Model Checking Lower Bounds for Simple Graphs

Michael Lampis
KTH Royal Institute of Technology

July 8th, 2013

Algorithmic Meta-Theorems

Positive results

- Problem X is tractable.

Negative results

- Problem X is hard.

Algorithmic Meta-Theorems

Positive results

- Problem X is tractable.

Negative results

- Problem X is hard.

- An algorithmic meta-theorem is a statement of the form:

  "All problems in a class C are tractable"

Positive results

- Problem X is tractable.

Negative results

- Problem X is hard.

- An algorithmic meta-theorem is a statement of the form:

  "All problems in a class C are tractable"

- Meta-theorems are great! (more in a second)

Algorithmic Meta-Theorems

Positive results

- Problem X is tractable.

Negative results

- Problem X is hard.

- An algorithmic meta-theorem is a statement of the form:

  “All problems in a class C are tractable”

- Meta-theorems are great! (more in a second)

Main objective of today’s talk: barriers to meta-theorems:

“There exists a problem in class C that is hard”

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

  Example:

  $$\exists S \forall x \forall y E(x,y) \to (x \in S \leftrightarrow y \notin S)$$

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?
  - More graphs?
  - Wider classes of problems?
  - Faster?

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?
  - More graphs? ✓
  - Wider classes of problems?
  - Faster?

  Meta-theorems for clique-width, local treewidth,. . .

Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?

  - More graphs? ✓
  - Wider classes of problems? ✓
  - Faster?

  This can be extended to optimization versions of MSO.

Good news so far

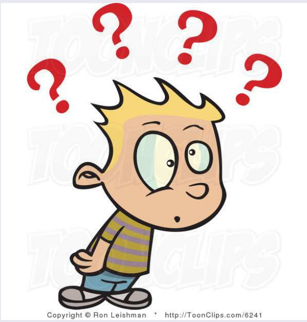
- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?

  - More graphs? ✓
  - Wider classes of problems? ✓
  - Faster? ?

  Faster than linear time?



Good news so far

- Most famous meta-theorem: Courcelle's theorem

  All MSO-expressible properties are solvable in linear time on graphs of bounded treewidth.

- Can we do better?

  - More graphs? ✓
  - Wider classes of problems? ✓
  - Faster? ?

  Faster than linear time?

This is the main question we are concerned with today.

Some bad news

- Courcelle's theorem:

  There exists an algorithm which, given an MSO formula ϕ and a graph G with treewidth w decides if $G \models \phi$ in time $f(w, \phi)|G|$.

Some bad news

- Courcelle's theorem:

  There exists an algorithm which, given an MSO formula ϕ and a graph G with treewidth w decides if $G \models \phi$ in time $f(w, \phi)|G|$.

- But the function f is a tower of exponentials!

Some bad news

- Courcelle's theorem:

  There exists an algorithm which, given an MSO formula ϕ and a graph G with treewidth w decides if $G \models \phi$ in time $f(w, \phi)|G|$.

- But the function f is a tower of exponentials!

- Unfortunately, this is not Courcelle's fault.

  Thm: If $G \models \phi$ can be decided in $f(w, \phi)|G|^c$ for elementary f then P=NP. [Frick & Grohe '04]

Some bad news

- Courcelle's theorem:

  There exists an algorithm which, given an MSO formula ϕ and a graph G with treewidth w decides if $G \models \phi$ in time $f(w, \phi)|G|$.

- But the function f is a tower of exponentials!

- Unfortunately, this is not Courcelle's fault.

  Thm: If $G \models \phi$ can be decided in $f(w, \phi)|G|^c$ for elementary f then P=NP. [Frick & Grohe '04]

- In fact, Frick and Grohe's lower bound applies to FO logic on trees!

There is still hope

This is bad! Can we somehow escape the Frick and Grohe lower bound?

There is still hope

This is bad! Can we somehow escape the Frick and Grohe lower bound?

There is still hope

This is bad! Can we somehow escape the Frick and Grohe lower bound?

Recently, a series of meta-theorems that evade it give "better" parameter dependence.

- For vertex cover, neighborhood diversity, max-leaf [L. '10]
- For twin cover [Ganian '11]
- For shrub-depth [Ganian et al. '12]
- For tree-depth [Gajarský and Hliňený '12]

There is still hope

This is bad! Can we somehow escape the Frick and Grohe lower bound?

Recently, a series of meta-theorems that evade it give “better” parameter dependence.

- For vertex cover, neighborhood diversity, max-leaf [L. ’10]
- For twin cover [Ganian ’11]
- For shrub-depth [Ganian et al. ’12]
- For tree-depth [Gajarský and Hliňený ’12]

Predominant idea: Removing isomorphic parts of the graph, when we have too many

There is still hope

This is bad! Can we somehow escape the Frick and Grohe lower bound?

Recently, a series of meta-theorems that evade it give “better” parameter dependence.

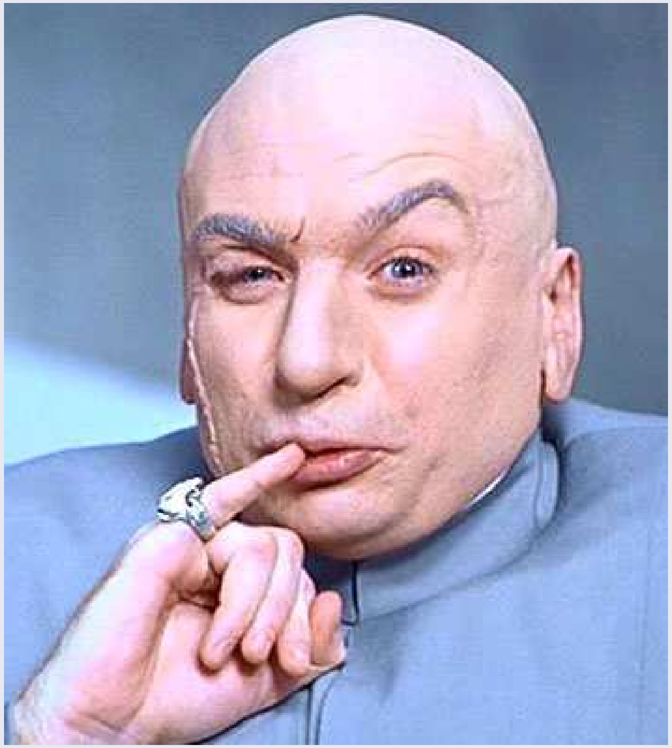
- For vertex cover, neighborhood diversity, max-leaf [L. ’10]
- For twin cover [Ganian ’11]
- For shrub-depth [Ganian et al. ’12]
- For tree-depth [Gajarský and Hliňený ’12]

Predominant idea: Removing isomorphic parts of the graph, when we have too many

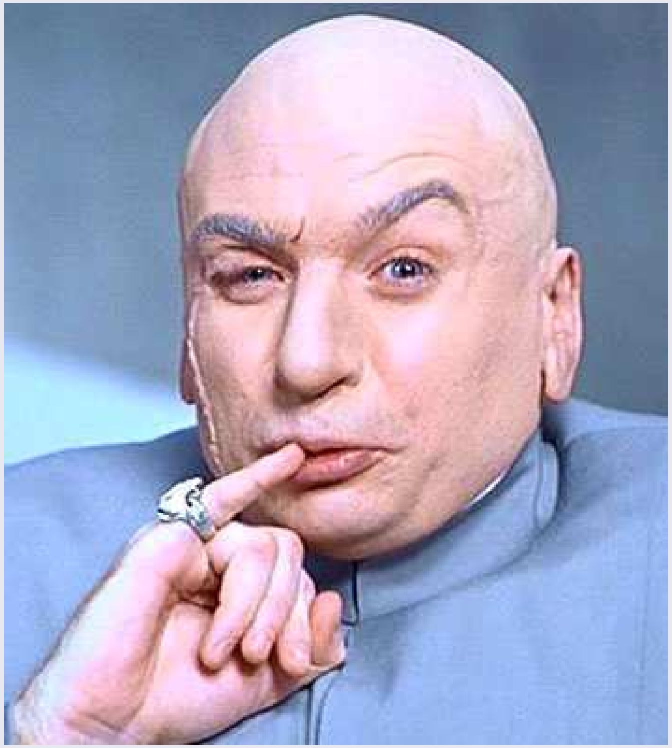
What’s next?

Let's destroy all hope!

- In this talk the pendulum swings again.
- Main goal: prove hardness results even more devastating than Frick& Grohe.
- Motivation: If we know what we can't do, we might find things we can do.



Let's destroy all hope!



- In this talk the pendulum swings again.
- Main goal: prove hardness results even more devastating than Frick& Grohe.
- Motivation: If we know what we can't do, we might find things we can do.

Today: Three new hardness results.

- Threshold graphs
- Paths
- Bounded-height trees

Threshold Graphs

More background

Theorem:

- MSO$_1$ expressible properties can be decided in linear time on graphs of bounded clique-width [Courcelle, Makowsky, Rotics ’00]

More background

Theorem:

- MSO_1 expressible properties can be decided in linear time on graphs of bounded clique-width [Courcelle, Makowsky, Rotics ’00]

- Trees have clique-width 3.

  Frick&Grohe $\rightarrow$ non-elementary dependence.

- Graphs with clique-width 1 are easy for MSO_1.

More background

Theorem:

- MSO_1 expressible properties can be decided in linear time on graphs of bounded clique-width [Courcelle, Makowsky, Rotics '00]

- Trees have clique-width 3.
  Frick&Grohe $\rightarrow$ non-elementary dependence.

- Graphs with clique-width 1 are easy for MSO_1.

What about clique-width 2?

Threshold Graphs

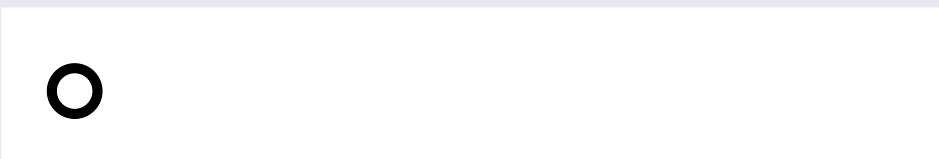
A graph is a threshold graph if it can be constructed with the following operations:

- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.

Threshold Graphs

A graph is a threshold graph if it can be constructed with the following operations:

- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.



u

Threshold Graphs

A graph is a threshold graph if it can be constructed with the following operations:

- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.

uj

Threshold Graphs

A graph is a threshold graph if it can be constructed with the following operations:

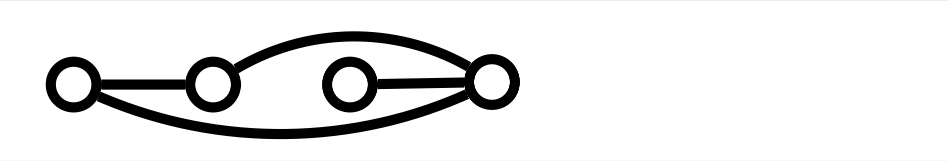
- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.

uju

Threshold Graphs

A graph is a threshold graph if it can be constructed with the following operations:

- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.

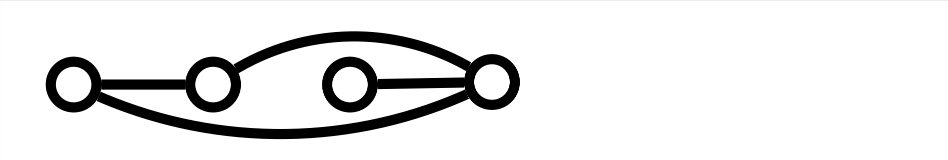


$$ujuj$$

Threshold Graphs

A graph is a threshold graph if it can be constructed with the following operations:

- Add a new vertex and connect it to everything.
- Add a new vertex and connect it to nothing.



$ujuj$

Thm: Threshold graphs have clique-width 2.

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- w :
- G : uuj

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- w : 0
- G : $uuj\ uj$

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- $w :$ 0 1
- $G :$ $uuj\ uj \quad ujj$

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- w : 0 1 1
- $G : \; uuj \; uj \quad ujj \quad ujj$

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- $w:$ 0 1 1 0...
- $G: \; uuj \; uj \quad ujj \quad ujj \quad uj \ldots$

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- w : 0 1 1 0...
- G : $uuj\ uj\ \ ujj\ \ ujj\ \ uj\ldots$

This allows us to interpret the string property as a graph question.

Hardness for threshold graphs

We use the following result of Frick& Grohe:

- There is no elementary-dependence model-checking algorithm for FO logic on binary strings.

Given a string w we construct a threshold graph G

- $w:$ 0 1 1 0...
- $G: \; uuj \; uj \quad ujj \quad ujj \quad uj\ldots$

This allows us to interpret the string property as a graph question.

Thm: There is no elementary-dependence model-checking algorithm for FO logic on threshold graphs.

Consequences

Recall some of the “good” graph classes we know

- Some are closed under complement (neighborhood diversity, shrub-depth)
- Some are closed under union (tree-depth)

Consequences

Recall some of the "good" graph classes we know

- Some are closed under complement (neighborhood diversity, shrub-depth)
- Some are closed under union (tree-depth)
- None are closed under both operations...

Any class of graph closed under both operations must contain threshold graphs.

Paths

Why paths?

Main question:

- Is there an elementary-dependence algorithm for MSO_1 on paths?

Equivalent question:

- Is there an elementary-dependence algorithm for MSO_1 on unary strings?

Why?

- Do Frick and Grohe really need all trees?
- FO is easy on paths.
- MSO is hard on binary strings/colored paths.
- MSO for max-leaf is open!

Why would this be easy?

- MSO on paths = Regular language over unary alphabet
- FO is easy
- Reduction seems impossible...

"Normal" reduction:

- Start with n-variable 3-SAT
- Construct graph G with $|G| = n^c$
- Construct formula ϕ with $|\phi| = \log^* n$
- Prove YES instance $\leftrightarrow G \models \phi$

Problem: New instance would be encodable with $O(\log n)$ bits. We are making a sparse NP-hard language!

How the reduction can work

Key idea: do not use P$\neq$NP but EXP$\neq$NEXP

- Motivation: reduction must construct exponential-size graph, so should be allowed exponential time.

How the reduction can work

Key idea: do not use P$\neq$NP but EXP$\neq$NEXP

- Motivation: reduction must construct exponential-size graph, so should be allowed exponential time.

Plan:

- Start with an NEXP-complete problem and n bits of input.
- Construct a path on 2^{n^c} vertices.
- Construct a formula ϕ with $|\phi| = \log^* n$.
- Prove YES instance $\leftrightarrow G \models \phi$.

Elementary parameter dependence gives EXP=NEXP.

How the reduction can work

Key idea: do not use P$\neq$NP but EXP$\neq$NEXP

- Motivation: reduction must construct exponential-size graph, so should be allowed exponential time.

Plan:

- Start with an NEXP-complete problem and n bits of input.
- Construct a path on 2^{n^c} vertices.
- Construct a formula ϕ with $|\phi| = \log^* n$.
- Prove YES instance $\leftrightarrow G \models \phi$.

Elementary parameter dependence gives EXP=NEXP.

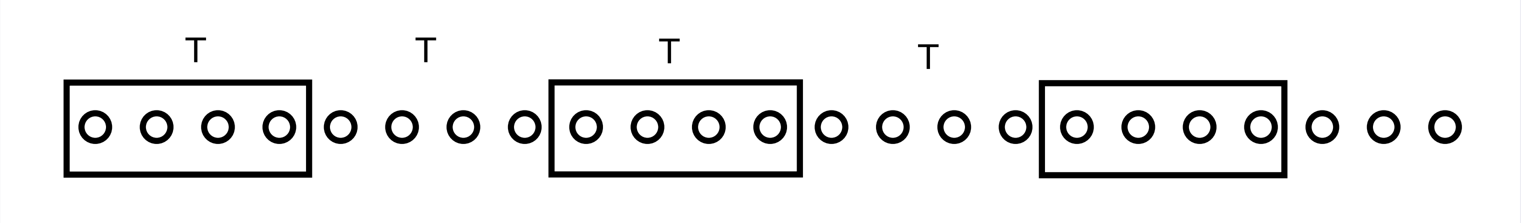
- Formula will be somewhat larger, but still small enough.

Rough sketch

- Start with an NEXP Turing machine, n bits of input. Does it accept?
- The machine runs in time $T = 2^{n^c}$.
- Is there a transcript (of length T^2) that proves acceptance?

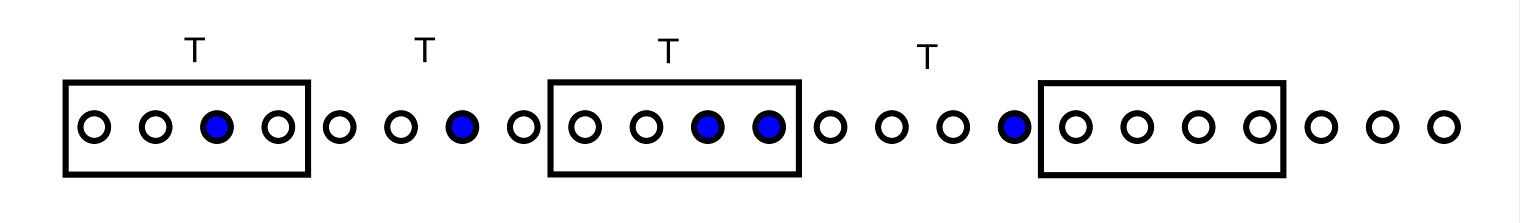
Rough sketch

- Start with an NEXP Turing machine, n bits of input. Does it accept?
- The machine runs in time $T = 2^{n^c}$.
- Is there a transcript (of length T^2) that proves acceptance?



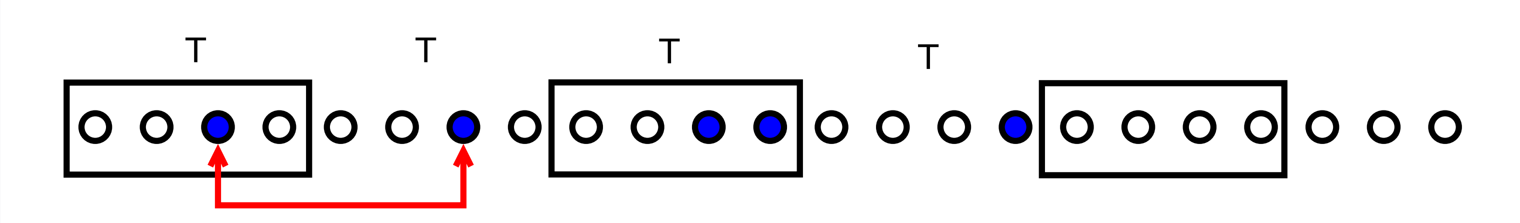
Rough sketch

- Start with an NEXP Turing machine, n bits of input. Does it accept?
- The machine runs in time $T = 2^{n^c}$.
- Is there a transcript (of length T^2) that proves acceptance?



Rough sketch

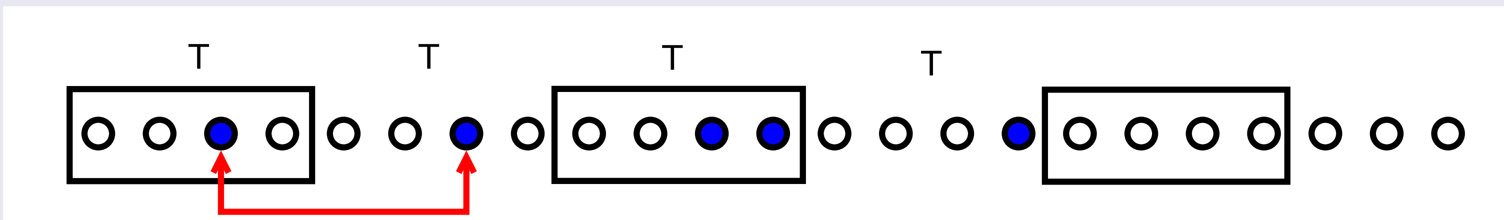
- Start with an NEXP Turing machine, n bits of input. Does it accept?
- The machine runs in time $T = 2^{n^c}$.
- Is there a transcript (of length T^2) that proves acceptance?



- We have to be able to express the property “these vertices are at distance T”.
- We have to do it with $\log^* n$ quantifiers.

Rough sketch

- Start with an NEXP Turing machine, n bits of input. Does it accept?
- The machine runs in time $T = 2^{n^c}$.
- Is there a transcript (of length T^2) that proves acceptance?



- We have to be able to express the property “these ver
  distance T”.

- We have to do it with $\log^* n$ quantifiers.
- This is possible by encoding counting in binary. . .

Consequences

Unless EXP=NEXP:

- Max-leaf is hard
- Graph classes closed under edge sub-divisions are hard
- Graph classes closed under induced subgraphs with unbounded (dense)* diameter are hard

Trees of bounded height

Why trees of bounded height?

This class of graphs is important for two recent meta-theorems:

- Shrub-depth in "When trees grow low: Shrubs and fast MSO_1" [Ganian et al. MFCS '12]
- Tree-depth in "Faster deciding MSO properties of trees of fixed height, and some consequences" [Gajarský and Hliněný FSTTCS '12]

In both cases the main tool is the following:

> MSO model-checking for q quantifiers on trees of height h colored with t colors can be done in $\exp^{(h+1)}(O(q(t+q))$ time.

Lower bound sketch

Thm: $h + 1$ levels of exponentiation are exactly necessary.

Rough idea: use Frick& Grohe proof for trees, use (few colors) to cut down their height.

- Start from an n-variable 3-SAT instance.
- Construct a tree of height h. Use $t = \log^{(h)}(n)$ colors.
- Construct a formula with $q = O(h)$ quantifiers.
- Prove equivalence between instances.

Lower bound sketch

Thm: $h+1$ levels of exponentiation are exactly necessary.

Rough idea: use Frick& Grohe proof for trees, use (few colors) to cut down their height.

- Start from an n-variable 3-SAT instance.
- Construct a tree of height h. Use $t = \log^{(h)}(n)$ colors.
- Construct a formula with $q = O(h)$ quantifiers.
- Prove equivalence between instances.

Argument: an algorithm running in $\exp^{(h+1)}(o(t))$ would run in $2^{o(n)}$ here, disproving ETH.

Conclusions - Open problems

- Three natural barriers to future improvements.
- Paths are probably the toughest to work around.

Future work

- (Uncolored) tree-depth?
- Height of tower for paths?

Conclusions - Open problems

- Three natural barriers to future improvements.
- Paths are probably the toughest to work around.

Future work

- (Uncolored) tree-depth?
- Height of tower for paths?
- Other logics?!?

Thank you!