On defining integers in the counting hierarchy and proving arithmetic circuit lower bounds

Peter Bürgisser
Paderborn University, Germany

STACS 2007, February 22, 2007

Arithmetic circuit complexity

The investigation of the complexity of evaluating polynomials by arithmetic circuits (or straight-line programs) is a main topic in algebraic complexity theory.

Let the (arithmetic) complexity $L(f)$ of a polynomial $f \in K[X_1, \ldots, X_m]$ over a field K be the minimum number of arithmetic operations $+, -, *, /$ sufficient to compute f from the variables X_i and constants in K.

Arithmetic circuit complexity

The investigation of the complexity of evaluating polynomials by arithmetic circuits (or straight-line programs) is a main topic in algebraic complexity theory.

Let the (arithmetic) complexity $L(f)$ of a polynomial $f \in K[X_1, \ldots, X_m]$ over a field K be the minimum number of arithmetic operations $+, -, *, /$ sufficient to compute f from the variables X_i and constants in K.

We call a sequence $(f_n)_{n\in\mathbb{N}}$ of univariate polynomials easy to compute if $L(f_n) = (\log n)^{\mathcal{O}(1)}$, otherwise hard to compute (usually n stands for $\deg f_n$).

Arithmetic circuit complexity

The investigation of the complexity of evaluating polynomials by arithmetic circuits (or straight-line programs) is a main topic in algebraic complexity theory.

Let the (arithmetic) complexity $L(f)$ of a polynomial $f \in K[X_1, \ldots, X_m]$ over a field K be the minimum number of arithmetic operations $+, -, *, /$ sufficient to compute f from the variables X_i and constants in K.

We call a sequence $(f_n)_{n\in\mathbb{N}}$ of univariate polynomials easy to compute if $L(f_n) = (\log n)^{\mathcal{O}(1)}$, otherwise hard to compute (usually n stands for $\deg f_n$).

For example, for fixed $r \in \mathbb{N}$, the sequence $(G_n^{(r)})_{n\in\mathbb{N}}$

$$G_n^{(r)} := \sum_{k=1}^{n} k^r X^k$$

is easy to compute. Proof: $G_n^{(0)} = \frac{X^{n+1}-1}{X-1}$ -1 (geometric series), hence $L(G_n^{(0)}) = \mathcal{O}(\log n)$ using "square and multiply". Now take derivatives.

Known hard to compute families of polynomials

In a landmark paper, Strassen proved in 1974 that various sequences (f_n) of specific polynomials are hard to compute.

Using these methods, von zur Gathen and Strassen showed in 1980 that $(G_n^{(r)})$ is hard to compute if $r \in \mathbb{Q} \setminus \mathbb{Z}$.

Known hard to compute families of polynomials

In a landmark paper, Strassen proved in 1974 that various sequences (f_n) of specific polynomials are hard to compute.

Using these methods, von zur Gathen and Strassen showed in 1980 that $(G_n^{(r)})$ is hard to compute if $r \in \mathbb{Q} \setminus \mathbb{Z}$.

The complexity status of

$$G_n^{(r)} = \sum_{k=1}^{n} k^r X^k$$

for negative integers r has ever since been an outstanding open problem.

It has been conjectured that $(G_n^{(r)})$ is hard to compute in this case.

Shub and Smale's tau-conjecture

For an integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_m]$, we define the tau-complexity $\tau(f)$ as $L(f)$, but allow only the constant 1 and disallow divisions.

Clearly, $L(f) \leq \tau(f)$.

Shub and Smale's tau-conjecture

For an integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_m]$, we define the tau-complexity $\tau(f)$ as $L(f)$, but allow only the constant 1 and disallow divisions.

Clearly, $L(f) \leq \tau(f)$.

Let $z(f)$ denote the number of distinct integer roots of a univariate $f \in \mathbb{Z}[X]$.

Shub and Smale's tau-conjecture claims that there exists $c > 0$ such that for all $f \in \mathbb{Z}[X]$:

$$z(f) \leq (1 + \tau(f))^c.$$

Shub and Smale's tau-conjecture

For an integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_m]$, we define the tau-complexity $\tau(f)$ as $L(f)$, but allow only the constant 1 and disallow divisions.

Clearly, $L(f) \leq \tau(f)$.

Let $z(f)$ denote the number of distinct integer roots of a univariate $f \in \mathbb{Z}[X]$.

Shub and Smale's tau-conjecture claims that there exists $c > 0$ such that for all $f \in \mathbb{Z}[X]$:

$$z(f) \leq (1 + \tau(f))^c.$$

Shub and Smale proved in 1994 that the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$ in the Blum-Shub-Smale model over $\mathbb{C}$.

Shub and Smale's tau-conjecture

For an integer polynomial $f \in \mathbb{Z}[X_1, \ldots, X_m]$, we define the tau-complexity $\tau(f)$ as $L(f)$, but allow only the constant 1 and disallow divisions.

Clearly, $L(f) \leq \tau(f)$.

Let $z(f)$ denote the number of distinct integer roots of a univariate $f \in \mathbb{Z}[X]$.

Shub and Smale's tau-conjecture claims that there exists $c > 0$ such that for all $f \in \mathbb{Z}[X]$:

$$z(f) \leq (1 + \tau(f))^c.$$

Shub and Smale proved in 1994 that the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$ in the Blum-Shub-Smale model over $\mathbb{C}$.

Resolving the tau-conjecture appears as the fourth problem in Smale's list (2000) of the most important problems for the mathematicians in the 21st century.

Complexity of factorials

Shub and Smale: The truth of the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$.

Complexity of factorials

Shub and Smale: The truth of the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$.

In fact, in order to show $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$, it suffices to prove that

for all nonzero integers m_n, the sequence $(m_n n!)_{n\in\mathbb{N}}$ is hard to compute.

We say that a sequence $(a(n))$ of integers is hard to compute iff $\tau(a(n))$ is not polynomially bounded in $\log n$.

Complexity of factorials

Shub and Smale: The truth of the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$.

In fact, in order to show $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$, it suffices to prove that

for all nonzero integers m_n, the sequence $(m_n n!)_{n\in\mathbb{N}}$ is hard to compute.

We say that a sequence $(a(n))$ of integers is hard to compute iff $\tau(a(n))$ is not polynomially bounded in $\log n$.

It is plausible that $(n!)$ is hard to compute, otherwise factoring integers could be done in (nonuniform) polynomial time (Strassen 1976).

Complexity of factorials

Shub and Smale: The truth of the tau-conjecture implies $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$.

In fact, in order to show $\mathsf{P}_{\mathbb{C}} \neq \mathsf{NP}_{\mathbb{C}}$, it suffices to prove that

for all nonzero integers m_n, the sequence $(m_n n!)_{n\in\mathbb{N}}$ is hard to compute.

We say that a sequence $(a(n))$ of integers is hard to compute iff $\tau(a(n))$ is not polynomially bounded in $\log n$.

It is plausible that $(n!)$ is hard to compute, otherwise factoring integers could be done in (nonuniform) polynomial time (Strassen 1976).

Lipton strengthened this implication in 1994 by showing that if factoring integers is "hard on average" (a common assumption in cryptography), then a somewhat weaker version of the tau-conjecture follows.

Arithmetic complexity of the permanent

Valiant proposed in a seminal paper (1979) an algebraic version of the P versus NP problem for explaining the hardness of computing the permanent.

He defined the classes VP of polynomially computable and VNP of polynomially definable families of multivariate polynomials over a fixed field K and proved that the family (PER_n) of permanent polynomials is VNP-complete (if $\mathrm{char}K \neq 2$).

Arithmetic complexity of the permanent

Valiant proposed in a seminal paper (1979) an algebraic version of the P versus NP problem for explaining the hardness of computing the permanent.

He defined the classes VP of polynomially computable and VNP of polynomially definable families of multivariate polynomials over a fixed field K and proved that the family (PER_n) of permanent polynomials is VNP-complete (if $\mathrm{char}K \neq 2$).

Recall that the permanent of the matrix $[X_{ij}]_{1\leq i,j\leq n}$ is defined as

$$\mathrm{PER}_n = \sum_{\pi\in S_n} X_{1\pi(1)}\cdots X_{n\pi(n)}.$$

Arithmetic complexity of the permanent

Valiant proposed in a seminal paper (1979) an algebraic version of the P versus NP problem for explaining the hardness of computing the permanent.

He defined the classes VP of polynomially computable and VNP of polynomially definable families of multivariate polynomials over a fixed field K and proved that the family (PER_n) of permanent polynomials is VNP-complete (if $\mathrm{char}K \neq 2$).

Recall that the permanent of the matrix $[X_{ij}]_{1\leq i,j\leq n}$ is defined as

$$\mathrm{PER}_n = \sum_{\pi\in S_n} X_{1\pi(1)}\cdots X_{n\pi(n)}.$$

Valiant's completeness result implies that

$$\mathrm{VP} \neq \mathrm{VNP} \Longleftrightarrow (\mathrm{PER}_n) \notin \mathrm{VP} \Longleftrightarrow L(\mathrm{PER}_n) \text{ is not polynomially bounded in } n.$$

This is called Valiant's hypothesis and considered a major conjecture in algebraic complexity theory.

Main result

We have encountered several notoriously difficult questions in algebraic complexity.

Our main result related them to Valiant's hypothesis and thus confirms the belief that solving any of these problems is indeed very hard.

Main result

We have encountered several notoriously difficult questions in algebraic complexity.

Our main result related them to Valiant's hypothesis and thus confirms the belief that solving any of these problems is indeed very hard.

Theorem 1.1. *Each of the statements listed below implies that the permanent of n by n matrices cannot be computed by constant-free and division-free arithmetic circuits of size polynomial in n: that is, $\tau(\mathrm{PER}_n)$ is not polynomially bounded in n.*

1. *The sequence of factorials $(n!)_{n\in\mathbb{N}}$ is hard to compute.*
2. *The tau-conjecture of Shub and Smale is true.*
3. *The sequence $(G_n^{(r)}) = (\sum_{k=1}^{n} k^r T^k)_{n\in\mathbb{N}}$ for a fixed negative integer r is hard to compute.*

Main result

We have encountered several notoriously difficult questions in algebraic complexity.

Our main result related them to Valiant's hypothesis and thus confirms the belief that solving any of these problems is indeed very hard.

Theorem 1.1. *Each of the statements listed below implies that the permanent of n by n matrices cannot be computed by constant-free and division-free arithmetic circuits of size polynomial in n: that is, $\tau(\mathrm{PER}_n)$ is not polynomially bounded in n.*

1. *The sequence of factorials $(n!)_{n\in\mathbb{N}}$ is hard to compute.*
2. *The tau-conjecture of Shub and Smale is true.*
3. *The sequence $(G_n^{(r)}) = (\sum_{k=1}^{n} k^r T^k)_{n\in\mathbb{N}}$ for a fixed negative integer r is hard to compute.*

Koiran 2004: weaker version of statement (1).

Counting hierarchy

The counting hierarchy introduced by K. Wagner in 1986 is a complexity class lying between PP and PSPACE that bears more or less the same relationship to #P as the polynomial hierarchy bears to NP.

Counting hierarchy

The counting hierarchy introduced by K. Wagner in 1986 is a complexity class lying between PP and PSPACE that bears more or less the same relationship to #P as the polynomial hierarchy bears to NP.

Def 1.2. *Let K be a complexity class. We define $\mathbf{C} \cdot K$ to be the set of all languages A such that there exist a language $B \in K$, a polynomial p, and a polynomial time computable function $f : \{0,1\}^* \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$:*

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid \langle x, y \rangle \in B\}| > f(x).$$

Counting hierarchy

The counting hierarchy introduced by K. Wagner in 1986 is a complexity class lying between PP and PSPACE that bears more or less the same relationship to #P as the polynomial hierarchy bears to NP.

Def 1.2. *Let K be a complexity class. We define $\mathbf{C} \cdot K$ to be the set of all languages A such that there exist a language $B \in K$, a polynomial p, and a polynomial time computable function $f : \{0,1\}^* \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$:*

$$x \in A \iff |\{y \in \{0,1\}^{p(|x|)} \mid \langle x, y \rangle \in B\}| > f(x).$$

Def 1.3. *The k-th level $\mathsf{C_kP}$ of the counting hierarchy is recursively defined by $\mathsf{C_0P} := \mathsf{P}$ and $\mathsf{C_{k+1}P} := \mathbf{C} \cdot \mathsf{C_kP}$ for $k \in \mathbb{N}$. One defines CH as the union of all classes $\mathsf{C_kP}$.*

Integers definable in the counting hierarchy

We consider sequences of integers $a(n)$ of polynomial bitsize, i.e., $\log |a(n)| \leq n^{\mathcal{O}(1)}$.

Example: Factorials $a(n) = n!$

Integers definable in the counting hierarchy

We consider sequences of integers $a(n)$ of polynomial bitsize, i.e., $\log |a(n)| \leq n^{\mathcal{O}(1)}$.

Example: Factorials $a(n) = n!$

We assign to a sequence $a = (a(n))$ of polynomial bitsize the following languages, where the integers n, j represented in binary (using $\mathcal{O}(\log n)$ bits):

$$\begin{aligned} \mathrm{Sgn}(a) &:= \{n \mid a(n) \geq 0\} \\ \mathrm{Bit}(|a|) &:= \{(n,j) \mid \text{ the } j\text{-th bit of } |a(n)| \text{ equals } 1\ \}. \end{aligned}$$

Def 1.4. *A sequence a of integers of polynomial bitsize is called definable in* CH *iff* $\mathrm{Sgn}(a) \in \mathsf{CH}$ *and* $\mathrm{Bit}(|a|) \in \mathsf{CH}$.

Outline of proof of main result

Prop 1.5. *Consider a sequence* $(a(n))_{n\in\mathbb{N}}$ *of integers definable in* CH. *If* $\tau(\mathrm{PER}_n) = n^{\mathcal{O}(1)}$, *then* $\tau(a(n)) = (\log n)^{\mathcal{O}(1)}$.

Outline of proof of main result

Prop 1.5. *Consider a sequence* $(a(n))_{n\in\mathbb{N}}$ *of integers definable in* CH. *If* $\tau(\mathrm{PER}_n) = n^{\mathcal{O}(1)}$, *then* $\tau(a(n)) = (\log n)^{\mathcal{O}(1)}$.

Proof. 1. $\tau(\mathrm{PER}_n) = n^{\mathcal{O}(1)}$ implies PP $\subseteq$ P/poly. This implies CH $\subseteq$ P/poly.

Outline of proof of main result

Prop 1.5. *Consider a sequence* $(a(n))_{n\in\mathbb{N}}$ *of integers definable in* CH. *If* $\tau(\text{PER}_n) = n^{\mathcal{O}(1)}$, *then* $\tau(a(n)) = (\log n)^{\mathcal{O}(1)}$.

Proof. 1. $\tau(\text{PER}_n) = n^{\mathcal{O}(1)}$ implies PP $\subseteq$ P/poly. This implies CH $\subseteq$ P/poly.

2. Let $a(n) = \sum_{j=0}^{p(n)} a(n,j)2^j$ be the binary representation of $a(n)$, where p is a polynomial. By assumption, we can decide $a(n,j) = b$ in CH, where n, j are given in binary. Hence we can decide $a(n,j) = b$ in P/poly.

Outline of proof of main result

Prop 1.5. *Consider a sequence* $(a(n))_{n\in\mathbb{N}}$ *of integers definable in* CH. *If* $\tau(\text{PER}_n) = n^{\mathcal{O}(1)}$, *then* $\tau(a(n)) = (\log n)^{\mathcal{O}(1)}$.

Proof. 1. $\tau(\text{PER}_n) = n^{\mathcal{O}(1)}$ implies PP $\subseteq$ P/poly. This implies CH $\subseteq$ P/poly.

2. Let $a(n) = \sum_{j=0}^{p(n)} a(n,j)2^j$ be the binary representation of $a(n)$, where p is a polynomial. By assumption, we can decide $a(n,j) = b$ in CH, where n, j are given in binary. Hence we can decide $a(n,j) = b$ in P/poly.

3. Consider the polynomial

$$A_n(Y_1, \ldots, Y_{\ell(n)}) = \sum_{j=0}^{p(n)} a(n,j) Y_1^{j_1} \cdots Y_{\ell(n)}^{j_{\ell(n)}},$$

where $\ell(n) = 1 + \lfloor \log p(n) \rfloor$ and j_i denotes the bit of j of weight 2^{i-1}. Note that

$$A_n(2^{2^0}, 2^{2^1}, \ldots, 2^{2^{\ell(n)-1}}) = a(n)$$

4. By the down scaled Valiant criterion due to Koiran (2004) there is a family $(G_r(Y_1,\ldots,Y_r,N_1,\ldots,N_r,P_1,\ldots,P_r))$ in VNP^0 that satisfies for all n

$$A_n(Y_1,\ldots,Y_{\ell(n)}) = G_{\ell(n)}(Y_1,\ldots,Y_{\ell(n)},n_1,\ldots,n_{\ell(n)},p_1,\ldots,p_{\ell(n)}),$$

where n_i and p_i denote the bits of n and $p(n)$ of weight 2^{i-1}, respectively. Hereby VNP^0 denotes the constant-free version of the class VNP (Malod 2003).

4. By the down scaled Valiant criterion due to Koiran (2004) there is a family $(G_r(Y_1, \ldots, Y_r, N_1, \ldots, N_r, P_1, \ldots, P_r))$ in VNP^0 that satisfies for all n

$$A_n(Y_1, \ldots, Y_{\ell(n)}) = G_{\ell(n)}(Y_1, \ldots, Y_{\ell(n)}, n_1, \ldots, n_{\ell(n)}, p_1, \ldots, p_{\ell(n)}),$$

where n_i and p_i denote the bits of n and $p(n)$ of weight 2^{i-1}, respectively. Hereby VNP^0 denotes the constant-free version of the class VNP (Malod 2003).

4. By a modification of Valiant's completeness proof, assuming $\tau(\textsc{Per}_n) = n^{\mathcal{O}(1)}$: There exists a poly bounded $s(r) \in \mathbb{N}$ such that $\tau(2^{s(r)} G_r) = r^{\mathcal{O}(1)}$.

This implies $\tau(2^{e(n)} G_{\ell(n)}) = (\log n)^{\mathcal{O}(1)}$, where $e(n) = s(\ell(n)) = (\log n)^{\mathcal{O}(1)}$.

4. By the down scaled Valiant criterion due to Koiran (2004) there is a family $(G_r(Y_1,\ldots,Y_r,N_1,\ldots,N_r,P_1,\ldots,P_r))$ in VNP^0 that satisfies for all n

$$A_n(Y_1,\ldots,Y_{\ell(n)}) = G_{\ell(n)}(Y_1,\ldots,Y_{\ell(n)},n_1,\ldots,n_{\ell(n)},p_1,\ldots,p_{\ell(n)}),$$

where n_i and p_i denote the bits of n and $p(n)$ of weight 2^{i-1}, respectively. Hereby VNP^0 denotes the constant-free version of the class VNP (Malod 2003).

4. By a modification of Valiant's completeness proof, assuming $\tau(\mathrm{PER}_n) = n^{\mathcal{O}(1)}$: There exists a poly bounded $s(r) \in \mathbb{N}$ such that $\tau(2^{s(r)}G_r) = r^{\mathcal{O}(1)}$.

This implies $\tau(2^{e(n)}G_{\ell(n)}) = (\log n)^{\mathcal{O}(1)}$, where $e(n) = s(\ell(n)) = (\log n)^{\mathcal{O}(1)}$.

We conclude from the above that

$$2^{e(n)}a(n) = 2^{e(n)}G_{\ell(n)}(2^{2^0},2^{2^1},\ldots,2^{2^{\ell(n)-1}},n_1,\ldots,n_{\ell(n)},p_1,\ldots,p_{\ell(n)}),$$

hence $\tau(2^{e(n)}a(n)) \leq \tau(2^{e(n)}G_{\ell(n)}) + \ell(n) \leq (\log n)^{\mathcal{O}(1)}$.

4. By the down scaled Valiant criterion due to Koiran (2004) there is a family $(G_r(Y_1,\ldots,Y_r,N_1,\ldots,N_r,P_1,\ldots,P_r))$ in VNP^0 that satisfies for all n

$$A_n(Y_1,\ldots,Y_{\ell(n)}) = G_{\ell(n)}(Y_1,\ldots,Y_{\ell(n)},n_1,\ldots,n_{\ell(n)},p_1,\ldots,p_{\ell(n)}),$$

where n_i and p_i denote the bits of n and $p(n)$ of weight 2^{i-1}, respectively. Hereby VNP^0 denotes the constant-free version of the class VNP (Malod 2003).

4. By a modification of Valiant's completeness proof, assuming $\tau(\textsc{Per}_n) = n^{\mathcal{O}(1)}$: There exists a poly bounded $s(r) \in \mathbb{N}$ such that $\tau(2^{s(r)}G_r) = r^{\mathcal{O}(1)}$.

This implies $\tau(2^{e(n)}G_{\ell(n)}) = (\log n)^{\mathcal{O}(1)}$, where $e(n) = s(\ell(n)) = (\log n)^{\mathcal{O}(1)}$.

We conclude from the above that

$$2^{e(n)}a(n) = 2^{e(n)}G_{\ell(n)}(2^{2^0},2^{2^1},\ldots,2^{2^{\ell(n)-1}},n_1,\ldots,n_{\ell(n)},p_1,\ldots,p_{\ell(n)}),$$

hence $\tau(2^{e(n)}a(n)) \leq \tau(2^{e(n)}G_{\ell(n)}) + \ell(n) \leq (\log n)^{\mathcal{O}(1)}$.

It is possible to deduce $\tau(a(n)) = (\log n)^{\mathcal{O}(1)}$. □

Factorials are definable in CH

Corollary 1.6. *The sequence of factorials* $(n!)$ *is definable in* CH.

Factorials are definable in CH

Corollary 1.6. *The sequence of factorials* $(n!)$ *is definable in* CH.

This follows immediately from the following closure properties of sequences of integers definable in CH w.r.t. iterated addition and iterated multiplication

Theorem 1.7. *1. Suppose* $a = (a(n,k))_{n\in\mathbb{N},k\leq q(n)}$ *is definable in* CH*, where* q *is polynomially bounded. Consider*

$$b(n) := \sum_{k=0}^{q(n)} a(n,k), \quad d(n) := \prod_{k=0}^{q(n)} a(n,k).$$

Then $b = (b(n))$ *and* $d = (d(n))$ *are both definable in* CH.

Ingredients for proof of closure properties

Ingredients for proof of closure properties

The counting hierarchy is closely tied to the theory of threshold circuits of constant depth.

Ingredients for proof of closure properties

The counting hierarchy is closely tied to the theory of threshold circuits of constant depth.

Beame et al. (1986) presented parallel NC^1-algorithms for iterated multiplication and division of integers. Crucial idea: Chinese remaindering.

Ingredients for proof of closure properties

The counting hierarchy is closely tied to the theory of threshold circuits of constant depth.

Beame et al. (1986) presented parallel NC^1-algorithms for iterated multiplication and division of integers. Crucial idea: Chinese remaindering.

Reif and Tate (1992) observed that these algorithms can also be implemented by constant depth threshold circuits, placing these problems in the class TC^0.

Ingredients for proof of closure properties

The counting hierarchy is closely tied to the theory of threshold circuits of constant depth.

Beame et al. (1986) presented parallel NC^1-algorithms for iterated multiplication and division of integers. Crucial idea: Chinese remaindering.

Reif and Tate (1992) observed that these algorithms can also be implemented by constant depth threshold circuits, placing these problems in the class TC^0.

The question of the degree of uniformity required for these circuits was only recently solved in a satisfactory way by Hesse et al. (2002), who showed that there are Dlogtime-uniform circuits performing these tasks. This result, scaled up to the counting hierarchy, is crucial for our study of sequences of integers definable in the counting hierarchy!

Ingredients for proof of closure properties

The counting hierarchy is closely tied to the theory of threshold circuits of constant depth.

Beame et al. (1986) presented parallel NC^1-algorithms for iterated multiplication and division of integers. Crucial idea: Chinese remaindering.

Reif and Tate (1992) observed that these algorithms can also be implemented by constant depth threshold circuits, placing these problems in the class TC^0.

The question of the degree of uniformity required for these circuits was only recently solved in a satisfactory way by Hesse et al. (2002), who showed that there are Dlogtime-uniform circuits performing these tasks. This result, scaled up to the counting hierarchy, is crucial for our study of sequences of integers definable in the counting hierarchy!

Even though the statement of the main Theorem involves only arithmetic circuits, its proof relies on uniformity arguments thus requiring the model of Turing machines!

THANK YOU!