

# Unambiguity of Circuits

Klaus-Jörn Lange\*

Institut für Informatik  
Technische Universität München  
Arcisstr. 21  
D-8000 München 2

## Abstract

The concept of unambiguity of circuits is considered. Several classes of unambiguous circuitfamilies within the  $NC$ -hierarchy are introduced and related to unambiguous automata and to  $PRAM$ 's with exclusive write-access. In particular, we show  $CREW-TIME(\log^k n) = UnambAC^k$  for each positive integer  $k$ .

## 1 Introduction

A central object of interest of parallel complexity theory is the class  $NC$  (see e.g. [7]), which can be characterized by several devices, among them: parallel random access machines (see [9] or [10]), boolean circuits (see [3]), alternating Turing machines (see [5]), and deterministic and nondeterministic auxiliary push-down automata (see [6]).

With appropriate complexity bounds these devices recognize exactly the languages in  $NC$ . If we take a closer look on the time or depth bounds, by considering functions of order  $O(\log^k n)$  for any fixed  $k$  instead of  $O(\log^{O(1)} n)$  we get several intertwining hierarchies, the union of which is  $NC$ . Located between  $DSPACE(\log n)$  and  $P$ , these two classes would be separated if those hierarchies could be shown to be proper, i.e.: if it could be shown, that the increase of time or depth from  $O(\log^k n)$  to  $O(\log^{k+1} n)$  leads to different classes for some  $k$ . Some of these hierarchies coincide, thus characterizing the relation between different models of parallel computations. In particular, the following equivalences have been obtained:

- Time of a  $CRCW$ -PRAM, depth of alternation on a Turing machine, and depth of an uniform circuit of unbounded fan-in (see [21]),
- Time of an nondeterministic auxiliary push-down automaton, Treesize of an alternating Turing machine, and depth of an uniform circuit of semi-unbounded fan-in (see [18] and [24]), and
- Time of an alternating Turing machine and depth of an uniform circuit of bounded fan-in (see [19]).

In addition, on a less general level, Dymond and Ruzzo showed in [8] a similar relation between time of an  $CROW$ -PRAM and time of a deterministic auxiliary push-down automaton. As we see, this list contains neither  $CREW$  nor  $EREW$ -PRAMs, which are not characterized by any other parallel or sequential device, albeit most PRAM algorithms are working on PRAMs of these types. Our idea to break this isolation is by pursuing the following relationship between context-free languages and  $PRAM$ -types:

- $CFL \subseteq CRCW-TIME(\log n)$  [18],
- $UnambCFL \subseteq CREW-TIME(\log n)$  [20],
- $DCFL \subseteq CROW-TIME(\log n)$  [8].

---

\*work supported by the Deutsche Forschungsgemeinschaft, SFB 342: TP A4, and by the International Computer Science Institute

It is the aim of this paper to further relate the concepts of exclusiveness and of unambiguity. This we want to do by considering the concept of unambiguity of circuits. Here we have to distinguish the notions of unambiguity and of uniqueness. While the first one uses uniqueness of an acceptance path as a restriction of computational power, the second one increases computational power by using uniqueness of an acceptance path as a tool. That is, in the first case we forbid the existence of multiple acceptance paths while in the second case we might use this fact to reject a computation. Thus the concept of uniqueness applied to circuits would lead us to use something like unbounded fan-in gates for unique existence. In contrast to that, an unambiguous circuit has to restrict ways of multiple acceptance. We will do this in the following way: Just as in a *CREW*- or *EREW-PRAM* we assume the machine and the algorithm to avoid any multiple access to a memory cell and may think, that in case of violation of this rule the corresponding cell might be destroyed, we assume the circuit to avoid any multiple 1-Input to *OR*-gates of unbounded fan-in (resp. multiple 0-Input to *AND*-gates of unbounded fan-in). Otherwise, that is if an *OR*-gate of unbounded fan-in gets two or more 1's as Input, we have no information of the output of this gate. Unambiguous circuit classes defined in this way show very close relationships not only to *CREW-PRAMs* but also to unambiguous Turing machines and auxiliary push-down automata (see also [14] and [15]).

In particular, we will characterize *CREW-PRAMs* in terms of unambiguous circuits by the equation **CREW-TIME**( $\log^k n$ ) = **UnambAC**<sup>k</sup>. This answers an open question posed in [21].

## 2 Preliminaries

The reader is assumed to be familiar with the basic concepts and notations in formal language and computational complexity theories as they are contained for instance in [11] or [25]. In addition, we often use the notion  $LOG(\mathcal{A})$  to denote the class of all languages reducible by logspace many-one reductions to members of the class  $\mathcal{A}$ . If  $\mathbf{X}(\mathbf{f}(n))$  denotes a complexity class with a resource bound  $\mathbf{f}(n)$  we let  $\mathbf{X}(\mathbf{pol})$  be the union of all  $\mathbf{X}(\mathbf{p}(n))$  over all polynomials  $\mathbf{p}(n)$ .

In the following we are going to review shortly some concepts and facts of parallel complexity theory. Detailed definitions and constructions may be found in [7], [1], [2], and in [16] or in the cited references.

### Circuits

In this subsection we present some basic facts concerning boolean circuits. A *circuit*  $C$  is a finite acyclic graph. Nodes of indegree (outdegree) zero are called *inputs* (*outputs*). Inner nodes are labelled by boolean functions, throughout this paper by negations, disjunctions, and conjunctions. The inner nodes are often called *gates* and the edges *wires*. Given an assignment of boolean values to all inputs each gate evaluates to either *TRUE* (or 1) or *FALSE* (or 0) according to the interconnection structure of  $C$ . If  $C$  has just one output, we might use  $C$  to recognize binary languages defining  $L(C)$  to be the set of assignments to the inputs which let the output evaluate to *TRUE*. The *size* of  $C$  is the number of its gates (i.e.: not counting the inputs).<sup>1</sup> The *depth* of  $C$  is the length of the longest path connecting an input with an output.

A *circuit family*  $\mathbf{C}$  is a set  $\{C_n | n \geq 0\}$  of circuits, where each  $C_n$  has exactly  $n$  inputs.  $\mathbf{C}$  has *polynomial size* iff for some polynomial  $p(\cdot)$ , the size of each  $C_n$  is bounded by  $p(n)$ . Similarly, the *depth* of  $\mathbf{C}$  is *bounded by*  $\log^k n$  iff for some constant  $c > 0$  the depth of each  $C_n$  is bounded by  $c \cdot \log^k n$ . Further on,  $\mathbf{C}$  is of *bounded fan-in* iff for some positive integer (usually 2) the indegree of each gate in each  $C_n$  is bounded by  $m$ .  $\mathbf{C}$  is of *semi-unbounded fan-in* iff no  $C_n$  contains a negation<sup>2</sup> and for some positive integer  $m$  in each  $C_n$  the indegree of each gate labelled as a conjunction is bounded by  $m$ . If there is no bound on the indegrees we say that  $\mathbf{C}$  is of *unbounded fan-in*. When working with unambiguous circuits we will distinguish between gates of bounded and of unbounded fan-in, since these will have to be treated differently. Thus, when necessary, we will denote *OR* and *AND*-gates of unbounded fan-in as  $\exists$  and  $\forall$ -gates. This means, that with  $\mathbf{C}$  there exists a positive integer

<sup>1</sup>Sometimes the number of wires is a more appropriate measure. But since we will work with circuits of polynomial and not of polylogarithmic size, this would make no difference

<sup>2</sup>To make this concept reasonable we have to assume, that all inputs are given together with their negations, i.e.: each  $C_n$  has  $2 \cdot n$  inputs.

$m$  such that in each  $C_n$  each *OR* and each *AND*-gate has a fan-in bounded by  $m$ , while each  $\exists$  and each  $\forall$ -gate within some  $C_n$  may have a fan-in as large as the size of  $C_n$ .

In order to relate classes of languages defined by circuits with complexity classes it is necessary to consider *uniform* circuit families by requiring that the members of a circuit family are “sufficiently similar” with each other. There are several uniformity conditions which fortunately turned out to be equivalent in most cases (see [19]). Throughout this paper we will use the notion of **DSPACE**( $\log n$ )-uniformity: **C** is called **DSPACE**( $\log n$ )-*uniform* iff the mapping  $n \mapsto \langle C_n \rangle$  is computable within logarithmic space. Here  $\langle C_n \rangle$  denotes an adequate coding of a circuit. We call the logspace machine computing this mapping the *uniformity machine*.

For  $k \geq 1$  we will denote by **NC** <sup>$k$</sup>  (**SAC** <sup>$k$</sup> , **AC** <sup>$k$</sup>  respectively) the families of languages recognizable by **DSPACE**( $\log n$ )-uniform, polynomially sized,  $O(\log^k n)$ -depth bounded circuit families of bounded (semi-unbounded, unbounded respectively) fan-in<sup>3</sup>. The following relations are well-known:

$$\mathbf{NC}^k \subseteq \mathbf{SAC}^k \subseteq \mathbf{AC}^k \subseteq \mathbf{NC}^{k+1} \quad \text{and}$$

$$\mathbf{NC}^1 \subseteq \mathbf{DSPACE}(\log n) \subseteq \mathbf{NSPACE}(\log n) \subseteq \mathbf{SAC}^1 \subseteq \mathbf{AC}^1.$$

It has to be mentioned, that throughout this paper only *layered* circuits are considered. That is, we assume that for every gate  $a$  in a circuit  $C$  there exists a *height*  $h(a)$  such that every directed path from any input to  $a$  has length  $h(a)$ ; in particular, each predecessor of  $a$  has height  $h(a) - 1$ . (Some authors demand, that in a *layered* circuit existential and universal gates alternate. In addition, it seems reasonable to include the height of gates into the description of a layered circuit, i.e.: to let the height function in a uniform circuit family be computable by the uniformity machine.) This is no restriction for *NC*, *SAC*, or *AC*-circuits. But we were unable to show a corresponding ‘normal form’ result for the case of unambiguous *AC*-circuits, introduced in the next section. On the other hand, all constructions of unambiguous circuits in this paper are of a levelled structure, in which each level is a circuit of bounded depth and its outputs are given as inputs either to the next level or to the final one. A circuit of this structure can be transformed easily into a layered circuit.

## Parallel Random Access Machines

The concept of a *PRAM* goes back to [9] and [10]. Roughly, a *PRAM* is a set of Random Access Machines, called *Processors*, working synchronously and communicating via a *Global Memory*. Each step takes one time unit regardless whether it performs a local or a global (i.e.: remote) operation. All processors execute in parallel the same sequence of statements  $S_1, S_2, \dots, S_K$  which is independent of the input. Let each processor have *Local Memory* cells  $L_1, L_2, \dots, L_{q(n)}$ , and let  $G_1, G_2, \dots, G_{q(n)}$  be the cells of global memory, where  $n$  is the length of the input, which is given in  $G_1, \dots, G_n$ , and  $q$  is a polynomial. A computation has ended if all processors have reached a *HALT*-statement. Each Statement  $S_\mu$  is of one of the following types:

*Indirect Writes:*

$$G_{L_a} = L_b \text{ and } L_{L_a} = L_b$$

*Indirect Reads:*

$$G_a = L_{L_b} \text{ and } L_a = L_{L_b}$$

*Binary Operation:*

$$L_a = L_b \circ L_c^4$$

*Constants:*

$$L_a = \langle \text{constant} \rangle, L_a = \text{LENGTH}^5, \text{ or } L_a = \text{PIN}^6$$

*Jumps:*

$$\text{GOTO } S_a \text{ or IF } L_a > 0 \text{ THEN GOTO } S_b$$

<sup>3</sup>Actually, a reasonable definition of **NC**<sup>1</sup> seems to require a uniformity notion more delicate than **DSPACE**( $\log n$ )-uniformity yielding the probably smaller class **ATIME**( $\log n$ ) (see [19]).

<sup>4</sup>Since the wordlength is logarithmically bounded and since we deal with **DSPACE**( $\log n$ )-uniform circuits, we can admit arbitrary **DSPACE**( $\log n$ )-computable functions.

<sup>5</sup>This gives the length of the input, i.e.:  $n$

<sup>6</sup>This gives the *Processor Identification Number*

*Others:*

*HALT* or *NOOP*.

Throughout this paper we will consider PRAMs where both the number of processors and all occurring data and addresses are bounded polynomially in the length of the input; the later condition is equivalent to a logarithmic bound on the word length of all global and local memory cells.

There are several versions of this device concerning the way how the simultaneous memory access is handled. There are three versions concerning the write access: a machine with *Concurrent Write* access allows the simultaneous write access of several processors to the same memory cell in one step. There are several conventions how to solve this conflict, i.e.: how to determine which will be the new value of the referenced cell. Fortunately, in our context all these methods are equivalent. A machine with *Exclusive Write* access forbids simultaneous writes and requires that in each step at most one processor may change the content of a global memory cell. A machine with *Owner Write* access is even more restricted by assigning to each cell of global memory a processor, called the *Write-Owner*, which is the only one to have write access to this memory cell. Correspondingly, we get three ways to manage read access to the global memory: *Concurrent Read*, *Exclusive Read*, and *Owner Read*. In this way we get nine versions of PRAMs, denoted as *XRYW-PRAMs* with  $X, Y \in \{O, E, C\}$ , where  $XR$  specifies the type of read access and  $YW$  that of the write access, where the access types are designated by their initials. By **XRYW-TIME(f(n))** we denote the class of all languages recognizable in time  $f$  by *XRYW-PRAMs* with a polynomial number of processors. By definition we know that **XRYW-TIME(f)  $\subseteq$  X'RY'W-TIME(f)** for  $X, X', Y, Y' \in \{O, E, C\}$  if  $X \leq X'$  and  $Y \leq Y'$  where we set  $O \leq E \leq C$ .

By [21] and [17] we have the following relationships for  $k \geq 1$ :

$$\mathbf{CRCW - TIME}(\log^k n) = \mathbf{AC}^k,$$

$$\mathbf{NC}^k \subseteq \mathbf{OROW - TIME}(\log^k n),$$

$$\mathbf{DSPACE}(\log n) \subseteq \mathbf{OROW - TIME}(\log n),$$

$$\mathbf{ORCW - TIME}(f) = \mathbf{ERCW - TIME}(f), \text{ and}$$

$$\mathbf{OREW - TIME}(f) = \mathbf{EREW - TIME}(f).$$

**Remark 1** In the most powerful model, the *CRCW-PRAM*, the global memory behaves like a shared memory, since each processor can access each cell of global memory. In the most restricted model, the *OROW-PRAM*, however, the global memory is deteriorated to a set of one-directional channels between pairs of processors. Thus an *OROW-PRAM* is something like a completely connected synchronous network. Although this model seems to be much more restricted, the relation

$$\mathbf{NC}^k \subseteq \mathbf{OROW - TIME}(\log^k n) \subseteq \mathbf{CRCW - TIME}(\log^k n) = \mathbf{AC}^k \subseteq \mathbf{NC}^{k+1}$$

indicates that it is a model as “parallel” as a *CRCW-PRAM*.

## Auxilliary Push-Down Automata

Another device interesting in this context is the *Auxilliary Push-Down Automaton* which might be thought of as a push-down automaton augmented with a  $S(n)$ -space bounded working tape and two-way access to its input. Cook showed in [6] that both the deterministic and the nondeterministic version of this automaton type recognize exactly the class  $\bigcup_{c>1} \mathbf{DTIME}(c^{\mathbf{S}(n)})$  of languages acceptable in deterministic time exponential in  $S(n)$ . Throughout this paper we will work with the case  $S(n) = \lceil \log n \rceil$ .

By restricting this device with an additional time bound we get classes down in the *NC*-hierarchy. Let **DAuxPDA-TIME(f(n))** be the class of all languages recognizable by deterministic auxilliary push-down automata with logarithmically space bounded working tapes which are time bounded by  $O(f(n))$ . **NAuxPDA-TIME(f(n))** denotes the corresponding nondeterministic class. By [24], [19], and [8] we have the following relationships for  $k \geq 1$ :

$$\mathbf{DAuxPDA - TIME}(\text{pol}) = \mathbf{CROW - TIME}(\log n),$$

$$\mathbf{NAuxPDA-TIME}(c^{\log^k n}) = \mathbf{SAC}^k, \text{ and}$$

$$\mathbf{NC}^k \subseteq \mathbf{DAuxPDA-TIME}(c^{\log^k n}).$$

Further on, Sudborough showed in [23]

$$\mathbf{DAuxPDA-TIME}(\text{pol}) = \mathbf{LOG}(\mathbf{DCFL}) \text{ and } \mathbf{NAuxPDA-TIME}(\text{pol}) = \mathbf{LOG}(\mathbf{CFL}).$$

Finally, we mention in passing, that many classes defined by circuits, PRAMs, or APDAs possess characterizations in terms of alternation. Thus  $\mathbf{CRCW-TIME}(\log^k n) = \mathbf{AC}^k$  corresponds to depth of alternation,  $\mathbf{NAuxPDA-TIME}(c^{\log^k n}) = \mathbf{SAC}^k$  corresponds to treesize bounded alternation, and  $\mathbf{NC}^k$  corresponds to alternating time (see [18], [19], [21], and [24]).

### 3 Unambiguity

The notion of unambiguity is well known with respect to automata and grammars. Usually, a nondeterministic automaton  $M$  is said to be *unambiguous*, if for every input there is at most one accepting computation in  $M$ . Of course, every deterministic automaton is unambiguous. This leads in a natural way to the classes  $\mathbf{UnambSPACE}(f)$  and  $\mathbf{UnambAuxPDA-TIME}(f)$ .

This concept is sometimes called *weak* unambiguity to distinguish it from the alternative notion of *strong unambiguity*, in which between any two configurations there is at most one computation path, which leads from the first configuration to the second one. Weak unambiguity demands this for pairs of initial and accepting configurations, only. In this way we get the classes  $\mathbf{StUnambSPACE}(f)$  and  $\mathbf{StUnambAuxPDA-TIME}(f)$  (see [4]). By definition we have

$$\mathbf{DSPACE}(f) \subseteq \mathbf{StUnambSPACE}(f) \subseteq \mathbf{UnambSpace}(f) \subseteq \mathbf{NSPACE}(f) \text{ and}$$

$$\mathbf{DAuxPDA-TIME}(f) \subseteq \mathbf{StUnambAuxPDA-TIME}(f) \subseteq \mathbf{UnambAuxPDA-TIME}(f) \subseteq \mathbf{NAuxPDA-TIME}(f).$$

Considering the families  $\mathbf{CFL}$  of context-free languages,  $\mathbf{DCFL}$  of deterministic context-free languages,  $\mathbf{LIN}$  of linear context-free languages, and  $\mathbf{DLIN}$ <sup>7</sup> of languages accepted by deterministic push-down automata, which do not push a symbol after a pop-move was performed, the following equations are well-known [22], [23]:

$$(*) \quad \begin{aligned} \mathbf{NAuxPDA-TIME}(\text{pol}) &= \mathbf{LOG}(\mathbf{CFL}), \\ \mathbf{DAuxPDA-TIME}(\text{pol}) &= \mathbf{LOG}(\mathbf{DCFL}), \\ \mathbf{NSPACE}(\log n) &= \mathbf{LOG}(\mathbf{LIN}), \text{ and} \\ \mathbf{DSPACE}(\log n) &= \mathbf{LOG}(\mathbf{DLIN}). \end{aligned}$$

Hence, one might be tempted to assume

$$(**) \quad \begin{aligned} \mathbf{UnambAuxPDA-TIME}(\text{pol}) &\stackrel{?}{=} \mathbf{LOG}(\mathbf{UnambCFL}) \text{ and} \\ \mathbf{UnambSPACE}(\log n) &\stackrel{?}{=} \mathbf{LOG}(\mathbf{UnambLIN}) \end{aligned}$$

where  $\mathbf{UnambCFL}$  ( $\mathbf{UnambLIN}$  resp.) is the family of languages generated by unambiguous (linear) context-free grammars. Unfortunately, the corresponding constructions in (\*) preserve determinism, but not unambiguity.

If we want to transfer these concepts to boolean circuits, we have to exchange the notion of an accepting computation by that of an accepting subcircuit. In this way we would consider circuits, which for all inputs have at most one accepting subcircuit, which is equivalent to the fact, that no member of a certain, input-dependent set of “relevant” OR-gates is ever reached by more than one predecessor carrying a 1. A closer investigation shows, that a more adequate notion is obtained by demanding this only for gates of unbounded fan-in and, in addition, by putting a dual restriction on AND-gates, too. Thus, we come to the following definition.

<sup>7</sup> $\mathbf{DLIN}$ , which is easily seen to be  $\mathbf{DSPACE}(\log n)$ -complete even with respect to  $\mathbf{NC}^1$ -reductions, is to be distinguished from the *Deterministic Linear Languages* introduced in [12], which are contained in  $\mathbf{NC}^1$

**Definition 1** Let  $C$  be a circuit on  $n$  inputs built up by *AND*- and *OR*-gates of bounded fan-in and by  $\exists$ - and  $\forall$ -gates of unbounded fan-in.  $C$  is *unambiguous* if for all  $2^n$  assignment of the inputs no  $\exists$ -gate receives a 1 by two or more of its predecessors and no  $\forall$ -gate receives a 0 by two or more predecessors. A circuit family  $\mathbf{C} = \{C_n | n \geq 0\}$  is called *unambiguous* if  $C_n$  is unambiguous for each  $n$ .

This notion may be illustrated by the idea of a ‘vulnerable gate’: We might think that all gates of unbounded fan-in in an unambiguous circuit are *vulnerable*, i.e.: may be destroyed if not handled properly. Thus, a vulnerable  $\exists$ -gate will correctly output a 0 if none of its inputs is equal to 1 and will output a 1 if exactly one of its inputs is a 1. But if there are more than one inputs carrying a 1 the gate is overloaded and, being vulnerable, will be destroyed. After that, we have no information concerning the output of this gate. Thus, in an unambiguous circuit all gates of unbounded fan-in are vulnerable and the unambiguity of the circuit implies that for no input-assignment any vulnerable gate is overloaded. Thus, we see that this notion of unambiguity corresponds to strong unambiguity. When constructing unambiguous circuits, terms like unambiguous disjunctions will be used to indicate that certain *OR*-gates will for no input receive more than one value 1 from their predecessors.

We mention in passing that in analogy to unambiguous context-free grammars, both the unambiguity of a circuit family and the exclusiveness (concerning read or write access) of a PRAM algorithm are undecidable.

Let us at this point stress the fundamental difference between the notions of unambiguity, that is demanding the uniqueness of certain computations, and of uniqueness, that is excluding inputs which are accepted by more than one computation. While the first one is a restriction and thus unambiguous classes are contained in the corresponding nondeterministic classes, the later one is a tool and usually classes defined by uniqueness are incomparable with or containing the corresponding nondeterministic classes.

Using Definition 1 to define an unambiguous version of  $\mathbf{AC}^k$ ,  $k \geq 0$ , leads straightforward to the family  $\mathbf{UnambAC}^k$ . For  $\mathbf{SAC}^k$ , however, we have two different possibilities which will lead to the families  $\mathbf{UnambSAC}^k$  and  $\mathbf{UnambS}^E\mathbf{AC}^k$ <sup>8</sup>:

**Definition 2** Let  $\mathbf{UnambAC}^k$  be the family of all languages recognizable by logspace-uniform unambiguous circuit families of polynomial size and depth bounded by  $O(\log^k n)$  using both *AND*-gates and *OR*-gates of bounded as well as of unbounded fan-in.  $\mathbf{UnambSAC}^k$  denotes the class of all languages recognizable by the corresponding circuit families using only  $\exists$ -gates of unbounded fan-in and *AND*-gates of bounded fan-in. (Thus, in this case all disjunctions are “vulnerable”). Finally,  $\mathbf{UnambS}^E\mathbf{AC}^k$  consists of all languages recognizable by the corresponding circuit families using  $\exists$ -gates of unbounded fan-in, *AND*-gates of bounded fan-in, and in addition *OR*-gates of bounded fan-in.

Thus the difference between *UnambSAC* and *UnambS<sup>E</sup>AC*-circuits is that in the later ones we may use small *OR*-gates which are not vulnerable.

**Proposition 3**  $\mathbf{NC}^k \subseteq \mathbf{UnambSAC}^k$  for  $k \geq 1$ .

**Proof:** Let  $C$  be an  $\mathbf{NC}^k$ -circuit. Without loss of generality we may assume  $C$  to be layered. Since we can modify an  $\mathbf{NC}^k$ -circuit  $C$  without leaving  $\mathbf{NC}^k$  in such a way that for every gate  $a$  in  $C$  there exists a gate  $\bar{a}$  computing the complement of  $a$ , we can replace a robust *OR* of two gates  $a$  and  $b$  by the *OR* of  $a \wedge b$ ,  $\bar{a} \wedge b$ , and  $a \wedge \bar{b}$ , which can never get a multiple input of 1’s. This results in an layered *UnambSAC<sup>k</sup>*-circuit (which does not contain gates of unbounded fan-in!).  $\square$

As a consequence the  $\mathbf{UnambXAC}^k$ -hierarchies intertwine with the  $\mathbf{NC}$ -hierarchy:

$$\mathbf{NC}^k \subseteq \mathbf{UnambSAC}^k \subseteq \mathbf{UnambS}^E\mathbf{AC}^k \subseteq \mathbf{UnambAC}^k \subseteq \mathbf{AC}^k \subseteq \mathbf{NC}^{k+1}$$

$$\text{and } \mathbf{UnambS}^E\mathbf{AC}^k \subseteq \mathbf{SAC}^k \subseteq \mathbf{AC}^k.$$

$\mathbf{SAC}^k$  and  $\mathbf{UnambAC}^k$  seem to be incomparable, since  $\mathbf{SAC}^k$  coincides with  $\mathbf{NAuxPDA-TIME}(c^{\log^k n})$  ([24]) and  $\mathbf{UnambAC}^k$  will be shown to be equal to  $\mathbf{CREW-TIME}(\log^k n)$ .

<sup>8</sup>The superscript “E” stands for “extended”. In [13] and [14]  $\mathbf{UnambSAC}^k$  was called  $\mathbf{UnambRAC}^k$  and  $\mathbf{UnambS}^E\mathbf{AC}^k$  was called  $\mathbf{UnambSAC}^k$ . But meanwhile the former  $\mathbf{UnambRAC}$ -families turned out to be the more appropriate unambiguous version of the  $\mathbf{SAC}$ -families

## 4 Results

In this main section we will investigate unambiguous circuits and thereby relate the concepts of unambiguity and of exclusiveness. Let us first consider the following relations known in the nondeterministic case:

$$\mathbf{NL} \subseteq \mathbf{SAC}^1 = \mathbf{NAuxPDA} - \mathbf{TIME}(\mathbf{pol}).$$

We are now going to show the following in the unambiguous case:

$$\mathbf{StUnambSPACE}(\mathbf{logn}) \subseteq \mathbf{UnambSAC}^1 \subseteq \mathbf{UnambAuxPDA}(\mathbf{pol}).$$

**Proposition 4**  $\mathbf{StUnambSPACE}(\mathbf{logn}) \subseteq \mathbf{UnambSAC}^1$

**Idea of proof:** Given a logspace machine  $A$  with polynomial upper bound  $p(n)$  of the running time, we let a uniformity machine  $B$  first compute  $n := |v|$  and  $T := p(n)$  for a given input  $v$ .  $B$  works with gates labelled  $\langle K, K', t \rangle$  meaning that the configuration  $K$  of  $A$  reaches  $K'$  in *exactly*  $t$  steps. The output is a vulnerable disjunction of all  $\langle K_0, K_f, t \rangle$ , where  $1 \leq t \leq T$ ,  $K_0$  is the starting configuration, and  $K_f$  is a final configuration. The usual Savitch-decomposition of  $\langle K, K'', t \rangle$  into  $\langle K, K', \lceil t/2 \rceil \rangle$  AND  $\langle K', K'', \lfloor t/2 \rfloor \rangle$  is unambiguous since there can be at most one configuration  $K'$ , which is reachable from  $K$  in exactly  $\lceil t/2 \rceil$  steps and which reaches  $K''$  within exactly  $\lfloor t/2 \rfloor$  steps. Otherwise  $A$  wouldn't be strongly unambiguous. Thus, the corresponding gate labelled  $\langle K, K'', t \rangle$  is a vulnerable disjunction (of unbounded fan-in) over all conjunctions of  $\langle K, K', \lceil t/2 \rceil \rangle$  with  $\langle K', K'', \lfloor t/2 \rfloor \rangle$ . To make the construction layered, we assume  $\langle K, K'', t \rangle$  to belong to the level  $\lceil \log t \rceil$ . If the levels of  $\lceil t/2 \rceil$  and  $\lfloor t/2 \rfloor$  are different, we have to insert appropriate delays between the conjunction belonging to  $K'$  and  $\langle K', K'', \lfloor t/2 \rfloor \rangle$  for each  $K'$ . In addition, delays are needed between the  $\langle K_0, K_f, t \rangle$ -gates and the output  $\exists$ -gate. Obviously, the constructed circuit is  $\mathbf{DSPACE}(\mathbf{logn})$ -uniform (In fact, it is even  $\mathbf{DLOGTIME}$ -uniform!).  $\square$

**Remark 2** This construction does not work for unambiguous logspace Turing machines, since there might be ambiguous partial computations which are not part of any accepting computation, but might lead to the destruction of a vulnerable gate. Meanwhile in [4] it was shown, that  $\mathbf{StUnambSPACE}(\mathbf{logn})$  is even contained in  $\mathbf{DAuxPDA-TIME}(\mathbf{pol})$ , which is a subset of  $\mathbf{UnambSAC}^1$  as will be shown in Theorem 9 below.

**Remark 3** As a consequence of proposition 4 we get  $\mathbf{DSPACE}(\mathbf{logn}) \subseteq \mathbf{UnambSAC}^1$ . Thus,  $\mathbf{DSPACE}(\mathbf{logn})$ -uniformity seems to be an adequate notion, when working with unambiguous circuits of unbounded or semi-unbounded fan-in.

Since it is possible in a similar way to build for each  $i$   $\mathbf{UnambSAC}^1$ -circuits  $C_i$  and  $C'_i$  which compute whether the  $i$ -th bit of  $f(v)$  is 1 or 0 for some logspace computable function  $f$ , we get:

**Corollary 5** For each  $k \geq 1$   $\mathbf{UnambSAC}^k$ ,  $\mathbf{UnambS}^{\mathbf{EAC}^k}$ , and  $\mathbf{UnambAC}^k$  are closed under  $\mathbf{LOG}$ -reducibilities.

**Proposition 6**  $\mathbf{UnambSAC}^k \subseteq \mathbf{UnambAuxPDA-TIME}(c^{\log^k n})$  for  $k \geq 1$ .

**Idea of proof:** We apply the corresponding algorithm for showing  $\mathbf{SAC}^k \subseteq \mathbf{NAPDA-TIME}(c^{\log^k n})$ . Working recursively (top-)down from the output gate to input-gates of an  $\mathbf{UnambRAC}^k$ -circuit we evaluate an  $\mathbf{AND}$ -gate by two recursive calls to its predecessors. For a vulnerable  $\exists$ -gate we guess nondeterministically but unambiguously the uniquely existing predecessor carrying a 1.  $\square$

Observe that this algorithm does not work strongly unambiguously: in case of rejection there are several paths leading to the same (rejecting) configuration!

**Remark 4** It seems to be at least difficult to extend Proposition 6 to the class  $\mathbf{UnambS}^{\mathbf{EAC}^1}$ . The trick to evaluate gates of bounded fan-in 'deterministically' by working through both predecessors does not work for  $\mathbf{OR}$ -gates: If an  $\mathbf{OR}$ -gate has two vulnerable  $\exists$ -gates as predecessors, one of them carrying a 1, then the  $\mathbf{OR}$ -gate will get a 1, too, regardless which (computation) path we will follow for the  $\exists$ -gate carrying a 0. Thus our simulating machine would no longer work (weakly) unambiguously. This explanation shows that  $\mathbf{UnambAPDA}$ -machines can simulate languages accepted by  $\mathbf{UnambS}^{\mathbf{EAC}}$ -circuits under the restriction, that no  $\mathbf{OR}$ -gate is least common ancestor of any two vulnerable  $\exists$ -gates. (It could be remarked here that the technique of Proposition 3 to get rid of robust  $\mathbf{OR}$ -gates is not applicable here, since we do not have negations in  $\mathbf{SAC}$ -circuits).

In the following we will relate *CREW-PRAM*'s and unambiguous *AC*-circuits.

**Theorem 7**  $\text{UnambAC}^k \subseteq \text{CREW-TIME}(\log^k n)$  for  $k \geq 1$ .

**Idea of Proof:** As in the usual evaluation of uniform circuits by PRAMs we first simulate the uniformity-machine and associate with each gate  $a$  a cell of global memory  $\text{Result}(a)$  and with each wire a processor. In addition, we use for each gate  $a$  a global memory cell  $\text{Reached}(a)$  which is initially set to 1 for all input gates, and to 0 otherwise. This can be performed by a *CREW-PRAM* since the uniformity-machine is logspace bounded and  $\text{DSPACE}(\log n) \subseteq \text{CREW-TIME}(\log n)$ . Then, as in [21], we run through a loop of length equal to the depth of the simulated circuit. For the simulation of a gate  $a$  of bounded fan-in the process associated with the wire leading to the first son of  $a$  sequentially asks the  $\text{Reached}$ -bits of all predecessors of  $a$ . As soon, as they are set, it computes  $\text{Result}(a)$  and sets  $\text{Reached}(a)$  to 1.

The simulation of an  $\exists$ -gate  $a$  of unbounded fan-in is done in the following way: As soon, as the predecessors of  $a$  are marked as reached (simultaneously, since the simulated circuit is layered!), the corresponding processors overwrite  $\text{Result}(a)$  with a 1, if their predecessor carries a 1 in its result-cell. Then the first son of  $a$  sets  $\text{Reached}(a)$  to 1.

Since the simulated circuit is unambiguous, there can be at most one successful predecessor, which gives us a *CREW*-algorithm.  $\forall$ -gates are treated in a dual way.  $\square$

This leaves open the question for showing  $\text{CREW-TIME}(\log^k n) \subseteq \text{UnambAC}^k$  in analogy to  $\text{CROW-TIME}(\log^k n) \subseteq \text{AC}^k$ . To do so, it might seem necessary to do the  $AC^0$ -computations in [21] with vulnerable gates, which should be a hard task. But by making intensive use of the logarithmic bound of the word length, it will be possible to show the converse of Theorem 7. As a preparation we begin with considering *CROW-PRAM*'s. First of all, we get as a consequence of Corollary 5:

**Corollary 8**  $\text{CROW-TIME}(\log n) \subseteq \text{UnambAC}^1$

**Proof:**  $\text{CROW-TIME}(\log n) \subseteq \text{DAuxPDA-TIME}(\text{pol})$  (by [8])  $\subseteq \text{LOG(DCFL)}$  (by [23])  $\subseteq \text{LOG(UnambCFL)} \subseteq \text{LOG(UnambAC}^1)$  (see [20])  $\subseteq \text{UnambAC}^1$  (by Corollary 5.)  $\square$

We now strengthen this result by showing:

**Theorem 9**  $\text{CROW-TIME}(\log^k n) \subseteq \text{UnambSAC}^k$  for  $k \geq 1$ .

**Proof:** The proof is based on the recognition of *CROW-TIME*( $\log n$ )-languages by deterministic auxiliary push-down automata in polynomial time due to Dymond and Ruzzo. The simulation following ideas of Fortune and Wyllie or Goldschlager begins with *DAuxPDA-TIME*( $\text{pol}$ )-computable functions *STATE*, *GLOBAL*, and *LOCAL*, stating

- i)  $\text{STATE}(p, t) = j \Leftrightarrow$  Processor  $p$  executes at step  $t$  program line  $j$ ,
- ii)  $\text{GLOBAL}(t, a) = b \Leftrightarrow$  The global memory cell  $a$  contains after step  $t$  the value  $b$ , and
- iii)  $\text{LOCAL}(p, t, a) = b \Leftrightarrow$  The memory cell  $a$  of processor  $p$  after step  $t$  contains the value  $b$

Here  $1 \leq p < P$ , where  $P$  is the number of processors, which is polynomial in  $n$ ,  $1 \leq t \leq T$ , where  $T$  is the running-time, which is  $O(\log n)$ , and  $1 \leq j \leq K$ , where  $K$  is the length of program of each processor, which is  $O(1)$ .

The values of these equations are determined recursively in the following way:

Recursion for  $\text{STATE}(p, t) = \mu$  :

We set  $\text{STATE}(p, 1) = 1$  and for  $t \geq 1$   $\text{STATE}(p, t) = j \Leftrightarrow$

*Unconditional jump:*

$(\exists_{1 \leq \mu \leq K}$  Statement  $S_\mu$  is of the form "GOTO  $S_j$ ":  $\text{STATE}(p, t - 1) = \mu$ ) OR



*Successful conditional jump* (only for  $t \geq 2$ ):

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form "IF } L_a > 0 \text{ THEN GOTO } S_j" : \right. \\ \left. STATE(p, t - 1) = \mu \text{ AND NOT LOCAL}(p, t - 2, a) = 0 \right) \text{ OR}$$

*Unsuccessful conditional jump* (only for  $t \geq 2$ ):

$$\left( \text{If statement } S_{j-1} \text{ is of the form "IF } L_a > 0 \text{ THEN GOTO } S_b" : \right. \\ \left. STATE(p, t - 1) = j - 1 \text{ AND LOCAL}(p, t - 2, a) = 0 \right) \text{ OR}$$

*Otherwise:*

$$\left( \text{If statement } S_{j-1} \text{ is not a jump : } STATE(p, t - 1) = j - 1 \right).$$

Recursion for  $GLOBAL(t, i) = j$  :

Let  $p$  be the (index of the) write-owner of global memory cell  $G_i$ .  $p$  is computable in logarithmic space according to our model of a CROW-PRAM and thus can be found out by the uniformity machine of the circuit to be constructed. We set  $GLOBAL(0, i) = j$  if  $j$  is the  $i$ -th bit of the input for  $i \leq n$ ,  $GLOBAL(0, i) = 0$  for  $i > n$ , and for  $t \geq 1$  we have  $GLOBAL(t, i) = j \Leftrightarrow$

*The content of  $G_i$  was rewritten in step  $t$ :*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } G_{L_a} = L_b" : \right. \\ \left. STATE(p, t) = \mu \text{ AND LOCAL}(p, t - 1, a) = i. \text{ AND LOCAL}(p, t - 1, b) = j \right) \text{ OR}$$

*$p$  accessed the global memory without affecting  $G_i$ :*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } G_{L_a} = L_b" : \right. \\ \left. STATE(p, t) = \mu \text{ AND NOT LOCAL}(p, t - 1, a) = i \text{ AND GLOBAL}(t - 1, i) = j \right) \text{ OR}$$

*Otherwise:*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is not of the form " } G_{L_a} = L_b" : STATE(p, t) = \mu \text{ AND GLOBAL}(t - 1, i) = j \right).$$

Recursion for  $LOCAL(p, t, i) = j$  :

The definition of  $LOCAL(p, t, i)$  is a bit more extensive. By definition we know  $LOCAL(p, 0, i) = 0$  and for  $t \geq 1$  we have  $LOCAL(p, t, i) = j \Leftrightarrow$

*Read from global memory:*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } L_i = G_{L_b}" : \right. \\ \left. STATE(p, t) = \mu \text{ AND } (\exists_{0 \leq a \leq q(n)} LOCAL(p, t - 1, b) = a \text{ AND GLOBAL}(t - 1, a) = j) \right) \text{ OR}$$

*Read from global memory without affecting  $L_i$ :*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } L_a = G_{L_b}, a \neq i : \right. \\ \left. STATE(p, t) = \mu \text{ AND LOCAL}(p, t - 1, i) = j \right) \text{ OR}$$

*Indirect local write:*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } L_{L_a} = L_b" : \right. \\ \left. STATE(p, t) = \mu \text{ AND LOCAL}(p, t - 1, a) = i \text{ AND LOCAL}(p, t - 1, b) = j \right) \text{ OR}$$

*Indirect local write without affecting  $L_i$ :*

$$\left( \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } L_{L_a} = L_b" : \right. \\ \left. STATE(p, t) = \mu \text{ AND NOT LOCAL}(p, t - 1, a) = i \text{ AND LOCAL}(p, t - 1, i) = j \right) \text{ OR}$$

*Indirect local read:*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_i = L_{L_b}":$

$STATE(p, t) = \mu \text{ AND } (\exists_{0 \leq a \leq q(n)} LOCAL(p, t-1, b) = a \text{ AND } LOCAL(p, t-1, a) = j) \text{ ) OR}$

*Indirect local read without affecting  $L_i$ :*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_a = L_{L_b}", a \neq i :$

$STATE(p, t) = \mu \text{ AND } LOCAL(p, t-1, i) = j) \text{ ) OR}$

*Binary operation:*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_i = L_a \circ L_b" : STATE(p, t) = \mu \text{ AND}$

$(\exists_{0 \leq a' \leq q(n)} \exists_{0 \leq b' \leq q(n)} j = a' \circ b' : LOCAL(p, t-1, a) = a' \text{ AND } LOCAL(p, t-1, b) = b') \text{ ) OR}$

*Binary operation not affecting  $L_i$ :*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_c = L_a \circ L_b", c \neq i :$

$STATE(p, t) = \mu \text{ AND } LOCAL(p, t-1, i) = j) \text{ ) OR}$

*PIN, LENGTH or constant assignement:*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_i = PIN" \text{ and } p = j, "L_i = LENGTH" \text{ and } n = j, \text{ or } "L_i = j" :$

$STATE(p, t) = \mu) \text{ ) OR}$

*PIN, LENGTH or constant assignement not affecting  $L_i$ :*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form } "L_a = PIN", "L_a = LENGTH" \text{ or } "L_a = j", a \neq i :$

$STATE(p, t) = \mu \text{ AND } LOCAL(p, t-1, i) = j) \text{ ) OR}$

*Otherwise:*

$(\exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is not of the form } "L_a = \dots" : STATE(p, t) = \mu \text{ AND } LOCAL(p, t-1, i) = j)$

In order to convert this structure into a circuit we will replace the predicates  $STATE(p, t) = j$ ,  $GLOBAL(t, i) = j$ , and  $LOCAL(p, t, i) = j$  by gates labelled  $\langle STATE(p, t), j \rangle$ ,  $\langle GLOBAL(t, i), j \rangle$ , and  $\langle LOCAL(p, t, i), j \rangle$ . This is possible, since we assumed a logarithmic bound on the word length and hence have both an address space and a data space of polynomial size. Contrast this with the construction of Stockmeyer and Vishkin in [21], where data and addresses were coded bitwise. Their result implies a kind of normal form, stating that *CRCW-PRAMs* with a polynomial number of processors and a polynomial time bound can be transformed into equivalent machines with a logarithmic bound on the word length. But this construction heavily depends in the use of the concurrent write feature.

According to the recursion structures of  $STATE(p, t) = j$ ,  $GLOBAL(t, i) = j$ , and  $LOCAL(p, t, i) = j$  the corresponding circuits will consist of negations, conjunctions of bounded fan-in, and disjunctions some of which are of unbounded fan-in. In this way it is obvious to translate these recursions into a layered and **DSPACE(log n)**-uniform  $AC^k$ -circuit. It is to be observed, that the outermost disjunctions are not only bounded (by  $K$ ), but are unambiguous, too, since the statement a processor is executing at a certain time is uniquely determined. Further on, the unbounded disjunctions used in the formulation of binary operations are unambiguous, since for each pair  $(a', b')$  of possible values of  $LOCAL(p, t, a)$  and  $LOCAL(p, t, b)$  exactly one is "true", i.e.: there is exactly one pair  $(a', b')$  fulfilling both  $LOCAL(p, t, a) = a'$  and  $LOCAL(p, t, b) = b'$ .

Thus, in order to provide an *UnambSAC<sup>k</sup>*-circuit the only remaining task is to get rid of negations. But this can be done, since all data and addresses are polynomially bounded; an expression  $NOT LOCAL(p, t, i) = j$  is simply translated into a disjunction of all values unequal to  $j$ :

$$NOT LOCAL(p, t, i) = j \Leftrightarrow \exists_{0 \leq b < j} LOCAL(p, t, i) = b \text{ OR } \exists_{i < b \leq q(n)} LOCAL(p, t, i) = b.$$

This construction is unambiguous, since the content of any cell of memory is uniquely determined at any time.

□

As a consequence we get:

**Corollary 10**  $\text{DCFL} \subseteq \text{UnambSAC}^1$

Extending this construction, we come to the main result of this paper:

**Theorem 11**  $\text{CREW-TIME}(\log^k n) \subseteq \text{UnambAC}^k$  for  $k \geq 1$ .

**Proof:** In order to extend the simulation of Theorem 9 to *CREW-PRAMs* we have to change the recursion structure of the equations for  $\text{GLOBAL}(t, i)$ . The processor  $p$  writing on a global memory cell no longer can be computed in advance by the uniformity machine, but now has to be determined by the circuit. This is done by the following recursion: For  $t \geq 1$  we have  $\text{GLOBAL}(t, i) = j \Leftrightarrow$

The content of  $G_i$  has been modified in step  $t$ :

$$(\exists_{1 \leq p \leq P(n)} \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } G_{L_a} = L_b \text{ "})$$

$$\text{STATE}(p, t) = \mu \text{ AND LOCAL}(p, t-1, a) = i \text{ AND LOCAL}(p, t-1, b) = j \text{ ) OR}$$

The content of  $G_i$  remained unchanged in step  $t$ :

$$\left( \text{GLOBAL}(t-1, i) = j \text{ AND NOT } (\exists_{1 \leq p \leq P(n)} \exists_{1 \leq \mu \leq K} \text{Statement } S_\mu \text{ is of the form " } G_{L_a} = L_b \text{ "}) \right.$$

$$\left. \text{STATE}(p, t) = \mu \text{ AND LOCAL}(p, t-1, a) = i \right).$$

Replacing this construction for the corresponding part used in the proof of Theorem 9 we get a layered  $\text{DSPACE}(\log n)$ -uniform  $\text{AC}^k$ -circuit. Since the simulated machine is a *CREW-PRAM*, we know there are no two processors  $p$  and  $p'$ ,  $p \neq p'$ , such that at any time  $t$   $p$  executes a statement of the form " $G_{L_a} = L_b$ ",  $p'$  executes a statement of the form " $G_{L_{a'}} = L_{b'}$ ", and  $\text{LOCAL}(p, t, a)$  coincides with  $\text{LOCAL}(p', t, a')$ . But this implies that the disjunctions over  $1 \leq p \leq P(n)$  are unambiguous, too. this proves our theorem.

□

**Corollary 12**  $\text{CREW-TIME}(\log^k n) = \text{UnambAC}^k$  for  $k \geq 1$ .

**Remark 5** Although the original method of [8] works on a *DAuxPDA* and helps us in simulating a *CREW-PRAM* by an unambiguous circuit, it does not seem possible to use it directly to show  $\text{CREW-TIME}(\log n) \subseteq \text{UnambAuxPDA-TIME}(\text{pol})$ , since we use negations (or equivalently, unambiguous  $\forall$ -gates) to treat the case, that a global memory cell remains unchanged. Otherwise we would have  $\text{CREW-TIME}(\log^k n) \subseteq \text{SAC}^k$ !

Rytter was able in [20] to recognize unambiguous context-free languages with *CREW-PRAMs* in logarithmic time, which gives us

**Corollary 13**  $\text{UnambCFL} \subseteq \text{UnambAC}^1$

## 5 Discussion

Applying the concept of unambiguity, well-known from Fomal Language Theory, to circuits provided a new characterization of *CREW-PRAMs*<sup>9</sup>. More precisely we may say, that the relation of a *CREW-PRAM* to a *CRCW-PRAM* is comparable to that of strong unambiguity to nondeterminism. That is why we introduced unambiguous circuits in a way corresponding to strong unambiguity. It is also possible to consider *weakly unambiguous circuits*, which as (strongly) unambiguous circuits are made of usual "robust" gates of bounded fan-in and of vulnerable gates of unbounded fan-in. But here we would allow vulnerable gates to be destroyed as long as this does not affect the result of the output. That means, that the undefined output of a destroyed vulnerable gate would on all following pathes finally be conjoined with the value *FALSE* or disjoined with the value *TRUE*. A corresponding "weak *CREW-PRAM*" model would allow the simultaneous write access

<sup>9</sup>Combining this new concept of an unambiguous circuit with a new type of gate called *select gate* Niepel and Rossmann were able to give in [15] a circuit-based characterization of *EREW-PRAMs*.

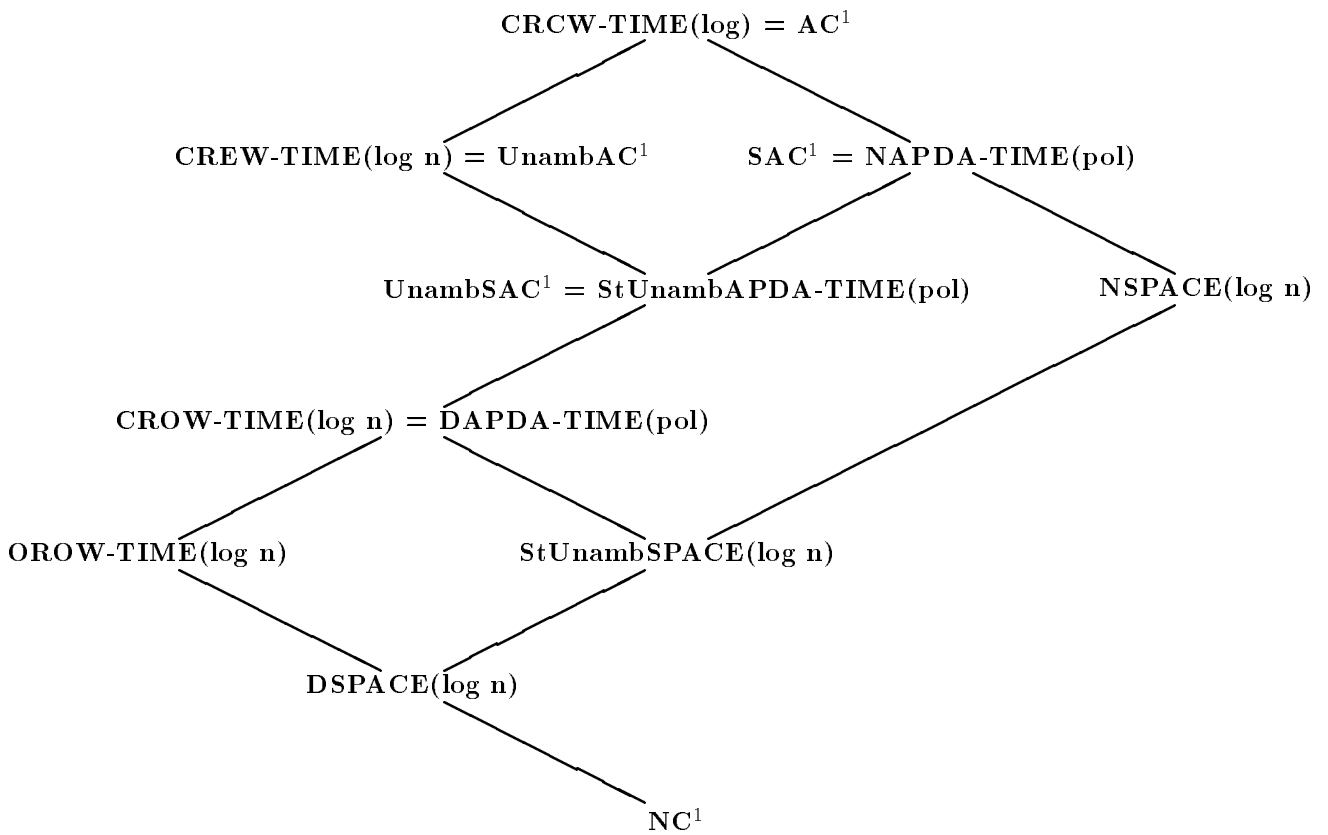


Figure 1: The  $NC$ -Structure between  $NC^1$  and  $AC^1$

to cells of global memory, which would be destroyed and made unreadable by that, but its final result must not be affected by this fact or by the content of the cell after its destruction. In [14] the equation  $SAC^k = NAuxPDA-TIME(c^{log^k n})$  could be extended to both types of unambiguities by showing both  $UnambSAC^k = StUnambAuxPDA-TIME(c^{log^k n})$  and  $Weak UnambSAC^k = UnambAuxPDA-TIME(c^{log^k n})$ . In particular this yields  $UnambCFL \subseteq UnambSAC^1$  thereby unifying and strengthening corollaries 10 and 13.

Finally, the relationships between the classes considered so far are depicted in Figure 1.

## Acknowledgements

I wish to thank Peter Rossmanith for many stimulating discussions. He and Inga Niepel pointed out the general form of Theorem 11 for arbitrary powers of the log-function. Finally, I am indebted to the anonymous referees for their very careful, comprehensive, and helpful reports.

## References

- [1] J. Balcácar, J. Díaz, and J. Gabárrro. *Structural Complexity Theory I*. Springer, 1988.
- [2] J. Balcácar, J. Díaz, and J. Gabárrro. *Structural Complexity Theory II*. Springer, 1990.
- [3] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, 1977.

- [4] G. Buntrock, B. Jenner, K.-J. Lange, and P. Rossmanith. Unambiguity and fewness for logarithmic space. In *Proc. of the 8th Conference on Fundamentals of Computation Theory*, LNCS, 1991. (to appear).
- [5] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *J. Assoc. Comp. Mach.*, 28:114–133, 1981.
- [6] S. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *J. Assoc. Comp. Mach.*, 18:4–18, 1971.
- [7] S. Cook. A taxonomy of problems with fast parallel algorithms. *Inform. and Control*, 64:2–22, 1985.
- [8] P. Dymond and W. Ruzzo. Parallel RAMs with owned global memory and deterministic context-free language recognition. In *Proc. of 13th ICALP*, number 226 in LNCS, pages 95–104. Springer, 1986.
- [9] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proc. of the 10th Annual ACM Symposium on Theory of Computing*, pages 114–118, 1978.
- [10] L. M. Goldschlager. A unified approach to models of synchronous parallel computation. In *Proc. of the 10th Annual ACM Symposium on Theory of Computing*, pages 89–94, 1978.
- [11] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Language, and Computation*. Addison-Wesley, Reading Mass., 1979.
- [12] O.H. Ibarra, T. Jiang, and B. Ravikumar. Some subclasses of context-free languages in  $NC^1$ . *Information Processing Letters*, 29:11–117, 1988.
- [13] K.-J. Lange. Unambiguity of circuits. In *Proc. of the 5th IEEE Structure in Complexity Conference*, pages 130–137, 1990.
- [14] K.-J. Lange and P. Rossmanith. Characterizing unambiguous augmented pushdown automata by circuits. In *Proc. of the 15th MFCS*, number 452 in LNCS, pages 399–406. Springer, 1990.
- [15] I. Niepel and P. Rossmanith. Uniform circuits and exclusive read PRAMs. SFB-Bericht 342/31/90 A, I9055, Institut für Informatik, Technische Universität München, December 1990.
- [16] K.R. Reischuk. *Einführung in die Komplexitätstheorie*. Teubner, Stuttgart, 1990.
- [17] P. Rossmanith. The owner concept for PRAMs. In *Proc. of the 8th STACS*, number 480 in LNCS, pages 172–183. Springer, 1991.
- [18] W. Ruzzo. Tree-size bounded alternation. *J. Comp. System Sci.*, 21:218–235, 1980.
- [19] W. Ruzzo. On uniform circuit complexity. *J. Comp. System Sci.*, 22:365–338, 1981.
- [20] W. Rytter. Parallel time  $O(\log n)$  recognition of unambiguous context-free languages. *Inform. and Control*, 73:75–86, 1987.
- [21] L. Stockmeyer and C. Vishkin. Simulation of random access machines by circuits. *SIAM J. Comp.*, 13:409–422, 1984.
- [22] I. Sudborough. A note on tape-bounded complexity classes and linear context-free languages. *J. Assoc. Comp. Mach.*, 22:499–500, 1975.
- [23] I. Sudborough. On the tape complexity of deterministic context-free languages. *J. Assoc. Comp. Mach.*, 25:405–414, 1978.
- [24] H. Venkateswaran. Properties that characterize LOGCFL. In *Proc. of 19th STOC*, pages 141–150, 1987.
- [25] K. Wagner and G. Wechsung. *Computational complexity*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1986.