

Complexity Classes Arising from Circuits over Finite Algebraic Structures

PIOTR KAWAŁEK & JACEK KRZACZKOWSKI

TU Wien

Maria Curie-Skłodowska
University, Lublin



European Research Council
Established by the European Commission

(ERC, SyG POCOCOP, 101071674).

Constant memory: can I count on you?

$$\text{MAJORITY}(b_1, \dots, b_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n b_i > \frac{n}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Constant memory: can I count on you?

$$\text{MAJORITY}(b_1, \dots, b_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n b_i > \frac{n}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm: scan the input with a counter...

Memory: $O(\log n)$

Constant memory: can I count on you?

$$\text{MAJORITY}(b_1, \dots, b_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^n b_i > \frac{n}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm: scan the input with a counter...

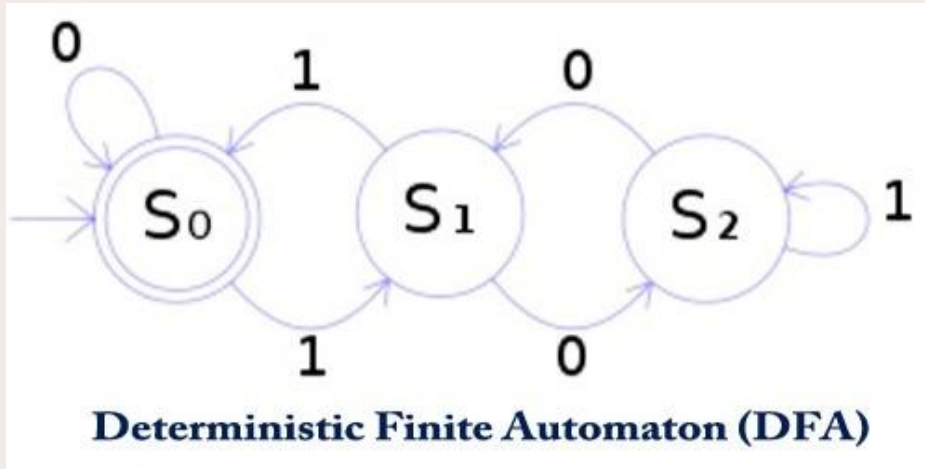
Memory: $O(\log n)$

Barrington '86:

You don't need a counter...

Constant memory is enough!

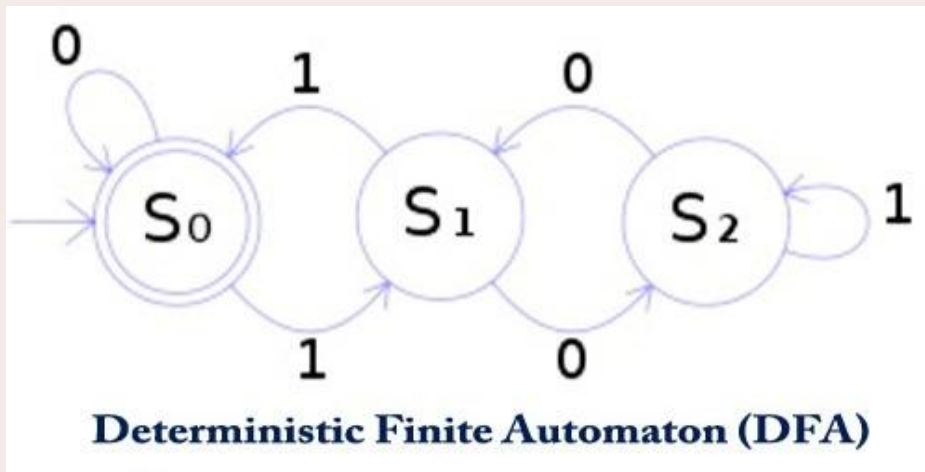
Barrington's Theorem and non-uniform DFAs



Barrington's Theorem and non-uniform DFAs

DFAs cannot count...

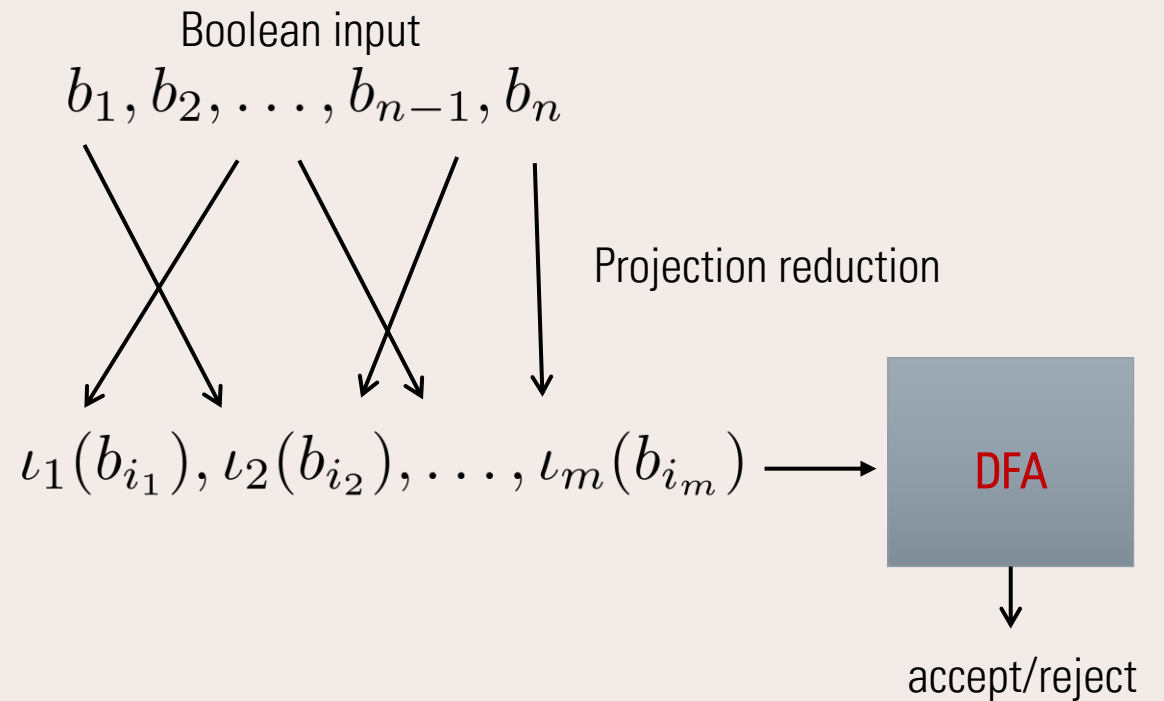
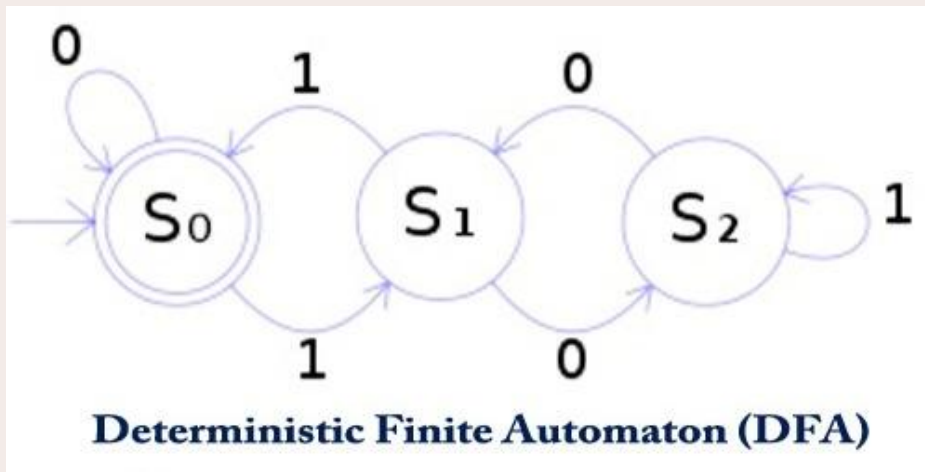
Example: $L = \{a^n b^n\}$



Barrington's Theorem and non-uniform DFAs

DFAs cannot count...

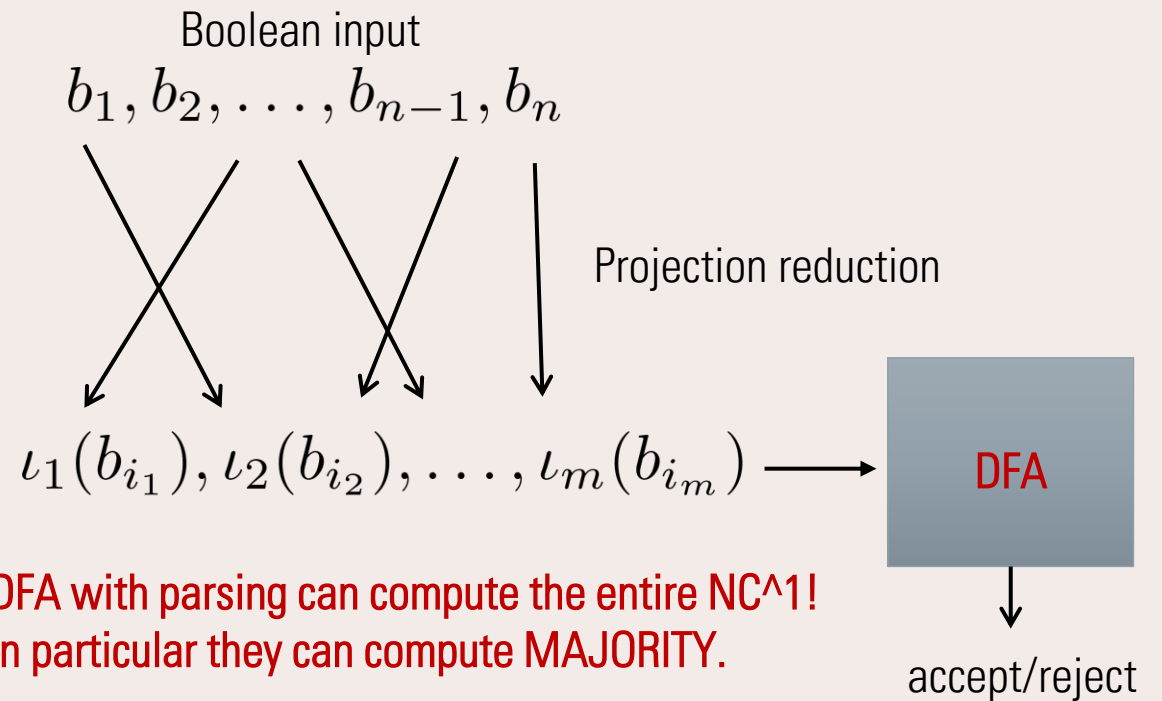
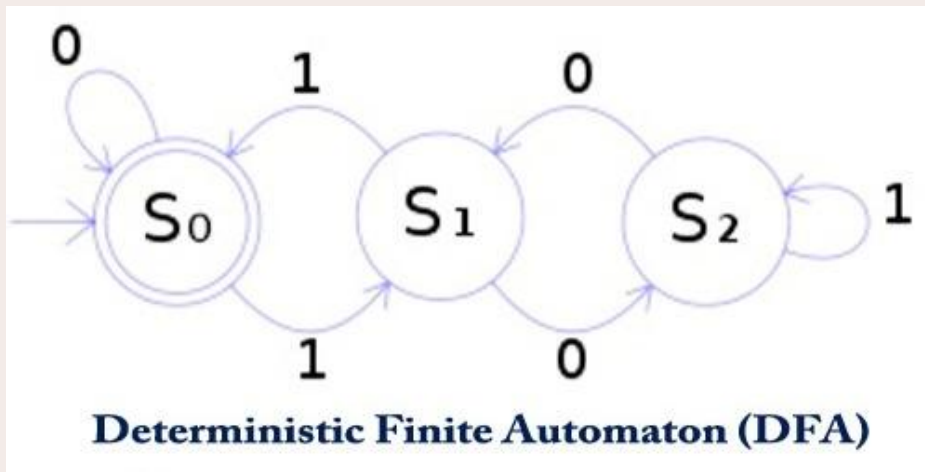
Example: $L = \{a^n b^n\}$



Barrington's Theorem and non-uniform DFAs

DFAs cannot count...

Example: $L = \{a^n b^n\}$



Automata and monoids...

nuDFA = DFA + parsing of the input

Equivalent definition of nuDFA connects them to monoids...

Automata and monoids...

nuDFA = DFA + parsing of the input

Equivalent definition of nuDFA connects them to monoids...

Program over monoid M is
any expression of the form:

$$\iota_1(b_{i_1}) \cdot \iota_2(b_{i_2}) \cdot \dots \cdot \iota_m(b_{i_m}) \in S$$

where each $\iota_j : \{0, 1\} \longrightarrow M$
and $S \subseteq M$

Automata and monoids...

nuDFA = DFA + parsing of the input

Equivalent definition of nuDFA connects them to monoids...

Program over monoid M is
any expression of the form:

$$\iota_1(b_{i_1}) \cdot \iota_2(b_{i_2}) \cdot \dots \cdot \iota_m(b_{i_m}) \in S$$

where each $\iota_j : \{0, 1\} \longrightarrow M$
and $S \subseteq M$

Word $b_1, b_2, \dots, b_{n-1}, b_n$

Is accepted by such program/expression
iff it satisfies it.

Automata and monoids...

nuDFA = DFA + parsing of the input

Equivalent definition of nuDFA connects them to monoids...

Program over monoid M is
any expression of the form:

$$\iota_1(b_{i_1}) \cdot \iota_2(b_{i_2}) \cdot \dots \cdot \iota_m(b_{i_m}) \in S$$

where each $\iota_j : \{0, 1\} \longrightarrow M$
and $S \subseteq M$

Word $b_1, b_2, \dots, b_{n-1}, b_n$

Is accepted by such program/expression
iff it satisfies it.

nuDFA over monoid M is just a
sequence of polynomial-size
programs/expressions, one for
each input length.

**Every DFA has an associated
transition monoid, so the two
definitions are equivalent.**

Short expressions over Monoids

Barrington STOC'86

Polynomial size formulas over the group S_5 can solve every problem in NC^1 .

Short expressions over Monoids

Barrington STOC'86

Polynomial size formulas over the group S_5 can solve every problem in NC^1 .

Equivalently: nuDFA over S_5 can solve every problem in NC^1 .

Short expressions over Monoids

Barrington STOC'86

Polynomial size formulas over the group S_5 can solve every problem in NC^1 .

Equivalently: nuDFA over S_5 can solve every problem in NC^1 .

Barrington, Straubing, Therien'90:

Formulas over any non-solvable monoid solve exactly problems in NC^1 .

Short expressions over Monoids

Barrington STOC'86

Polynomial size formulas over the group S_5 can solve every problem in NC^1 .

Equivalently: nuDFA over S_5 can solve every problem in NC^1 .

Barrington, Straubing, Therien'90:

Formulas over any non-solvable monoid solve exactly problems in NC^1 .

Idea: use commutator expressions to interpret Boolean AND, OR, $\neg$.

What about formulas over solvable groups?

Solvable group

Syntactic definition:

some commutator

expression trivializes.

What about formulas over solvable groups?

Solvable group

Syntactic definition:

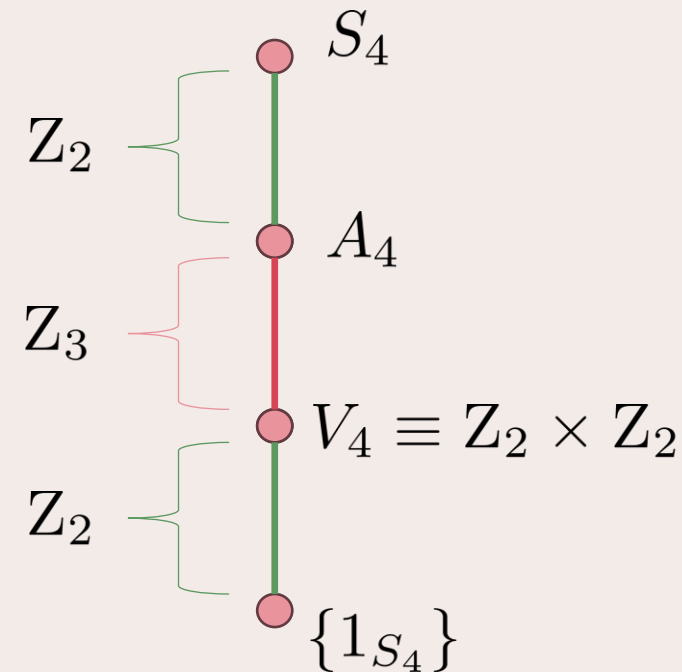
some commutator
expression trivializes.

Semantic definition:

we can divide the group
into abelian slices.

Example: S_4, A_4

Normal subgroups of S_4

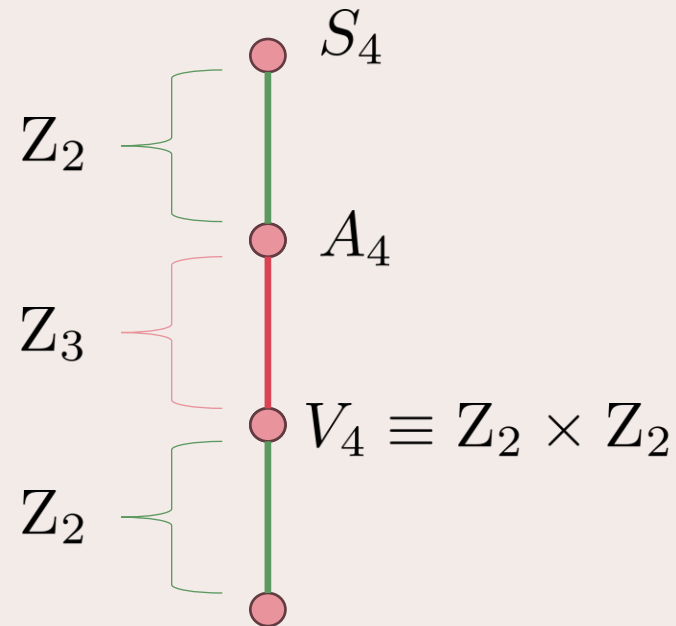


Modulo counting inside groups

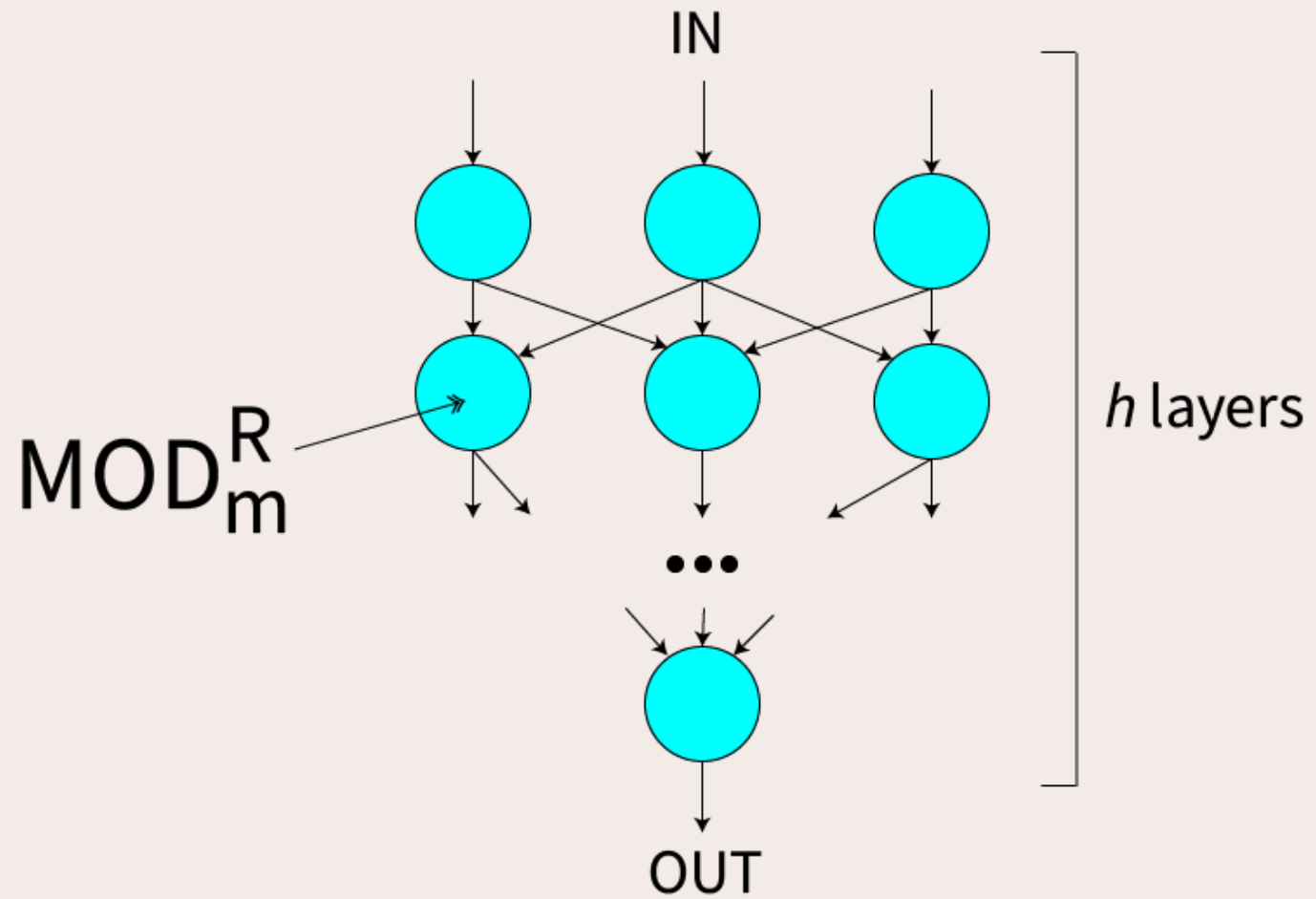
The computing power of nuDFA over the group S_4 is captured by polynomial size circuits of the form

$$\text{MOD}_2 \circ \text{MOD}_3 \circ \text{MOD}_2$$

Normal subgroups of



CC circuits



Various complexity classes

Monoid structure	Corresponding complexity class
Non-solvable	NC^1
Solvable group	CC^0
Solvable monoid	ACC^0
Aperiodic monoids	AC^0

Various complexity classes

Monoid structure	Corresponding complexity class
Non-solvable	NC^1
Solvable group	CC^0
Solvable monoid	ACC^0
Aperiodic monoids	AC^0

NC^1 = logarithmic depth circuits

CC^0 = constant-depth polynomial-size circuits with modulo counting gates

ACC^0 = constant-depth polynomial-size circuits with modulo counting gates + standard AND, OR, $\neg$ gates

AC^0 = constant-depth polynomial-size circuits with standard AND, OR, $\neg$ gates only

Various complexity classes

Monoid structure	Corresponding complexity class
Non-solvable	NC^1
Solvable group	CC^0
Solvable monoid	ACC^0
Aperiodic monoids	AC^0

$$CC^0 \not\subseteq AC^0$$

$$AC^0 \not\subseteq CC^0?$$

$$CC^0 \not\subseteq ACC^0?$$

Is modulo counting more powerful than Boolean operations?

NC^1 = logarithmic depth circuits

CC^0 = constant-depth polynomial-size circuits with modulo counting gates

ACC^0 = constant-depth polynomial-size circuits with modulo counting gates + standard AND, OR, $\neg$ gates

AC^0 = constant-depth polynomial-size circuits with standard AND, OR, $\neg$ gates only

Is there any life outside monoids?



*	a	b	c
a	a	b	c
b	b	a	b
c	c	b	c

Monoids



Magmas



Rings



Lattices

Computing power of arbitrary expressions

Algebraic structure/algebra **A** is a tuple $(A, o_1, o_2, \dots, o_k)$

- A – finite set
- $o_1, o_2, \dots, o_k$ – list of operations of arbitrary arity

What is the power of formulas that use only operations and constants from this algebraic structure?

Computing power of arbitrary expressions

Algebraic structure/algebra **A** is a tuple $(A, o_1, o_2, \dots, o_k)$

- A – finite set
- $o_1, o_2, \dots, o_k$ – list of operations of arbitrary arity

What is the power of formulas that use only operations and constants from this algebraic structure?

Example:

$$A = \{\{0, 1\}, \wedge, \vee\}$$

Can we solve the CLIQUE problem with polynomial size monotone expressions?

Computing power of arbitrary expressions

Algebraic structure/algebra **A** is a tuple $(A, o_1, o_2, \dots, o_k)$

- A – finite set
- $o_1, o_2, \dots, o_k$ – list of operations of arbitrary arity

What is the power of formulas that use only operations and constants from this algebraic structure?

Example:

$$A = \{\{0, 1\}, \wedge, \vee\}$$

Razborov'84: No

Can we solve the CLIQUE problem with polynomial size monotone expressions?

Computing power of arbitrary expressions

Algebraic structure/algebra **A** is a tuple $(A, o_1, o_2, \dots, o_k)$

- A – finite set
- $o_1, o_2, \dots, o_k$ – list of operations of arbitrary arity

What is the power of formulas that use only operations and constants from this algebraic structure?

Example:

$$A = \{\{0, 1\}, \wedge, \vee\}$$

Can we solve the CLIQUE problem with polynomial size monotone expressions?

Razborov'84: No

Raz, Wigderson'92: UCON requires at least quasipolynomial size expressions.

Circuits are better!

We would like to say: If algebraic structure **A** has certain algebraic property, then polynomial-size expressions over **A** can/cannot solve some problem. **But there is an issue...**

Circuits are better!

We would like to say: If algebraic structure **A** has certain algebraic property, then polynomial-size expressions over **A** can/cannot solve some problem. **But there is an issue...**

$$\mathbf{A} = (A_4, \circ)$$

Cannot compute AND_n
with expressions of polynomial size.

Circuits are better!

We would like to say: If algebraic structure $\mathbf{A}$ has certain algebraic property, then polynomial-size expressions over $\mathbf{A}$ can/cannot solve some problem. **But there is an issue...**

$$\mathbf{A} = (A_4, \circ)$$

Cannot compute AND_n
with expressions of polynomial size.

$$\mathbf{A} = (A_4, \circ, [x, y] = xyx^{-1}y^{-1})$$

Can compute AND_n
with expressions of polynomial size.

Circuits are better!

We would like to say: If algebraic structure $\mathbf{A}$ has certain algebraic property, then polynomial-size expressions over $\mathbf{A}$ can/cannot solve some problem. **But there is an issue...**

$$\mathbf{A} = (A_4, \circ)$$

Cannot compute AND_n
with expressions of polynomial size.

$$\mathbf{A} = (A_4, \circ, [x, y] = xyx^{-1}y^{-1})$$

Can compute AND_n
with expressions of polynomial size.

Why it matters?

Expressions over solvable algebras



ACC

Circuits are better!

We would like to say: If algebraic structure $\mathbf{A}$ has certain algebraic property, then polynomial-size expressions over $\mathbf{A}$ can/cannot solve some problem. **But there is an issue...**

$$\mathbf{A} = (A_4, \circ)$$

Cannot compute AND_n
with expressions of polynomial size.

$$\mathbf{A} = (A_4, \circ, [x, y] = xyx^{-1}y^{-1})$$

Can compute AND_n
with expressions of polynomial size.

Why it matters?

Expressions over solvable algebras



ACC

Solvability in Universal Algebra does not
depend on the signature!

Circuits are better!

We would like to say: If algebraic structure $\mathbf{A}$ has certain algebraic property, then polynomial-size expressions over $\mathbf{A}$ can/cannot solve some problem. **But there is an issue...**

$$\mathbf{A} = (A_4, \circ)$$

Cannot compute AND_n
with expressions of polynomial size.

$$\mathbf{A} = (A_4, \circ, [x, y] = xyx^{-1}y^{-1})$$

Can compute AND_n
with expressions of polynomial size.

Why it matters?

Expressions over solvable algebras



ACC

Solvability in Universal Algebra does not depend on the signature!

Solution: use circuits instead of formulas.

Universal Algebraic Circuit Complexity

$\text{NUP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

Universal Algebraic Circuit Complexity

$\text{NUP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NUP}_{\mathbf{A}} = \text{P/poly}$$

Universal Algebraic Circuit Complexity

$\text{NUP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NUP}_{\mathbf{A}} = \text{P/poly}$$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee)$$

$$\text{UCON} \in \text{NUP}_{\mathbf{A}}$$

Universal Algebraic Circuit Complexity

$\text{NuP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NuP}_{\mathbf{A}} = \text{P/poly}$$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee)$$

$$\text{UCon} \in \text{NuP}_{\mathbf{A}}$$

$$\text{CLIQUE} \notin \text{NuP}_{\mathbf{A}} \quad \longrightarrow \text{Razborov'85}$$

$$\text{MATCHING} \notin \text{NuP}_{\mathbf{A}} \quad \nearrow$$

Universal Algebraic Circuit Complexity

$\text{NuP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NuP}_{\mathbf{A}} = \text{P/poly}$$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee)$$

$$\text{UCON} \in \text{NuP}_{\mathbf{A}}$$

$$\text{CLIQUE} \notin \text{NuP}_{\mathbf{A}} \quad \xrightarrow{\text{Razborov'85}}$$

$$\text{MATCHING} \notin \text{NuP}_{\mathbf{A}} \quad \xrightarrow{\text{Razborov'85}}$$

$$\text{NuP}_{\mathbf{A}} = \text{MONOTONE} \cup \overline{\text{MONOTONE}}$$

Universal Algebraic Circuit Complexity

$\text{NuP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NuP}_{\mathbf{A}} = \text{P/poly}$$

Can we understand how algebraic properties of $\mathbf{A}$ influence computing power of circuits over $\mathbf{A}$.

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee)$$

$$\text{UCon} \in \text{NuP}_{\mathbf{A}}$$

$$\text{CLIQUE} \notin \text{NuP}_{\mathbf{A}} \quad \rightarrow \text{Razborov'85}$$

$$\text{MATCHING} \notin \text{NuP}_{\mathbf{A}} \quad \rightarrow$$

$$\text{NuP}_{\mathbf{A}} = \text{MONOTONE} \cup \overline{\text{MONOTONE}}$$

Universal Algebraic Circuit Complexity

$\text{NuP}_{\mathbf{A}}$ - Boolean languages recognized by polynomial size circuits over $\mathbf{A}$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee, \neg)$$

$$\text{NuP}_{\mathbf{A}} = \text{P/poly}$$

$$\mathbf{A} = (\{0, 1\}, \wedge, \vee)$$

$$\text{UCON} \in \text{NuP}_{\mathbf{A}}$$

$$\text{CLIQUE} \notin \text{NuP}_{\mathbf{A}} \quad \rightarrow \text{Razborov'85}$$

$$\text{MATCHING} \notin \text{NuP}_{\mathbf{A}} \quad \rightarrow$$

$$\text{NuP}_{\mathbf{A}} = \text{MONOTONE} \cup \overline{\text{MONOTONE}}$$

Can we understand how algebraic properties of $\mathbf{A}$ influence computing power of circuits over $\mathbf{A}$.

Can we prove more general lower bounds in different algebraic settings?

Results

Circuit Complexity Dichotomy Theorem in CM

For a finite algebra $\mathbf{A}$ in CM either:

- $\text{NuP}_{\mathbf{A}} = \text{P/poly}$

or

- $\text{NuP}_{\mathbf{A}} \subseteq \text{NC}^2 \circ \text{MONOTONE}$

Results

Circuit Complexity Dichotomy Theorem in CM

For a finite algebra $\mathbf{A}$ in CM either:

- $\text{NuP}_{\mathbf{A}} = \text{P/poly}$

or

- $\text{NuP}_{\mathbf{A}} \subseteq \text{NC}^2 \circ \text{MONOTONE}$

Congruence Modular Varieties

Generalize groups, rings, lattices,
latin matrices, Boolean Algebras,
Heyting Algebras, Lie Algebras, loops
and many more...

Local behaviours in algebraic structures

In some sense finite algebraic structures admit only five types of local behaviour...

Local behaviours in algebraic structures

In some sense finite algebraic structures admit only five types of local behaviour...

1. Unary
2. Vector space
3. Boolean algebra
4. Lattice
5. Semilattice

Local behaviours in algebraic structures

In some sense finite algebraic structures admit only five types of local behaviour...

1. Unary
2. Vector space
3. Boolean algebra
4. Lattice
5. Semilattice

Difficulty: going from local to global.

Local behaviours in algebraic structures

In some sense finite algebraic structures admit only five local behaviours...

1. Unary

2. Vector space

3. Boolean algebra  $3 \in \text{typ}(\mathbf{A}) \implies \text{NUP}_{\mathbf{A}} = \text{P/poly}$

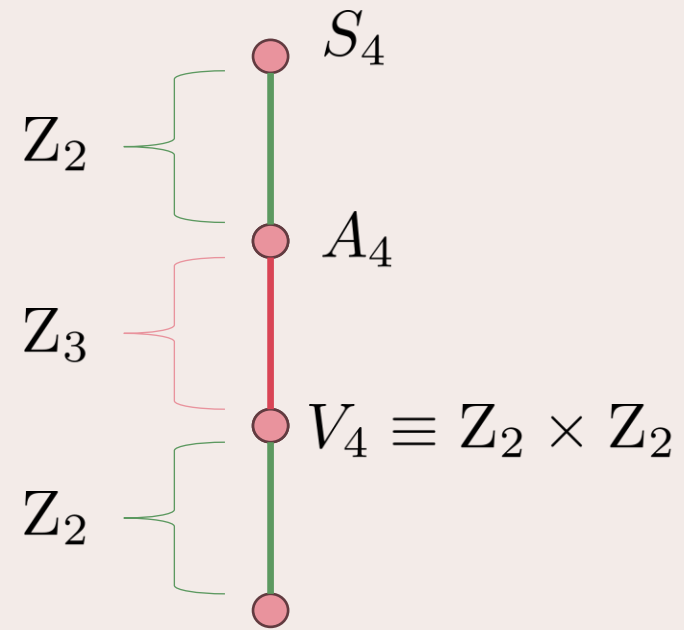
4. Lattice

5. Semilattice

Difficulty: going from local to global.

Solvable structures

Local fragments look like vector spaces everywhere.

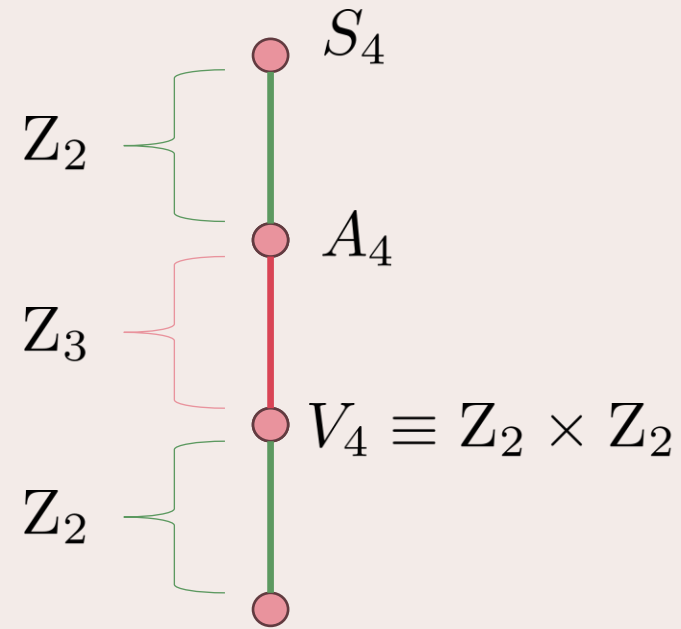


Solvable structures

Local fragments look like vector spaces everywhere.

LOCAL TO GLOBAL THEOREM (Krohn-Rhodes style):

Global structure can be retrived from local behaviours through matrix products and some very general wreath products.



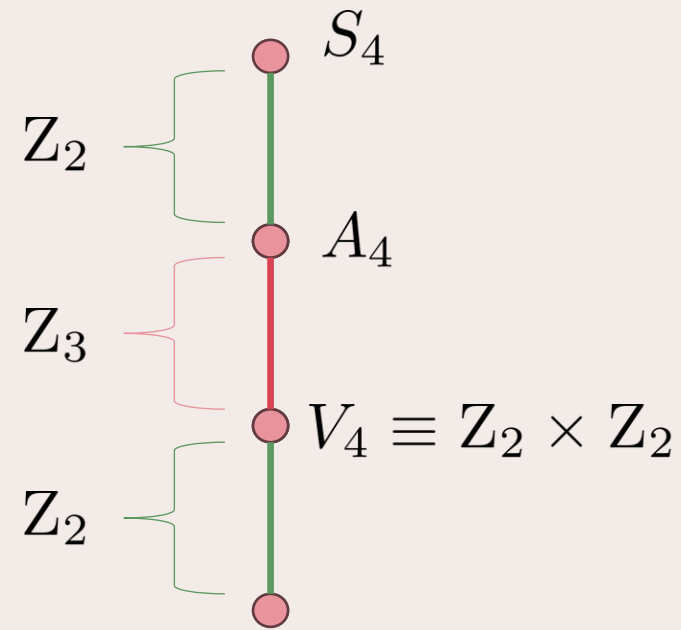
Solvable structures

Local fragments look like vector spaces everywhere.

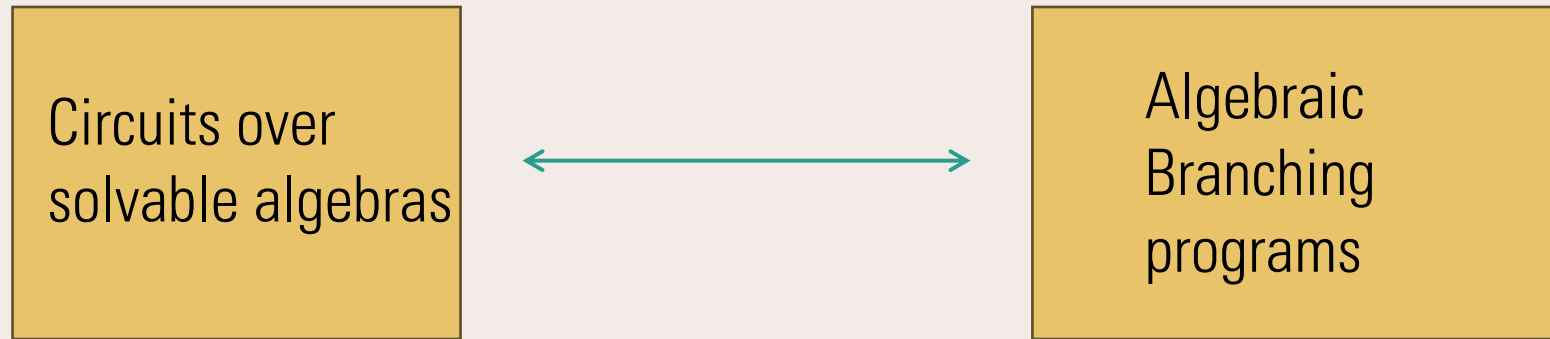
LOCAL TO GLOBAL THEOREM (Krohn-Rhodes style):

Global structure can be retrieved from local behaviours through matrix products and some very general wreath products.

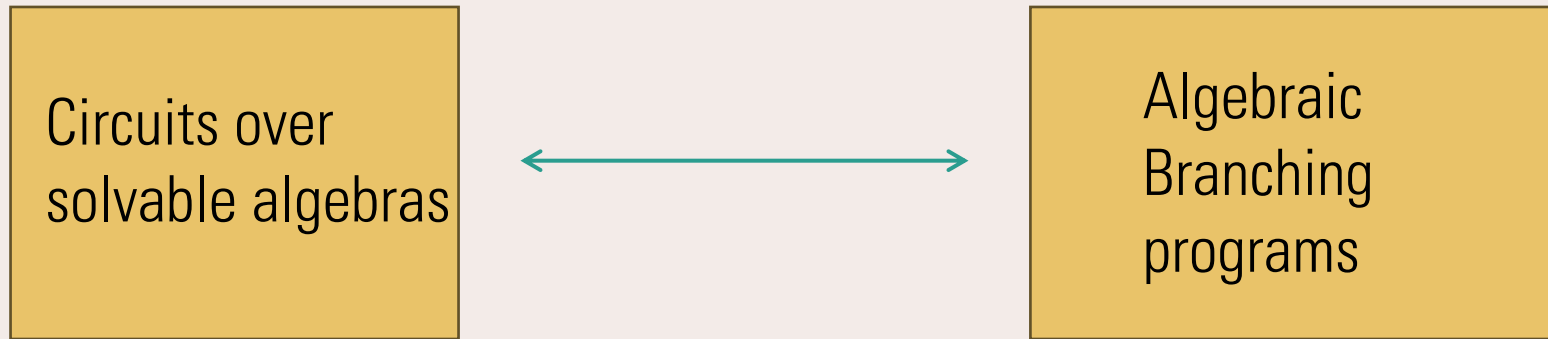
This paradigm works in CM!



Solvable structures and Arithmetic Complexity

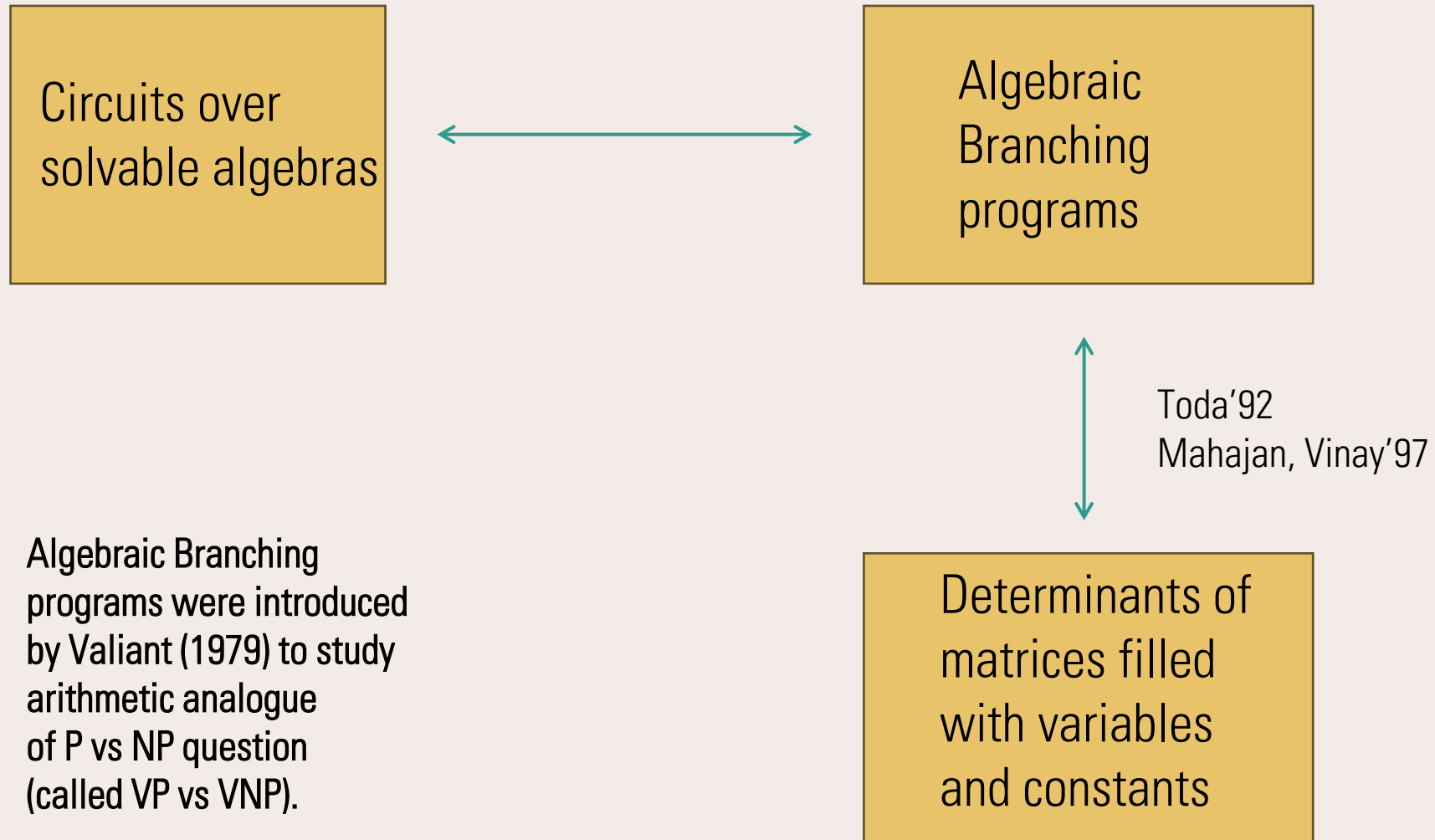


Solvable structures and Arithmetic Complexity



Algebraic Branching programs were introduced by Valiant (1979) to study arithmetic analogue of P vs NP question (called VP vs VNP).

Solvable structures and Arithmetic Complexity



Solvable structures and Arithmetic Complexity

Theorem

Let $\mathcal{C}$ be the class of all finite solvable algebras from CM. Then

$$\text{NUP}_{\mathcal{C}} = \text{NUITDET}$$

Solvable structures and Arithmetic Complexity

Theorem

Let $\mathcal{C}$ be the class of all finite solvable algebras from CM. Then

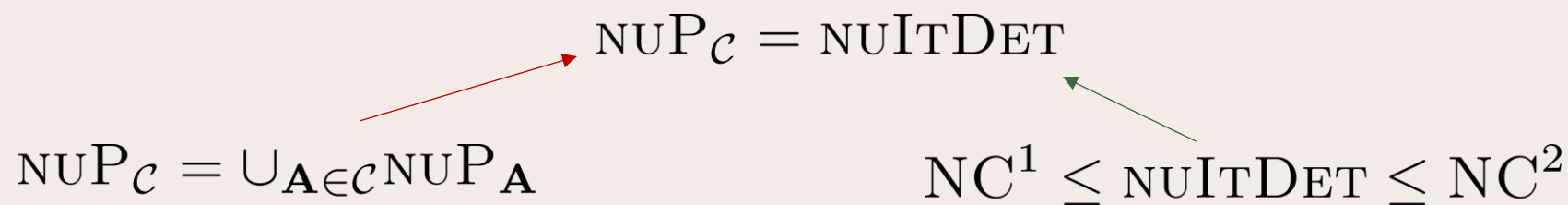
$$\text{NUP}_{\mathcal{C}} = \cup_{\mathbf{A} \in \mathcal{C}} \text{NUP}_{\mathbf{A}}$$

$$\text{NUP}_{\mathcal{C}} = \text{NITDET}$$

Solvable structures and Arithmetic Complexity

Theorem

Let $\mathcal{C}$ be the class of all finite solvable algebras from CM. Then

$$\text{NUP}_{\mathcal{C}} = \text{NITDET}$$

$$\text{NUP}_{\mathcal{C}} = \cup_{\mathbf{A} \in \mathcal{C}} \text{NUP}_{\mathbf{A}}$$
$$\text{NC}^1 \leq \text{NITDET} \leq \text{NC}^2$$

Solvable structures and Arithmetic Complexity

Theorem

Let $\mathcal{C}$ be the class of all finite solvable algebras from CM. Then

$$\text{NUP}_{\mathcal{C}} = \text{NITDET}$$

$\text{NUP}_{\mathcal{C}} = \cup_{\mathbf{A} \in \mathcal{C}} \text{NUP}_{\mathbf{A}}$ $\text{NC}^1 \leq \text{NITDET} \leq \text{NC}^2$

Solvable circuits are more expressive than Boolean formulas!

What about local lattice behaviour?

If $\mathbf{A}$ exhibits only local lattice behaviour then either

1. $\text{NUP}_{\mathbf{A}} = \text{P/poly}$
2. or $\text{NUP}_{\mathbf{A}} = \text{MULTIMONOTONE}$

What about local lattice behaviour?

If $\mathbf{A}$ exhibits only local lattice behaviour then either

1. $\text{NUP}_{\mathbf{A}} = \text{P/poly}$
2. or $\text{NUP}_{\mathbf{A}} = \text{MULTIMONOTONE}$



Circuits over $\mathbf{A}$ can be effectively simulated by constant number of monotone circuits.

What about local lattice behaviour?

If $\mathbf{A}$ exhibits only local lattice behaviour then either

1. $\text{NUP}_{\mathbf{A}} = \text{P/poly}$
2. or $\text{NUP}_{\mathbf{A}} = \text{MULTIMONOTONE}$



Circuits over $\mathbf{A}$ can be effectively simulated by constant number of monotone circuits.

IDEA:

Look at maximal order preserved on subdirect components.

If it is not total we can express the negation.

What about local lattice behaviour?

If $\mathbf{A}$ exhibits only local lattice behaviour then either

1. $\text{NUP}_{\mathbf{A}} = \text{P/poly}$
2. or $\text{NUP}_{\mathbf{A}} = \text{MULTIMONOTONE}$

Circuits over $\mathbf{A}$ can be effectively simulated by constant number of monotone circuits.



IDEA:

Look at maximal order preserved on subdirect components.

If it is not total we can express the negation.

Amano, Maruoka'06

$\text{CLIQUE} \notin \text{MULTIMONOTONE}$

Main result

Circuit Complexity Dichotomy Theorem in CM

For a finite algebra $\mathbf{A}$ in CM either:

- $\text{NuP}_{\mathbf{A}} = \text{P/poly}$

or

- $\text{NuP}_{\mathbf{A}} \subseteq \text{NC}^2 \circ \text{MONOTONE}$

Main result

Circuit Complexity Dichotomy Theorem in CM

For a finite algebra $\mathbf{A}$ in CM either:

- $\text{NuP}_{\mathbf{A}} = \text{P/poly}$

or

- $\text{NuP}_{\mathbf{A}} \subseteq \text{NC}^2 \circ \text{MONOTONE}$



If solvable and lattice behaviours interact in a right way.

BST'90 classes are also obtainable!

Theorem

If $\mathbf{A}$ is nilpotent then $\text{NUP}_{\mathbf{A}} \subseteq \text{CC}^0$.

Moreover $\text{NUP}_{\text{NILP}} = \text{CC}^0$.

BST'90 classes are also obtainable!

Theorem

If $\mathbf{A}$ is nilpotent then $\text{NUP}_{\mathbf{A}} \subseteq \text{CC}^0$.

Moreover $\text{NUP}_{\text{NILP}} = \text{CC}^0$.

One can also capture $\text{AC}^0, \text{ACC}^0$ with proper constructions (outside CM)...

BST'90 classes are also obtainable!

Theorem

If $\mathbf{A}$ is nilpotent then $\text{NUP}_{\mathbf{A}} \subseteq \text{CC}^0$.

Moreover $\text{NUP}_{\text{NILP}} = \text{CC}^0$.

One can also capture $\text{AC}^0, \text{ACC}^0$ with proper constructions (outside CM)...

Question: Can we capture classes like NC^1, TC^0 in our framework?

Outside CM – simple algebras

We can characterize circuit complexity classes for simple algebras...

1. Unary
2. Vector space
3. Boolean algebra
4. Lattice
5. Semilattice

Outside CM – simple algebras

We can characterize circuit complexity classes for simple algebras...

1. Unary
2. Vector space
3. Boolean algebra
4. Lattice
5. Semilattice

Fully

Outside CM – simple algebras

We can characterize circuit complexity classes for simple algebras...

1. Unary
2. Vector space
3. Boolean algebra
4. Lattice

Fully

5. Semilattice

These guys are really complex!
They can do far more than just semilattice operations...

Perspectives

1. Going outside CM – Krohn-Rhodes and more...

Perspectives

1. Going outside CM – Krohn-Rhodes and more...
2. Most complexity classes define pseudovariety. They are closed under HSP_{fin} .

Perspectives

1. Going outside CM – Krohn-Rhodes and more...
2. Most complexity classes define pseudovariety. They are closed under HSP_{fin} .
3. Algebraic methods in separating complexity classes?

Perspectives

1. Going outside CM – Krohn-Rhodes and more...
2. Most complexity classes define pseudovariety. They are closed under HSP_{fin} .
3. Algebraic methods in separating complexity classes?
4. Solvable algebras can hide many interesting circuit complexity classes...

Perspectives

1. Going outside CM – Krohn-Rhodes and more...
2. Most complexity classes define pseudovariety. They are closed under HSP_{fin} .
3. Algebraic methods in separating complexity classes?
4. Solvable algebras can hide rich structure...
5. How to extend TCT to understand simple algebras of semilattice type?

THANK YOU!