Proving Properties of Constraint Logic Programs by Eliminating Existential Variables

Alberto Pettorossi (Univ. Tor Vergata, Rome, Italy),

Maurizio Proietti (IASI-CNR, Rome, Italy),

Valerio Senni (Univ. Tor Vergata, Rome, Italy)

ICLP 2006 – Seattle (Wa), USA **August 17-20, 2006**

Theorem Proving via Program Transformation

- Transformation techniques for ATP:
  - proving equivalences via unfold/fold **[Kott-82, PP-99, Roychoudhury et al-99]**
  - proving first order formulas via unfold/fold **[PP-00]**
  - verifying temporal properties of infinite state systems via specialization and u/f **[Leuschel 99,00, Roychoudhury et al-00, Fioravanti et al-01]**

- This work: techniques for the elimination of intermediate data structures (*deforestation*) can be used as methods for quantifier elimination.

- Intermediate data structures can be avoided by eliminating *existential* (or *local*, or *redundant*) variables, i.e., variables *occurring in the body* of a clause and *not in the head*. For instance, X is an existential variable in

$$p \leftarrow q(X) \wedge r(X)$$

[Recall: $\forall X\ (p \leftarrow q(X) \wedge r(X)) \equiv p \leftarrow \exists X\ (q(X) \wedge r(X))$]

The Quantifier Elimination Method

- ***QE***: Given a theory T and a formula φ, find a quantifier-free formula ψ s.t.
  - freevars(ψ) $\subseteq$ freevars(φ)
  - $T \models \varphi$ iff $T \models \psi$

- If ***QE*** is ***algorithmic*** and the problem of checking $T \models A$, for any ground atom A, is ***decidable***, then the problem of checking $T \models \varphi$, for any closed formula φ, is ***decidable***.

- The theory $\boldsymbol{R}$ of real numbers (as an ordered field) admits ***QE*** and for any ground atom A (*i.e.*, ground real arithmetic expression), $\boldsymbol{R} \models A$ can be decided by direct evaluation.
  Thus, $\boldsymbol{R} \models \varphi$ is decidable [Tarski 48].

Quantifier Elimination for Linear Constraints

- For the first order theory $\boldsymbol{R}_{lin}$ of linear equations/inequations over the real numbers, ***QE*** makes use of Fourier-Motzkin elimination procedure.

- Example.

$\forall x\ (x \geq 1 \rightarrow \exists y\ (x - y > 0 \wedge y > 0))$	
$\neg \exists x\ (x \geq 1 \wedge \neg \exists y\ (x - y > 0 \wedge y > 0))$	$[\forall x\ (\alpha \rightarrow \beta) \equiv \neg \exists x\ (\alpha \wedge \neg \beta)]$
$\neg \exists x\ (x \geq 1 \wedge \neg x > 0)$	[F-M: elimination of y]
$\neg \exists x\ (x \geq 1 \wedge 0 \geq x)$	[negation of $>$]
$\neg 0 \geq 1$	[F-M: elimination of x]
True	[evaluation of ground formula]

QE for CLP(R_{lin}) via Program Transformation

- We are given a CLP($\boldsymbol{R}_{lin}$) program

$$P:\quad \text{member}\ (X,[Y|L]) \leftarrow X=Y$$
$$\qquad \text{member}\ (X,[Y|L]) \leftarrow \text{member}\ (X,L)$$

and a closed first order formula

$$\varphi: \quad \forall L\ \exists U\ \forall X\ (\ \text{member}(X,L) \rightarrow X \leq U\ )$$

- **Step1.** By a variant of Lloyd-Topor transformation we get a *clause form* *CF* for φ s.t. $M(P) \models \varphi$ iff $M(P \cup CF) \models f$, where $M(\ )$ denotes the *perfect* $\boldsymbol{R}_{lin}$*-model*.

$$CF: \quad f \leftarrow \neg p$$
$$p \leftarrow \text{list}(L) \wedge \neg q(L)$$
$$q(L) \leftarrow \text{list}(L) \wedge \neg r(L,U)$$
$$r(L,U) \leftarrow X > U \wedge \text{list}(L) \wedge \text{member}(X,L)$$

...QE for CLP(R_{lin}) via Program Transformation

- **Step 2.** By *unfold/fold transformations and quantifier elimination from constraints in* $\boldsymbol{R}_{lin}$, from $P \cup CF$ we derive a *propositional* program *Prop* s.t. $M(P \cup CF) \models$ f iff $M(Prop) \models$ f.

$$\begin{array}{lll} Prop: & \mathrm{f} & \leftarrow \neg \mathrm{p} \\ & \mathrm{p} & \leftarrow \mathrm{p1} \\ & \mathrm{p1} & \leftarrow \mathrm{p1} \end{array}$$

- Thus, $M(P) \models \varphi$ iff $M(Prop) \models$ f.

- The transformation from $P \cup CF$ to *Prop* consists in eliminating all *existential variables* from *CF*.

Overview

- *LR-programs*: A class of CLP programs on lists of real numbers with linear constraints;

- *Clause form transformation* for first order formulas;

- *Unfold/fold* transformations of clause forms;

- An *automatic strategy* for deriving propositional programs by eliminating existential variables.

Programs on Lists of Real Numbers

- $a \in \mathbb{R}$, $X \in Var_{\mathbb{R}}$, $L \in Var_{List}$
- Polynomials: $p ::= a \mid X \mid p_1 + p_2 \mid a\,X$
- Constraints: $c ::= p_1 = p_2 \mid p_1 < p_2 \mid p_1 \leq p_2 \mid c_1 \wedge c_2$
- LR-programs:
  - head terms $h ::= X \mid [\,] \mid [X|L]$
  - body terms $b ::= p \mid L$
  - clauses $cl ::= r_1(h_1,\ldots,h_n) \leftarrow c \mid$

    $r_1(h_1,\ldots,h_n) \leftarrow c \wedge r_2(b_1,\ldots,b_n) \mid$

    $r_1(h_1,\ldots,h_n) \leftarrow c \wedge \neg\, r_2(b_1,\ldots,b_n)$

where: (i) *cl* has *no existential variables*, (ii) $r_1(h_1,\ldots,h_n)$ is a *linear* atom, and (iii) $vars(p) \neq \emptyset$.

LR-Programs: Examples

- LR-programs:

  member(X,[Y|L]) $\leftarrow$ X=Y

  member(X,[Y|L]) $\leftarrow$ member(X,L)

  pos_sumlist([ ], Y) $\leftarrow$ Y=0

  pos_sumlist([X|L],Y) $\leftarrow$ X>0 $\wedge$ pos_sumlist(L,Y–X)

  pos_sumlist([X|L],Y) $\leftarrow$ X≤0 $\wedge$ pos_sumlist(L,Y)

- Not an LR-program:

  permutation([ ], [ ]) $\leftarrow$

  permutation([X|L1],L2) $\leftarrow$ permutation(L1,L3) $\wedge$ insert(X,L3,L2)

  - L2 is neither [ ] nor [X|L]
  - L3 is existential
  - the body has two literals

Properties of LR-Programs

- The problem of checking $M(P) \models \varphi$, for any LR-program P and closed formula φ, is ***undecidable***. Peano arithmetic can be encoded via an LR-program.
- The transformation from $P \cup CF$ to *Prop* cannot be algorithmic.
- If $P \cup CF$ is transformed into an LR-program T, then the 0-ary predicate f is defined by a set $Prop \subseteq T$ of propositional clauses. Thus, quantifiers can be eliminated by deriving LR-programs.

Clause-Form Transformation

$$\varphi: \quad \forall L\, \exists U\, \forall X\, (\, member(X,L) \rightarrow X \leq U\,)$$

$$f \leftarrow \neg \underbrace{\exists L\, \neg \underbrace{\exists U\, \neg \underbrace{\exists X\, (\, member(X,L) \wedge \neg X \leq U\,)}_{r}}_{q}}_{p}$$

- Lloyd-Topor transformation + addition of a list(L) atom for each list variable L (needed for unfolding).

CF:

$D4$: $f \leftarrow \neg\, p$

$D3$: $\boxed{p \leftarrow list(L) \wedge \neg\, q(L)}$

$D2$: $\boxed{q(L) \leftarrow list(L) \wedge \neg\, r(L,U)}$

$D1$: $\boxed{r(L,U) \leftarrow X > U \wedge list(L) \wedge member(X,L)}$

- stratified program
- *not* LR-clauses (with existential variables)

- *D*1,*D*2,*D*3,*D*4 will be transformed to LR-clauses by unfold/fold transformations.

Eliminating Existential Variables via U/F

*D*1: $r(L,U) \leftarrow X > U \wedge list(L) \wedge member(X,L)$

Eliminating Existential Variables via U/F

*D*1: $r(L,U) \leftarrow X > U \wedge \underline{list(L)} \wedge \underline{member(X,L)}$

Unfold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow Y > U \wedge list(T) \wedge member(Y,T)$

Eliminating Existential Variables via U/F

*D*1: $r(L,U) \leftarrow \boxed{X > U \wedge list(L) \wedge member(X,L)}$

Unfold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow \boxed{Y > U \wedge list(T) \wedge member(Y,T)}$

☺

Eliminating Existential Variables via U/F

*D*1: $r(L,U) \leftarrow \boxed{X > U \wedge list(L) \wedge member(X,L)}$

☺

Unfold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow \boxed{Y > U \wedge list(T) \wedge member(Y,T)}$

Fold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow \boxed{r(T,U)}$

Eliminating Existential Variables via U/F

*D*1: $r(L,U) \leftarrow \boxed{X > U \wedge list(L) \wedge member(X,L)}$

☺

Unfold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow \boxed{Y > U \wedge list(T) \wedge member(Y,T)}$

Fold: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow \boxed{r(T,U)}$

LR-clauses
(no existential variables)

Transformed program (1)

*D*4: $\mathrm{f} \leftarrow \neg\, \mathrm{p}$

*D*3: $\mathrm{p} \leftarrow \mathrm{list}(L) \wedge \neg\, \mathrm{q}(L)$

*D*2: $\mathrm{q}(L) \leftarrow \mathrm{list}(L) \wedge \neg\, \mathrm{r}(L,U)$

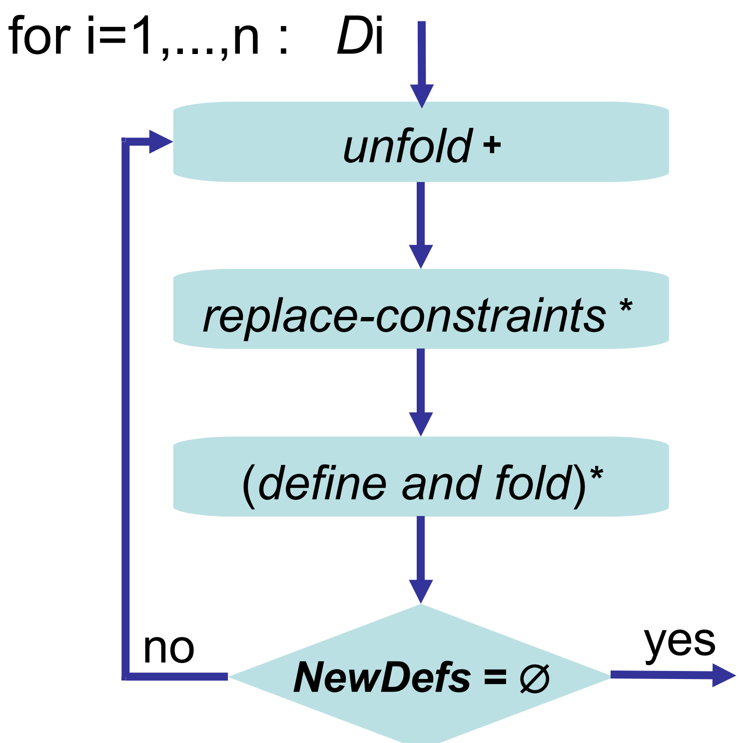
*D*1: $\mathrm{r}([X|T],U) \leftarrow X > U \wedge \mathrm{list}(T)$
$\mathrm{r}([X|T],U) \leftarrow \mathrm{r}(T,U)$

No existential variables

An Unfold-Fold Strategy for Deriving LR-programs

Input: an LR-program *P* and the clause form *CF: D1,...,Dn* of a closed first order formula φ

Output: an LR-program *T* s.t. f is defined by a propositional *Prop* $\subseteq$ *T* and *M(P)* $\models \varphi$ iff *M(Prop)* $\models$ f.



QE from constraints

Introduction of new predicates

Final program *T*

Introducing New Definitions

$D2$: $\quad q(L) \leftarrow list(L) \wedge \neg\, r(L,U)$

Introducing New Definitions

*D*2 : $q(L) \leftarrow \underline{list(L)} \wedge \neg\ \underline{r(L,U)}$

Unfold : $q([\,]) \leftarrow$

$q([X|T]) \leftarrow X \leq U \wedge list(T) \wedge \neg\ r(T,U)$

Introducing New Definitions

*D*2 : $q(L) \leftarrow \boxed{list(L) \wedge \neg\, r(L,U)}$

Unfold : $q([\,]) \leftarrow$

$q([X|T]) \leftarrow X \leq U \wedge \boxed{list(T) \wedge \neg\, r(T,U)}$

☹ Bad folding!
Existential variable
not eliminated.

Introducing New Definitions

*D*2 : $q(L) \leftarrow list(L) \wedge \neg\, r(L,U)$

Unfold : $q([\,]) \leftarrow$

$q([X|T]) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Define : $q1(X,T) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Introducing New Definitions

D2 : $q(L) \leftarrow list(L) \wedge \neg\, r(L,U)$

Unfold : $q([\,]) \leftarrow$

$q([X|T]) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Define : $q1(X,T) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Fold : $q([\,]) \leftarrow$

$q([X|T]) \leftarrow \boxed{q1(X,T)}$

LR-clauses ☺
(no existential variables)

Existential variables to be eliminated from the new definition

Transforming New Definitions

We transform the new definition into a set of LR-clauses

New Def.: $q1(X,T) \leftarrow X \leq U \wedge list(T) \wedge \neg\, r(T,U)$

Transforming New Definitions

We transform the new definition into a set of LR-clauses

New Def.: $q1(X,T) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Unfold: $q1(X,[\,]) \leftarrow$

$q1(X,[Y|T]) \leftarrow X \leq U \wedge \boxed{Y \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

☹

Bad folding!
Existential variable
not eliminated

Transforming New Definitions

We transform the new definition into a set of LR-clauses

New Def.: $q1(X,T) \leftarrow X \le U \wedge list(T) \wedge \neg\, r(T,U)$

Unfold: $q1(X,[\,]) \leftarrow$

$q1(X,[Y|T]) \leftarrow \boxed{X \le U \wedge Y \le U} \wedge list(T) \wedge \neg\, r(T,U)$

linear order

$\equiv \boxed{(X>Y \wedge X \le U) \vee (X \le Y \wedge Y \le U)}$

Replace-constraints:

$q1(X,[Y|T]) \leftarrow \boxed{X>Y \wedge X \le U} \wedge list(T) \wedge \neg\, r(T,U)$

$q1(X,[Y|T]) \leftarrow \boxed{X \le Y \wedge Y \le U} \wedge list(T) \wedge \neg\, r(T,U)$

Transforming New Definitions

We transform the new definition into a set of LR-clauses

New Def.: $q1(X,T) \leftarrow \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Unfold: $q1(X,[\,]) \leftarrow$

$q1(X,[Y|T]) \leftarrow X \leq U \wedge Y \leq U \wedge list(T) \wedge \neg\, r(T,U)$

☺

Replace-constraints:

$q1(X,[Y|T]) \leftarrow X > Y \wedge \boxed{X \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

$q1(X,[Y|T]) \leftarrow X \leq Y \wedge \boxed{Y \leq U \wedge list(T) \wedge \neg\, r(T,U)}$

Fold: $q1(X,[\,]) \leftarrow$

$q1(X,[Y|T]) \leftarrow X > Y \wedge \boxed{q1(X,T)}$

$q1(X,[Y|T]) \leftarrow X \leq Y \wedge \boxed{q1(Y,T)}$

LR-clauses
(no existential variables)

Transformed program (2)

*D*4: $f \leftarrow \neg\, p$

*D*3: $p \leftarrow list(L) \wedge \neg\, q(L)$

*D*2: $q([\,]) \leftarrow$

$q([X|T]) \leftarrow q1(X,T)$

$q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\,,\,L)$

$q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\,,\,L)$

No existential variables

*D*1: $r([X|T],U) \leftarrow X > U \wedge list(T)$

$r([X|T],U) \leftarrow r(T,U)$

Deriving a Propositional Program

*D*3: $p \leftarrow list(L) \wedge \neg\, q(L)$

Deriving a Propositional Program

*D*3: $p \leftarrow \underline{list(L)} \wedge \underline{\neg\, q(L)}$

Unfold: $p \leftarrow list(T) \wedge \neg\, q1(X,T)$ Folding not possible

Deriving a Propositional Program

*D*3: $p \leftarrow list(L) \wedge \neg\, q(L)$

Unfold: $p \leftarrow \boxed{list(T) \wedge \neg\, q1(X,T)}$

Define: $p1 \leftarrow \boxed{list(T) \wedge \neg\, q1(X,T)}$

Deriving a Propositional Program

D3:	$p \leftarrow list(L) \wedge \neg q(L)$
Unfold:	$p \leftarrow \boxed{list(T) \wedge \neg q1(X,T)}$
Define:	$p1 \leftarrow \boxed{list(T) \wedge \neg q1(X,T)}$
Fold:	$p \leftarrow \boxed{p1}$

LR-clause
(no existential variables,
propositional)

Deriving a Propositional Program

New Def.: $p1 \leftarrow list(T) \land \neg\, q1(X,T)$

Deriving a Propositional Program

New Def.: $p1 \leftarrow \underline{list(T)} \wedge \underline{\neg\, q1(X,T)}$

Unfold:

$p1 \leftarrow X > Y \wedge list(T) \wedge \neg\, q1(X,T)$

$p1 \leftarrow X \leq Y \wedge list(T) \wedge \neg\, q1(Y,T)$

Deriving a Propositional Program

New Def.: $p1 \leftarrow list(T) \wedge \neg\, q1(X,T)$

Unfold:
$p1 \leftarrow \boxed{X>Y} \wedge list(T) \wedge \neg\, q1(X,T)$
$p1 \leftarrow \boxed{X\leq Y} \wedge list(T) \wedge \neg\, q1(Y,T)$

$\boxed{\exists Y\ X>Y \ \equiv\ true}$

$\boxed{\exists X\ X\leq Y \ \equiv\ true}$

Replace-Constraints (variable elimination):
$p1 \leftarrow list(T) \wedge \neg\, q1(Y,T)$

Deriving a Propositional Program

New Def.: $p1 \leftarrow \boxed{list(T) \wedge \neg\, q1(X,T)}$

Unfold:

$p1 \leftarrow \boxed{X>Y} \wedge list(T) \wedge \neg\, q1(X,T)$

$p1 \leftarrow \boxed{X\leq Y} \wedge list(T) \wedge \neg\, q1(Y,T)$

☺

Replace-Constraints (variable elimination):

$p1 \leftarrow \boxed{list(T) \wedge \neg\, q1(Y,T)}$

Fold: $p1 \leftarrow \boxed{p1}$

LR-clause
(no existential variables,
propositional)

The Final LR-Program

No existential variables

Prop

$$T \begin{cases} D4: & f \leftarrow \neg\, p \\ D3: & p \leftarrow p1 \\ & p1 \leftarrow p1 \\ D2: & q([\,]) \leftarrow \\ & q([X|T]) \leftarrow q1(X,T) \\ & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\,,\,L) \\ & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\,,\,L) \\ D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\ & r([X|T],U) \leftarrow r(T,U) \end{cases}$$

Proving Propositional Formulas

Prop

$$
T \left\{
\begin{array}{ll}
D4: & f \leftarrow \neg p \\
D3: & p \leftarrow p1 \\
 & \cancel{p1 \leftarrow p1} \quad \text{tautology} \\
D2: & q([\,]) \leftarrow \\
 & q([X|T]) \leftarrow q1(X,T) \\
 & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\,,\,L) \\
 & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\,,\,L) \\
D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\
 & r([X|T],U) \leftarrow r(T,U)
\end{array}
\right.
$$

Proving Propositional Formulas

Prop

$$
T\left\{
\begin{array}{ll}
D4: & f \leftarrow \neg p \\
D3: & p \leftarrow p1 \\
 & \\
D2: & q([\,]) \leftarrow \\
 & q([X|T]) \leftarrow q1(X,T) \\
 & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\,,\,L) \\
 & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\,,\,L) \\
D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\
 & r([X|T],U) \leftarrow r(T,U)
\end{array}
\right.
$$

Proving Propositional Formulas

Prop

$$T \left\{ \begin{array}{ll} D4: & \boxed{\begin{array}{l} f \leftarrow \neg\, p \\ \cancel{p \leftarrow p1} \end{array}} \quad \text{unfold} \\ D3: & \\ D2: & q([\,]) \leftarrow \\ & q([X|T]) \leftarrow q1(X,T) \\ & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\,,\,L) \\ & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\,,\,L) \\ D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\ & r([X|T],U) \leftarrow r(T,U) \end{array} \right.$$

Proving Propositional Formulas

$$
T \left\{
\begin{array}{ll}
D4: & \boxed{\begin{array}{l} \overset{Prop}{f \leftarrow \neg p} \\ \\ \\ \end{array}} \\
D3: & \\
D2: & q([\ ]) \leftarrow \\
 & q([X|T]) \leftarrow q1(X,T) \\
 & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\ ,\ L) \\
 & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\ ,\ L) \\
D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\
 & r([X|T],U) \leftarrow r(T,U)
\end{array}
\right.
$$

Proving Propositional Formulas

$$
T \left\{
\begin{array}{ll}
D4: & \boxed{\begin{array}{l} \overset{\textit{Prop}}{} \\ f \leftarrow \cancel{\neg p} \\ \\ \\ \end{array}} \quad \text{unfold} \\
D3: & \\
D2: & q([\,]) \leftarrow \\
 & q([X|T]) \leftarrow q1(X,T) \\
 & q1(X,[Y|T]) \leftarrow X > Y \land q1(X\ ,\ L) \\
 & q1(X,[Y|T]) \leftarrow X \leq Y \land q1(Y\ ,\ L) \\
D1: & r([X|T],U) \leftarrow X > U \land list(T) \\
 & r([X|T],U) \leftarrow r(T,U)
\end{array}
\right.
$$

Proving Propositional Formulas

$$T \left\{ \begin{array}{ll} D4: & \boxed{\begin{array}{l} \overset{Prop}{f \leftarrow} \\ \\ \\ \end{array}} \quad \Rightarrow \quad M(Prop) \models f \quad \Rightarrow \quad M(P) \models \varphi \\ D3: & \\ D2: & q([\,]) \leftarrow \\ & q([X|T]) \leftarrow q1(X,T) \\ & q1(X,[Y|T]) \leftarrow X > Y \wedge q1(X\ ,\ L) \\ & q1(X,[Y|T]) \leftarrow X \leq Y \wedge q1(Y\ ,\ L) \\ D1: & r([X|T],U) \leftarrow X > U \wedge list(T) \\ & r([X|T],U) \leftarrow r(T,U) \end{array} \right.$$

Experimental Results

- The Unfold/Fold transformation strategy for elimination of existential variables is implemented on the MAP system (www.iasi.cnr.it/~proietti/system.html).

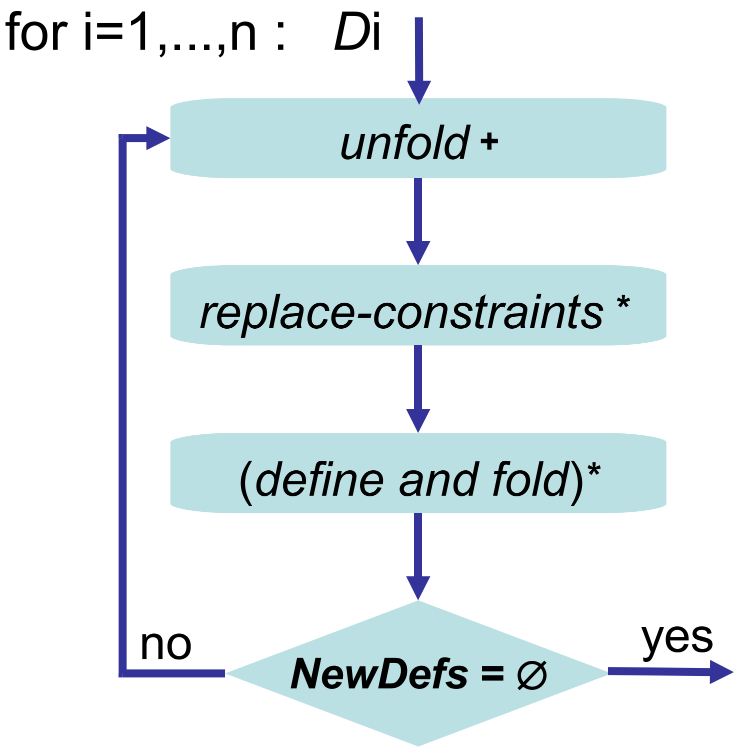
- Constraints are handled using the clp(r) module of SICStus Prolog (implementing a variant of Fourier-Motzkin variable elimination)

- Proven formulas in the theory of linear orders, lists, and addition

Property	Time (PM 1.73)
$\forall L\ \exists U\ \forall Y\ (\ member(Y,L) \rightarrow Y \leq U\ )$	31 ms
$\forall L\ \forall Y\ (\ (sumlist(L,Y) \wedge Y > 0) \rightarrow \exists X\ (member(X,L) \rightarrow X > 0)\ )$	15 ms
$\forall L\ \forall M\ \forall N\ (\ (ord(L) \wedge ord(M) \wedge sumzip(L,M,N)) \rightarrow ord(N)\ )$	16 ms
$\forall L\ \forall M\ \forall X\ \forall Y\ (\ (leqlist(L,M) \wedge sumlist(L,X) \wedge sumlist(M,Y)) \rightarrow X \leq Y\ )$	16 ms

Termination of the Unfold-Fold Strategy

The only source for nontermination is the possible introduction of infinitely many new predicates.



Introduction of new predicates

Final program *T*

Conclusions

- Program transformations to eliminate existential variables (deforestation) can be used for automated theorem proving;

- Future work

  - Identify theories of interest for which the unfold/fold strategy succeeds and, thus, works as a decision procedure;

  - Extend to other data structures (e.g. trees) and other domains closed under projection;

  - Investigate generalization techniques to avoid non-termination, with partial elimination of existential variables.