Logic, Complexity, and Symmetry

Erich Grädel

Conference in Honor of Erwin Engeler and Ernst Specker

ETH Zürich, February 2020

Logic and Computation

Connections between logic and algorithms have been important for the scientific work of both Erwin Engeler and Ernst Specker.

There are many facets of this relationship:

- Logic poses many algorithmic problems: Model checking, satisfiability testing, entailment, provability, . . . ,
- Logical representation of knowledge and data
- Definability versus complexity: Logic capturing complexity classes
- Logic as a technology!

Logic and Computation

Connections between logic and algorithms have been important for the scientific work of both Erwin Engeler and Ernst Specker.

There are many facets of this relationship:

- Logic poses many algorithmic problems: Model checking, satisfiability testing, entailment, provability, . . . ,
- Logical representation of knowledge and data
- Definability versus complexity: Logic capturing complexity classes
- Logic as a technology!

And then, logic has this imperialist claim on the foundations of everything ....

Logic and Computation

Connections between logic and algorithms have been important for the scientific work of both Erwin Engeler and Ernst Specker.

There are many facets of this relationship:

- Logic poses many algorithmic problems: Model checking, satisfiability testing, entailment, provability, ...,
- Logical representation of knowledge and data
- Definability versus complexity: Logic capturing complexity classes
- Logic as a technology!

And then, logic has this imperialist claim on the foundations of everything ....

But this relationship is not without tensions. Many problems are surprisingly difficult, and some researchers even speak of a mismatch between logic and computation. Why?

The tension between logic and computation

Classical computation devices (such as Turing machines) work on ordered representations of data, such as words, strings of numbers, etc. When solving a problem on, say, graphs, they are given ordered representations of them, e.g. via adjacency matrices. The implicit order on the vertices may be used, by the algorithm but the result must be invariant under the chosen ordering.

The tension between logic and computation

Classical computation devices (such as Turing machines) work on ordered representations of data, such as words, strings of numbers, etc. When solving a problem on, say, graphs, they are given ordered representations of them, e.g. via adjacency matrices. The implicit order on the vertices may be used, by the algorithm but the result must be invariant under the chosen ordering.

Logic and logic based computation models work on abstract mathematical structures. Inherent symmetries, and indistinguishability between individual elements are respected not only for the final result, but at each step of the evaluation or computation.

Symmetry and choice

Many important algorithms (depth first search, Gaussian elimination, . . .) rely on explicit choices: at some steps, out of a collection of "equivalent" objects, they choose one, and proceed.

Symmetry and choice

Many important algorithms (depth first search, Gaussian elimination, . . .) rely on explicit choices: at some steps, out of a collection of "equivalent" objects, they choose one, and proceed.

Logic and logical computation models cannot make such explicit choices, because these would break symmetries!

Question: Can we replace these classical algorithm by symmetric ones that avoid such choices, without paying a huge prize, in terms of computation time and/or other resources?

Symmetry and choice

Many important algorithms (depth first search, Gaussian elimination, . . .) rely on explicit choices: at some steps, out of a collection of "equivalent" objects, they choose one, and proceed.

Logic and logical computation models cannot make such explicit choices, because these would break symmetries!

Question: Can we replace these classical algorithm by symmetric ones that avoid such choices, without paying a huge prize, in terms of computation time and/or other resources?

This is possible for depth-first search, but open for, say, solving linear equation system over finite fields.

The most important problem of Finite Model Theory

Is there a logic that captures PTIME?

The most important problem of Finite Model Theory

Is there a logic that captures PTIME?

Informal definition: A logic L captures PTIME if it defines precisely those properties of finite structures that are decidable in polynomial time:

(1) For every sentence $\psi \in L$, the set of finite models of ψ is decidable in polynomial time.

(2) For every PTIME-property S of finite τ-structures, there is a sentence $\psi \in L$ such that $S = \{\mathfrak{A} \in \mathrm{Fin}(\tau) : \mathfrak{A} \models \psi\}$.

The most important problem of Finite Model Theory

Is there a logic that captures PTIME?

Informal definition: A logic L captures PTIME if it defines precisely those properties of finite structures that are decidable in polynomial time:

(1) For every sentence $\psi \in L$, the set of finite models of ψ is decidable in polynomial time.

(2) For every PTIME-property S of finite τ-structures, there is a sentence $\psi \in L$ such that $S = \{\mathfrak{A} \in \mathrm{Fin}(\tau) : \mathfrak{A} \models \psi\}$.

The precise definition is more subtle. It includes effectiveness requirements to exclude pathological 'solutions'.

First-Order Logic

First-order logic (FO) is far too weak to capture PTIME.

- FO can express only local properties of finite structures
  Theorems of Gaifman and Hanf

  Global properties (e.g. planarity of graphs) are not expressible.

- FO has no mechanism for recursion or unbounded iteration.

  Transitive closures, reachability or termination properties, winning regions in games, etc. are not FO-definable.

- FO can only express properties in AC^0

  AC^0 is constant parallel time with polynomial hardware. In particular, FO $\subseteq$ LOGSPACE.

Second-Order Logic

Second-order logic (SO) is probably too strong to capture PTIME.

Fagin's Theorem. Existential SO captures NP.

Corollary. SO captures the polynomial hierarchy.

Thus SO captures polynomial time if, and only if, P = NP.

Second-Order Logic

Second-order logic (SO) is probably too strong to capture PTIME.

Fagin's Theorem. Existential SO captures NP.

Corollary. SO captures the polynomial hierarchy.

Thus SO captures polynomial time if, and only if, P = NP.

Monadic second-order logic (MSO) is orthogonal to PTIME:

On words, MSO captures the regular languages, and not all PTIME-languages are regular.

On graphs, MSO can express NP-complete properties, such as 3-colourability.

Fixed-point logic with counting

(FP + C): Two-sorted fixed-point logic with counting terms.

Two sorts of variables:

- $x, y, z, \ldots$ ranging over the domain of the given finite structure
- $\mu, \nu, \ldots$ ranging over natural numbers

On natural numbers, operations $+$, $\cdot$ and $<$ are available, but variables must be explicitly restricted to take only polynomially bounded values.

Counting terms: For a formula $\varphi(x)$, the term $\#_x \varphi(x)$ denotes the number of elements a of the structure that satisfy $\varphi(a)$.

Mechanism for polynomial-time relational recursion:
Fixed points of update operators $R \mapsto R \cup \{(\overline{a}, \overline{m}) : \mathfrak{A} \models \varphi(R, \overline{a}, \overline{m})\}$

Fixed-point logic with counting is close to PTIME

Fixed-point logic with counting is powerful enough to express fundamental algorithmic techniques (such as the ellipsoid method) and captures PTIME on many interesting classes of finite structures, including

- linearly ordered structures (Immerman, Vardi)
- trees (Immerman, Lander) and
  structures of bounded tree-width (Grohe, Marino)
- planar graphs and graphs of bounded genus (Grohe)
- chordal line graphs (Grohe) and interval graphs (Laubner)
- all classes of graphs that exclude a minor (Grohe)

Fixed-point logic with counting is close to PTIME

Fixed-point logic with counting is powerful enough to express fundamental algorithmic techniques (such as the ellipsoid method) and captures PTIME on many interesting classes of finite structures, including

- linearly ordered structures (Immerman, Vardi)
- trees (Immerman, Lander) and
  structures of bounded tree-width (Grohe, Marino)
- planar graphs and graphs of bounded genus (Grohe)
- chordal line graphs (Grohe) and interval graphs (Laubner)
- all classes of graphs that exclude a minor (Grohe)

(FP+C) is the logic of reference in this area!
(see survey by A. Dawar, SIGLOG-News, 2015)

The CFI-query

Given a connected graph $G = (V, E)$, and a subset $T \subseteq E$, construct the CFI-graph $X_T(G)$:

- replace every node v by a gadget $H(v)$, which has two exit points a_{vw} and b_{vw} for every neighbour $w \in vE$
- replace every edge by two edges that connnect corresponding exit points: a_{vw} with a_{wv} and b_{vw} with b_{wv}
- twist the double-edges in T

The CFI-query

Given a connected graph $G = (V, E)$, and a subset $T \subseteq E$, construct the CFI-graph $X_T(G)$:

- replace every node v by a gadget $H(v)$, which has two exit points a_{vw} and b_{vw} for every neighbour $w \in vE$
- replace every edge by two edges that connnect corresponding exit points: a_{vw} with a_{wv} and b_{vw} with b_{wv}
- twist the double-edges in T

Fact: $X_S(G) \cong X_T(G) \iff |S| = |T| \pmod 2$

Thus, for every G, there are up to isomorphism exactly two CFI-graphs: $X(G) := X_\emptyset(G)$ and $\tilde{X}(G) := X_{\{e\}}(G)$

The CFI-query

Given a connected graph $G = (V, E)$, and a subset $T \subseteq E$, construct the CFI-graph $X_T(G)$:

- replace every node v by a gadget $H(v)$, which has two exit points a_{vw} and b_{vw} for every neighbour $w \in vE$
- replace every edge by two edges that connnect corresponding exit points: a_{vw} with a_{wv} and b_{vw} with b_{wv}
- twist the double-edges in T

Fact: $X_S(G) \cong X_T(G) \iff |S| = |T| \pmod 2$

Thus, for every G, there are up to isomorphism exactly two CFI-graphs: $X(G) := X_\emptyset(G)$ and $\tilde{X}(G) := X_{\{e\}}(G)$

The CFI-query: Given a CFI-graph, determine whether it is $X(G)$ or $\tilde{X}(G)$.

Fixed-point logic with counting versus polynomial time

Theorem. The CFI-query is in PTIME, but not in (FP + C).

(Cai, Fürer, Immerman 1992)

Fixed-point logic with counting versus polynomial time

Theorem. The CFI-query is in PTIME, but not in (FP + C).
(Cai, Fürer, Immerman 1992)

The CFI-construction separating PTIME from (FP+C) is interesting and sophisticated, but originally seemed somewhat artificial.

However, Atserias, Bulatov, and Dawar proved that it very closely related to the fundamental problem of solving linear equation systems over finite Abelian groups, rings, and fields.

Fixed-point logic with counting versus polynomial time

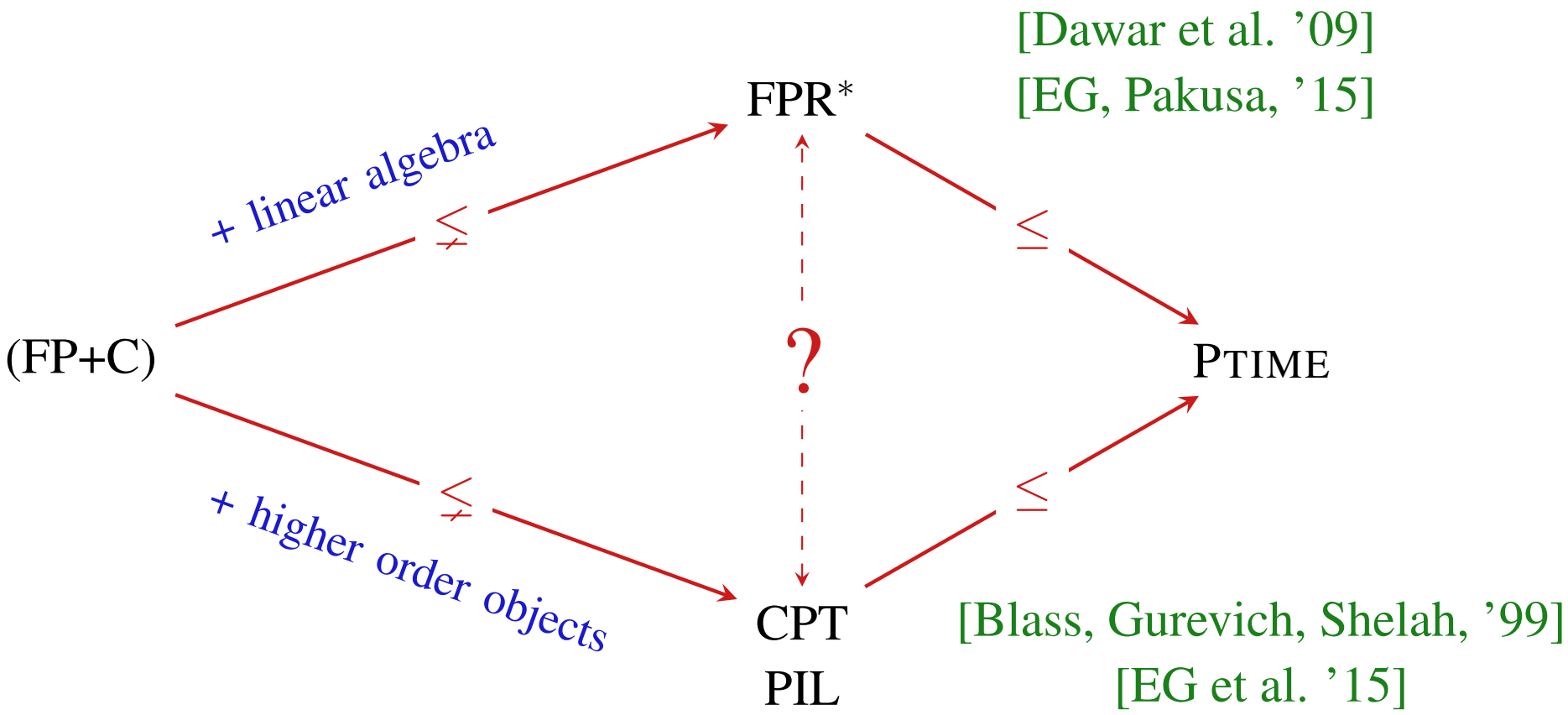
Theorem. The CFI-query is in PTIME, but not in (FP + C).
(Cai, Fürer, Immerman 1992)

The CFI-construction separating PTIME from (FP+C) is interesting and sophisticated, but originally seemed somewhat artificial.

However, Atserias, Bulatov, and Dawar proved that it very closely related to the fundamental problem of solving linear equation systems over finite Abelian groups, rings, and fields.

Today, the CFI-query and its variants and generalizations still provide interesting benchmarks and challenges for any candidate for a logic for polynomial time.

Candidates for a logic for PTIME



Fixed-point logic with rank

Rank logic FPR: Extend fixed-point logic by rank operators $\mathrm{rk}_p\varphi$, to denote the rank (over the prime field $\mathbb{F}_p$) of the matrix defined by φ.

proposed by Dawar et al. (2009) as a candidate for a logic for PTIME

Fixed-point logic with rank

Rank logic FPR: Extend fixed-point logic by rank operators $\mathrm{rk}_p\varphi$, to denote the rank (over the prime field $\mathbb{F}_p$) of the matrix defined by φ.

proposed by Dawar et al. (2009) as a candidate for a logic for PTIME

FPR can express the solvability of linear equation systems over finite fields, and thus the isomorphism of CFI-graphs: (FP+C) < FPR $\leq$ PTIME.

Fixed-point logic with rank

Rank logic FPR: Extend fixed-point logic by rank operators $\mathrm{rk}_p\varphi$, to denote the rank (over the prime field $\mathbb{F}_p$) of the matrix defined by φ.

proposed by Dawar et al. (2009) as a candidate for a logic for PTIME

FPR can express the solvability of linear equation systems over finite fields, and thus the isomorphism of CFI-graphs: (FP+C) < FPR $\leq$ PTIME.

Theorem. (EG, Pakusa, JSL 2019) Rank logic is dead, long live rank logic!

In its original form, FPR fails to capture PTIME ! We must replace it by a stronger variant, FPR*, where the rank operator takes the prime as an additional input.

Fixed-point logic with rank

Rank logic FPR: Extend fixed-point logic by rank operators $\mathrm{rk}_p\varphi$, to denote the rank (over the prime field $\mathbb{F}_p$) of the matrix defined by φ.

proposed by Dawar et al. (2009) as a candidate for a logic for PTIME

FPR can express the solvability of linear equation systems over finite fields, and thus the isomorphism of CFI-graphs: (FP+C) < FPR $\leq$ PTIME.

Theorem. (EG, Pakusa, JSL 2019) Rank logic is dead, long live rank logic!

In its original form, FPR fails to capture PTIME ! We must replace it by a stronger variant, FPR*, where the rank operator takes the prime as an additional input.

Open problem. Does FPR* capture PTIME ?

Fixed-point logic with rank

Rank logic FPR: Extend fixed-point logic by rank operators $\mathrm{rk}_p\varphi$, to denote the rank (over the prime field $\mathbb{F}_p$) of the matrix defined by φ.

proposed by Dawar et al. (2009) as a candidate for a logic for PTIME

FPR can express the solvability of linear equation systems over finite fields, and thus the isomorphism of CFI-graphs: (FP+C) < FPR $\leq$ PTIME.

Theorem. (EG, Pakusa, JSL 2019) Rank logic is dead, long live rank logic!

In its original form, FPR fails to capture PTIME ! We must replace it by a stronger variant, FPR*, where the rank operator takes the prime as an additional input.

Open problem. Does FPR* capture PTIME ?

(Actually, nobody believes that it really does!)

Choiceless Polynomial Time (Blass, Gurevich, Shelah 1999)

Idea. Model for computation on abstract structures that preserves symmetries.
Disallow explicit choice, but permit essentially everything else, including
fancy data structures and parallelism (explore all possible choices in parallel).

Choiceless Polynomial Time (Blass, Gurevich, Shelah 1999)

Idea. Model for computation on abstract structures that preserves symmetries. Disallow explicit choice, but permit essentially everything else, including fancy data structures and parallelism (explore all possible choices in parallel).

States: sets in the hereditarily finite expansion HF($\mathfrak{A}$) of the input $\mathfrak{A}$
- atoms: the elements of $\mathfrak{A}$
- all finite sets of elements of HF($\mathfrak{A}$)

Compute with set-theoretic operations such as $\varnothing, \in, \bigcup, |\ |$, and comprehension.

Choiceless Polynomial Time is the set of properties computable by such machines such that
- computations have polynomial length
- only a polynomial number of sets are activated.

The power of choiceless polynomial time

CPT is a proper extension of (FP + C)

CPT can define any polynomial time property of small definable substructures X of the input structure $\mathfrak{A}$.

The power of choiceless polynomial time

CPT is a proper extension of (FP + C)

CPT can define any polynomial time property of small definable substructures X of the input structure $\mathfrak{A}$.
Small: $|X|! \leq |A|$. Generate in parallel all linear orders on X and simulate a polynomial time computation on an ordered structure by the usual techniques.

The power of choiceless polynomial time

CPT is a proper extension of (FP + C)

CPT can define any polynomial time property of small definable substructures X of the input structure $\mathfrak{A}$.
Small: $|X|! \leq |A|$. Generate in parallel all linear orders on X and simulate a polynomial time computation on an ordered structure by the usual techniques.

CPT can solve some cases of the Cai-Fürer-Immerman problem

The power of choiceless polynomial time

CPT is a proper extension of (FP + C)

CPT can define any polynomial time property of small definable substructures X of the input structure $\mathfrak{A}$.
Small: $|X|! \leq |A|$. Generate in parallel all linear orders on X and simulate a polynomial time computation on an ordered structure by the usual techniques.

CPT can solve some cases of the Cai-Fürer-Immerman problem

CPT can solve certain systems of linear equations that cannot be solved in (FP+C), with an appropriate pre-order on the variables

A different view: computing by interpretations

Idea: Replace the manipulation of hereditarily finite sets by first-order interpretations.

Instead of a sequence of hereditarily finite sets, a computation then is a sequence of finite structures obtained by repeated application of a fixed first-order interpretation.

A different view: computing by interpretations

Idea: Replace the manipulation of hereditarily finite sets by first-order interpretations.
Instead of a sequence of hereditarily finite sets, a computation then is a sequence of finite structures obtained by repeated application of a fixed first-order interpretation.

Interpretations: A FO$[\tau,\sigma]$-interpretation is a sequence

$$I = (\delta(\bar{x}), \varepsilon(\bar{x},\bar{y}), (\varphi_R(\bar{x}_1,\ldots,\bar{x}_{s(R)})_{R\in\sigma})$$

of FO$[\tau]$-formulae. It maps a τ-structure $\mathfrak{A}$ to a σ-structure

$$I(\mathfrak{A}) = (\delta^{\mathfrak{A}}, (\varphi_R^{\mathfrak{A}})_{R\in\sigma})/\varepsilon^{\mathfrak{A}}$$

Notice that interpretations may change the size of the structures.

Computing by interpretations

Polynomial Time Interpretation Logic PIL $\quad \Pi = (I_{\text{init}}, I_{\text{step}}, \varphi_{\text{halt}}, \varphi_{\text{out}})$

- I_{init} is an interpretation defining from the input structure $\mathfrak{A}$ an initial state $\mathfrak{A}_0 := I_{\text{init}}(\mathfrak{A})$
- I_{step} is an interpretation defining from a state $\mathfrak{A}_i$ the next state $\mathfrak{A}_{i+1} := I_{\text{step}}(\mathfrak{A}_i)$
- the run $\mathfrak{A}_0, \mathfrak{A}_1, \ldots$ of Π in $\mathfrak{A}$ terminates at the first state $\mathfrak{A}_n$ with $\mathfrak{A}_n \models \varphi_{\text{halt}}$
- Π accepts $\mathfrak{A}$ if the run terminates at state $\mathfrak{A}_n$ with $\mathfrak{A}_n \models \varphi_{\text{out}}$

Explicit polynomial bounds on the length of the run and the size of all states.

To get the full power of CPT, interpretations have to be equipped with a counting construct, such as the Härtig quantifier.

Theorem $\quad$ CPT $\equiv$ PIL $\quad$ (EG, Kaiser, Pakusa, Schalthöfer, 2015)

The surprising power of CPT and (FP+C)

Many candidates have been proposed for separating PTIME from CPT. However, for most of them, it has turned out that they are CPT-computable, or even definable in (FP+C).

The summation problem for Abelian groups and semigroups:
Given: A finite (semi)group $(G, +, 0)$ and a subset $X \subset G$.
Question: Determine $\sum X$.

> *"This is the most basic problem I can think of that appears difficult for CPT but is obviously polynomial time. I don't even know the answer when G is an Abelian group, or even a direct product of cyclic groups $\mathbb{Z}_2$."* *(Ben Rossman, 2005)*

The surprising power of CPT and (FP+C)

Many candidates have been proposed for separating PTIME from CPT. However, for most of them, it has turned out that they are CPT-computable, or even definable in (FP+C).

The summation problem for Abelian groups and semigroups:
Given: A finite (semi)group $(G, +, 0)$ and a subset $X \subset G$.
Question: Determine $\sum X$.

> *"This is the most basic problem I can think of that appears difficult for CPT but is obviously polynomial time. I don't even know the answer when G is an Abelian group, or even a direct product of cyclic groups $\mathbb{Z}_2$."* *(Ben Rossman, 2005)*

Theorem. (Abu Zaid, Dawar, EG, Pakusa, 2017)
The summation problem for Abelian semigroups is even definable in (FP+C).

Symmetric circuits

A circuit family $(C_n)_{n\in\mathbb{N}}$ decides a property of finite τ-structures if C_n takes as inputs the truth values of atomic τ-formulae of structures with universe $[n] = \{0, \ldots, n-1\}$, and if it is invariant under isomorphisms.

Invariance: Any permutation of $[n]$ induces a permutation of the input gates of C_n. The result of the computation of C_n must be invariant under this.

Symmetric circuits

A circuit family $(C_n)_{n\in\mathbb{N}}$ decides a property of finite τ-structures if C_n takes as inputs the truth values of atomic τ-formulae of structures with universe $[n] = \{0, \ldots, n-1\}$, and if it is invariant under isomorphisms.

Invariance: Any permutation of $[n]$ induces a permutation of the input gates of C_n. The result of the computation of C_n must be invariant under this.

Translate any formula from FO or LFP into a circuit family $C = (C_n)_{n\in\mathbb{N}}$. Then this sequence is

p-uniform: The circuit C_n is polynomial-time computable in n
symmetric: Every permutation of $[n]$ induces an automorphism of C_n

Symmetric circuits are always invariant. The converse is not true.

Symmetric threshold circuits

For logics with counting it is natural to consider circuits with threshold gates.

The extension by threshold gates does not increase the power of polynomial-size circuits. But it can make a difference for restricted classes, such as bounded-depth circuits or symmetric circuits.

Symmetric threshold circuits

For logics with counting it is natural to consider circuits with threshold gates.

The extension by threshold gates does not increase the power of polynomial-size circuits. But it can make a difference for restricted classes, such as bounded-depth circuits or symmetric circuits.

Every formula in (FP+C) can be translated into a p-uniform sequence of symmetric threshold circuits.

Question. Can this also be done for Choiceless Polynomial Time?

Is there a circuit model for CPT?

Theorem (Anderson, Dawar)

p-uniform symmetric threshold circuits are equivalent to (FP+C).

Thus, translations from Choiceless Polynomial Time into equivalent sequences of symmetric threshold circuits are not p-uniform.

Is there a circuit model for CPT?

Theorem (Anderson, Dawar)
p-uniform symmetric threshold circuits are equivalent to (FP+C).

Thus, translations from Choiceless Polynomial Time into equivalent sequences of symmetric threshold circuits are not p-uniform.

To put it differently, p-uniform translations form CPT into threshold circuits must break symmetry in some way. But how?

Challenge: Find a circuit model for CPT, based on a weaker notion of symmetry.

Challenges for future research

CFI-graphs: Can the isomorphism problem for CFI-graphs constructed from arbitrary input graphs be solved in CPT ?

Actually this might be a candidate for separating CPT from PTIME

Challenges for future research

CFI-graphs: Can the isomorphism problem for CFI-graphs constructed from arbitrary input graphs be solved in CPT ?
Actually this might be a candidate for separating CPT from PTIME

Choiceless Polynomial-Time versus Rank Logic: Besides CPT, logics with operators from linear algebra, such as the rank logic FPR*, seem to be the most prominent candidates for a logic for PTIME. The relationship between CPT and FPR* is unclear but cyclic equation systems (CES) over rings might separate the two logics.

Conjecture. Solvability of CES over $\mathbb{Z}_4$ is definable in CPT but not in FPR*.

Challenges for future research

CFI-graphs: Can the isomorphism problem for CFI-graphs constructed from arbitrary input graphs be solved in CPT ?
Actually this might be a candidate for separating CPT from PTIME

Choiceless Polynomial-Time versus Rank Logic: Besides CPT, logics with operators from linear algebra, such as the rank logic FPR*, seem to be the most prominent candidates for a logic for PTIME. The relationship between CPT and FPR* is unclear but cyclic equation systems (CES) over rings might separate the two logics.

Conjecture. Solvability of CES over $\mathbb{Z}_4$ is definable in CPT but not in FPR*.

Symmetric circuits for CPT: Find a circuit model for CPT. Understand better the symmetries inherent in CPT-computations.