Information and Computation 248 (2016) 82–103

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Two function algebras defining functions in NC^k boolean circuits

Guillaume Bonfante [a,1], Reinhard Kahle [b,2], Jean-Yves Marion [a,1], Isabel Oitavem [b,2]

[a] LORIA, BP239, 615, rue du jardin botanique, 54506 Villers-lès-Nancy, France
[b] CMA and DM, FCT-UNL, Monte de Caparica, 2829-516 Caparica, Portugal

ARTICLE INFO

Article history:
Received 1 December 2013
Available online 6 January 2016

Keywords:
Boolean circuits
NC^k
Parallel computation class
Transducers

ABSTRACT

We describe the functions computed by boolean circuits in NC^k by means of functions algebra for $k \geq 1$ in the spirit of implicit computational complexity. The whole hierarchy defines NC. In other words, we give a recursion-theoretic characterization of the complexity classes NC^k for $k \geq 1$ without reference to a machine model, nor explicit bounds in the recursion schema. Actually, we give two equivalent descriptions of the classes NC^k, $k \geq 1$. One is based on a tree structure à la Leivant, the other is based on words. This latter puts into light the role of computation of pointers in circuit complexity. We show that transducers are a key concept for pointer evaluation.

The core of *implicit computational complexity* is to provide descriptions of complexity classes which are independent from the notion of time or of space related to the underlying machine's definition. For instance, polynomial time complexity has been thoroughly examined under these terms considered Cobham–Edmonds's thesis that polynomial time is the class of feasible functions (see [8,9,12]). Doing so, some of the key concept of implicit computational complexity have been introduced. For instance, Harold Simmons [22] justifies the equivalence of some recursive schema with primitive recursion. Daniel Leivant in [14], Stephen Cook and Steve Bellantoni in [3] brought to light the notion of ramification in recursion theory. Based on logics, there are two main directions: one is based on the Curry–Howard paradigm, see Girard's Light Linear Logic [11], the other is known as Descriptive Complexity, illustrated by Immerman's characterization of polynomial time (see [13]).

In this paper, we characterize functions computed by uniform boolean circuits of polylogarithmic depth and polynomial time. More specifically, we will describe precisely each layers of the hierarchy NC^k for any $k \geq 1$. This is, to our knowledge, the first exact characterization of each layer of the hierarchy by function algebra over infinite domains in implicit complexity. The classes NC^k were firstly described based on circuits. NC^k is the class of functions accepted by uniform boolean circuit families of depth $O(\log^k(n))$ and polynomial size with bounded fan-in gates, where n is the length of the input—see [1] or [13]. In [20], Ruzzo identifies NC^k with the classes of languages recognized by alternating Turing machines (in short ATMs) in time $O(\log^k(n))$ and space $O(\log(n))$.

E-mail addresses: bonfante@loria.fr (G. Bonfante), kahle@mat.uc.pt (R. Kahle), jean-yves.marion@loria.fr (J.-Y. Marion), oitavem@fct.unl.pt (I. Oitavem).

[1] The first and the third authors received the support of ANR Elica—ANR-14-CE25-0005.

[2] The second author was partially supported by the Portuguese Science Foundation, FCT, through the projects *Hilbert's Legacy in the Philosophy of Mathematics*, PTDC/FIL-FCI/109991/2009 and *The Notion of Mathematical Proof*, PTDC/MHC-FIL/5363/2012. The second and forth authors are also partially supported by FCT through the project *Hilbert's 24th Problem*, PTDC/MHC-FIL/2583/2014 and through UID/MAT/00297/2013 (*Centro de Matemática e Aplicações*).

http://dx.doi.org/10.1016/j.ic.2015.12.009
0890-5401/

Compared to say polynomial time Turing Machine, computations with uniform boolean circuits rely on a description of inputs by pointers[3]; moreover, the machine stores only such pointers (this is the space bound). The second ingredient is parallelism, which is reflected by a tree of computation of depth (poly-)logarithmic with respect to the inputs. We will propose two different solutions to cope with these features.

If one embeds words into well-balanced binary trees, one gets structurally a) trees of logarithmic depth with respect to the size of inputs, which—by means of a tiering mechanism—may serve as a basis for time iteration, b) any sub-term of the tree is a window on a sub-word of the input, that is a pointer on the input and c) structural induction on trees fits with the tree-like nature of alternating computing. Daniel Leivant gathered these three salient aspects in his description of NC by means of so-called ramified tree recurrence. His schema is parametrized by a number k and he proves $\mathsf{RSR}_k \subseteq \mathsf{NC}^k \subseteq \mathsf{RSR}_{k+2}$ for any $k > 1$, missing however the exact delineation of the hierarchy. His ideas have been reworked by Guillaume Bonfante, Reinhard Kahle, Jean-Yves Marion and Isabel Oitavem in [5] where mutual in place recursion (MIP) is introduced. An other source of inspiration of [5] was the description of NC^1, that is ALOGTIME, by Daniel Leivant and Jean-Yves Marion [17].

Based on a variant of ramified recurrence, Steve Bloch characterized ALOGTIME in [4]. Ramified recurrence over trees was introduced by Daniel Leivant in [15], then reconsidered by Steve Bellantoni in [2] where he gives a characterization of alternating poly-log functions.

Using trees however leads to (at least) two issues. First, computations are done on well-balanced tree, thus not on a free algebra. Hence, we are not talking of an intrinsic property of the recursion schema, but on a property of the schema for some restricted subset of trees. Second, since trees serve both for inputs and pointers, the schema does not reveal the realm of pointer computations. For that reason, we provide a second description of the hierarchy based on two other recursion schema, one is called *rational bitwise equations* (RBE). This schema describes basic functions as a two step process: first, transducers connect some input bits together, second a finite map is applied on these latter bits. The key ingredient is that bits indices within inputs—in other words, pointers—are computed by transducers, thus involving a very weak form of induction. The second schema is time iteration, which corresponds to a ramified version of primitive recursion on pointers. Again, time iteration is performed on pointers. To sum up, the control of the computations in RBE only relies on pointers. Computations on data boils down to finite maps on the input alphabet.

Based on some word recurrence schema, we mention here the work of Clote on the class NC and the hierarchy which appeared in [7]. Compared to our proposition, Clote needs explicit bounds on the recursion schema, thus violating one of the "rules" of implicit computational complexity. Taking a view based on logics, boolean circuits where addressed by Immerman in terms of Descriptive Complexity, see for instance [13]. They have been considered by Mogbil et al. in [19].

The paper grew out of our earlier work [5]. Compared to it, we present and discuss some variations on the schema which we prove to be all equivalent. This shows that the schema is natural and robust. More importantly, we introduce a new schema, namely RBE, and we prove its equivalence with MIP. Our thesis is that this result enforces the definition of MIP, showing even more its robustness. Second point, the new recursion schema opens a new window on computation on pointers in implicit computational complexity.

In Section 1, we recall some facts concerning words, trees and finite state transducers. We present words and trees since we define algebras of functions on words and trees. Concerning finite state transducers, we use them for pointer computation. In Section 2, we present the Mutual In Place Recursion Schema and some variations on it. Together with a structural recursion schema and a parametrized schema k-TI, $k \in \mathbb{N}$, we define algebras of functions INC^k. Each algebra INC^k, for $k \geq 1$, describes exactly functions in NC^k. We come in Section 3 to Rational Bitwise Equations, a model of computations based on the separation between pointers computation and data computations. It is shown to be equivalent to the Mutual In Place schema. In order to describe all the layers of the hierarchy, we use a mechanism of strict ramification à la Leivant–Marion; again we provide a parametrized hierarchy of functions $(\mathsf{RBE}^k)_{k\in\mathbb{N}}$. Section 4 is devoted to the proof of Proposition 34, that is any function in NC^k can be computed within INC^k and Section 5 to the converse part, that is Proposition 36. Finally, Section 6 proves the equality $\mathsf{RBE}^k = \mathsf{INC}^k$ for any $k \geq 0$.

1. Preliminaries

Given some set X, we define $\mathfrak{P}(X) = \{U \mid U \subseteq X\}$ of subsets of X. The set 1 is an arbitrary singleton set. It is clear that $1 \times X$ is isomorphic to X. Thus, a sequence λ indexed by $1 \times X$ will be presented as $(\lambda_x)_{x\in X}$.

As we will come back to it later, we recall that, given a semi-ring $(A, 0, +, 1, \times)$ and two finite sets P and Q, a matrix of dimension $P \times Q$ on A is a table data $m = (m_{p,q})_{(p,q)\in P\times Q}$ whose entries are in A. The set of such matrices is written $A^{P\times Q}$. Matrices of equal dimensions can be summed. Given m and n of dimension $P \times Q$, $m+n = (m_{p,q}+n_{p,q})_{p,q}$. Matrices are multiplied according to the usual rules. Given $m = (m_{p,q})_{(p,q)\in P\times Q}$ and $n = (n_{q,r})_{(q,r)\in Q\times R}$, we set $m \times n$—of dimension $P \times R$—defined by its components $(m \times n)_{p,r} = \sum_{q\in Q} m_{p,q} \times n_{q,r}$.

Given a matrix $m \in A^{P\times Q}$, $m_{p,q}$ denotes the entry at position p, q. Sometimes, when indices become too heavy, the entry is denoted $m[p, q]$.

To end with general notations, all along, sequences $x_1, \ldots, x_k$ are written $\vec{x}$ when the context makes it clear.

[3] As for Random Access Machines, a sequential model of computation.