the semantics of P into the rule set, say $R(Q)$, that defines the semantics of Q. Thus, the preservation of the semantics of P which expresses the *correctness* of the transformation as we will specify below, can be established by proving that the two sets of rules $R(P)$ and $R(Q)$ define the same inductive set.

In this paper we consider transformations of sets of rules, called *rule replacements*, which are very general. Indeed, we have that the most commonly used program transformations, such as folding and unfolding [4], induce rule replacements.

Usually, a program transformation is a *conditional* transformation, that is, a transformation which is applied to a program P only if a given property holds w.r.t. the semantics of P. Typical properties which are required to hold, are often expressed as *implications* or *equivalences.* Thus, we are lead to consider the problem of showing the correctness of a rule replacement when a suitable implication or equivalence formula holds in the inductive set defined by the given rule set. For instance, consider the rule set:

$$R: \quad \frac{}{a} \qquad \frac{a}{b} \qquad \frac{a \;\; b}{c}$$

We have that the inductive set $I(R)$ defined by R is $\{a, b, c\}$. We may say that the sets $\{a, b\}$ and $\{a\}$ of premises are equivalent w.r.t. the inductive set $I(R)$ in the sense that $\{a, b\}$ is a subset of $I(R)$ iff so is $\{a\}$. Now, if we replace the rule $\frac{a \;\; b}{c}$ with the set $\{a, b\}$ of premises, by the rule $\frac{a}{c}$ with the equivalent set $\{a\}$ of premises (and this replacement can be viewed as an unfolding transformation whereby b is replaced by a according to the second rule $\frac{a}{b}$), then we get:

$$S_1: \quad \frac{}{a} \qquad \frac{a}{b} \qquad \frac{a}{c}$$

and we have that the inductive set $I(S_1)$ defined by the new set S_1 of rules, is equal to $I(R)$.

Also $\{a\}$ and $\{b\}$ are equivalent w.r.t. $I(R)$. Unfortunately, in spite of this equivalence, if we replace a with b in the premise of the second rule of R (and this replacement can be viewed as a folding transformation), we get the following new set of rules:

$$S_2: \quad \frac{}{a} \qquad \frac{b}{b} \qquad \frac{a \;\; b}{c}$$

and now we have that the inductive set defined by S_2, which is $\{a\}$, is *not* equal to $I(R)$.

This example shows that when transforming an old rule set R into a new rule set by replacing some rules in R with other rules which have equivalent sets of premises w.r.t. $I(R)$, one may change the corresponding inductive set. Below we will present some sufficient conditions which ensure the preservation of the inductive sets, both in a weak form (*partial correctness*) and in a strong form (*total correctness*), and we will also present two methods which guarantee total correctness.

The plan of the paper is as follows. In Section 2 we give some preliminary definitions concerning rule sets and the corresponding inductive sets, and we present some basic results. In Section 3 we introduce our rule replacement transformation and we prove its partial correctness and completeness. Then in Sections 4 and 5, we consider two methods, namely the Unique Fixpoint Method and the Improvement Method, to prove the total correctness of a rule replacement. In Section 6 we provide a generalization of the Improvement Method where the notion of improvement is parametric w.r.t. the measure of the proof trees constructed via rule sets.

2 Inductive Definitions by Rule Sets

In this section we summarize some well-known basic concepts and results in the theory of inductive definitions (see, for instance, [1, 16]).

Definition 1 (Rule Set) Let D be a set.
(i) A *rule* over D is a pair (X/y) where $X \subseteq D$ and $y \in D$. X is said to be the set of *premises* of the rule and y is said to be the *conclusion* of the rule. A rule (X/y) will also be written as

$$\frac{\quad}{y} \quad \text{if } X = \emptyset \text{, and as} \quad \frac{x_1 \ \ldots \ x_n \ \ldots}{y} \quad \text{if } X = \{x_1, \ldots, x_n, \ldots\}.$$

(ii) A *rule set* over D is a (finite or infinite) set of rules over D.
(iii) Given a rule set R over D, a set $A \subseteq D$ is *R-closed* iff for every rule (X/y) in R, if $X \subseteq A$ then $y \in A$.

Given a rule set R over D we can introduce, as indicated in the following definitions, a set $I(R) \subseteq D$, called the *inductive set* defined by R. Let $\wp(D)$ denote the powerset of D.

Definition 2 (Immediate Consequence Operator) Let R be a rule set over D. (i) The *immediate consequence operator associated with* R is the function $T_R : \wp(D) \rightarrow \wp(D)$, defined as follows:

$$T_R(A) = \{y \in D \mid \text{there exists } (X/y) \in R \text{ and } X \subseteq A\}$$

(ii) A set $A \subseteq D$ is a *fixpoint* of T_R iff $T_R(A) = A$. A is a *prefixpoint* of T_R iff $T_R(A) \subseteq A$. A is a *postfixpoint* of T_R iff $T_R(A) \supseteq A$.

T_R is a monotonic operator on the complete lattice $\wp(D)$ ordered by $\subseteq$. We have the following properties.

Theorem 1 (i) *T_R has a least fixpoint, denoted by $lfp(T_R)$, which is equal to its least prefixpoint.*
(ii) *T_R has a greatest fixpoint, denoted by $gfp(T_R)$, which is equal to its greatest postfixpoint.*

Definition 3 (Inductive Set) *The* inductive set *defined by a rule set R over D, denoted by $I(R)$, is the least fixpoint of the operator T_R.*

It follows directly from the definitions and Theorem 1 that $I(R)$ is the *smallest R-closed subset of D.*

3 Transformations of Rule Sets

In this section we consider the *rule replacement* transformation which given a rule set R, generates a new rule set S by replacing some rules of R by new rules, if suitable implications or equivalences hold w.r.t. the inductive set $I(R)$. We present a sufficient condition for the partial correctness of rule replacement, that is, a condition which ensures that $I(R) \supseteq I(S)$. We also show that the rule replacement transformation is *complete*, in the sense that given any two rule sets which define the same inductive set, we can derive one from the other by suitable rule replacements.

Every set of rules can be partitioned into subsets of rules which have the same conclusion. Thus, every rule replacement which derives the rule set S from the rule set R, can be viewed as replacing, for each conclusion d, the set R_d of the sets of premises relative to d, by a new set S_d of sets of premises. In what follows we will study the correctness of rule replacements when suitable implications or equivalences hold between the sets of sets of premises which correspond to the same conclusion.

We start off by defining the following concept of implication between sets of sets of premises.

Definition 4 Let R be a rule set over D and $\mathbf{X}$, $\mathbf{Z}$ be sets of subsets of D. We write:
(i) $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ iff for all $X \in \mathbf{X}$, if $X \subseteq I(R)$ then there exists $Z \in \mathbf{Z}$ such that $Z \subseteq I(R)$,
(ii) $I(R) \models \mathbf{X} \leftarrow \mathbf{Z}$ iff $I(R) \models \mathbf{Z} \rightarrow \mathbf{X}$,
(iii) $I(R) \models \mathbf{X} \leftrightarrow \mathbf{Z}$ iff $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ and $I(R) \models \mathbf{X} \leftarrow \mathbf{Z}$.

The following properties follow directly from the definition. Let us consider a rule set R over D. Let A be a subset of D and $\mathbf{W}$, $\mathbf{X}$, and $\mathbf{Z}$ be sets of subsets of D. We have that:

(i) if $\forall X \in \mathbf{X}.\ \exists Z \in \mathbf{Z}.\ Z \subseteq X$ then $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$

(ii) for any $\mathbf{X} \subseteq \mathbf{Z}$, $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$, and in particular,
for any $\mathbf{X}$, $I(R) \models \{\} \rightarrow \mathbf{X}$, $I(R) \models \mathbf{X} \rightarrow \wp(D)$, and $I(R) \models \mathbf{X} \rightarrow \mathbf{X}$ (*reflexivity*)

(iii) for any $\mathbf{X}$, $\mathbf{W}$, and $\mathbf{Z}$, if $I(R) \models \mathbf{X} \rightarrow \mathbf{W}$ and $I(R) \models \mathbf{W} \rightarrow \mathbf{Z}$ then $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ (*transitivity*)

(iv) for any A, $\mathbf{X}$, $\mathbf{W}$, and $\mathbf{Z}$, if $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ then
$I(R) \models \mathbf{W} \cup \mathbf{X} \ \rightarrow\ \mathbf{W} \cup \mathbf{Z}$ (*monotonicity* w.r.t. $\cup$) and
$I(R) \models \{A \cup X \mid X \in \mathbf{X}\} \ \rightarrow\ \{A \cup Z \mid Z \in \mathbf{Z}\}$ (*distributivity*).

Definition 5 (Rule Replacement) (i) A *rule replacement* ρ is a pair (R, S) of rule sets over D, denoted by $\rho : R \mapsto S$. Given $\rho : R \mapsto S$, we say that the rule set S *is derived from* the rule set R *via* ρ.

(ii) Given a rule set R over D, for each $d \in D$ we define $R_d = \{X \subseteq D \mid (X/d) \in R\}$.

(iii) The rule replacement $\rho : R \mapsto S$ is *implication-based*, written $I(R) \models R \Rightarrow S$, iff for all $d \in D$, $I(R) \models R_d \rightarrow S_d$.
ρ is *reverse-implication-based*, written $I(R) \models R \Leftarrow S$, iff for all $d \in D$, $I(R) \models R_d \leftarrow S_d$.
ρ is *equivalence-based*, written $I(R) \models R \Leftrightarrow S$, iff for all $d \in D$, $I(R) \models R_d \leftrightarrow S_d$.

(iv) The rule replacement $\rho : R \mapsto S$ is *partially correct* iff $I(R) \supseteq I(S)$; it is *increasing* iff $I(R) \subseteq I(S)$; it is *totally correct* iff it is partially correct and increasing, that is, $I(R) = I(S)$.

The rules for unfolding, folding, and goal replacement usually considered in logic program transformation are examples of rule replacements. For instance, let us consider the following logic program:

$$P \;=\; \{p \leftarrow q, r \qquad q \leftarrow t \qquad q \leftarrow \qquad r \leftarrow\}.$$

Every clause $h \leftarrow body$ of P can be viewed as the rule $\dfrac{body}{h}$ over the domain $D = \{p, q, r, t\}$. By unfolding the atom q in the first clause of P, we get the new program:

$$Q \;=\; \{p \leftarrow t, r \qquad p \leftarrow r \qquad q \leftarrow t \qquad q \leftarrow \qquad r \leftarrow\}.$$

We have that $I(P) = \{p, q, r\}$. We also have that for each element $d \in D$, $I(P) \models P_d \leftrightarrow Q_d$. Indeed,
for p, $I(P) \models \{\{q, r\}\} \leftrightarrow \{\{t, r\}, \{r\}\}$,
for q, $I(P) \models \{\{t\}, \emptyset\} \leftrightarrow \{\{t\}, \emptyset\}$,
for r, $I(P) \models \{\emptyset\} \leftrightarrow \{\emptyset\}$, and
for t, $I(P) \models \{\} \leftrightarrow \{\}$. Thus, $I(P) \models P \Leftrightarrow Q$.

We show below that reverse-implication-based rule replacement is partially correct. However, in general, equivalence-based rule replacement is not increasing, and thus, it is not totally correct, that is, $I(R) \models R \Leftrightarrow S$ does not imply $I(R) = I(S)$.

To see this, let us consider the following rule replacement $\rho : R \mapsto S$, where R and S are the following rule sets over $D = \{a, b\}$:

$$R : \frac{}{a} \quad \frac{a}{b} \qquad\qquad S : \frac{}{a} \quad \frac{b}{b}$$

We have that $I(R) = \{a, b\}$. $I(R) \models R \Leftrightarrow S$ holds because: (i) for a we have that $I(R) \models \{\emptyset\} \leftrightarrow \{\emptyset\}$, and (ii) for b we have that $I(R) \models \{\{a\}\} \leftrightarrow \{\{b\}\}$. However, $I(S) = \{a\} \neq I(R)$.

The following theorem shows that equivalence-based rule replacement is *complete* for the derivation of equivalent rule sets, in the sense that for any two rule sets, say R and S, which define the same inductive set, we can derive S from R via an equivalence-based rule replacement.

Theorem 2 (Completeness of Equivalence-Based Rule Replacement) *If $I(R) = I(S)$ then $I(R) \models R \Leftrightarrow S$.*

Proof. Since $I(S)$ is a fixpoint of T_S and $I(R) = I(S)$, we have that $I(R) = T_S(I(R))$.

We have to prove that for any $d \in D$, $I(R) \models R_d \rightarrow S_d$ and $I(R) \models S_d \rightarrow R_d$. Let us now consider an element $d \in D$.

If $X \in R_d$ and $X \subseteq I(R)$ then $d \in I(R)$, because $I(R)$ is R-closed. Then, because $I(R) = T_S(I(R))$, there exists $Z \in S_d$ such that $Z \subseteq I(R)$. Thus, we have that $I(R) \models R_d \rightarrow S_d$.

If $Z \in S_d$ and $Z \subseteq I(R)$ then $d \in I(R)$, because $I(R) = T_S(I(R))$. Since $I(R) = T_R(I(R))$, there exists $X \in R_d$ such that $X \subseteq I(R)$. Thus, we have that $I(R) \models S_d \rightarrow R_d$. □

We end this section by proving that reverse-implication-based rule replacements are partially correct.

Theorem 3 (Partial Correctness) *If $I(R) \models R \Leftarrow S$ then $I(R) \supseteq I(S)$.*

Proof. If $d \in T_S(I(R))$ then, by definition of T_S, there exists $(Z/d) \in S$ (i.e., $Z \in S_d$) and $Z \subseteq I(R)$. Since $I(R) \models R \Leftarrow S$, there exists $(X/d) \in R$ (i.e., $X \in R_d$) such that $X \subseteq I(R)$. By the fact that $I(R)$ is R-closed, it follows that $d \in I(R)$. Thus, $T_S(I(R)) \subseteq I(R)$, that is, $I(R)$ is a prefixpoint of T_S and, since $lfp(T_S)$ is the *least* prefixpoint of T_S (see Theorem 1), we have that $I(R) \supseteq lfp(T_S) = I(S)$. □

4 The Unique Fixpoint Method

In this section we study a method for proving that a rule replacement $\rho : R \mapsto S$ is increasing, that is $I(R) \subseteq I(S)$. This method is based on the fact that, when a rule set R is *terminating* in the sense specified below, its corresponding immediate consequence operator T_R has a unique fixpoint. By combining the method for proving that $\rho : R \mapsto S$ is increasing, with the Partial Correctness Theorem 3, we have a method for for showing that $I(R) = I(S)$, that is, the *total correctness* of the rule replacement.

Other sufficient conditions for showing the total correctness of rule replacements, based on the notion of *proof*, will be considered in the subsequent sections.

Definition 6 A rule set R over D is *terminating* iff for all d_0 in D there is no infinite sequence $s = \langle d_0, d_1, \ldots, d_i, d_{i+1}, \ldots \rangle$ of elements of D, such that for all d_i and d_{i+1} in s, there exists a rule (X/d_i) in R and d_{i+1} in X.

A well-founded relation is a binary relation $\succ$ on a set D such that there is no infinite sequence $d_0 \succ d_1 \succ \ldots \succ d_i \succ d_{i+1} \succ \ldots$ of elements of D.

Theorem 4 (Characterization of Terminating Rule Sets) *A rule set R over D is terminating iff there exists a well-founded relation $\succ$ on D such that for all rules (X/y) in R and for all x in X we have that $y \succ x$.*

Proof. (If part.) Let s be a sequence $\langle d_0, d_1, \ldots, d_i, d_{i+1}, \ldots \rangle$ of elements of D. Then for all d_i and d_{i+1} in s we have that $d_i \succ d_{i+1}$ and thus, by the hypothesis that $\succ$ is well-founded, we have that s is finite.
(Only if part.) Let us consider the relation $\succ$ on D such that $y \succ x$ iff there exists a rule $(X/y) \in R$ and $x \in X$. If R is terminating then $\succ$ is a well-founded relation. □

Bezem [2] gives a characterization of terminating definite logic programs which is similar to our characterization of terminating rule sets. In [2] a program P is said to be terminating iff for all ground goals G, all SLD-derivations of P starting from G are finite. Bezem's result says that a definite logic program P is terminating iff there exists a function $|.|$ from the set B_P of all ground atoms in the Herbrand universe to the set of natural numbers such that, for all ground instances $A \leftarrow B_1, \ldots, B_n$ of clauses in P, and for all $i \in \{1, \ldots, n\}$, we have that $|A| > |B_i|$, where $>$ is the *greater than* relation on natural numbers.

However, it may happen that a program P has an infinite SLD-derivation for some ground goal and, at the same time, there exists a well-founded relation, say $\succ$, on B_P such that for each ground instance of a clause in P the head is in the $\succ$ relation with each atom of the body. This means that in Bezem's characterization the $>$ ordering on natural numbers cannot be replaced by an arbitrary well-founded relation.

Consider, for instance, the program Q:

$p(a) \leftarrow q(X)$
$q(f(X)) \leftarrow q(X)$

For this program there exists the following infinite SLD-derivation:

$p(a),\ q(X1),\ q(X2),\ \ldots$

and there exists also a well-founded relation $\succ$ on B_Q such that for each ground instance $h \leftarrow b$ of a clause in Q, it is the case that $h \succ b$. Indeed, such a well-founded relation $\succ$ is defined as follows:

for all ground terms t and u in the Herbrand universe,
$p(t) \succ q(u)$ and $q(f(t)) \succ q(t)$. □

Theorem 5 *If R is a terminating rule set, then $I(R)$ is the unique fixpoint of T_R.*

Proof. By Theorem 4 there exists a well-founded relation $\succ$ such that, for all rules $(X/y) \in R$ and for all $x \in X$, we have that $y \succ x$. We now prove that T_R has a unique fixpoint. Let us assume that there are two fixpoints A and B of T_R and we prove by well-founded induction w.r.t. $\succ$ that $y \in A$ iff $y \in B$.

The inductive hypothesis is: for all $x \in D$, if $y \succ x$ then ($x \in A$ iff $x \in B$).

If $y \in A$ then there exists a rule $(X/y) \in R$ with $X \subseteq A$, because A is a fixpoint of T_R. We also have that for all $x' \in X$, $y \succ x'$ and therefore, by the inductive hypothesis, $x' \in B$. Thus, $X \subseteq B$ and, since B is a fixpoint of T_R and $(X/y) \in R$, we have that $y \in B$. Similarly, we can prove that if $y \in B$ then $y \in A$.

Thus, $I(R)$, which is the least fixpoint of T_R (see Theorem 1), is also the unique fixpoint of T_R. □

Theorem 5 is related to a result of Bezem [2], which states that if a definite logic program P is terminating then $T_P^\omega(\emptyset) = T_P^\omega(B_P)$. Indeed, from $T_P^\omega(\emptyset) = \mathit{lfp}(T_P) \subseteq \mathit{gfp}(T_P) \subseteq T_P^\omega(B_P)$ and $T_P^\omega(\emptyset) = T_P^\omega(B_P)$, we have that $\mathit{lfp}(T_P) = \mathit{gfp}(T_P)$ and thus, T_P has a unique fixpoint.

In order to prove that a rule replacement is increasing we may use the following lemma.

Lemma 1 *If $I(R) \models R \Rightarrow S$ then $I(R) \subseteq \mathit{gfp}(T_S)$.*

Proof. Take any $d \in I(R)$. Then there exists $(X/d) \in R$ (i.e., $X \in R_d$) and $X \subseteq I(R)$, because $I(R) = T_R(I(R))$. Since $I(R) \models R \Rightarrow S$, there exists $(Z/d) \in S$ (i.e., $Z \in S_d$) such that $Z \subseteq I(R)$. By the definition of T_S, from $Z \subseteq I(R)$ we have that $d \in T_S(I(R))$. Thus, $T_S(I(R)) \supseteq I(R)$, that is, $I(R)$ is a postfixpoint of T_S and, since $\mathit{gfp}(T_S)$ is the *greatest* postfixpoint of T_S (see Theorem 1), we have that $I(R) \subseteq \mathit{gfp}(T_S)$. □

We now present our sufficient condition for a rule replacement to be increasing. This condition is based on the assumption that the rule set derived by rule replacement is terminating.

Lemma 2 (Unique Fixpoint Lemma) *If $I(R) \models R \Rightarrow S$ and S is terminating, then $I(R) \subseteq I(S)$.*

Proof. By Lemma 1 and Theorem 5, we have that $I(R) \subseteq \mathit{gfp}(T_S) = \mathit{lfp}(T_S) = I(S)$. □

By combining Theorem 3 and Lemma 2 we have the following result.

Theorem 6 (Total Correctness: Unique Fixpoint Method) *If $\rho : R \mapsto S$ is a rule replacement such that $I(R) \models R \Leftrightarrow S$ and S is terminating, then $I(R) = I(S)$.*

5 The Improvement Method

The Unique Fixpoint Method (that is, Theorem 6) cannot be used to prove the total correctness of an equivalence-based rule replacement $\rho : R \mapsto S$ when the rule set S is not terminating. In this section we consider a different method, called *Improvement Method* for proving the total correctness of $\rho : R \mapsto S$, which is applicable also in the case where S is not terminating. The Improvement Method relies on the notion of *R-proofs* (that is, proofs which use the rules of R) and it ensures the total correctness of $\rho : R \mapsto S$ when the premises of S have smaller R-proofs than the corresponding premises of R.

Our Improvement Method is named after Sands' method for proving the total correctness of some transformations of higher-order functional programs [11]. Sands' Improvement Method can be derived from our Improvement Method by defining the semantics of higher-order functional programs by means of rule sets (see, for instance, the 'big step' operational semantics of higher-order functional languages in [16]).

Also the method based on the size of the smallest proof of an atom used in [13] to prove the correctness of the unfolding, folding, and goal replacement transformations of logic programs can be derived by the Improvement Method presented here.

In this section we consider *finitary* rule sets, that is, we assume that the set of premises of each rule is finite.

Definition 7 (Finitary R-Proofs) Let R be a finitary rule set over D and $d \in D$. A *finitary R-proof of d* is a finite tree π_d whose nodes are labeled by elements of D such that:

(i) the root of π_d is labeled by d,

(ii) if a leaf of π_d is labeled by y, then $(\emptyset/y) \in R$, and

(iii) if a node N of π_d is labeled by y and the sons of N are labeled by $x_1, \ldots, x_n$, then $(\{x_1, \ldots, x_n\}/y) \in R$.

Given a finite subset A of D, a *finitary R-proof of A* is the set $\pi_A = \{\pi_d \mid d \in A\}$. The *size* of π_A is the number $\sigma(\pi_A)$ of nodes in π_A.

We denote by $\Pi(R)$ the union of the set of all finitary R-proofs of elements of D and the set of all finitary R-proofs of finite subsets of D.

When understood from the context, we will simply say R-proof, instead of finitary R-proof.

Definition 8 Let R be a finitary rule set over D. For any finite set $A \subseteq D$, we say that A is *R-provable*, written $R \vdash A$, iff there exists an R-proof of A.

We have the following characterization of the inductive set defined by a finitary rule set (see, for instance, [1, 16]).

Theorem 7 *Given a finitary rule set R over D and a finite subset A of D, we have that $A \subseteq I(R)$ iff $R \vdash A$.*

Definition 9 Let R be a finitary rule set over D and $\rho : R \mapsto S$ be a rule replacement.
(i) For $\mathbf{X}, \mathbf{Z} \subseteq \wp(D)$, $\mathbf{X}$ *is* $\geq$*-improved by* $\mathbf{Z}$ *in* R, written $\Pi(R) \models \mathbf{X} \geq \mathbf{Z}$, iff for all $X \in \mathbf{X}$, if there exists an R-proof π_X of X then there exist $Z \in \mathbf{Z}$ and $\pi_Z \in \Pi(R)$ such that $\sigma(\pi_X) \geq \sigma(\pi_Z)$.
(ii) We write $\Pi(R) \models R \geq S$ iff for all $d \in D$, $\Pi(R) \models R_d \geq S_d$.

From Theorem 7 it follows that $\Pi(R) \models \mathbf{X} \geq \mathbf{Z}$ implies $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ and, thus, $\Pi(R) \models R \geq S$ implies $I(R) \models R \Rightarrow S$.

Lemma 3 (Improvement) *Let R be a finitary rule set over D and $\rho : R \mapsto S$ be a rule replacement such that $\Pi(R) \models R \geq S$. Then $I(R) \subseteq I(S)$.*

Proof. By Theorem 7 it is enough to prove that for all $\pi_d \in \Pi(R)$ with $d \in D$, we have that $d \in I(S)$. We will prove this fact by well-founded induction w.r.t. $>$.

Take $\pi_d \in \Pi(R)$ with $d \in D$. Assume that for all R-proofs π_y with $y \in D$, if $\sigma(\pi_d) > \sigma(\pi_y)$ then $y \in I(S)$.

If the root of π_d has $m \geq 0$ sons, labeled by $x_1, \ldots, x_m$, then by the definition of R-proof (see Definition 7), $\{x_1, \ldots, x_m\} \in R_d$ and there exists an R-proof $\pi_{\{x_1,\ldots,x_m\}}$ of $\{x_1, \ldots, x_m\}$ such that $\sigma(\pi_d) > \sigma(\pi_{\{x_1,\ldots,x_m\}})$. Now, since $\Pi(R) \models R \geq S$, there exist $\{z_1, \ldots, z_n\} \in S_d$ and an R-proof $\pi_{\{z_1,\ldots,z_n\}}$ of $\{z_1, \ldots, z_n\}$ such that $\sigma(\pi_{\{x_1,\ldots,x_m\}}) \geq \sigma(\pi_{\{z_1,\ldots,z_n\}})$.

Thus, for $i = 1, \ldots, n$, we have that $\sigma(\pi_d) > \sigma(\pi_{\{x_1,\ldots,x_m\}}) \geq \sigma(\pi_{\{z_1,\ldots,z_n\}}) \geq \sigma(\pi_{z_i})$. Then, by the induction hypothesis, we have that for $i = 1, \ldots, n$, $z_i \in I(S)$ and, since $\{z_1, \ldots, z_n\} \in S_d$ and $I(S)$ is S-closed, we have that $d \in I(S)$. □

By combining Theorem 3 and Lemma 3 we have the following result.

Theorem 8 (Total Correctness: Improvement Method) *Let R be a finitary rule set. If $\rho : R \mapsto S$ is a rule replacement such that $I(R) \models R \Leftarrow S$ and $\Pi(R) \models R \geq S$, then $I(R) = I(S)$.*

6 The Well-Founded Improvement Method

In this section we generalize the Improvement Method in two ways: (i) we consider rule sets that are possibly not finitary, that is, rule sets with a possibly infinite number of premises, such that their associated immediate consequence operators are *continuous* (notice that for any *finitary* rule set R, the operator T_R is continuous), and (ii) we replace the concept of *size* of a proof by that of a function into an arbitrary well-founded set. The resulting method will be called *Well-Founded Improvement Method.*

Definition 10 (Finite Depth R-Proofs) Let R be a (possibly not finitary) rule set over D and $d \in D$. A *finite depth R-proof of d* is a tree φ_d whose nodes are labeled by elements of D such that:

(i) the root of φ_d is labeled by d,

(ii) if a leaf of φ_d is labeled by y, then $(\emptyset/y) \in R$,

(iii) if a node N of φ_d is labeled by y and $X = \{x \mid x \text{ is the label of a son of } N\}$, then $(X/y) \in R$, and

(iv) each branch of φ_d has finite length.

Given a (finite or infinite) subset A of D, a *finite depth R-proof of A* is the set $\varphi_A = \{\varphi_d \mid d \in A\}$.

Notice that a finite depth R-proof is a tree with possibly infinite branching. When understood from the context we will simply say R-proof, instead of finite depth R-proof.

As in the case of finitary rule sets, for a finite or infinite A, we write $R \vdash A$ iff there exists a finite depth R-proof of A.

We denote by $\Phi(R)$ the union of the set of all finite depth R-proofs of elements of D and the set of all finite depth R-proofs of subsets of D.

Obviously, we have that for every A, if $R \vdash A$ then $A \subseteq I(R)$. The reader may verify that the converse does not hold for rule sets which are not finitary. However, we have the following result.

A rule set R whose immediate consequence operator T_R is continuous, is said to be a *continuous rule set.*

Theorem 9 *Let R be a continuous rule set over D. For every (finite or infinite) $A \subseteq D$, we have that: $A \subseteq I(R)$ iff $R \vdash A$.*

Let $\succ$ be a transitive well-founded relation on a set W and let $\succeq$ be the partial order $(\succ \cup =)$. Let R be a rule set over D and $\mu : \Phi(R) \rightarrow W$ be a function such that the following two properties hold:

- (*Father-Son Monotonicity*) for all $\varphi_d \in \Phi(R)$, with $d \in D$, such that X is the set of elements of D labeling the sons of the root of φ_d, there exists $\varphi_X \in \Phi(R)$ such that $\mu(\varphi_d) \succ \mu(\varphi_X)$, and
- (*Membership Monotonicity*) for all $\varphi_Z \in \Phi(R)$, with $Z \subseteq D$, and for all $z \in Z$, there exists φ_z such that $\mu(\varphi_Z) \succeq \mu(\varphi_z)$.

Definition 11 Let R be a rule set over D and $\rho : R \mapsto S$ be a rule replacement.
(i) For $\mathbf{X}, \mathbf{Z} \subseteq \wp(D)$, $\mathbf{X}$ *is $\succeq$-improved by* $\mathbf{Z}$ *in* R, written $\Phi(R) \models \mathbf{X} \succeq \mathbf{Z}$, iff for all $X \in \mathbf{X}$, if there exists an R-proof φ_X of X then there exist $Z \in \mathbf{Z}$ and $\varphi_Z \in \Phi(R)$ such that $\mu(\varphi_X) \succeq \mu(\varphi_Z)$.
(ii) We write $\Phi(R) \models R \succeq S$ iff for all $d \in D$, $\Phi(R) \models R_d \succeq S_d$.

Similarly to Section 5, for a *continuous* rule set R, we have that $\Phi(R) \models \mathbf{X} \succeq \mathbf{Z}$ implies $I(R) \models \mathbf{X} \rightarrow \mathbf{Z}$ and, thus, $\Phi(R) \models R \succeq S$ implies $I(R) \models R \Rightarrow S$.

Lemma 4 (Well-Founded Improvement) *Let R be a continuous rule set over D and $\rho : R \mapsto S$ be a rule replacement such that $\Phi(R) \models R \succeq S$. Then $I(R) \subseteq I(S)$.*

Proof. Similarly to the proof of Theorem 3, we show, by well-founded induction w.r.t. $\succ$, that for all R-proofs φ_d with $d \in D$, we have that $d \in I(S)$. This property implies the thesis because R is continuous and, by Theorem 9, d has an R-proof iff $d \in I(R)$.

Take $\varphi_d \in \Phi(R)$ with $d \in D$. Suppose that for all R-proofs φ_y with $y \in D$, if $\mu(\varphi_d) \succ \mu(\varphi_y)$ then $y \in I(S)$.

Let X be the set of elements of D that label the sons of the root of φ_d. Then by the definition of R-proof (see Definition 10), $X \in R_d$ and by the father-son monotonicity property there exists an R-proof φ_X of X such that $\mu(\varphi_d) \succ \mu(\varphi_X)$. Since $\Phi(R) \models R \succeq S$, there exist $Z \in S_d$ and an R-proof φ_Z of Z such that $\mu(\varphi_X) \succeq \mu(\varphi_Z)$.

By using the membership monotonicity property of μ, for all $z \in Z$, there exists φ_z such that $\mu(\varphi_Z) \succeq \mu(\varphi_z)$. Thus, for all $z \in Z$, we have that: $\mu(\varphi_d) \succ \mu(\varphi_X) \succeq \mu(\varphi_Z) \succeq \mu(\varphi_z)$. Then, by the induction hypothesis, for all $z \in Z$, we have that $z \in I(S)$ and thus, $Z \subseteq I(S)$. Since $Z \in S_d$ (i.e., $(Z/d) \in S$) and $I(S)$ is S-closed, we conclude that $d \in I(S)$. □

Thus, by combining Theorem 3 and Lemma 4 we have the following result.

Theorem 10 (Total Correctness: Well-Founded Improvement Method) *Let R be a continuous rule set and $\rho : R \mapsto S$ be a rule replacement. If $I(R) \models R \Leftarrow S$ and $\Phi(R) \models R \succeq S$, then $I(R) = I(S)$.*

7 Related Work and Conclusions

We have presented a theory of *transformations of inductive definitions* which provides a common foundation for the theories of *transformations of programs* developed within various programming languages.

The methods for proving the correctness of the transformations of rule sets we have presented here, are generalizations of methods used in specific frameworks, mainly in the field of functional and logic programming. Our Unique Fixpoint Method is related to McCarthy's recursion induction principle for proving the equivalence of two continuous functions on flat cpo's [8]. The use of McCarthy's principle was also suggested as a technique for proving the correctness of the folding and unfolding transformations in the seminal paper by Burstall and Darlington [4]. A method for proving the equivalence of recursive applicative program schemata based on the existence of a unique fixpoint of a set of equations, was also studied by Courcelle [5].

The Improvement Method and the Well-Founded Improvement Method are generalizations of several methods used in functional and logic programming for proving that program transformations, such as folding, unfolding, and replacement, are correct. In particular, Sands [11] has proposed the so called *Local Improvement* Method, which consists in replacing an expression e_1 in the definition of a higher-order function by a new expression e_2 which is semantically equivalent to e_1 and, for each closing context $C[\_]$, $C[e_2]$ evaluates to weak head normal form in no more reduction steps than $C[e_1]$. This method can be derived from ours because: (i) the semantics of higher-order functional programs can be described in terms of inductive definitions (over the set of closed expressions), (ii) the replacement of semantically equivalent expressions can be viewed as a particular case of equivalence-based rule replacement, and (iii) the evaluation of an expression to weak head normal form can be viewed as a proof whose size corresponds to the number of reduction steps.

Various other methods which have been used for proving the correctness of the folding, unfolding, and goal replacement transformations of definite logic programs, can be derived by our Well-Founded Improvement Method. Among these methods we would like to mention the ones used in: (i) the landmark paper by Tamaki and Sato [13], where the authors prove a correctness result for the above mentioned transformations w.r.t. the least Herbrand model, (ii) the generalization of [13] that the same authors presented in [14], where a lexicographic ordering between atoms is adopted for proving the correctness of a more powerful folding transformation, and (iii) the papers [3, 6, 12] which extend the proofs of correctness of the folding and unfolding transformations w.r.t. universal termination, computed answer substitutions, and finite failure semantics.

For lack of space, we have not presented these special cases in detail. However, in all these cases one may: (i) describe the appropriate semantics for definite logic programs as an inductive set defined by a set of rules over a suitable domain, (ii) show that the folding, unfolding, and goal replacement transformations induce rule replacements of this set of rules, and (iii) derive the specific measure on proofs to be used for proving the correctness of the program transformations as an instance of our μ measure enjoying the father-son and membership monotonicity properties.

Finally, as work related to ours, we would like to mention a recent paper by Roychoudhury et al. [10], where the authors present a parametrized framework for proving correctness conditions w.r.t. the least Herbrand model for various extensions of the folding rule originally proposed in [13]. This framework is parametrized w.r.t. an abstract measure of atoms (not *proofs*, as we do). The properties of this abstract measure are more specific than the ones required here for the function μ used in our Well-Founded Improvement Method. In particular, the framework of [10] requires the existence of a 4-tuple $\langle \mathcal{M}, \oplus, \preceq, \mathcal{W} \rangle$ such that: (i) $\langle \mathcal{M}, \oplus \rangle$ is a *commutative group* with identity $0 \in \mathcal{M}$, (ii) $\preceq$ is a *linear order*, (iii) $\oplus$ is a *monotone* operation w.r.t. $\preceq$, and $\mathcal{W}$ is a subset of $\{x \in \mathcal{M} \mid \ 0 \preceq x\}$, over which $\preceq$ is *well-founded*. In contrast, we only require the existence of a function μ from the set of all proofs into a set with a well-founded, transitive relation, such that the *father-son monotonicity* and *membership monotonicity* properties hold (see Section 6).

Besides being parametric w.r.t. the well-founded relation used for the Well-Founded Improvement Method, our approach is also parametric w.r.t. other relevant aspects. Indeed, it is parametric w.r.t.: (1) *the programming language*, because it provides theoretical tools for proving the correctness of program transformations in all languages whose semantics can be defined by means of sets of inductive rules on arbitrary domains, (2) *the inductive rules* used for defining the semantics of the programming language of interest, and indeed, these rules are allowed to be either finitary or infinitary (as long as they are continuous), and finally, (3) *the program transformations*, because in our approach we can deal with a very general, in fact, complete, set of transformations.

Acknowledgements

We would like to thank C.R. Ramakrishnan for stimulating conversations on the issue of proving the correctness of logic program transformations. This work has been partially supported by MURST Progetto Cofinanziato 'Tecniche Formali per la Specifica, l'Analisi, la Verifica, la Sintesi e la Trasformazione di Sistemi Software' (Italy), Progetto Coordinato CNR 'Programmazione Logica' (Italy), and Progetto Coordinato CNR 'Verifica, Analisi e Trasformazione dei Programmi Logici' (Italy).

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, chapter 7, pages 739–782. North-Holland, 1977.

[2] M. Bezem. Characterizing termination of logic programs with level mappings. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of the North American*

Conference on Logic Programming, Cleveland, Ohio (USA), pages 69–80. MIT Press, 1989.

[3] A. Bossi and N. Cocco. Preserving universal termination through unfold/fold. In *Proceedings ALP '94*, Lecture Notes in Computer Science 850, pages 269–286. Springer-Verlag, 1994.

[4] R. M. Burstall and J. Darlington. A transformation system for developing recursive programs. *Journal of the ACM*, 24(1):44–67, January 1977.

[5] B. Courcelle. Recursive applicative program schemes. In J. van Leuveen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 459–492. Elsevier, 1990.

[6] T. Kawamura and T. Kanamori. Preservation of stronger equivalence in unfold/fold logic program transformation. *Theoretical Computer Science*, 75:139–156, 1990.

[7] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987. Second Edition.

[8] J. McCarthy. Towards a mathematical science of computation. In C.M. Popplewell, editor, *Information Processing : Proceedings of IFIP 1962*, pages 21–28, Amsterdam, 1963. North Holland.

[9] A. Pettorossi and M. Proietti. Transformation of logic programs: Foundations and techniques. *Journal of Logic Programming*, 19,20:261–320, 1994.

[10] A. Roychoudhury, K.N. Kumar, C.R. Ramakrishnan, and I.V. Ramakrishnan. A generalized unfold/fold transformation system for definite logic programs. Technical report, Department of Computer Science, SUNY at Stony Brook, USA, 1998.

[11] D. Sands. Total correctness by local improvement in the transformation of functional programs. *ACM Toplas*, 18(2):175–234, 1996.

[12] H. Seki. Unfold/fold transformation of stratified programs. *Theoretical Computer Science*, 86:107–139, 1991.

[13] H. Tamaki and T. Sato. Unfold/fold transformation of logic programs. In S.-Å. Tärnlund, editor, *Proceedings of the Second International Conference on Logic Programming, Uppsala, Sweden*, pages 127–138. Uppsala University, 1984.

[14] H. Tamaki and T. Sato. A generalized correctness proof of the unfold/fold logic program transformation. Technical Report 86-4, Ibaraki University, Japan, 1986.

[15] M. H. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[16] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. The MIT Press, 1993.