Program complexity analysis by semantics interpretation

Romain Péchoux

CS department, Trinity College, Dublin

Foundations@Lero Seminar, 17/10/2008

Motivations

- Resource control of Time/Space by static analysis
- A ("decidable") certificate should ensure a given ("undecidable") property P:

Example (Termination)

- P(x): The property of the program x to be a terminating.
- A proof of termination is a certificate.

Example (Complexity classes)

- P(x): The property of the program x to compute a function of a given complexity class C.
- A proof that the program corresponds to some characterization of C is a certificate.

If **p** is in FPTIME wrt to some criterion, $\exists P,\ s.t.\ [\![\mathbf{p}]\!](v) \Downarrow^{n \leq P(|v|)}$

- A next step is to compute the P wrt some system.

Historical background

Functions algebra: $[Op; F]$

Theorem [Cobham [1965]]

$\mathrm{FPTIME} \equiv [Op; COMP(f, g), BRN(g_0, g_1, h_0, h_1, k)]$

$$Op = \{\ 0, \pi_i^n, s_i(x) = 2x + i, \#(x, y) = 2^{|x|\times|y|}\ \}$$

$$BRN(g_0, g_1, h_0, h_1, k): \quad f(s_i(0), y) = g_i(y)$$
$$f(s_i(x), y) = h_i(x, y, f(x, y))$$
$$f(x, y) < k(x, y)$$

- Undecidable because of the bound k.
  - FPTIME Bellantoni and Cook[1992] and by Leivant and Marion.
- Weak intensionality (Few programs can be written)
  - Distinct answers provided by the Implicit Computational Complexity.

Historical background

Implicit computational complexity (ICC)

- Linear type discipline BLL and LLL by Girard et al.[1992,1998] and SLL by Lafont[2004].
- Linear use of variables using a special diamond resource by Hofmann[1999]
- Studies on the complexity of imperative programs Jones-Kristiansen[2005] and Niggl-Wunderlich[2006].
- Interpretations and quasi-interpretations by Bonfante, Marion and Moyen[2000,2001]

Our approach

Interpretation of each symbol + constraints

- concern Term Rewriting Systems
- introduced for program termination by Manna-Ness[1972] and Lankford[1979]
- are closer to intensionality than to extensionality

1. Polynomial interpretation
2. Quasi-interpretation
3. Sup-interpretation
4. Extension to OO paradigm

Syntax and semantics of First Order Language

A program is a quadruplet $\langle \mathcal{X}, \mathcal{C}, \mathcal{F}, \mathcal{R} \rangle$

Constructors: $\mathbf{c} \in \mathcal{C}$, functions: $\mathtt{f} \in \mathcal{F}$, variables: $x \in \mathcal{X}$.

$$\begin{array}{lllll} \texttt{(Terms)} & \mathcal{T} \ni t & ::= & x \mid \mathbf{c}(t_1, \cdots, t_n) \mid \mathtt{f}(t_1, \cdots, t_n) \\ \texttt{(Patterns)} & \mathcal{P} \ni p & ::= & \mathbf{c}(p_1, \cdots, p_n) \mid x \\ \texttt{(Rules)} & \mathcal{R} \ni r & ::= & \mathtt{f}(p_1, \cdots, p_n) \to t \\ \texttt{(Values)} & \mathcal{V} \ni v & ::= & \mathbf{c}(v_1, \cdots, v_n) \end{array}$$

- The computational domain is the set of values $\mathcal{V}^* = \mathcal{V} \cup \{\mathbf{Err}\}$
- Define $[\![e]\!] = w$ iff $e \xrightarrow{*} w$ and $w \in \mathcal{V}^*$
- Linear and disjoint patterns (Confluence: Huet[1980])

Program example

Example (Unary logarithm)

$$\begin{aligned}
\mathtt{half}(\mathbf{0}) &\to^1 \mathbf{0} && \lfloor 0/2 \rfloor = 0\\
\mathtt{half}(\mathbf{S}(\mathbf{0})) &\to^2 \mathbf{0} && \lfloor 1/2 \rfloor = 0\\
\mathtt{half}(\mathbf{S}(\mathbf{S}(x))) &\to^3 \mathbf{S}(\mathtt{half}(x)) && \lfloor (x+2)/2 \rfloor = 1 + \lfloor x/2 \rfloor\\
\mathtt{log}(\mathbf{0}) &\to^4 \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{0})) &\to^5 \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{S}(y))) &\to^6 \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y)))) && \mathtt{log}(y+2) = 1 + \mathtt{log}(1 + y/2)
\end{aligned}$$

$$\mathtt{log}(\mathbf{S}(\mathbf{S}(\mathbf{S}(0)))) \to^6 \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(\mathbf{S}(0))))) \to^2 \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathbf{0}))) \to^5 \mathbf{S}(\mathbf{0})$$

Semantics:

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

Polynomial interpretations

Definition [Assignment]

An assignment $(\!|-|\!)$ is an application mapping:

- $x \in \mathcal{X}$ to a new variable $(\!|x|\!) = X \in \mathbb{N}$
- $b \in Cns \cup Fct$ of arity n to a function $(\!|b|\!) : (\mathbb{N})^n \to \mathbb{N}$

Definition [Polynomial interpretation]

A program $\mathbf{p}$ admits a polynomial interpretation if there is a

- total: $\forall b$, $(\!|b|\!)$ is defined,
- monotonic: $X_i > Y_i \Rightarrow (\!|b|\!)(\ldots, X_i, \ldots) > (\!|b|\!)(\ldots, Y_i, \ldots)$,
- and polynomial: $\forall b, (\!|b|\!)(X_1, \cdots, X_n) \in \mathbb{N}[X_1, \cdots, X_n]$

assignment $(\!|-|\!)$ such that for each rule $l \to r$:

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

Example of polynomial interpretation

Example (Assignment of logarithm)

$$\begin{aligned}
\mathtt{half}(\mathbf{0}) &\to \mathbf{0}\\
\mathtt{half}(\mathbf{S}(\mathbf{0})) &\to \mathbf{0}\\
\mathtt{half}(\mathbf{S}(\mathbf{S}(y))) &\to \mathbf{S}(\mathtt{half}(y))\\
\mathtt{log}(\mathbf{0}) &\to \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{0})) &\to \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{S}(y))) &\to \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))
\end{aligned}$$

$$\begin{aligned}
(\!|\mathbf{0}|\!) &= 0\\
(\!|\mathbf{S}|\!)(X) &= X + 2\\
(\!|\mathtt{half}|\!)(X) &= X + 1\\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{half}(y))|\!) \\
(\!|\mathtt{log}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0 \\
(\!|\mathbf{S}|\!)(X) &= X + 2 \\
(\!|\mathtt{half}|\!)(X) &= X + 1 \\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}
$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}|\!)((\!|\mathbf{S}|\!)((\!|\mathbf{S}|\!)((\!|y|\!)))) &> (\!|\mathbf{S}|\!)((\!|\mathtt{half}|\!)((\!|y|\!))) \quad \text{Canonically} \\
(\!|\mathtt{log}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0 \\
(\!|\mathbf{S}|\!)(X) &= X + 2 \\
(\!|\mathtt{half}|\!)(X) &= X + 1 \\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}
$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}|\!)((\!|\mathbf{S}|\!)((\!|\mathbf{S}|\!)(Y))) &> (\!|\mathbf{S}|\!)((\!|\mathtt{half}|\!)(Y)) \quad (\!|y|\!) = Y \\
(\!|\mathtt{log}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0 \\
(\!|\mathbf{S}|\!)(X) &= X + 2 \\
(\!|\mathtt{half}|\!)(X) &= X + 1 \\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}
$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathbf{S}|\!)((\!|\mathbf{S}|\!)(Y)) + 1 &> (\!|\mathbf{S}|\!)(Y+1) \quad (\!|\mathtt{half}|\!)(X) = X + 1 \\
(\!|\mathtt{log}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0 \\
(\!|\mathbf{S}|\!)(X) &= X + 2 \\
(\!|\mathtt{half}|\!)(X) &= X + 1 \\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}
$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
Y + 5 &> Y + 3 \quad (\!|\mathbf{S}|\!)(X) = X + 2 \\
(\!|\mathtt{log}(\mathbf{0})|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &> (\!|\mathbf{0}|\!) \\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &> (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0 \\
(\!|\mathbf{S}|\!)(X) &= X + 2 \\
(\!|\mathtt{half}|\!)(X) &= X + 1 \\
(\!|\mathtt{log}|\!)(X) &= X^2 + 1
\end{aligned}
$$

Example of polynomial interpretation

Example (Polynomial interpretation of logarithm)

$$1 > 0$$
$$3 > 0$$
$$Y + 5 > Y + 3$$
$$1 > 0$$
$$5 > 0$$
$$(Y + 4)^2 + 1 > (Y + 3)^2 + 3$$

$$(\!|\mathbf{0}|\!) = 0$$
$$(\!|\mathbf{S}|\!)(X) = X + 2$$
$$(\!|\mathtt{half}|\!)(X) = X + 1$$
$$(\!|\mathtt{log}|\!)(X) = X^2 + 1$$

Termination

Theorem [Termination]

A program admitting a polynomial interpretation is terminating (strongly normalizing).

Proof.

If $u \to v$ then $(\!|u|\!) > (\!|v|\!)$
There are: a context $C[\diamond]$, a rule $l \to r$ and a substitution σ such that $u = C[l\sigma]$ and $v = C[r\sigma]$.

- Polynomial interpretations imply $(\!|l\sigma|\!) > (\!|r\sigma|\!)$
- Monotonicity implies $(\!|C[l\sigma]|\!) > (\!|C[r\sigma]|\!)$

Consequently:

$$\begin{array}{ccccccc} u_1 & \to & u_2 & \to & \dots & \to & u_n \\ (\!|u_1|\!) & > & (\!|u_2|\!) & > & \dots & > & (\!|u_n|\!) \end{array}$$

Quasi-interpretations (QI)

Definition [Quasi-interpretation]

A program **p** admits a quasi-interpretation if there is a

- total: $\forall b$, $(\!|b|\!)$ is defined,
- monotonic: $X_i \geq Y_i \Rightarrow (\!|b|\!)(\ldots, X_i, \ldots) \geq (\!|b|\!)(\ldots, Y_i, \ldots)$,
- and subterm: $(\!|b|\!)(X_1, \cdots, X_n) \geq X_i$

assignment $(\!|-|\!)$ such that for each rule $l \to r$:

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

Example of quasi-interpretation

Example (Assignment of logarithm)

$$\begin{aligned}
\mathtt{half}(\mathbf{0}) &\to \mathbf{0}\\
\mathtt{half}(\mathbf{S}(\mathbf{0})) &\to \mathbf{0}\\
\mathtt{half}(\mathbf{S}(\mathbf{S}(y))) &\to \mathbf{S}(\mathtt{half}(y))\\
\mathtt{log}(\mathbf{0}) &\to \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{0})) &\to \mathbf{0}\\
\mathtt{log}(\mathbf{S}(\mathbf{S}(y))) &\to \mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))
\end{aligned}$$

$$\begin{aligned}
(\!|\mathbf{0}|\!) &= 0\\
(\!|\mathbf{S}|\!)(X) &= X + 1\\
(\!|\mathtt{log}|\!)(X) = (\!|\mathtt{half}|\!)(X) &= X
\end{aligned}$$

Example of quasi-interpretation

Example (Quasi-interpretation of logarithm)

$$
\begin{aligned}
(\!|\mathtt{half}(\mathbf{0})|\!) &\geq (\!|\mathbf{0}|\!)\\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{0}))|\!) &\geq (\!|\mathbf{0}|\!)\\
(\!|\mathtt{half}(\mathbf{S}(\mathbf{S}(y)))|\!) &\geq (\!|\mathbf{S}(\mathtt{half}(y))|\!)\\
(\!|\mathtt{log}(\mathbf{0})|\!) &\geq (\!|\mathbf{0}|\!)\\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{0}))|\!) &\geq (\!|\mathbf{0}|\!)\\
(\!|\mathtt{log}(\mathbf{S}(\mathbf{S}(y)))|\!) &\geq (\!|\mathbf{S}(\mathtt{log}(\mathbf{S}(\mathtt{half}(y))))|\!)
\end{aligned}
$$

$$
\begin{aligned}
(\!|\mathbf{0}|\!) &= 0\\
(\!|\mathbf{S}|\!)(X) &= X + 1\\
(\!|\mathtt{log}|\!)(X) &= (\!|\mathtt{half}|\!)(X) = X
\end{aligned}
$$

Example of quasi-interpretation

Example (Quasi-interpretation of logarithm)

$$0 \geq 0$$
$$1 \geq 0$$
$$Y + 2 \geq Y + 1$$
$$0 \geq 0$$
$$1 \geq 0$$
$$Y + 2 \geq Y + 2$$

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

Restrictions on QI

Definition [Polynomial quasi-interpretation]

A quasi-interpretation is polynomial if $\forall b\ (\!|b|\!)$ is a max-polynomial.

Definition [Additive quasi-interpretation]

A quasi-interpretation is additive if $\forall \mathbf{c} \in Cns$ of arity n:

- if $n > 0$ then $(\!|\mathbf{c}|\!)(X_1, \cdots, X_n) = \sum_{i=1}^{n} X_i + \alpha_{\mathbf{c}}$ with $\alpha_{\mathbf{c}} \geq 1$
- if $n = 0$ then $(\!|\mathbf{c}|\!) = 0$

Small Lemma

If a program admits an additive QI then there is a constant $\alpha > 0$ s.t. $\forall v \in \mathcal{V}$:

$$|v| \leq (\!|v|\!) \leq \alpha \times |v|$$

Fundamental Lemma

Lemma

If a program admits a polynomial and additive QI then there is a polynomial P s.t. $\forall b$ and $\forall v_1, \ldots, v_n \in$ Values:

$$\text{if } [\![b(v_1, \ldots, v_n)]\!] \in \mathcal{V}, \text{ then } P(\max_{i=1,n}(|v_i|)) \geq |[\![b(v_1, \ldots, v_n)]\!]|$$

Proof.

$P(X) = (\!|b|\!)(\alpha \times X, \ldots, \alpha \times X)$ is the required polynomial

$$\begin{array}{ccccccc} b(v_1, \ldots, v_n) & \rightarrow & u_2 & \rightarrow & \ldots & \rightarrow & [\![b(v_1, \ldots, v_n)]\!] \\ (\!|b(v_1, \ldots, v_n)|\!) & \geq & (\!|u_2|\!) & \geq & \ldots & \geq & (\!|[\![b(v_1, \ldots, v_n)]\!]|\!) \end{array}$$

$$\begin{array}{lll} (\!|b|\!)(\alpha \times |v_1|, \ldots, \alpha \times |v_n|) \geq (\!|b|\!)((\!|v_1|\!), \ldots, (\!|v_n|\!)) & & \text{Small Lemma} \\ \geq (\!|[\![b(v_1, \ldots, v_n)]\!]|\!) & & \\ \geq |[\![b(v_1, \ldots, v_n)]\!]| & & \text{Small Lemma} \end{array}$$

Recursive Path Ordering

Definition

Given a precedence $\geq_{Fct}$ and a status $st(\mathtt{f}) \in \{p, l\}$ define :

$$\frac{u \preceq_{rpo} t_i}{u \prec_{rpo} f(\ldots, t_i, \ldots)} \; f \in Fct \cup Cns$$

$$\frac{u_i \prec_{rpo} \mathtt{f}(t_1, \cdots, t_n) \qquad g \prec_{Fct} \mathtt{f}}{g(u_1, \cdots, u_m) \prec_{rpo} \mathtt{f}(t_1, \cdots, t_n)} \; g \in Fct \cup Cns$$

$$\frac{(u_1, \cdots, u_n) \prec_{rpo}^{st(\mathtt{f})} (t_1, \cdots, t_n) \qquad \mathtt{f} \approx_{Fct} \mathtt{g} \qquad u_j \prec_{rpo} \mathtt{f}(t_1, \cdots, t_n)}{\mathtt{g}(u_1, \cdots, u_n) \prec_{rpo} \mathtt{f}(t_1, \cdots, t_n)}$$

- An program is ordered by RPO if $\forall l \to r \in \mathcal{R}$, $l \succ_{rpo} r$

Characterizations of FPTIME and FPSPACE

Theorem [Bonfante, Marion and Moyen[2000,2001]]

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

- ordered by product $\prec_{rpo}$ is exactly the set of FPTIME functions.
- ordered by product and lexicographic $\prec_{rpo}$ is exactly the set of FPSPACE functions.

Quasi-interpretations synthesis

Definition [The synthesis problem (Amadio[2004])]

- Input: a program **p**
- Output: a quasi-interpretation of **p** (if there is one)

Definition [The verification problem]

- Input: a program **p** and an assignment $(\!|-|\!)$
- Output: true if $(\!|-|\!)$ is a quasi-interpretation of **p** (false otherwise)

QI synthesis on **Max-Plus**

$$\textbf{Max-Plus}\,\{\mathbb{K}\} = \left[\alpha \in \mathbb{K}, \pi_i^j; +, \max\right]$$

Theorem [Amadio[2004]]

The synthesis problem is:

- NPTIME-hard on **(k,d)-Max-Plus** $\{\mathbb{N}\}$
- NPTIME-complete on **Max-Plus** $\{0, 1\}$

Theorem [Péchoux[2004]]

The synthesis problem is:

- NPTIME-complete on **(k,d)-Max-Plus** $\{\mathbb{N}\}$
- NPTIME-hard on **(k,d)-Max-Plus** $\{\mathbb{R}^+\}$

The verification problem is in FPTIME over **(k,d)-Max-Plus** $\{\mathbb{R}^+\}$

QI synthesis on **Max-Poly**

$$\textbf{Max-Poly}(\mathbb{K}) = \left[\alpha \in \mathbb{K}, \pi_i^j; +, \max, \times\right]$$

Theorem [Bonfante, Marion, Moyen and Péchoux[2005]]

The synthesis problem is decidable in exponential time in **(k,d)-Max-Poly** $\{\mathbb{R}^+\}$.

Quantifier elimination of first order formula is:

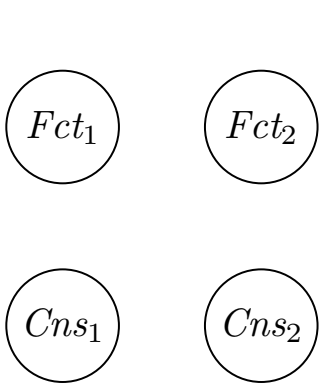
- decidable (Tarski[1951])
- decidable in $2^{2^{O(n)}}$ (Collins[1975])
- decidable in $2^{n^{O(m)}}$ (Roy-Heintz-Solernó[1990])

Modularity: a way to decrease the complexity of the synthesis problem

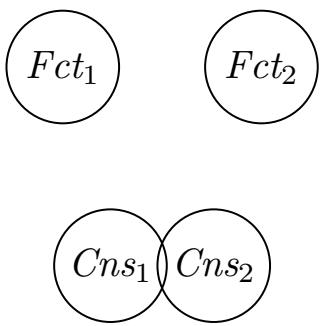
- Modular property P: $P(\mathbf{p}_1)$ and $P(\mathbf{p}_2)$ iff $P(\mathbf{p}_1 \cup \mathbf{p}_2)$
- Introduced by Toyama[1987] for TRS:
  - Termination is not a modular property:

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

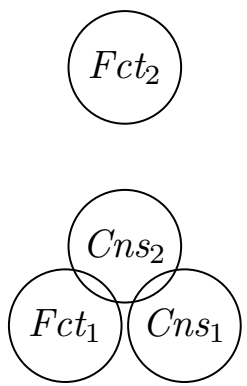
- 3 decompositions are considered (Gramlich, Klop[1992]):



Disjoint union



Constructor-sharing union



Hierarchical union

Modularity of quasi-interpretations

Theorem [Bonfante, Marion and Péchoux[2007]]

Property / Union	QI	Fundamental Lemma	RPO	FPspace FPtime
Disjoint	✓	✓	✓	✓
Constructor-sharing	✗	✓	✓	✓
Hierarchical	✗	✗	✓	✗

Non modular Fundamental Lemma for hierarchical union

Proposition

The Fundamental Lemma does not hold for hierarchical union.

Example (Exponential)

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

New notion: stratified union (syntactical restriction of hierarchical union)

Results on quasi-interpretations

- Upper bounds on the size of computed values
- Characterizations of FPTIME and $\mathrm{FPSPACE}$
- Decidable synthesis problem
- Implementation in CROCUS
  `http://libresource.inria.fr/projects/crocus`
- Extensions to other languages:
  - Higher-order by defunctionalization (Bonfante, Marion and Péchoux[2007]),
  - Bytecode verification(Amadio et al.[2004-2005]),
  - Concurrent and interactive threads (Amadio and Dal-Zilio[2004], Dabrowski[2007])

Quasi-interpretations fail on natural algorithms

Example (A motivating example: Division)

$$\begin{aligned}
\mathtt{m}(\mathbf{0}, y) &\to \mathbf{0} & 0 - y &= 0\\
\mathtt{m}(\mathbf{S}(x), \mathbf{0}) &\to \mathbf{S}(x) & (x+1) - 0 &= x+1\\
\mathtt{m}(\mathbf{S}(x), \mathbf{S}(y)) &\to \mathtt{m}(x, y) & (x+1) - (y+1) &= x - y\\
\mathtt{q}(\mathbf{0}, \mathbf{S}(y)) &\to \mathbf{0} & \frac{0}{y+1} &= 0\\
\mathtt{q}(\mathbf{S}(x), \mathbf{S}(y)) &\to \mathbf{S}(\mathtt{q}(\mathtt{m}(x, y), \mathbf{S}(y))) & \frac{x+1}{y+1} &= 1 + \frac{x-y}{y+1}
\end{aligned}$$

$$[\![\mathtt{q}(\mathbf{S}^n(\mathbf{0}), \mathbf{S}^m(\mathbf{0}))]\!] = \mathbf{S}^{\lceil n/m \rceil}(\mathbf{0})$$

Is there a quasi-interpretation of division?

Example (Division)

Suppose that $(\!|-|\!)$ is an additive QI s.t. $(\!|\mathbf{S}|\!)(X) = X + k$.

$$
\begin{aligned}
(\!|\mathtt{q}(\mathbf{S}(x), \mathbf{S}(y))|\!) &\geq (\!|\mathbf{S}(\mathtt{q}(\mathtt{m}(x,y), \mathbf{S}(y)))|\!) \\
&\geq k + (\!|\mathtt{q}|\!)((\!|\mathtt{m}|\!)(X,Y), Y + k) && (\!|\mathbf{S}|\!)(X) = X + k \\
&> (\!|\mathtt{q}|\!)(\max(X,Y), Y + k) && \text{Subterm property} \\
&\geq (\!|\mathtt{q}|\!)(X + k, Y + k) && \text{Pour } Y \geq X + k \\
&= (\!|\mathtt{q}(\mathbf{S}(x), \mathbf{S}(y))|\!)
\end{aligned}
$$

Develop a tool with more intensionality:

- Partial instead of total assignments
- No more subterm property
- Possibility to consider operators

Sup-interpretations

Definition [Sup-interpretation]

A program admits a sup-interpretation if there is a partial assignment θ such that:

1. θ is monotonic.
2. $\forall v \in \mathcal{V}$, $\theta(v) \geq |v|$
3. $\forall b \in \mathrm{dom}(\theta)$ of arity n and for each values $v_1, \ldots, v_n$, if $[\![b(v_1, \ldots, v_n)]\!] \in \mathcal{V}$, then:

$$\theta(b(v_1, \ldots, v_n)) \geq \theta([\![b(v_1, \ldots, v_n)]\!])$$

Example of additive and polynomial sup-interpretation

Example (Sup-interpretation of division)

$$\theta(\mathbf{0}) = 0 \qquad \theta(\mathbf{S})(X) = X + 1$$
$$\theta(\mathtt{m})(X, Y) = X \quad \theta(\mathtt{q})(X, Y) = X + Y$$

1. θ is monotonic
2. $\forall n \in \mathbb{N},\ \theta(\mathbf{S}^n(\mathbf{0})) = n = |\mathbf{S}^n(\mathbf{0})|$
3. In particular, for $\mathtt{q}$ we check:

$$\begin{aligned}
\theta(\mathtt{q}(\mathbf{S}^n(\mathbf{0}), \mathbf{S}^m(\mathbf{0})) &= \theta(\mathtt{q})(\theta(\mathbf{S}^n(\mathbf{0})), \theta(\mathbf{S}^m(\mathbf{0}))) \\
&= \theta(\mathbf{S}^n(\mathbf{0})) + \theta(\mathbf{S}^m(\mathbf{0})) \\
&= n + m \\
&\geq \lceil n/m \rceil \\
&= \theta(\llbracket \mathtt{q}(\mathbf{S}^n(\mathbf{0}), \mathbf{S}^m(\mathbf{0})) \rrbracket)
\end{aligned}$$

Fraternities

Definition [Precedence]

- $\mathtt{f} \geq_{Fct} \mathtt{g}$ if $\mathtt{f}(\vec{p}) \rightarrow C[\mathtt{g}(\vec{t})] \in \mathcal{R}$.
- $\mathtt{f} \approx_{Fct} \mathtt{g}$ if $\mathtt{f} \geq_{Fct} \mathtt{g}$ and $\mathtt{g} \geq_{Fct} \mathtt{f}$.
- $\mathtt{f} >_{Fct} \mathtt{g}$ if $\mathtt{f} \geq_{Fct} \mathtt{g}$ and not $\mathtt{f} \geq_{Fct} \mathtt{g}$

Definition [Fraternity]

$\mathtt{f}(\overline{p}) \rightarrow e$ is a fraternity if:

1. $e = C[\mathtt{g}_1(\overline{e_1}), \ldots, \mathtt{g}_m(\overline{e_m})]$
2. $\forall\ i \in \{1, m\}$, $\mathtt{g}_i \approx_{Fct} \mathtt{f}$
3. $\forall\ \mathtt{h} \in C[-]$, $\mathtt{f} >_{Fct} \mathtt{h}$

Example of Fraternity

Example (Fraternities of division)

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

- Precedence: $\mathtt{q} >_{Fct} \mathtt{m}$
- Fraternities:
  - $\mathtt{m}(\mathbf{S}(x), \mathbf{S}(y)) \to \mathtt{m}(x, y)$ with $C[\diamond] = \diamond$
  - $\mathtt{q}(\mathbf{S}(x), \mathbf{S}(y)) \to \mathbf{S}(\mathtt{q}(\mathtt{m}(x, y), \mathbf{S}(y)))$ with $C[\diamond] = \mathbf{S}(\diamond)$

Quasi-friendly criterion

Definition [Quasi-friendly program]

A program **p** is *quasi-friendly* iff there are:

- an additive and polynomial sup-interpretation θ
- a polynomial, monotonic and subterm partial assignment ω

s.t. for each fraternity $\mathtt{f}(p_1, \cdots, p_n) \rightarrow \mathsf{C}[\mathsf{g}_1(\overline{e_1}), \ldots, \mathsf{g}_r(\overline{e_r})]$:

$$\omega_{\mathtt{f}}(\theta(p_1), \ldots, \theta(p_n)) \geq \theta(\mathsf{C})(\omega_{\mathsf{g}_1}(\theta(\overline{e_1})), \ldots, \omega_{\mathsf{g}_r}(\theta(\overline{e_r})))$$

Theorem [Marion and Péchoux[2006]]

If a program is quasi-friendly then the Fundamental Lemma holds.

Example of quasi-friendly program

Example (Division is quasi-friendly)

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

Quasi-friendly criterion:

$$\omega_{\mathtt{q}}(\theta(\mathbf{S}(x)), \theta(\mathbf{S}(y))) \geq \theta(\mathbf{S})(\omega_{\mathtt{q}}(\theta(\mathtt{m}(x, y)), \theta(\mathbf{S}(y))))$$

$$\begin{aligned}
\theta(\mathbf{0}) &= 0 & \theta(\mathbf{S})(X) &= X + 1 \\
\theta(\mathtt{m})(X, Y) &= X & \omega_{\mathtt{q}}(X, Y) &= X + Y
\end{aligned}$$

Results on sup-interpretations

- Criterion for:
  - non-terminating programs, programs over streams
  - programs using destructors (including Quicksort)

Proposition

- Every additive quasi-interpretation is a sup-interpretation.
- Every program having an additive QI is quasi-friendly.

- Characterizations of the NC^k complexity Péchoux et al.[2006,2008]
- Synthesis of non subterm sup-interpretation with dependency pairs Péchoux and Marion [2008]

Is this framework adaptable for object-oriented programs ?

- Sup-interpretations allow to consider operators.
- Studies on the complexity of imperative programs using matrix algebra:
  - Jones-Kristiansen[2005]
  - Niggl-Wunderlich[2006]
- Polynomial algebra instead of Matrix algebra:
  - Conditions are more intuitive
  - allows to consider method calls easily
- Non-recursive object oriented programs inspired by:
  - Java fragment of Drossopoulou-Eisenbach[1998] (Side effects)
  - FeatherweightJava[1999] (Objects as values)

Syntax and semantics of object-oriented programs

$$
\begin{array}{lll}
\texttt{Attributes} & \ni A & ::= \texttt{var}\ X; \mid \texttt{var}\ X; A \\
\texttt{Expressions} & \ni t & ::= x \mid X \mid X.\texttt{f}(t_1, \cdots, t_n) \mid \mathbf{op}(t_1, \cdots, t_n) \\
 & & \mid \texttt{new C}(t_1, \cdots, t_n) \\
\texttt{Commands} & \ni \texttt{Cm} & ::= \texttt{skip} \mid X := t \mid \texttt{Cm}_1; \texttt{Cm}_2 \mid \texttt{loop}\ X\,\{\texttt{Cm}\} \\
 & & \mid \texttt{if}\ e\ \texttt{then Cm}_1\ \texttt{else Cm}_2 \mid \texttt{while}\ e\,\{\texttt{Cm}\} \\
\texttt{Methods} & \ni \texttt{M} & ::= \texttt{f}(x_1, ..., x_n)\,\{\texttt{Cm}\ ;\ \texttt{return}\ X;\} \\
\texttt{Class} & \ni \texttt{C} & ::= \texttt{Class C}\,\{A\ ;\ \ M_1; \ldots; M_n;\} \\
 & \texttt{Main} & ::= \texttt{Class Main}\,\{A\ ;\ \texttt{Cm}\}
\end{array}
$$

- Domain: $\texttt{Objects} \ni o ::= \texttt{new}\ b(o_1, \cdots, o_n),\ \ b \in \texttt{Class}$
- Store $\sigma : A \to \texttt{Objects}$
- Semantics of an expression $\langle e, \sigma \rangle \to \langle o, \sigma' \rangle$
- Semantics of a command $\langle \texttt{Cm}, \sigma \rangle \to \langle \sigma' \rangle$

Example (Night of the living dead)

$$\begin{array}{l}
\texttt{Class Zombie}\{\texttt{var } X;\ \texttt{var } Y;\\
\qquad\qquad \texttt{move}(x,y)\{\\
\qquad\qquad X := X.\texttt{add}(x);\\
\qquad\qquad Y := Y.\texttt{add}(y);\\
\qquad\qquad \texttt{return } X;\ \}\\
\}\\
\quad \texttt{Class Main}\{\texttt{var } U; \texttt{var } V;\\
\qquad\qquad V := \texttt{new Zombie}(4,0);\\
\qquad\qquad U := V.\texttt{move}(5,5);\\
\}
\end{array}$$

$$\langle V := \texttt{new Zombie}(4,0), \sigma\rangle \to \langle \sigma\,\{V \leftarrow \texttt{new Zombie}(4,0)\}\rangle$$
$$\langle V.\texttt{move}(5,5), \sigma\rangle \to \langle 9, \sigma\,\{V \leftarrow \texttt{new Zombie}(9,5)\}\rangle$$

Sup-interpretation adaptation

Definition

Given a program **p** of main class having n attributes $X_1, \ldots, X_n$, a sup-interpretation is an additive assignment θ of **p** which satisfies:

1. θ is monotonic,
2. $\forall o \in \texttt{Objects}$, $\theta(o) \geq |o|$,
3. $\forall \texttt{f} \in \mathrm{dom}(\theta)$ of arity m, $\forall \sigma$ and $\forall o_1, \cdots, o_m \in \texttt{Objects}$, if $\langle X_i.\texttt{f}(o_1, \cdots, o_m), \sigma \rangle \rightarrow \langle v, \sigma' \rangle$ then

$$\theta(\texttt{f})(\theta(o_1), \ldots, \theta(o_m), \theta(X_i\sigma)) \geq \max(\theta(v), \theta(X_i\sigma'))$$

- Object as input: $\theta(X_i\sigma)$
- Computed value: $\theta(v)$
- Side effect: $\theta(X_i\sigma')$

Example of polynomial and additive sup-interpretations

Example (Translation)

$$\theta(n) = |n| \qquad \theta(\texttt{Zombie})(X, Y) = X + Y + 1$$
$$\theta(\texttt{add})(X, Y) = X + Y \qquad \theta(\texttt{move})(X, Y, Z) = X + Y + Z$$

1. These functions are monotonic.
2. $\forall o \in \texttt{Objects}, \theta(o) \geq |o|$
3. If $Z\sigma = \texttt{Zombie}(u, v)$ then $\langle Z.\texttt{move}(n, m), \sigma\rangle \rightarrow \langle u + n, \sigma'\rangle$ with $\sigma' = \sigma\{Z \leftarrow \texttt{Zombie}(u + n, v + m)\}$

$$\begin{aligned}
&\theta(\texttt{move})(\theta(n), \theta(m), \theta(\texttt{Zombie}(u, v))) \\
&= \theta(\texttt{move})(|n|, |m|, |u| + |v| + 1) \\
&= |n| + |m| + |u| + |v| + 1 \\
&\geq \max(\theta(u + n), \theta(\texttt{Zombie}(u + n, v + m)))
\end{aligned}$$

Results on object oriented Sup-interpretations

- Brotherly criterion (Marion and Péchoux[2007])

Fundamental Lemma

The Fundamental Lemma holds for brotherly programs.

  - Criterion for method sup-interpretation synthesis
  - Restrictions on the Brotherly criterion give characterizations of FPTIME and $\mathrm{FPSPACE}$.

- Complementary approaches using abstract interpretations by Miné[2006], Jung et al.[2003]
- Future work:
  - Adapt the brotherly criterion to recursive methods
  - Adapt the framework to superclass, casting...

Conclusion

- Everything can be interpreted:
  - Polynomial interpretations for program termination
  - Abstract interpretations Cousot et al.[1977] for runtime errors
  - Quasi-interpretation for program complexity
  - Sup-interpretations for a better intensionality
- A lot of work remains:
  - Develop CROCUS and the synthesis
  - Extension to other programming paradigms
  - Compare/Combine the QI/SI with other ICC techniques
  - Compare the QI/SI with the abstract interpretations