A Note on the Reuse of the Results of a Termination Analysis Based on Polymorphic Types

Maurice Bruynooghe *Michael Codish*
Samir Genaim *Wim Vanhoof*

Report CW 383, September 2003

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Note on the Reuse of the Results of a Termination Analysis Based on Polymorphic Types

Maurice Bruynooghe *Michael Codish*
Samir Genaim *Wim Vanhoof*

Report CW 383, September 2003

Department of Computer Science, K.U.Leuven

Abstract

This short report complements a paper published in the proceedings of SAS 2002. It gives an improved version of the main theorem and a detailed sketch of a proof.

The original paper was about the use of polymorphic type-based norms in termination analysis of logic programs.

Keywords : termination analysis.
CR Subject Classification : D.3, F.3.2,

A Note on the Reuse of the Results of a Termination Analysis Based on Polymorphic Types

Maurice Bruynooghe Michael Codish Samir Genaim
Wim Vanhoof

September 2003

Abstract

This short report complements a paper published in the proceedings of SAS 2002. It gives an improved version of the main theorem and a detailed sketch of a proof.

The original paper was about the use of polymorphic type-based norms in termination analysis of logic programs.

1 Introduction

This note addresses a problem in the SAS2002 paper [1] of the same authors. Peter Stuckey pointed out to us that the result claimed in Theorem 1 not always holds. Analysis revealed that we overlooked the possibility that $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) \neq \emptyset$ in which case there is undesirable interference between the σ'_i and the σ'_j abstractions in the formula obtained through reuse of the result of the polymorphic analysis. The overlooked problem is solved by introducing extra constituents in the polymorphic type such that $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) = \emptyset$ holds.

A minor problem, observed when working out the details of the proof is that, even when $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) = \emptyset$, the formula obtained through reuse is not necessarily equivalent with the formula obtained by using the types of the instance. In fact, a stronger result holds: The formulae obtained through reuse imply the formulae obtained by a direct analysis using the type instance (while still being correct).

In what follows we provide a new version for this theorem and prove it's correctness.

2 The setting

In the rest of this note we assume that the types are defined by a polymorphic type definition ρ with polymorphic parameters T_i. We assume also that p is

imported in another program component Q imposing a monomorphic type instance $\rho' = \rho[\ldots, T_i \mapsto \tau_i, \ldots]$ for p. Note that p is typed by ρ as well as by ρ'. We enumerate the (*polymorphic*) constituents of ρ and the (*monomorphic*) constituents of ρ' as follows:

$$C(\rho) = \{\sigma_1, \ldots, \sigma_k, \ldots, T_i, \ldots\} \quad \textit{and} \quad C(\rho') = \{\sigma'_1, \ldots, \sigma'_l\}$$

For two types τ_1 and τ_2, if τ_1 is a constituents of τ_2 we write $\tau_1 \preceq \tau_2$, and if it is a strict constituents we write $\tau_1 \prec \tau_2$.

2.1 Normal Form

For the sake of simplicity we assume the program is given in the following normal form:

- facts (sometime we call them base cases) can be of the form:
  - $p(x, y) \leftarrow x = y$
  - $p(x, y_1, \ldots, y_n) \leftarrow x = f(y_1, \ldots, y_n)$
- clauses are of the form $p(\bar{x_0}) \leftarrow p_1(\bar{x_1}), \ldots, p_n(\bar{x_n})$ where each $\bar{x}_i$ is a tuple of variable and in particular $\bar{x}_0$ a tuple of distinct variables.

Note that any program can be transformed to this normal form.

2.2 Abstraction

Given a program in the above normal form, its abstraction to Pos with respect to a set of constituents $\mathcal{C} = \{\sigma_1, \ldots, \sigma_l\}$ is done as follows:

Base Case I: The abstraction of a fact of the form $p(x, y) \leftarrow x = y$ is:

$$p(x^{\sigma_1}, y^{\sigma_1}, \ldots, x^{\sigma_l}, y^{\sigma_l}) \leftarrow \\ X_g^{\mathcal{C}} \wedge \alpha^{\mathcal{C}}(x^{\sigma_1} = y^{\sigma_1}) \wedge \ldots \wedge \alpha^{\mathcal{C}}(x^{\sigma_l} = y^{\sigma_l})$$

where $X_g^{\mathcal{C}}$ is a conjunction of the variables such that x^{σ_j} (y^{σ_j}) in the conjunction iff $\sigma_j \not\preceq \mathit{type}(x)$. In other words, those variables of size zero are assumed to be rigid. $\alpha^{\mathcal{C}}(x^{\sigma_j} = y^{\sigma^j})$ is equal to $x^{\sigma_j} \leftrightarrow y^{\sigma_j}$ if $\sigma_j \preceq \mathit{type}(x)$; otherwise true.

Base Case II The abstraction of a fact of the form $p(x, y_1, \ldots, y_n) \leftarrow x = f(y_1, \ldots, y_n)$ is:

$$p(x^{\sigma_1}, y_1^{\sigma_1}, \ldots, y_n^{\sigma_1}, \ldots, x^{\sigma_l}, y_1^{\sigma_l}, \ldots, y_n^{\sigma_l}) \leftarrow \\ X_g^{\mathcal{C}} \wedge \alpha^{\mathcal{C}}(x^{\sigma_1} = f(y_1^{\sigma_1}, \ldots, y_n^{\sigma_1})) \wedge \ldots \wedge \alpha^{\mathcal{C}}(x^{\sigma_l} = f(y_1^{\sigma_l}, \ldots, y_n^{\sigma_l}))$$

where $X_g^{\mathcal{C}}$ as for the base case I. $\alpha^{\mathcal{C}}(x^{\sigma_j} = f(y_1^{\sigma_j}, \ldots, y_n^{\sigma_j}))$ is equal to:

$$x^{\sigma_j} \leftrightarrow \wedge\{y_i^{\sigma_j} | \sigma_j \preceq \mathit{type}(y_i)\}$$

if $\sigma_j \preceq \mathit{type}(x)$; otherwise true.

Clauses: The abstraction of a clause $p(\bar{x}_0) \leftarrow p_1(\bar{x}_1), \ldots, p_n(\bar{x}_n)$ is the abstract clause:

$$p(\bar{X}_0^{\mathcal{C}}) \leftarrow p_1(\bar{X}_1^{\mathcal{C}}), \ldots, p_n(\bar{X}_n^{\mathcal{C}})$$

where $\bar{X}_i^{\mathcal{C}}$ is a combination of the variables of $\bar{X}_i$ with respect to the constituents (of) $\mathcal{C}$. When we want to refer to the set of variables in a tuple $\bar{X}$ we write X.

3 Reuse of polymorphic results

Definition 3.1 (T'-abstraction) *For a type definition ρ with a polymorphic parameter T_i we define the T_i'-abstraction as a regular type-based abstraction where $T_i' \neq T_i$ is a fresh type symbol under the assumption that $T_i' \prec T_i$.* □

For the polymorphic type based analysis, we extend the constituents of ρ to include $\{T'_{i,\sigma'} | \sigma' \neq \tau_i \in C(\tau_i)\}$ where each $T'_{i,\sigma'}$ is considered as a T_i' constituent. Note that this assumes τ is know; in fact, the assumption is that one knows a finite upper bound on the number of constituents of τ. So, in a sense it would be preferable to name the extra constituents $T_1', \ldots T_m'$ but that would make the next definition more complex.

Definition 3.2 *Let ρ be a polymorphic type definition and $\rho' = \rho[\ldots, T_i \mapsto \tau_i, \ldots]$ a monomorphic instance of ρ. Let T_i' be a type variable not occurring in ρ which is treated as a proper constituent of T_i ($T_i' \preceq T_i$ and $T_i' \neq T_i$). We denote by $\mathcal{I}(\sigma')$ the type constituents in ρ that correspond to the constituent σ' of ρ'. It is defined as:*

$$\mathcal{I}(\sigma') = \{\sigma \in C(\rho) \mid \sigma' = \sigma[T_i \mapsto \tau_i]\} \cup \begin{cases} \{T_i\} & \text{if } \sigma' = \tau_i \\ \{T'_{i,\sigma'}\} & \text{if } \sigma' \preceq \tau_i \text{ and } \tau_i \neq \sigma' \\ \emptyset & \text{otherwise} \end{cases}$$

□

Observation 3.1 *$\mathcal{I}(\sigma_i') \cap \mathcal{I}(\sigma_j') = \emptyset$ for $i \neq j$.* □

Theorem 3.1 (reuse of polymorphic results) *Let p/n be a predicate typed under the polymorphic type definition ρ. Let $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ denote the* extended *(polymorphic) constituents of ρ. Let:*

$$\begin{array}{ll} p(x_1^{\sigma_1}, \ldots, x_n^{\sigma_1}, \ldots, x_1^{\sigma_k}, \ldots, x_n^{\sigma_k}) & \leftarrow \pi \quad \in [\![P^\rho]\!]_{CLP(N)} \\ p(x_1^{\sigma_1}, \ldots, x_n^{\sigma_1}, \ldots, x_1^{\sigma_k}, \ldots, x_n^{\sigma_k}) & \leftarrow \varphi \quad \in [\![P^\rho]\!]_{Pos} \end{array}$$

denote respectively size relation and rigidity information (success set), derived in the combined analysis using the type-based *norms corresponding to the extended constituents of ρ (π is a linear constraint and φ is a Boolean formula). Also let:*

$$\begin{array}{ll} p(x_1^{\sigma_1'}, \ldots, x_1^{\sigma_l'}, \ \ldots, \ x_n^{\sigma_1'}, \ldots, x_n^{\sigma_l'}) & \leftarrow \pi' \in [\![P^{\rho'}]\!]_{CLP(N)} \\ p(x_1^{\sigma_1'}, \ldots, x_1^{\sigma_l'}, \ \ldots, \ x_n^{\sigma_1'}, \ldots, x_n^{\sigma_l'}) & \leftarrow \varphi' \in [\![P^{\rho'}]\!]_{Pos} \end{array}$$

denote respectively size relation and rigidity information (success set), derived in the combined analysis using the type-based *norms corresponding to the constituents $\sigma'_1, \ldots, \sigma'_l$ of ρ' (which is an instance of ρ). Under the assumption that no widening was used in the analysis, the following:*

$$p(x_1^{\sigma'_1}, \ldots, x_n^{\sigma'_1}, \ \ldots, \ x_1^{\sigma'_l}, \ldots, x_n^{\sigma'_l}) \ \leftarrow \ \exists X : \pi \wedge \bigwedge_{j=1}^{l} \bigwedge_{i=1}^{n} (x_i^{\sigma'_j} = \sum_{\sigma \in \mathcal{I}(\sigma'_j)} x_i^{\sigma})$$

$$p(x_1^{\sigma'_1}, \ldots, x_n^{\sigma'_1}, \ \ldots, \ x_1^{\sigma'_l}, \ldots, x_n^{\sigma'_l}) \ \leftarrow \ \exists X : \varphi \wedge \bigwedge_{j=1}^{l} \bigwedge_{i=1}^{n} (x_i^{\sigma'_j} \leftrightarrow \bigwedge_{\sigma \in \mathcal{I}(\sigma'_j)} x_i^{\sigma})$$

where $X = \{x_1^{\sigma_1}, \ldots, x_n^{\sigma_1} \ldots, x_1^{\sigma_k}, \ldots x_n^{\sigma_k}\}$ provide correct size relation and rigidity information for the monomorphic version of the predicate p/n that is at least as precise as φ' and π'. □

4 Proof Sketch for Theorem 3.1

In what follows we will proof Theorem 3.1 for the case of the Pos analysis. Assume the program is polymorphically typed under ρ and that the (monomorphic) typing ρ' is an instance of ρ. Assume that $C = \{\sigma_1, \ldots, \sigma_k\}$ are the (*extended*) constituents of ρ and that $C' = \{\sigma'_1, \ldots, \sigma'_l\}$ are the constituents of ρ'. The proof is done in several steps: (1) In Section 4.1 we provide some notation and lemmas; (2) In Section 4.2 we prove that the reuse of polymorphic results is more precise; and (3) In Section 4.3 we prove that it's also correct (sound).

4.1 Some notations and lemmas

For the sake of simplicity we want to use a new notation for Theorem 3.1: Given a polymorphic formula $p(\bar{X}^C) \leftarrow \varphi$, we define the monomorphic version, i.e. the reuse, of it to be $p(\bar{X}^{C'}) \leftarrow \exists X^C.\ [\mathcal{U}(X^{C'}) \wedge \varphi]$ where:

$$\mathcal{U}(X) = \bigwedge_{x^{\sigma'} \in X} (x^{\sigma'} \leftrightarrow \bigwedge_{\sigma \in \mathcal{I}(\sigma')} x^{\sigma})$$

Lemma 4.1 *For a given set of variables X, X_1, X_2, if $X = X_1 \cup X_2$ and $X_1 \cap X_2 = \emptyset$ then: (1) $vars(\mathcal{U}(X_1)) \cap vars(\mathcal{U}(X_2)) = \emptyset$; and (2) $\mathcal{U}(X) = \mathcal{U}(X_1) \wedge \mathcal{U}(X_2)$.*
Proof. *By observation 3.1 and the definition of $\mathcal{U}$.* □

Lemma 4.2 $\exists X.\ \mathcal{U}(X) \approx true$ □

4.2 The reuse is more precise

The proof is by induction and is done according to the following steps:

(1) we show that the reuse of the base cases gives the same result;

(2) we show that the lub does not cause precision loss; and

(3) assuming that the reuse is at least as precise up to iteration $i-1$, we show that it is also at least as precise in the i-th iteration.

4.2.1 The base case of the induction

We will show that for the base cases the reuse of the polymorphic results is exactly equivalent the monomorphic results.

Base case I:

We will show that:

$$X_g^{\rho'} \wedge \bigwedge_{\sigma' \in \rho'} \alpha(x^{\sigma'} = y^{\sigma'}) = \exists X^{\rho}.\ [\mathcal{U}(X^{\rho'}) \wedge X_g^{\rho'} \wedge \bigwedge_{\sigma \in \rho} \alpha(x^{\sigma} = y^{\sigma})]$$

The idea is that any $\alpha(x^{\sigma'} = y^{\sigma'})$ can be represented as follows:

$$\exists x^{\sigma}, y^{\sigma}.\ [\mathcal{U}(\{x^{\sigma'}, y^{\sigma'}\}) \wedge \bigwedge_{\sigma \in \mathcal{I}(\sigma')} \alpha(x^{\sigma} = y^{\sigma})]$$

Now since:

(1) $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) = \emptyset$ (thanks to the different T' constituents); and

(2) The Boolean formulas obtained by abstracting two *different equations* $\alpha(x^{\sigma_i} = x^{\sigma_i})$ and $\alpha(x^{\sigma_j} = x^{\sigma_j})$ do not share any variable

the instances of the above equation for two different constituents σ'_i and σ'_j do not share variables – exactly like $\alpha(x^{\sigma'_i} = y^{\sigma'_i})$ and $\alpha(x^{\sigma'_j} = y^{\sigma'_j})$. Hence no relation are introduced when putting all of them in one conjunction.

For X_g^{ρ} and $X_g^{\rho'}$. If $x^{\sigma'} \in X_g^{\rho'}$ then for any $\sigma \in \mathcal{I}(\sigma')$ it must be the case that $x^{\sigma} \in X^{\rho}$, hence $x^{\sigma'}$ can be obtained using $\mathcal{U}(\{x^{\sigma'}\}) \wedge X_g^{\rho}$.

Base case II: Similar to the above base case.

4.2.2 The LUB does not cause precision loss

Let φ_1 and φ_2 be Boolean formulas obtained for the same predicate $p(\bar{X}_0^C)$ in the polymorphic analysis, and let φ'_1 and φ'_2 be Boolean formulas obtained for the same predicate but in the mono analysis, i.e for $p(\bar{X}_0^{C'})$, such that $\exists X_0^C.\, \mathcal{U}(X_0^{C'}) \wedge \varphi_i$ is more precise than φ'_i, we will show that $\exists X_0^C.\, [\mathcal{U}(X_0^{C'}) \wedge (\varphi_1 \vee \varphi_2)]$ is more precise than $\varphi'_1 \vee \varphi'_2$:

$$\begin{array}{ll} & \exists X_0^C.\, [\mathcal{U}(X_0^{C'}) \wedge (\varphi_1 \vee \varphi_2)] \\ = & [\exists X_0^C.\, \mathcal{U}(X_0^{C'}) \wedge \varphi_1] \vee [\exists X_0^C.\, \mathcal{U}(X_0^{C'}) \wedge \varphi_2] \\ \rightarrow & \varphi'_1 \vee \varphi'_2 \end{array}$$

4.2.3 The induction step

Suppose the theorem holds for all iterations smaller than i, now we will prove that it holds also for the i-th iteration. Given a clause: $C \equiv p(\bar{x}_0) \leftarrow p_1(\bar{x}_1), \ldots, p_n(\bar{x}_n)$ the abstract clause wrt ρ is:

$$C^C \equiv p(\bar{X}_0^C) \leftarrow p_1(\bar{X}_1^C), \ldots, p_n(\bar{X}_n^C)$$

and wrt ρ' is:

$$C^{C'} \equiv p(\bar{X}_0^{C'}) \leftarrow p_1(\bar{X}_1^{C'}), \ldots, p_n(\bar{X}_n^{C'})$$

Let us denote the set $X_0^C \cup \cdots \cup X_n^C$ by X^C and the set $X_0^{C'} \cup \cdots \cup X_n^{C'}$ by $X^{C'}$. Let $p(\bar{X}_0^C) \leftarrow \varphi$ be a formula obtained in the i-th iteration of the polymorphic analysis as follows:

$$\varphi = \exists X^C \setminus X_0^C . \varphi_1, \ldots, \varphi_i$$

where φ_j is a formula over X_j^C and it represents the approximation of the predicate p_j obtained in the previous iterations. Let $p(\bar{X}_0^{C'}) \leftarrow \varphi'$ be a formula obtained in the i-th iteration of the monomorphic analysis as follows:

$$\varphi' = \exists X^{C'} \setminus X_0^{C'} . \varphi'_1, \ldots, \varphi'_i$$

where φ'_j is a formula over $X_j^{C'}$ and it represents the approximation of the predicate p_j obtained in the previous iterations. Now we will show that the reuse of φ according to Theorem 3.1 is more precise than φ' (explanations are provided below):

$$\begin{array}{ll}
 & \varphi' \\
=^0 & \exists X^{C'} \setminus X_0^{C'} .\ \varphi'_1, \ldots, \varphi'_i \\
\leftarrow^1 & \exists X^{C'} \setminus X_0^{C'} .\ [(\exists X_1^C .\ [\mathcal{U}(X_1^{C'}) \wedge \varphi_1]) \wedge \cdots \wedge (\exists X_n^C .\ [\mathcal{U}(X_n^{C'}) \wedge \varphi_n])] \\
=^2 & \exists X^{C'} \setminus X_0^{C'} .\ [(\exists X^C .\ [\mathcal{U}(X_1^{C'}) \wedge \varphi_1]) \wedge \cdots \wedge (\exists X^C .\ [\mathcal{U}(X_n^{C'}) \wedge \varphi_n])] \\
=^3 & \exists X^{C'} \setminus X_0^{C'} .\ [(\exists X^C .\ [\mathcal{U}(X^{C'}) \wedge \varphi_1]) \wedge \cdots \wedge (\exists X^C .\ [\mathcal{U}(X^{C'}) \wedge \varphi_n])] \\
\leftarrow^4 & \exists X^{C'} \setminus X_0^{C'} .\ [\exists X^C .\ [\mathcal{U}(X^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n]] \\
=^5 & \exists X^C .\ [\exists X^{C'} \setminus X_0^{C'} .\ [\mathcal{U}(X^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n]] \\
=^6 & \exists X^C .\ [\exists X^{C'} \setminus X_0^{C'} .\ [\mathcal{U}(X^{C'} \setminus X_0^{C'}) \wedge \mathcal{U}(X_0^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n]] \\
=^7 & \exists X^C .\ [\mathcal{U}(X_0^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n \wedge \exists X^{C'} \setminus X_0^{C'} .\ [\mathcal{U}(X^{C'} \setminus X_0^{C'})]] \\
=^8 & \exists X^C .\ [\mathcal{U}(X_0^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n] \\
=^9 & \exists X_0^C .\ [\exists X^C \setminus X_0^C .\ [\mathcal{U}(X_0^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n]] \\
=^{10} & \exists X_0^C .\ [\mathcal{U}(X_0^{C'}) \wedge \exists X_C \setminus X_0^C .\ [\varphi_1 \wedge \cdots \wedge \varphi_n]] \\
=^{11} & \exists X_0^C .\ [\mathcal{U}(X_0^{C'}) \wedge \varphi]
\end{array}$$

We proved that: $\left[\exists X_0^C .\ [\mathcal{U}(X_0^{C'}) \wedge \varphi]\right] \rightarrow \varphi'$. Here are the explanations to each of the above steps:

(0) definition of φ'.

(1) By the induction hypotheses $\varphi'_i \leftarrow (\exists X_i^C.\ [\mathcal{U}(X_i^{C'}) \wedge \varphi_i])$

(2) $\exists x.\psi = \exists y.\exists x.\psi$ when $y \notin vars(\psi)$. In our case the variables of φ_i are only from X_i^C, hence $\exists X_i^C$ can be replaced by $\exists X^C$ since $X_i^C \subseteq X^C$.

(3) since the formula $\mathcal{U}(X^{C'} \setminus X_i^{C'})$ does not share any variable with $\mathcal{U}(X_i^{C'})$ (by Lemma 4.1) or with φ_i (by definition). Actually this step is redundant but it simplifies the presentation.

(4) $(\exists x.\varphi_1) \wedge (\exists x\varphi_2) \geq \exists x[\varphi_1 \wedge \varphi_2]$.

(5) just switching the order of the variables in the existential quantification.

(6) recall that $\mathcal{U}(X^{C'}) = \mathcal{U}(X^{C'} \setminus X_0^{C'}) \wedge \mathcal{U}(X_0^{C'})$ (by Lemma 4.1)

(7) $\exists x.\psi_1 \wedge \psi_2 = \psi_1 \wedge \exists x.\psi_2$ when $x \notin vars(\psi_1)$. In our case $vars(\mathcal{U}(X_0^{C'}) \wedge \varphi_1 \wedge \cdots \wedge \varphi_n) \cap (X^{C'} \setminus X_0^{C'}) = \emptyset$.

(8) recall that $\exists X^{C'} \setminus X_0^{C'}.\ [\mathcal{U}(X^{C'} \setminus X_0^{C'})] \approx true$ (by Lemma 4.2)

(9) splitting X^C to X_0^C and $X^C \setminus X_0^C$

(10) $\exists x.\psi_1 \wedge \psi_2 = \psi_1 \wedge \exists x.\psi_2$ when $x \notin vars(\psi_1)$. In our case $vars(\mathcal{U}(X_0^{C'})) \cap (X^C \setminus X_0^C) = \emptyset$ since $\mathcal{U}(X_0^{C'})$ uses only variables form $X_0^{C'}$ and X_0^C.

(11) definition of φ above.

Actually what we have proved above is that the monomorphic result cannot be more precise than the result obtained by reuse of the polymorphic results. The question arises whether it can be less precise. The following example demonstrate that it can.

Example 4.1 *This example demonstrates how the reuse of the polymorphic results can be more precise than the monomorphic results. Also, it shows how a termination proof can be obtained when reusing the polymorphic results and cannot be obtained using directly the results of the monomorphic analysis. The example is similar to the one in the SAS2002 paper [1] that demonstrates the reuse of size relations. Note that the example is not based on the fact that we have several T' constituents, we even don't use them, but on the fact that $\mathcal{I}(\sigma)$ is sometimes not a singleton. Consider the following poly-typed logic program:*

```
:- type pair(T) --> t(list(int),list(T)).
:- type list(T) --> []; [T|list(T)].
:- type list(int) --> []; [int|list(int)].

:- pred p(pair(T)).
p(X) :- q(X),h(X).

:- pred q(pair(T)).
q(t([_],[_|_])).

:- pred h(pair(T)).
h(t([_|_],[_])).
```

and suppose it is used in a context where T=int. The polymorphic results wrt the constituent $T, list(T), int$ and $list(int)$ are:

$$p(X_T, X_{l(T)}, X_i, X_{l(i)}) \leftarrow X_{l(T)} \wedge X_{l(i)}$$
$$q(X_T, X_{l(T)}, X_i, X_{l(i)}) \leftarrow X_{l(i)}$$
$$h(X_T, X_{l(T)}, X_i, X_{l(i)}) \leftarrow X_{l(T)}$$

and the results in the monomorphic context for the constituents $int, list(int)$ are:

$$p(X'_i, X'_{l(i)}) \leftarrow true$$
$$q(X'_i, X'_{l(i)}) \leftarrow true$$
$$h(X'_i, X'_{l(i)}) \leftarrow true$$

but the reuse of the polymorphic results gives more precise information for $p/1$:

$$p(X'_i, X'_{l(i)}) \leftarrow \exists X_{l(T)}, X_{l(i)}. \\ [(X'_{l(i)} \leftrightarrow X_{l(i)} \wedge X_{l(T)})) \wedge (X_{l(i)} \wedge X_{l(T)})] = X'_{l(i)}$$
$$q(X'_i, X'_{l(i)}) \leftarrow true$$
$$h(X'_i, X'_{l(i)}) \leftarrow true$$

Now suppose this predicate is used as follow:

```
h :- p(X), X=t(A,B), app(A,[],C).
```

Using the monomorphic results one cannot infer that A is $list(int)$-rigid when $app/3$ is called, hence one cannot prove that $app/3$ terminates. But reusing the polymorphic results one can infer that A is $list(int)$-rigid when $app/3$ is called, hence one can prove that $app/3$ terminates.

□

4.3 The reuse of the polymorphic result is sound

Note that $\mathcal{U}(X^{C'})$ is an abstraction function from Pos_{X^C} (the polymorphic domain) to $Pos_{X^{C'}}$ (the monomorphic domain). We denote this abstraction by α. Now let α^C be the abstract function from the concrete domain to Pos_{X^C} and let $\alpha^{C'}$ be the abstraction function from the concrete domain to $Pos_{X^{C'}}$.

Moreover, we can define the monomorphic abstraction function $\alpha^{C'}$ in terms of α^C and $\mathcal{U}$ as follows: $\alpha^{C'} = \alpha(\alpha^C)$.

Let F be the concrete semantics, F^C the abstract polymorphic semantics, and $F^{C'}$ the abstract mono semantics. Note that the reuse of F^C is $\alpha(F^C)$. The following shows that the reuse is sound:

$$\begin{array}{ll} & \alpha^C(F) \leq F^C \\ \Rightarrow & \alpha(\alpha^C(F)) \leq \alpha(F^C) \\ \Rightarrow & \alpha^{C'}(F) \leq \alpha(F^C) \end{array}$$

which mean that the reuse $\alpha(F^C)$ is an approximation of $\alpha^{C'}(F)$. Note that $\alpha^{C'}$ is the abstraction of the concrete semantics into the monomorphic domain and not the one obtained in the monomorphic analysis, namely $F^{C'}$. Actually from the reuse and the above proof we can conclude that: $\alpha^{C'}(F) \leq \alpha(F^C) \leq F^{C'}$.

5 Conclusion

A problem in the SAS2002 paper [1] was pointed out to us by Peter Stuckey. We noticed that the problem occurs when $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) \neq \emptyset$. We solved the problem by changing the way of the reuse, we introduced extra constituents in the polymorphic type system such that $\mathcal{I}(\sigma'_i) \cap \mathcal{I}(\sigma'_j) = \emptyset$ holds. The method is not as modular as the one presented in [1] because it assumes an upper bound on the number of constituents of the instance of the polymorphic parameter. When confronted with a call to an instance with more strict constituents of τ_i than provided by the polymorphic analysis, one can still construct a correct formula through reuse (though precision loss can occur) by splitting the strict constituents over several sets, applying the reuse formula on each set and performing a conjunction.

The proof of Theorem 3.1 for the case of size-relation analysis is different from the rigidity analysis, this is because in practice we use the Polyhedra domain (part of CLP(N)) with convex-hull operation as LUB and also with a widening operator in order to avoid infinite chains:

- If we assume that the size-relation analysis is applied over the CLP(N) domain (where the LUB is sets union) and without applying any widening then the proof is similar to that of Pos – since Lemma 4.1, Lemma 4.2 and the different properties of Boolean functions used in the proof hold also for the case CLP(N) constraints. In this case we also should use transfinite induction. Again, recall that this is a theoretical case since the analysis may be infinite.

- If we assume that the analysis is done over the Polyhedra domain with convex-hull as LUB and also with widening operator, then in addition we must show that the application of widening and convex-hull operations applied on a more precise formula yields a more precise formula (confer to what we proved about the LUB in Section 4.2.2).

References

[1] Maurice Bruynooghe, Michael Codish, Samir Genaim, and Wim Vanhoof. Reuse of results in termination analysis of typed logic programs. In *Static Analysis, 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, 2002. URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=38931.