Towards an implicit characterization of NC^k

Reinhard Kahle

Departamento de Matemática
Universidade de Coimbra

Joint work with:
Guillaume Bonfante, Nancy
Jean-Yves Marion, Nancy
Isabel Oitavem, Lisbon

Project

The work presented here is joint work (in progress) with

- Guillaume Bonfante, Nancy
- Jean-Yves Marion, Nancy, and
- Isabel Oitavem, Lisbon,

supported by the project

Teorias e linguagens de programação para computações com recursos limitados

within the *Programa PESSOA 2005/2006* of *GRICES - EGIDE*.

NC^k

NC^k is the class of languages accepted by uniform boolean circuit families of depth $O(\log^k n)$ and size $2^{O(\log n)}$ with bounded gates; and $NC = \bigcup_k NC^k$.

For our approach we will look to NC^k from *Alternating Turing Machines* (ATMs).

Using the following theorem (relating the original definition of NC^k in terms of circuits to ATMs) we can actually use the characterization in terms of ATMs as *definition* of NC^k:

Theorem (Ruzzo, [BO04, Theorem 4.1.(4)])

Let $k \geq 1$. For any language $L \subseteq \{0,1\}^*$, L is recognized by an ATM in $O(\log^k n)$ time and $O(\log n)$ space iff it is in (uniform) NC^k.

Rough idea

Thus, a function in NC^k needs $O(\log^k n)$ *time* and $O(\log n)$ *space* on an ATM.

The underlying intuition for the implicit characterization is:

- (Ordinary) ramified recursion captures the *time* aspect.
- Recursion with *pointers* captures the *space* aspect.

We will work in a *tiered* context using $k + 2$ tiers:

- k *tiers* for the nested recursions dealing with the *time* aspect;
- a separate tier for a recursion capturing the *space* aspect; and
- a tier for the *safe* arguments.

Remark

Recursion with Pointers is a special case of Recursion with Substitution.

Towards NC^2

Of course, NC^2 should serve as a generic case for NC^k.

According to the underlying idea, the characterization of NC^2 requires 4 tiers:

- Tier 0: Safe terms.
- Tier 1: Recursion with pointers (Space tier).
- Tier 2: Recursion without pointers (1st time tier).
- Tier 3: Recursion without pointers (2nd time tier), modifying Tier 2.

Remark

Modifying the following schemes carefully, one can use a common tier for tier 1 and tier 2; we prefer to separate them to stress the different role with respect to Space and Time.

T^2

Formally, we will work with binary trees build from 0 and 1 as use of the binary constructor $*$.

For simplicity, we will use natural numbers for the time tiers (one can think of a representation of the natural numbers by trees).

Recursion for tier 1—with pointers

$$f(;;p,0,\vec{t};\vec{w}) = g(;;p,0,\vec{t};\vec{w}),\quad f(;;p,1,\vec{t};\vec{w}) = g(;;p,1,\vec{t};\vec{w}),$$

$$f(;;p,u*v,\vec{t};\vec{w}) = h(;;;\vec{w},f(;;p*0,u,\vec{t};\vec{w}),f(;;p*1,v,\vec{t};\vec{w})).$$

Recursion for tier 2

$$f(;0,\vec{y};\vec{t};\vec{w}) = g(;\vec{y};\vec{t};\vec{w}),$$

$$f(;y+1,\vec{y};\vec{t};\vec{w}) = h(;;;\vec{w},f(;y,\vec{y};\vec{t};\vec{w})).$$

Recursion for tier 3

$$f(0,\vec{x};\vec{y};\vec{t};\vec{w}) = g(\vec{x};\vec{y};\vec{t};\vec{w}),$$

$$f(x+1,\vec{x};\vec{y};\vec{t};\vec{w}) = h(;\vec{y};;\vec{w},f(x;\vec{y};\vec{t};\vec{w})).$$

T^3

Recursion for tier 1—with pointers

$$f(;;;p,0,\vec{t};\vec{w}) = g(;;;p,0,\vec{t};\vec{w}), \quad f(;;;p,1,\vec{t};\vec{w}) = g(;;;p,1,\vec{t};\vec{w}),$$

$$f(;;;p,u*v,\vec{t};\vec{w}) = h(;;;;\vec{w}, f(;;;p*0,u,\vec{t};\vec{w}), f(;;;p*1,v,\vec{t};\vec{w})).$$

Recursion for tier 2

$$f(;;0,\vec{y};\vec{t};\vec{w}) = g(;;\vec{y};\vec{t};\vec{w}),$$

$$f(;;y+1,\vec{y};\vec{t};\vec{w}) = h(;;;;\vec{w}, f(;;y,\vec{y};\vec{t};\vec{w})).$$

Recursion for tier 3

$$f(;0,\vec{x};\vec{y};\vec{t};\vec{w}) = g(;\vec{x};\vec{y};\vec{t};\vec{w}),$$

$$f(;x+1,\vec{x};\vec{y};\vec{t};\vec{w}) = h(;;\vec{y};;\vec{w}, f(;x;\vec{y};\vec{t};\vec{w})).$$

Recursion for tier 4

$$f(0,\vec{r};\vec{x};\vec{y};\vec{t};\vec{w}) = g(\vec{r};\vec{x};\vec{y};\vec{t};\vec{w}),$$

$$f(r+1,\vec{r};\vec{x};\vec{y};\vec{t};\vec{w}) = h(;\vec{x};\vec{y};;\vec{w}, f(r;\vec{x};\vec{y};\vec{t};\vec{w}))$$

Work in progress result

So far we work forward to the following *Work in progress result*:

$$T^2 \subseteq NC^2 \subseteq T^3$$

with its generalization to

$$T^k \subseteq NC^k \subseteq T^{k+1}$$

$NC^2 \subseteq T^3$

1. We use *configuration trees*, in which all configurations of the ATM are represented as paths.
   The values at the leafs of a configuration tree at time t correspond to the values at level t in the bottom-up labeling of the computation tree.
2. The definition of the configuration tree at time 0 needs a recursion in tier 1 (the space tier).
3. We define an *update function* which updates the configuration tree from time t to $t+1$. This definition needs a recursion in tier 2.
4. Two additional nested recursions allow the iteration of this update function needed for the $O(\log^2 n)$ computation steps of the ATM.
5. We define an *eval function* to read off the result of the computation. This eval function needs a recursion, but it is not a nested one.

Remark

The idea of the proof is the same as for the proof of the lower bound in Bellantoni/Oitavem [BO04].

$T^2 \subseteq NC^2$

The idea for the proof of this inclusion is to model the computations of T^2 by circuits.

In general, these circuits will be too big. However, in a second step an ATM (equivalent to NC^2) can compute these circuits bitwise.

Remarks

- Leivant-Marion: *Linear ramified recurrence with substitution*

$$LRRS = NC^1$$

- Leivant: *Ramified schematic recurrence*

$$RSR^m \subseteq NC^m \subseteq RSR^{m+2}, \quad m \geq 2.$$

- AC^k

- T^k as a "natural" theory.

References

- S. Bellantoni and I. Oitavem. *Separating NC along the δ axis*, Theoretical Computer Science 318, 2004, pp. 57–78.
- D. Leivant. *A characterization of NC by tree recursion*, FOCS 1998. IEEE Computer Society, 1998, pp. 716–724.
- D. Leivant and J.-Y. Marion. *A characterization of alternating log time by ramified recurrence*, Theoretical Computer Science 236(1–2), 2000, pp. 192–208.