Proving Theorems by Program Transformation

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

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

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

Pescara, 16 Aprile 2010

An Introductory Example

- A property of the sum of integers:

  $\forall X\ \forall Y\ \forall Z\ (Y \geq 0 \wedge sum(X,Y,Z) \rightarrow Z \geq X)$

- A CLP encoding: no counter-example to the property can be found:

  $prop \leftarrow \neg\, negprop$

  $negprop \leftarrow Y \geq 0 \wedge sum(X,Y,Z) \wedge Z < X$

  $sum(0,Y,Z) \leftarrow Y = Z$

  $sum(X+1,Y,Z+1) \leftarrow sum(X,Y,Z)$

- SLDNF-resolution does not terminate for the goal

  $\leftarrow prop$

An Introductory Example

$negprop \leftarrow Y \geq 0 \wedge sum(X,Y,Z) \wedge Z < X$

An Introductory Example

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X,Y,Z) \wedge Z<X$

Unfold sum(X,Y,Z) (a resolution step):

negprop $\leftarrow$ $Y \geq 0 \wedge Y=Z \wedge Z+1<0+1$ [X=0]

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X',Y,Z') \wedge Z'+1<X'+1$ [X=X'+1, Z=Z'+1]

An Introductory Example

negprop $\leftarrow$ Y≥0 ∧ sum(X,Y,Z) ∧ Z<X

Unfold sum(X,Y,Z) (a resolution step):

~~negprop $\leftarrow$ Y≥0 ∧ Y=Z ∧ Z+1<0+1~~ (unsatisfiable body) [X=0]

negprop $\leftarrow$ Y≥0 ∧ sum(X',Y,Z') ∧ Z'+1<X'+1 [X=X'+1, Z=Z'+1]

Replacement of equivalent constraints:

negprop $\leftarrow$ Y≥0 ∧ sum(X',Y,Z') ∧ Z'<X'

An Introductory Example

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X,Y,Z) \wedge Z < X$

Unfold sum(X,Y,Z) (a resolution step):

~~negprop $\leftarrow$ $Y \geq 0 \wedge Y = Z \wedge Z+1 < 0+1$~~ (unsatisfiable body) [X=0]

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X',Y,Z') \wedge Z'+1 < X'+1$ [X=X'+1, Z=Z'+1]

Replacement of equivalent constraints:

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X',Y,Z') \wedge Z' < X'$

An Introductory Example

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X,Y,Z) \wedge Z<X$

Unfold sum(X,Y,Z) (a resolution step):

~~negprop $\leftarrow$ $Y \geq 0 \wedge Y=Z \wedge Z+1<0+1$~~ (unsatisfiable body) [X=0]

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X',Y,Z') \wedge Z'+1<X'+1$ [X=X'+1, Z=Z'+1]

Replacement of equivalent constraints:

negprop $\leftarrow$ $Y \geq 0 \wedge sum(X',Y,Z') \wedge Z'<X'$

Fold:

negprop $\leftarrow$ negprop

An Introductory Example

Transformed Program T:

prop $\leftarrow \neg$ negprop

negprop $\leftarrow$ negprop

T is *propositional*: the transformation has *eliminated all variables*.

The perfect model of T is

M(T) = {prop}

that is, prop holds in T.

By the correctness of the transformation, prop holds in the initial program.

The Transformational Proof Method

- We are given a CLP(R_{lin}) program

  P: $sum(0,Y,Z) \leftarrow Y=Z$

  $sum(X+1,Y,Z+1) \leftarrow sum(X,Y,Z)$

  and a closed first order formula

  φ: $\forall X\ \forall Y\ \forall Z\ (Y \geq 0 \wedge sum(X,Y,Z) \rightarrow Z \geq X)$

  we want to prove:

  $M(P) \models \varphi$

The Transformational Proof Method

- **Step1.** By a variant of *Lloyd-Topor transformation* we get a set of clauses (clause form)

  CF: $prop \leftarrow \neg negprop$

  $negprop \leftarrow Y \geq 0 \wedge sum(X,Y,Z) \wedge Z < X$

  s.t.

  $M(P) \models \varphi$ iff $M(P \cup CF) \models prop$

  where M( ) denotes the *perfect model*.

The Transformational Proof Method

- **Step 2.** By *unfold/fold* transformations from $P \cup CF$ we derive a *propositional* program

Prop: $prop \leftarrow \neg negprop$

$negprop \leftarrow negprop$

s.t. $M(P \cup CF) \models prop$ iff $M(Prop) \models prop$

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

- The transformation from $P \cup CF$ to Prop consists in *eliminating the existential variables* from CF, i.e., the variables occurring in the body of a clause and not in the head. For instance, X, Y, Z are existential variables in

$negprop \leftarrow Y \geq 0 \wedge sum(X,Y,Z) \wedge Z < X$

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

Related Work

- Transformation strategies for *eliminating existential variables* [PP-95]

- 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,PPS 09]
  - Coinductive proofs via unfold/fold [Seki 99]
  - Alberto's talk on transforming programs on infinite lists [PPS 09]

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.

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]) ← X=Y

  member(X,[Y|L]) ← member(X,L)

  pos_sumlist([ ], Y) ← Y=0

  pos_sumlist([X|L],Y) ← X>0 ∧ pos_sumlist(L,Y−X)

  pos_sumlist([X|L],Y) ← X≤0 ∧ pos_sumlist(L,Y)

- Not an LR-program:

  permutation([ ], [ ]) ←

  permutation([X|L1],L2) ← permutation(L1,L3) ∧ 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 prop is defined by a set $Prop \subseteq T$ of propositional clauses. Thus, quantifiers can be eliminated by deriving LR-programs.

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.

Clause-Form Transformation

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

$$prop \leftarrow \neg\exists L\ \neg\exists U\ \neg\exists X\ (\ member(X,L) \wedge \neg X \leq U\ )$$

r

q

p

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

CF:

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

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

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

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

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

- $M(P) \models \varphi$ iff $M(P \cup CF) \models prop$

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.

Eliminating Existential Variables via U/F

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

Eliminating Existential Variables via U/F

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

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

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

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

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

☺

Eliminating Existential Variables via U/F

D1: $r(L,U) \leftarrow X > U \land list(L) \land member(X,L)$

☺

Unfold:

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

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

Fold:

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

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

Eliminating Existential Variables via U/F

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

☺

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

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

LR-clauses
(no existential variables)

Transformed program (1)

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

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

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

D1: $r([X|T],U) \leftarrow X > U \wedge list(T)$

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

No existential variables

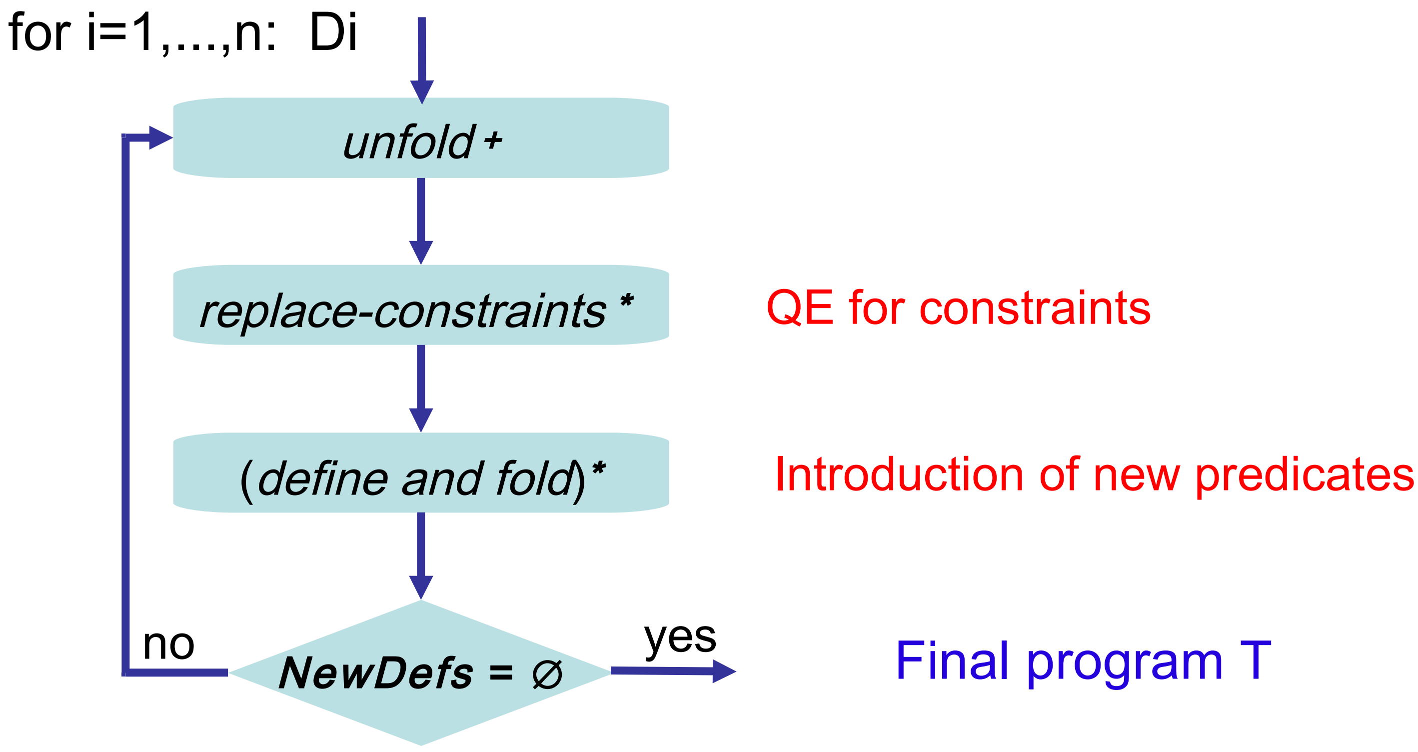
Overview

- ✔ *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.

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. prop is defined by a propositional Prop ⊆ T and M(P) ⊨ φ iff M(Prop) ⊨ prop.



Introducing New Definitions

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

Introducing New Definitions

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

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

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

Introducing New Definitions

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

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

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

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

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

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

Introducing New Definitions

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

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

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

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

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

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

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

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

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

☹

Bad folding!
Existential variable
not eliminated

Transforming New Definitions

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

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

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

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

linear order

$\equiv\ (X>Y \wedge X \leq U) \vee (X \leq Y \wedge Y \leq U)$

Replace-constraints:

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

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

Transforming New Definitions

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

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

Unfold:

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

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

☺

Replace-constraints:

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

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

Fold:

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

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

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

LR-clauses
(no existential variables)

Transformed program (2)

D4: $f \leftarrow \neg p$

D3: $p \leftarrow list(L) \wedge \neg q(L)$

D2: $q([\,]) \leftarrow$

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

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

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

No existential variables

D1: $r([X|T],U) \leftarrow X > U \wedge list(T)$

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

Deriving a Propositional Program

D3: $p \leftarrow list(L) \wedge \neg\ q(L)$

Deriving a Propositional Program

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

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

Deriving a Propositional Program

D3: $p \leftarrow list(L) \land \neg\ q(L)$

Unfold: $p \leftarrow list(T) \land \neg\ q1(X,T)$

Define: $p1 \leftarrow list(T) \land \neg\ q1(X,T)$

Deriving a Propositional Program

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

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

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

Fold: $p \leftarrow p1$

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

Unfold: $p1 \leftarrow X>Y \land list(T) \land \neg\ q1(X,T)$

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

Deriving a Propositional Program

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

Unfold:

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

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

$\exists Y\; X>Y \equiv true$

$\exists X\; X\leq Y \equiv true$

Replace-Constraints (variable elimination):

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

Deriving a Propositional Program

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

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

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

☺

Replace-Constraints (variable elimination):

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

Fold: $p1 \leftarrow p1$

LR-clause
(no existential variables,
propositional)

The Final LR-Program

Prop

No existential variables

T

D4: $\text{prop} \leftarrow \neg\, p$

D3: $p \leftarrow p1$

$p1 \leftarrow p1$

$M(Prop)=\{prop\} \;\Rightarrow\; M(P) \models \varphi$

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

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

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).

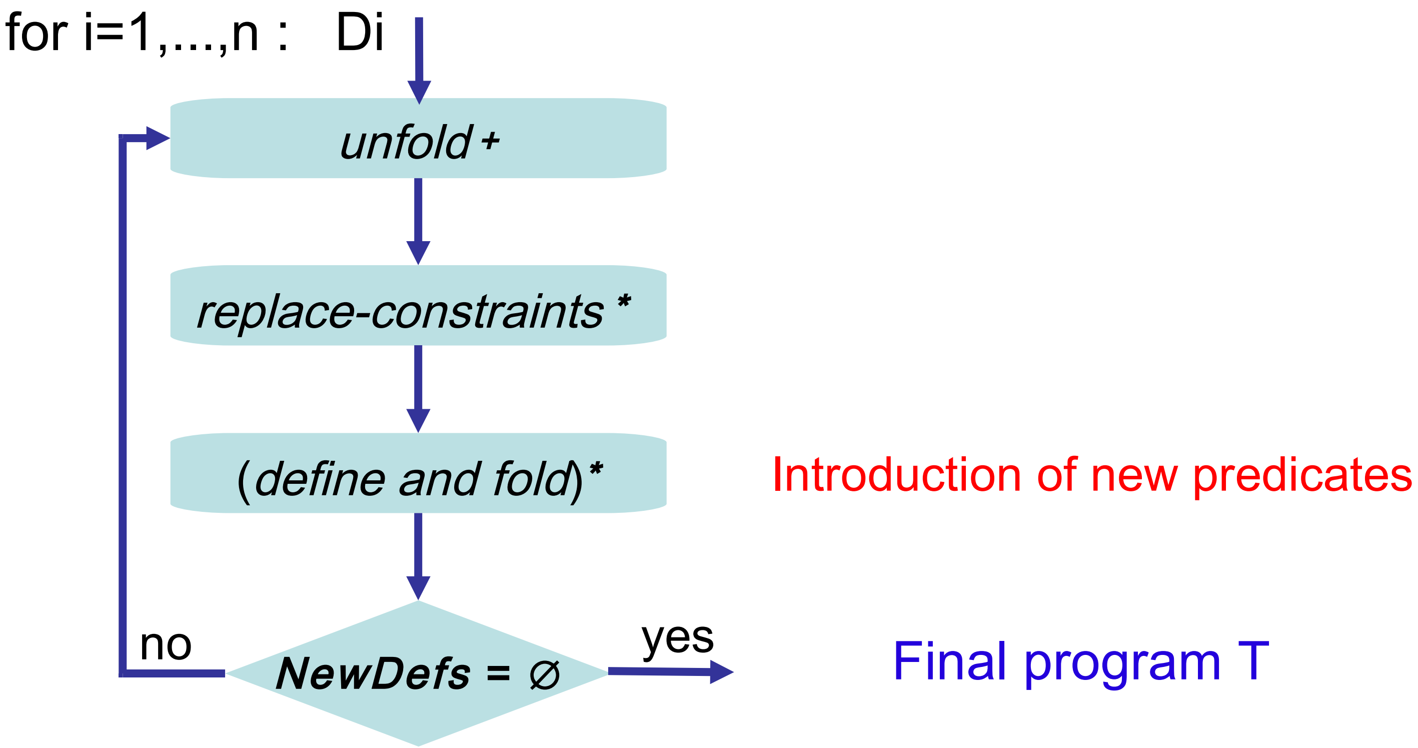
- Constraints are handled using the clp(r) module of SICStus Prolog (implementing a variant of Fourier-Motzkin variable elimination)

- 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.



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;
  - Extend to infinite structures and apply to the verification of reactive systems (Alberto's talk)