Resource analysis by Sup-interpretations

Jean-Yves Marion et Romain Péchoux

jean-yves.marion@loria.fr, romain.pechoux@loria.fr

Calligramme-Loria

Motivations

Develop a criterion to bound the size of the stack frames computed by a program which terminates or not:

- Dependency pairs method (Arts et Giesl 96)
- the notion of sup-interpretation distinct from the one of quasi-interpretation (Bonfante, Moyen, Marion 00):
  - It ranges over a part of the set of function symbols
  - A relaxed subterm property
- Static analyse
- Practical issues : Computer security
- It applies to functional first order programs

First order Language: syntax

Constructors: $\mathcal{C}$, Operators: Op, functions: $\mathcal{F}$, variables: $\mathcal{X}$.

$$
\begin{array}{lllll}
\texttt{(Terms)} & \ni t & ::= & x \mid \mathbf{c}(t_1, \cdots, t_n) \mid \mathtt{f}(t_1, \cdots, t_n) \\
 & & & \mid \mathbf{Case}\ \overline{e}\ \mathbf{of}\ \overline{p_1} \rightarrow e^1, \ldots, \overline{p_l} \rightarrow e^l \\
 & & & \mid \mathbf{op}(e_1, \cdots, e_n) \\
\texttt{(Patterns)} & \ni p & ::= & \mathbf{c}(p_1, \cdots, p_n) \mid x \\
\texttt{(Definitions)} & \ni d & ::= & \mathtt{f}(x_1, \cdots, x_n) = t \\
\texttt{(Programs)} & \ni p & ::= & d_1, \ldots, d_n
\end{array}
$$

First order Language: semantics

- Call-by-value evaluation:

$$\frac{CBV[\![e_i]\!]\vec{v} = v_i}{CBV[\![c(e_1, ..., e_n)]\!]\vec{v} = c(v_1, ..., v_n)}$$

$$\frac{CBV[\![e_i]\!]\vec{v} = v_i,\ \mathsf{f}(p_1, ..., p_n) \to e,\ \sigma(p_i) = v_i\ \ CBV[\![\sigma(e)]\!]\vec{v} = v}{CBV[\![\mathsf{f}(e_1, ..., e_n)]\!]\vec{v} = v}$$

- The set $Values$: $Values \ni v ::= \ \mathbf{c} \mid \mathbf{c}(v_1, \cdots, v_n)\ \mathbf{c} \in \mathcal{C}$
- Error : $\mathbf{Err}$
- Non-termination : $\perp$

Examples (1/2)

$$\underline{n} = s^n(0)$$

1. Multiplication

$$\begin{aligned} add(0, y) &\rightarrow y \\ add(s(x), y) &\rightarrow s(add(x, y)) \\ mul(0, y) &\rightarrow 0 \\ mul(s(x), y) &\rightarrow add(y, mul(x, y)) \\ CBV[\![mul(x, y)]\!](\underline{n}, \underline{m}) &= \underline{n \times m} \end{aligned}$$

Examples (2/2)

2. Exponential

$$d(0) \to 0$$
$$d(s(x)) \to s(s(d(x)))$$
$$exp(0) \to s(0)$$
$$exp(s(x)) \to d(exp(x))$$
$$CBV[\![exp(x)]\!](\underline{n}) = \underline{2^n}$$

3. Non-termination

$$\mathtt{f}(x) \to \mathtt{f}(x)$$
$$CBV[\![\mathtt{f}(x)]\!](v) = \bot$$

Dependency Pairs (Arts et Giesl 96)

- $< \mathsf{f}(\vec{p}), g(\vec{t}) >$ dependency pair if $\mathsf{f}(\vec{p}) \rightarrow C[g(\vec{t})]$

- Dependency graph:

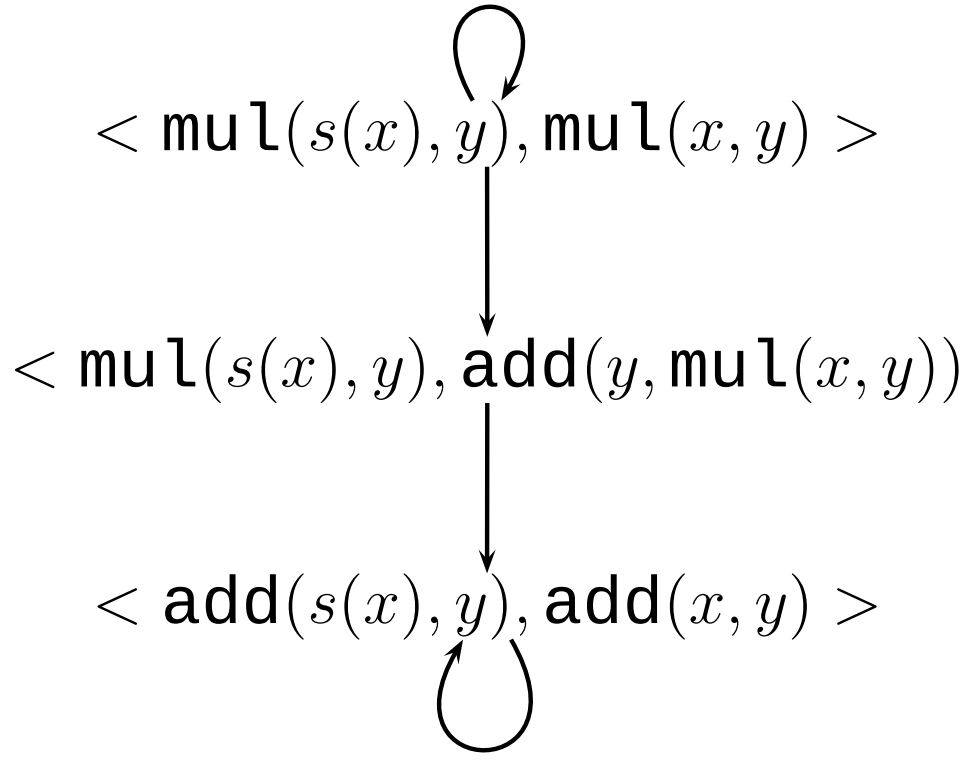
  1. States are dependency pairs

  2. edges are defined by :

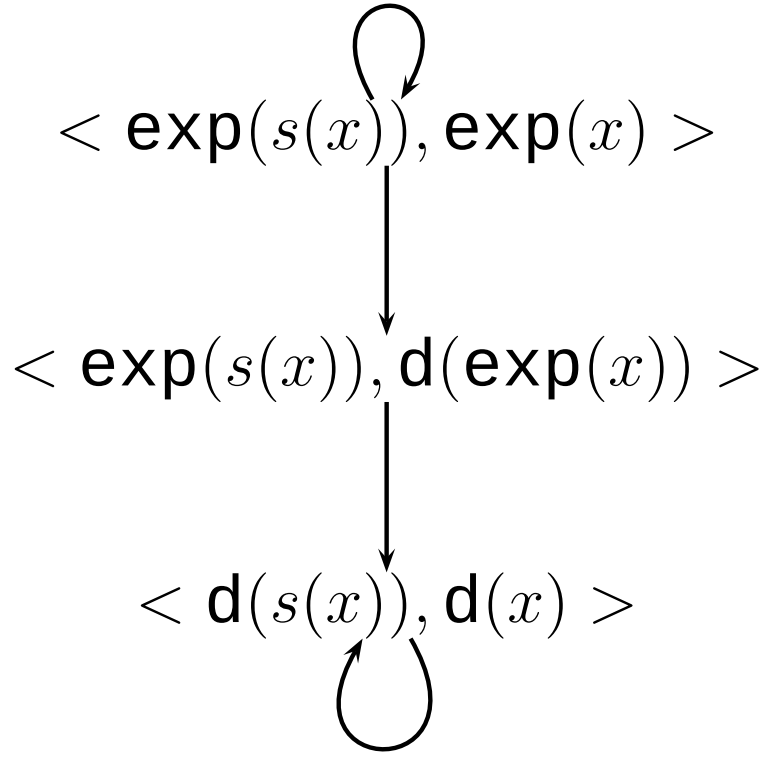
     Given $u =< \mathsf{f}_1(\vec{t_1}), \mathsf{f}_2(\vec{t_2}) >$ and $v =< \mathsf{f}_3(\vec{t_3}), \mathsf{f}_4(\vec{t_4}) >$

     $u \longrightarrow v$ if $\mathsf{f}_2 = \mathsf{f}_3$

Examples : Multiplication (1/2)



Examples : Exponential (2/2)



Precedence

- $\mathsf{f} \geq_{\mathcal{F}} \mathsf{g}$ if $< \mathsf{f}(\vec{p}), \mathsf{g}(\vec{t}) >$ dependency pairs
- $\mathsf{f} \approx_{\mathcal{F}} \mathsf{g}$ if f and g belong to the same cycle
- Precedence used in the construction of the bound.
- Exemple : $mul >_{\mathcal{F}} add$

Sup-interpretation

- Assignment θ such that for f function ou constructor having a sup-interpretation, $\theta(\mathsf{f}) \ : \ \mathbb{R}^n \to \mathbb{R}$
- For x $\in \mathcal{X}$, we take $\theta(x) = X$
- We extend this assignment to terms by θ^*:

$$\theta^*(\mathsf{f}(\vec{v})) = \theta(\mathsf{f})(\theta^*(\vec{v}))$$

  - $\theta^*(\overline{e}) = \max(\theta^*(e_1), \ldots, \theta^*(e_n))$
  - $\theta^*(\mathbf{Case}\ \overline{e}\ \mathbf{of}\ \overline{p_1} \to e^1 \ldots \overline{p_\ell} \to e^\ell) = \max(\theta^*(\overline{e}), \theta^*(e^1), \ldots, \theta^*(e^\ell))$

Sup-interpretation

- A monotonic assignment is a sup-interpretation if it satisfies

$$\forall \vec{v} \text{ vector of values } \theta^*(\mathsf{g}(\vec{v})) \geq \theta^*(CBV[\![\mathsf{g}]\!]\vec{v})$$

$$\theta^*(v) \geq |v| \text{ if } v \text{ value}$$

  - Remark1: Quasi-interpretations are a special case of sup-interpretation (total, subterm property)
  - Remark2: Sup-interpretations define a complexity measure (Blum).

Examples of sup-interpretation

1. Multiplication
   - $\theta(add(x, y)) = X + Y$
   - $\theta(mul(x, y)) = X \times Y$

2. Exponential
   - $\theta(d(x)) = 2X$
   - $\theta(exp(x)) = 2^X$

3. Non-termination
   - $\theta(\mathsf{f}(\vec{x}))$: any since $|\perp| = 0$

Fraternities

- A context C of a rule $\mathsf{f}(\vec{p}) \rightarrow C[\mathsf{g}_1(\vec{s_1}), ..., \mathsf{g}_n(\vec{s_n})]$ is a fraternity if:
  - $\forall\, i \in [1, n]$, $\mathsf{g}_i \approx_{\mathcal{F}} \mathsf{f}$
  - $\forall\, \mathsf{h} \in C$, $\mathsf{f} >_{\mathcal{F}} \mathsf{h}$
- Example:
  - $mul(s(x), y) \rightarrow add(y, mul(x, y))$
  - $add(y, mul(x, y))$ is a fraternity with corresponding context $C[\diamond] = add(y, \diamond)$

Weight

- A weight ω is an assignment which ranges over function symbols and such:
  - For f function symbol, ω_{f} is weakly monotonic
  - For f function symbol, $\omega_{\mathsf{f}}(..., X_i, ...) \geq X_i$
- Examples: $\max$ ou $\sum$
- Application: If $\omega_{\mathsf{f}} = \max$, then :

$$\omega_{\mathsf{f}}(\theta(x), \theta(y)) = \max(\theta(x), \theta(y))$$

Weight

- A weight is a quasi-interpretation in the sense that it has the same functional properties.
- A weight is a control point for the recursive calls
- Some criteria on Sup-interpretations + Weight will provide the polynomial space bound...

Friendly programs

A program P is friendly:

- if there are a polynomial sup-interpretation θ
- and a weight ω
- such that for every rule $\mathsf{f}(\vec{p}) \to C[g_1(\vec{s_1}), ..., g_n(\vec{s_n})]$ with C fraternity:
- and every substitution σ

$$\omega_{\mathsf{f}}(\theta^*(p_1\sigma), ..., \theta^*(p_n\sigma)) \geq \omega_{\mathsf{g}_i}(\theta^*(s_j^1\sigma), ..., \theta^*(s_j^{L(j)}\sigma))$$

(condition I)

friendly programs

Where the fraternity $C[\mathsf{g}_1(\vec{s_1}), ..., \mathsf{g}_n(\vec{s_n})]$ verifies:

$$\theta^*(C[\diamond_1, ..., \diamond_n]) = \max_{i=1..n}(\diamond_i + \alpha_i)$$

and if $\exists\sigma$ such that

$$\omega_{\mathsf{f}}(\theta^*(p_1\sigma), ..., \theta^*(p_n\sigma)) = \omega_{\mathsf{g}_i}(\theta^*(s_j^1\sigma), ..., \theta^*(s_j^{L(j)}\sigma))$$

then α_i is the null polynomial.

(condition II)

- Remark: We are in the case of recursive calls since $\mathsf{f} \approx_{\mathcal{F}} \mathsf{g}_i$

Multiplication: Condition I

$$\mathtt{add}(s(x), y) \rightarrow s(\mathtt{add}(x, y))$$

$$\mathtt{mult}(s(x), y) \rightarrow \mathtt{add}(y, \mathtt{mult}(x, y))$$

$$< \mathtt{add}(s(x), y), \mathtt{add}(x, y) >$$

$$< \mathtt{mult}(s(x), y), \mathtt{mult}(x, y) >$$

- Condition I:

$$\omega_{\mathsf{f}}(\theta^*(p_1\sigma), ..., \theta^*(p_n\sigma)) \geq \omega_{\mathsf{g}_i}(\theta^*(s_j^1\sigma), ..., \theta^*(s_j^{L(j)}\sigma))$$

  - $\omega_{\mathsf{add}}(\theta^*(s(x)), \theta^*(y))) > \omega_{\mathsf{add}}(\theta^*(x), \theta^*(y))$
  - $\theta(s)(X) = X + 1$ and $\omega_{\mathsf{add}}(X, Y) = X + Y$

Multiplication: Condition II

$$\texttt{add}(s(x), y) \rightarrow s(\texttt{add}(x, y))$$

$$\texttt{mult}(s(x), y) \rightarrow \texttt{add}(y, \texttt{mult}(x, y))$$

$$< \texttt{add}(s(x), y), \texttt{add}(x, y) >$$

$$< \texttt{mult}(s(x), y), \texttt{mult}(x, y) >$$

- Condition II:

$$\theta^*(C[\diamond_1, ..., \diamond_n]) = \max_{i=1..n}(\diamond_i + \alpha_i)$$

  - $C = s[\diamond]$ et $\theta^*(s(\diamond)) = \diamond + 1 = \max(\alpha + \diamond)$
  - $C = add[y, \diamond]$ et $\theta^*(add(y, \diamond)) = Y + \diamond = \max(\alpha + \diamond)$

Theorem 1:

Given a friendly program **p**, for every function symbol f, there is P_{f} a polynomial such that:

For every $\vec{v}$, $|CBV[\![\mathsf{f}]\!]\vec{v}| \leq P_{\mathsf{f}}(|\vec{v}|)$

if $CBV[\![\mathsf{f}]\!]\vec{v}$ is defined.

- Proof: We assign a polynomial P_{f} to the program and check that it bounds the values added by the context plus the values added by the recursive calls.

Another example : Division

$$\begin{aligned}
\texttt{minus}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, z &\rightarrow 0 \\
s(z), 0 &\rightarrow s(z) \\
s(u), s(v) &\rightarrow \texttt{minus}(u, v) \\
\texttt{q}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, s(z) &\rightarrow 0 \\
s(z), s(u) &\rightarrow s(\texttt{q}(\texttt{minus}(z, u), s(u)))
\end{aligned}$$

Another example : Division

$$\begin{aligned}
\texttt{minus}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, z &\to 0 \\
s(z), 0 &\to s(z) \\
s(u), s(v) &\to \texttt{minus}(u, v) \\
\texttt{q}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, s(z) &\to 0 \\
s(z), s(u) &\to s(\texttt{q}(\texttt{minus}(z, u), s(u)))
\end{aligned}$$

1. $\omega_{\texttt{minus}}(\theta^*(s(x)), \theta^*(s(y))) \geq \omega_{\texttt{minus}}(\theta(x), \theta(y))$

   $\omega_{\texttt{q}}(\theta^*(s(z)), \theta^*(s(u))) \geq \omega_{\texttt{q}}(\theta^*(\texttt{minus}(z, u)), \theta^*(s(u)))$

Another example : Division

$$\begin{aligned}
\texttt{minus}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, z &\to 0 \\
s(z), 0 &\to s(z) \\
s(u), s(v) &\to \texttt{minus}(u, v) \\
\texttt{q}(x, y) = \textbf{Case}\ x, y\ \textbf{of}\ 0, s(z) &\to 0 \\
s(z), s(u) &\to s(\texttt{q}(\texttt{minus}(z, u), s(u)))
\end{aligned}$$

1. $\omega_{\texttt{minus}}(\theta^*(s(x)), \theta^*(s(y))) \geq \omega_{\texttt{minus}}(\theta(x), \theta(y))$

   $\omega_{\texttt{q}}(\theta^*(s(z)), \theta^*(s(u))) \geq \omega_{\texttt{q}}(\theta^*(\texttt{minus}(z, u)), \theta^*(s(u)))$
2. $\theta(s)(\diamond) = \diamond + \alpha$

Another example : Division

1. $\omega_{\texttt{minus}}(\theta^*(s(x)), \theta^*(s(y))) \geq \omega_{\texttt{minus}}(\theta(x), \theta(y))$

   $\omega_{\texttt{q}}(\theta^*(s(z)), \theta^*(s(u))) \geq \omega_{\texttt{q}}(\theta^*(\texttt{minus}(z, u)), \theta^*(s(u)))$

2. $\theta(s)(\diamond) = \diamond + \alpha$

$\theta(s)(X) = X + 1$,

$\omega_{\texttt{minus}}(X, Y) = \max(X, Y)$,

$\omega_{\texttt{q}}(X, Y) = X + Y$

$\theta(\texttt{minus})(X, Y) = X$

Exponential: an unfriendly program

$$\begin{aligned}
\mathsf{double}(x) &= \mathbf{Case}\ x\ \mathbf{of}\ 0 \to 0 \\
&\qquad\qquad s(y) \to s(s(\mathsf{double}(y))) \\
\mathsf{exp}(x) &= \mathbf{Case}\ x\ \mathbf{of}\ 0 \to s(0) \\
&\qquad\qquad s(y) \to \mathsf{double}(\mathsf{exp}(y))
\end{aligned}$$

1. $\omega_{\mathsf{double}}(\theta^*(s(y))) \geq \omega_{\mathsf{double}}(\theta(y))$

   $\omega_{\mathsf{exp}}(\theta^*(s(y))) \geq \omega_{\mathsf{exp}}(\theta^*(y))$

2. $\theta(\mathsf{double})(\diamond) = \diamond + \alpha$ but $\theta(\mathsf{double})(\diamond) \geq 2\diamond$

Call-tree

- If there is a rule $\mathsf{f}(p_1, \cdots, p_n) \to C[\mathsf{g}(e_1, \cdots, e_k)]$, a subsitution σ such that:
  - $p_i\sigma = u_i$
  - $CBV[\![e_j\sigma]\!] = v_j$
- Then we write $\langle \mathsf{f}, u_1, \cdots, u_n \rangle \overset{\mathsf{C}[\diamond]}{\rightsquigarrow} \langle \mathsf{g}, v_1, \cdots, v_k \rangle$
- The corresponding graph is called a call-tree of f over values $u_1, \ldots, u_n$ if $\langle \mathsf{f}, u_1, \cdots, u_n \rangle$ is its root.
- A call-tree gives a static view of an execution of a program.
- The states can be seen as stack frames

Theorem 2:

Given a friendly program **p**, for every function symbol f, there is P_{f} a polynomial such that for every state $\langle \mathsf{g}, v_1, \cdots, v_k \rangle$ of the call-tree of root $\langle \mathsf{f}, u_1, \cdots, u_n \rangle$:

$$\max_{i=1..k}(|v_i|) \leq P_{\mathsf{f}}(|\vec{u}|)$$

even if the program is non-terminating.

- non-terminating programs
- programs over unbounded datas such as infinite streams

List of examples

- logarithm
- division
- reachability over graphs
- elimination of duplicates in a list

- All these programs admit an S.I. but no Q.I.
- However there are programs admitting QI and not friendly (In particular, programs over trees)

Open problems

- Try to capture the QI with a more general criteria
- Problem of synthesis (as Amadio did for the QI Amdio04)
- Try to find restriction in order to capture polynomial time