Algorithmic Meta-Theorems for Restrictions of Treewidth

Michael Lampis

Computer Science Dept.

Graduate Center, City University of New York

Outline

- Introduction and Background
  - Algorithmic Meta-Theorems
  - FO and MSO logic
  - Courcelle's theorem and lower bounds
- Algorithmic Results
  - FO logic for Vertex Cover
  - FO logic for Max-Leaf number
  - MSO logic for Vertex Cover
- Hardness results
  - Lower bounds for Vertex Cover
- Conclusions and further work

Algorithmic Meta-Theorems

- Algorithmic Theorems
  - Vertex Cover, Dominating Set, 3-Coloring are solvable in linear time on graphs of constant treewidth.
  - Vertex Cover, Feedback Vertex Set can be solved in sub-exponential time on planar graphs

Algorithmic Meta-Theorems

- Algorithmic Meta-Theorems
  - All MSO-expressible problems are solvable in linear time on graphs of constant treewidth.
  - All minor closed optimization problems can be solved in sub-exponential time on planar graphs
- Main uses: quick complexity classification tools, mapping the limits of applicability for specific techniques.

Algorithmic Meta-Theorems

- Algorithmic Meta-Theorems
  - All MSO-expressible problems are solvable in linear time on graphs of constant treewidth.
  - All minor closed optimization problems can be solved in sub-exponential time on planar graphs
- Main uses: quick complexity classification tools, mapping the limits of applicability for specific techniques.
- This talk: Algorithmic Meta-Theorems where the class of problems is defined using logic.

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \dots$
  - Edge predicate $E(x, y)$, Equality $x = y$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x,y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example:

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Dominating Set of size 2

$$\exists x_1 \exists x_2 \forall y E(x_1, y) \vee E(x_2, y) \vee x_1 = y \vee x_2 = y$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Vertex Cover of size 2

$$\exists x_1 \exists x_2 \forall y \forall z E(y, z) \rightarrow (y = x_1 \vee y = x_2 \vee z = x_1 \vee z = x_2)$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Clique of size 3

$$\exists x_1 \exists x_2 \exists x_3 E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_1, x_3)$$

First Order Logic on graphs

- We express graph properties using logic
- Basic vocabulary
  - Vertex variables: $x, y, z, \ldots$
  - Edge predicate $E(x, y)$, Equality $x = y$
  - Boolean connectives $\vee, \wedge, \neg$
  - Quantifiers $\forall, \exists$

Example: Many standard (parameterized) problems can be expressed in FO logic. But some easy problems are inexpressible (e.g. connectivity).
Rule of thumb: FO = local properties

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges

Example: 2-coloring

$$\exists V_1 \exists V_2 \forall x \forall y E(x,y) \rightarrow (x \in V_1 \leftrightarrow y \in V_2)$$

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO$_1$ logic: we can quantify over sets of vertices only
  - MSO$_2$ logic: we can quantify over sets of edges
- MSO$_2$ $\neq$ MSO$_1$. Examples: Hamiltonicity, Edge dominating set

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges
- $\text{MSO}_2 \neq \text{MSO}_1$. Examples: Hamiltonicity, Edge dominating set
- Optimization variants of MSO exist, questions of the form find min S s.t. $\phi(S)$ holds.

(Monadic) Second Order Logic

- MSO logic: we add set variables $S_1, S_2, \ldots$ and a $\in$ predicate. We are now allowed to quantify over sets.
  - MSO_1 logic: we can quantify over sets of vertices only
  - MSO_2 logic: we can quantify over sets of edges
- $\mathrm{MSO}_2 \neq \mathrm{MSO}_1$. Examples: Hamiltonicity, Edge dominating set
- Optimization variants of MSO exist, questions of the form find min S s.t. $\phi(S)$ holds.
- SO logic: allows to quantify over relations on vertices, e.g. vertex orderings. All problems in PH are expressible in SO logic.

Logic and Complexity

- Descriptive complexity: look at classes of (fixed) formulas and estimate the complexity of the corresponding problems
  - Most famous result: Fagin's theorem, $\exists$ SO = NP.

Logic and Complexity

- Descriptive complexity: look at classes of (fixed) formulas and estimate the complexity of the corresponding problems
  - Most famous result: Fagin's theorem, $\exists$ SO = NP.
- Drawback: Length and complexity of the formula are not taken into account.

Logic and Complexity

- Descriptive complexity: look at classes of (fixed) formulas and estimate the complexity of the corresponding problems
  - Most famous result: Fagin's theorem, $\exists$ SO = NP.
- Drawback: Length and complexity of the formula are not taken into account.
- If we consider the formula part of the input, then the problem of deciding if a formula holds is PSPACE-complete even for FO logic and trivial graphs!

Logic and Complexity

- Descriptive complexity: look at classes of (fixed) formulas and estimate the complexity of the corresponding problems
  - Most famous result: Fagin's theorem, $\exists$ SO = NP.
- Drawback: Length and complexity of the formula are not taken into account.
- If we consider the formula part of the input, then the problem of deciding if a formula holds is PSPACE-complete even for FO logic and trivial graphs!
- Solution:
  - Use parameterized complexity.
  - The main part of the input is the graph. The parameter is the length of the formula ϕ which describes the problem.

The model checking problem

Problem: **p-Model Checking**
Input: Graph G and formula ϕ
Parameter: $|\phi|$
Question: $G \models \phi$?

- For general graphs, this problem is W-hard even for FO logic

The model checking problem

Problem: **p-Model Checking**
Input: Graph G and formula ϕ
Parameter: $|\phi|$
Question: $G \models \phi$?

- For general graphs, this problem is W-hard even for FO logic
  - 30-second question: Why?

The model checking problem

Problem: **p-Model Checking**
Input: Graph G and formula ϕ
Parameter: $|\phi|$
Question: $G \models \phi$?

- For general graphs, this problem is W-hard even for FO logic
  - 30-second question: Why?
- We are interested in finding tractable, i.e. FPT, cases for more restricted classes of graphs.
- The most famous such result is Courcelle's theorem which states that p-Model Checking for MSO_2 logic is FPT when also parameterized by the graph's treewidth.

The model checking problem

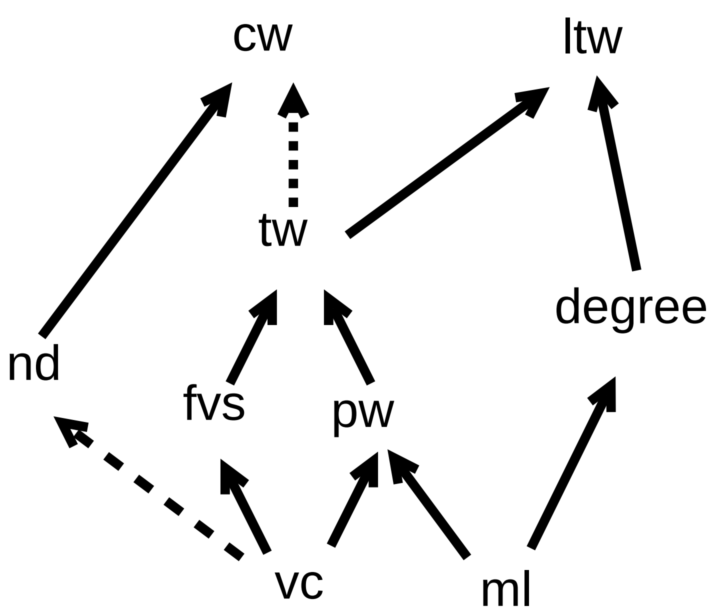
Problem: **p-Model Checking**
Input: Graph G and formula ϕ
Parameter: $|\phi|$
Question: $G \models \phi$?

- For general graphs, this problem is W-hard even for FO logic
  - 30-second question: Why?
- Because the property "the graph has a clique of size k" can be encoded in an FO formula of size $O(k)$
- The problem is in XP though, by the trivial exhaustive algorithm.

Lower Bounds

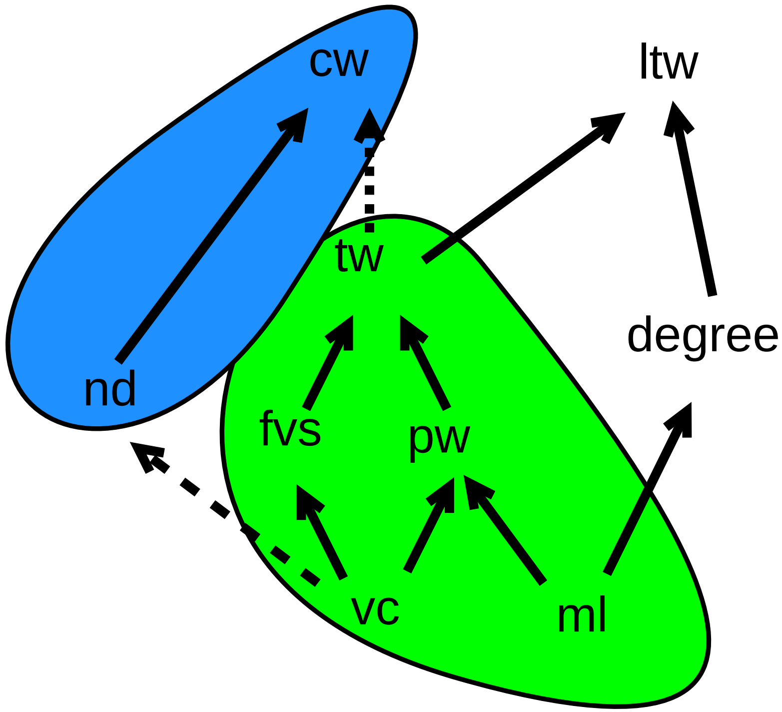
- Courcelle's theorem states that deciding if $G \models \phi$ can be done in time $f(tw(G), \phi) \cdot |G|$, for some function f.
- Unfortunately, in the worst case this function is horrible!
  - [Frick and Grohe 2004]: There is no algorithm which solves p-Model Checking on trees in time $O(f(\phi) \cdot n)$ for any elementary function f unless P=NP.
  - The lower bound applies also to FO logic, under the stronger assumption FPT$\neq$AW[*]
- Motivation: see if things improve when one looks at more restricted classes of graphs.

Graph classes



Some popular graph classes

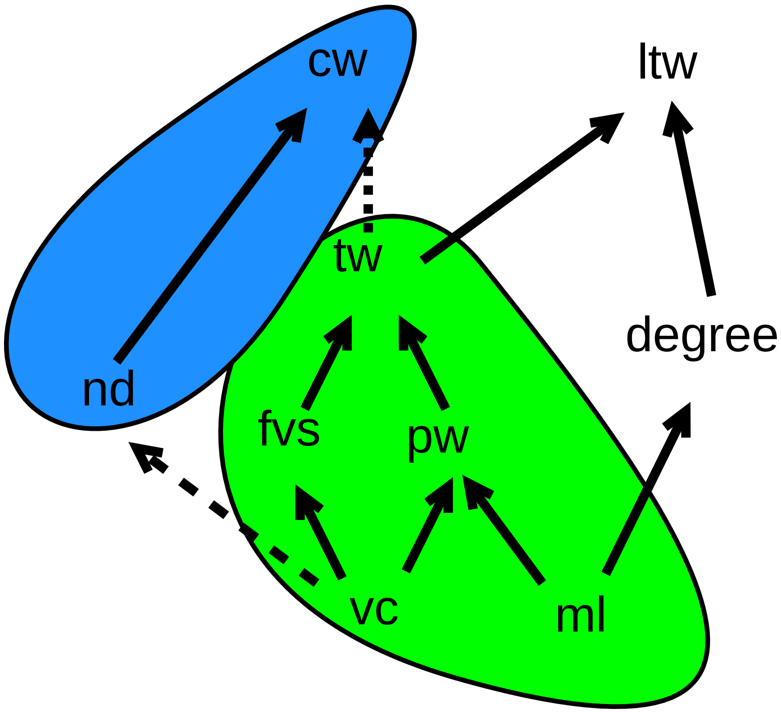
Graph classes



Some popular graph classes

- FO logic is FPT for all, MSO_1 for the blue area, MSO_2 for the green area.
- Lower bounds:
  - FO logic is non-elementary for trees, triply exponential for binary trees.

Graph classes



Some popular graph classes

- FO logic is FPT for all, MSO_1 for the blue area, MSO_2 for the green area.
- Lower bounds:
  - FO logic is non-elementary for trees, triply exponential for binary trees.

Our focus is on improving on the bottom.

Summary of results

- FO logic for graphs of bounded vertex cover is singly exponential
- FO logic for graphs of bounded max-leaf number is singly exponential
- MSO logic for graphs of bounded vertex cover is doubly exponential
- Tight lower bounds (under the ETH) for vertex cover
- Generalize FO and MSO_1 results to neighborhood diversity

Graphs with small Vertex Cover

- A vertex cover is a set of vertices whose removal makes the graph an independent set.
- Usually viewed as just an optimization problem, but the existence of a small vertex cover gives a graph a very special form.
- Small vertex cover trivially implies small treewidth.
- It makes sense to study problems hard for treewidth parameterized by vertex cover
  - Good example: Bandwidth

Vertex cover - A warm-up

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.

Vertex cover - A warm-up

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Intuition:
  - Model checking FO logic on general graphs is in XP: each time we see a quantifier, we try all possible vertices.
  - The existence of a vertex cover of size k partitions the remainder of the graph into at most 2^k sets of vertices, depending on their neighbors in the vertex cover.
  - Crucial point: Trying all possible vertices in a set is wasteful. One representative suffices.

Vertex cover - A warm-up

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Definition: u, v have the same type iff $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.
- Lemma: If $\phi(x)$ is a FO formula with a free variable and u, v have the same type then $G \models \phi(u)$ iff $G \models \phi(v)$.

Vertex cover - A warm-up

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Algorithm: For each of the q quantified vertex variables in the formula try the following
  - Each of the vertices of the vertex cover (k choices)
  - Each of the previously selected vertices (q choices)
  - An arbitrary representative from each type (2^k choices)
- Total time: $O^*(k + q + 2^k)^q = O^*(2^{kq+q\log q})$

Vertex cover - A warm-up

- Model checking FO logic on graphs of bounded vertex cover is singly exponential.
- Algorithm: For each of the q quantified vertex variables in the formula try the following
  - Each of the vertices of the vertex cover (k choices)
  - Each of the previously selected vertices (q choices)
  - An arbitrary representative from each type (2^k choices)
- Total time: $O^*(k+q+2^k)^q = O^*(2^{kq+q\log q})$
- Trivial technique, but singly exponential time. Can we do better?

Max-Leaf Number

- The max-leaf number of graph $ml(G)$ is the maximum number of leaves of any sub-tree of G.
- Again, small max-leaf number implies a special structure
  - Trivially, small degree and small treewidth
  - $[$Kleitman and West$]$ A graph of max-leaf number k is a sub-division of a graph of at most $O(k)$ vertices.
- Again, it makes sense to study problems hard for treewidth parameterized by max-leaf number
  - Good example: Bandwidth

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- This is an important special case of max-leaf number graphs. We cannot use the previous technique since the vertex cover is high.

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- Key intuition: if the path is very long, its precise length does not matter.

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- Lemma: If ϕ has q quantified vertex variables and $n \geq 2^q$ then $P_n \models \phi$ iff $P_{n-1} \models \phi$

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- Lemma: If ϕ has q quantified vertex variables and $n \geq 2^q$ then $P_n \models \phi$ iff $P_{n-1} \models \phi$
  - Proof: Induction on q
  - Suppose that $P_n \models \phi$ when the first quantified variable is mapped to some vertex in the path.
  - We now have two pieces, one of length at least 2^{q-1} and $q-1$ variables left. From the inductive hypothesis, this can be shortened without affecting the outcome of the computation.
  - Therefore the original path can be shortened.

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- Lemma: If ϕ has q quantified vertex variables and $n \geq 2^q$ then $P_n \models \phi$ iff $P_{n-1} \models \phi$
- By applying the lemma, any path can be shortened to size 2^q. Applying the trivial algorithm for FO logic gives a time bound of $O^*(2^{q^2})$

FO logic on paths

- Let us first try to solve this basic problem: Given a path on n vertices and a FO sentence ϕ, decide if ϕ holds on that path.
- Lemma: If ϕ has q quantified vertex variables and $n \geq 2^q$ then $P_n \models \phi$ iff $P_{n-1} \models \phi$
- By applying the lemma, any path can be shortened to size 2^q. Applying the trivial algorithm for FO logic gives a time bound of $O^*(2^{q^2})$
- This is a classic idea related to Ehrenfaucht-Fraisse games in logic.

FO logic for Max-Leaf

- Generalize this idea to graphs of small max-leaf number.

FO logic for Max-Leaf

- Generalize this idea to graphs of small max-leaf number.
- Definition: a topo-edge is a vertex-maximal induced path
- The vast majority of vertices belong in topo-edges

FO logic for Max-Leaf

- Generalize this idea to graphs of small max-leaf number.
- Lemma: If a topo-edge has length at least 2^q it can be shortened without affecting the truth value of any FO sentence with at most q quantifiers.
- Proof: Similar as in the case of paths

FO logic for Max-Leaf

- Generalize this idea to graphs of small max-leaf number.
- The graph can be reduced to size $O(k^2 2^q)$ so the trivial FO algorithm runs in $2^{O(q^2+q\log k)}$

FO logic for Max-Leaf

- Generalize this idea to graphs of small max-leaf number.
- The graph can be reduced to size $O(k^2 2^q)$ so the trivial FO algorithm runs in $2^{O(q^2+q\log k)}$

Again, trivial algorithmic ideas but singly exponential running time.

MSO logic for VC - first attempt

- In MSO logic our formula contains quantified set variables.

MSO logic for VC - first attempt

- In MSO logic our formula contains quantified set variables.
- Trying all possible sets of vertices would of course take time 2^n, which is not allowed.

MSO logic for VC - first attempt

- In MSO logic our formula contains quantified set variables.
- Trying all possible sets of vertices would of course take time 2^n, which is not allowed.
- However, since all vertices of a given type are equivalent, it only matters how many of a given type are selected in a set.

MSO logic for VC - first attempt

- In MSO logic our formula contains quantified set variables.
- Trying all possible sets of vertices would of course take time 2^n, which is not allowed.
- However, since all vertices of a given type are equivalent, it only matters how many of a given type are selected in a set.
- This leads to at most n^{2^k} choices for each set and an algorithm running in time $n^{2^k q}$.

MSO logic for VC - first attempt

- In MSO logic our formula contains quantified set variables.
- Trying all possible sets of vertices would of course take time 2^n, which is not allowed.
- However, since all vertices of a given type are equivalent, it only matters how many of a given type are selected in a set.
- This leads to at most n^{2^k} choices for each set and an algorithm running in time $n^{2^k q}$.
- This isn't even FPT. Must do better...

MSO logic for independent sets

- Let’s try to analyze the simplest possible case for bounded vertex cover graphs.

MSO logic for independent sets

- Let's try to analyze the simplest possible case for bounded vertex cover graphs.
- We are given an empty graph on n vertices and an MSO sentence ϕ and must decide if ϕ holds.

MSO logic for independent sets

- Let's try to analyze the simplest possible case for bounded vertex cover graphs.
- We are given an empty graph on n vertices and an MSO sentence ϕ and must decide if ϕ holds.
- This probably sounds like a really silly problem, but surprisingly it captures the complexity of the problem we are interested in quite well...

MSO logic for independent sets

- Let's try to analyze the simplest possible case for bounded vertex cover graphs.
- We are given an empty graph on n vertices and an MSO sentence ϕ and must decide if ϕ holds.
- This probably sounds like a really silly problem, but surprisingly it captures the complexity of the problem we are interested in quite well...
- Observe that all the vertices are equivalent/have the same type, so there exists a trivial n^q algorithm, corresponding to our previous idea.

MSO logic for independent sets

- Target: we would like to prove a lemma of the form: “if $n > f(q)$ then we can delete some vertices without affecting the outcome”.

MSO logic for independent sets

- Target: we would like to prove a lemma of the form: “if $n > f(q)$ then we can delete some vertices without affecting the outcome”.
- Lemma: For FO logic we can prove this with $f(q) = q$. In other words, FO sentences with q variables cannot distinguish between independent sets of q or more vertices.

MSO logic for independent sets

- Target: we would like to prove a lemma of the form: "if $n > f(q)$ then we can delete some vertices without affecting the outcome".
- Lemma: For FO logic we can prove this with $f(q) = q$. In other words, FO sentences with q variables cannot distinguish between independent sets of q or more vertices.
- FO logic has limited counting power.

MSO logic for independent sets

- Target: we would like to prove a lemma of the form: "if $n > f(q)$ then we can delete some vertices without affecting the outcome".
- Lemma: For FO logic we can prove this with $f(q) = q$. In other words, FO sentences with q variables cannot distinguish between independent sets of q or more vertices.
- FO logic has limited counting power.
- Using this fact we would like to prove that MSO logic also has limited counting power.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S|-1$ vertices for MSO sentences of at most q quantifiers.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S| - 1$ vertices for MSO sentences of at most q quantifiers.
- Proof: We must show that removing one vertex from S makes no difference. Let $S' = S \setminus \{u\}$.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S| - 1$ vertices for MSO sentences of at most q quantifiers.
- Proof: We must show that removing one vertex from S makes no difference. Let $S' = S \setminus \{u\}$.
- Let ϕ be an MSO formula with a free set variable. We must show that $\phi(S) \leftrightarrow \phi(S')$.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S| - 1$ vertices for MSO sentences of at most q quantifiers.
- Proof: We must show that removing one vertex from S makes no difference. Let $S' = S \setminus \{u\}$.
- Let ϕ be an MSO formula with a free set variable. We must show that $\phi(S) \leftrightarrow \phi(S')$.
- The only way a difference could arise is if u is used for a vertex variable.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S| - 1$ vertices for MSO sentences of at most q quantifiers.
- Proof: We must show that removing one vertex from S makes no difference. Let $S' = S \setminus \{u\}$.
- Let ϕ be an MSO formula with a free set variable. We must show that $\phi(S) \leftrightarrow \phi(S')$.
- The only way a difference could arise is if u is used for a vertex variable.
- It is possible to avoid this if there are other vertices with the same type.

MSO logic for independent sets

- Lemma: Let S be a set of vertices such that $|S| > 2^q$ and $|\overline{S}| > 2^q$. Then S is equivalent to any set of $|S| - 1$ vertices for MSO sentences of at most q quantifiers.
- Proof: We must show that removing one vertex from S makes no difference. Let $S' = S \setminus \{u\}$.
- Let ϕ be an MSO formula with a free set variable. We must show that $\phi(S) \leftrightarrow \phi(S')$.
- The only way a difference could arise is if u is used for a vertex variable.
- It is possible to avoid this if there are other vertices with the same type.
- If S has the required size, it is possible to make sure that u is always a member of a type with enough other vertices so that it is never picked.

MSO logic for vertex cover

- Using a more general version of the previous lemma, we can show that there are at most $O(2^q)$ different sets of vertices from each type worth trying.

MSO logic for vertex cover

- Using a more general version of the previous lemma, we can show that there are at most $O(2^q)$ different sets of vertices from each type worth trying.
- There are 2^k different types of vertices. So for a set variable we will try $(2^q)^{2^k}$ different sets.

MSO logic for vertex cover

- Using a more general version of the previous lemma, we can show that there are at most $O(2^q)$ different sets of vertices from each type worth trying.
- There are 2^k different types of vertices. So for a set variable we will try $(2^q)^{2^k}$ different sets.
- In the end we get a $2^{2^{O(k+q)}}$ (doubly exponential) algorithm.

MSO logic for vertex cover

- Using a more general version of the previous lemma, we can show that there are at most $O(2^q)$ different sets of vertices from each type worth trying.
- There are 2^k different types of vertices. So for a set variable we will try $(2^q)^{2^k}$ different sets.
- In the end we get a $2^{2^{O(k+q)}}$ (doubly exponential) algorithm.
- Simple techniques, much better than a tower of exponentials. Can we do better?

MSO logic for vertex cover

- Using a more general version of the previous lemma, we can show that there are at most $O(2^q)$ different sets of vertices from each type worth trying.
- There are 2^k different types of vertices. So for a set variable we will try $(2^q)^{2^k}$ different sets.
- In the end we get a $2^{2^{O(k+q)}}$ (doubly exponential) algorithm.
- Simple techniques, much better than a tower of exponentials. Can we do better?
- Interesting point: here MSO is exponentially worse than FO. Not so for treewidth. . .

Lower Bounds

- Natural question: can doubly exponential be improved to singly exponential?

Lower Bounds

- Natural question: can doubly exponential be improved to singly exponential?
- Also: can the exponents in singly exponential running times (2^{kq}, 2^{q^2}) be improved?

Lower Bounds

- Natural question: can doubly exponential be improved to singly exponential?
- Also: can the exponents in singly exponential running times (2^{kq}, 2^{q^2}) be improved?
- We will show a lower bound argument that will resolve the questions related to vertex cover in a negative way.

Lower Bounds

- Natural question: can doubly exponential be improved to singly exponential?
- Also: can the exponents in singly exponential running times (2^{kq}, 2^{q^2}) be improved?
- We will show a lower bound argument that will resolve the questions related to vertex cover in a negative way.
- Our results will rely on the ETH

Lower Bounds

- Natural question: can doubly exponential be improved to singly exponential?
- Also: can the exponents in singly exponential running times (2^{kq}, 2^{q^2}) be improved?
- We will show a lower bound argument that will resolve the questions related to vertex cover in a negative way.
- Our results will rely on the ETH
- ETH: There is no $2^{o(n)}$ algorithm for 3SAT.

Reduction

- Reduction from 3-SAT to model checking.
- We will create a graph G to encode a propositional formula with n variables.
- G will have vertex cover $O(\log n)$. The MSO formula we will build will have constant size.
- A $2^{2^{o(k+q)}}$ algorithm would then give $2^{2^{o(\log n)}} = 2^{o(n)}$ algorithm for 3SAT.

Reduction

- Create $\log n$ disjoint copies of K_7.
- Create n vertices for the variables. Connect them to one vertex of a K_7 that corresponds to a 1 in the binary representation of the variable's index.
- Create m vertices for the clauses. Connect them in a similar way to the K_7's, encoding also in which position each variable appears and whether it is negated.
- Create an MSO formula that asks for a set of variables corresponding to vertices which satisfy the original formula if set to true.

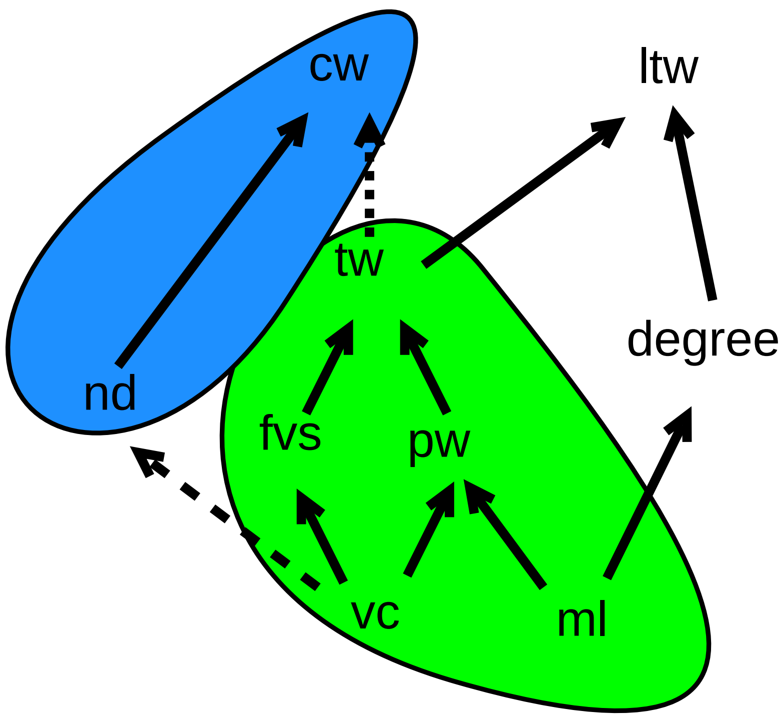
Reduction

- The same reduction can be used to show that no $2^{O(k+q)}$ algorithm is possible for FO logic.
- In this case we start the reduction from the parameterized problem Weighted 3-SAT.
- The part of the formula which asks for a set of vertices is replaced by w existentially quantified vertex variables.
- A $2^{O(k+q)}$ algorithm now gives an FPT algorithm for this problem.

Neighborhood diversity

- We have seen that we can prove stronger meta-theorems for bounded vertex cover than we can for bounded treewidth.
- However, we are essentially only using one property of bounded vertex cover graphs: the fact that vertices can be partitioned into a small number of types.
- This motivates the following definition:
  - The neighborhood diversity of a graph is the minimum number $nd(G)$ s.t. the vertices of G can be partitioned in $nd(G)$ sets with all vertices in each set having the same type.
- Observe that this is a strict superset! Example: complete bipartite graphs.

Graph classes



- Neighborhood diversity is a special case of clique-width but incomparable to treewidth.
- Our results for FO logic and MSO_1 logic can trivially be extended to nd.
- MSO_2 is FPT for vertex cover (Courcelle) but W-hard for clique-width. What about nd?

$\mathbf{MSO_2}$

- We would like to extend our technique to handle edge sets.
- Can we partition the set of edges into a few equivalence classes as we did with vertices?
  - Not so straightforward... An edge is not fully characterized by the type of its endpoints.
- However, there exists a simple work-around:
  - Remember that all edges touch k specific vertices.
  - Every edge set can be partitioned into k parts, which are fully characterized by the set of the second endpoints of the edges.
- Corollary: MSO_2 can also be solved in doubly exponential parameter dependence for bounded vertex cover.

MSO_2 for nd

- This trick does not help with the case of neighborhood diversity.
- If we cannot extend our algorithms from below, can we extend our hardness results from above?
  - [Fomin et al. 2009] Hamiltonicity, Edge dominating set and Graph coloring are W-hard parameterized by clique-width.
- (Un)Fortunately, all three are FPT parameterized by nd.
  - Intuition: in graphs of small nd vertices are partitioned into a few groups of independent sets or cliques.
  - These are either disconnected or fully connected to each other.

Conclusions - Open problems

- Stronger meta-theorems (and some lower bounds) for restrictions of treewidth.
- Interesting to continue this line of work for other such graph classes or for other logics.
- More concrete open problems:
  - MSO_2 for nd
  - Lower bound for FO on max-leaf
  - MSO for max-leaf…

MSO for max-leaf

- Observe that our techniques for vertex cover also apply if someone gives us a “colored graph”: just include this information in the concept of vertex types.
- What if someone asks us to model-check an MSO sentence on a colored path?
- Not hard to see: this is similar to model-checking on a string
  - Classical result from automata theory: MSO logic on strings = Regular languages
  - But parameter dependence is a tower of exponentials!
- Maybe a completely different idea?

Thank you!