Automatic Correctness Proofs for Logic Program Transformations

Alberto Pettorossi (1)
Maurizio Proietti (2)
Valerio Senni (1)

(1) DISP, University of Rome “Tor Vergata” (Italy)
(2) IASI-CNR, Rome (Italy)

ICLP 2007, Porto (Portugal)
8–13 September 2007

Correctness of Transformations

A.

P: 1. p ← q
2. r ← q
3. q ←

fold 2 using 1 →

Q: p ← q
r ← p
q ←

in Q: ← p, ← q, □

M(P) = {p,q,r} = M(Q) = {p,q,r} Total Correctness

B.

P: 1. p ← q
2. r ← q
3. q ←

fold 1 using 1 →

Q: p ← p
r ← q
q ←

in Q: ← p, ← p, …

M(P) = {p,q,r} ⊇ M(Q) = { q,r} Partial Correctness

Correctness Theorems

Partial Correctness Theorem

if $M(P) \models P \Rightarrow Q$ i.e., $\forall$ ground $H \leftarrow G_2$ in Q such that $M(P) \models G_2$
$\exists$ ground $H \leftarrow G_1$ in P such that $M(P) \models G_1$

then $M(P) \supseteq M(Q)$

Total Correctness Theorem

if 1. $M(P) \models P \Rightarrow Q$ and
2. $M(P) \models P \Leftarrow Q$ and
3. extra conditions on suitable clause measures

then $M(P) = M(Q)$

Examples of clause measures:

- counters of Kanamori-Fujita [1986],
- counters of Tamaki-Sato [1986],
- measures of Roychoudhury, Narayan Kumar, C.R. Ramakrishnan, I.V. Ramakrishnan [2004]

How to find suitable clause measures ?

Finding Clause Measures

We propose a technique for automatically finding clause measure.

These measures are generated by solving linear constraints on integers
(no complex lexicographic orderings are required).

Constraints depend on the program transformation at hand
(measures are not fixed in advance).

Summary

- Total Correctness Theorem for definition introduction, unfolding, and folding rules.
- Example.
- Extension of the Total Correctness Theorem for the goal replacement rule.
  Unfold/fold proof method for validating goals replacements.

Transformation Rules with Measures and Constraints

- definition introduction

measure

$Newp \leftarrow A_1 \wedge \ldots \wedge A_n$: u

Defs: set of the new definitions.

- unfolding

measure

$H \leftarrow \ldots \wedge A \wedge \ldots$: u

θ_1

in P_0:

$H_1 \leftarrow G_1$: u_1

$\ldots$ θ_m : $\ldots$

$H_m \leftarrow G_m$: u_m

$\longrightarrow$

measure

$(H \leftarrow \ldots \wedge G_1 \wedge \ldots)\,\theta_1$: $u + u_1$

$\ldots$: $\ldots$

$(H \leftarrow \ldots \wedge G_m \wedge \ldots)\,\theta_m$: $u + u_m$

- folding

measure

$H \leftarrow \ldots \wedge B_1\theta \wedge \ldots$: u_1

$\ldots$: $\ldots$

$H \leftarrow \ldots \wedge B_m\theta \wedge \ldots$: u_m

in $P_0 \cup Defs$:

$K \leftarrow B_1$: v_1

$\ldots$: $\ldots$

$K \leftarrow B_m$: v_m

$\longrightarrow$

$H \leftarrow \ldots \wedge K\theta \wedge \ldots$: u

constraints

$u \leq u_1 - v_1$

$\ldots$

$u \leq u_m - v_m$

Total Correctness Theorem (2)

$P_0 \rightarrow \ldots \rightarrow P_n$:

program transformation using definition introduction, unfolding, and folding

Constraints C :

- positive constraints ($u \geq 1$) for each clause with measure u of the final program P_n
- inequalities constrants ($\leq$) for folding steps

Total Correctness Theorem.

If C is satisfiable in **Nat** **then** $M(P_0 \cup Defs) = M(P_n)$.

Example: Transformation to Continuation Passing Style (1)

		measure	constraints
P_0:	1. $p \leftarrow$	u_1	
	2. $q \leftarrow p \wedge q$	u_2	
	3. $q \leftarrow$	u_3	

Looking for a Continuation Passing Style final program $\mathbf{P_n}$ for a new predicate p_{cont} equivalent to p.

- definition introduction:

4. $p_{cont} \leftarrow p$	u_4	
5. $cont(f_{true}) \leftarrow$	u_5	
6. $cont(f_p(X)) \leftarrow p \wedge cont(X)$	u_6	
7. $cont(f_q(X)) \leftarrow q \wedge cont(X)$	u_7	

- fold 4 using 5:

8. $p_{cont} \leftarrow p \wedge cont(f_{true})$	u_8	$u_8 \leq u_4 - u_5$

- fold 8 using 6:

9. $p_{cont} \leftarrow cont(f_p(f_{true}))$	u_9	$u_9 \leq u_8 - u_6$

- unfold 6 using 1 and 2:

10. $cont(f_p(X)) \leftarrow cont(X)$	$u_6 + u_1$
11. $cont(f_p(X)) \leftarrow p \wedge q \wedge cont(X)$	$u_6 + u_2$

Transformation to Continuation Passing Style (2)

		measure	constraints
- fold 11 using 7:			
	12. $\text{cont}(f_p(X)) \leftarrow p \wedge \text{cont}(f_q(X))$	u_{12}	$u_{12} \le u_6 + u_2 - u_7$
- fold 12 using 6:			
	13. $\text{cont}(f_p(X)) \leftarrow \text{cont}(f_p(f_q(X)))$	u_{13}	$u_{13} \le u_{12} - u_6$
- unfold 7 using 3:			
	14. $\text{cont}(f_q(X)) \leftarrow \text{cont}(X)$	$u_7 + u_3$	

C : constraints

- positive constraints for final program $\mathbf{P_n}$: $u_1 \ge 1$ $u_2 \ge 1$ $u_3 \ge 1$
  $u_5 \ge 1$ $u_9 \ge 1$ $u_6 + u_1 \ge 1$ $u_{13} \ge 1$ $u_7 + u_3 \ge 1$
- for folding steps: $u_8 \le u_4 - u_5$ $u_9 \le u_8 - u_6$ $u_{12} \le u_6 + u_2 - u_7$ $u_{13} \le u_{12} - u_6$

C is satisfiable in **Nat**. By Total Correctness Theorem: $\mathbf{M(P_0 \cup Defs) = M(P_n)}$.

Thus, $M(P_0 \cup Defs) \models p$ iff $M(P_0 \cup Defs) \models p_{cont}$ (by clause 4.)
iff $M(P_n) \models p_{cont}$ (by Total Correctness Theorem)

The proof of our Total Correctness Theorem is based on clauses with **weights**.

clause C :

$p_0(...) \leftarrow p_1(...) \wedge ... \wedge p_m(...)$

clause C with **weight** w :

$p_0(..., N_0) \leftarrow N_0 \geq N_1 + ... + N_m + w \quad \wedge \quad p_1(..., N_1) \wedge ... \wedge p_m(..., N_m)$

where the N_i's are natural numbers

Many program transformations require Goal Replacements.

	measure		measure	constraints
$H \leftarrow \ldots \wedge G_1 \wedge \ldots$	w	goal replacement $\longrightarrow$	$H \leftarrow \ldots \wedge G_2 \wedge \ldots$	$w - w_1 + w_2$	$\mathcal{R}$

based on the replacement law :

$$(G_1, w_1) \Rightarrow_X (G_2, w_2)$$

where: - X are the linking variables, i.e., variables of G_1 and G_2

also occurring in the rest of the clause

- w_1, w_2, and $\mathcal{R}$ are determined by the Unfold/Fold Proof Method.

Example: From Quadratic to Linear List Reversal. (1)

		measure	constraints
P_0:	1. app([ ],L,L) ←	u_1	
	2. app([H|T],L,[H|R]) ← app(T,L,R)	u_2	
	3. rev([ ],[ ]) ←	u_3	
	4. rev([H|T],L) ← rev(T,R) ∧ app(R,[H],L)	u_4	
- definition introduction (Defs):			
	5. $g(L_1,L_2,A)$ ← rev(L_1,R) ∧ app(R,A,L_2)	u_5	
- unfold 5 twice:			
	6. g([ ],L,L) ←	$u_5+u_3+u_1$	
	7. g([H|T],L,A) ← rev(T,R) ∧ app(R,[H],S) ∧ app(S,A,L)	u_5+u_4	
- goal replacement (associativity of app):			
	8. g([H|T],L,A) ← rev(T,R) ∧ app([H],A,S) ∧ app(R,S,L)	$u_5+u_4-w_1+w_2$	$\mathcal{R}$
- unfold 8 twice:			
	9. g([H|T],L,A) ← rev(T,R) ∧ app(R,[H|A],L)	$u_5+u_4-w_1+w_2+u_2+u_1$	
- fold 9 using 5:			
	10. g([H|T],L,A) ← g(T,L,[H|A])	u_6	$u_6 \leq u_5+u_4-w_1+w_2+u_2+u_1 - u_5$

Example: From Quadratic to Linear List Reversal. (2)

	measure	constraints
- fold 4 using 5:		
11. rev([H|T],L) ← g(T,L,[H])	u_7	$u_7 \leq u_4 - u_5$

C : constraints

- for final program $\mathbf{P_n}$: $u_1 \geq 1$ $u_2 \geq 1$ $u_3 \geq 1$ $u_5 + u_3 + u_1 \geq 1$ $u_6 \geq 1$ $u_7 \geq 1$
- for folding steps: $u_6 \leq u_4 - w_1 + w_2 + u_2 + u_1$ $u_7 \leq u_4 - u_5$
- for goal replacement $\mathcal{R}$: $u_1 \geq 1$ $u_2 \geq 1$ $u_8 \geq 1$ $u_8 \leq 2u_2$ $u_9 \geq 1$ $u_9 = u_2$
  $w_2 + u_1 \geq 1$ $w_1 \geq w_2$ $u_8 \geq u_9$ (see below)

C is satisfiable in **Nat**. Thus, $\mathbf{M(P_0 \cup Defs) = M(P_n)}$ (by Total Correctness Theorem)

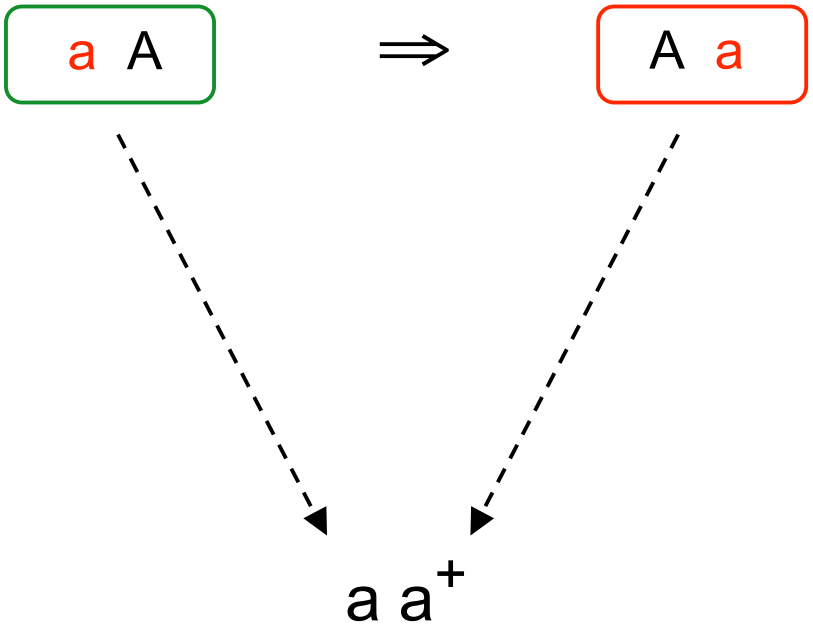
Unfold/Fold Proof Method for Grammars

1. A → a
2. A → a A

replacement →

A → a
A → A a

based on the replacement law:



Unfold/Fold Proof Method for Clauses

1. reach(X,X) ←
2. reach(X,Z) ← arc(X,Y) ∧ reach(Y,Z)

goal replac. →

reach(X,X) ←
reach(X,Z) ← reach(X,Y) ∧ arc(Y,Z)

based on the replacement law:

arc(X,Y) ∧ reach(Y,Z) $\Rightarrow_{\{X,Z\}}$ reach(X,Y) ∧ arc(Y,Z)

L1. new1(X,Z) ← arc(X,Y) ∧ reach(Y,Z)

unfold L1 using 2:
L2. new1(X,Y) ← arc(X,Y)
L3. new1(X,Z) ← arc(X,Y) ∧
arc(Y,Y1) ∧ reach(Y1,Z)
fold L3 using L1:
L4. new1(X,Z) ← arc(X,Y) ∧ new1(Y,Z)

R1. new2(X,Z) ← reach(X,Y) ∧ arc(Y,Z)

unfold R1 using 2:
R2. new2(X,Y) ← arc(X,Y)
R3. new2(X,Z) ← arc(X,Y1) ∧
reach(Y1,Y) ∧ arc(Y,Z)
fold R3 using R1:
R4. new2(X,Z) ← arc(X,Y) ∧ new2(Y,Z)

{L2, L4} new1/new2 ≡ {R2, R4}

Unfold/Fold Proof Method

1. app([ ],L,L) ←
2. app([H|T],L,[H|R]) ← app(T,L,R)

Replacement law:

$app(L_1,L_2,M) \wedge app(M,L_3,L) \Rightarrow_{\{L_1,L_2,L_3,L\}} app(L_2,L_3,R) \wedge app(L_1,R,L)$

L1. $new1(L_1,L_2,L_3,L) \leftarrow app(L_1,L_2,M) \wedge app(M,L_3,L)$

unfold L1:

L2. $new1([\,],L_2,L_3,L) \leftarrow app(L_2,L_3,L)$

L3. $new1([H|T],L_2,L_3,[H|R]) \leftarrow app(T,L_2,M) \wedge app(M,L_3,R)$

fold L3 using L1:

L4. $new1([H|T],L_2,L_3,[H|R]) \leftarrow new1([H|T],L_2,L_3,[H|R])$

R1. $new2(L_1,L_2,L_3,L) \leftarrow app(L_2,L_3,R) \wedge app(L_1,R,L)$

unfold R1:

R2. $new2([\,],L_2,L_3,L) \leftarrow app(L_2,L_3,L)$

R3. $new2([H|T],L_2,L_3,[H|R]) \leftarrow app(L_2,L_3,M) \wedge app(T,M,R)$

fold R3 using R1:

R4. $new2([H|T],L_2,L_3,[H|R]) \leftarrow new2([H|T],L_2,L_3,[H|R])$

new1/new2

{L2, L4} ≡ {R2, R4}

Unfold/Fold Proof Method with Measures and Constraints

1. app([ ],L,L) ← u_1
2. app([H|T],L,[H|R]) ← app(T,L,R) u_2

Replacement law:

$app(L_1,L_2,M) \wedge app(M,L_3,L)$ w_1 $\Rightarrow_{\{L_1,L_2,L_3,L\}}$ $app(L_2,L_3,R) \wedge app(L_1,R,L)$ w_2

L1. w_1

unfold L1 (twice):

L2. $w_1 + u_1$

L3. $w_1 + 2u_2$

fold L3 using L1:

L4. u_8 $u_8 \leq w_1 + 2u_2 - w_1$

R1. w_2

unfold R1:

R2. $w_2 + u_1$

R3. $w_2 + u_2$

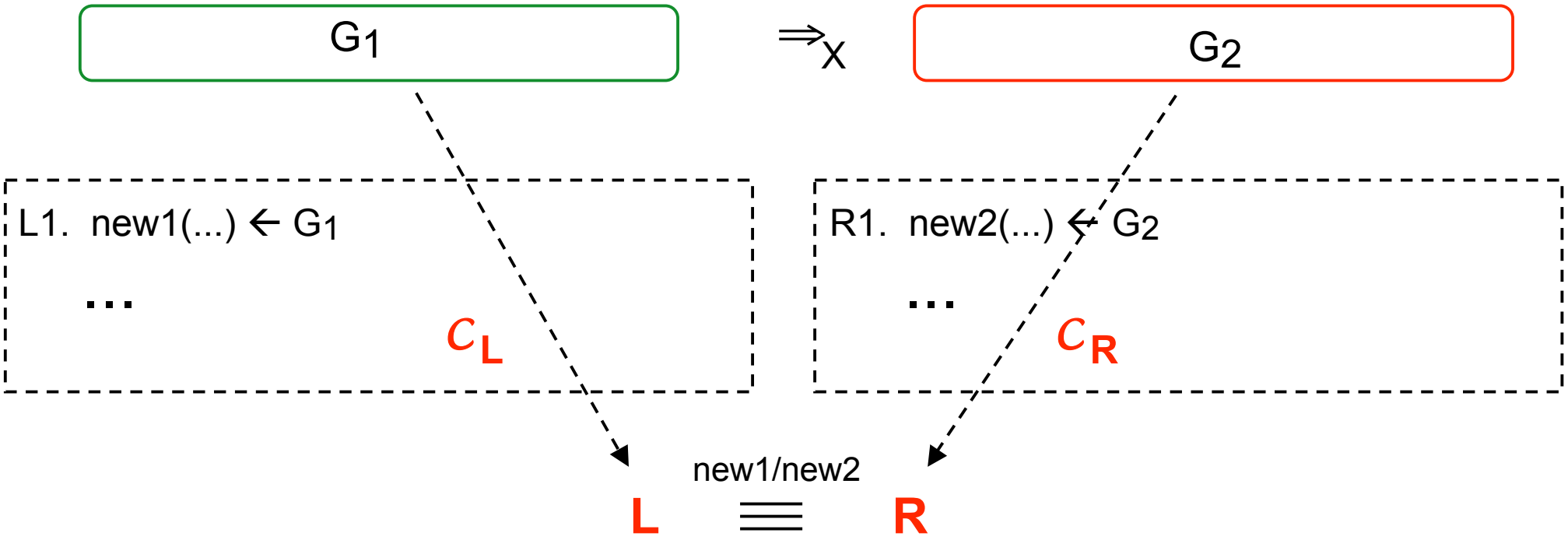
fold R3 using R1:

R4. u_9 $u_9 \leq w_2 + u_2 - w_2$

new1/new2

{L2, L4} ≡ {R2, R4}

Constraints for Goal Replacements



Constraints $\mathcal{R}$: C_L and

$C_R^{=}$ (i.e., C_R with $=$, instead of $\leq$) and

measures of **L** $\geq$ measures of **R**

Constraints for associativity of append

For associativity of append:

$u_1 \geq 1$ $u_2 \geq 1$ $w_1+u_1 \geq 1$ $u_8 \geq 1$ $u_8 \leq w_1 + 2u_2 - w_1$ and

$u_1 \geq 1$ $u_2 \geq 1$ $w_2+u_1 \geq 1$ $u_9 \geq 1$ $u_9 = w_2 + u_2 - w_2$ and

$w_1+u_1 \geq w_2+u_1$ $u_8 \geq u_9$

which are equivalent to:

$u_1 \geq 1$ $u_2 \geq 1$ $u_8 \geq 1$ $u_8 \leq 2u_2$ $u_9 \geq 1$ $u_9 = u_2$

$w_2+u_1 \geq 1$ $w_1 \geq w_2$ $u_8 \geq u_9$

Conclusions and Future Work

- We have reduced the proof of total correctness of a program transformations to the satisfiability of a system of linear constraints on Nat.
- Our technique has been implemented (www.iasi.cnr.it/~proietti/) and several transformation examples have been performed.
- Our technique is similar to the one used for automatically generating suitable orderings on operators (of r.p.o.'s) for proving termination of rewriting systems (see P. Lescanne).
- Our technique is necessary incomplete due to undecidability results.

We will:

- investigate other domains besides Nat for clause measures.
- automate the proofs of goal replacements.