Algorithmic Completeness of Imperative Languages

Yoann Marquer

LACL, University Paris-Est Créteil

June 16 2014

Introduction

1) Palindrome recognition

- one tape : $O(n^2/log(n))$
- two tapes : $O(n)$

Equivalence between models : **extensionally** $\neq$ **intentionally**

2) Lack of efficient *PR* algorithms :

- *min* (1991 : Colson)
- *gcd* (2003 : Moschovakis)

Completeness of a model : **functionally** $\neq$ **algorithmically**

3) Formalization of sequential algorithms

- Recursors (Moschovakis)
- Abstract State Machines (Gurevich)

Contents

Definition (Gurevich's Algo)

1) Sequential time : $S(A)$, $I(A) \subseteq S(A)$, $\tau_A : S(A) \rightarrow S(A)$
2) Abstract states : states are first order structures, $\mathcal{L}(A), \mathcal{U}(A)$
3) Bounded exploration : the number of terms read by A is finite.

An **execution** of A is $\vec{X} = X_0, X_1, X_2, \ldots$ such that
- $X_0 \in I(A)$
- for all $i \in \mathbb{N} : X_{i+1} = \tau_A(X_i)$

Terminal execution : $X_0, X_1, \ldots X_m, X_m, \ldots$
$time(A, X) = min\{i \in \mathbb{N}^\star \, ; \tau_A^i(X) = \tau_A^{i-1}(X)\}$

Contents

Definition (ASM)

$$\Pi =_{def} ft_1 \ldots t_k := t_0$$
$$\mid \texttt{if } F \texttt{ then } \Pi_1 \texttt{ else } \Pi_2 \texttt{ endif}$$
$$\mid \texttt{par } \Pi_1 \| \ldots \| \Pi_n \texttt{ endpar}$$

$\tau_\Pi(X) = X + \Delta(\Pi, X)$, where :

- $\Delta(ft_1 \ldots t_k := t_0, X) = \{(f, \overline{t_1}^X, \ldots \overline{t_k}^X, \overline{t_0}^X)\}$
- $\Delta(\texttt{if } F \texttt{ then } \Pi_1 \texttt{ else } \Pi_2 \texttt{ endif}, X) = \Delta(\Pi_i, X)$
  where $i = 1$ if $\overline{F}^X = true$ and $i = 2$ if $\overline{F}^X = false$
- $\Delta(\texttt{par } \Pi_1 \| \ldots \| \Pi_n \texttt{ endpar}, X) = \Delta(\Pi_1, X) \cup \ldots \cup \Delta(\Pi_n, X)$

Theorem (Gurevich, 2000)

$Algo = ASM$

Contents

Temporary Variables ?

`loop` n $\{s\}$

$i := n$; `while` $i > 0$ $\{s;\ i := i - 1\}$

Arbitrary Time Unit ?

$X_0, X_1, X_2, X_3, X_4, X_5, X_6, \ldots$

$Y_0, \qquad Y_1, \qquad Y_2, \ldots$

Definition (M_1 simulates M_2)

For all $P_2 \in M_2$ there exists $P_1 \in M_1$ and $d \in \mathbb{N}^\star$ such that :

1) $\mathcal{L}(P_1) \supseteq \mathcal{L}(P_2)$ and $\mathcal{L}(P_1) \setminus \mathcal{L}(P_2)$ is a finite set of variables

2) for all execution $\vec{Y}$ of P_2 there exists an execution $\vec{X}$ of P_1 :

- for all $i \in \mathbb{N}$ $X_{d\times i}|_{\mathcal{L}(P_2)} = Y_i$

- $d \times time(P_1, X_0) = time(P_2, Y_0)$

If bisimulation $M_1 \simeq M_2$: they are **algorithmically** *equivalent.*

Contents

Definition (While)

(commands) $c ::= ft_1 \dots t_k := t_0 \mid \texttt{while } F \ \{s\}$
(sequences) $s ::= \epsilon \mid c; s$
(programs) $P ::= \{s\}$

$$\{ft_1 \dots t_k := t_0; s\} \star X \succ \{s\} \star X + \{(f, \overline{t_1}^X, \dots \overline{t_k}^X, \overline{t_0}^X)\}$$

$$\{\texttt{while } F \ \{s_1\}; s_2\} \star X \succ \{s_1; \texttt{while } F \ \{s_1\}; s_2\} \star X \text{ if } \overline{F}^X = true$$

$$\{\texttt{while } F \ \{s_1\}; s_2\} \star X \succ \{s_2\} \star X \text{ if } \overline{F}^X = false$$

`skip` and `if` can be simulated.

$P \star X \succ_i \tau_X^i(P) \star \tau_P^i(X)$
$time(P, X) =_{def} min\{i \in \mathbb{N} \ ; \ \tau_X^i(P) = \{\} \ \}$
If finite : P terminates on X, and $P(X) =_{def} \tau_P^{time(P,X)}(X)$

Definition (Updates of an Imperative Program)

$\Delta(P, X) =_{def} \bigcup_{i \in \mathbb{N}} \tau_P^{i+1}(X) - \tau_P^i(X)$

Contents

Theorem (2014, M.)

While $\simeq$ *Algo*

Sketch of the proof :
An imperative program can be simulated, with $d = 1$
and an instruction counter ℓ initialized at 0 :

Example : a program for min

P is $\{x := 0;\ \texttt{while}\ \neg(x = m \vee x = n)\ \{x := x + 1;\};\}$

$$\begin{aligned}
\Pi_P \text{ is } & \texttt{if } \ell = 0 \texttt{ then } x := 0 \| \ell := 1 \\
& \| \texttt{if } \ell = 1 \texttt{ then } \texttt{if } \neg(x = m \vee x = n) \\
& \qquad\qquad\qquad \texttt{then } x := x + 1 \| \ell := 1 \\
& \qquad\qquad\qquad \texttt{else } \ell := 2
\end{aligned}$$

Contents

Definition (Naive Translation Π^{tr} of an ASM program Π)

- $(ft_1 \ldots t_k := t_0)^{tr}$ *is* $\{ft_1 \ldots t_k := t_0;\}$
- $(\texttt{if } F \texttt{ then } \Pi_1 \texttt{ else } \Pi_2 \texttt{ endif})^{tr}$ *is* $\{\texttt{if } F\ \Pi_1^{tr} \texttt{ else } \Pi_2^{tr};\}$
- $(\texttt{par } \Pi_1 \| \ldots \| \Pi_k \texttt{ endpar})^{tr}$ *is* $\Pi_1^{tr} \ldots \Pi_k^{tr}$ *(composition)*

Example : $(\texttt{par } x := y \| y := x \texttt{ endpar})^{tr}$ is $\{x := y; y := x;\}$

Proposition (Correct Semantics for the Translation)

Let $\{\vec{t}\}$ *be the terms read by* Π *and* $\vec{v}$ *be fresh variables.*

$\Delta(\Pi^{tr}[\vec{v}/\vec{t}], X + \{(\vec{v}, \vec{t}^X)\}) = \Delta(\Pi, X|_{\mathcal{L}(\Pi)})$

Let P_Π be $\vec{v} := \vec{t}; \Pi^{tr}[\vec{v}/\vec{t}]$ (+ padding with `skip`)

Proposition

For all X $\Delta(P_\Pi, X)|_{\mathcal{L}(\Pi)} = \Delta(\Pi, X|_{\mathcal{L}(\Pi)})$
There exists $t_\Pi \in \mathbb{N}$ *such that for all X* $time(P_\Pi, X) = t_\Pi$
For all X $time(\Pi, X) = min\{i \in \mathbb{N} \; ; \; \overline{F_\Pi}^{P_\Pi^i(X)} = true\}$
where F_Π *is* $\bigwedge \vec{v} = \vec{t}$

P_Π `while` $\neg F_\Pi$ $\{P_\Pi\}$ simulates Π, with :
- temporal dilatation : $d = t_\Pi + 1$
- temporary variables : $\vec{v}$

Conclusion and Future Work

Theorem (2014, M.)

While $\simeq$ *Algo*

Same states (first order structures)
Faithful implementation of the usual data structures ?
Restrictions on programs or data structures for subclasses of Algo ?

Thank you !

Operational semantics of the imperative programs :

$$\{\texttt{skip}; s\} \star X \succ \{s\} \star X$$
$$\{ft_1 \dots t_k := t_0; s\} \star X \succ \{s\} \star X[f(\overline{t_1}^X, \dots \overline{t_k}^X) = \overline{t_0}^X)]$$
$$\{\texttt{if } F \ \{s_1\} \texttt{ else } \{s_2\}; s_3\} \star X \succ \{s_1; s_3\} \star X \text{ if } \overline{F}^X = \mathit{true}$$
$$\{\texttt{if } F \ \{s_1\} \texttt{ else } \{s_2\}; s_3\} \star X \succ \{s_2; s_3\} \star X \text{ if } \overline{F}^X = \mathit{false}$$
$$\{\texttt{while } F \ \{s_1\}; s_2\} \star X \succ \{s_1; \texttt{while } F \ \{s_1\}; s_2\} \star X \text{ if } \overline{F}^X = \mathit{true}$$
$$\{\texttt{while } F \ \{s_1\}; s_2\} \star X \succ \{s_2\} \star X \text{ if } \overline{F}^X = \mathit{false}$$
$$\{\texttt{loop } n \ \{s_1\}; s_2\} \star X \succ \{s_1; \texttt{loop } S^{\overline{n}^X - 1} 0 \ \{s_1\}; s_2\} \star X \text{ if } \overline{n}^X > 0$$
$$\{\texttt{loop } n \ \{s_1\}; s_2\} \star X \succ \{s_2\} \star X \text{ if } \overline{n}^X = 0$$
$$\{\texttt{exit}; s\} \star X \succ \{\} \star X$$

Loop$_e$ is *Imp* restricted to updates, `if`, `loop` and `exit` commands.
is from APRA (2010 : Andary, Patrou, Valarcher)
Let $|a|$ be the size of $A \in \mathcal{U}(A)$.

Definition ($Algo_{PR} = \{A \in Algo \; ; \; c_A \in PR\}$)

$c_A : \vec{n} \mapsto max\{time(A, X) \; ; \; \vec{s} \text{ inputs of } A \text{ and } |\vec{s}|_X = \vec{n}\}$
$where \; |x|_X = |\overline{x}^X| \; and \; |f|_X = max\{|\overline{f}^X(\vec{t}^X)| \; ; \; f\,\vec{t} \in sub(T(A))\}$

But $n := ackermann(n, n)$; `loop` n $\{\}$?
PR data structures : for all operation f
there exists $\varphi_f \in PR$ monotonic such that $|\overline{f}(\vec{a})| \leq \varphi_f(|\vec{a}|)$

Theorem (2014, M.)

For PR data structures *Loop*$_e \simeq Algo_{PR}$

Theorem (2014, M.)

For all data structures While $\simeq$ Algo
For PR data structures $Loop_e \simeq Algo_{PR}$

The control flow is known, but with the same data structures.
They are first-order structures : implementation ?

Proposition (Constructive Second Postulate)

Usual data structures (integers, words, lists, arrays) can be faithfully implemented as first order structure.

Conjecture (Characterization of Complexity)

For Pol data structures $Loop_e + C_1 \simeq Algo_{Pol}$
For Lin data structures $Loop_e + C_2 \simeq Algo_{Lin}$

(C_1) : for all $\texttt{loop} \in P$ $VarCon(\texttt{loop}) \cap VarUpd(\texttt{loop}) = \varnothing$
(C_2) : for all $\texttt{loop} \in P$ $card(VarCon(\texttt{loop})) \leq 1$

Example (Cost of Operations)

For unary integers : $+ \in Lin$, $\times \in Pol$ *and* $pow \in PR$
For binary integers : $+ \in Lin$, $\times \in Lin$ *and* $pow \in Pol$