Combining Logic Programs and Monadic Second Order Logics by Program Transformation

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

Lopstr'02 - Madrid, Spain

September 17-20, 2002

Motivations

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]

MSO logics cannot express many properties of infinite state systems. Combinations of MSO provers and general purpose theorem provers have been proposed. [Basin-Friedrich 00, Fioravanti et al. 01]

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

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

- synthesizing definite logic programs from WS1S specifications, thereby yielding a method for combining WS1S and LP.

Combining WS1S and LP

Specifying the reachable states of a multiprocess system.

The number of processes of the system may change dinamically.
A state of the system is represented by a set of processes.
A process is identified by a natural number.

$$reach(S) \leftarrow init(S)$$

$$reach(S) \leftarrow create(S1,S) \wedge reach(S1)$$

$$reach(S) \leftarrow use(S1,S) \wedge reach(S1)$$

$$reach(S) \leftarrow release(S1,S) \wedge reach(S1)$$

where init, create, use, release are specified by WS1S formulas, e.g.,

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

The predicate reach cannot be expressed by a WS1S formula.

Goals of this work

Automatic methods based on unfold/fold transformations for

- proving WS1S formulas, such as:

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

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

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

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

Overview

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

WS1S

The Weak Monadic Second Order theory of 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

Interpretation $\mathfrak{I}$

- Domain of $\mathfrak{I}$: $\quad \mathrm{Nat} \cup \mathrm{P}_{\mathrm{fin}}(\mathrm{Nat}) \quad$ where $\mathrm{Nat} = \{0, 1, 2, \ldots\}$
- 0 is interpreted as the zero of Nat
  s(_) is interpreted as the successor function +1
  $\in$ is interpreted as the membership relation on $\mathrm{Nat} \times \mathrm{P}_{\mathrm{fin}}(\mathrm{Nat})$
- $\models_{\mathrm{WS1S}} \varphi$ iff $\mathfrak{I} \models \varphi$

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

WS1S . . .

WS1S specifications (i.e. open formulas)

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

order over numbers:

S is 'downwards' closed

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

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 exixts no finite set which is nonempty and upwards closed:

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

An Encoding of WS1S into LP

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

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

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

{0,3,4} is represented by [**y**,**n**,**n**,**y**,**y**] and also by [**y**,**n**,**n**,**y**,**y**,**n**]

∅ is represented by [ ] and also by [**n**,**n**,**n**,**n**]

Program Member

nat(0) ←

nat(s(N)) ← nat(N)

set([ ]) ←

set([**y**|S]) ← set(S)

set([**n**|S]) ← set(S)

0 ≤ N ←

s(N1) ≤ s(N2) ← N1 ≤ N2

0 ∈ [**y**|S] ←

s(N) ∈ [B|S] ← N ∈ S

. . . An Encoding WS1S into LP

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 $f \leftarrow \psi$

$f \leftarrow \neg\ \exists N1 \neg\ \exists N2\ N1 \leq N2$

$f \leftarrow \neg\ g$

$g \leftarrow \neg\ h(N1)$

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

Add types

Cls(f,φ):

$f \leftarrow \neg\ g$

$g \leftarrow \underline{nat(N1)} \wedge \neg\ h(N1)$

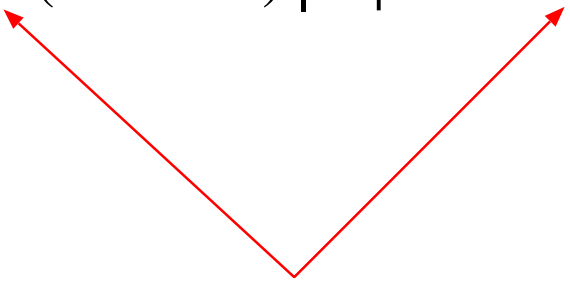
$h(N1) \leftarrow \underline{nat(N1) \wedge nat(N2)} \wedge N1 \leq N2$

Stratified program

Correctness of the Encoding

Theorem

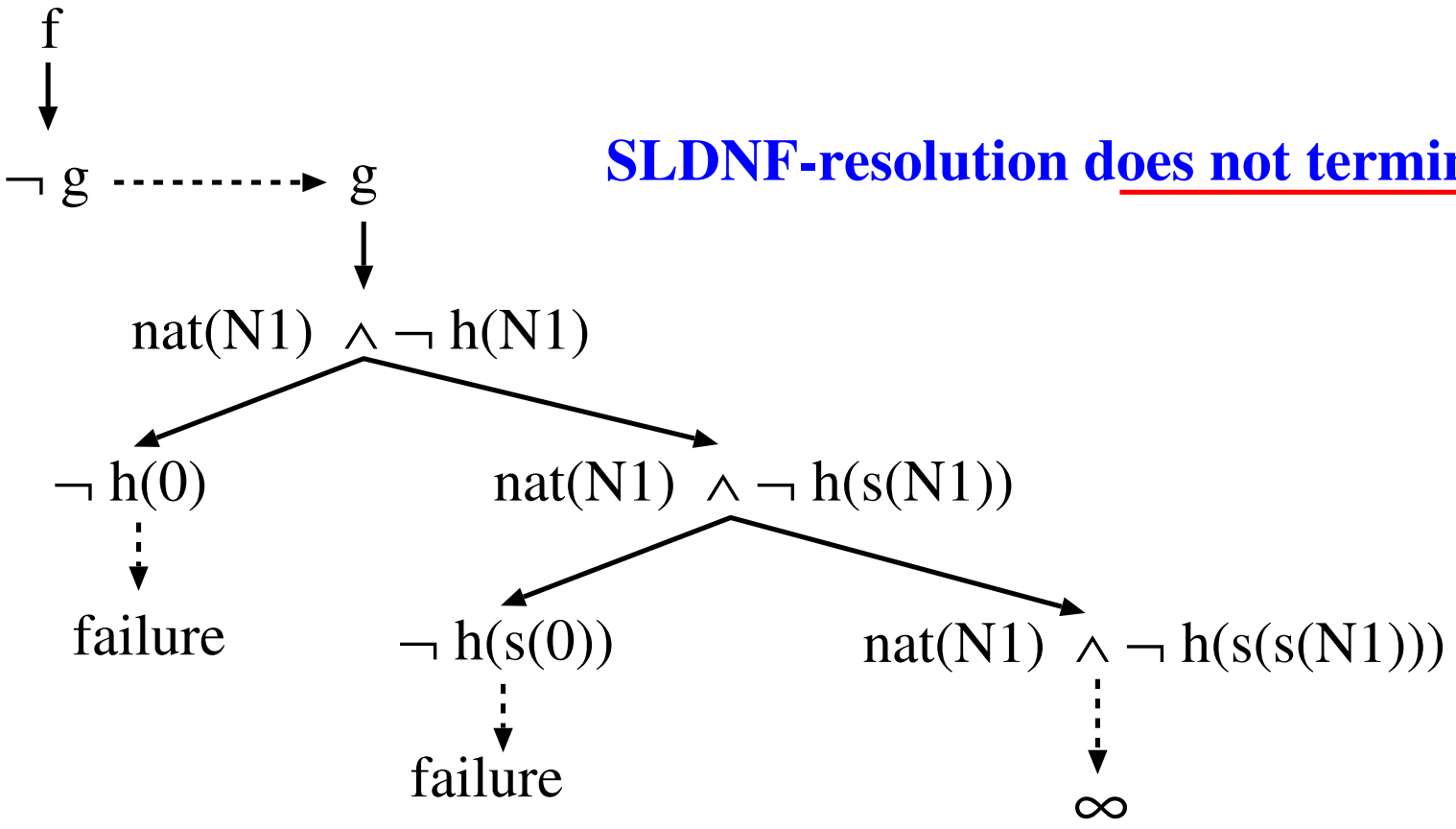
$\models_{WS1S} \varphi$ iff $M(\text{Member}) \models \varphi$ iff $M(\text{Member} \cup \text{Cls}(f,\varphi)) \models f$



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

Proving WS1S Formulas via SLDNF

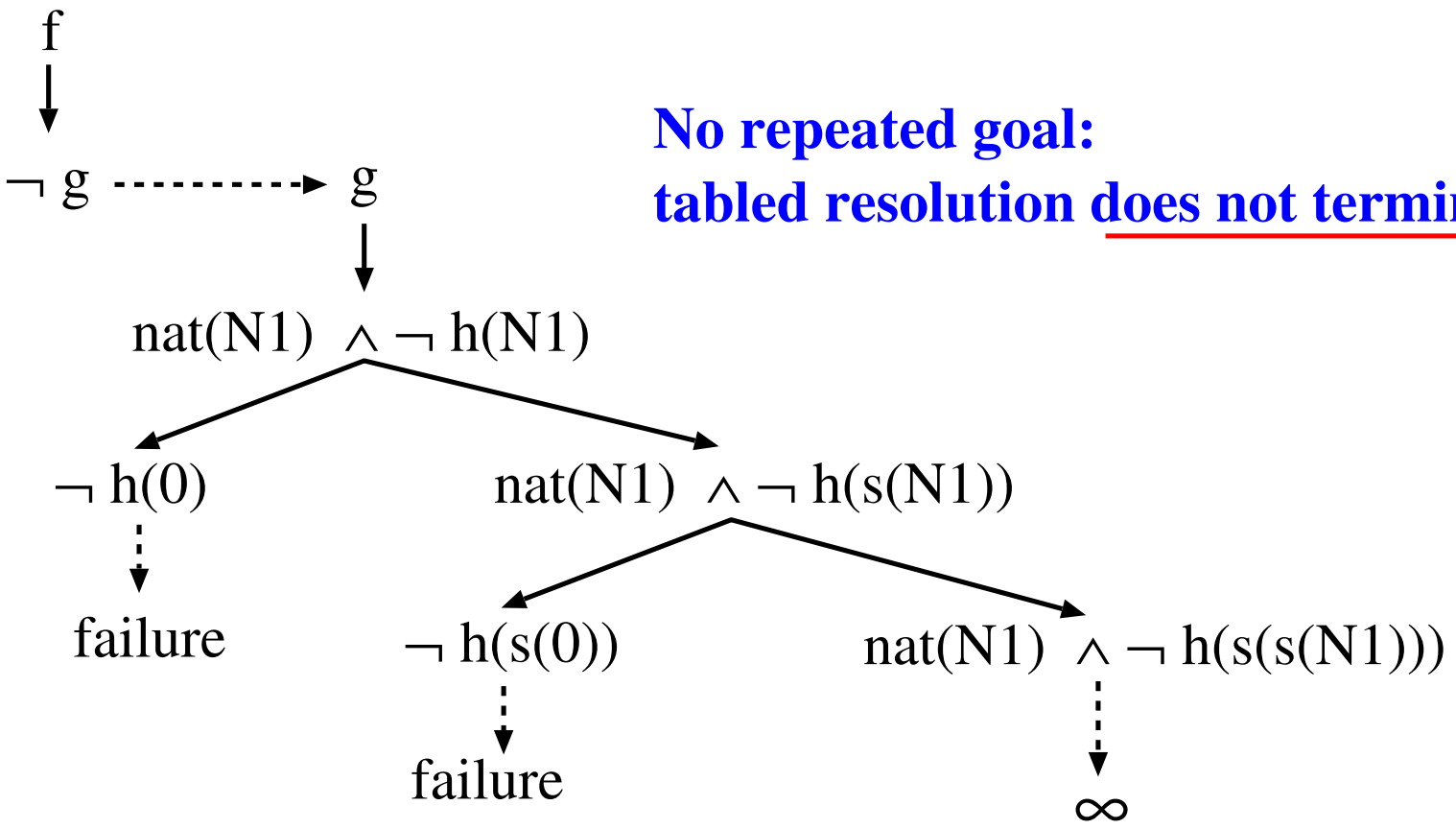
$$
\begin{aligned}
& f \leftarrow \neg\, g \\
\text{Cls}(f,\varphi):\ & g \leftarrow \text{nat}(N1) \wedge \neg\, h(N1) \\
& h(N1) \leftarrow \text{nat}(N1) \wedge \text{nat}(N2) \wedge N1 \leq N2
\end{aligned}
$$



SLDNF-resolution does not terminate

Proving WS1S Formulas via Tabled Resolution

$$
\begin{array}{ll}
 & f \leftarrow \neg\, g \\
\text{Cls}(f,\varphi): & g \leftarrow \text{nat}(N1) \wedge \neg\, h(N1) \\
 & h(N1) \leftarrow \text{nat}(N1) \wedge \text{nat}(N2) \wedge N1 \leq N2
\end{array}
$$



No repeated goal:
tabled resolution does not terminate

Proving WS1S Formulas via Program Transformation

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

Bottom up over strata:

$h(N1) \leftarrow nat(N1) \wedge nat(N2) \wedge N1 \leq N2$ — **stratum 1**

$h(0) \leftarrow$

$h(s(N1)) \leftarrow nat(N1) \wedge nat(N2) \wedge N1 \leq N2$ — **unfolding**

$h(0) \leftarrow$

$h(s(N)) \leftarrow h(N)$ — **folding**

. . . Proving WS1S via Program Transformation

$g \leftarrow \text{nat}(N1) \wedge \neg\, h(N1)$ — **stratum 2**

$g \leftarrow \text{nat}(N1) \wedge \neg\, h(N1)$ — **unfolding**

$g \leftarrow g$ — **folding**

g **is false in the perfect model**

$f \leftarrow \neg\, g$ — **stratum 3**

$f \leftarrow$ — **unfolding**

$\Rightarrow$ $M(\text{Member} \cup \text{Cls}(f,\varphi)) \models f$

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

$$\models_{\text{WS1S}} \varphi$$

The Unfold/Fold Proof Method

Let φ be a closed WS1S formula.

Step 1. (Encoding into stratified LP)

$\varphi \longrightarrow Cls(f,\varphi)$

Lloyd-Topor transformation + type addition

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

Step 2. (Unfold/fold transformations)

$Member \cup Cls(f,\varphi) \rightarrow \cdots \rightarrow T$

unfolding and folding rules applied according to a strategy

$M(Member \cup Cls(f,\varphi)) \models f$ 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

$$P_0 \rightarrow \ \ldots \rightarrow P_n$$

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

- The transformation rules preserve the Perfect Model.

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

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

The Transformation Rules

R1. Definition Introduction
R2. Unfolding (w.r.t. positive or negative literals)
R3. Folding (w.r.t. conjunctions of literals)
R4. Propositional Simplification

We also apply the following transformations (not called 'transformation rules'):

- **renaming of variables**
- **reordering of conjuncts:** $\varphi_1 \wedge \varphi_2$ **is transformed into** $\varphi_2 \wedge \varphi_1$
- **removal of duplicate conjuncts:** $\varphi \wedge \varphi$ **is transformed into** φ

R1. Definition Introduction

[Tamaki-Sato 84, Seki 91]

Introduce a new definition, that is, a clause

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

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

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

$\mathbf{Defs_k}$ **is the set of definitions introduced up to step** k

---

No recursive definitions, no multiple clause definitions

R2$^+$. Unfolding w.r.t. a positive literal

[Tamaki-Sato 84, Seki 91]

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

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

If m=0 then delete λ fromP_k

R2⁻. Unfolding w.r.t. a negative literal

[P&P, CL2000]

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

• $Body_1, \ldots, Body_m$ have no existential variables

then take the disjunctive normal form

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

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

If m=0 then delete ¬A from the body of λ; if Aθ ← is a fact in P_k then delete λ.

. . . R2$^-$. Unfolding w.r.t. a negative literal

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

$p(0) \leftarrow$

$p(s(X)) \leftarrow q_1(X)$

$p(X) \leftarrow q_2(X) \wedge q_3(X)$

$h(X) \leftarrow nat(X) \wedge \neg\, (q_1(X) \vee (q_2(s(X)) \wedge q_3(s(X))))$

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

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

R3. Folding

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

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

if
- $\theta = \theta_1 \circ \theta_2$ **where:**
  - θ_1 **and** θ_2 **share no variables**
  - θ_2 **is a renaming of the existential variables of** δ

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

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

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

. . . R3. Folding

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

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

P_2: $newp(X) \leftarrow \neg\, p(X) \wedge \neg\, q(X)$ unfolding ($R2^{-}$)

P_3: $newp(X) \leftarrow newp(X) \wedge \neg\, q(X)$ folding

$newp(0)$ true in P_1 and false in P_3

R4. Propositional Simplification

Suppose that in P_k the predicate p is defined by a set D_p of propositional clauses.

If $M(D_p) \models p$ **then** $P_{k+1} = (P_k - D_p) \cup \{p \leftarrow\}$

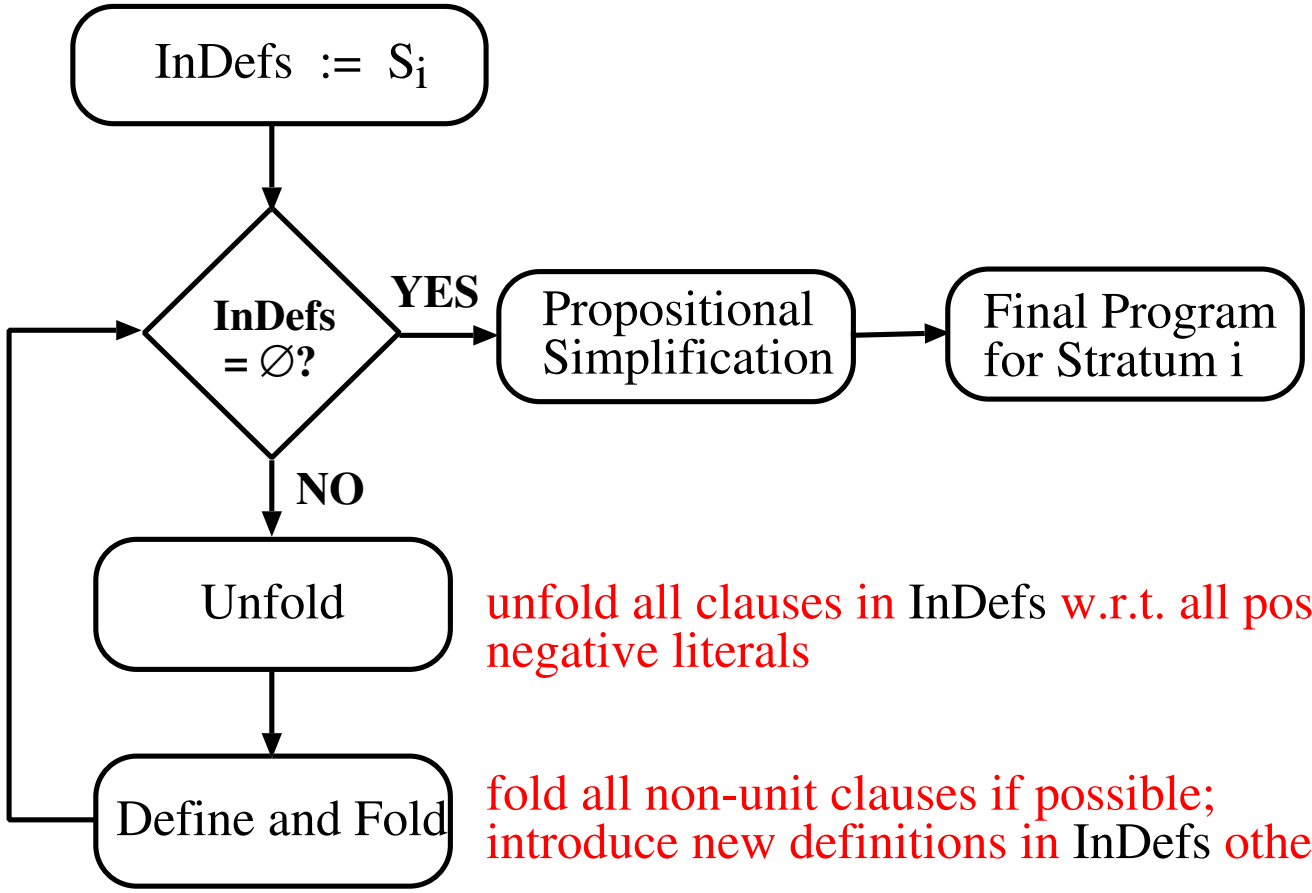
If $M(D_p) \models \neg p$ **then** $P_{k+1} = (P_k - D_p)$

The Unfold/Fold Transformation Strategy

Member $\cup$ Cls(f,φ) = $S_0 \cup \ldots \cup S_k$ a stratification where:

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

For i=1, . . . , k



unfold all clauses in InDefs w.r.t. all positive and negative literals

fold all non-unit clauses if possible; introduce new definitions in InDefs otherwise

Termination

Theorem. For all WS1S formulas φ, the unfold/fold strategy Member $\cup$ Cls(f,φ) 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.

Program Synthesis from WS1S Formulas

Example (Maximum of a set).

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

free variables

Apply the Lloyd-Topor transformation starting from the statement:

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

and add types

Cls(max,φ):

$$\max(S,N) \leftarrow set(S) \wedge nat(N) \wedge N \in S \wedge \neg\ newp(S,N)$$

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

. . . Program Synthesis from WS1S Formulas

Apply the unfold/fold transformation strategy starting from:

$$\text{Member} \cup \text{Cls}(\max, \varphi)$$

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

$$\max([\mathbf{y}|S],0) \leftarrow \text{new1}(S)$$
$$\max([\mathbf{y}|S],s(N)) \leftarrow \max(S,N)$$
$$\max([\mathbf{n}|S],s(N)) \leftarrow \max(S,N)$$
$$\text{new1}([\,]) \leftarrow$$
$$\text{new1}([\mathbf{n}|S]) \leftarrow \text{new1}(S)$$

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 $M(Member \cup Cls(f,\varphi)) \models f(t_1, \ldots, t_n)$

Step 2. (Unfold/fold transformations)

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

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

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

Optimizations: (1) Discarding useless types

Example:

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

Cls(max,φ):

$$\max(S,N) \leftarrow set(S) \wedge nat(N) \wedge N \in S \wedge \neg\ newp(S,N)$$

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

Some type atoms can be discarded.

Optimizations: (2) Subsumption

$H \leftarrow G_1$

$(H \leftarrow G_1 \wedge G_2)\,\theta$

. . . but the subsumption test may be costly

Optimizations: (3) Determinization

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

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

$new(X) \leftarrow q(X)$
$new(X) \leftarrow r(X)$

multiple clause definition

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

unfolding

$new(0) \leftarrow$
$new(s(X)) \leftarrow new(X)$

multiple clause folding

Optimizations: (4) Minimization

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

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

p and q have the same definition modulo predicate names.

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

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

goal replacements

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.