Quasi-interpretation Synthesis by Decomposition

Guillaume Bonfante, Jean-Yves Marion and Romain Péchoux
{ Guillaume.bontante, Jean-Yves.Marion, Romain.Pechoux }@loria.fr

Carte project-Loria,Nancy,France

ICTAC 2007

Motivations

- Resource control of functional programs
  - Using Quasi-interpretations (QI) [Bonfante, Marion, Moyen 00]
    - ⋆ Synthesis problem is decidable [Amadio04,BMMP05](Exponential in the size of the program)
- Modularity as a way to decompose programs
  - $P(\mathbf{p}_1)$ and $P(\mathbf{p}_2)$ iff $P(\mathbf{p}_1 \cup \mathbf{p}_2)$
  - Decrease the time complexity of QI search (Divide-and-conquer strategy)
  - Increase the number of algorithms captured
  - Adapt QI to higher-order programs

Modularity

- Termination is not a modular property of TRS [Toyama 87]:

$$\mathtt{f}(a, b, x) \to \mathtt{f}(x, x, x) \qquad \mathtt{g}(a, b) \to a \qquad \mathtt{g}(a, b) \to b$$

$$\mathtt{f}(a, b, \mathtt{g}(a, b)) \to \mathtt{f}(\mathtt{g}(a, b), \mathtt{g}(a, b), \mathtt{g}(a, b)) \to \ldots \to \mathtt{f}(a, b, \mathtt{g}(a, b))$$

- In the litterature [Gramlich, Klop, Ohlebusch], 3 decompositions are considered:
  - Disjoint union
  - Constructor-sharing union
  - Hierarchical union

First Order Language: syntax

Definition

Constructors: Cns , functions: Fct, variables: $\mathcal{X}$.

$$\begin{array}{lllll} \texttt{(Patterns)} & \mathcal{P} \ni p & ::= & \mathbf{c}(p_1, \cdots, p_n) \mid x \\ \texttt{(Rules)} & \mathcal{R} \ni r & ::= & \mathtt{f}(p_1, \cdots, p_n) \rightarrow t \\ \texttt{(Programs)} & P \ni p & ::= & \langle \mathcal{X}, Cns, Fct, \mathcal{R} \rangle \\ \texttt{(Values)} & \mathcal{V} \ni v & ::= & \mathbf{c}(v_1, \cdots, v_n) \mid \mathbf{c} \end{array}$$

First Order Language: semantics

Definition

Define the semantics $[\![-]\!]$ by: $[\![\mathtt{f}(v_1, \cdots, v_n)]\!] = w$ iff $\mathtt{f}(v_1, \cdots, v_n) \xrightarrow{*} w$ with $w \in \mathcal{V}$.

Example (Logarithm)

$$\begin{array}{rl} \mathtt{half}(\mathbf{0}) & \to \mathbf{0} \\ \mathtt{half}(\mathbf{S}(\mathbf{0})) & \to \mathbf{0} \\ \mathtt{half}(\mathbf{S}(\mathbf{S}(y))) & \to \mathbf{S}(\mathtt{half}(y)) \end{array} \quad \left| \quad \begin{array}{rl} \mathtt{log}(\mathbf{0}) & \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{0})) & \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{S}(y))) & \to \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y)))) \end{array} \right.$$

We have:

- $[\![\mathtt{log}(\underline{n})]\!] = \underline{\lfloor \log(n) \rfloor}$
- $[\![\mathtt{half}(\underline{n})]\!] = \underline{\lfloor n/2 \rfloor}$ with $\underline{n} = \underbrace{\mathbf{S}(\ldots\mathbf{S}(\mathbf{0})\ldots)}_{n \text{ times } \mathbf{S}}$.

Additive and polynomial quasi-interpretations (QI)

Definition

A quasi-interpretation is a total assignment defined by:

- $\forall \mathbf{c} \in Cns$, $(\!|\mathbf{c}|\!)(X_1, \cdots, X_n) = \sum_{i=1}^{n} X_i + \alpha$ with $\alpha \geq 1$
- $\forall \mathtt{f} \in Fct$ of arity n, $(\!|\mathtt{f}|\!)$ is a max-polynomial from $(\mathbb{R}^+)^n \rightarrow \mathbb{R}^+$.
- $\forall \mathtt{f} \in Fct$, $(\!|\mathtt{f}|\!)(X_1, \cdots, X_n) \geq X_i$ (*Subterm Property*)
- $\forall \mathtt{f} \in Fct$, $X_i \leq Y_i \Rightarrow (\!|\mathtt{f}|\!)(X_1, \cdots, X_n) \leq (\!|\mathtt{f}|\!)(Y_1, \cdots, Y_n)$ (*Monotonicity*)
- $\forall x \in \mathcal{X}$, $(\!|x|\!) = X$
- $\forall\ l \rightarrow r \in \mathcal{R}$, we have:

$$(\!|l|\!) \geq (\!|r|\!)$$

Quasi-interpretation of logarithm

Example

$$\begin{array}{rl} \mathtt{half}(\mathbf{0}) & \to \mathbf{0} \\ \mathtt{half}(\mathbf{S}(\mathbf{0})) & \to \mathbf{0} \\ \mathtt{half}(\mathbf{S}(\mathbf{S}(y))) & \to \mathbf{S}(\mathtt{half}(y)) \end{array} \quad \left| \quad \begin{array}{rl} \mathtt{log}(\mathbf{0}) & \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{0})) & \to \mathbf{0} \\ \mathtt{log}(\mathbf{S}(\mathbf{S}(y))) & \to \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y)))) \end{array} \right.$$

$$(\!|\mathbf{0}|\!) = 0,\ (\!|\mathbf{S}|\!)(X) = X + 1,$$
$$(\!|\mathtt{log}|\!)(X) = X,\ (\!|\mathtt{half}|\!)(X) = X$$

$$\begin{aligned} (\!|\mathtt{half}(\mathbf{S}(\mathbf{S}(y)))|\!) &= (\!|\mathtt{half}|\!)((\!|\mathbf{S}|\!)((\!|\mathbf{S}|\!)(Y))) \\ &= Y + 2 \\ &\geq Y + 1 = (\!|\mathbf{S}(\mathtt{half}(y))|\!) \end{aligned}$$

Characterizations

Theorem (BMM07)

The set of functions computed by programs having an additive and polynomial QI and:

- *ordered by $\prec_{rpo}$ where each function symbol has a status p is exactly the set of* FPTIME *functions.*
- *ordered by $\prec_{rpo}$ is exactly the set of* $\mathrm{FPSPACE}$ *functions.*

Lemma (Fundamental Lemma)

Given a program admitting an additive and polynomial quasi-interpretation $(\!|-|\!)$, there is a polynomial P such that if $[\![f(v_1, \cdots, v_n)]\!]$ is defined then:

$$|[\![f(v_1, \cdots, v_n)]\!]| \leq P(\max_{i=1..n} |v_i|)$$

A simple case : Disjoint union

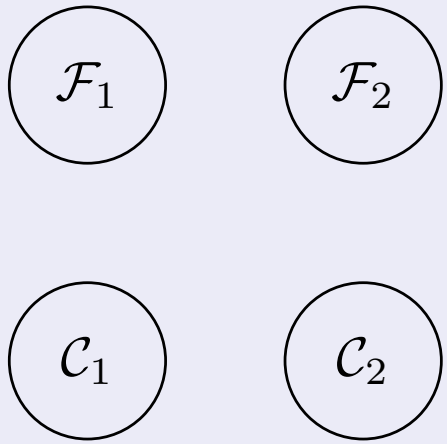
Definition

- Two programs $\langle \mathcal{X}_1, Cns_1, Fct_1, \mathcal{R}_1 \rangle$ and $\langle \mathcal{X}_2, Cns_2, Fct_2, \mathcal{R}_2 \rangle$ are disjoint if:

$$Fct_1 \cap Fct_2 = Cns_1 \cap Cns_2 = Fct_1 \cap Cns_2 = Fct_2 \cap Cns_1 = \emptyset$$

- $\langle \mathcal{X}_1, Cns_1, Fct_1, \mathcal{R}_1 \rangle \biguplus \langle \mathcal{X}_2, Cns_2, Fct_2, \mathcal{R}_2 \rangle$ is defined by:

$$\langle \mathcal{X}_1 \cup \mathcal{X}_2, Cns_1 \cup Cns_2, Fct_1 \cup Fct_2, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$$



Disjoint union

Modularity of Disjoint union

Example

$$\begin{aligned} \texttt{double}(\mathbf{0}) &\to \mathbf{0} \\ \texttt{double}(\mathbf{S}(x)) &\to \mathbf{S}(\mathbf{S}(\texttt{double}(x))) \end{aligned}$$

$$\begin{aligned} \texttt{triple}(\mathbf{0'}) &\to \mathbf{0'} \\ \texttt{triple}(\mathbf{S'}(x)) &\to \mathbf{S'}(\mathbf{S'}(\mathbf{S'}(\texttt{triple}(x)))) \end{aligned}$$

$$(\!|\texttt{triple}|\!)(X) = 3 \times X, \quad (\!|\texttt{double}|\!)(X) = 2 \times X,$$
$$(\!|\mathbf{0}|\!) = (\!|\mathbf{0'}|\!) = 0, \quad (\!|\mathbf{S}|\!)(X) = (\!|\mathbf{S'}|\!)(X) = X + 1$$

- If we cut a program into k programs having n_j variables complexity decreases from $\alpha^{\sum_{i=1}^{k} n_i}$ to $\sum_{i=1}^{k} \alpha^{n_i}$.

Modularity of Disjoint union

Proposition

RPO is modular as a consequence of [Kurihara-Ohuchi92].

Proposition

Quasi-interpretations are modular.

Lemma (Fundamental Lemma)

The fundamental lemma **holds**.

Theorem

The set of functions computed by **Disjoint Union** *of programs having an additive and polynomial QI and*

- *ordered by $\prec_{rpo}$ where each function symbol has a status p is exactly the set of* FPTIME *functions.*
- *ordered by $\prec_{rpo}$ is exactly the set of* $\mathrm{FPSPACE}$ *functions.*

Constructor-sharing union

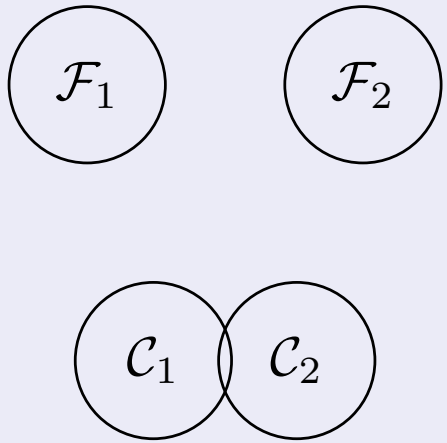
Definition

- Two programs $\langle \mathcal{X}_1, \mathit{Cns}_1, \mathit{Fct}_1, \mathcal{R}_1 \rangle$ and $\langle \mathcal{X}_2, \mathit{Cns}_2, \mathit{Fct}_2, \mathcal{R}_2 \rangle$ are constructor-sharing if:

$$\mathit{Fct}_1 \cap \mathit{Fct}_2 = \mathit{Fct}_1 \cap \mathit{Cns}_2 = \mathit{Fct}_2 \cap \mathit{Cns}_1 = \emptyset \text{ and } \mathit{Cns}_1 \cap \mathit{Cns}_2 \neq \emptyset$$

- $\langle \mathcal{X}_1, \mathit{Cns}_1, \mathit{Fct}_1, \mathcal{R}_1 \rangle \bigsqcup \langle \mathcal{X}_2, \mathit{Cns}_2, \mathit{Fct}_2, \mathcal{R}_2 \rangle$ is defined by:

$$\langle \mathcal{X}_1 \cup \mathcal{X}_2, \mathit{Cns}_1 \cup \mathit{Cns}_2, \mathit{Fct}_1 \cup \mathit{Fct}_2, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$$



Constructor sharing union

Modularity of constructor-sharing union

Example

$$\begin{aligned}
\mathtt{minus}(x, \mathbf{0}) &\to x \\
\mathtt{minus}(\mathbf{0}, \mathbf{S}(y)) &\to \mathbf{0} \\
\mathtt{minus}(\mathbf{S}(x), \mathbf{S}(y)) &\to \mathtt{minus}(x, y)
\end{aligned}$$

$$\begin{aligned}
\mathtt{add}(x, \mathbf{0}) &\to x \\
\mathtt{add}(\mathbf{0}, y) &\to y \\
\mathtt{add}(\mathbf{S}(x), \mathbf{S}(y)) &\to \mathbf{S}(\mathbf{S}(\mathtt{add}(x, y)))
\end{aligned}$$

$$\begin{aligned}
(\!|\mathbf{0}|\!)_1 = 0, \quad & (\!|\mathbf{0}|\!)_2 = 0, \\
(\!|\mathbf{S}|\!)_1(X) = X + 1, \quad & (\!|\mathbf{S}|\!)_2(X) = X + 1, \\
(\!|\mathtt{minus}|\!)_1(X, Y) = X + Y \quad & (\!|\mathtt{add}|\!)_2(X, Y) = X + Y
\end{aligned}$$

Modularity of constructor-sharing union

Proposition

RPO is modular.

Proposition

Quasi-interpretations are **not** *modular.*

Lemma (Fundamental Lemma)

The fundamental lemma **holds**.

Theorem

The set of functions computed by **constructor-sharing union** *of programs having an additive and polynomial QI and*

- *ordered by $\prec_{rpo}$ where each function symbol has a status p is exactly the set of* FPTIME *functions.*
- *ordered by $\prec_{rpo}$ is exactly the set of* $\mathrm{FPSPACE}$ *functions.*

Modularity of constructor-sharing union

Example (Non modular Quasi-interpretation)

$$\begin{array}{l|l} \mathtt{f}_0(\mathbf{a}(x)) \to \mathtt{f}_0(\mathtt{f}_0(x)) & \mathtt{f}_1(\mathbf{b}(x)) \to \mathtt{f}_1(\mathtt{f}_1(x)) \\ \mathtt{f}_0(\mathbf{b}(x)) \to \mathbf{a}(\mathbf{a}(\mathtt{f}_0(x))) & \mathtt{f}_1(\mathbf{a}(x)) \to \mathbf{b}(\mathbf{b}(\mathtt{f}_1(x))) \end{array}$$

$$\begin{array}{rl|rl} (\!|\mathbf{a}|\!)_0(X) & = X+1 & (\!|\mathbf{a}|\!)_1(X) & = X+2 \\ (\!|\mathbf{b}|\!)_0(X) & = X+2 & (\!|\mathbf{b}|\!)_1(X) & = X+1 \\ (\!|\mathtt{f}_0|\!)_0(X) & = X & (\!|\mathtt{f}_1|\!)_1(X) & = X \end{array}$$

- If we cut a program into k programs having n_j variables complexity decreases from $\alpha^{\sum_{i=1}^{k} n_i}$ to $\sum_{i=1}^{k} \alpha^{n_i}$.

Hierarchical union

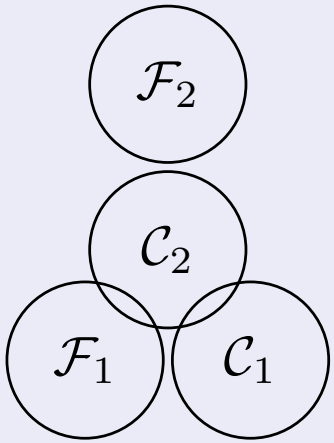
Definition

- Two programs $\langle \mathcal{X}_1, Cns_1, Fct_1, \mathcal{R}_1 \rangle$ and $\langle \mathcal{X}_2, Cns_2, Fct_2, \mathcal{R}_2 \rangle$ are hierarchical if:

  $Fct_1 \cap Fct_2 = Fct_2 \cap Cns_1 = \emptyset$ and $Cns_1 \cap Cns_2 \neq \emptyset$ and $Fct_1 \cap Cns_2 \neq \emptyset$

- $\langle \mathcal{X}_1, Cns_1, Fct_1, \mathcal{R}_1 \rangle \ll \langle \mathcal{X}_2, Cns_2, Fct_2, \mathcal{R}_2 \rangle$ is defined by:

$$\langle \mathcal{X}_1 \cup \mathcal{X}_2, Cns_1 \cup Cns_2 - Fct_1, Fct_1 \cup Fct_2, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$$



Hierarchical union

Modularity of hierarchical union

Example

$$\begin{aligned}
\mathtt{add}(x, \mathbf{0}) &\to x \\
\mathtt{add}(\mathbf{0}, y) &\to y \\
\mathtt{add}(\mathbf{S}(x), \mathbf{S}(y)) &\to \mathbf{S}(\mathbf{S}(\mathtt{add}(x, y)))
\end{aligned}$$

$$\begin{aligned}
\mathtt{mult}(\mathbf{0}, x) &\to \mathbf{0} \\
\mathtt{mult}(\mathbf{S}(x), y) &\to \mathtt{add}(y, \mathtt{mult}(x, y))
\end{aligned}$$

$$\begin{aligned}
(\!|\mathbf{0}|\!)_1 &= 0, & (\!|\mathbf{0}|\!)_2 &= 0, \\
(\!|\mathbf{S}|\!)_1(X) &= X + 1, & (\!|\mathbf{S}|\!)_2(X) &= X + 1, \\
(\!|\mathtt{add}|\!)_1(X, Y) &= X + Y & (\!|\mathtt{add}|\!)_2(X, Y) &= X + Y + 1 \\
& & (\!|\mathtt{mult}|\!)_2(X, Y) &= X \times (Y + 1)
\end{aligned}$$

Modularity for hierarchical union

Proposition

RPO is modular.

Proposition

Quasi-interpretations are **not** *modular.*

Lemma

The fundamental lemma **does not** *hold.*

Example

$$
\begin{array}{rl}
\mathtt{double}(\mathbf{S}(x)) & \to \mathbf{S}(\mathbf{S}(\mathtt{double}(x))) \\
\mathtt{double}(\mathbf{0}) & \to \mathbf{0} \\
(\!|\mathbf{0}|\!)_1 & = 0 \\
(\!|\mathbf{S}|\!)_1(X) & = X + 1 \\
(\!|\mathtt{double}|\!)_1(X) & = 2 \times X
\end{array}
\qquad
\begin{array}{rl}
\exp(\mathbf{S}(x)) & \to \mathtt{double}(\exp(x)) \\
\exp(\mathbf{0}) & \to \mathbf{S}(\mathbf{0}) \\
(\!|\mathbf{0}|\!)_2 = 0, & (\!|\mathbf{S}|\!)_2(X) = X + 1 \\
(\!|\exp|\!)_2(X) & = X + 1 \\
(\!|\mathtt{double}|\!)_2(X) & = X + 1
\end{array}
$$

Restriction on QI ?

Does the problem of previous counter-example come from the distinct kind of polynomials ? ($(\!|\texttt{double}|\!)_1(X) = 2 \times X$ and $(\!|\texttt{double}|\!)_2(X) = X + 1$)
Answer: consider the following example:

Example (Exponential)

$$\begin{array}{rl} \mathrm{g}(t) & \to \mathbf{S}(\mathbf{S}(t)) \end{array}$$

$$\begin{array}{rl} \mathtt{f}(\mathbf{S}(x), \mathbf{0}, t) & \to \mathtt{f}(x, t, t) \\ \mathtt{f}(x, \mathbf{S}(z), t) & \to \mathtt{f}(x, z, \mathrm{g}(t)) \\ \mathtt{f}(\mathbf{0}, \mathbf{0}, t) & \to t \end{array}$$

admit the following quasi-interpretations :

$$\begin{array}{rl} (\!|\mathbf{S}|\!)_1(X) & = X + 1 \\ (\!|\mathrm{g}|\!)_1(X) & = X + 2 \end{array}$$

$$\begin{array}{rl} (\!|\mathbf{0}|\!)_2 & = 0 \\ (\!|\mathbf{S}|\!)_2(X) = (\!|\mathrm{g}|\!)_2(X) & = X + 1 \\ (\!|\mathtt{f}|\!)_2(X, Y, Z) & = \max(X, Y, Z) \end{array}$$

However, for $\mathbf{p}_1 \ll \mathbf{p}_2$, we have $[\![\mathtt{f}]\!](\underline{n}, \underline{m}, \underline{p}) = \underline{3^n \times (2 \times m + p)}$

Restrictions on hierarchical unions

Definition (Restrictions on Quasi-interpretation)

We say that $(\!|-|\!)_1$ and $(\!|-|\!)_2$ are **Kind preserving** if $\forall b \in Cns_2 \cap Fct_1$:

1. for every monomial m, $m \sqsubseteq (\!|b|\!)_1 \Leftrightarrow m \sqsubseteq (\!|b|\!)_2$
2. for every monomial m, $\mathtt{coef}_{(\!|b|\!)_2}(m) = 1 \Leftrightarrow \mathtt{coef}_{(\!|b|\!)_1}(m) = 1$

Definition (Syntactical restrictions)

The hierarchical union $\langle \mathcal{X}_1, Cns_1, Fct_1, \mathcal{R}_1 \rangle \ll \langle \mathcal{X}_2, Cns_2, Fct_2, \mathcal{R}_2 \rangle$ is called **stratified** if:

- For each recursive call $\mathtt{f}(p) \rightarrow \mathtt{f}(e)$ then $\forall b$ symbol of e, $b \notin Fct_1$
  - **NOT** $\mathcal{R}_2 = \{\mathtt{f}(\mathbf{S}(x)) \rightarrow \mathtt{f}(\mathtt{g}(x))\} \mid \mathcal{R}_1 = \{\mathtt{g}(x) \rightarrow x\}$
- For each rule $\mathtt{f}(p_1, \cdots, p_n) \rightarrow e$ of $\mathcal{R}_2$, e has no nesting of function symbols.
  - **NOT** $\mathcal{R}_2 = \{\mathtt{f}(\mathbf{S}(x)) \rightarrow \mathtt{f}(\mathtt{g}(x)), \mathtt{g}(x) \rightarrow x\}$

Modularity for Hierarchical union

Lemma (Fundamental Lemma)

Given the stratified union $\mathbf{p}_1 \ll \mathbf{p}_2$ *of two programs having additive Kind preserving QIs* $(\!|-|\!)_1$ *and* $(\!|-|\!)_2$*, the Fundamental Lemma holds.* $\exists P$ *polynomial s.t.* $|[\![f(v_1, \cdots, v_n)]\!]| \leq P(\max_{i=1..n} |v_i|)$

Theorem

The set of functions computed by a hierarchical union of two programs $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *such that*

1. $\mathbf{p}_1 \ll \mathbf{p}_2$ *is a stratified union,*
2. $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *admit the respective additive Kind preserving QIs,*
3. $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *are ordered by* $\prec_{rpo}$ *and each function symbol has a product status,*

is exactly the set of functions computable in FPTIME.
If (3) is replaced by: $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *are ordered by* $\prec_{rpo}$ *then we characterize* FPSPACE.

Example

Example

The hierarchical union of $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$\begin{array}{rl} \mathtt{double}(\mathbf{S}(x)) & \to \mathbf{S}(\mathbf{S}(\mathtt{double}(x))) \\ \mathtt{double}(\mathbf{0}) & \to \mathbf{0} \\ \hline \mathtt{add}(\mathbf{S}(x), y) & \to \mathbf{S}(\mathtt{add}(x, y)) \\ \mathtt{add}(\mathbf{0}, y) & \to y \end{array} \qquad \begin{array}{rl} \mathtt{sq}(\mathbf{S}(x)) & \to \mathbf{S}(\mathtt{add}(\mathtt{sq}(x), \mathtt{double}(x))) \\ \mathtt{sq}(\mathbf{0}) & \to \mathbf{0} \end{array}$$

is stratified ($\mathbf{p}_2$ is flat and the argument of the recursive call $\mathtt{sq}(x)$ is a variable).

$(\!|-|\!)_1$ and $(\!|-|\!)_2$ are additive Kind preserving quasi-interpretations:

$$\begin{array}{rl} (\!|\mathbf{0}|\!)_1 & = 0 \\ (\!|\mathbf{S}|\!)_1(X) & = X + 1 \\ (\!|\mathtt{double}|\!)_1(X) & = 2 \times X \\ (\!|\mathtt{add}|\!)_1(X, Y) & = X + Y \end{array} \qquad \begin{array}{rl} (\!|\mathbf{0}|\!)_2 & = 0 \\ (\!|\mathbf{S}|\!)_2(X) & = X + 1 \\ (\!|\mathtt{double}|\!)_2(X) & = 2 \times X \\ (\!|\mathtt{add}|\!)_2(X, Y) & = X + Y + 1 \\ (\!|\mathtt{sq}|\!)_2(X) & = 2 \times X^2 \end{array}$$

Application to higher-order programs

Example

Consider the following higher-order program $\mathbf{p}$:

$$\begin{aligned} \mathtt{fold}(\lambda x.\mathtt{f}(x), \mathbf{nil}) &\to \mathbf{0} \\ \mathtt{fold}(\lambda x.\mathtt{f}(x), \mathbf{c}(y, l)) &\to \mathtt{f}(\mathtt{fold}(\lambda x.\mathtt{f}(x), l)) \\ \mathtt{h}(l) &\to \mathtt{fold}(\lambda x.\mathtt{g}(x), l) \end{aligned}$$

From $\mathbf{p}$, we obtain $\hat{\mathbf{p}}$ by defunctionalization:

$$\mathbf{q}_1 = \begin{cases} \hat{\mathtt{fold}}(\mathbf{nil}) \to \mathbf{0} \\ \hat{\mathtt{fold}}(\mathbf{c}(y, l)) \to \mathtt{app}(\mathbf{c}_0, \hat{\mathtt{fold}}(l)) & \mathbf{c}_0 \text{ is a new constructor} \\ \hat{\mathtt{h}}(l) \to \hat{\mathtt{fold}}(l) \end{cases}$$

$$\mathbf{q}_2 = \{ \mathtt{app}(\mathbf{c}_0, x) \to \mathtt{g}(x)$$

Results

- We decrease the complexity of the synthesis procedure with flexible decompositions
- We increase the number of captured algorithms
- These results are implemented in CROCUS `http://libresource.inria.fr/projects/crocus`
  - experiments show that the synthesis is reasonable (1 s for 50 lines of code)