A Parameterized Unfold/Fold Transformation Framework for Definite Logic Programs*

Abhik Roychoudhury[1], K. Narayan Kumar[1,2], C.R. Ramakrishnan[1], and I.V. Ramakrishnan[1]

[1] Dept. of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794, USA.
{abhik,kumar,cram,ram}@cs.sunysb.edu
[2] Chennai Mathematical Institute, 92 G.N. Chetty Road, Chennai, India.
kumar@smi.ernet.in

Abstract. Given a program P, an unfold/fold program transformation system derives a sequence of programs $P = P_0, P_1, \ldots, P_n$, such that P_{i+1} is derived from P_i by application of either an unfolding or a folding step. Existing unfold/fold transformation systems for definite logic programs differ from one another mainly in the kind of folding transformations they permit at each step. Some allow folding using a single (possibly recursive) clause while others permit folding using multiple non-recursive clauses. However, none allow folding using *multiple recursive* clauses that are drawn from some previous program in the transformation sequence. In this paper we develop a *parameterized* framework for unfold/fold transformations by suitably abstracting and extending the proofs of existing transformation systems. Various existing unfold/fold transformation systems can be obtained by instantiating the parameters of the framework. This framework enables us to not only understand the relative strengths and limitations of these systems but also construct new transformation systems. Specifically we present a more *general* transformation system that permits folding using multiple recursive clauses that can be drawn from any previous program in the transformation sequence. This new transformation system is also obtained by instantiating our parameterized framework.

1 Introduction

Some of the most extensively studied transformation systems for definite logic programs are the so called *unfold/fold* transformation systems. At a high level unfold and fold transformations can be viewed as follows. Definite logic programs consist of definitions of the form $A{:-}\ \phi$ where A is an atom and ϕ is a positive boolean formula over atoms. Unfolding replaces an occurrence of A in a program with ϕ while folding replaces an occurrence of ϕ with A. Folding is called *reversible* if its effects can be undone by an unfolding, and *irreversible*

* The work of Abhik Roychoudhury, C.R. Ramakrishnan and I.V. Ramakrishnan was partially supported by NSF grants CCR-9711386 and EIA-9705998. The work of K. Narayan Kumar was partially supported by NSF grant CDA-9805735.

otherwise. An unfold/fold transformation system for definite logic programs was first described in a seminal paper by Tamaki and Sato [20]. In the flurry of research activity that followed, a number of unfold/fold transformation systems were developed. Kanamori and Fujita [8] proposed a transformation system that was based on maintaining counters to guide folding. Maher described a system that permits only reversible folding [10]. The basic Tamaki-Sato system itself was extended in several directions (e.g., to handle folding with multiple clauses [7], negation [1, 18, 19]) and applied to practical problems (e.g., [2, 3, 12]). (See [11] for an excellent survey of research on this topic over the past decade).

Correctness of Unfold/Fold Transformations Correctness proofs for unfold/fold transformations consider *transformation sequences* of the form $P_0, P_1, \ldots$, where P_0 is an initial program and P_{i+1} is obtained from P_i by applying an unfolding or folding transformation. The proofs usually show that all programs in the transformation sequence have the same least Herbrand model. It is easy to verify that transforming P_i to P_{i+1} using unfolding or folding is *partially correct*, i.e., the least model of P_{i+1} is a subset of that of P_i. It is also easy to show, by induction on the structure of the proof trees, that unfolding transformation is *totally* correct, i.e., it *preserves* the least model. However, as illustrated below, indiscriminate folding may introduce circularity in definitions, thereby replacing finite proof paths with infinite ones.

Consider the sequence of programs in Figure 1. In the figure, P_1 is derived by unfolding the occurrence of `q(X)` in the first clause of P_0. P_2 is derived from P_1 by folding the literal `q(X)` in the body of the second clause of predicate `p` into `p(X)` using the clause `p(X) :- q(X)` in P_0. Alternatively, consider the transformation sequence in figure 2. By folding `q(X)` in the second clause of `p` in P_1 (using the second clause defining `q` in P_1), we obtain program P_2'. Now folding `q(X)` in the second clause of `q` in P_2' (using second clause of `p` in P_1), we get program P_3', whose least model differs from that of P_0.

Transformation Systems with Irreversible Folding If the folding transformation is reversible, then since its effect can be undone by an unfolding, any partially correct unfold/fold transformation sequence is also totally correct. However, for reversibility, folding at step i of the transformation can *only* use the clauses in P_i. Therefore reversibility is a *restrictive* condition that seriously limits the

```
p(X):-q(X).
q(a).
q(f(X)):-q(X).
```

Program P_0

```
p(a).
p(f(X)):-q(X).
q(a).
q(f(X)):-q(X).
```

Program P_1

```
p(a).
p(f(X)):-p(X).
q(a).
q(f(X)):-q(X).
```

Program P_2

Fig. 1. An example of correct unfold/fold transformation sequence

power of unfold/fold systems by disallowing many correct folding transformations, such as the one used to derive P_2 from P_1. Hence almost all research on unfold/fold transformations have focused on constructing systems that permit irreversible folding. In such systems folding at step i can use clauses that are not in P_i. For example, in the original and extended Tamaki-Sato systems [20, 21] folding always uses clauses in P_0 whereas in the Kanamori-Fujita system [8] the clauses can come from any P_j ($j \leq i$). But ensuring total correctness of irreversible transformation sequences is difficult. In order to ensure that folding is still totally correct, these systems permit folding using only clauses with certain (syntactic) properties. For instance, the original Tamaki-Sato system permits folding using a single clause only (*conjunctive* folding) and this clause is required to be non-recursive. In [7] the above system was extended to allow folding with multiple clauses (*disjunctive* folding) but all the clauses are required to be be non-recursive. Kanamori and Fujita [8] as well Tamaki and Sato in a later paper [21] gave two different approaches for conjunctive folding using recursive clauses. But the design of a transformation system that allows folding in the presence of both disjunction *and* recursion has remained open so far. We will describe such a system in this paper.

To generalize in this direction one needs to first understand the strengths and limitations of the above systems. The key observation is that, although the book-keeping needed to determine permissible foldings appear radically different in the different systems, there is a striking similarity in how the transformations are proved correct. Essentially, these systems associate some *measure* with different program elements, namely, atoms and clauses to determine whether folding is permissible in that step (e.g., "foldable" flag in [20], descent levels/strata numbers in [21], and counters in [8]). Moreover, they ensure that each transformation step maintains an invariant relating proofs in the derived program to the various measures (e.g., the notions of rank-consistency in [8, 20], weight-consistency in [7] and μ-completeness in [21]). This raises another interesting question: can we exploit the similarities in the correctness proofs of irreversible unfold/fold systems to develop an abstract framework. Such a framework will specify the obligations that must be satisfied to ensure total correctness and hence can simplify construction of unfold/fold systems to the extent that one is relieved of the burden of giving correctness proofs. We propose such a framework in this paper.

`p(X):-q(X).`
`q(a).`
`q(f(X)):-q(X).`
	`p(a).`
	`p(f(X)):-q(X).`
	`q(a).`
	`q(f(X)):-q(X).`
		`p(a).`
		`p(f(X)):-q(f(X)).`
		`q(a).`
		`q(f(X)):-q(X).`
			`p(a).`
			`p(f(X)):-q(f(X)).`
			`q(a).`
			`q(f(X)):-p(f(X)).`

Program P_0	Program P_1	Program P_2'	Program P_3'

Fig. 2. An example of incorrect unfold/fold transformation sequence

Summary of Results In this paper, we develop a general transformation framework for definite logic programs parameterized by certain abstract measures by suitably abstracting and extending the measures used in [7, 8, 20, 21] (see Section 2). We relax the invariants needed in the proofs to permit *approximation* of measure values. This is the key idea that enables us to fold using multiple recursive clauses. We prove the correctness of transformations in the framework based only on the properties of the abstract measures. We show that various existing unfold/fold transformation systems can be derived from the framework by instantiating these abstract measures (see Section 3). We also show how the framework can be extended to include the Goal Replacement transformation (see Section 4).

The parameterized framework presented in this paper is useful for understanding the strengths and limitations of existing transformation systems. It also enables the construction of new unfold/fold systems. As evidence we obtain SCOUT (Strata and COunter based Unfold/fold Transformations), a transformation system that permits disjunctive folding using recursive clauses. The development of SCOUT was based on two crucial observations made possible by the framework. First, when instantiating the framework to obtain the Kanamori-Fujita system, it is easy to see that the counters (the measure used in their system) may come from any linearly ordered set; this permits us to incorporate stratification into the counters to obtain a system that generalizes the extended Tamaki-Sato system [21] as well as the Kanamori-Fujita system. Secondly, the framework enables us to maintain approximate counters; we can hence generalize the combination of the Kanamori-Fujita and the extended Tamaki-Sato systems to fold using multiple recursive clauses.

2 A Parameterized Transformation Framework

We now describe our parameterized unfold/fold transformation framework and illustrate the abstractions by drawing analogies to the Kanamori-Fujita system.

We assume familiarity with the standard notions of terms, models, substitutions, unification, most general unifier (mgu), definite clauses, SLD resolution, and proof trees [9]. We will use the following symbols (possibly with primes and subscripts): P to denote a definite logic program; $M(P)$ its least Herbrand model; C and D for clauses; A, B to denote atoms and literals and σ for mgu.

2.1 Unfolding and Folding

The unfolding and folding rules are defined as follows:

Rule 1 (Unfolding) Let C be a clause in P_i and A an atom in the body of C. Let $C_1, \ldots, C_m$ be the clauses in P_i whose heads are unifiable with A with most general unifier $\sigma_1, \ldots, \sigma_m$. Let C'_j be the clause that is obtained by replacing $A\sigma_j$ by the body of $C_j\sigma_j$ in $C\sigma_j$ ($1 \leq j \leq m$). Assign $(P_i - \{C\}) \cup \{C'_1, \ldots, C'_m\}$ to P_{i+1}. $\square$

Rule 2 (Folding) Let $\{C_1, \ldots, C_m\} \subseteq P_i$ where C_l denotes the clause $A{:-}\ A_{l,1}, \ldots, A_{l,n_l}, A'_1, \ldots, A'_n$, and $\{D_1, \ldots, D_m\} \subseteq P_j$ $(j \leq i)$ where D_l is the clause $B_l{:-}\ B_{l,1}, \ldots, B_{l,n_l}$. Further, let:
1. $\forall 1 \leq l \leq m\ \exists \sigma_l\ \forall 1 \leq k \leq n_l\ A_{l,k} = B_{l,k}\sigma_l$
2. $B_1\sigma_1 = B_2\sigma_2 = \cdots = B_m\sigma_m = B$
3. $D_1, \ldots, D_m$ are the only clauses in P_j whose heads are unifiable with B.
4. $\forall 1 \leq l \leq m$, σ_l substitutes the internal variables[1] of D_l to distinct variables which do not appear in $\{A, B, A'_1, \ldots A'_n\}$.

Then $P_{i+1} := (P_i - \{C_1, \ldots, C_m\}) \cup \{C'\}$ where $C' \equiv A{:-}\ B, A'_1, \ldots, A'_n$. □

$D_1, \ldots, D_m$ are the *folder* clauses, $C_1, \ldots, C_m$ are the *folded* clauses, and B is the *folder* atom. A folding step is *conjunctive* whenever both the folder and folded clauses are singleton sets and is *disjunctive* otherwise. Note that in the latter step a set of folded clauses is *simultaneously* replaced by a single clause using a set of folder clauses. We say that $P_0, P_1, \ldots, P_n$ is an unfold/fold transformation sequence if the program P_{i+1} is obtained from P_i $(i \geq 0)$ by application of an unfold or a fold rule. Partial correctness of an unfold/fold transformation sequence (Theorem 1) is established by showing that a proof T of any ground atom $A \in M(P_{i+1})$, has a corresponding proof T' in P_i. This can be proved by induction on the structure of T.

Theorem 1 (Partial Correctness) *Let $P_0, P_1, \ldots, P_i$ be a program transformation sequence where $M(P_j) = M(P_0)$ for all $0 \leq j \leq i$. If P_{i+1} is obtained from P_i by applying either unfolding or folding, then $M(P_{i+1}) \subseteq M(P_i)$.* □

2.2 Measures, Measure-Consistent Proofs and Total Correctness

Total correctness of an unfold/fold transformation sequence is established by inducting on some well-founded order to construct a proof in P_{i+1} for any atom A in $M(P_i)$. To see the subtleties in showing total correctness, consider transforming P_i to P_{i+1} using a conjunctive folding step. To construct a proof of A (the head of the folded clause) in P_{i+1}, we need a proof of B (the folder atom) in P_{i+1}. But the existence of such a proof can be established (by induction hypothesis) only if B is less than A in the well-founded order on which we are inducting. Note that if the folder clause is picked from P_j, $j < i$, we cannot use simple well-founded orders like size of proof trees in P_i, since proof of B in P_i can be larger in size than the proof of A in P_i. Here we develop an abstract formulation of certain well-founded orders (which we call *measures*) on which we can induct to establish total correctness.

It is worth noting that we do not attempt to translate every proof of A in P_i to a proof of A in P_{i+1}. Instead, following [8, 20, 21] we consider a "special proof" called *strongly measure-consistent proof* (see Definition 6) of A in P_i and construct a proof of A in P_{i+1}. The induction proof for establishing total correctness is completed by showing that the proof of A in P_{i+1} thus constructed is itself strongly measure consistent.

[1] Variables appearing in the body of a clause, but not its head

Recall that irreversible folding steps need to be constrained in order to preserve the semantics. In order to enforce these constraints, we maintain some book-keeping information as we perform the transformations, formalized using the following notions of *Measure structure*, *Atom measure*, and *Clause measure*.

Definition 1 (Measure Structure) *A Measure Structure is a 4-tuple* $\mu = \langle \mathcal{M}, \oplus, \prec, \mathcal{W} \rangle$ *where* $\langle \mathcal{M}, \oplus \rangle$ *is a commutative group with* $\mathbf{0} \in \mathcal{M}$ *as its identity element,* $\prec$ *is a linear order on* $\mathcal{M}$*,* $\oplus$ *is monotone w.r.t.* $\prec$*, and* $\mathcal{W}$ *is a subset of* $\{x \in \mathcal{M} \mid \mathbf{0} \preceq x\}$*, over which* $\prec$ *is well-founded.*

We will refer to $\mathcal{M}$, the first component of the measure structure, as the *measure space*. We let $\preceq$ denote $\prec$ or $=$. Moreover, we use $\ominus$ to denote the inverse operation of the group $\langle \mathcal{M}, \oplus \rangle$. We also use $\ominus$ as a binary operator, $a \ominus b$ meaning $a \oplus (\ominus b)$ (where $(\ominus b)$ is the inverse of b). The Kanamori-Fujita system [8] keeps track of integer counters. Thus the measure structure is $\langle \mathbb{Z}, +, <, \mathbb{N} \rangle$, where $\mathbb{Z}$ and $\mathbb{N}$ are the set of integers and natural numbers respectively, $+$ denotes integer addition, and $<$ is the arithmetic comparison operator.

Definition 2 (Atom Measure) *An atom measure* α *of a program* P *w.r.t. a measure structure* μ *is a partial function from the Herbrand base of* P *to* $\mathcal{W}$ *such that it is total on the least Herbrand model of* P*. For our purposes, it suffices to use the same atom measure for each program in a transformation sequence.*

In the Kanamori-Fujita system, the atom measure of any P_i in the transformation sequence is the number of nodes in the shortest proof tree of A in the initial program P_0. The proof of total correctness for folding will induct on the atom measure, relating the atom measure of A (the head of the folded clauses) with the atom measure of B (the folder atom).

Definition 3 (Clause Measure) *A clause measure* $(\gamma_{lo}, \gamma_{hi})$ *of a program* P *w.r.t. a measure structure* μ *is a pair of total functions from clauses of* P *to* $\mathcal{M}$ *such that* $\forall C \in P\ \gamma_{lo}(C) \preceq \gamma_{hi}(C)$*.*

In the Kanamori-Fujita system, γ_{lo} and γ_{hi} are the same and map each clause to its corresponding counter value. However, as we will see later, to allow disjunctive folding we will need the two distinct functions γ_{lo} and γ_{hi}. Henceforth, we denote the clause measure of a program P_i by $(\gamma^i_{lo}, \gamma^i_{hi})$. We will now develop the idea of "special proofs" mentioned earlier. For that purpose, we need the definition:

Definition 4 (Ground Proof of an Atom) *Let* T *be a tree, each of whose nodes is labeled with a ground atom. Then* T *is a ground proof in program* P*, if every node* A *in* T *satisfies the condition :* A:− $A_1, ..., A_n$ *is a ground instance of a clause in* P*, where* $A_1, ..., A_n$ $(n \geq 0)$ *are the children of* A *in* T*.*

Consider transforming P_i to P_{i+1} by a folding step (see figure below). C and D are the folded and folder clauses respectively and $j < i$.

.....	.....	.....
D : `q:−` $\mathtt{q_1, ..., q_k}$	C : `p:−` $\mathtt{q_1, ..., q_k, q_{k+1}, ..., q_n}$	C' : `p:−` $\mathtt{q, q_{k+1}, ..., q_n}$
.....	.....	.....
Program P_j	Program P_i	Program P_{i+1}

In order to show that $\mathtt{p} \in M(P_i) \Rightarrow \mathtt{p} \in M(P_{i+1})$ by induction on $\prec$, we would like to show that $\alpha(\mathtt{q}) \prec \alpha(\mathtt{p})$. The atoms $\mathtt{p}$ and $\mathtt{q}$ are related by what is shared between the bodies of the clauses C and D. Hence we attempt to relate their measures via the measures of bodies of C and D. Suppose D satisfies : (i) $\alpha(\mathtt{q}) \preceq \sum_{1 \leq i \leq k} \alpha(\mathtt{q_i})$, then we can relate $\alpha(\mathtt{q})$ to the sum of the measures of the body atoms of the folded clause C (since $k \leq n$). Further if C satisfies : (ii) $\alpha(\mathtt{p}) \succeq \sum_{1 \leq i \leq n} \alpha(\mathtt{q_i})$, then we can establish that $\alpha(\mathtt{q}) \preceq \alpha(\mathtt{p})$. If either (i) or (ii) is a strict relationship then we can establish that $\alpha(\mathtt{q}) \prec \alpha(\mathtt{p})$. Relations (i) and (ii) form the basis for the notions of *weak* and *strong measure consistency*.

Definition 5 (Weakly Measure Consistent Proof) *A ground proof T in program P_i is weakly measure consistent w.r.t. atom measure α and clause measure $(\gamma_{lo}^i, \gamma_{hi}^i)$ if every ground instance A:– $A_1, ..., A_n$ of a clause $C \in P_i$ used in T satisfies $\alpha(A) \preceq \gamma_{hi}^i(C) \oplus \sum_{1 \leq l \leq n} \alpha(A_l)$.*

Definition 6 (Strongly Measure Consistent Proof) *A ground proof T in program P_i is strongly measure consistent w.r.t. atom measure α and clause measure $(\gamma_{lo}^i, \gamma_{hi}^i)$ if every ground instance A:– $A_1, ..., A_n$ of a clause $C \in P_i$ used in T satisfies $\forall 1 \leq l \leq n\ \alpha(A_l) \prec \alpha(A)$ and $\alpha(A) \succeq \gamma_{lo}^i(C) \oplus \sum_{1 \leq l \leq n} \alpha(A_l)$*

Definition 7 (Measure Consistent Proof) *A ground proof T in program P_i is said to be measure consistent w.r.t. atom measure α and clause measure $(\gamma_{lo}^i, \gamma_{hi}^i)$, if it is strongly and weakly measure consistent w.r.t. α and $(\gamma_{lo}^i, \gamma_{hi}^i)$.*

We point out that our abstract notion of measure consistency relaxes the concrete notion of rank consistency of [8]. While rank consistency of [8] imposes a strict equality constraint on $\alpha(A)$, measure consistency only *bounds it from above and below.* As we will show later, this facilitates maintenance of approximate information. This is the central idea that permits us to do disjunctive folding using recursive clauses. For proving total correctness, we need :

Definition 8 (Measure consistent Program) *A program P is measure consistent w.r.t. atom measure α and clause measure $(\gamma_{lo}, \gamma_{hi})$, if for all $A \in M(P)$, we have : (1) All ground proofs of A in P are weakly measure consistent w.r.t. α and $(\gamma_{lo}, \gamma_{hi})$ (2) A has a ground proof in P which is strongly measure consistent w.r.t. α and $(\gamma_{lo}, \gamma_{hi})$*

We are now ready to define the abstract conditions on folding and constraints on how the clause measures are to be updated after an unfold/fold step. For each clause C obtained by applying an unfold/fold transformation on program P_i, we derive a lower bound on $\gamma_{hi}^{i+1}(C)$ and an upper bound on $\gamma_{lo}^{i+1}(C)$, denoted by $GLB^{i+1}(C)$ and $LUB^{i+1}(C)$ respectively. We will see later that the conditions on when the rules become applicable, as well as these bounds will be based on the requirements of the proof of total correctness.

We assume that for any atom A (not necessarily ground), $\alpha_{min}(A)$ denotes a lower bound on the measure of any provable ground instantiation of A *i.e.* $\forall\theta\ \alpha_{min}(A) \preceq \alpha(A\theta)$. We use α_{min} in the folding condition of rule 4 below.

Rule 3 (Measure Preserving Unfolding) Let P_{i+1} be obtained from P_i by an unfolding transformation as described in Rule 1. Then, $\forall 1 \le j \le m$

$$\gamma_{lo}^{i+1}(C'_j) \preceq GLB^{i+1}(C'_j) = \gamma_{lo}^{i}(C) \oplus \gamma_{lo}^{i}(C_j) \quad (1)$$
$$\gamma_{hi}^{i+1}(C'_j) \succeq LUB^{i+1}(C'_j) = \gamma_{hi}^{i}(C) \oplus \gamma_{hi}^{i}(C_j) \quad (2)$$

The clause measure of all other clauses in P_{i+1} are inherited from P_i. □

Rule 4 (Measure Preserving Folding) Let P_{i+1} be obtained from P_i by a folding transformation as described in Rule 2, such that $\forall 1 \le l \le m.\ \gamma_{hi}^{j}(D_l) \prec \gamma_{lo}^{i}(C_l) \oplus \sum_{1 \le k \le n} \alpha_{min}(A'_k)$.[2] Then,

$$\gamma_{lo}^{i+1}(C') \preceq GLB^{i+1}(C') = \min_{1 \le l \le m} (\ \gamma_{lo}^{i}(C_l) \ominus \gamma_{hi}^{j}(D_l)\ ) \quad (3)$$
$$\gamma_{hi}^{i+1}(C') \succeq LUB^{i+1}(C') = \max_{1 \le l \le m} (\gamma_{hi}^{i}(C_l) \ominus \gamma_{lo}^{j}(D_l)) \quad (4)$$

and the clause measure of all other clauses in P_{i+1} are inherited from P_i. □

It should be noted that the above rules do not prescribe *unique* values for upper and lower clause measures for the clauses generated by the transformations. Instead, they only specify bounds of these values; the values themselves are chosen only when instantiating the framework to a concrete system.

Observe from the definition of atom measures that we can always assign $\mathbf{0}$ to α_{min}. However, by setting a more accurate estimate of α_{min}, we can allow more folding steps. As an example, consider any conjunctive folding step where the folded clause $C \in P_i$ has more body atoms than the folder clause $D \in P_j$, and $\gamma_{lo}^{i}(C) = \gamma_{hi}^{j}(D)$. Such a folding step will not be allowed if $\forall A\ \alpha_{min}(A) = \mathbf{0}$.

The Need for Approximate Clause Measures : In the Kanamori-Fujita system, a counter (corresponding to our clause measure) is associated with every clause. Roughly speaking, the counter associated with a clause $C \in P_i$ where $C \equiv A{:-}\ A_1, \ldots, A_n$ indicates the number of interior nodes in the smallest proof tree in P_0 that derives $A_1, \ldots, A_n$ from A. Thus, it is the amount saved (in terms of proof tree size, compared to the smallest proof in P_0) whenever C is used in a proof in P_i. The folding rule is applicable provided the savings accrued in the folded clause is more than that in the folder clause.

To see why a single counter is inadequate for disjunctive folding, consider the following example:

C_1: `p :- r, t.` (x_1)	
C_2: `p :- s, t.` (x_2)	C': `p :- q, t.` (?)
C_3: `q :- r.` (x_3)	C_3: `q :- r.` (x_3)
C_4: `q :- s.` (x_4)	C_4: `q :- s.` (x_4)
Program P_i	Program P_{i+1}

[2] Intuitively, if the clause measure of C_l "exceeds" the clause measure of D_l then we can fold C_l using D_l.

P_{i+1} is obtained from P_i by folding $\{C_3, C_4\}$ into $\{C_1, C_2\}$. Now, the savings due to C' in a proof of P_{i+1} depends on whether C_3 or C_4 is used to resolve $\mathtt{q}$ in that proof. Since this information is unknown at transformation time, we can only keep approximate information about savings. In our framework we choose to approximate the savings by the closed interval $[\gamma_{lo}, \gamma_{hi}]$.

We now have the necessary machinery for establishing total correctness of a sequence of unfold/fold transformations.

Lemma 1 (Preserving Weak Measure Consistency) *Let $P_0, \ldots, P_i$ be a transformation sequence of measure consistent programs such that $M(P_0) = M(P_j)$ for all $0 \leq j \leq i$. Let P_{i+1} be obtained from P_i by applying measure-preserving unfolding or measure-preserving folding. Then, all ground proofs of P_{i+1} are weakly measure consistent.*

Proof Sketch. The proof proceeds by induction on the size of ground proofs of P_{i+1}. Let T be a ground proof of some ground atom A in P_{i+1}, and let $A{:-}\ A_1, ..., A_n$ (where $n \geq 0$) be the ground instance of a clause $C \in P_{i+1}$ that is used at the root of the proof T. Then the subproofs of $A_1, ..., A_n$ in T are weakly measure consistent by induction hypothesis.

Hence, it suffices to show that, $\alpha(A) \preceq \gamma_{hi}^{i+1}(C) \oplus \sum_{1 \leq l \leq n} \alpha(A_l)$. To show this, we consider three cases: (1) C was inherited from P_i. (2) C was obtained from P_i by unfolding; and (3) C was obtained from P_i by folding. In each of these three cases, we can show the above inequality by assuming $M(P_{i+1}) \subseteq M(P_i)$ (which follows from theorem 1). □

Theorem 2 (Total Correctness) *Let $P_0, P_1, \ldots, P_i$ be a transformation sequence of measure consistent programs such that $M(P_0) = M(P_j)$ for all $0 \leq j \leq i$. Let P_{i+1} be obtained from P_i by applying measure-preserving unfolding or measure-preserving folding. Then,* (i) *$M(P_{i+1}) = M(P_i)$ and* (ii) *P_{i+1} is a measure-consistent program.*

Proof. By theorem 1, we have $M(P_{i+1}) \subseteq M(P_i)$, and by lemma 1 we know that all ground proofs of P_{i+1} are weakly measure consistent. Hence it is sufficient to prove that (1) $M(P_i) \subseteq M(P_{i+1})$ and (2) $\forall A \in M(P_{i+1})$, A has a strongly measure consistent proof in P_{i+1}.

Consider any ground atom $A \in M(P_i)$. Since P_i is measure consistent, A has a strongly measure consistent proof T in P_i. We now construct a strongly measure consistent proof T' of A in P_{i+1}. Construction of T' proceeds by induction on atom measures. Let C be a clause used at the root of T. Let $A{:-}\ A_1, ..., A_n$ (where $n \geq 0$) be the ground instantiation of C at the root of T. Since T is strongly measure consistent $\alpha(A_i) \prec \alpha(A)$, for all $1 \leq i \leq n$. Hence, we have strongly measure consistent proofs $T'_1, ..., T'_n$ of $A_1, ..., A_n$ in P_{i+1}. We construct T' by considering the following cases:

Case 1: C is *inherited* from P_i into P_{i+1}

T' is constructed with $A{:-}\ A_1, ..., A_n$ at its root and $T'_1, ..., T'_n$ as its children. This proof T' is strongly measure consistent.

Case 2: C is *unfolded.*
Let A_1 be the atom in the body of C which is unfolded. Let the clause used to resolve A_1 in T be C_1 and the ground instance of C_1 used be A_1:– $A_{1,1},...,A_{1,l_1}$. By definition of unfolding, A:– $A_{1,1},...,A_{1,l_1},A_2,...,A_n$ is a ground instance of a clause C_1' in P_{i+1} with $\gamma_{lo}^{i+1}(C_1') \preceq \gamma_{lo}^i(C) \oplus \gamma_{lo}^i(C_1)$. Also, $\alpha(A_{1,j}) \prec \alpha(A_1) \prec \alpha(A)$, for all $1 \leq j \leq l_1$. Thus, we have strongly measure consistent proofs $T'_{1,1},...,T'_{1,l_1}$ of $A_{1,1},...,A_{1,l_1}$ in P_{i+1}. The proof T' is now constructed by applying A:– $A_{1,1},...,A_{1,l_1},A_2,...,A_n$ at the root, and putting $T'_{1,1},...,T'_{1,l_1},T'_2,...,T'_n$ as the children. Since T is strongly measure consistent,

$$\begin{aligned}
&\alpha(A) \succeq \gamma_{lo}^i(C) \oplus \textstyle\sum_{1\leq j\leq n}\alpha(A_j) \text{ and } \alpha(A_1) \succeq \gamma_{lo}^i(C_1) \oplus \textstyle\sum_{1\leq j\leq l_1}\alpha(A_{1,j})\\
\Longrightarrow\ &(\alpha(A)\oplus\alpha(A_1)) \succeq \gamma_{lo}^i(C) \oplus \gamma_{lo}^i(C_1) \oplus \textstyle\sum_{1\leq j\leq n}\alpha(A_j) \oplus \textstyle\sum_{1\leq j\leq l_1}\alpha(A_{1,j})\\
\Longrightarrow\ &\alpha(A) \succeq \gamma_{lo}^{i+1}(C_1') \oplus \textstyle\sum_{2\leq j\leq n}\alpha(A_j) \oplus \textstyle\sum_{1\leq j\leq l_1}\alpha(A_{1,j})
\end{aligned}$$

Hence, T' is a strongly measure consistent proof in P_{i+1}.
Case 3: C is *folded.*
Let C (potentially with other clauses) be folded, using folder clauses from P_j, $j \leq i$, to clause C' in P_{i+1}. Assume that $A_1,...,A_k$ are the instances of the folded atoms in C. Then, C' has a ground instance of the form A:– $B,A_{k+1},...,A_n$ where B:– $A_1,...,A_k$ is a ground instance of a folder clause $D \in P_j$.[3] Since $M(P_i) = M(P_j)$ and $A_1,...,A_k$ are provable in P_i they must also be provable in P_j. Moreover, since $D \in P_j$, $B \in M(P_j) = M(P_i)$. Since P_j is measure consistent, $\alpha(B) \preceq \gamma_{hi}^j(D) \oplus \sum_{1\leq l\leq k}\alpha(A_l)$.

Now, by the strong measure consistency of T,

$$\begin{aligned}
\alpha(A) &\succeq \gamma_{lo}^i(C) \oplus \textstyle\sum_{1\leq l\leq k}\alpha(A_l) \oplus \textstyle\sum_{k+1\leq l\leq n}\alpha(A_l)\\
&\succeq \gamma_{lo}^i(C) \oplus (\alpha(B) \ominus \gamma_{hi}^j(D)) \oplus \textstyle\sum_{k+1\leq l\leq n}\alpha(A_l) \qquad \cdots\cdots(*)\\
&\succeq (\gamma_{lo}^i(C) \ominus \gamma_{hi}^j(D)) \oplus \alpha(B) \oplus \textstyle\sum_{k+1\leq l\leq n}\alpha_{min}(A_l)\\
&\succ \alpha(B) \text{ (by condition of measure preserving folding)}
\end{aligned}$$

Now, by induction hypothesis, B has a strongly measure consistent proof T'_B in P_{i+1}. We construct T', the proof of A in P_{i+1}, with A:– $B,A_{k+1},...,A_n$ at its root, and $T'_B,T'_{k+1},...,T'_n$ as its children. To show that T' is strongly measure consistent, note that $\gamma_{lo}^{i+1}(C') \preceq (\gamma_{lo}^i(C) \ominus \gamma_{hi}^j(D))$ according to the definition of measure preserving folding, as C and D are folded and folder clauses. Combining this with (*) we get,

$$\alpha(A) \succeq \gamma_{lo}^{i+1}(C') \oplus \alpha(B) \oplus \textstyle\sum_{k+1\leq l\leq n}\alpha(A_l)$$

This completes the proof. □

Note that by applying measure preserving unfolding/folding to program P_i, we can generate a clause which is also inherited from P_i. It is straightforward to adjust the clause measures of P_{i+1} that will still ensure that P_{i+1} remains measure consistent (details are omitted).

[3] Recall that in the folding transformation, all clauses in P_j whose head is unifiable with B are folder clauses.

3 Constructing Concrete Unfold/Fold Systems by Instantiating the Framework

To construct a concrete unfold/fold transformation system from our abstract framework, the following parameters need to be instantiated :

1. a measure structure μ;
2. atom measure α and α_{min};
3. clause measure $(\gamma_{lo}, \gamma_{hi})$ for clauses in the initial program P_0 such that P_0 is measure consistent; and
4. functions to compute the clause measure of new clauses obtained by the transformations such that they satisfy the constraints imposed by equations (1) through (4) (refer Rules 3 and 4).

Note that there are *no further* proof obligations. Once the above four elements are defined, total correctness of the transformation system is *guaranteed* by the framework.

3.1 Existing Unfold/fold Systems

We first show how our framework can be instantiated to obtain the Kanamori-Fujita and the extended Tamaki-Sato systems. To the best of our knowledge, these are the only two existing systems that allow folding using recursive clauses. However in both of these systems folding is conjunctive.

The Kanamori-Fujita System [8]: This system can be obtained as an instance of our framework as follows:

1. $\mu = \langle \mathbb{Z}, +, <, \mathbb{N} \rangle$. This measure structure corresponds to the use of integer counters in [8].
2. $\alpha(A)$ = number of nodes in the smallest proof of A in P_0, and for any atom A, $\alpha_{min}(A) = 1$. Thus, $\alpha(A)$ denotes the *rank* of A described in [8].
3. $\forall C \in P_0 \; \gamma^0_{lo}(C) = \gamma^0_{hi}(C) = 1$. Since all clause measures are 1, it follows immediately from the definition of atom measures that the smallest proofs of any ground goal G are strongly measure consistent, and all proofs in P_0 are weakly measure consistent. Hence P_0 is measure consistent.
4. $\forall C \in P_{i+1} - P_i$ (i.e., new clauses in P_{i+1}), $\gamma^{i+1}_{lo}(C) = GLB^{i+1}(C)$ and $\gamma^{i+1}_{hi}(C) = LUB^{i+1}(C)$. Under the given measure structure, it is immediate that the above definition is identical to the computation on counters in [8].

Furthermore, the measure preserving folding rule (Rule 4) is applied only when both folder and folded clauses are singleton sets. It is easy to see a one-to-one correspondence between the conditions on unfold/fold transformations of the above instantiation and the Kanamori-Fujita system.

The Extended Tamaki-Sato System [21]: In this system all the predicate symbols are partitioned into n strata. In the initial program a predicate from stratum j is defined using only predicates from strata $\leq j$. We can obtain this system as an instance of our framework as follows:

1. $\mu = \langle \mathbb{Z}^n, \oplus, \prec, \mathbb{N}^n \rangle$ where $\oplus$ denotes coordinate-wise integer addition of n-tuples of integers, and $\prec$ denotes the lexicographic $<$ order over n-tuples of integers. The n-tuples in the measure structure will correspond to the n strata of the original program.
2. $\alpha(A) = \min(\{w(T) \mid T \text{ is a proof of } A \text{ in } P_0\})$, where $w(T)$ is the *weight* of the proof T defined as an n-tuple $\langle w_1, \ldots, w_n \rangle$ such that $\forall 1 \leq j \leq n$, w_j is the number of nodes of predicates from stratum j in T. $\alpha(A)$ corresponds to the notion of *weight-tuple measure* of A defined in [21].
   For any atom A, $\alpha_{min}(A) = \mathbf{0} = \langle 0, \ldots, 0 \rangle$.
3. $\forall C \in P_0$, $\gamma^0_{lo}(C) = \gamma^0_{hi}(C) = \langle w_1, \ldots, w_n \rangle$, where $C \equiv A{:-}\ A_1, \ldots, A_n$ and for $1 \leq j \leq n$, $w_j = 1$ if the predicate symbol of A is from stratum j, and 0 otherwise.
   For any $A \in M(P_0)$, the proof T that defines $\alpha(A)$ (item 2 above) is strongly measure consistent. Weak measure consistency of ground proofs in P_0 is established by induction on their size.
4. $\forall C \in P_{i+1} - P_i$, $\gamma^{i+1}_{hi}(C) = LUB^{i+1}(C)$ and $\gamma^{i+1}_{lo}(C) = \mathit{approx}(GLB^{i+1}(C))$. The function *approx* reduces a measure as follows. Let $u = \langle u_1, \ldots, u_n \rangle$ and k_{min} be the smallest index k such that $u_k > 0$. Then $\mathit{approx}(u) = \langle u'_1, \ldots, u'_n \rangle$ where $u'_{k_{min}} = 1$ and is 0 elsewhere.

As in the Kanamori-Fujita system, here also the measure preserving folding rule is applied only when both folder and folded clauses are singleton sets.

To establish the correspondence between the above instantiation and the extended Tamaki-Sato system, recall that the latter associates a descent level with each clause of every program in a transformation sequence. If a clause C in P_i has the descent level k, then with the above instantiation, $\gamma^i_{lo}(C) = \langle l_1, \ldots, l_n \rangle$ where $l_k = 1$ and 0 elsewhere; i.e. the only non-zero entry in its lower clause measure appears in the k^{th} position. Thus our lower clause measure precisely captures the information that is kept track of by the extended Tamaki-Sato system.

Assigning Measure Structures and Clause Measures Observe that our framework does not prescribe exact values to the clause measures. Instead it bounds the clause measures from above and below. So an important aspect of our instantiation involves assigning values to the clause measures that satisfy these constraints. From an abstract point of view, the Kanamori-Fujita system uses a relatively coarse measure space ($\mathbb{Z}$) but within this space it maintains accurate clause measures (integer counters). Our instantiation reflects this by not relaxing the bounds while updating the clause measures (see step 4 of the instantiation). On the other hand, the extended Tamaki-Sato system uses a more fine-grained measure space ($\mathbb{Z}^n$). But this measure space is not completely utilized since clause measures are the descent level of clauses, which can be simply represented by an integer. Therefore in step 4 of our instantiation we accordingly loosened the bound. As far as the Gergatsoulis-Katzouraki [7] and original Tamaki-Sato systems [20] are concerned, first note that they do not permit folding using recursive clauses. These systems use coarse measure spaces. Moreover they do not even fully utilize these measure spaces as is evident from the lesser amount of book keeping performed by them. By choosing a coarse measure structure and relaxing the bounds along lines similar to the extended Tamaki-Sato system we have been able to instantiate these two systems as well. Details are omitted.

3.2 SCOUT— A New Unfold/fold System

We now construct SCOUT, an unfold/fold transformation system for definite logic programs that allows disjunctive folding using recursive clauses. It incorporates the notion of strata from the extended Tamaki-Sato system into the counters of the Kanamori-Fujita system. Thus with every clause it maintains a *pair* of stratified counters as the clause measure. The instantiation is as follows. We assume that the predicate symbols appearing in the initial program P_0 are partitioned into n strata, as in the extended Tamaki-Sato system.

1. $\mu = \langle \mathbb{Z}^n, \oplus, \prec, \mathbb{N}^n \rangle$ where $\oplus$ denotes coordinate-wise integer addition of n-tuples of integers, and $\prec$ denotes the lexicographic $<$ order over n-tuples of integers.
2. $\alpha(A)$ is defined exactly as in the instantiation of the extended Tamaki-Sato system above. For any atom A we set $\alpha_{min}(A) = \langle w_1, \ldots, w_n \rangle$ where $w_j = 1$ if A is from stratum j and 0 elsewhere.
3. Clause measure of clauses in P_0 is defined exactly as in the instantiation of the extended Tamaki-Sato system above. Therefore the proofs of measure consistency are also identical.
4. $\forall C \in P_{i+1} - P_i$, $\gamma_{lo}^{i+1}(C) = GLB^{i+1}(C)$ and $\gamma_{hi}^{i+1}(C) = LUB^{i+1}(C)$.

SCOUT provides a solution to two important (and orthogonal) problems that have thus far remained open: folding using clauses that have disjunctions as well as recursion, and combining the stratification-based (extended) Tamaki-Sato system with the counter-based Kanamori-Fujita system thereby obtaining a single system that strictly subsumes either of them even when restricted to conjunctive folding (See [13] for a formal proof of this claim).

It is interesting to note that by simple inspection of the instantiations, one can see that when the number of strata is 1 and only conjunctive folding is permitted, SCOUT collapses to the Kanamori-Fujita system. Collapsing SCOUT to other existing unfold/fold systems by varying the number of strata and extending the parameters (e.g. measure structure) remains an interesting open problem.

4 Goal Replacement

Augmenting an unfold/fold transformation system with the goal replacement rule makes it more powerful. In this section we incorporate goal replacement to our parameterized framework. Goal replacement allows semantically equivalent conjunctions of atoms to be freely interchanged. We formally define it below. For a conjunction of atoms $A_1, ..., A_n$, we use the notation $vars(A_1, ..., A_n)$ to denote the set of variables in $A_1, ..., A_n$.

Rule 5 (Goal Replacement) Let C be a clause $A{:-}\ A_1, \ldots, A_k, G$ in P_i, and G' be an atom such that $vars(G) = vars(G') \subseteq vars(A, A_1, ..., A_k)$. Suppose for all ground instantiation θ of G, G' we have $P_i \vdash G\theta \Leftrightarrow P_i \vdash G'\theta$. Then $P_{i+1} := (P_i - \{C\}) \cup \{C'\}$ where $C' \equiv A{:-}\ A_1, \ldots, A_k, G'$. □

Note that although we replace a single atom G by another atom G' (where G and G' do not contain any internal variables), we can replace conjunctions of atoms using a sequence of folding, goal replacement and unfolding transformations.

The above transformation is partially correct (a formal proof appears in [13]). However, if goal replacement is applied to a measure consistent program P_i it is totally correct. But then we also need to ensure that the resulting program P_{i+1} is measure consistent. If this is ensured, then even if goal replacement is interleaved with irreversible folding total correctness will be preserved. Formally,

Rule 6 (Measure Preserving Goal Replacement) Suppose program P_{i+1} is obtained from program P_i by applying the goal replacement transformation as described in Rule 5. Let there exist $\delta, \delta' \in \mathcal{M}$ (where measure structure is $\mu = \langle \mathcal{M}, \oplus, \prec, \mathcal{W} \rangle$) such that for all ground instantiation θ of G, G', we have: *(i)* $\delta \preceq \alpha(G\theta) \ominus \alpha(G'\theta) \preceq \delta'$ *(ii)* $\gamma^i_{lo}(C) \oplus \delta \oplus \sum_{1 \le p \le k} \alpha_{min}(A_p) \succ \mathbf{0}$. Then

$$\gamma^{i+1}_{lo}(C') \preceq \gamma^i_{lo}(C) \oplus \delta \tag{5}$$
$$\gamma^{i+1}_{hi}(C') \succeq \gamma^i_{hi}(C) \oplus \delta' \tag{6}$$

The clause measures of the other clauses of P_{i+1} are inherited from P_i. □

We now present a formal proof of total correctness and preservation of measure consistency of the above rule.

Theorem 3 *Let P_{i+1} be derived from P_i by applying measure preserving goal replacement as described in rule 6. If P_i is measure consistent, then $M(P_i) = M(P_{i+1})$ and P_{i+1} is also measure consistent.*

Proof. Since measure preserving goal replacement is a special case of the goal replacement transformation in rule 5, we have $M(P_{i+1}) \subseteq M(P_i)$ by partial correctness of rule 5. Therefore it is sufficient to prove that : (1) all ground proofs of P_{i+1} are weakly measure consistent (2) $M(P_i) \subseteq M(P_{i+1})$ (3) $\forall B \in M(P_{i+1})$ there exists a strongly measure consistent proof of B in P_{i+1}. We prove proof obligation (1) separately. Proof obligations (2) and (3) are proved by showing that : $\forall B \in M(P_i)$ there exists a strongly measure consistent proof of B in P_{i+1}. This is sufficient since we know $M(P_{i+1}) \subseteq M(P_i)$.

First, we prove that all ground proofs of P_{i+1} are weakly measure consistent. The proof proceeds by induction on the size of ground proofs in P_{i+1}. Let T be a ground proof of a ground atom B in P_{i+1}. If the clause used at the root of T

is not the new clause C', then the proof follows by induction hypothesis and the measure consistency of P_i. If the clause used at the root of T is C', then let the ground instance of C' used at the root of T be $A\theta$:– $A_1\theta, \ldots, A_k\theta, G'\theta$. By induction hypothesis, the proofs of $A_1\theta, \ldots, A_k\theta, G'\theta$ in T are weakly measure consistent. It suffices to show that $\alpha(A) \preceq \gamma_{hi}^{i+1}(C') \oplus \sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus \alpha(G'\theta)$ Now, $G'\theta \in M(P_{i+1}) \Rightarrow G'\theta \in M(P_i)$. Hence by rule 5 we have $G\theta \in M(P_i)$. Also, $\forall 1 \leq l \leq k\ A_l\theta \in M(P_i)$ (as $M(P_{i+1} \subseteq M(P_i))$. Then, $A\theta$:– $A_1\theta, \ldots A_k\theta, G\theta$ is a ground instantiation of C which appears at the root of some ground proof in P_i. Since P_i is measure consistent we have

$$\begin{aligned}\alpha(A) &\preceq \gamma_{hi}^{i}(C) \oplus \textstyle\sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus \alpha(G\theta)\\ &\preceq \gamma_{hi}^{i}(C) \oplus \textstyle\sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus (\ \alpha(G'\theta) \oplus \delta'\ )\\ &\preceq \gamma_{hi}^{i+1}(C') \oplus \textstyle\sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus \alpha(G'\theta)\end{aligned}$$

Now, we prove that $\forall B \in M(P_i)$ there is a strongly measure consistent proof of B in P_{i+1}. Since P_i is measure consistent, it suffices to translate a strongly measure consistent proof T of B in P_i to a strongly measure consistent proof T' of B in P_{i+1} for all $B \in M(P_i)$. We do this translation by induction on the atom measures. If the clause used at the root of T is not C (where C is the clause in P_i that is replaced) then the proof follows from the definition of strong measure consistency and induction hypothesis. Let C be the clause used at the root of T (a strongly measure consistent proof of A in P_i) and let $A\theta$:– $A_1\theta, \ldots, A_k\theta, G\theta$ be the ground instance of C used. Then, by strong measure consistency of T, $\alpha(A_l\theta) \prec \alpha(A\theta)$ for all $1 \leq l \leq k$. By induction hypothesis, we then have strongly measure consistent ground proofs $T_1', \ldots, T_k'$ of $A_1\theta, \ldots, A_k\theta$ in P_{i+1}. Also, by strong measure consistency of T

$$\begin{aligned}\alpha(A) &\succeq \gamma_{lo}^{i}(C) \oplus \textstyle\sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus \alpha(G\theta)\\ &\succeq \gamma_{lo}^{i}(C) \oplus \textstyle\sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus (\ \alpha(G'\theta) \oplus \delta\ ) \quad \cdots\cdots (*)\\ &\succeq (\ \gamma_{lo}^{i}(C) \oplus \textstyle\sum_{1\leq l\leq k} \alpha_{min}(A_l\theta) \oplus \delta\ ) \oplus \alpha(G'\theta)\\ &\succ \alpha(G'\theta) \text{ (By condition (ii) of rule 6)}\end{aligned}$$

Then, by induction hypothesis, $G'\theta$ has a proof $T'_{G'\theta}$ in P_{i+1}. The ground proof T' is constructed with $A\theta$:– $A_1\theta, \ldots, A_k\theta, G'\theta$ at the root (this is a ground instance of C', the new clause in P_{i+1}) and $T_1', \ldots, T_k', T'_{G'\theta}$ as its children. To show that this proof T' is measure consistent, note that $\gamma_{lo}^{i+1}(C') \preceq \gamma_{lo}^{i}(C) \oplus \delta$. Combining this with (*), we get

$$\alpha(A) \succeq \gamma_{lo}^{i+1}(C') \oplus \sum_{1\leq l\leq k} \alpha(A_l\theta) \oplus \alpha(G'\theta)$$

This completes the proof. □

Observe that, similar to the goal replacement transformation in [8, 20, 21] the conditions under which rule 6 may be applied are not testable at transformation time. For testability we need to (1) determine whether G and G' are semantically equivalent, and (2) estimate δ and δ' such that the clause measures of P_{i+1} can be computed.

Semantic equivalence is undecidable in general and can be conservatively approximated using program analysis. To estimate δ and δ' observe that any δ' which dominates the atom measure of all ground atoms satisfies the conditions of Rule 6. However, such a δ' may not always exist in the given measure structure. In such cases, we can extend the measure structure $\mu = \langle \mathcal{M}, \oplus, \prec, \mathcal{W} \rangle$ to $\langle \mathbb{Z} \times \mathcal{M}, \oplus', \preceq', \mathbb{N} \times \mathcal{W} \rangle$, where $\forall z_1, z_2 \in \mathbb{Z}$ and $\forall m_1, m_2 \in \mathcal{M}$ $(z_1, m_1) \oplus' (z_2, m_2) = (z_1 + z_2, m_1 \oplus m_2)$, and $\preceq'$ is the lexicographic ordering of pairs from $\mathbb{Z} \times \mathcal{M}$. Atom measures in this extended measure space are of the form $(0, w)$ (where $w \in \mathcal{W}$). We set $\delta' = (1, \mathbf{0})$, which is lexicographically greater than all atom measures. Also, in certain cases we can define a lower bound of δ as follows. Let B be the atom in the body of a clause in P_i that is replaced and let $\{C_1, \ldots, C_n\}$ be the clauses in P_i that unify with B. Then, $\delta \preceq \min_{1 \le k \le n}(\gamma^i_{lo}(C_k) - \alpha_{min}(hd(C_k)))$, where $hd(C_k)$ is the head atom of C_k (for details see [14]).

The above steps define a procedure to add goal replacement to any arbitrary unfold/fold system instantiated in our framework. More importantly, this is done by simply manipulating the measures; the proofs of correctness of the augmented transformation system follow immediately from the proofs of our framework.

5 Conclusion

The development of a parameterized framework for unfold/fold transformations has several important implications. It enables us to compare existing transformation systems and modify them without redoing the correctness proofs (e.g., extending measures for goal replacement in Section 4). It also facilitates the development of new transformations systems. For instance, we derived SCOUT which permits folding using multiple recursive clauses. Such a transformation system is particularly important for verifying parameterized concurrent systems (such as a n-process token ring for arbitrary n) using logic program evaluation and deduction [4, 16].

In [15], we have extended the work reported in this paper to obtain generalized unfold/fold transformation systems for normal logic programs. Aravindan and Dung [1] developed an approach to parameterize the correctness proofs of the original Tamaki-Sato system with respect to various semantics based on the notion of *semantic kernels*. Incorporating the idea of semantic kernel into our framework yields a framework that is parameterized with respect to the measure structures as well as semantics.

In future, it would be interesting to study whether we can develop similar parameterized unfold/fold transformation frameworks for other programming paradigms such as functional and concurrent constraint programming languages [5, 17] as well as process algebraic specification languages (*e.g.* CCS) [6].

Acknowledgements: We would like to thank the anonymous referees for their valuable comments. We thank Alberto Pettorossi and Maurizio Proietti for useful discussions and pointers to earlier work. We also thank Sandro Etalle for relevant references, Taisuke Sato for providing us with a copy of his technical report [21], and David S. Warren for his comments about an earlier draft of this paper.

References

1. C. Aravindan and P.M. Dung. On the correctness of unfold/fold transformations of normal and extended logic programs. *Journal of Logic Programming*, pages 295–322, 1995.
2. A. Bossi, N. Cocco, and S. Dulli. A method of specializing logic programs. *ACM TOPLAS*, pages 253–302, 1990.
3. D. Boulanger and M. Bruynooghe. Deriving unfold/fold transformations of logic programs using extended OLDT-based abstract interpretation. *Journal of Symbolic Computation*, pages 495–521, 1993.
4. B. Cui, Y. Dong, X. Du, K. Narayan Kumar, C.R. Ramakrishnan, I.V. Ramakrishnan, A. Roychoudhury, S.A. Smolka, and D.S. Warren. Logic programming and model checking. In *Proceedings of PLILP/ALP, LNCS 1490*, pages 1–20, 1998.
5. S. Etalle, M. Gabrielli, and M.C. Meo. Unfold/fold transformations of CCP programs. In *Proceedings of CONCUR*, 1998.
6. Nicoletta De Francesco and Antonella Santone. A transformation system for concurrent processes. *Acta Informatica*, 35(12):1037–1073, 1998.
7. M. Gergatsoulis and M. Katzouraki. Unfold/fold transformations for definite clause programs. In *Proceedings of PLILP, LNCS 844*, pages 340–354, 1994.
8. T. Kanamori and H. Fujita. Unfold/fold transformation of logic programs with counters. In *USA-Japan Seminar on Logics of Programs*, 1987.
9. J.W. Lloyd. *Foundations of Logic Programming, Second Edition*. Springer-Verlag, 1993.
10. M. J. Maher. Correctness of a logic program transformation system. Technical report, IBM T.J. Watson Research Center, 1987.
11. A. Pettorossi and M. Proietti. *Transformation of logic programs*, volume 5 of *Handbook of Logic in Artificial Intelligence*, pages 697–787. Oxford University Press, 1998.
12. A. Pettorossi, M. Proietti, and S. Renault. Reducing nondeterminism while specializing logic programs. In *Proceedings of POPL*, pages 414–427, 1997.
13. A. Roychoudhury, K. Narayan Kumar, C.R. Ramakrishnan, and I.V. Ramakrishnan. A generalized unfold/fold transformation system for definite logic programs. Technical Report 98/37, Dept. of Computer Science, SUNY Stony Brook, 1998.
14. A. Roychoudhury, K. Narayan Kumar, C.R. Ramakrishnan, and I.V. Ramakrishnan. Proofs by program transformations. *Accepted for LOPSTR*, 1999.
15. A. Roychoudhury, K. Narayan Kumar, and I.V. Ramakrishnan. Beyond Tamaki-Sato style unfold/fold transformations for normal logic programs. Technical Report 99/21, Dept. of Computer Science, SUNY Stony Brook, 1999.
16. A. Roychoudhury, C. R. Ramakrishnan, I. V. Ramakrishnan, and S. A. Smolka. Tabulation based Induction proofs with applications to Automated Verification. In *Workshop on Tabulation in Parsing and Deduction*, pages 83–88, 1998.
17. David Sands. Total correctness by local improvement in the transformation of functional programs. *ACM TOPLAS*, 18(2):175–234, 1996.
18. H. Seki. Unfold/fold transformation of stratified programs. *In Theoretical Computer Science*, pages 107–139, 1991.
19. H. Seki. Unfold/fold transformation of general logic programs for well-founded semantics. *In Journal of Logic Programming*, pages 5–23, 1993.
20. H. Tamaki and T. Sato. Unfold/fold transformations of logic programs. In *Proceedings of International Conference on Logic Programming*, pages 127–138, 1984.
21. H. Tamaki and T. Sato. A generalized correctness proof of the unfold/ fold logic program transformation. Technical report, Ibaraki University, Japan, 1986.