A characterization of NC^k by first order functional programs

Jean-Yves Marion and Romain Péchoux

CARTE-LORIA-INPL

TAMC 2008

Historical background

Characterizations of NC^1 and NC^k:

- Machine level:
  - Ruzzo[1981] (NC^k)
  - Barrington[1987] (NC^1)
  - Immerman, Barrington, Straubing[1990] (NC^k)
- Machine independent characterizations (à la Cobham[1965]):
  - Compton and Laflamme[1987] (NC^1)
  - Clote[1990] (NC^k)
  - Bloch[1994] (NC^1)
  - Leivant-Marion[1999] (NC^1)

We need more intensionality and an alternative to Clote[1990].

Increase the intensionality

- Resource certificates by static analysis of FO functional programs
- Quasi-interpretation by Bonfante, Marion and Moyen [2000-2005]:
  - PTIME [2000]
  - PSPACE [2001]
  - LOGSPACE [2005]

  Quasi-interpretation = a total assignment with **subterm property** ($\mathtt{f}(X) \geq X$)
- The notion of sup-interpretation by Marion and Péchoux [2006]:
  - A generalization of quasi-interpretation:
    - ⋆ A relaxed subterm property
    - ⋆ partial assignment
  - NC^1 Bonfante, Marion and Péchoux [2006]

First Order functional 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 functional language

Call-by-value semantics

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

$$\frac{[\![e_i]\!] = v_i \;\; [\![\mathbf{op}]\!](v_1, ..., v_n) = v}{[\![\mathbf{op}(e_1, ..., e_n)]\!] = v}$$

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

- The set: $\mathcal{V} \ni v ::= \mathbf{c} \mid \mathbf{c}(v_1, \cdots, v_n)\ \mathbf{c} \in \mathcal{C}$
- $[\![\mathbf{op}]\!] : \mathcal{V}^n \to \mathcal{V}$
- Linear and disjoint patterns: programs are confluent.

Example: Unary Logarithm

Example ($\mathcal{C} = \{\mathbf{0}, \mathbf{S}\}$, $Op = \{\mathtt{half}\}$ and $\mathcal{F} = \{\log\}$)

$$\log(\mathbf{0}) \to \mathbf{0}$$
$$\log(\mathbf{S}(\mathbf{0})) \to \mathbf{0}$$
$$\log(\mathbf{S}(\mathbf{S}(n))) \to \mathbf{S}(\log(\mathbf{S}(\mathtt{half}(n))))$$

$$\text{If } [\![\mathtt{half}]\!](\underline{n}) = \underline{\lfloor n/2 \rfloor} \text{ with } \underline{n} = s^n(0)$$
$$\text{then } [\![\log(\underline{n})]\!] = \underline{\lfloor \log_2(n) \rfloor},\ n \neq \mathbf{0}$$

Assignment

Definition

- A partial assignment θ is a partial mapping from symbols of the vocabulary to functions over reals:

$$\theta(b) \ : \ (\mathbb{R}^+)^n \to \mathbb{R}^+$$

- For each x $\in \mathcal{X}$, $\theta(x) = X$ with X a fresh variable.

- We extend this assignment to terms by θ^*:

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

Example

The function θ defined over `half` by $\theta(\mathtt{half})(X) = X/2$ is an assignment.

Additive and Polynomial Assignment

Definition (Additive Assignment)

A partial assignment θ is additive if for each constructor symbol $\mathbf{c}$:

$$\theta(\mathbf{c})(X_1, \cdots, X_n) = \sum_{j=1}^{n} X_j + \alpha_{\mathbf{c}},\ \alpha_{\mathbf{c}} \geq 1 \qquad \text{if } n > 0$$

$$\theta(\mathbf{c}) = 0 \qquad \text{Otherwise}$$

Definition (Polynomial Assignment)

An assignment θ is polynomial if for each symbol b (of the domain) $\theta(b)$ is a max-polynomial.

Example (Additive and Polynomial Assignment)

The function θ defined by $\theta(\mathbf{0}) = 0$, $\theta(\mathbf{S})(X) = X + 1$ and $\theta(\texttt{half})(X) = X/2$.

Sup-interpretation

Definition

- A sup-interpretation is a partial assignment which satisfies:
  1. $\theta(b)$ is weakly monotonic

$$X_i \geq Y_i \Rightarrow \theta(b)(\ldots, X_i, \ldots) \geq \theta(b)(\ldots, Y_i, \ldots)$$

  2. For each $v \in \mathcal{V}$, $\theta^*(v) \geq |v|$
  3. For each symbol b and each $v_1, \cdots, v_n \in \mathcal{V}$,

$$\theta^*(b(v_1, \cdots, v_n)) \geq \theta^*([\![b(v_1, \cdots, v_n)]\!])$$

Lemma (Additive assignments)

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

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

Examples of polynomial sup-interpretation

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)

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

Examples of polynomial sup-interpretation

Example (Sup-interpretation of logarithm)

$\theta(\mathbf{0}) = 0$, $\theta(\mathbf{S})(X) = X + 1$, $\theta(\mathtt{half})(X) = X/2$ and $\theta(\log)(X) = X$

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

$$\begin{aligned}
\theta^*(\log(\mathbf{S}^n(\mathbf{0})) &= \theta(\log)(\theta^*(\mathbf{S}^n(\mathbf{0}))) \\
&= \theta^*(\mathbf{S}^n(\mathbf{0})) \\
&= n \\
&\geq \lfloor \log_2(n) \rfloor \\
&= \theta^*(\mathbf{S}^{\lfloor \log_2(n) \rfloor}(\mathbf{0})) \\
&= \theta^*([\![\log(\mathbf{S}^n(\mathbf{0}))]\!])
\end{aligned}$$

Precedence $\geq_{Fct}$

Definition

- $\mathtt{f} \geq_{Fct} \mathtt{g}$ if there is a rule $\mathtt{f}(\vec{p}) \to C[\mathtt{g}(\vec{t})] \in \mathtt{Rules}$.
- $\mathtt{f} \approx_{Fct} \mathtt{g}$ 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 $\mathtt{f} \approx_{Fct} \mathtt{g}$

Example

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

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

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

Arboreal program

Definition

A program **p** is *arboreal* if there are:

- an additive and polynomial sup-interpretation θ,
- a polynomial, monotonic and partial assignment ω,
- and a constant $K > 1$

s.t. for each fraternity $\mathtt{f}(p_1, \cdots, p_n) \longrightarrow \mathsf{C}[\mathsf{g}_1(\overline{t_1}), \ldots, \mathsf{g}_r(\overline{t_r})]$ and $\forall \sigma$:

1. $\omega_{\mathtt{f}}(\theta^*(p_1\sigma), \ldots, \theta^*(p_n\sigma)) \geq 1$
2. $\omega_{\mathtt{f}}(\theta^*(p_1\sigma), \ldots, \theta^*(p_n\sigma)) \geq K \times \omega_{\mathsf{g}_i}(\theta(\overline{t_i}\sigma))$

Example of arboreal program

Example (Logarithm is arboreal)

$$\omega_{\log}(X) = X,\ \theta(\mathbf{0}) = 0,\ \theta(\mathbf{S})(X) = X + 1,\ \theta(\mathtt{half})(X) = X/2$$

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

$$\begin{aligned}
\omega_{\log}(\theta(\mathbf{S}(\mathbf{S}(n)))) &= N + 2 \geq 1 && (1)\\
&\geq 2 \times (N/2 + 1)\\
&= 2 \times \omega_{\log}(N/2 + 1)\\
&= 2 \times \omega_{\log}(\theta(\mathbf{S}(\mathtt{half}(n)))) && (2) \text{ with } K = 2
\end{aligned}$$

Exponential is not arboreal

Example (Exponential)

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

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

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

Fraternal programs

Definition

A program **p** is *fraternal* if there is a polynomial and additive sup-interpretation θ s.t.:

- for each fraternity of the shape $C[g_1(\overline{e_1}), \ldots, g_r(\overline{e_r})]$
- and for each symbol b appearing in C or in $\overline{e_j}$

$$\theta(b)(X_1, \cdots, X_n) = \sum_{i=1}^{n} \alpha_i \times X_i + \beta, \text{for some constants } \alpha_i, \beta$$

Example (Logarithm is fraternal)

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

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

Characterization of NC^k

Definition (Rank)

- Rank of a variable or a constructor symbol: 0
- Rank of an operator simulated by a NC^k circuits family: k
- Rank of a function $\mathtt{f}(\overline{p}) \longrightarrow \mathtt{g}(\mathtt{f}(x))$: $k+1$ if rank of g is k.

Theorem

A function ϕ over $\mathcal{V}$ is computed by a

- *fraternal*
- *and arboreal program*
- *of rank k*

if and only if ϕ is computed in NC^k.

Sketch of the proof

Completeness: $\supseteq$.

- We simulate the Clote's schemas by arboreal and fraternal programs (CRN and WBRN)

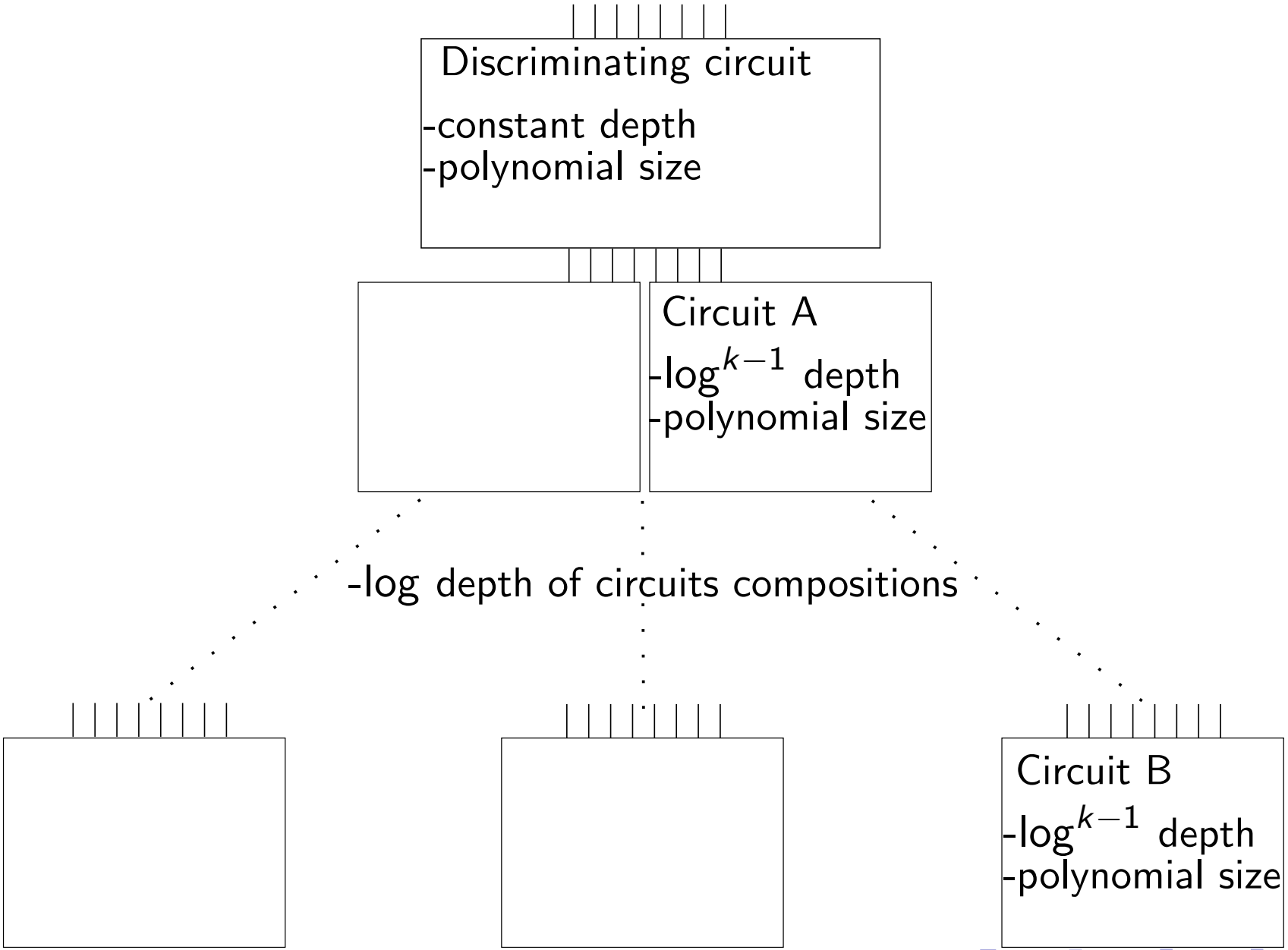
□

Soundness: $\subseteq$.

- Polynomial computations:
  - Arboreal programs + Polynomial sup-interpretations: at most $\log(n)$ depth recursive calls.
  - Fraternal programs: for each fraternity $\mathtt{f}(\overline{p}) \longrightarrow \mathrm{C}[\mathrm{g}_1(\overline{t_1}), \ldots, \mathrm{g}_r(\overline{t_r})]$, $\mathrm{C}[\diamond_1, \cdots, \diamond_r]$ and each $\overline{t_i}$ computations are bounded by $a \times n + l$.
- We simulate the program by a uniform family of circuits.

□

Sketch of the proof



Conclusion

- Resource certificates for programs in NC^k.
- The analysis can be performed automatically:
  - Heuristics for polynomial (bounded degree) sup-interpretations.
- Strongest intensionality: capturing more natural algorithms than:
  - Clote for NC^k
  - Bloch and Leivant and Marion for NC^1.