GOAL GENERALIZATION FOR LOGIC PROGRAM TRANSFORMATION

Workshop Progetto Cofinanziato
"Tecniche formali per la specifica,
l'analisi, la verifica, la sintesi
e la trasformazione di sistemi software"

Roma, 21-23 Dicembre 1998

Alberto Pettorossi
University of Roma Tor Vergata
Roma, Italy

Maurizio Proietti
IASI CNR
Roma, Italy

{adp,proietti}@iasi.rm.cnr.it
http://www.iasi.rm.cnr.it/~{adp,proietti}

PLAN OF THE TALK

Background:
- Rules for logic program transformation.
- Preservation of universal termination.
- Goal rearrangement does not preserve universal termination.

Avoiding unsafe goal rearrangements by goal generalization.

Logic Programming with "goals as arguments".

Transformation rules for LP with "goals as arguments".

Preservation of universal termination.

PROGRAM TRANSFORMATION VIA RULES AND STRATEGIES

$P_0 \xrightarrow{R1} P_1 \xrightarrow{R2} P_2 \ldots \xrightarrow{Rk} P_k$: a transformation sequence

every Ri is an application of a transformation *rule*

every rule application is semantics preserving

- usually a small, local replacement
- many steps to get the final program
- need for *strategies* to get efficiency improvements

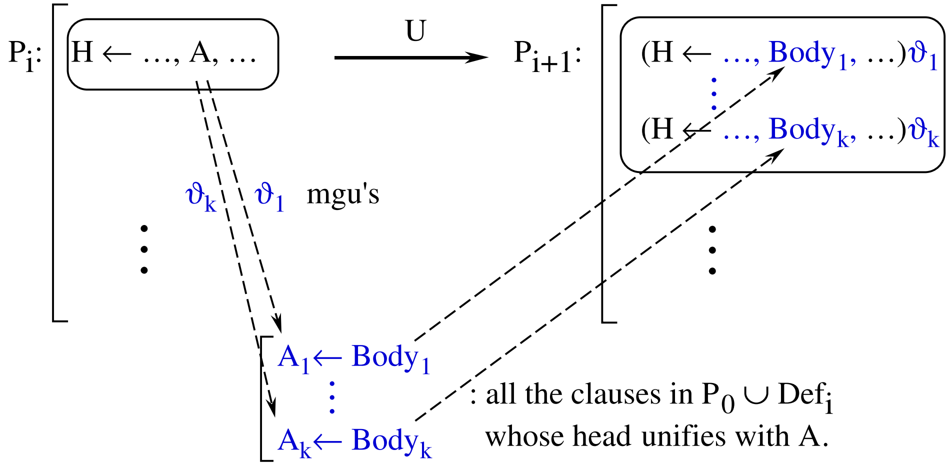
DEFINITION INTRODUCTION

$$P_i \xrightarrow{DI} P_{i+1} = P_i \cup \{ newp(\ldots) \leftarrow Body \}$$

if newp is a new predicate symbol, neither in $P_0, \ldots, P_i$ nor in Body

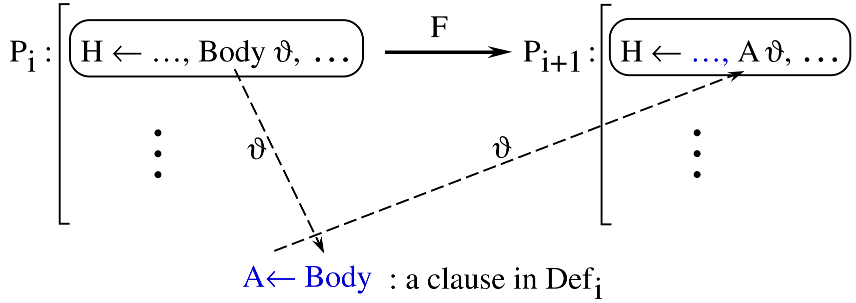
- Def_i = Definitions introduced up to step i

UNFOLDING



For k = 0, unfolding is *clause deletion*.

FOLDING



if "extra conditions" on ϑ hold.

---

by requiring that (folding ; unfolding) = id the "extra conditions" on ϑ hold.

GOAL REARRANGEMENT

(Commutativity of conjunction)

$$P_i : \left[\begin{array}{l} H \leftarrow \ldots, A, B, \ldots \\ \vdots \end{array} \right. \xrightarrow{C} P_{i+1} : \left[\begin{array}{l} H \leftarrow \ldots, B, A, \ldots \\ \vdots \end{array} \right.$$

CORRECTNESS OF THE RULES

$$P_0 \xrightarrow{R1} P_1 \xrightarrow{R2} P_2 \ldots \xrightarrow{Rk} P_k$$

- *Correctness of the transformation rules wrt the least Herbrand model* [Tamaki-Sato 84]. Suppose that:
  - Ri $\in$ {DI, U, F, C} and
  - if an Ri is "folding of clause c using clause d", then an Rj is "unfolding of clause d".

  Then $M(P_0 \cup Def_k) = M(P_k)$.

- *Universal termination (* $\forall$-termination*) of goal G in program P:* Every maximal SLD-derivation starting from G and using the left-to-right selection rule is finite (ending with a *success* or a *failure*).

  Preservation of $\forall$*-termination* [P&P 91, Bossi-Cocco 94]. Suppose that:
  - Ri $\in$ {DI, U, F} and
  - if an Ri is "folding of clause c using clause d", then an Rj is "unfolding of clause d wrt the leftmost atom in its body".

  Then for every goal G, if G $\forall$-terminates in $P_0 \cup Def_k$ then G $\forall$-terminates in P_k.

- The goal rearrangement rule does *not* preserve $\forall$-termination.

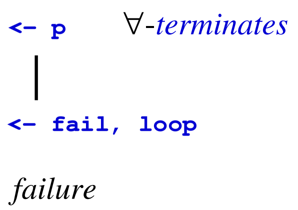
EXAMPLES OF GOAL REARRANGEMENTS

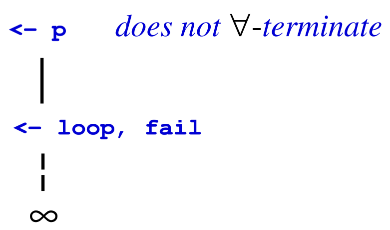
P_i :

```
p <- fail, loop
loop <- loop
```

$\xrightarrow{C}$ P_{i+1} :

```
p <- loop, fail
loop <- loop
```



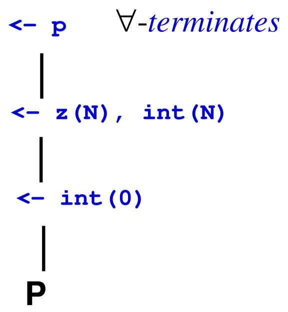


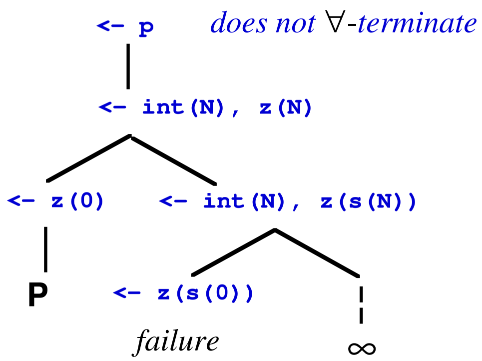
P_i :

```
p <- z(N), int(N)
z(0) <-
int(0) <-
int(s(N)) <- int(N)
```

$\xrightarrow{C}$ P_{i+1} :

```
p <- int(N), z(N)
⋮
```





APPROACHES TO THE GOAL REARRANGEMENT PROBLEM

1. [Pettorossi-Proietti 91, Jørgensen-Leuschel-Martens 96]: *Never* apply goal rearrangement.

   ☺ ∀-termination is preserved.

   ☹ Many useful transformations are prevented.

2. [Bossi-Cocco-Etalle 95]: Apply goal rearrangement with conditions based on *modes* and *failure* information.

   ☺ ∀-termination is preserved.

   ☺ Many useful transformations are allowed.

   ☹ Many useful transformations are still prevented.

   ☹ Undecidable conditions (decidable sufficient conditions further restrict the applicability of the method).

3. This talk: Avoid unsafe goal rearrangements by using a *higher-order extension* of LP.

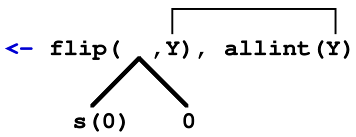
   ☺ ∀-termination is preserved.

   ☺ No restriction based on modes or failure information.

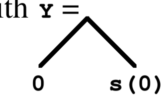
EXAMPLE: TWO-PASS TREE PROCESSING

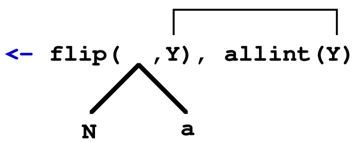
INITIAL PROGRAM P_0

```
flip(leaf(N),leaf(N)) <-
flip(tree(L,N,R),tree(FR,N,FL)) <- flip(L,FL), flip(R,FR)

allint(leaf(N)) <- int(N)
allint(tree(L,N,R)) <- int(N), allint(L), allint(R)

int(0) <-
int(s(N)) <- int(N)
```



succeeds with `Y` =





fails

AVOIDING MULTIPLE TREE TRAVERSALS

Automatic strategy in [P&P 95]

<u>Definition Introduction</u>

P_1: `fa(X,Y) <- flip(X,Y), allint(Y)` (D_1)

<u>Unfolding</u> (three times)

P_2:
```
fa(leaf(N),leaf(N)) <- int(N)
fa(tree(L,N,R),tree(FR,N,FL)) <-
        flip(L,FL), flip(R,FR),
        int(N), allint(FR), allint(FL)
```

<u>Goal Rearrangement</u> (*unsafe*)

P_3:
```
fa(tree(L,N,R),tree(FR,N,FL)) <-
        int(N), flip(L,FL), allint(FL),
        flip(R,FR), allint(FR)
```

<u>Folding</u> (twice)

P_4:
```
fa(tree(L,N,R),tree(FR,N,FL)) <-
        int(N), fa(L,FL), fa(R,FR)
```

ONE-PASS TREE PROCESSING BY UNSAFE GOAL REARRANGEMENT

FINAL PROGRAM P_4

```
fa(leaf(N),leaf(N)) <- int(N)
fa(tree(L,N,R),tree(FR,N,FL)) <- int(N), fa(L,FL), fa(R,FR)
```

☺ One tree traversal only

☺ $M(P_0 \cup \{D_1\}) = M(P_4)$.

☹ $\forall$-termination *not* preserved:

`<- fa(tree(leaf(N),0,leaf(a)),Y)` *fails* in P_0 *loops* in P_4

AVOIDING UNSAFE REARRANGEMENTS

Definition Introduction

P_1: `fa(X,Y) <- flip(X,Y), allint(Y)` (D_1)

Unfolding (three times)

...

P_2:
```
fa(tree(L,N,R),tree(FR,N,FL)) <-
        flip(L,FL), flip(R,FR),
        int(N), allint(FR), allint(FL)
```

Goal Generalization + Equality Intro.

P_3:
```
fa(tree(L,N,R),tree(FR,N,FL)) <-
        flip(L,FL), flip(R,FR),
        int(N), P = allint(FR), Q = allint(FL), P, Q
```

where `=` is defined by the clause: `G = G <- .`

Safe Goal Rearrangement

P_4:
```
fa(tree(L,N,R),tree(FR,N,FL)) <-
        flip(L,FL), Q = allint(FL),
        flip(R,FR), P = allint(FR), int(N), P, Q
```

☺ This rearrangement preserves $\forall$-termination.
Proof: by U wrt `=` we get back to P_2, and U preserves $\forall$-termination.

☹ Folding using D_1 is impossible.

GOAL GENERALIZATION

Definition Introduction

P_5:
```
g(X,Y,G) <- flip(X,Y), G = allint(Y)
```
(D_2)

Unfolding *leftmost*

P_6:
```
g(leaf(N),leaf(N),G) <- G = allint(leaf(N))
g(tree(L,N,R),tree(FR,N,FL),G) <-
          flip(L,FL), flip(R,FR),
          G = allint(tree(FR,N,FL))
```

Argument Unfolding (twice)

P_7:
```
g(leaf(N),leaf(N),G) <- G = int(N)
g(tree(L,N,R),tree(FR,N,FL),G) <-
          flip(L,FL), flip(R,FR),
          G = (int(N), allint(FR), allint(FL))
```

Goal Generalization + Eq.Intro. ; *Rearrangement of* =

P_8:
```
g(tree(L,N,R),tree(FR,N,FL),G) <-
          flip(L,FL), Q = allint(FL),
          flip(R,FR), P = allint(FR), G = (int(N),P,Q)
```

Folding using D_2

P_9:
```
g(tree(L,N,R),tree(FR,N,FL),G) <-
          g(L,FL,Q), g(R,FR,P), G = (int(N),P,Q)
```


ONE-PASS TREE PROCESSING WITH GOALS AS ARGUMENTS

Expressing `fa` in terms of `g` by GA ; F:
Goal Generalization + Eq. Intro.

P_{10}:
```
fa(X,Y) <- flip(X,Y), G = allint(Y), G
```

Folding

P_{11}:
```
fa(X,Y) <- g(X,Y,G), G
```

Simplification of `=` by Unfolding

FINAL PROGRAM P_{12}:

```
fa(X,Y) <- g(X,Y,G), G
g(leaf(N),leaf(N),int(N)) <-
g(tree(L,N,R),tree(FR,N,FL),(int(N), P, Q)) <-
          g(L,FL,Q), g(R,FR,P)
```

☺ One tree traversal only.

☺ If `<- fa(t,u)` $\forall$-terminates in $P_0 \cup \{D_1\}$
then `<- fa(t,u)` $\forall$-terminates in P_{12}.

☺ `<- fa(t,u)` succeeds in $P_0 \cup \{D_1\}$
iff `<- fa(t,u)` succeeds in P_{12}.

A HIGHER-ORDER LOGIC PROGRAMMING LANGUAGE

`x` ∈ Ivar	Individual variables
`f` ∈ Fun	Function symbols
`p` ∈ Pred	Predicate symbols
`G` ∈ Gvar	Goal variables
`t` ::= `x` | `f(t1,…,tk)`	Terms
`g` ::= `G` | `true` | `false` | `t1 = t2` | `G = g` | `p(t1,…,tm,g1,…,gn)` | `g1, g2` | `g1; g2`	Goals
`c` ::= `p(t1,…,tm,g1,…,gn) <- g`	Clauses

Programs ::= sets of clauses.

The language is typed: *term* occurrences can be distinguished from *goal* occurrences.

<u>Operational semantics:</u>
SLD-resolution with left-to-right selection rule augmented with rules for
- `true` : succeeds
- `false` : fails
- `t1 = t2` : syntactically unifies terms `t1` and `t2`
- `g1 = g2` : syntactically unifies goals `g1` and `g2`
- `g1; g2` : solves `g1` or `g2`
- `G` : stops with error

TRANSFORMATION RULES FOR HIGHER-ORDER LOGIC PROGRAMS

DEFINITION INTRODUCTION
UNFOLDING : *possibly non outermost* occurrence
FOLDING : *possibly non outermost* occurrence
RULES for `=` of terms
of goals (*goal geneneralization* + *eq. intro.*
eq. simplification)

RULES for `true` `false` `,` `;`

DEFINITION INTRODUCTION

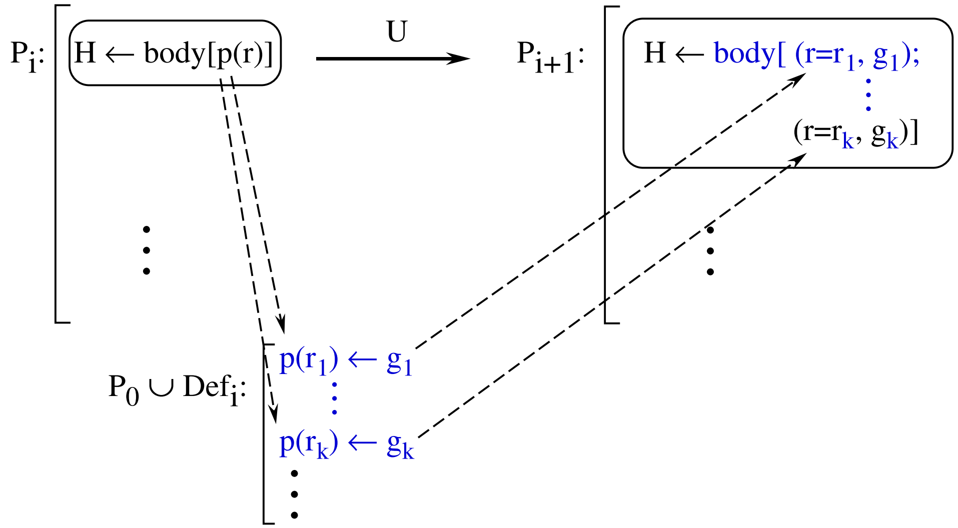
$$P_i \xrightarrow{DI} P_{i+1} = P_i \cup \{\ newp(\ldots) \leftarrow g\ \}$$

if newp is a new predicate symbol, neither in $P_0, \ldots, P_i$ nor in goal g.

- Def_i = Definitions introduced up to step i

UNFOLDING

- *Outermost non-variable* occurrence : unchanged
- *Non-outermost non-variable* occurrence p(r)



In particular, if for all $j \in \{1,\ldots,k\}$ the equality $r=r_j$ is false:

$$P_{i+1}: \left[\ \boxed{H \leftarrow body[\ false]}\ \ \vdots \right.$$

FOLDING

$P_i : \left[\begin{array}{l} H \leftarrow \text{body}[g\,\vartheta] \\ \vdots \end{array}\right] \xrightarrow{F} P_{i+1} : \left[\begin{array}{l} H \leftarrow \text{body}[A\,\vartheta] \\ \vdots \end{array}\right]$

ϑ ϑ

$A \leftarrow g$: a clause in Def_i

if "extra conditions" on ϑ hold.

---

by requiring that (folding ; unfolding) = id the "extra conditions" on ϑ hold.

RULES FOR EQUALITY

$P_i : \left[\begin{array}{l} H \leftarrow \text{body}[g1] \\ \vdots \end{array}\right] \xrightarrow{E} P_{i+1} : \left[\begin{array}{l} H \leftarrow \text{body}[g2] \\ \vdots \end{array}\right]$

if g1 $\longrightarrow$ g2 (and extra conditions on variables hold)

where g1 and g2 are (i) equivalent wrt Clark's Equality Theory (extended with goal equality) and (ii) g2 $\forall$-terminates more often than g1.

Examples:

GOAL GENERALIZATION + EQ. INTRO.

g2[g1] $\longrightarrow$ (G = g1), g2[G] G not occurring in g2[g1]

SIMPLIFICATION OF EQUALITIES

(G = g1), g2[G] $\longrightarrow$ g2[g1] G not occurring in g2[g1]

REARRANGEMENT OF EQUALITIES

g[(G = g1), g2] $\longrightarrow$ (G = g1), g[g2] G not occurring in g[ ]

(G = g1), g[g2] $\longrightarrow$ g[(G = g1), g2] G not occurring in g[ ]

g2[g1] $\not\longrightarrow$ g2[G], (G = g1)

RULES FOR TAUTOLOGIES

$$P_i : \left[\begin{array}{l} H \leftarrow \text{body}[g1] \\ \vdots \end{array}\right. \xrightarrow{T} P_{i+1} : \left[\begin{array}{l} H \leftarrow \text{body}[g2] \\ \vdots \end{array}\right.$$

if g1 $\longrightarrow$ g2   (and extra conditions on variables hold)

where (i) g1 $\longleftrightarrow$ g2 is a tautology and (ii) g2 $\forall$-terminates more often than g1.

Examples:
g, false $\longrightarrow$ false
g1; g2 $\longrightarrow$ g2; g1

false $\not\longrightarrow$ g, false
g1, g2 $\not\longrightarrow$ g2, g1

INCORRECTNESS OF THE RULES

The transformation rules are *incorrect*:

```
p(G) <- G = q                  <- p(q)
```
succeeds

```
q <- r
```

Unfolding

```
p(G) <- G = r                  <- p(q)
```
fails

- The incorrectness of the rules is due to the *incompleteness* of syntactic unification of goals. Syntactic unification of equivalent goals may fail.
- In order to have correct rules we restrict to programs that unify two goals g1 and g2 only if either g1 is a variable or g2 is a variable.

MODES FOR GOAL ARGUMENTS

- We introduce input/output MODES for goal arguments (e.g. [Chandha-Plaisted 91]):

  p(t,ig,og)

  first argument position(s) : terms
  second argument position(s) : *input* goals
  third argument position(s) : *output* goals

- Mode for goal equality: og = ig

- *Goals* have the form*:

  $$p_1(t_1,g_1,G_1), \ldots, p_n(t_n,g_n,G_n) \qquad (\alpha)$$

  where:
  - $G_1, \ldots, G_n$ are distinct goal variables
  - for j = 1,…,n, $Var(g_j) \cap \{G_j, \ldots, G_n\} = \emptyset$
    (a variable occurring in an output position does not occur earlier in the goal)

- *Clauses* are of the form*:

  $$p_0(t_0,G_0,g_0) \leftarrow p_1(t_1,g_1,G_1), \ldots, p_n(t_n,g_n,G_n) \qquad (\beta)$$

  where G_0 is a goal variable not among $G_1, \ldots, G_n$.

---

* outermost goal variables G are rewritten as call(G)

CORRECTNESS OF THE RULES

$$P_0 \xrightarrow{R1} P_1 \xrightarrow{R2} P_2 \ldots \xrightarrow{Rk} P_k$$

Suppose that:
- All clauses in $P_0 \cup Def_k$ are of the form (α),
- $Ri \in \{DI, U, F, E, T\}$ and
- if an Ri is "folding of clause c using clause d", then an Rj is "unfolding of clause d <u>wrt the *leftmost outermost* atom in its body</u>".

Then for every goal G of the form (α),
- if G $\forall$-terminates in $P_0 \cup Def_k$ then G $\forall$-terminates in P_k
- G succeeds in $P_0 \cup Def_k$ iff G succeeds in P_k.

FINAL REMARKS

- Goal generalization is *complementary* to the approach in [Bossi-Cocco-Etalle]. Sometimes goal rearrangement may improve program efficiency.

- Strategies: when and how to apply the transformation rules.

- Related strategies in functional programming [Pettorossi-Skowron 87] : the introduction of higher-order functions is guided by the *need for folding* and the *mismatch* information (*higher-order generalization*).