

Descriptive complexity and graphs with excluded minors

Martin Grohe

Department of Computer Science
Humboldt University Berlin



1. Descriptive Complexity Theory
2. Fixed-Point Logics
3. Applications to the Isomorphism Problem
4. Stronger Logics

1. Descriptive Complexity Theory

2. Fixed-Point Logics

3. Applications to the Isomorphism Problem

4. Stronger Logics

Fagin's Theorem

Fagin's Theorem (1974)

Existential second-order logic Σ_1^1 captures NP.

Fagin's Theorem (1974)

Existential second-order logic Σ_1^1 captures NP.

That is, a property of finite structures is definable in Σ_1^1 iff it is decidable by a nondeterministic polynomial time algorithm.

Fagin's Theorem (1974)

Existential second-order logic Σ_1^1 captures NP.

That is, a property of finite structures is definable in Σ_1^1 iff it is decidable by a nondeterministic polynomial time algorithm.

Question (Chandra and Harel 1982, Gurevich 1988)

Is there a logic that captures PTIME ?

Key Issue

Instances of algorithmic problems are (finite) structures and not words encoding them.

Examples

Graphs, Boolean circuits, words over some finite alphabet, finite groups

Key Issue

Instances of algorithmic problems are (finite) structures and not words encoding them.

Examples

Graphs, Boolean circuits, words over some finite alphabet, finite groups

Encodings of Structures

- ▶ **Ordered structures** are structures with a distinguished relation that is a linear order of the elements

Key Issue

Instances of algorithmic problems are (finite) structures and not words encoding them.

Examples

Graphs, Boolean circuits, words over some finite alphabet, finite groups

Encodings of Structures

- ▶ **Ordered structures** are structures with a distinguished relation that is a linear order of the elements
- ▶ Ordered structures have a **canonical** encoding by words over $\{0, 1\}$

Key Issue

Instances of algorithmic problems are (finite) structures and not words encoding them.

Examples

Graphs, Boolean circuits, words over some finite alphabet, finite groups

Encodings of Structures

- ▶ **Ordered structures** are structures with a distinguished relation that is a linear order of the elements
- ▶ Ordered structures have a **canonical** encoding by words over $\{0, 1\}$
- ▶ Arbitrary structures have no obvious canonical encoding (that is efficiently computable), but many different encodings

Consequence

Decision problems are no longer languages, but **properties of structures** (a.k.a. **Boolean queries**).

Consequence

Decision problems are no longer languages, but **properties of structures** (a.k.a. **Boolean queries**).

A **(Boolean) query of vocabulary σ** is a class of structures of vocabulary σ that is closed under isomorphism.

Consequence

Decision problems are no longer languages, but **properties of structures** (a.k.a. **Boolean queries**).

A **(Boolean) query of vocabulary σ** is a class of structures of vocabulary σ that is closed under isomorphism.

Examples

Connectedness of graphs, satisfiability of Boolean circuits, the language $\{a^n b^n \mid n \geq 0\}$, solvability of groups

Complexity of Queries

A query Q belongs to complexity class K if the language $L(Q)$ of all encodings of all structures in Q is in Q .

Complexity of Queries

A query \mathcal{Q} belongs to complexity class K if the language $L(\mathcal{Q})$ of all encodings of all structures in \mathcal{Q} is in \mathcal{Q} .

Observation

It is undecidable whether a given Turing machine decides a query.

Model-theoretic view on logics

Logic consists of set of sentences and a satisfaction relation \models between structures and sentences:

$A \models \phi : \iff$ structure A satisfies sentence ϕ

Model-theoretic view on logics

Logic consists of set of sentences and a satisfaction relation \models between structures and sentences:

$$A \models \phi \iff \text{structure } A \text{ satisfies sentence } \phi$$

such that for every sentence ϕ the set

$$Q_\phi := \{A \mid A \models \phi\}$$

is closed under isomorphism (i.e., a Boolean query).

Model-theoretic view on logics

Logic consists of set of sentences and a satisfaction relation \models between structures and sentences:

$$A \models \phi \iff \text{structure } A \text{ satisfies sentence } \phi$$

such that for every sentence ϕ the set

$$Q_\phi := \{A \mid A \models \phi\}$$

is closed under isomorphism (i.e., a Boolean query).

Some effectiveness conditions to exclude pathological examples.

Logics Capturing Complexity Classes

Definition

A logic L captures a complexity class K

if for every query Q :

Q definable in $L \iff Q$ belongs to K .

Logics Capturing Complexity Classes

Definition

A logic L captures a complexity class K

if for every query Q :

$$Q \text{ definable in } L \iff Q \text{ belongs to } K.$$

Uniformity condition to exclude pathological examples.

Logics Capturing Complexity Classes

Definition

A logic L captures a complexity class K

on a class \mathcal{C} of structures

if for every query Q :

Q definable in L on $\mathcal{C} \iff Q$ belongs to K on \mathcal{C} .

Uniformity condition to exclude pathological examples.

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely the queries computable in polynomial time are expressible ?

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely the queries computable in polynomial time are expressible ?

“Is there a query language that expresses precisely those queries that can be answered efficiently ?”

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely the queries computable in polynomial time are expressible ?

“Is there a query language that expresses precisely those queries that can be answered efficiently ?”

Observation

Chandra and Harel’s question is equivalent to the question for a logic capturing PTIME:

Motivation from Database Theory

Question (Chandra and Harel 1982)

Is there a query language for relational databases in which precisely the queries computable in polynomial time are expressible ?

“Is there a query language that expresses precisely those queries that can be answered efficiently ?”

Observation

Chandra and Harel’s question is equivalent to the question for a logic capturing PTIME:

relational database \approx finite structure

query \approx property of finite structures

- ▶ A **canonization algorithm** for a class \mathcal{C} of structures transforms an arbitrary encoding of a structure into a canonical one.

- ▶ A **canonization algorithm** for a class \mathcal{C} of structures transforms an arbitrary encoding of a structure into a canonical one.
- ▶ If there is a polynomial time canonization algorithm for a class \mathcal{C} , then there is a logic that captures PTIME on \mathcal{C} . But the logics obtained this way are very artificial.

- ▶ A **canonization algorithm** for a class \mathcal{C} of structures transforms an arbitrary encoding of a structure into a canonical one.
- ▶ If there is a polynomial time canonization algorithm for a class \mathcal{C} , then there is a logic that captures PTIME on \mathcal{C} . But the logics obtained this way are very artificial.

Examples

planar graphs and graphs of bounded genus, graphs of bounded degree, graphs of bounded tree width

State of the Art in Descriptive Complexity Theory

- ▶ For most natural complexity classes, we know logics capturing them on the class of ordered structures.

State of the Art in Descriptive Complexity Theory

- ▶ For most natural complexity classes, we know logics capturing them on the class of ordered structures.
- ▶ For classes above NP we do not need an explicit order of the structures because we can nondeterministically guess one.
Hence we can capture these classes on arbitrary structures.

State of the Art in Descriptive Complexity Theory

- ▶ For most natural complexity classes, we know logics capturing them on the class of ordered structures.
- ▶ For classes above NP we do not need an explicit order of the structures because we can nondeterministically guess one.
Hence we can capture these classes on arbitrary structures.
- ▶ For classes below PTIME, we only know capturing results on restricted classes of structures.

Gurevich's Conjecture

Conjecture (Gurevich 1985)

There is no logic that captures PTIME.

Gurevich's Conjecture

Conjecture (Gurevich 1985)

There is no logic that captures PTIME.

Corollary (of Fagin's Theorem)

If no logic captures PTIME, then $PTIME \neq NP$.

Dawar's Dichotomy Theorem

Dawar's Dichotomy Theorem (1995)

1. *If no logic captures PTIME, then the class of queries in PTIME is not recursively enumerable.*

Dawar's Dichotomy Theorem

Dawar's Dichotomy Theorem (1995)

1. *If no logic captures PTIME, then the class of queries in PTIME is not recursively enumerable.*
2. *If there is a logic capturing PTIME, then there is a problem that is complete for PTIME under first-order reductions.*

Dawar's Dichotomy Theorem

Dawar's Dichotomy Theorem (1995)

1. *If no logic captures PTIME, then the class of queries in PTIME is not recursively enumerable.*
“The class of PTIME problems eludes any systematic classification.”
2. *If there is a logic capturing PTIME, then there is a problem that is complete for PTIME under first-order reductions.*
“All problems in PTIME are variants or special cases of the same problem.”

Dawar's Dichotomy Theorem

Dawar's Dichotomy Theorem (1995)

1. *If no logic captures PTIME, then the class of queries in PTIME is not recursively enumerable.*
“The class of PTIME problems eludes any systematic classification.”
2. *If there is a logic capturing PTIME, then there is a problem that is complete for PTIME under first-order reductions.*
“All problems in PTIME are variants or special cases of the same problem.”

NP satisfies the analogue of (2).

Dawar's Dichotomy Theorem

Dawar's Dichotomy Theorem (1995)

1. *If no logic captures PTIME, then the class of queries in PTIME is not recursively enumerable.*
“The class of PTIME problems eludes any systematic classification.”
2. *If there is a logic capturing PTIME, then there is a problem that is complete for PTIME under first-order reductions.*
“All problems in PTIME are variants or special cases of the same problem.”

NP satisfies the analogue of (2).

Hence if no logic captures PTIME, then the structure of PTIME is fundamentally different from the structure of NP.

1. Descriptive Complexity Theory

2. Fixed-Point Logics

3. Applications to the Isomorphism Problem

4. Stronger Logics

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Immerman-Vardi Theorem (1982)

FP captures PTIME on the class of all ordered structures.

The Immerman-Vardi Theorem

Fixed-Point Logic

FP = first-order logic + inductive definitions

Immerman-Vardi Theorem (1982)

FP captures PTIME on the class of all ordered structures.

Remark

FP does not capture PTIME on the class of all structures.
There is no sentence in FP distinguishing between sets of even and odd cardinality.

Fixed-point logic with counting

FP+C = extension of FP by counting quantifiers

Fixed-point logic with counting

FP+C = extension of FP by counting quantifiers

Conjecture (Immerman 1987)

FP+C captures PTIME on the class of all structures.

Fixed-point logic with counting

FP+C = extension of FP by counting quantifiers

Conjecture (Immerman 1987)

FP+C captures PTIME on the class of all structures.

Theorem (Cai, Fürer, Immerman 1992)

FP+C does not capture PTIME.

Theorem (Hella, Kolaitis, Luosto 1996)

$FP+C$ captures $PTIME$ *almost everywhere*.

Theorem (Hella, Kolaitis, Luosto 1996)

FP+C captures PTIME almost everywhere.

Theorem

FP+C capture PTIME on the following classes of graphs:

1. *trees (Immerman, Lander 1990)*
2. *classes of graphs of bounded treewidth (G., Mariño 1999)*
3. *planar graphs (G. 1998)*

Classes with excluded minors

Graph G is a **minor** of graph H ($G \preceq H$)

$:\Leftrightarrow G$ obtained from H by deleting edges and vertices
and **contracting** edges



Classes with excluded minors

Graph G is a **minor** of graph H ($G \preceq H$)

$:\iff G$ obtained from H by deleting edges and vertices
and **contracting** edges



H **excluded minor** for class \mathcal{C} of graphs

$:\iff H \not\preceq G$ for any $G \in \mathcal{C}$

Classes with excluded minors

Graph G is a **minor** of graph H ($G \preceq H$)

$:\iff G$ obtained from H by deleting edges and vertices
and **contracting** edges



H **excluded minor** for class \mathcal{C} of graphs

$:\iff H \not\preceq G$ for any $G \in \mathcal{C}$

Theorem (G. 2008)

FP+C captures PTIME on every class of graphs with an excluded minor.

1. Descriptive Complexity Theory
2. Fixed-Point Logics
3. Applications to the Isomorphism Problem
4. Stronger Logics

Colour refinement

Input: Vertex-coloured graph $G = (V, E, C)$, where $C : V \rightarrow \mathbb{N}$

Colour refinement

Input: Vertex-coloured graph $G = (V, E, C)$, where $C : V \rightarrow \mathbb{N}$

Algorithm: Repeat the following steps until **stable colouring** is reached:

1. Extend colour of each vertex by colours of its neighbours and their multiplicities:

$$C'(v) := \left(C(v), \{(c, i) \mid v \text{ has } i \text{ neighbours } w \text{ with } C(w) = c\} \right)$$

Colour refinement

Input: Vertex-coloured graph $G = (V, E, C)$, where $C : V \rightarrow \mathbb{N}$

Algorithm: Repeat the following steps until **stable colouring** is reached:

1. Extend colour of each vertex by colours of its neighbours and their multiplicities:

$$C'(v) := \left(C(v), \{(c, i) \mid v \text{ has } i \text{ neighbours } w \text{ with } C(w) = c\} \right)$$

2. Sort new colours lexicographically and number them.
Let C be the colouring obtained by replacing each C' -colour by its number.

Colour refinement

Input: Vertex-coloured graph $G = (V, E, C)$, where $C : V \rightarrow \mathbb{N}$

Algorithm: Repeat the following steps until **stable colouring** is reached:

1. Extend colour of each vertex by colours of its neighbours and their multiplicities:

$$C'(v) := \left(C(v), \{(c, i) \mid v \text{ has } i \text{ neighbours } w \text{ with } C(w) = c\} \right)$$

2. Sort new colours lexicographically and number them.
Let C be the colouring obtained by replacing each C' -colour by its number.

Analysis

- ▶ at most $|V|$ refinement steps
- ▶ running time $O((|V| + |E|) \cdot \log |V|)$

Isomorphism testing by colour refinement

Input: Graphs G and H

Isomorphism testing by colour refinement

Input: Graphs G and H

Algorithm

1. Compute stable colouring of disjoint union $G \dot{\cup} H$
2. If G and H have same colours with same multiplicities then answer “yes” else answer “no”

Isomorphism testing by colour refinement

Input: Graphs G and H

Algorithm

1. Compute stable colouring of disjoint union $G \dot{\cup} H$
2. If G and H have same colours with same multiplicities then answer “yes” else answer “no”

Correctness

- ▶ Negative answers are always correct.
- ▶ Positive answers may be false.

Isomorphism testing by colour refinement

Input: Graphs G and H

Algorithm

1. Compute stable colouring of disjoint union $G \dot{\cup} H$
2. If G and H have same colours with same multiplicities then answer “yes” else answer “no”

Correctness

- ▶ Negative answers are always correct.
- ▶ Positive answers may be false.

We say that **colour refinement decides isomorphism on class \mathcal{C}** if for all $G, H \in \mathcal{C}$, the answer is correct.

Theorem (Babai, Erdős, and Selkow 1980)

Colour refinement decides isomorphism on almost all graphs.

Theorem (Babai, Erdős, and Selkow 1980)

Colour refinement decides isomorphism on almost all graphs.

Remark

Colour refinement does not work on regular (initially uncoloured) graphs

The Weisfeiler-Leman algorithm

- ▶ Colour refinement can easily be extended to vertex and edge coloured graphs

The Weisfeiler-Leman algorithm

- ▶ Colour refinement can easily be extended to vertex and edge coloured graphs
- ▶ For graph G and $k \geq 2$, let $G^{(k)}$ be the coloured graph with

The Weisfeiler-Leman algorithm

- ▶ Colour refinement can easily be extended to vertex and edge coloured graphs
- ▶ For graph G and $k \geq 2$, let $G^{(k)}$ be the coloured graph with
 - ▶ Vertex set $V(G)^k$, vertices coloured by isomorphism types of labelled induced subgraph

The Weisfeiler-Leman algorithm

- ▶ Colour refinement can easily be extended to vertex and edge coloured graphs
- ▶ For graph G and $k \geq 2$, let $G^{(k)}$ be the coloured graph with
 - ▶ Vertex set $V(G)^k$, vertices coloured by isomorphism types of labelled induced subgraph
 - ▶ Edges of colour i between (v_1, \dots, v_k) and (w_1, \dots, w_k) if $v_j = w_j$ for all $j \neq i$

The Weisfeiler-Leman algorithm

- ▶ Colour refinement can easily be extended to vertex and edge coloured graphs
- ▶ For graph G and $k \geq 2$, let $G^{(k)}$ be the coloured graph with
 - ▶ Vertex set $V(G)^k$, vertices coloured by isomorphism types of labelled induced subgraph
 - ▶ Edges of colour i between (v_1, \dots, v_k) and (w_1, \dots, w_k) if $v_j = w_j$ for all $j \neq i$
- ▶ The **k -dimensional WL algorithmus** applies colour refinement to $G^{(k)}, H^{(k)}$.

Theorem (Cai, Fürer, Immerman 1992)

For every k there are 3-regular graphs G_k, H_k of size $O(k)$ such that the k -dimensional WL algorithm fails to recognize that G_k, H_k are nonisomorphic.

WL on graphs with excluded minors

Theorem (G. 2008)

For every class \mathcal{C} of graphs with excluded minors there is a k such that the k -dimensional WL algorithm decides isomorphism on \mathcal{C} .

WL on graphs with excluded minors

Theorem (G. 2008)

For every class \mathcal{C} of graphs with excluded minors there is a k such that the k -dimensional WL algorithm decides isomorphism on \mathcal{C} .

Remark

The first polynomial time isomorphism algorithm for graph classes with excluded minors is due to Ponomarenko (1988).

Theorem (G. and Verbitsky 2006, Verbitsky 2007)

On planar graphs and graphs of bounded tree width, a logarithmic number of refinement steps of the k -dimensional WL algorithm suffice to distinguish nonisomorphic graphs.

Theorem (G. and Verbitsky 2006, Verbitsky 2007)

On planar graphs and graphs of bounded tree width, a logarithmic number of refinement steps of the k -dimensional WL algorithm suffice to distinguish nonisomorphic graphs.

Corollary

Isomorphism of planar graphs and graphs of bounded tree width is in TC^1 .

1. Descriptive Complexity Theory
2. Fixed-Point Logics
3. Applications to the Isomorphism Problem
4. Stronger Logics

Stronger logics

Both of the following logics are contained in PTIME and strictly more expressive than FP+C:

Both of the following logics are contained in PTIME and strictly more expressive than FP+C:

Fixed-Point Logic with Rank (FP+R)

Extension of FP by operator for computing the rank of matrices over finite fields.

Both of the following logics are contained in PTIME and strictly more expressive than FP+C:

Fixed-Point Logic with Rank (FP+R)

Extension of FP by operator for computing the rank of matrices over finite fields.

Choiceless Polynomial Time with Counting (CP+C)

Programming language operating directly on structures.

Both of the following logics are contained in PTIME and strictly more expressive than FP+C:

Fixed-Point Logic with Rank (FP+R)

Extension of FP by operator for computing the rank of matrices over finite fields.

Choiceless Polynomial Time with Counting (CP+C)

Programming language operating directly on structures. Same expressiveness as variant of fixed-point logic with variables ranging over higher-order objects that can be described with $O(\log n)$ bits.