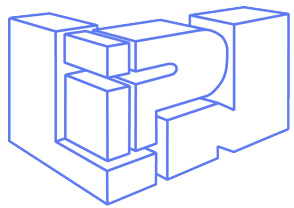
On pointers, graphs, and Logspace complexity.

9th project meeting of Complice

Clément Aubert



Institut Galilée - Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse
aubert@lipn.fr

29 novembre 2012

1. L, graphs and tree
2. Pure Pointer Programs
3. Finite Automatons
4. SMM and KUM
5. Other models

The following problems are **L**-complete [JLM97]:

- Is Tree?
- Is Forest?
- Is Line?
- Acyclicity of undirected graph.
- Undirected s-t connectivity / Undirected reachability.

The following problems are **L**-complete [JLM97]:

- Is Tree?
- Is Forest?
- Is Line?
- Acyclicity of undirected graph.
- Undirected s-t connectivity / Undirected reachability.

The following problem is **NL**-complete:

- Directed s-t connectivity / Directed reachability

Presentation

- *PURPLE = PURe Pointer LanguagE ([HS08]),*
- *Only abstract pointers, no internal structure,* `forall`*-loops, [*`while`*-loops],*
- *Take as input a locally ordered graph.*

Presentation

- *PURPLE = PURe Pointer LanguagE ([HS08]),*
- *Only abstract pointers, no internal structure,* `forall`*-loops,* *[*`while`*-loops],*
- *Take as input a locally ordered graph.*

Remark

If there is a total ordering on the nodes of the input graph, it is possible to encode any ***L*** *algorithm.*

Results

- *There is no PURPLE program that decides undirected reachability.*

But. . .

Results

- *There is no PURPLE program that decides undirected reachability.*

But. . .

- *Acyclicity of undirected graphs can be decided in PURPLE.*

Results

- *There is no PURPLE program that decides undirected reachability.*

But. . .

- *Acyclicity of undirected graphs can be decided in PURPLE.*

Links to other models of computation

- *Each* `forall-free` *program can be compiled into a JAG and vice versa.*
- *For each closed DTC*[a] *formula* ϕ *for locally ordered graphs there exists a program* M_ϕ *such that, for any finite locally ordered graph* Γ *of degree* d*,* $\Gamma \models \phi$ *iff* M_ϕ *recognizes* Γ*.*

[a] Deterministic Transitive Closure

Presentation

- *k read-only tapes, cannot write,*
- *States and transition function as usual,*
- *Can go 1- or 2-way, can be deterministic or non-deterministic.*

Presentation

- *k read-only tapes, cannot write,*
- *States and transition function as usual,*
- *Can go 1- or 2-way, can be deterministic or non-deterministic.*

Remark

$$\boldsymbol{L} = \cup_{k \geqslant 1} \mathcal{L}(2DFA(k))$$

$$\boldsymbol{NL} = \cup_{k \geqslant 1} \mathcal{L}(2NFA(k))$$

Results

Let $k \geqslant 1$

- *There are unary languages that shows* $\mathcal{L}(2DFA(k)) \subset \mathcal{L}(2DFA(k+1))$ *and the same holds for* $\mathcal{L}(2NFA(k))$.
- $\boldsymbol{NL} = \boldsymbol{L}$ *iff* $\mathcal{L}(1NFA(2)) \subseteq \cup_k \mathcal{L}(2DFA(k))$

Results

Let $k \geqslant 1$

- *There are unary languages that shows* $\mathcal{L}(2DFA(k)) \subset \mathcal{L}(2DFA(k+1))$ *and the same holds for* $\mathcal{L}(2NFA(k))$.
- $\boldsymbol{NL} = \boldsymbol{L}$ *iff* $\mathcal{L}(1NFA(2)) \subseteq \cup_k \mathcal{L}(2DFA(k))$

Links to other models of computation

- *The trade-off between (non-)deterministic two-way* $(k+1)$*-head finite automata and (non-)deterministic two-way* k*-head finite automata is non-recursive.*
- *Let* $k \geqslant 2$*, the trade-off between deterministic* $1DFA(k)$ *and non-deterministic pushdown automata is non-recursive.*

  "The gain in economy of description can be arbitrary."
  [HKM09]

Presentation (Storage Modification Machines [Sch80])

- *Data structure processed:* $\Delta = (X, a, p)$ *where*
  - X *is a finite set of nodes,*
  - $a \in X$ *is the center (the active node),*
  - $p = (p_\alpha \mid \alpha \in X)$ *is the family of pointer mappings* $p_\alpha : X \to X$

  $p_\alpha(x) = y$ *means that the pointer with label* α *originating from* x *goes to* y*. We define* $p^* : \Delta^* \to X$ *recursively:*

$$p^*(\square) = a, \text{ were } \square \text{ is the empty word}$$

$$p^*(W\alpha) = p_\alpha(p^*(W)) \text{ for all } \alpha \in \Delta, W \in \Delta^*$$

- *Finite control given as a program made of labels (as in ALGOL) and instructions (such as input, output, goto, new, set, if).*

Presentation (Kolmogorov Uspenskii Machines [KU58])

Exactly like the SMM, except that it operates on undirected graph.

Look at the board!

Look at the board!

In space n, with k symbols, a Turing machine can encode

$$O(n \log k)$$

bits of information, a SMM can encode

$$O(n\, k \log n)$$

bits of information.

Links to other models of computation

- *SMM* $\equiv$ *SRAM (Real-time)*
- *Existence of a non-linear gap between the SMM and the KUM?*

Links to other models of computation

- *SMM* $\equiv$ *SRAM (Real-time)*
- *Existence of a non-linear gap between the SMM and the KUM?*

Results

No.

Presentation (Jumping Automaton for Graphs)

- *Takes a d-maze as input,*
- *Deterministic control,*
- *Finite set of pebbles.*

The storage of a JAG with P pebbles and Q states is $P \log N + \log Q$.

Presentation (Jumping Automaton for Graphs)

- *Takes a d-maze as input,*
- *Deterministic control,*
- *Finite set of pebbles.*

The storage of a JAG with P pebbles and Q states is $P \log N + \log Q$.

Results

- *For all $n \in \mathbb{N}$ a JAG can determine reachability of an arbitrary N nodes input graph in storage $O(\log N)^2$.*
- *No JAG can solve undirected reachability.*

Presentation (Jumping Automaton for Graphs)

- *Takes a d-maze as input,*
- *Deterministic control,*
- *Finite set of pebbles.*

The storage of a JAG with P pebbles and Q states is $P \log N + \log Q$.

Results

- *For all $n \in \mathbb{N}$ a JAG can determine reachability of an arbitrary N nodes input graph in storage $O(\log N)^2$.*
- *No JAG can solve undirected reachability.*

Not to mention other pebbles automaton, Knuth's Linking automaton...


M. Holzer, M. Kutrib, and A. Malcher.
Multi-head finite automata: Characterizations, concepts and open problems.
arXiv preprint arXiv:0906.3051, 2009.


Martin Hofmann and Ulrich Schöpp.
Pure pointer programs with iteration.
In *Computer Science Logic*, pages 79–93. Springer, 2008.


Birgit Jenner, Klaus-Jörn Lange, and Pierre McKenzie.
Tree isomorphism and some other complete problems for deterministic logspace.
DIRO, Université de Montréal, 1997.


A.N. Kolmogorov and V.A. Uspenskii.
On the definition of an algorithm.
Uspekhi Matematicheskikh Nauk, 13(4):3–28, 1958.


A. Schönhage.
Storage modification machines.
SIAM Journal on Computing, 9(3):490–508, 1980.