Software Verification and Synthesis via Program Transformation A Case Study: Monadic Second Order Logic

Fabio Fioravanti*, Alberto Pettorossi**, Maurizio Proietti*

{fioravanti,adp,proietti}@iasi.rm.cnr.it

(*) IASI-CNR
Viale Manzoni 30
00185 Rome, Italy

(**) DISP - Universita' di Roma Tor Vergata
Via del Politecnico
00133 Rome, Italy

Waseda University, Tokyo, Japan

March 4th, 2003

Goals of this work

- Establish a correspondence between
  Theorem Proving and ***Program Transformation***

- Exploit this correspondence for performing
  software verification
  by means of
  program transformers, ***program specializers***, ...

Long term goal: Design of a uniform framework based on unfold/fold transformations for software development (synthesis, verification, transformation, specialization).

A Case Study: Monadic Second Order Logics

Monadic second order (MSO) logics are logics of membership to sets of strings. Very expressive, decidable. [Büchi 60, Thatcher-Wright 68, Rabin 69]

MSO logics are useful for the automatic verification of finite state systems. [MONA, Klarlund et al. 96]

We will propose methods based on unfold/fold transformations for:

- proving WS1S formulas (a fragment of MSO), thereby yielding a 'completeness' result for unfold/fold transformations w.r.t. WS1S;

- synthesizing definite logic programs from WS1S specifications.

Modelling a multiprocess system

The number of processes of the system may change dynamically.
A process is identified by a natural number.
A state S of the system is represented by a pair $\langle W,U \rangle$ of sets of processes:
(1) the set W of processes waiting for a resource
(2) the set U of processes using a resource

$$\mathrm{reach}(S) \leftarrow \mathrm{init}(S)$$
$$\mathrm{reach}(S) \leftarrow \mathrm{create}(S1,S) \wedge \mathrm{reach}(S1)$$
$$\mathrm{reach}(S) \leftarrow \mathrm{use}(S1,S) \wedge \mathrm{reach}(S1)$$
$$\mathrm{reach}(S) \leftarrow \mathrm{release}(S1,S) \wedge \mathrm{reach}(S1)$$

where init, create, use, release represent the transition relation and are specified by WS1S formulas.

The Transition Relation

Initial state:

$init(\langle W,U \rangle) \equiv empty(W) \wedge empty(U)$

Create a new process:

$cre(\langle W,U \rangle,\langle W1,U1 \rangle) \equiv$

$\exists Z\ (Z = W \cup U \wedge$

$((empty(Z) \wedge W1=\{0\}) \vee$

$(\neg empty(Z) \wedge \exists M\ (max(Z,M) \wedge W1=W \cup \{M+1\})))) \wedge$

$U1=U$

Use the resource:

$use(\langle W,U \rangle,\langle W1,U1 \rangle) \equiv$

$\exists N\ (N \in W \wedge \exists Z\ (Z = W \cup U \wedge min(Z,N)) \wedge W1=W-\{N\} \wedge$

$U1=U \cup \{N\})$

Release the resource:

$rel(\langle W,U \rangle,\langle W1,U1 \rangle) \equiv$

$W1=W \wedge$

$\exists N\ (N \in U \wedge U1=U-\{N\})$

Verification and Synthesis Examples

We will present automatic methods based on unfold/fold transformations for

- proving WS1S formulas, such as:

$\forall W\ \forall U\ \forall W1\ \forall U1\ (use(\langle W,U\rangle,\langle W1,U1\rangle) \rightarrow \neg empty(U1))$

- synthesizing terminating definite logic programs from WS1S specifications, such as init, create, use, release:

$use(\langle W,U\rangle,\langle W1,U1\rangle) \equiv$

$\exists N\ (N \in W \wedge \exists Z\ (Z = W \cup U \wedge min(Z,N)) \wedge W1 = W - \{N\} \wedge U1 = U \cup \{N\})$

Also empty, $\cup$, -, min, have WS1S specifications.

Overview

- Weak monadic second order theory of one successor (WS1S)
- Encoding WS1S into stratified logic programs
- The unfold/fold proof method:
  - transformation rules
  - transformation strategy
- Termination of the strategy
- The unfold/fold synthesis method
- Optimizations: Discarding useless types, Determinization, Minimization, Deletion of useless clauses
- Implementation

WS1S: Syntax

The Weak Monadic Second Order theory of one successor (WS1S) is the theory of membership to finite sets of natural numbers.

Syntax **(2-sorted)**

Individual variables: N in Ivars

Set variables: S in Svars

Function symbols: 0, s(_)

Predicate symbols: $\in$

Individual terms: $n ::= 0 \mid N \mid s(n)$

Formulas: $\varphi ::= n \in S \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists N\, \varphi \mid \exists S\, \varphi$

We also use $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$ as abbreviations.

We consider a first order presentation.

For a second order presentation, write S(n) instead of $n \in S$.

WS1S: Semantics

Semantics

Interpretation $\mathfrak{I}$

- Domain of $\mathfrak{I}$: $\quad Nat \cup P_{fin}(Nat)$ $\quad$ where $Nat = \{0, 1, 2, \ldots\}$

- 0 is interpreted as the zero of Nat
  $s(\_)$ is interpreted as the successor function $+1$
  $\in$ is interpreted as the membership relation on $Nat \times P_{fin}(Nat)$

- $\models_{WS1S} \varphi$ iff $\mathfrak{I} \models \varphi$

$\models_{WS1S} \varphi$ is decidable [Büchi 60, by using automata-theory]

WS1S

WS1S specifications (i.e. open formulas)

set equality: $S1=S2 \equiv \forall N\ (N\in S1 \leftrightarrow N\in S2)$

S is 'downward' closed

order over numbers:

$$N1\leq N2 \equiv \forall S\ (N2\in S \wedge \boxed{\forall N3\ (s(N3)\in S \rightarrow N3\in S)} \rightarrow N1\in S)$$

WS1S properties (i.e. closed formulas)

Every finite set of natural numbers has a maximum element:

$$\forall S\ \exists N(N\in S \wedge \neg\exists N1\ (N1\in S \wedge \neg N1\leq N))$$

There exists no finite set which is nonempty and 'upward' closed:

$$\neg\exists S\ (\exists N1\ (N1\in S) \wedge \forall N2\ (N2\in S \rightarrow s(N2)\in S))$$

Encoding WS1S into LP (1)

Natural numbers: 0, s(0), s(s(0)), . . .

Finite sets: $[b_0, b_1, \ldots, b_n]$ where b_i in {**t**,**f**}

$s^k(0)$ belongs to $[b_1, \ldots, b_n]$ iff $0 \le k \le m$ and b_k=**t**

{0,3,4} is represented by [**t**,**f**,**f**,**t**,**t**] and also by [**t**,**f**,**f**,**t**,**t**,**f**]

∅ is represented by [ ] and also by [**f**,**. . .**,**f**]

Program Natset

nat(0) ←	0 ≤ N ←
nat(s(N)) ← nat(N)	s(N1) ≤ s(N2) ← N1 ≤ N2
set([ ]) ←	0 ∈ [**t**|S] ←
set([**t**|S]) ← set(S)	s(N) ∈ [B|S] ← N ∈ S
set([**f**|S]) ← set(S)	

Encoding WS1S into LP (2)

Given the WS1S formula: $\varphi \equiv \forall N1\ \exists N2\ N1 \leq N2$

Rewrite as: $\psi \equiv \neg\exists N1 \neg\exists N2\ N1 \leq N2$

Apply the Lloyd-Topor transformation starting from the statement $f \leftarrow \psi$

$$f \leftarrow \neg\ \boxed{\exists N1 \neg\ \boxed{\exists N2\ N1 \leq N2}}$$

$$\boxed{\begin{array}{l} f \leftarrow \neg\ \boxed{g} \\ g \leftarrow \neg\ \boxed{h(N1)} \\ h(N1) \leftarrow N1 \leq N2 \end{array}}$$

Add types

Cls(f,φ):

$$\boxed{\begin{array}{l} f \leftarrow \neg\ g \\ g \leftarrow \underline{nat(N1)} \wedge \neg\ h(N1) \\ h(N1) \leftarrow \underline{nat(N1) \wedge nat(N2)} \wedge N1 \leq N2 \end{array}}$$

Stratified program

f>g>h

Semantics of Definite Logic Programs

- Every definite logic program (no negation in bodies) has a least Herbrand model.
- B_L: the set of all ground atoms in the first order language L used for writing programs and formulas
- Interpretation: a subset $\Im$ of B_L (i.e. $\Im \in 2^{B_L}$)
  $\Im \models A$ iff $A \in \Im$ and $\Im \models \neg A$ iff $A \notin \Im$
- Immediate consequence operator: $T_P: 2^{B_L} \rightarrow 2^{B_L}$

$$T_P(\Im) = \{A \mid A \leftarrow L_1 \wedge \ldots \wedge L_n \text{ is a ground instance of a clause in P, and, for } i = 1,\ldots,n,\ \Im \models L_i\}$$

- T_P is a continuous operator on the lattice 2^{B_L} of all interpretations. The least Herbrand model of P is: $M(P) = \mathrm{lfp}(T_P) = T_P^{\omega}(\emptyset)$.

Nonmonotonicity and Semantics of Negation

- For general logic programs (with negated atoms in bodies) T_P is nonmonotonic and some programs have no least Herbrand model. For instance:

$$p \leftarrow \neg q$$

  has two minimal models: $\{p\}$ and $\{q\}$

- We can associate a program with a unique model (possibly not least) by ordering the atoms: $q < p$.
  We compute the model bottom-up wrt $<$:
  1. The least model of the clauses with head q is $\emptyset$ and q is false in $\emptyset$.
  2. Assuming that q is false, the least model of $p \leftarrow \neg q$ is $\{p\}$.

Stratified Logic Programs

- A level mapping is a function σ: $B_L \rightarrow \omega$ where ω is the set of natural numbers. $\sigma(\neg A)=\sigma(A)$

- A clause $H \leftarrow L_1 \wedge \ldots \wedge L_n$ is stratified wrt σ iff
  for i=1,...,n, if L_i is an atom then $\sigma(H) \geq \sigma(L_i)$
  if L_i is a negated atom then $\sigma(H) > \sigma(L_i)$
  (no recursion through negation)

- A program P is stratified iff there exists a level mapping σ such that every clause of P is stratified wrt σ

- P: $p \leftarrow \neg q$ is stratified (wrt any level mapping σ such that $\sigma(p) > \sigma(q)$)
  $q \leftarrow q$

- $p \leftarrow \neg p$ is not stratified: there is no level mapping σ such that $\sigma(p) > \sigma(p)$

Perfect Model

- Immediate consequence operator $T_{P,n}: \wp(B_L) \rightarrow \wp(B_L)$
  where P is a stratified program and $n \in \omega$

$$T_{P,n}(\Im) = \{A \mid A \leftarrow L_1 \wedge \ldots \wedge L_n \text{ is a ground instance of a clause in P, } \sigma(A)=n, \text{ and for } i = 1,\ldots,n \text{ if } \sigma(L_i)=n \text{ then } \Im \models L_i \text{ else if } \sigma(L_i)=m<n \text{ then } lfp(T_{P,m}) \models L_i\}$$

$T_{P,n}(\Im)$ is the set of atoms at level n that are one-step consequences (using the clauses in P) of the literals that are true in $\Im$ and of the literals that are true at any level $m<n$

- For every ordinal $n \in \omega$, $T_{P,n}$ is a continuous operator on the lattice 2^{B_L}. The perfect model of P is defined as:

$$M(P) = \bigcup_{n \in \omega} lfp(T_{P,n})$$

- P: $p \leftarrow \neg q$ $\quad\quad M(P)=\{p\}$
  $\quad q \leftarrow q$

Lloyd-Topor Transformation

Apply as long as possible the following transformations
($C[\psi]$ denotes a formula of the form: $\ldots \wedge \psi \wedge \ldots$)

$$H \leftarrow C[\neg\neg\psi] \quad \Rightarrow \quad H \leftarrow C[\psi]$$

$$H \leftarrow C[\neg(\psi_1 \wedge \psi_2)] \quad \Rightarrow \quad \begin{cases} H \leftarrow C[\neg newp(X_1,\ldots,X_k)] \\ newp(X_1,\ldots,X_k) \leftarrow \psi_1 \wedge \psi_2 \end{cases}$$

where $X_1,\ldots,X_k$ are the free variables of $\psi_1 \wedge \psi_2$

$$H \leftarrow C[\neg \exists X\ \psi] \quad \Rightarrow \quad \begin{cases} H \leftarrow C[\neg newp(X_1,\ldots,X_k)] \\ newp(X_1,\ldots,X_k) \leftarrow \psi \end{cases}$$

where $X_1,\ldots,X_k$ are the free variables of ψ

$$H \leftarrow C[\exists X\ \psi] \quad \Rightarrow \quad H \leftarrow C[\psi\{X/Y\}]$$

where Y is a new variable

Addition of Types

$$H \leftarrow A_1 \wedge \ldots \wedge A_n$$

$$\downarrow$$

$$H \leftarrow nat(N_1) \wedge \ldots \wedge nat(N_k) \wedge$$
$$set(S_1) \wedge \ldots \wedge set(S_m) \wedge$$
$$A_1 \wedge \ldots \wedge A_n$$

Individual variables occurring in $H \leftarrow A_1 \wedge \ldots \wedge A_n$

Set variables occurring in $H \leftarrow A_1 \wedge \ldots \wedge A_n$

Correctness of the Encoding

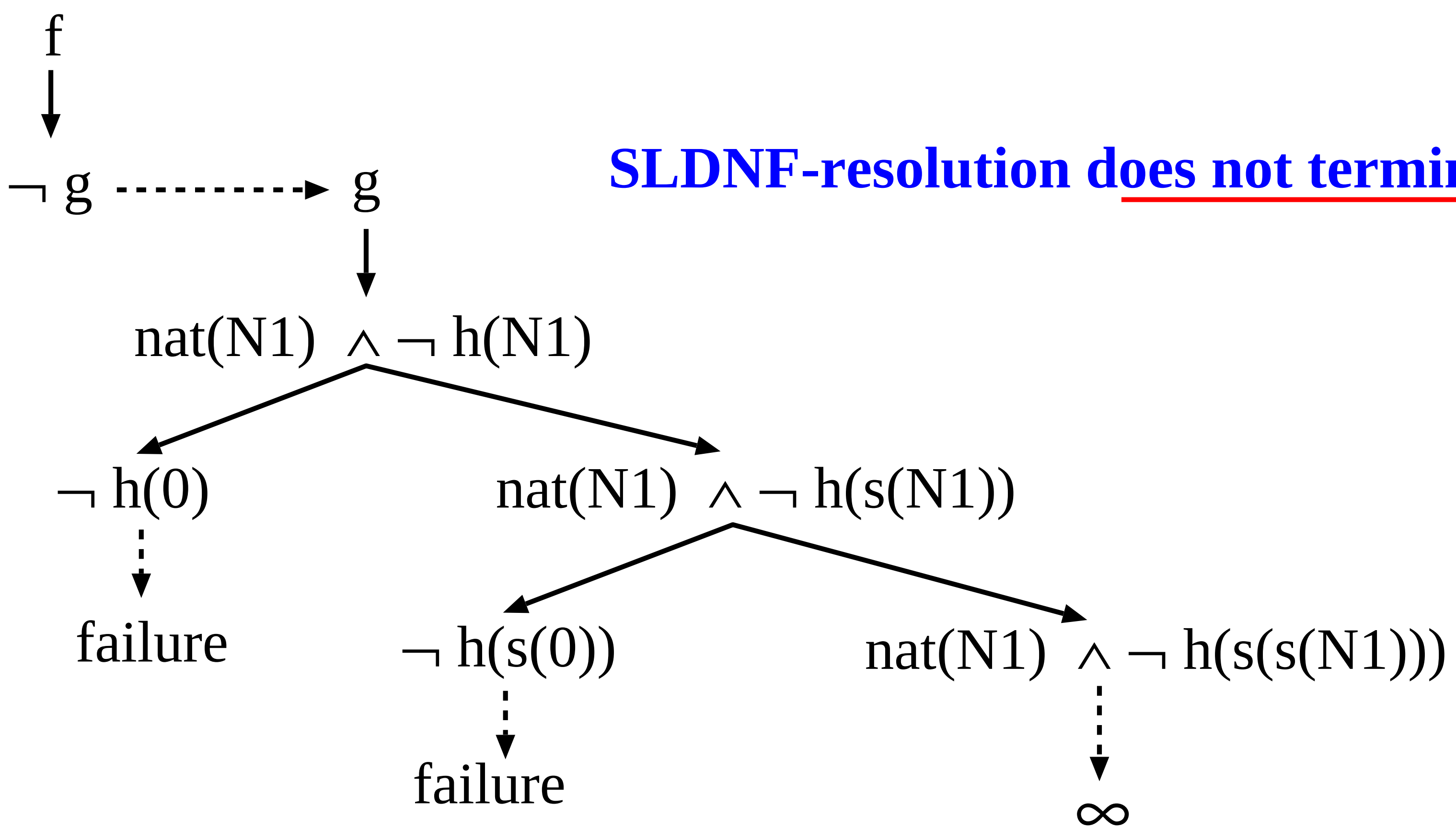
Theorem

$|=_{WS1S} \varphi$ iff $M(\text{Natset} \cup \text{Cls}(f,\varphi)) \models f$

Perfect (= Standard = Stable = Well-founded) model

Limitations of SLDNF-resolution (Prolog)

$$
\mathrm{Cls}(f,\varphi):\quad
\begin{aligned}
& f \leftarrow \neg\, g \\
& g \leftarrow nat(N1) \wedge \neg\, h(N1) \\
& h(N1) \leftarrow nat(N1) \wedge nat(N2) \wedge N1 \le N2
\end{aligned}
$$

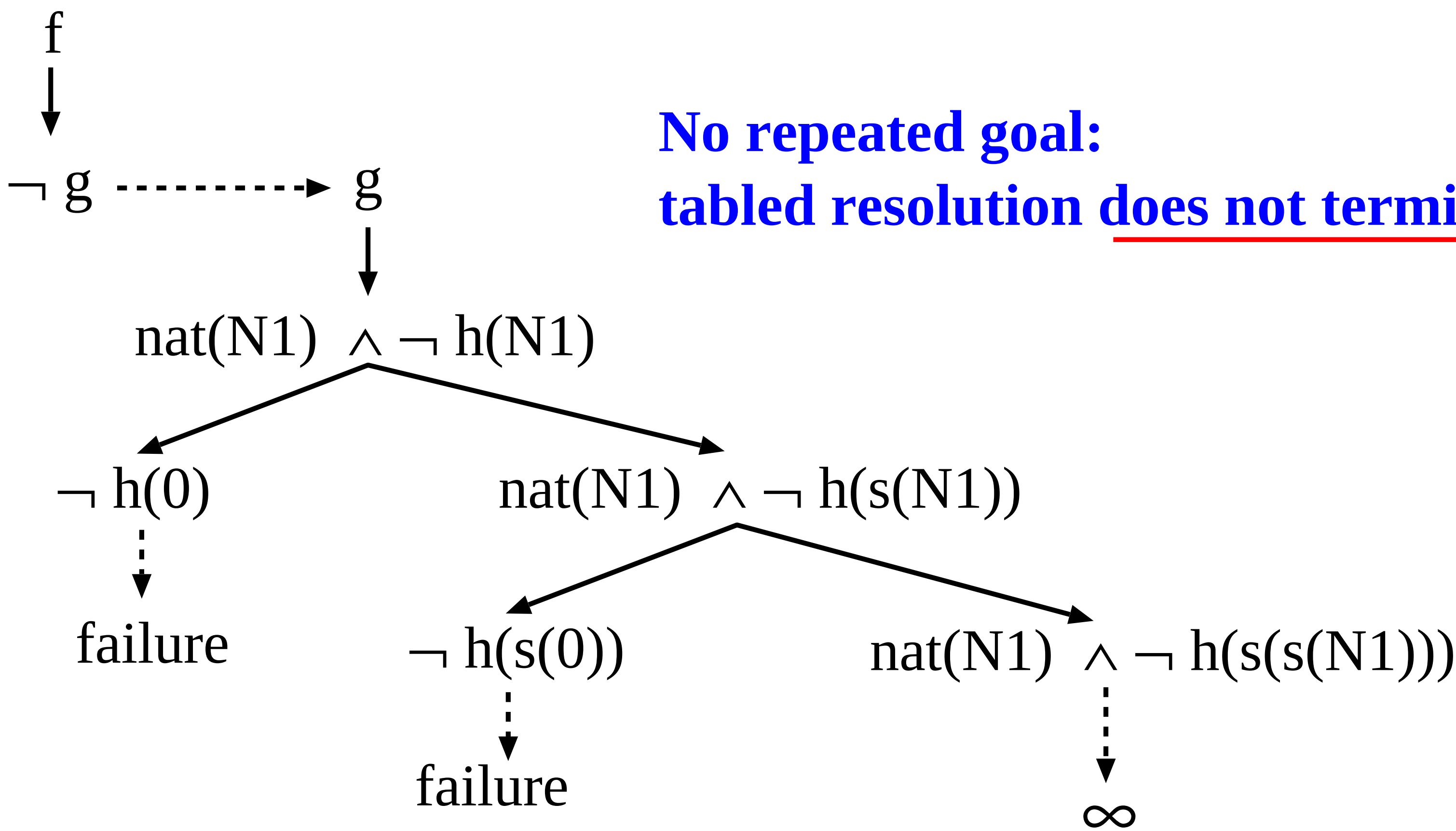


SLDNF-resolution does not terminate

Limitations of Tabled Resolution (XSB)

	$f \leftarrow \neg\, g$
Cls(f,φ):	$g \leftarrow nat(N1) \wedge \neg\, h(N1)$
	$h(N1) \leftarrow nat(N1) \wedge nat(N2) \wedge N1 \leq N2$

No repeated goal:
tabled resolution does not terminate



. . . Proving WS1S via Program Transformation

$\boxed{g \leftarrow \boxed{nat(N1) \wedge \neg h(N1)}}$ **level 2**

$\boxed{g \leftarrow \boxed{nat(N1) \wedge \neg h(N1)}}$ **unfolding**

$\cancel{\boxed{g \leftarrow \boxed{g}}}$ **folding**

tautology

$\boxed{f \leftarrow \neg g}$ **level 3**

$\boxed{f \leftarrow}$ **unfolding**

$\Rightarrow$ $M(Natset \cup Cls(f,\varphi)) \models f$

$\Rightarrow$ by the correctness of the encoding and the correctness of the transformations,

$$\boxed{\models_{WS1S} \varphi}$$

The Unfold/Fold Proof Method

Let φ be a closed WS1S formula.

Step 1. (Encoding into stratified LP)

$$\varphi \longrightarrow \mathrm{Cls}(f,\varphi)$$

Lloyd-Topor transformation + type addition

$\models_{WS1S} \varphi$ iff $M(\mathrm{Natset} \cup \mathrm{Cls}(f,\varphi)) \models f$

Step 2. (Unfold/fold transformations)

$$\mathrm{Natset} \cup \mathrm{Cls}(f,\varphi) \rightarrow \cdots \rightarrow T$$

unfolding and folding rules applied according to a strategy

$M(\mathrm{Natset} \cup \mathrm{Cls}(f,\varphi)) \models f$ iff $f \leftarrow$ belongs to T

Rule-based Program Transformation

- We consider stratified normal logic programs with the Perfect Model (= Standard Model) semantics.

- Program transformation: Construct a sequence of programs

$$P_0 \longrightarrow \;\dots\; \longrightarrow P_n$$

where P_{k+1} is derived from P_k by applying a transformation rule.

- The transformation rules preserve the Perfect Model.

$$\boxed{M(P_0 \cup Defs_n) = M(P_n)}$$

where $Defs_n$ is the set of new definitions introduced during program transformation.

The Unfold/Fold Transformation Rules

- Construct a transformation sequence, that is, a sequence of programs

$$P_0 \longrightarrow \;\cdots\; \longrightarrow P_n$$

  where P_{k+1} is derived from P_k by applying a transformation rule

- **Transformation Rules**:
  - R1. Definition Introduction
  - R2. Unfolding (w.r.t. positive or negative literals)
  - R3. Folding
  - R4. Tautologies

LT Transformation: Less-or-Equal Example

P: $0 \leq N \leftarrow$ $\quad$ $nat(0) \leftarrow$

$s(N1) \leq s(N2) \leftarrow N1 \leq N2$ $\quad$ $nat(s(X)) \leftarrow nat(X)$

φ: $\forall N\ N \leq s(N)$

Rewrite φ as: $\neg \exists N\ \neg N \leq s(N)$

Introduce the statement: $f \leftarrow \neg \exists N \neg N \leq s(N)$

LT transformation + type addition

Cls(f,φ): $\begin{cases} f \leftarrow \neg g \\ g \leftarrow \underline{nat(N)} \wedge \neg N \leq s(N) \end{cases}$

Locally stratified

Note:

$\forall N\ (g \leftarrow nat(N) \wedge \neg N \leq s(N)) \equiv g \leftarrow \exists N (nat(N) \wedge \neg N \leq s(N))$

Definition Introduction: Less-or-Equal Example

P_0: $0 \leq N \leftarrow$

$s(N1) \leq s(N2) \leftarrow N1 \leq N2$

$nat(0) \leftarrow$

$nat(s(X)) \leftarrow nat(X)$

Definition Introduction (twice): δ_1: $g \leftarrow nat(N) \wedge \neg N \leq s(N)$

δ_2: $f \leftarrow \neg g$

$Defs_2 = \{\delta_1, \delta_2\}$

P_2: $0 \leq N \leftarrow$

$s(N1) \leq s(N2) \leftarrow N1 \leq N2$

$nat(0) \leftarrow$

$nat(s(X)) \leftarrow nat(X)$

δ_1: $g \leftarrow nat(N) \wedge \neg N \leq s(N)$

δ_2: $f \leftarrow \neg g$

R1. Definition Introduction

Introduce a new definition, that is, a clause

$$\delta:\ newp(X_1,\ldots,X_h) \leftarrow L_1 \wedge \ldots \wedge L_n$$

where:

- newp is a new predicate symbol
- $X_1,\ldots,X_h$ are distinct variables occurring in $L_1 \wedge \ldots \wedge L_n$
- the predicate symbols of $L_1 \wedge \ldots \wedge L_n$ occur in P_k

$$\boxed{P_{k+1} = P_k \cup \{\delta\}}$$

$Defs_k$ is the set of definitions introduced up to step k

No recursive definitions, no multiple clause definitions

Positive Unfolding: Less-or-Equal Example

P_2:
$0 \leq N \leftarrow$
$s(N1) \leq s(N2) \leftarrow N1 \leq N2$
$\boxed{nat(0)} \leftarrow$
$\boxed{nat(s(X))} \leftarrow \boxed{nat(X)}$
$\boxed{\delta_1: \quad g \leftarrow \boxed{nat(N)} \wedge \neg N \leq s(N)}$
$f \leftarrow \neg g$

replace

Unfolding δ_1 wrt $\boxed{nat(N)}$

P_3:
$0 \leq N \leftarrow$
$s(N1) \leq s(N2) \leftarrow N1 \leq N2$
$nat(0) \leftarrow$
$nat(s(X)) \leftarrow nat(X)$
$\lambda_1: \quad g \leftarrow \neg\, 0 \leq s(0)$ $\{N/0\}$
$\lambda_2: \quad g \leftarrow \boxed{nat(X)} \wedge \neg\, s(X) \leq s(s(X))$ $\{N/s(X)\}$
$f \leftarrow \neg g$

$Defs_3 = \{\delta_1, \delta_2\}$

$R2^{+}$. Positive Unfolding

Given a clause in P_k

$$\lambda:\ H \leftarrow G_1 \wedge \boxed{A} \wedge G_2$$

take all clauses in P_k whose head H_i unifies with A via an mgu θ_i

$$\boxed{H_1} \leftarrow \boxed{Body_1} \quad \ldots \quad \boxed{H_m} \leftarrow \boxed{Body_m}$$

and replace λ by all its resolvents w.r.t. the atom A

$$P_{k+1} = (P_k \setminus \{\lambda\}) \cup \left\{ \begin{array}{l} (H \leftarrow G_1 \wedge \boxed{Body_1} \wedge G_2)\ \theta_1 \\ \quad \cdots \\ (H \leftarrow G_1 \wedge \boxed{Body_m} \wedge G_2)\ \theta_m \end{array} \right\}$$

If m=0 then delete λ from P_k

Negative Unfolding: Less-or-Equal Example

P_3:
$\boxed{0 \leq N} \leftarrow$
$\boxed{s(N1) \leq s(N2)} \leftarrow \boxed{N1 \leq N2}$
$nat(0) \leftarrow$
$nat(s(X)) \leftarrow nat(X)$
λ_1: $g \leftarrow \boxed{\neg\, 0 \leq s(0)}$
λ_2: $g \leftarrow nat(X) \wedge \boxed{\neg\, s(X) \leq s(s(X))}$
$f \leftarrow \neg g$

replace

Unfolding λ_1 wrt $\boxed{\neg\, 0 \leq s(0)}$ and
λ_2 wrt $\boxed{\neg\, s(X) \leq s(s(X))}$

P_4:
$0 \leq N \leftarrow$
$s(N1) \leq s(N2) \leftarrow N1 \leq N2$
$nat(0) \leftarrow$
$nat(s(X)) \leftarrow nat(X)$
$\cancel{g \leftarrow \neg\, 0 \leq s(0)}$ $\{N/0\}$
λ_3: $g \leftarrow nat(X) \wedge \boxed{\neg\, X \leq s(X)}$ $\{N1/X,\ N2/X\}$
$f \leftarrow \neg g$

$Defs_4 = \{\delta_1, \delta_2\}$

$R2^-$. Negative Unfolding

Given a clause in P_k $\quad\lambda$: $H \leftarrow G_1 \wedge \boxed{\neg A} \wedge G_2$

take all clauses in P_k whose head H_i unifies with A via an mgu θ_i

$$\boxed{H_1} \leftarrow \boxed{Body_1} \quad \ldots \quad \boxed{H_m} \leftarrow \boxed{Body_m}$$

if (1) $A = H_1\,\theta_1 = \ldots = H_m\,\theta_m$ (A is an instance of $H_1, \ldots, H_m$)

(2) $Body_1, \ldots, Body_m$ have no existential variables

then take the disjunctive normal form

$$\boxed{Q_1} \vee \ldots \vee \boxed{Q_r} = DNF\,(\,G_1 \wedge \neg(\boxed{Body_1}\,\theta_1 \vee \ldots \vee \boxed{Body_m}\,\theta_m) \wedge G_2\,)$$

$$\boxed{P_{k+1} = (P_k \setminus \{\lambda\}) \cup \{H \leftarrow \boxed{Q_1}, \ldots, H \leftarrow \boxed{Q_r}\}}$$

If m=0 then delete $\neg A$ from the body of λ; if $A\theta \leftarrow$ is a clause in P_k then delete λ.

Negative Unfolding: Example

$h(X) \leftarrow nat(X) \wedge \neg\, \boxed{p(s(X))}$

$p(0) \leftarrow$

$\boxed{p(s(X))} \leftarrow \boxed{q_1(X)}$

$\boxed{p(X)} \leftarrow \boxed{q_2(X) \wedge q_3(X)}$

$$nat(X) \wedge \neg\, (\boxed{q_1(X)} \vee (\boxed{q_2(s(X)) \wedge q_3(s(X))}))$$

$$\big\downarrow \text{DNF}$$

$$\boxed{nat(X) \wedge \neg\, q_1(X) \wedge \neg\, q_2(s(X))} \vee$$
$$\boxed{nat(X) \wedge \neg\, q_1(X) \wedge \neg\, q_3(s(X))}$$

$h(X) \leftarrow \boxed{nat(X) \wedge \neg\, q_1(X) \wedge \neg\, q_2(s(X))}$

$h(X) \leftarrow \boxed{nat(X) \wedge \neg\, q_1(X) \wedge \neg\, q_3(s(X))}$

Folding: Less-or-Equal Example

P_4: $0 \leq N \leftarrow$
$s(N1) \leq s(N2) \leftarrow N1 \leq N2$
$nat(0) \leftarrow$
$nat(s(X)) \leftarrow nat(X)$
λ_3: $g \leftarrow \boxed{nat(X) \wedge \neg X \leq s(X)}$
$f \leftarrow \neg g$

$Defs_4$: δ_1: $\boxed{g} \leftarrow \boxed{nat(N) \wedge \neg N \leq s(N)}$
δ_2: $f \leftarrow \neg g$

replace

folding λ_3 wrt $\boxed{nat(X) \wedge \neg X \leq s(X)}$

P_5: $0 \leq N \leftarrow$
$s(N1) \leq s(N2) \leftarrow N1 \leq N2$
$nat(0) \leftarrow$
$nat(s(X)) \leftarrow nat(X)$
λ_4: $g \leftarrow \boxed{g}$
$f \leftarrow \neg g$

R3. Folding

Given a clause in P_k $\quad\lambda$: $H \leftarrow G_1 \wedge \boxed{G_2}\theta \wedge G_3$

and a definition in $Defs_k$ $\quad\delta$: $\boxed{Newp} \leftarrow \boxed{G_2}$

if (1) $\theta = \theta_1 \circ \theta_2$ where:

- θ_1 and θ_2 share no variables
- θ_2 is a renaming of the existential variables of δ

and (2) δ has been (or will be) unfolded w.r.t. a positive literal (*)

then $\boxed{P_{k+1} = (P_k \setminus \{\lambda\}) \cup \{H \leftarrow G_1 \wedge \boxed{Newp}\,\theta \wedge G_3\}}$

Similar to [Tamaki-Sato 84, Seki 91], except for (*)

Folding: Condition (2)

P_0: $p(X) \leftarrow p(X)$
$p(X) \leftarrow q$
$q \leftarrow fail$

P_1: $newp \leftarrow \neg\, p(X)$ definition introduction

P_2: $newp \leftarrow \neg\, p(X) \wedge \neg\, q$ unfolding wrt $\neg\, p(X)$

P_3: $newp \leftarrow newp \wedge \neg\, q$ folding

$newp \in M(P_0 \cup Defs_3) = M(P_1)$ and $newp \notin M(P_3)$

Tautologies: Less-or-Equal Example

P_5: $0 \leq N \leftarrow$

$s(N1) \leq s(N2) \leftarrow N1 \leq N2$

$nat(0) \leftarrow$

$nat(s(X)) \leftarrow nat(X)$

$\cancel{g \leftarrow g}$

$f \leftarrow \neg g$

$H \leftarrow H \wedge G$ is a tautology

Finally, by unfolding $f \leftarrow \neg g$ we get (there is no clause for g):

P_7: $0 \leq N \leftarrow$

$s(N1) \leq s(N2) \leftarrow N1 \leq N2$

$nat(0) \leftarrow$

$nat(s(X)) \leftarrow nat(X)$

$f \leftarrow$

$f \leftarrow$ belongs to P_7

$\Leftrightarrow \quad f \in M(P_7)$

$\Leftrightarrow \quad M(P_0) \models \forall N\, (nat(N) \rightarrow N \leq s(N))$

Tautologies

$$P_{k+1} = (P_k \setminus Cs) \cup Ds$$

where $Cs \Rightarrow Ds$ is an instance of one of the following rewritings:

$$\{H \leftarrow A \wedge \neg A \wedge G\} \quad \Rightarrow \quad \emptyset$$

$$\{H \leftarrow H \wedge G\} \quad \Rightarrow \quad \emptyset$$

$$\{H \leftarrow G_1 \wedge L_1 \wedge L_2 \wedge G_2\} \quad \Rightarrow \quad \{H \leftarrow G_1 \wedge L_2 \wedge L_1 \wedge G_2\}$$

$$\{H \leftarrow L \wedge L \wedge G\} \quad \Rightarrow \quad \{H \leftarrow L \wedge G\}$$

$$\{H \leftarrow G_1,\ H \leftarrow G_1 \wedge G_2\} \quad \Rightarrow \quad \{H \leftarrow G_1\}$$

$$\{H \leftarrow A \wedge G_1 \wedge G_2,\ H \leftarrow \neg A \wedge G_1\} \quad \Rightarrow \quad \{H \leftarrow G_1 \wedge G_2,\ H \leftarrow \neg A \wedge G_1\}$$

A Derived Rule: Propositional Simplification

Suppose that in P_k a predicate p depends on nullary predicates only.
Then $p \in M(P_k)$ is decidable and,by unfolding and tautologies,

if $p \in M(P_k)$ then $\boxed{P_{k+1} = (P_k \setminus D_p) \cup \{p \leftarrow\}}$

if $p \notin M(P_k)$ then $\boxed{P_{k+1} = (P_k \setminus D_p)}$

where D_p is the set of clauses in P_k with head p

Correctness of the Unfold/Fold rules

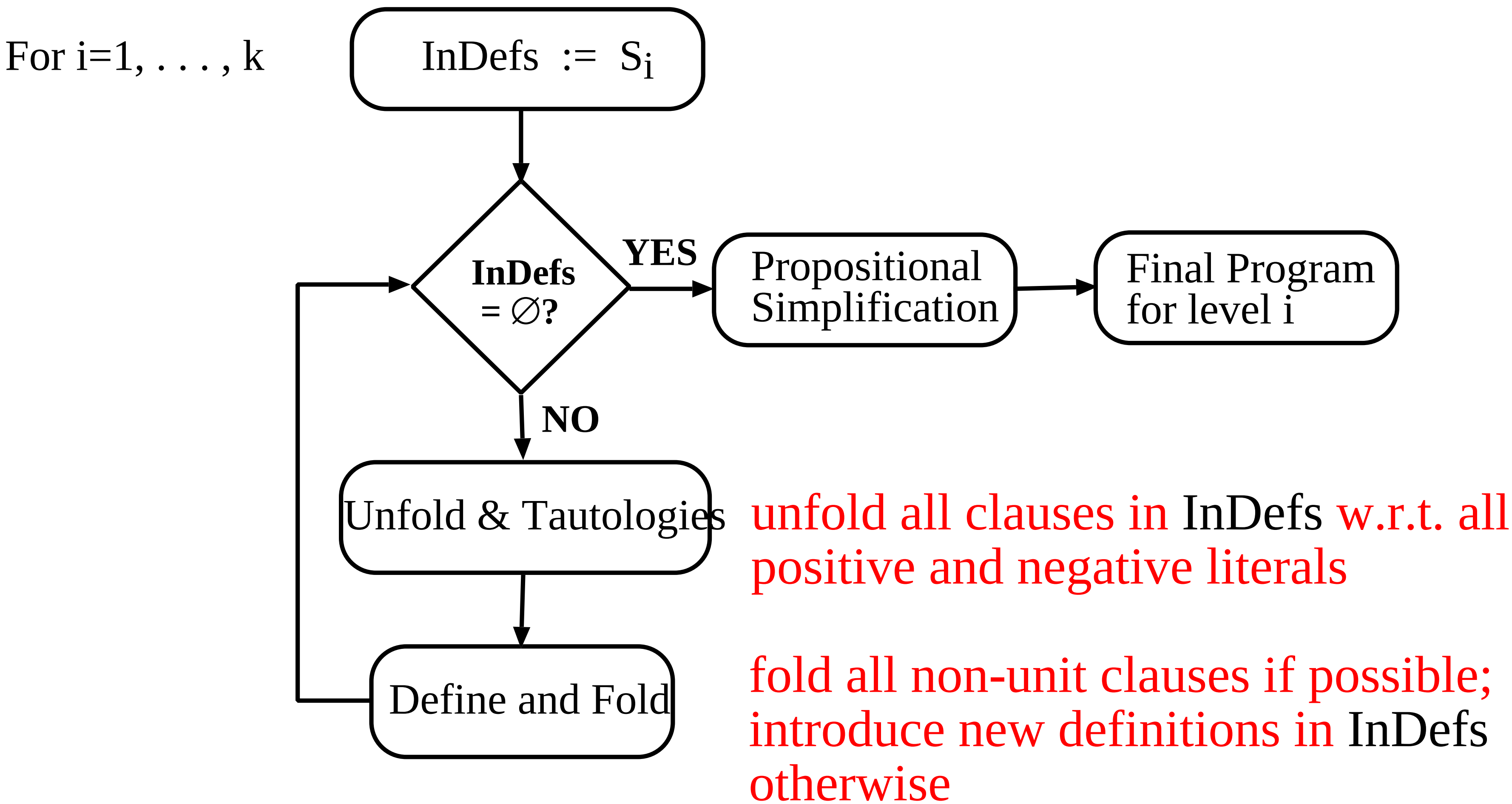
Theorem: Let $P_0, \ldots, P_n$ be a transformation sequence. Let $Defs_n$ be the set of definitions introduced in that sequence. Then

$$\boxed{M(P_0 \cup Defs_n) = M(P_n)}$$

The Unfold/Fold Transformation Strategy

$P \cup Cls(f,\varphi) = S_0 \cup \ldots \cup S_k$ is a finite partition into levels where:

- $S_0 = P$
- the predicates in S_i depend only on predicates in $S_0 \cup \ldots \cup S_{i-1}$



Termination

Theorem. For all WS1S formulas φ, the unfold/fold strategy $Natset \cup Cls(f,\varphi)$ terminates and the final program T contains either $f \leftarrow$ or no clause for f.

Proof. Only a finite number of new definitions are generated (no generalization is needed).

By construction, all clauses in T are definite clauses of the form:

$$p(t_1, \ldots, t_k) \leftarrow q_1(X_1, \ldots, X_u) \wedge \ldots \wedge q_r(X_v, \ldots, X_w)$$

where: $t_i := 0 \mid s(N) \mid [\,] \mid [B|S]$ and

$X_1, \ldots, X_i, \ldots, X_j, \ldots, X_s$ are distinct variables occurring in $t_1, \ldots, t_k$

In particular, if p is 0-ary, then we derive a set of propositional clauses for p, and by the propositional simplification the final program T contains either $p \leftarrow$ or no clause for p.

The Transformation Strategy (Example 1)

Cls(f,φ):	$f \leftarrow \neg\, g$	**level 3**
	$g \leftarrow nat(N1) \wedge \neg\, h(N1)$	**level 2**
	$h(N1) \leftarrow \boxed{nat(N1) \wedge nat(N2) \wedge N1 \leq N2}$	**level 1**

Bottom up over levels:

$h(N1) \leftarrow \boxed{nat(N1) \wedge nat(N2) \wedge N1 \leq N2}$ **level 1**

$h(0) \leftarrow$
$h(s(N1)) \leftarrow \boxed{nat(N1) \wedge nat(N2) \wedge N1 \leq N2}$ **unfolding**

$h(0) \leftarrow$
$h(s(N)) \leftarrow \boxed{h(N)}$ **folding**

The Transformation Strategy (Example 2)

$g \leftarrow \boxed{nat(N1) \wedge \neg h(N1)}$ **level 2**

$g \leftarrow \boxed{nat(N1) \wedge \neg h(N1)}$ **unfolding**

~~$g \leftarrow \boxed{g}$~~ **folding**

tautology

$f \leftarrow \neg g$ **level 3**

$f \leftarrow$ **unfolding**

$\Rightarrow$ M(Natset $\cup$ Cls(f,φ)) |= f

$\Rightarrow$ by the correctness of the encoding and the correctness of the transformations,

$$\models_{WS1S} \varphi$$

Program Synthesis from WS1S Formulas

Example (Maximum of a set).

$$\varphi \equiv \underline{N} \in \underline{S} \wedge \neg\exists N1\ (N1 \in \underline{S} \wedge \neg N1 \leq \underline{N})$$

free variables

Apply the Lloyd-Topor transformation starting from the statement:

$$max(S,N) \leftarrow N \in S \wedge \neg\ \boxed{\exists N1\ (N1 \in S \wedge \neg N1 \leq N)}$$

and add types

Cls(max,φ):

$$max(S,N) \leftarrow \underline{set(S)} \wedge \underline{nat(N)} \wedge N \in S \wedge \neg\ \boxed{newp(S,N)}$$

$$newp(S,N) \leftarrow \underline{set(S)} \wedge \underline{nat(N)} \wedge \underline{nat(N1)} \wedge N1 \in S \wedge \neg N1 \leq N$$

. . . Program Synthesis from WS1S Formulas

Apply the unfold/fold transformation strategy starting from:

Natset $\cup$ Cls(max, φ)

and derive a program for computing the maximum of a set:

max([**t**|S],0) ← new1(S)
max([**t**|S],s(N)) ← max(S,N)
max([**f**|S],s(N)) ← max(S,N)
new1([ ]) ←
new1([**f**|S]) ← new1(S)

The Unfold/Fold Synthesis Method

Let φ be a WS1S formula with free variables $X_1, \ldots, X_n$.

Step 1. (Encoding into stratified LP)

$\varphi \longrightarrow Cls(f,\varphi)$ Lloyd-Topor transformation + type addition

for all ground terms $t_1, \ldots, t_n$,

$\models_{WS1S} \varphi\{X_1/t_1, \ldots, X_n/t_n\}$ iff $M(Natset \cup Cls(f,\varphi)) \models f(t_1, \ldots, t_n)$

Step 2. (Unfold/fold transformations)

$Natset \cup Cls(f,\varphi) \longrightarrow \cdots \longrightarrow T$ unfold/fold strategy

for all ground terms $t_1, \ldots, t_n$,

$M(Natset \cup Cls(f,\varphi)) \models f(t_1, \ldots, t_n)$ iff $M(T) \models f(t_1, \ldots, t_n)$

Optimizations: (1) Discarding useless types

Some type atoms can be discarded.

Example:

$$\varphi \equiv N \in S \wedge \neg\exists N1\ (N1 \in S \wedge \neg N1 \leq N)$$

Cls(max,φ):

$$\begin{cases} max(S,N) \leftarrow set(S) \wedge \cancel{nat(N)} \wedge N \in S \wedge \neg\ newp(S,N) \\ newp(S,N) \leftarrow set(S) \wedge nat(N) \wedge \cancel{nat(N1)} \wedge N1 \in S \wedge \neg N1 \leq N \end{cases}$$

because $\quad M(Natset) \models \forall N\ \forall S\ (N \in S \rightarrow nat(N))$

Optimizations: (2) Determinization

The program derived by the u/f strategy may be nondeterministic

$p(s(X)) \leftarrow q(X)$
$p(s(X)) \leftarrow r(X)$
$q(0) \leftarrow$
$q(s(X)) \leftarrow r(X)$
$r(s(X)) \leftarrow q(X)$

$new(X) \leftarrow \boxed{q(X)}$
$new(X) \leftarrow \boxed{r(X)}$

multiple clause definition

$new(0) \leftarrow$
$new(s(X)) \leftarrow \boxed{r(X)}$
$new(s(X)) \leftarrow \boxed{q(X)}$

unfolding

$new(0) \leftarrow$
$new(s(X)) \leftarrow \boxed{new(X)}$

multiple clause folding

Optimizations: (3) Minimization

The programs derived during the unfold/fold strategy may contain equivalent predicates.

$$
\begin{aligned}
&p(0) \leftarrow \\
&p(s(X)) \leftarrow \boxed{q(X)} \\
&q(0) \leftarrow \\
&q(s(X)) \leftarrow \boxed{p(X)}
\end{aligned}
$$

p and q have the same definition modulo predicate names.

$$M(P) \models \forall X (p(X) \leftrightarrow q(X))$$

$$
\begin{aligned}
&p(0) \leftarrow \\
&p(s(X)) \leftarrow \boxed{p(X)} \\
&q(0) \leftarrow \\
&q(s(X)) \leftarrow \boxed{q(X)}
\end{aligned}
$$

goal replacements

Deletion of Useless Clauses

Suppose that P is a definite program.

$$P_k \quad \rightarrow \quad Prop(P_k)$$

$$p(\ldots) \leftarrow q(\ldots) \qquad p \leftarrow q$$

if $p \notin M(Prop(P_k))$ then $\boxed{P_{k+1} = (P_k \setminus D_p)}$

where D_p is the set of clauses in P_k with head p

Example: $P_k \quad \rightarrow \quad Prop(P_k)$

$$p(s(X)) \leftarrow q(X) \qquad p \leftarrow q$$

$$q(s(X)) \leftarrow p(X) \qquad q \leftarrow p$$

the clauses for p and q can be deleted

Incremental Verification

To prove the WS1S formula:

$$\forall W\ \forall U\ \forall W1\ \forall U1\ (use(\langle W,U\rangle,\langle W1,U1\rangle) \rightarrow \neg empty(U1))$$

where:

$$use(\langle W,U\rangle,\langle W1,U1\rangle) \equiv$$

$$\exists N\ (N \in W \wedge \exists Z\ (Z = W \cup U \wedge min(Z,N)) \wedge W1 = W - \{N\} \wedge U1 = U \cup \{N\})$$

Synthesize programs for:

- set union
- min
- set difference
- singleton
- empty set
- use

Add the synthesized programs to Natset and prove the WS1S formula in the derived program

Implementation

The unfold/fold proof and synthesis methods have been implemented on the MAP transformation system, available at

http://www.iasi.rm.cnr.it/~proietti/system.html

Reasonable efficiency for small formulas.

Program Verification using CLP

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 $Cls(f,\varphi)$

s. t. $M(P) \models \varphi$ iff $M(P \cup Cls(f,\varphi)) \models f$

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 $M(P) \models \varphi$)

or no clause for f is in T (in which case $M(P) \not\models \varphi$)

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)

Program Synthesis

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

a first order formula $\varphi[X]$

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

$\forall$ ground term t

$M(P) \models \varphi[t]$ iff $M(S) \models s(t)$

---

1. Apply Lloyd-Topor transformation starting from :

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

and derive a possibly inefficient, locally stratified program $Cls(s,\varphi)$
s.t. $\forall$ ground term t, $M(P) \models \varphi[t]$ iff $M(P \cup Cls(s,\varphi)) \models s(t)$

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

[Kowalski's Festschrift 2002]

Program P :

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

$\varphi[L,M]$: $list(L) \wedge member(M,L) \wedge \forall X\ (member(X,L) \rightarrow X \leq M)$

We start from :

$max(L,M) \leftarrow list(L) \wedge member(M,L) \wedge \forall X(member(X,L) \rightarrow X \leq M)$

After Lloyd-Topor we get :

6. $max(L,M) \leftarrow list(L) \wedge member(M,L) \wedge \neg\, new1(L,M)$
7. $new1(L,M) \leftarrow member(X,L) \wedge \neg\, X \leq M$

Synthesized Program :

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