An in-between "implicit" and "explicit" complexity: Automata

CLÉMENT AUBERT

Abstract. Implicit Computational Complexity (ICC) makes two aspects implicit, by manipulating programming languages rather than models of computation, and by internalizing the bounds rather than using external measure. We survey how automata theory contributed to complexity with a machine-dependant with implicit bounds model.

Introduction

This survey justifies a fine-grained view on the definition of *what* Implicit Computational Complexity (ICC) wants to keep implicit:

	Machine-dependant	**Machine-independant**
Explicit bounds	Turing machine, Random access machine, Counter machine, ...	Bounded recursion on notation [9], Bounded arithmetic [6], Bounded linear logic [13], ...
Implicit bounds	Automaton, Auxiliary pushdown machine [10], Boolean circuit, ...	Descriptive complexity [11], Recursion on notation [5], Tiered recurrence [19], ...

It is common to refer to the top-left as the *classical*, or *explicit*, approach to complexity. The three others techniques *keep something implicit*. The bottom-right way is considered as the *true* implicit complexity, the one that produces all the motivating perspectives: quasi-interpretations, non-size-increasing computation, soft lambda-calculus and linear logics, to name a few. But there might be hesitations when defining ICC, for neither the "machine-independent" neither the "without explicit bounds" slogans are precise enough.

The machine-dependant with implicit bounds variant will be our subject. It characterizes complexity classes by *bounding the* quality *of the resources of a machine rather than its* quantity. To our knowledge, automata are the most canonical model of this enterprise, which dates back to the 70's: "we have attempted to characterize several tape and time complexity classes of Turing machines in terms of devices whose definitions involve only ways in which their infinite memory may be manipulated and no restrictions are imposed on the amount of memory that they use." ([16, p. 88])

Date: February 4, 2015.

This work was partly supported by the ANR-10-BLAN-0213 Logoi, the ANR-14-CE25-0005 ELICA and the ANR-11-INSE-0007 REVER.

Automata, characterizations and separation results

Definition 1. *For $k \geqslant 1$, $l \geqslant 0$, a* 2-way non-deterministicfinite automaton with k-heads and l pushdown stacks (2NFA(k,l)) *is a tuple* $M = \{\boldsymbol{S}, \boldsymbol{i}, \boldsymbol{F}, A, B, \rhd, \lhd, \boxdot, \sigma\}$ *where:*

- $\boldsymbol{S}$ *is the finite set of* states*, with* $\boldsymbol{i} \in \boldsymbol{S}$ *the* initial state *and* $\boldsymbol{F} \subseteq \boldsymbol{S}$ *the set of* accepting states*;*
- A *is the* input alphabet*,* B *is the* stack alphabet*;*
- $\rhd$ *and* $\lhd$ *are the* left *and* right endmarkers*,* $\rhd, \lhd \notin A$*;*
- $\boxdot$ *is the* bottom symbol of the stack*,* $\boxdot \notin B$*;*

From now on, we let $A_{\bowtie}$ *(resp.* $B_{\boxdot}$*) be* $A \cup \{\rhd, \lhd\}$ *(resp.* $B \cup \{\boxdot\}$*).*

- $\sigma \subseteq (\boldsymbol{S} \times (A_{\bowtie})^k \times (B_{\boxdot})^l) \times (\boldsymbol{S} \times \{-1, 0, +1\}^k \times \{\texttt{pop}, \texttt{peek}, \texttt{push}(b)\}^l)$ *is the* transition relation*, where* -1 *means to move the head one cell to the left,* 0 *means to keep the head on the current cell and* $+1$ *means to move it one cell to the right. Regarding the pushdown stacks,* `pop` *means "erase the top symbol",* `peek` *"do nothing", and, for all* $b \in B$*,* `push`(b) *is "write* b *on top of the stack".*

Given an input $n \in A^*$*, the automata is initiated in state* $\boldsymbol{i}$*, with* $\rhd\, n \lhd$ *written on its only tape, all its heads at* $\rhd$ *and all its stacks containing* $\boxdot$*. It makes (non-deterministic) transitions according to* σ *and halt accepting as soon as it reaches a state belonging to* $\boldsymbol{F}$*. We impose that the heads cannot move beyond the endmarkers and that the bottom stack symbol* $\boxdot$ *cannot be erased ("popped").*

Finally, we denote with $\mathbf{2NFA}(\mathbf{k},\mathbf{l})$ *the class of languages recognized by the* 2NFA(k,l) *automaton. Moreover, we let* $\mathbf{2NFA}(*,\mathbf{l}) = \cup_{k \geqslant 1} \mathbf{2NFA}(\mathbf{k},\mathbf{l})$*.*

Theorem 1. *The following table gives the correspondence between automata and languages (or predicates):*

	Automata	***Language***
a)	$\mathbf{2NFA}(\mathbf{1},\mathbf{2})$	*Computable*
b)	$\mathbf{2NFA}(*,\mathbf{1})$	*Polynomial time*
c)	$\mathbf{2NFA}(*,\mathbf{0})$	*Logarithmic space*
d)	$\mathbf{2NFA}(\mathbf{1},\mathbf{1})$	*Context-free*
e)	$\mathbf{2NFA}(\mathbf{1},\mathbf{0})$	*Regular*

Proof. We just sketch them, and provide references for the most difficult one. We always suppose that n is the input and $|n|$ its size.

a) $\subseteq$: Given a computable language, by the Church–Turing thesis, there exists a Turing machine T that decides it. Using some classical theorem, we can always assume that T has a single reading head and a single read-write tape. A 2NFA(1,2) can simulate T by simulating the movement of the read-only head with its head, and the content of the read-write tape with its two pushdown stacks. The first (resp. second) pushdown stack store the content on the left (resp. right) of the read-write head.

a) $\supseteq$: As 2NFA(1,2) are restrictions of Turing Machines, their simulation is obvious.

b) $\subseteq$: This part amounts to designing an equivalent Turing machine whose movements of heads follow a regular pattern. That permits to seamlessly simulate the content of the read-write tape with a pushdown stack. A complete proof [10,

pp. 9–11] as well as a precise algorithm [27, pp. 238–240] can be found in the literature.

b) $\supseteq$: This way gave birth to memoization. In a nutshell, simulating a 2NFA$(*,1)$ with a polynomial-time Turing machine cannot amount to simulate step-by-step the automaton. The reason is that for any automaton, one can design an automaton that recognizes the same language but runs exponentially slower [1, p. 197]. The technique invented by Alfred V. Aho et. al [1] and made popular by Stephen A. Cook consists in building a "memoization table" that allows the Turing machine to create shortcuts in the simulation of the automaton, decreasing drastically its computation time. It is a "*clever evaluation strategy*, applicable whenever the results of certain computations are needed more than once" [2, p. 348]. A nice explanation in the case of single head automata is in a recent and short article by R. Glück [14].

c) $\subseteq$: Let T be a Turing machine deciding the predicate with $m \times \log(|n|)$ space. Remark that an integer represented by a binary string of length p is no greater than 2^p, which takes 2^p bits in unary. Then, a binary string of length $\log(|n|)$ cannot be greater than $|n|$. So the content of the read-write tape and the position of its head is encoded as distances between $\triangleright$ and the read-only heads of the automata. That needs to compute simple operations (modulo, multiplication and remaining of a division) thanks to an extra head, but $m+3$ heads [25, pp. 191–192] are sufficient to perform this simulation. Simplest and the fastest simulations exists [27, pp. 223–225], but the overhead in terms of heads is more important.

c) $\supseteq$: The addresses of the heads on the input n takes $\log(|n|)$ to we written, hence a log-space Turing machine can easily simulate a 2NFA$(*,0)$.

d) and e): The proofs of those two fundamentals results are beyond the scope of this small survey. Their original proofs are respectively due to Chomsky [7, Theorem 1, p. 188] and Kleene [18, Theorem 6].

□

In the c) case, taking a *deterministic* automata 2DFA$(*,0)$ yields a characterization of *deterministic* log-space predicate **L** (whereas we characterized here the non-deterministic case **NL**). In the b) case, both deterministic *and* non-deterministic automata characterize *deterministic* polynomial time.

Automata can also be restricted to be 1-way (simply remove the -1 instruction from the transition relation). In the e) case of regular language (which are equal to linear space), we get in fact $\mathbf{1DFA}(\mathbf{1},\mathbf{0}) = \mathbf{2DFA}(\mathbf{1},\mathbf{0}) = \mathbf{1NFA}(\mathbf{1},\mathbf{0}) = \mathbf{2DFA}(\mathbf{1},\mathbf{0})$.

Theorem 2 (Other results of interest)**.**

([15, p. 95])
$\mathbf{1DFA}(\mathbf{k},\mathbf{0}) \subsetneq \mathbf{2DFA}(\mathbf{k},\mathbf{0})$

([15, p. 95])
$\mathbf{1NFA}(\mathbf{k},\mathbf{0}) \subsetneq \mathbf{2NFA}(\mathbf{k},\mathbf{0})$

([28, p. 339])
$\mathbf{1DFA}(\mathbf{k},\mathbf{0}) \subsetneq \mathbf{1NFA}(\mathbf{2},\mathbf{0})$

([8]) $\mathbf{1DFA}(\mathbf{k},\mathbf{1}) \subsetneq \mathbf{1NFA}(\mathbf{k},\mathbf{1})$

([25, p. 190])
$\mathbf{1NFA}(\mathbf{k},\mathbf{0}) \subseteq \mathbf{2DFA}(\mathbf{k},\mathbf{0})$

([24, p. 214])
$\mathbf{2NFA}(\mathbf{k},\mathbf{1}) \subseteq \mathbf{2DFA}(\mathbf{4}\times\mathbf{k},\mathbf{1})$

([25, p. 189])
$\mathbf{2NFA}(\mathbf{k},\mathbf{0}) \subseteq \mathbf{2NFA}(\mathbf{2}\times\mathbf{k},\mathbf{1})$

([8]) $\mathbf{1DFA}(\mathbf{k},\mathbf{1}) \subsetneq \mathbf{1NFA}(\mathbf{k},\mathbf{1})$

([23, pp. 75–76])
$\mathbf{NL} = \mathbf{L}$
iff
$\mathbf{1NFA}(\mathbf{2},\mathbf{0}) \subseteq \mathbf{2DFA}(\mathbf{k},\mathbf{0})$

Theorem 3. *For $l < 2$, $k+1$ heads are strictly stronger than k heads.*

This results holds in all situations, whenever automata are 1- or 2-way, deterministic or not, with or without pushdown stacks.[1] Even if some notes on the techniques and historic of those results exist [26, Chap. 4, Sect. 5.3, 22, pp. 67–68], no extensive survey was found, so the reader has to go for the original research papers [28, p. 338, 20, p. 106, 21, p. 383, 8, p. 179, 17, p. 35].

Conclusion

Automata theory is a major subject of computer science, very active and providing all kind of extensions and restrictions to automata. It contributed to "implicit" complexity in the sense that no external bound or measure is needed to tame their computational power: by specifying their number of ways, heads and stacks, one knows *in advance* the computational power of the device.

Those "implicit characterizations" provided decisive hints and tools to design two (truly implicit) bounded programming languages [3, 4]. Numerous other inspiring results remains to be explored and should benefit to ICC.

We did not mentioned finite state transducer, which are functional finite automata, but they also provide nice characterizations of functional complexity and beautiful results (even recently [12], where the main theorem fits in the title). The status of the input is also of interest: automata can bee feeded with 1-way inputs, trees, etc. That makes the expressive power to be really parametric in the input *and* the machine

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. "Time and Tape Complexity of Pushdown Automaton Languages". In: *Inform. Control* 13.3 (1968), pp. 186–206. DOI: 10.1016/S0019-9958(68)91087-5.

[2] T. Amtoft and J. L. Träff. "Partial Memoization for Obtaining Linear Time Behavior of a 2DPDA". In: *Theoret. Comput. Sci.* 98.2 (1992), pp. 347–356. DOI: 10.1016/0304-3975(92)90008-4.

[3] C. Aubert, M. Bagnol, P. Pistone, and T. Seiller. "Logic Programming and Logarithmic Space". In: *APLAS*. Ed. by J. Garrigue. Vol. 8858. LNCS. Springer, 2014, pp. 39–57. DOI: 10.1007/978-3-319-12736-1_3.

[4] M. Bagnol. "On the Resolution Semiring". PhD thesis. Aix-Marseille Université – Institut de Mathématiques de Marseille, 2014.

[5] S. J. Bellantoni and S. A. Cook. "A New Recursion-Theoretic Characterization of the Polytime Functions (Extended Abstract)". In: *STOC*. Ed. by S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis. ACM, 1992, pp. 283–93. DOI: 10.1145/129712.129740.

[6] S. R. Buss. *Bounded Arithmetic*. Ed. by Bibliopolis. Vol. 3. Studies in Proof Theory. Lecture Notes. 1986.

[7] N. Chomsky. *Context-Free Grammars and Pushdown Storage*. Quarterly Progress Report 65. MIT RLE, Apr. 1962, pp. 187–194.

[8] M. Chrobak. "Hierarchies of One-Way Multihead Automata Languages". In: *Theoret. Comput. Sci.* 48.3 (1986), pp. 153–181. DOI: 10.1016/0304-3975(86)90093-9.

[9] A. Cobham. "The intrinsic computational difficulty of functions". In: *Logic, methodology and philosophy of science: Proceedings of the 1964 international congress held at the Hebrew university of Jerusalem, Israel, from August 26 to September 2, 1964*. Ed. by Y. Bar-Hillel. Studies in Logic and the foundations of mathematics. North-Holland Publishing Company, 1965, pp. 24–30.

[1] To be fair, the author could not find a proof that this is the case for 1-way non-deterministic finite automata with one stack.

[10] S. A. Cook. “Characterizations of Pushdown Machines in Terms of Time-Bounded Computers”. In: *J. ACM* 18.1 (1971), pp. 4–18. DOI: 10.1145/321623.321625.
[11] R. Fagin. “Contributions to the Model Theory of Finite Structures”. PhD thesis. University of California, Berkeley, 1973.
[12] E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. “From Two-Way to One-Way Finite State Transducers”. In: *LICS*. IEEE Computer Society, 2013, pp. 468–477. DOI: 10.1109/LICS.2013.53.
[13] J.-Y. Girard, A. Scedrov, and P. J. Scott. “Bounded linear logic: a modular approach to polynomial-time computability”. In: *Theoret. Comput. Sci.* 97.1 (1992), pp. 1–66. DOI: 10.1016/0304-3975(92)90386-T.
[14] R. Glück. “Simulation of Two-Way Pushdown Automata Revisited”. In: *Festschrift for Dave Schmidt*. Ed. by A. Banerjee, O. Danvy, K.-G. Doh, and J. Hatcliff. Vol. 129. EPTCS. 2013, pp. 250–258. DOI: 10.4204/EPTCS.129.15.
[15] M. Holzer, M. Kutrib, and A. Malcher. “Multi-Head Finite Automata: Characterizations, Concepts and Open Problems”. In: *CSP*. Ed. by T. Neary, D. Woods, A. K. Seda, and N. Murphy. Vol. 1. EPTCS. 2008, pp. 93–107. DOI: 10.4204/EPTCS.1.9.
[16] O. H. Ibarra. “Characterizations of Some Tape and Time Complexity Classes of Turing Machines in Terms of Multihead and Auxiliary Stack Automata”. In: *J. Comput. Syst. Sci.* 5.2 (1971), pp. 88–117. DOI: 10.1016/S0022-0000(71)80029-6.
[17] O. H. Ibarra. “On Two-way Multihead Automata”. In: *J. Comput. Syst. Sci.* 7.1 (1973), pp. 28–36. DOI: 10.1016/S0022-0000(73)80048-0.
[18] S. C. Kleene. “Representation of events in nerve nets and finite automata”. In: vol. 34. Annals of Mathematics Studies. Princeton, NJ, USA: Princeton University Press, 1956, pp. 3–42.
[19] D. Leivant. “Stratified Functional Programs and Computational Complexity”. In: *POPL*. Ed. by M. S. Van Deusen and B. Lang. Charleston, South Carolina, USA: ACM Press, 1993, pp. 325–333. DOI: 10.1145/158511.158659.
[20] B. Monien. “Transformational Methods and their Application to Complexity Problems”. In: *Acta Inform.* 6 (1976), pp. 95–108. DOI: 10.1007/BF00263746. See also [21].
[21] B. Monien. “Corrigenda: Transformational Methods and Their Application to Complexity Problems”. In: *Acta Inform.* 8 (1977), pp. 383–384. DOI: 10.1007/BF00271346.
[22] B. Monien. “Two-Way Multihead Automata Over a One-Letter Alphabet”. In: *Theor. Inform. Appl.* 14.1 (1980), pp. 67–82.
[23] I. H. Sudborough. “On Tape-Bounded Complexity Classes and Multi-Head Finite Automata”. In: *SWAT (FOCS)*. IEEE Computer Society, 1973, pp. 138–144. DOI: 10.1109/SWAT.1973.20.
[24] I. H. Sudborough. “Separating Tape Bounded Auxiliary Pushdown Automata Classes”. In: *STOC*. Ed. by J. E. Hopcroft, E. P. Friedman, and M. A. Harrison. ACM, 1977, pp. 208–217. DOI: 10.1145/800105.803410.
[25] I. H. Sudborough. “Some Remarks on Multihead Automata”. In: *Theor. Inform. Appl.* 11.3 (1977), pp. 181–195.
[26] P. M. B. Vitányi and M. Li. “Kolmogorov complexity and its applications”. In: *Handbook of Theoretical Computer Science. volume A: Algorithms and Complexity*. Ed. by J. Van Leeuwen. Elsevier, 1990, pp. 187–254.
[27] K. W. Wagner and G. Wechsung. *Computational Complexity*. Vol. 21. Mathematics and its Applications. Springer, 1986.
[28] A. C.-C. Yao and R. L. Rivest. “$k+1$ Heads Are Better Than k”. In: *J. ACM* 25.2 (1978), pp. 337–340. DOI: 10.1145/322063.322076.

Acknowledgements

The author wish to thank, in no particular order, M. Bagnol, G. Bonfante, A. Durand, P. Jacobé de Naurois, U. dal Lago, P. Pistone, P.-A. Reynier, U. Schöpp, T. Seiller and the TEX-LATEX Stack Exchange community.

INRIA, Université Paris-Est, LACL (EA 4219), UPEC, F-94010 Créteil, France
E-mail address: clement.aubert@lacl.fr
URL: https://lacl.fr/~caubert/