A characterization of Alternating log time by first order functional programs

Guillaume Bonfante, Jean-Yves Marion and Romain Péchoux

CARTE-LORIA-INPL

LPAR 2006

Motivations

- Implicit Computational Complexity
- Resource control, resource certificates
- Static analysis of first order functional programs
- We are interested in smaller and parallel complexity classes

Increase the intensionality

- Machine independent characterizations à la Cobham[1965]
- Quasi-interpretations by Bonfante, Marion and Moyen [2000-2005] of PTIME, PSPACE, LOGSPACE which consist in:
  - an assignment being:
    - ★ subterm
    - ★ weakly monotonic
- The notion of sup-interpretation by Marion and Péchoux [2006] distinct from the one of quasi-interpretation:
  - It ranges over a part of the set of function symbols
  - A relaxed subterm property

Characterizations of NC^1

- Machine level:
  - Ruzzo[1981]
  - Barrington[1987]
  - Immerman, Barrington, Straubing[1990]
- Machine independent characterizations:
  - Compton and Laflamme[1987]
  - Clote[1990]
  - Bloch[1994],
  - Leivant-Marion[1999]

First order Language

Syntax

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

$$
\begin{array}{lllll}
\texttt{(Terms)} & \ni t & ::= & x \mid \mathbf{c}(t_1, \cdots, t_n) \mid \mathbf{op}(t_1, \cdots, t_n) \\
 & & & \mid \mathtt{f}(t_1, \cdots, t_n) \\
\texttt{(Patterns)} & \ni p & ::= & \mathbf{c}(p_1, \cdots, p_n) \mid x \\
\texttt{(Rules)} & \ni r & ::= & \mathtt{f}(p_1, \cdots, p_n) \longrightarrow t \\
\texttt{(Programs)} & \ni p & ::= & r_1, \ldots, r_n
\end{array}
$$

First order Language

semantics

- Call-by-value evaluation:

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

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

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

Example

Example (Logarithm)

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

$$[\![\mathtt{half}]\!](\underline{n}) = \underline{\lfloor n/2 \rfloor}$$

$$\log(\mathbf{S}(\mathbf{S}(n))) \to \mathbf{S}(\log(\mathbf{S}(\mathtt{half}(n))))$$

$$\log(\mathbf{S}(\mathbf{0})) \to \mathbf{0}$$

$$\log(\mathbf{0}) \to \mathbf{0}$$

$$[\![\log]\!](\underline{n}) = \underline{\lfloor \log_2(n) \rfloor}, \ n \neq \mathbf{0}$$

Precedence $\geq_{Fct}$

Definition

- $\mathtt{f} \geq_{Fct} \mathtt{g}$ if $\mathtt{f}(\vec{p})$ calls $\mathtt{g}(\vec{t}) >$ in some rule.
- $\mathtt{f} \approx_{Fct} \mathtt{g}$ if $\mathtt{f}$ calls $\mathtt{g}$ and vice versa.
- $\mathtt{f} >_{Fct} \mathtt{g}$ if $\mathtt{f} \geq_{Fct} \mathtt{g}$ and not $\mathtt{f} \approx_{Fct} \mathtt{g}$

- Precedence used for putting restrictions on the considered recursive calls.

Example

If `half` is a function symbol:

- $\log >_{Fct} \mathtt{half}$

Fraternities

Definition (Fraternity)

A rule of the shape

$$\mathtt{f}(\overline{p}) \to \mathrm{C}[\mathrm{g}_1(\overline{e_1}), \ldots, \mathrm{g}_m(\overline{e_m})]$$

is a fraternity if:

- $\forall\ i \in \{1, m\},\ \mathrm{g}_i \approx_{Fct} \mathtt{f}$
- $\forall\ \mathtt{h} \in \mathrm{C}[-],\ \mathtt{f} >_{Fct} \mathtt{h}$

Example

$$\log(\mathbf{S}(\mathbf{S}(n))) \to \mathbf{S}(\log(\mathbf{S}(\mathtt{half}(n))))$$

- $\log \approx_{Fct} \log$ and $\log >_{Fct} \mathtt{half}$
- $\mathrm{C}[-] = \mathbf{S}(-)$

Assignment

Definition (Assignment)

- A partial assignment θ is a mapping from the symbols which assigns to each symbol b of the domain with arity n a function $\theta(b) \ : \ (\mathbb{R}^+)^n \to \mathbb{R}^+$
- For x $\in \mathcal{X}$, we take $\theta(x) = X$
- We extend this assignment to terms by θ^*:

$$\theta^*(\mathtt{f}(\overline{v})) = \theta(\mathtt{f})(\theta^*(\overline{v}))$$

  - $\theta^*(\overline{e}) = \max(\theta^*(e_1), \ldots, \theta^*(e_n))$

Additive Assignment

Definition (Additive Assignment)

- A partial assignment θ is additive if for each constructor symbol $\mathbf{c}$ of arity > 0:

$$\theta(\mathbf{c})(X_1, \cdots, X_n) = \sum_{j=1}^{n} X_j + \alpha_{\mathbf{c}}, \ \alpha_{\mathbf{c}} \geq 1$$

Definition (Size)

Define $|t|$ to be the size of a term t.

Lemma

Given an additive assignment θ, there is a constant α s.t. for every $v \in \mathcal{V}$:

$$|v| \leq \theta^*(v) \leq \alpha|v|$$

Sup-interpretation

Definition (Sup-interpretation)

- A sup-interpretation is a partial additive assignment which verifies:

  1. $\theta(b)(X_1, \cdots, X_n)$ is weakly monotonic
  2. For each $v \in \mathcal{V}$, $\theta^*(v) \geq |v|$
  3. For each symbol b and each $v_1, \cdots, v_n \in \mathcal{V}$, $\theta^*(b(v_1, \cdots, v_n)) \geq \theta^*([\![b]\!](v_1, \cdots, v_n))$

Definition (Polynomial Sup-interpretation)

A sup-interpretation θ is a polynomial if:

- θ is additive.
- for each symbol b, $\theta(b)$ is bounded by some polynomial.

Examples of polynomial sup-interpretations

Example (Multiplication)

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

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

$$[\![\mathtt{mult}(x, y)]\!](\underline{n}, \underline{m}) = \underline{n \times m}$$

Examples of polynomial sup-interpretations

Example (Sup-interpretation)

- $\theta(\mathbf{0}) = 0$, $\theta(\mathbf{S})(X) = X + 1$
- $\theta(\mathtt{mult})(X, Y) = X \times Y$

1. Weakly monotonic
2. $\theta^*(\mathbf{S}^n(\mathbf{0})) \geq |\mathbf{S}^n(\mathbf{0})| = n$
3.

$$\begin{aligned}
\theta^*(\mathtt{mult}(\mathbf{S}^n(\mathbf{0}), \mathbf{S}^m(\mathbf{0})) &= \theta(\mathtt{mult})(\theta^*(\mathbf{S}^n(\mathbf{0})), \theta^*(\mathbf{S}^m(\mathbf{0}))) \\
&= \theta^*(\mathbf{S}^n(\mathbf{0})) \times \theta^*(\mathbf{S}^m(\mathbf{0})) \\
&= n \times m \\
&= \theta^*(\mathbf{S}^{n \times m}(\mathbf{0})) \\
&= \theta^*([\![\mathtt{mult}]\!](\mathbf{S}^n(\mathbf{0}), \mathbf{S}^m(\mathbf{0})))
\end{aligned}$$

Examples of polynomial sup-interpretations

Example (Logarithm)

$$[\![\mathtt{half}]\!](\underline{n}) = \underline{\lfloor n/2 \rfloor}$$

$$\log(\mathbf{S}(\mathbf{S}(n))) \to \mathbf{S}(\log(\mathbf{S}(\mathtt{half}(n))))$$

$$\log(\mathbf{S}(\mathbf{0})) \to \mathbf{0}$$

$$\log(\mathbf{0}) \to \mathbf{0}$$

Example (Sup-interpretation of logarithm)

- $\theta(\mathbf{0}) = 0$
- $\theta(\mathbf{S})(X) = X + 1$
- $\theta(\mathtt{half})(X) = \lfloor X/2 \rfloor$
- $\theta(\log)(X) = \lfloor \log_2(X) \rfloor$

Lighweight

Definition (Lighweight)

A lightweight ω is an assignment which ranges over function symbols and such that:

- For $\mathtt{f}$ function symbol, $\omega_{\mathtt{f}}$ is weakly monotonic

Example

$\lfloor X/2 \rfloor$

- We combine weights and sup-interpretations:
  If $\omega_{\mathtt{f}} = \max$, then $\omega_{\mathtt{f}}(\theta(x), \theta(y)) = \max(\theta(x), \theta(y))$
- we also consider polynomial lightweight.

Arboreal program

Definition (Arboreal program)

A program **p** admits an *arboreal* sup-interpretation iff there are

- a polynomial sup-interpretation θ,
- a polynomial lightweight ω
- and a constant $K > 1$

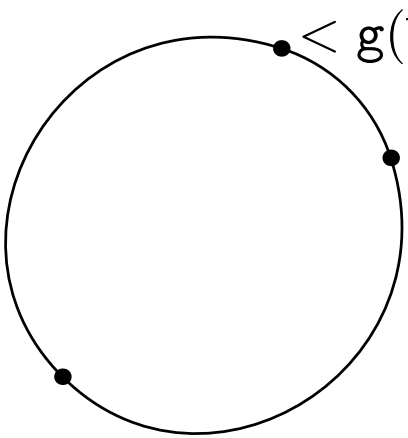
such that

- for each fraternity $\mathtt{f}(p_1, \cdots, p_n) \to \mathsf{C}[\mathsf{g}_1(\overline{t_1}), \ldots, \mathsf{g}_r(\overline{t_r})]$,
- and any substitutions σ,

$$\omega_{\mathtt{f}}(\theta^*(p_1\sigma), \ldots, \theta^*(p_n\sigma)) > 1$$
$$\omega_{\mathtt{f}}(\theta^*(p_1\sigma), \ldots, \theta^*(p_n\sigma)) \geq K \times \omega_{\mathsf{g}_i}(\theta^*(t_{i,1}\sigma), \ldots, \theta^*(t_{i,m}\sigma)) \quad \forall 1 \leq i \leq r$$

Arboreal programs

Fraternities correspond to cycles in the dependency pairs graph:



$$< \mathtt{g}(\overline{q}), \mathtt{h}(\overline{d}) >$$

$$< \mathtt{f}(p_1, \cdots, p_n), \mathtt{g}_i(\overline{t_i}) >$$

$$\omega_{\mathtt{f}}(\theta^*(p_1\sigma), \ldots, \theta^*(p_n\sigma)) \geq K \times \omega_{\mathtt{g}_i}(\theta^*(t_{i,1}\sigma), \ldots, \theta^*(t_{i,m}\sigma))$$

with $\mathtt{f} \approx_{Fct} \mathtt{g} \approx_{Fct} \mathtt{h} \approx_{Fct} \mathtt{g}_i$.
At most $\log(n)$ recursive calls in depth, with n the inputs size.

Logarithm is arboreal

Fraternity

$$\log(\mathbf{S}(\mathbf{S}(n))) \to \mathbf{S}(\log(\mathbf{S}(\mathtt{half}(n))))$$

$$\omega_{\log}(X) = X,\ \theta(\mathbf{S})(X) = X + 1,\ \theta(\mathtt{half})(X) = X/2$$

Condition 1

$$\begin{aligned}
\omega_{\log}(\theta^*(\mathbf{S}(\mathbf{S}(n)))) &= N + 2 \\
&\geq K \times (3 + N)/2 && K = 4/3 > 1 \\
&= K \times (1 + (N + 1)/2) \\
&= K \times \theta(\mathbf{S})((N + 1)/2) \\
&= K \times \theta(\mathbf{S})(\omega_{\log}((N + 1)/2)) \\
&= K \times \theta(\mathbf{S})(\omega_{\log}(\theta^*(\mathbf{S}(\mathtt{half}(n)))))
\end{aligned}$$

Exponential is not arboreal

Fraternities

$$[\![\mathtt{double}]\!](\underline{n}) = \underline{2 \times n}$$
$$\mathtt{exp}(\mathbf{S}(x)) \to \mathtt{double}(\mathtt{exp}(x))$$

$$\theta(\mathbf{S})(X) = X + 1,\ \omega_{\mathtt{exp}}(X, Y) = ?,\ \text{and } K > 1$$

Condition 2

$$\omega_{\mathtt{exp}}(\theta^*(\mathbf{S}(x))) = \omega_{\mathtt{exp}}(X + 1)$$
$$\geq K \times \omega_{\mathtt{exp}}(X)$$

Flat operators and Explicitly defined symbols

Definition (Flat operators)

A flat operator **op** is satisfies:

- U_{E^*}-uniform, polynomial size and constant depth circuits family
- $\theta(\mathbf{op})(X_1, \cdots, X_n) \leq \sum_{i=1}^{n} \alpha_i \times X_i + \alpha$

Definition (Explicitly defined, Explicitly fraternal)

A function symbol $\mathtt{f}$ is *explicitly defined* iff for each rule $\mathtt{f}(p_1, \cdots, p_n) \longrightarrow e$, the expression e is built from:

- variables, constructors, operators
- and explicitly defined function symbols g s.t. $\mathtt{f} >_{Fct} \mathtt{g}$.

A program is *explicitly fraternal* iff for each fraternity $\mathtt{f}(\overline{p}) \longrightarrow \mathtt{C}[\mathtt{g}_1(\overline{t_1}), \ldots, \mathtt{g}_r(\overline{t_r})]$, $\mathtt{C}[\diamond_1, \cdots, \diamond_r]$ and $\overline{t_i}$ are *explicitly defined*.

Alogtime

Theorem

A function ϕ over $\mathcal{V}$ is computed by an explicitly fraternal and arboreal program having flat operators if and only if ϕ is computed in ALogTime.

Soundness: $\subseteq$.

We first show that computed value is polynomially bounded:

- Arboreal programs + Polynomial sup-interpretations: at most $\log(n)$ depth recursive calls for inputs of size n.
- explicitly fraternal programs: for each fraternity $\mathtt{f}(\overline{p}) \to \mathrm{C}[\mathrm{g}_1(\overline{t_1}), \ldots, \mathrm{g}_r(\overline{t_r})]$, the context $\mathrm{C}[\diamond_1, \cdots, \diamond_r]$ and each $\overline{t_i}$ are bounded by $a \times m + l$, with m the size of the inputs.
- The composition of log affine functions remains polynomial

Alogtime

Theorem

A function ϕ over $\mathcal{V}$ is computed by an explicitly fraternal and arboreal program having flat operators if and only if ϕ is computed in ALogTime.

Proof.

We obtain a log-depth and polynomial size circuit

- arboreal programs + Polynomial sup-interpretations: at most $\log(n)$ depth recursive calls for inputs of size n.
- explicitly fraternal programs and flat operators: for each fraternity $\mathtt{f}(\overline{p}) \longrightarrow \mathtt{C}[\mathtt{g}_1(\overline{t_1}), \ldots, \mathtt{g}_r(\overline{t_r})]$, the context $\mathtt{C}[\diamond_1, \cdots, \diamond_r]$ and each $\overline{t_i}$ are NC^0 computable.
- We obtain log compositions of constant depth circuits

Completeness: $\supseteq$

Proof.

- $\supseteq$ uses a characterization LRRS of Leivant and Marion [1999] by linear ramified recursion over well-balanced trees:

$$\mathtt{f}(\mathbf{c}, \overline{u}; \overline{x}) = \mathtt{g}_{\mathbf{c}}(\overline{u}; \overline{x}) \qquad \mathbf{c} = \mathbf{0}, 1, \bot$$
$$\mathtt{f}(t * t', \overline{u}; \overline{x}) = \mathtt{g}(; \mathtt{f}(t, \overline{u}; \mathtt{h}_1(; \overline{x})), \ldots, \mathtt{f}(t', \overline{u}; \mathtt{h}_k(; \overline{x})), \overline{x})$$

- The fraternities are explicitly fraternal
- Taking $\theta^*(t * t') = 1 + 2 \times \theta^*(t)$ and $\omega_{\mathtt{f}}(X_1, \cdots, X_n) = X_1$, the program is arboreal:

$$\begin{aligned} \omega_{\mathtt{f}}(\theta^*(t * t'), \theta^*(\overline{u}), \theta^*(\overline{x})) &= 2 \times \theta^*(t) + 1 \\ &\geq K \times \theta^*(t) \qquad \text{with K=2} \\ &= \omega_{\mathtt{f}}(\theta^*(t), \theta^*(\overline{u}), \theta^*(\mathtt{h}_j(; \overline{x}))) \end{aligned}$$

Conclusion

- Resource certificates for programs in *ALogTime*.
- The analysis can be performed automatically.
- Strongest intensionality: capturing more natural algorithms.