Giornata GULP

Fabio Fioravanti and Maurizio Proietti

IASI-CNR Viale Manzoni 30 00185 Rome, Italy

Alberto Pettorossi

DIPARTIMENTO DI INFORMATICA, SISTEMI E PRODUZIONE

Università di Roma Tor Vergata, Via del Politecnico 1, 00133 Rome, Italy

Bologna, 25 Ottobre 2002

Use of **Program Transformation** methodologies
for Program Improvement and Program Specialization :

- program improvement by
  eliminating unnecessary data structures,
  knowing the context of use, ...
  (e.g., sorting when ordered --> identity or reverse)

for Program Verification :

- proving liveness or safety properties of protocols

for Program Synthesis :

- from first order specifications to constraint logic programs

Transformation via Rules + Strategies

initial program $P_0 \longrightarrow \ldots \longrightarrow P_n$ final program

$$M(P_0) = \ldots = M(P_n)$$

Each rule transforms a program P_i into a new program P_{i+1}

Each rule application preserves the model of interest.

Program Specialization

Given a locally stratified CLP(*D*) program P and

a constrained atom $c(X) \wedge p(X)$

Derive an efficient program P_{sp} defining p_{sp} s. t.

$\forall$ ground term t

$M(P) \models c(t) \wedge p(t)$ iff $M(P_{sp}) \models c(t) \wedge p_{sp}(t)$

where M(P) denotes the perfect *D*-model

---

1. Introduce δ : $p_{sp}(X_1, ..., X_h) \leftarrow c(X) \wedge p(X)$

2. Apply rules according to a strategy: $P \cup \{\delta\} \xrightarrow{\text{rules}}{}^{*} P_{sp}$

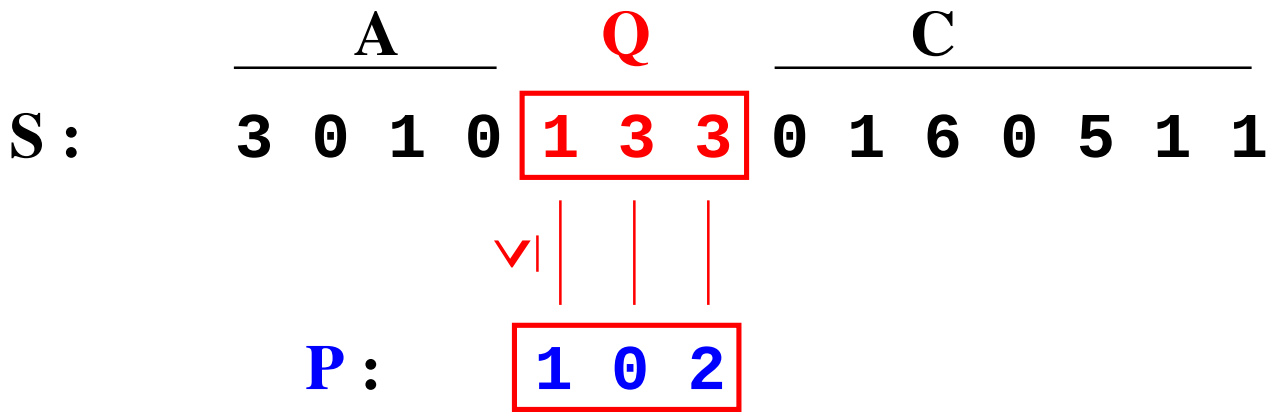
[Lopstr 2000, AGP 2002]

Leq_Match 1

1. leq_match(P,S) ← append(B,C,S) ∧ append(A,Q,B) ∧ leq(P,Q)

2. append([ ], Ys, Ys) ←
3. append([X|Xs], Ys, [X|Zs]) ← append(Xs, Ys, Zs)

4. leq([ ], [ ]) ←
5. leq([X|Xs], [Y|Ys]) ← X ≤ Y ∧ leq(Xs,Ys)



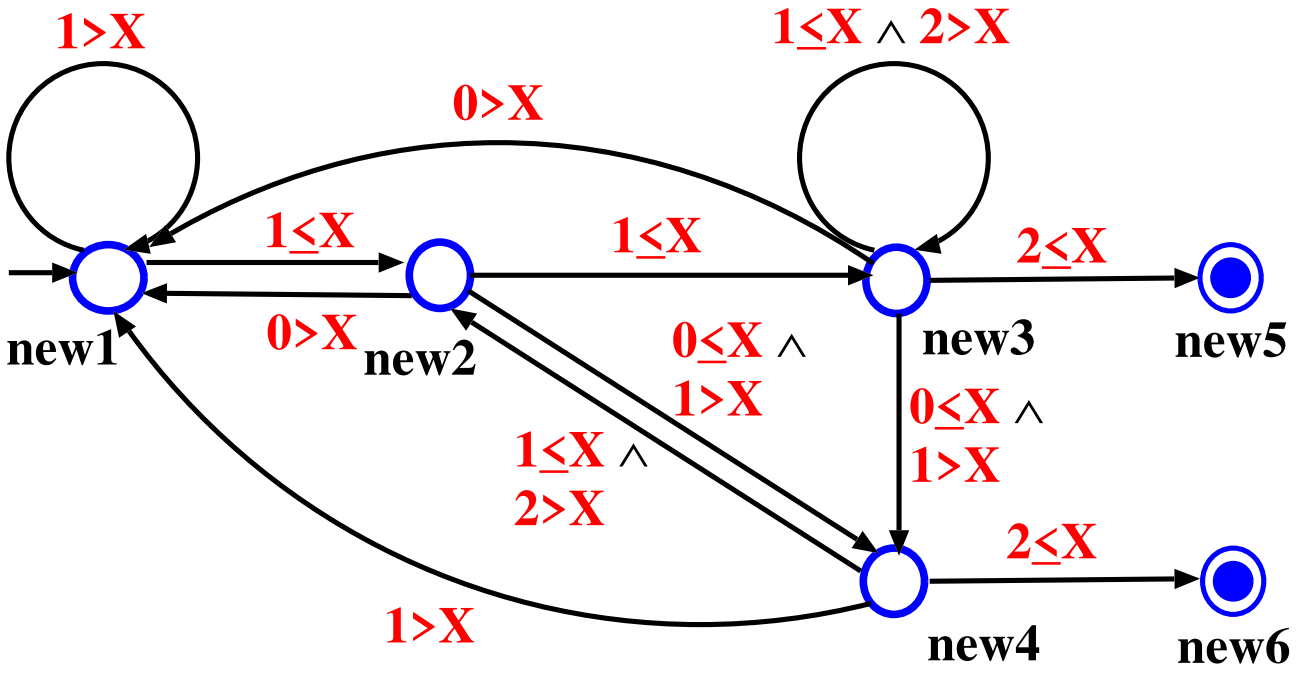
6. leq_match_{sp}(S) ← leq_match([1,0,2],S)

Leq_Match 2

by specializing w.r.t. the goal leq_match([1,0,2], S)
we get the deterministic program:

9. $leq_match_{sp}(S) \leftarrow new1(S)$
16. $new1([X|Xs]) \leftarrow 1 \leq X \wedge new2(Xs)$
17. $new1([X|Xs]) \leftarrow 1 > X \wedge new1(Xs)$
$new2([X|Xs]) \leftarrow 1 \leq X \wedge new3(Xs)$
$new2([X|Xs]) \leftarrow 0 \leq X \wedge 1 > X \wedge new4(Xs)$
$new2([X|Xs]) \leftarrow 0 > X \wedge new1(Xs)$
$new3([X|Xs]) \leftarrow 2 \leq X \wedge new5(Xs)$
$new3([X|Xs]) \leftarrow 1 \leq X \wedge 2 > X \wedge new3(Xs)$
$new3([X|Xs]) \leftarrow 0 \leq X \wedge 1 > X \wedge new4(Xs)$
$new3([X|Xs]) \leftarrow 0 > X \wedge new1(Xs)$
$new4([X|Xs]) \leftarrow 2 \leq X \wedge new6(Xs)$
$new4([X|Xs]) \leftarrow 1 \leq X \wedge 2 > X \wedge new2(Xs)$
$new4([X|Xs]) \leftarrow 1 > X \wedge new1(Xs)$
$new5(Xs) \leftarrow$
$new6(Xs) \leftarrow$

Leq_Match 3



efficiency = deterministic automaton

Basic Idea

- determinism = mutually exclusive clauses

clauses : $H_1 \leftarrow c_1 \wedge G_1$, $H_2 \leftarrow c_2 \wedge G_2$

if $c_1 \wedge c_2 \wedge H_1 = H_2$ is satisfiable

then split :

$H_1 \leftarrow c_1 \wedge G_1$

$H_2 \leftarrow c_1 \wedge c_2 \wedge G_2$

$H_2 \leftarrow \neg c_1 \wedge c_2 \wedge G_2$

$\Rightarrow$

mutually exclusive packets

$H_1 \leftarrow c_1 \wedge G_1$

$H_2 \leftarrow c_1 \wedge c_2 \wedge G_2$

multiple folding $\rightarrow \gamma 1$

$H_2 \leftarrow \neg c_1 \wedge c_2 \wedge G_2$ $\rightarrow \gamma 2$

Transformation Rules

R1. Definition
R2. Unfolding
R3. Folding
R4. Clause Removal
R5. Constraint Replacement
R6-R7. Clause Fusion and Clause Splitting

extension of rules of :
- **Bensaou-Guessarian**
- **Etalle-Gabbrielli**
- **Gergatsoulis-Katzouraki**
- **Tamaki-Sato**

R1. Definition

Introduce the clauses

$$\delta_1 : \quad newp(X_1, ..., X_h) \leftarrow c_1 \wedge G_1$$
$$\ldots$$
$$\delta_m : \quad newp(X_1, ..., X_h) \leftarrow c_m \wedge G_m$$

where

- **newp is a new predicate symbol**
- $\{X_1, ..., X_h\} \subseteq FV(c_1 \wedge G_1, ..., c_m \wedge G_m)$
- **the predicates occurring in $G_1, ..., G_m$ occur also in P_0**

R2. Unfolding

By unfolding $\gamma: \; H \leftarrow c \wedge A \wedge G$

w.r.t. A we get the clauses

$$\Gamma: \; \{ H \leftarrow \overbrace{c \wedge A = H_1 \wedge c_1}^{\text{satisfiable}} \wedge G_1 \wedge G$$

for every clause $H_1 \leftarrow c_1 \wedge G_1$ **in** P_k $\}$

R3. Folding

Given the clauses

$$\gamma_1 : \; H \leftarrow c \wedge c_1\vartheta \wedge G_1\vartheta \wedge G$$
$$\ldots$$
$$\gamma_m : \; H \leftarrow c \wedge c_m\vartheta \wedge G_m\vartheta \wedge G$$

and the definitions

$$\delta_1 : \; newp(X_1, ..., X_h) \leftarrow c_1 \wedge G_1$$
$$\ldots$$
$$\delta_m : \; newp(X_1, ..., X_h) \leftarrow c_m \wedge G_m$$

by folding γ_1, ..., γ_m we get the clause

$$\eta : \; H \leftarrow c \wedge newp(X_1, ..., X_h)\,\vartheta \wedge G$$

R4. Clause Removal

- **unsatisfiable constraint**

$D \models \neg\exists(c)$

γ : $H \leftarrow c \wedge G$

- **subsumed clause**

$D \models \forall(c_1 \rightarrow \exists Z c_2)$

with $Z = FV(c_2) - vars(H, G_2)$

γ : $(H \leftarrow c_1 \wedge G_1 \wedge G_2)\vartheta$

δ : $H \leftarrow c_2 \wedge G_2$

R5. Constraint Replacement

A clause $\gamma_1 : \; H \leftarrow c_1 \wedge G$

A new clause $\gamma_2 : \; H \leftarrow c_2 \wedge G$

if $D \models \forall(\exists Y c_1 \leftrightarrow \exists Z c_2)$

with $Y = FV(c_1) - vars(H,G)$ and $Z = FV(c_2) - vars(H,G)$

R6-R7. Clause Splitting and Clause Fusion

One clause

γ : $H \leftarrow (c \vee d) \wedge G$

splitting ↓ ↑ fusion

Two clauses

γ_1 : $H \leftarrow c \wedge G$

γ_2 : $H \leftarrow d \wedge G$

In particular:

true $\wedge$ ...

↓ splitting

$c \wedge$ ...

$\neg c \wedge$ ...

Correctness of the Rules

Let $P_0, ..., P_n$ be transformation sequence.

$Defs_n$: all definitions introduced until P_n.

If "conditions of unfolding before folding or non-recursion introduction"

then $M(P_0 \cup Defs_n) = M(P_n)$

Determinization Strategy

Given a program P and a constrained atom $c \wedge p(t_1, ..., t_h)$,

Initial definition δ_{sp} : $p_{sp}(X_1, ..., X_r) \leftarrow c \wedge p(t_1, ..., t_h)$

$Defs := \{\delta_{sp}\}$; $Cls := \{\delta_{sp}\}$; $P_{sp} := \{\}$;

while $Cls \neq \{\}$ **do**

(1) **Unfold-Simplify**: Cls --> UnfCls

(2) **Partition** : UnfCls are partitioned into mutually exclusive packets
--> PartCls

(3) **Define-Fold** :
packets in PartCls are folded using new definitions
NewDefs --> FldCls
clauses with constraints only in their bodies --> CFacts

(4) $Defs := Defs \cup NewDefs$; $Cls := NewDefs$; $P_{sp} := P_{sp} \cup CFacts \cup FldCls$

Specializing Leq_Match 1

leq_match(P,S) ← append(B,C,S) ∧ append(A,Q,B) ∧ leq(P,Q)
append([ ], Ys, Ys) ←
append([X|Xs], Ys, [X|Zs]) ← append(Xs, Ys, Zs)
leq([ ], [ ]) ←
leq([X|Xs], [Y|Ys]) ← X ≤ Y ∧ leq(Xs,Ys)

6. $leq_match_{sp}(S) \leftarrow leq_match([1,0,2],S)$

Unfold-Simplify clause 6 :

7. $leq_match_{sp}(S) \leftarrow append(B,C,S) \wedge append(A,Q,B) \wedge leq([1,0,2],Q)$

This clause is a packet in itself. Define :

8. $new1(S) \leftarrow append(B,C,S) \wedge append(A,Q,B) \wedge leq([1,0,2],Q)$

Fold clause 7 :

9. $leq_match_{sp}(S) \leftarrow new1(S)$

Specializing Leq_Match 2

Unfold-Simplify clause 8 :

10. new1([X|Xs]) ← 1≤X ∧ append(Q,C,Xs) ∧ leq([0,2],Q)
11. new1([X|Xs]) ← append(B,C,Xs) ∧ append(A,Q,B) ∧ leq([1,0,2],Q)

Clauses 10 and 11 are not mutually exclusive. Clause Splitting of clause 11 :

12. new1([X|Xs]) ←1≤X∧append(B,C,Xs)∧append(A,Q,B)∧leq([1,0,2],Q)
13. new1([X|Xs]) ←1>X∧append(B,C,Xs)∧append(A,Q,B)∧leq([1,0,2],Q)

Two packets of clauses : 10, 12 **and** 13

Specializing Leq_Match 3

Packet (10, 12) :

10. new1([X|Xs])←1≤X∧append(Q,C,Xs)∧leq([0,2],Q)
12. new1([X|Xs])←1≤X∧append(B,C,Xs)∧append(A,Q,B)∧leq([1,0,2],Q)

Folding packet (10, 12) using Definition {14, 15} :

14. new2(Xs) ← append(Q,C,Xs) ∧ leq([0,2],Q)
15. new2(Xs) ← append(B,C,Xs) ∧ append(A,Q,B) ∧ leq([1,0,2],Q)

We get :

16. new1([X|Xs]) ← 1≤X ∧ new2(Xs)

Folding packet (13) using Definition 8 we get :

17. new1([X|Xs]) ← 1>X ∧ new1(Xs)

•••

Leq_Match 2

by specializing w.r.t. the goal leq_match([1,0,2], S)
we get the deterministic program:

9. $leq_match_{sp}(S) \leftarrow new1(S)$
16. $new1([X|Xs]) \leftarrow 1 \leq X \wedge new2(Xs)$
17. $new1([X|Xs]) \leftarrow 1 > X \wedge new1(Xs)$

$new2([X|Xs]) \leftarrow 1 \leq X \wedge new3(Xs)$
$new2([X|Xs]) \leftarrow 0 \leq X \wedge 1 > X \wedge new4(Xs)$
$new2([X|Xs]) \leftarrow 0 > X \wedge new1(Xs)$
$new3([X|Xs]) \leftarrow 2 \leq X \wedge new5(Xs)$
$new3([X|Xs]) \leftarrow 1 \leq X \wedge 2 > X \wedge new3(Xs)$
$new3([X|Xs]) \leftarrow 0 \leq X \wedge 1 > X \wedge new4(Xs)$
$new3([X|Xs]) \leftarrow 0 > X \wedge new1(Xs)$
$new4([X|Xs]) \leftarrow 2 \leq X \wedge new6(Xs)$
$new4([X|Xs]) \leftarrow 1 \leq X \wedge 2 > X \wedge new2(Xs)$
$new4([X|Xs]) \leftarrow 1 > X \wedge new1(Xs)$
$new5(Xs) \leftarrow$
$new6(Xs) \leftarrow$

Program Verification

Given a locally stratified CLP(*D*) program P and

a first order formula φ

Check whether or not $M(P) \models \varphi$

---

1. Apply Lloyd-Topor transformation starting from :

$f \leftarrow \varphi$

and derive a locally stratified program Cls(f,φ)

s. t. $M(P) \not\models \varphi$ iff $M(P \cup Cls(f,\varphi)) \not\models f$

2. Apply rules according to a strategy: $P \cup Cls(f,\varphi) \xrightarrow{\text{rules}}{}^{*} T$

s.t. *either* $f \leftarrow$ is in T (in which case $M(P) \not\models \varphi$)

or no clause for f is in T (in which case $M(P) \models \varphi$)

[CL 2000]

Verifying a Semaphore 1

Program P :

1. down(X1) ← X1=X+1 ∧ ¬ down(X)
2. up(0, 0) ←
3. up(X1, 0) ← X1=X+1 ∧ down(X)
4. up(X1, Y1) ← X1=X+1 ∧ Y1=Y+1 ∧ X>Y ∧ up(X1,Y)

φ : ∀X,Y (X>Y ∧ X2=X+2 ∧ up(X,Y) → up(X2, 0))

up	0,0		2,0	2,1		4,0	4,1	4,2	4,3		6,0	...	6,5		8,0	...		
down		1			3					5				7		...		

We start from :

f ← ∀X,Y (X>Y ∧ X2=X+2 ∧ up(X,Y) → up(X2, 0))

After Lloyd-Topor we get :

6. f ← ¬ g
7. g ← X>Y ∧ X2=X+2 ∧ up(X,Y) ∧ ¬ up(X2, 0)

Verifying a Semaphore 2

During the transformation we introduce the definitions :

$h \leftarrow X>Y \wedge X1=X+1 \wedge X3=X+3 \wedge up(X1,Y) \wedge \neg up(X3, 0)$

$k(N) \leftarrow X>Z \wedge X1=X+1 \wedge X3=X+3 \wedge Z=Y+N \wedge up(X1, Y) \wedge \neg up(X3, 0)$

We get the program :

6. $f \leftarrow \neg g$
7. $g \leftarrow h$
8. $h \leftarrow k(1)$
9. $k(N) \leftarrow N1=N+1 \wedge k(N1)$

k(N) has no base cases.

In the perfect *D*-model k(N) is false.

Unfolding 8 we get no clause for h : h is false.

Unfolding 7 we get no clause for g : g is false.

Unfolding 6 we get :

6*. $f \leftarrow$ **Thus, M(P) $\models \varphi$**

Program Synthesis

Given a locally stratified CLP(*D*) program P and

a first order formula φ[X]

Derive an efficient program S defining s(X) s.t.

∀ ground term t

M(P) ⊨φ[t] iff M(S) s(t)

---

1. Apply Lloyd-Topor transformation starting from :

$s(X) \leftarrow \varphi[X]$

and derive a possibly inefficient, locally stratified program Cls(s,φ)
s.t. ∀ ground term t, M(P) φ[t]⊨ iff M(P ∪ Cls(s,φ)) s(t) ⊨

2. Apply rules according to a strategy: $P \cup Cls(s,\varphi) \xrightarrow{\text{rules}}{}^{*} S$

[Kowalski's Festschrift 2002]

Synthesis of max 1

Program P :

1. list([ ]) ←
2. list([X|Xs]) ← list(Xs)
3. member(X, [A|As]) ← X = A
4. member(X, [A|As]) ← member(X,As)

φ[L,M] : list(L) ∧ member(M,L) ∧ ∀X (member(X,L) → X≤M)

We start from :

max(L,M) ← list(L) ∧ member(M,L) ∧∀X(member(X,L) → X≤M)

After Lloyd-Topor we get :

6. max(L,M) ← list(L) ∧ member(M,L) ∧ ¬ new1(L,M)
7. new1(L,M) ← member(X,L) ∧ ¬ X≤M

Synthesis of max 2

Synthesized Program :

16. $\max([A|As], M) \leftarrow new2(A, As, M)$
17. $new2(A, [\,], M) \leftarrow M = A$
21. $new2(A, [B|As], M) \leftarrow B \leq A \wedge new2(A, As, M)$
22. $new2(A, [B|As], M) \leftarrow A \leq B \wedge new2(B, As, M)$

Future Work

Specialization:

- Experimenting with other classes of algorithms : scheduling
- Adding a generalization technique to get termination for general classes of programs

Verification and Synthesis:

- Experimenting with other protocols and properties (or specifications)
- Looking for complete strategies w.r.t. some given classes of programs

References

AGP 2002

CL 2000 : LNAI 1861

Fioravanti's Thesis 2002

Kowalski Festschrift 2002 : LNAI 2407

Lopstr 2000 : LNCS 2042

See also :

`http://www.iasi.rm.cnr.it/~proietti`

MAP Transformation System available at :

`http://www.iasi.rm.cnr.it/~proietti`