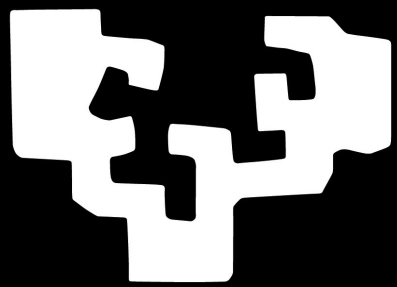
A Personal Perspective on SAT and CSP: Tractability, Complexity, Quantification, Proofs

Hubie Chen
Univ. del País Vasco & Ikerbasque
San Sebastián, Spain

Beyond NP Workshop, Paris, May 2017



This talk

Talk will be a personal, biased, editorialized survey of some theoretical results on SAT, CSP & quantified versions

I will try to highlight conceptual aspects and high-level ideas, and present open questions/directions

SAT & QSAT

SAT: given a CNF formula, decide if $\exists$ a satisfying assignment

Example: $(\ldots(u \vee \overline{v} \vee x) \wedge (y \vee \overline{u} \vee z \vee \overline{w})\ldots)$

QSAT: given a CNF formula where all variables are quantified, decide if true

Example: $\exists x_1 \exists x_2 \forall y_3 \exists x_4 (\ldots (x_2 \vee y_3 \vee x_4) \wedge (\overline{x_4} \vee \overline{x_1})\ldots)$

Remark (SAT as QSAT): SAT can be viewed as the case of QSAT where all variables are $\exists$-quantified

CSP — traditional definition

Defs: Let B be a "domain" (a set)

- A constraint is a pair $((v_1, \ldots, v_k), R)$
  where $R \subseteq B^k$ is a relation;
  satisfied by $f : V \to B$ if $(f(v_1), \ldots, f(v_k)) \in R$
- CSP: given a set of constraints, decide if $\exists$ a
  satisfying assignment (assignment satisfying all constraints)

Remark (SAT as CSP): SAT can be viewed as CSP,
take $B = \{0, 1\}$ and introduce a constraint for each clause

Example: $(u \vee \overline{v} \vee x) \rightsquigarrow ((u, v, x), \{0, 1\}^3 \setminus \{(0, 1, 0)\})$

Oops: If we have a clause of length k, doesn't this translation
involve computing $\{0, 1\}^k$ (minus a tuple)? Yes...

- In CSP/DB theory, typical to use positive representation
  — relation is explicit list of included tuples
- SAT as CSP: suggests a negative representation
  — relation is explicit list of excluded tuples
- This talk: generally won't matter, arity k "bounded"

CSP/QCSP — logical definitions

Instance of CSP consists of two parts...

- csp-sentence: first-order logical sentence built from $\{\exists, \wedge\}$
  Form: $\exists x_1 \ldots \exists x_n (\cdots \wedge S(x_2, x_4, x_5) \wedge T(x_1, x_2) \wedge \cdots)$
- structure: $\mathbf{B} = (B; S^{\mathbf{B}}, T^{\mathbf{B}}, \ldots)$ consists of domain B and an *interpretation* of each relation symbol,
  so for example $S^{\mathbf{B}} \subseteq B^3$, $T^{\mathbf{B}} \subseteq B^2$, ...

CSP, logical def: Given a csp-sentence ϕ and a structure $\mathbf{B}$, decide if ϕ is true on $\mathbf{B}$

- Specification of which variables are constrained together is separated from definition of relations
- DB theorist: CSP $\equiv$ conjunctive query evaluation
  $\equiv$ conjunctive query containment
- Graph theorist: CSP $\equiv$ relational homomorphism problem, generalization of (di)graph homomorphism problem

QCSP, logical def: similar, but consider qcsp-sentences, built from $\{\forall, \exists, \wedge\}$

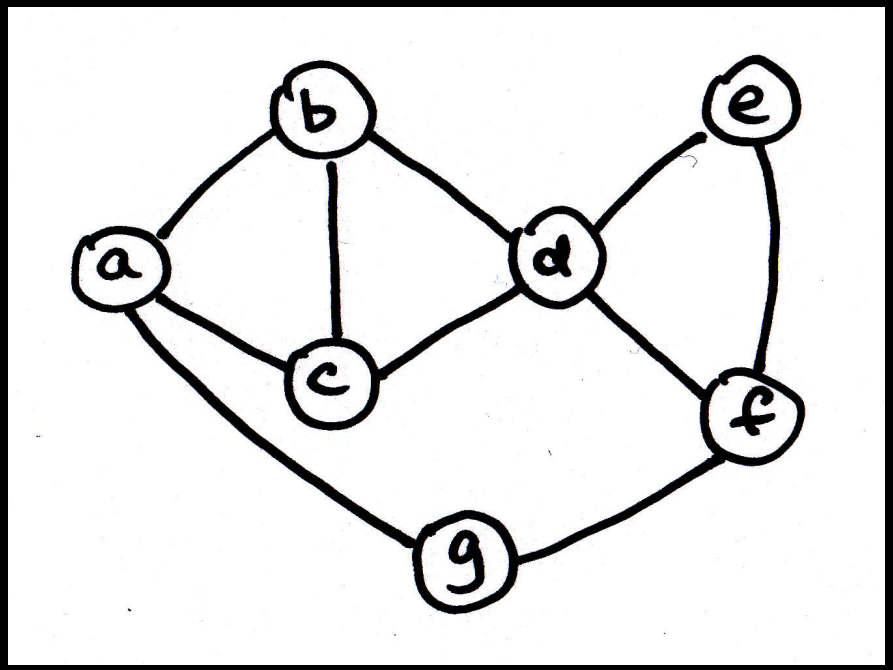
Plan for rest of talk

- Tractability and complexity for CSP/SAT
  - Study impact of sentence on complexity
- ...and for QCSP/QSAT
- Proof complexity for quantified formulas

Act: Tractability and Complexity

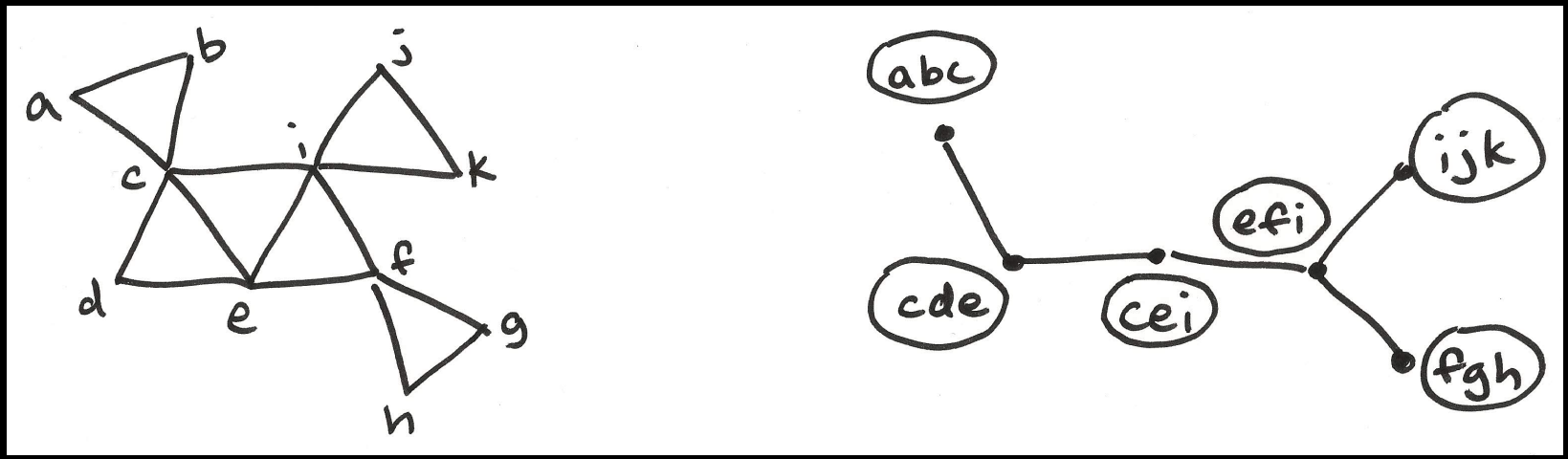
Primal graph

$$\exists a \exists b \exists c \exists d \exists e \exists f \exists g \quad (R(a,b,c) \wedge S(b,c,d) \wedge T(d,e,f) \wedge U(a,g) \wedge W(f,g))$$



Treewidth

Idea: the treewidth of a graph is a natural number
that measures how tree-like a graph is,
lower numbers indicating higher similarity to a tree



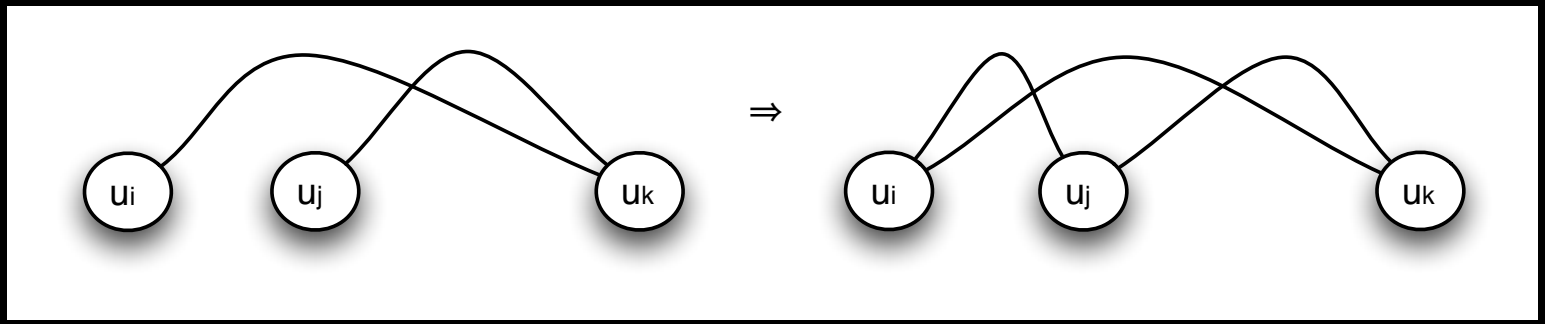
Def: width of a decomposition $\approx$ max size of a bag needed

Def: graph has treewidth $\leqslant k$ iff
has decomposition of width $\leqslant k$

Treewidth via elimination orderings

An elimination ordering of a graph $G = (V, E)$ is a set $E' \supseteq E$ and an ordering $u_1, \ldots, u_n$ of V such that:

- if $i < k$, $j < k$, $\{u_i, u_k\} \in E'$ and $\{u_j, u_k\} \in E'$, then $\{u_i, u_j\} \in E'$



Width of an elimination ordering: $\max_i\{j \mid j < i, \{u_j, u_i\} \in E'\}$

Fact: Graph has treewidth $\leqslant k$ iff
has elimination ordering of width $\leqslant k$

CSP under bounded treewidth

If we impose a constant bound on treewidth of primal graphs, can solve CSP in poly-time [Freuder, AAAI '90]:

(1) compute decomposition/elimination ordering

(2) dynamic programming:

iteratively eliminate the last variable v by joining all constraints with v into one constraint, then projecting out v

...constraints never get big, due to treewidth bound!

Remark: In dynamic programming phase, can even *count* number of solutions

Dichotomy for primal graphs

Thm (Grohe, Schwentick & Segoufin, STOC '01):

Let Φ be any class of csp-sentences ($\{\exists, \wedge\}$),
where each relation symbol is binary and occurs at most once

- If the primal graphs of Φ have bounded treewidth,
  Φ-CSP is poly-time tractable
- Else, Φ-CSP is NOT poly-time tractable,
  unless complexity class collapse occurs

(Example sentence: $\exists v \exists x \exists y \exists z (Q(v,x) \wedge R(x,y) \wedge S(y,z))$.
Binary signature & "self-join free", for database theorist)

Corollary: for any class of primal graphs with unbd treewidth,
can prove hardness of CSP via such sentences

Grohe's dichotomy theorem

Thm (Grohe, JACM '07):

Let Φ be any class of csp-sentences ($\{\exists, \wedge\}$)
having bounded arity.

- If the primal graphs of *cores of* Φ have bounded treewidth,
  Φ-CSP is fixed-parameter tractable (FPT)
- Else, Φ-CSP is NOT fixed-parameter tractable,
  unless complexity class collapse occurs

Remarks:

- Core of csp-sentence ϕ: semantically minimized version
- Here, FPT means in poly-time after
  an arbitrary computation on ϕ
- Algorithm: compute core,
  then apply algorithm for bd treewidth

Generalizations

- Dichotomy without assumption of bounded arity
  — introduces new width measure for hypergraphs
  [Dániel Marx, JACM ’13]

- Fine classification of csp-sentences
  up to very weak reduction
  [Chen & Müller, PODS ’13/CSL-LICS ’14]

- Trichotomy theorem for *counting answers* to csp-formulas
  (with free variables) & unions thereof
  [Chen & Mengel, ICDT ’15/PODS ’16]

  Building on [Durand & Mengel, ICDT ’13], ...

Running time

We use $\tilde{O}(\cdot)$ notation to suppress factors of form $\text{poly}(n)$

Running time of CSP, where
$n = \#$ variables, $d =$ domain size, $k =$ treewidth bound ± 1:

- $\tilde{O}(2^{\text{poly}(k)})$ time — compute decomposition
- $\tilde{O}(d^k)$ time — dynamic programming

SAT in FPT time $\tilde{O}(2^{\text{poly}(k)}) + \tilde{O}(2^k)$, with param $k =$ treewidth

CSP is NOT FPT via this algorithm since may need time $\approx n^k$

($\exists$ evidence that CSP not FPT at all with param treewidth)

Parameter $=$ treewidth: sufficient condition for SAT to be FPT

Dichotomies for SAT?

SAT differs from “usual” CSP in two respects:

- bounded domain size
- negative representation of relations

Sophisticated positive, algorithmic results for SAT:
for example, [Ordyniak, Paulusma & Szeider, TCS ’13], [Capelli, Durand & Mengel, SAT ’14]

Questions: Is it possible to prove any dichotomy theorems?
What if we focus on just one of the differences?

[Chen & Grohe, ’10]:
Study of CSP under alternative succinct representations (without bounding domain size)

e.g. “DNF-like” representation generalizing clauses

Proof systems

For a SAT/CSP instance...

- Certifying satisfiability: easy — give satisfying assignment!
- Certifying unsatisfiability: not possible with poly-time checkable, poly-size proofs, unless NP = coNP

CSP proof system (Atserias, Kolaitis & Vardi, CP '04):
proof for a CSP instance is a sequence of constraints,

where each is: original constraint, join of previous constraints, projection of previous constraint

Fact: For unsat CSP instances having treewidth $< k$,
have proofs with width $\leqslant k$, poly-size!

- Width $\leqslant k$ means each constraint involves $\leqslant k$ variables

Fact: For unsat SAT instances having treewidth $< k$,
have resolution proofs with width $\leqslant k$, poly-size!

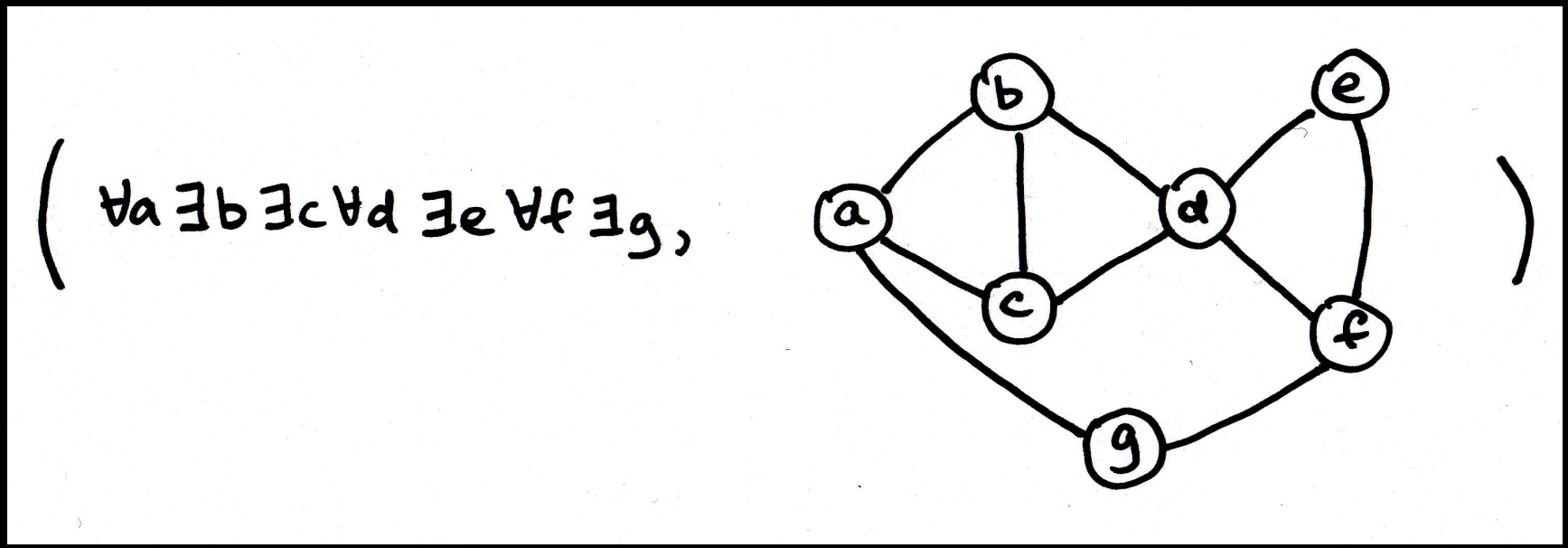
Remark: bd width proofs $\equiv$ "interpretation" of k-consistency

Act: Quantification

Prefixed graph

$$\forall a \exists b \exists c \forall d \exists e \forall f \exists g \quad (R(a,b,c) \wedge S(b,c,d) \wedge T(d,e,f) \wedge U(a,g) \wedge W(f,g))$$

$$\downarrow$$



Dichotomy for prefixed graphs

Generalizes result of (Grohe, Schwentick & Segoufin)
on primal graphs

Thm (Chen & Dalmau, LICS '12):

Let Φ be any class of qcsp-sentences ($\{\forall, \exists, \wedge\}$),
where each relation symbol is binary and occurs at most once

- If the prefixed graphs of Φ have bounded Q-width,
  Φ-QCSP is poly-time tractable
- Else, Φ-QCSP is NOT poly-time tractable,
  unless complexity class collapse occurs

Corollary: for any class of prefixed graphs with unbd Q-width,
can prove hardness of QCSP via such sentences

Q-width

Def: An elimination ordering of a prefixed graph $(P, (V, E))$ is a pair $(E', u_1, \ldots, u_n)$ where $E' \supseteq E$ and $u_1, \ldots, u_n$ is an ordering of V such that, for all u_i, u_j:

- If u_k is $\exists$, $i < k, j < k$ and $\{u_i, u_k\}, \{u_j, u_k\} \in E'$, then $\{u_i, u_j\} \in E'$
- If $\{u_i, u_j\} \in E'$, u_i is $\forall$, u_j is $\exists$ and $u_i <_P u_j$, then $i < j$
- If u_i is $\exists$, u_j is $\forall$ and $u_i <_P u_j$, then $i < j$

Def: Width of an elimination ordering: max of $\{j \mid j < i, \{u_j, u_i\} \in E'\}$ over all $\exists$-variables u_i

Def: Prefixed graph has Q-width $\leqslant k$ if exists elim. ordering of width $\leqslant k$

Observations:

- Q-width $=$ treewidth, if the prefix P is purely $\exists$
- $\forall/\exists$ variables treated in an asymmetric fashion!

Generalization...

Can we generalize this theorem, to classify any class of qcsp-sentences having bounded arity?

That is, can we give a QCSP analog of Grohe's theorem?

Oops: Uf... this is an OPEN QUESTION,
for standard first-order logic!

Perhaps a notion of "core" for qcsp-sentences would help!

Known: Containment/entailment for qcsp-sentences is decidable [Chen, Madelaine & Martin, LICS '08/LMCS '15]

Related: Dichotomy for "block-sorted" qcsp-sentences, in multi-sorted first-order logic
[Chen & D. Marx, ICALP '13]

Running time

We use $\tilde{O}(\cdot)$ notation to suppress factors of form $\text{poly}(n)$

Running time of QCSP, where
$n = \#$ variables, $d =$ domain size, $k =$ Q-width bound ± 1:

- $\tilde{O}(2^{\text{poly}(k)})$ time — compute decomposition
- $\tilde{O}(d^k)$ time — dynamic programming

QSAT in FPT time $\tilde{O}(2^{\text{poly}(k)}) + \tilde{O}(2^k)$, with param $k =$ Q-width

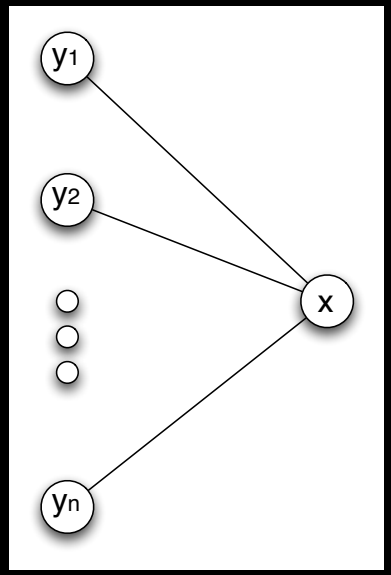
Parameter $=$ Q-width — is sufficient for QSAT to be FPT!

Width notion: example 1

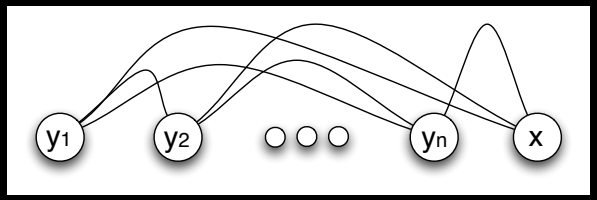
Decomposing the prefixed graph with prefix

$$\forall y_1 \ldots \forall y_n \exists x$$

and graph



Elimination ordering: width *n needed*

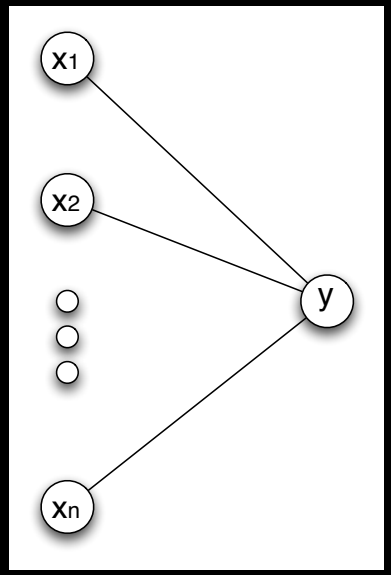


Width notion: example 2

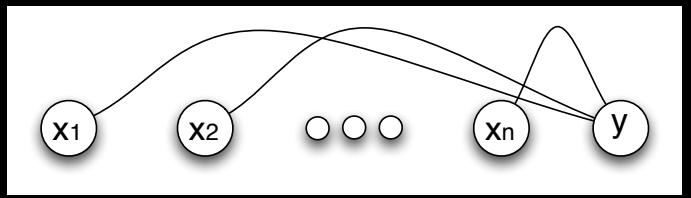
Decomposing the prefixed graph with prefix

$$\exists x_1 \ldots \exists x_n \forall y$$

and graph



Elimination ordering: width 0.



Bounded variable logic

In last example, formulas look something like

$$\exists x_1 \ldots \exists x_n \forall y (E_1(x_1, y) \wedge \cdots \wedge E_n(x_n, y))$$

Logically equivalent to:

$$\exists x_1 \ldots \exists x_n ((\forall y E_1(x_1, y)) \wedge \cdots \wedge (\forall y E_n(x_n, y)))$$

Logically equivalent to:

$$(\exists x_1 (\forall y E_1(x_1, y))) \wedge \cdots \wedge (\exists x_n (\forall y E_n(x_n, y)))$$

Renaming variables, obtain a 2-variable sentence!

$$(\exists x (\forall y E_1(x, y))) \wedge \cdots \wedge (\exists x (\forall y E_n(x, y)))$$

Thm: If ϕ has Q-width $\leqslant k$, symbols with arity $\leqslant k$, can (poly-time) syntactically transform into k-variable sentence

...which is very not prenex, in general!

Prop (Immerman '82, Vardi '95): for each $k \geqslant 1$, evaluating a k-variable sentence on a struct doable in poly-time $\tilde{O}(d^k)$

Proof system for non-prenex QCSP [Chen, LMCS '14]

Saw: Have low-width proofs for CSP instances w/ low treewidth
Question: Can we give low-width proofs for QCSP instances whose sentence has Q-width $\leqslant k$?

Prop: if qcsp-sentence ψ is k-variable & false on structure $\mathbf{B}$, then there exists a width k falsity proof

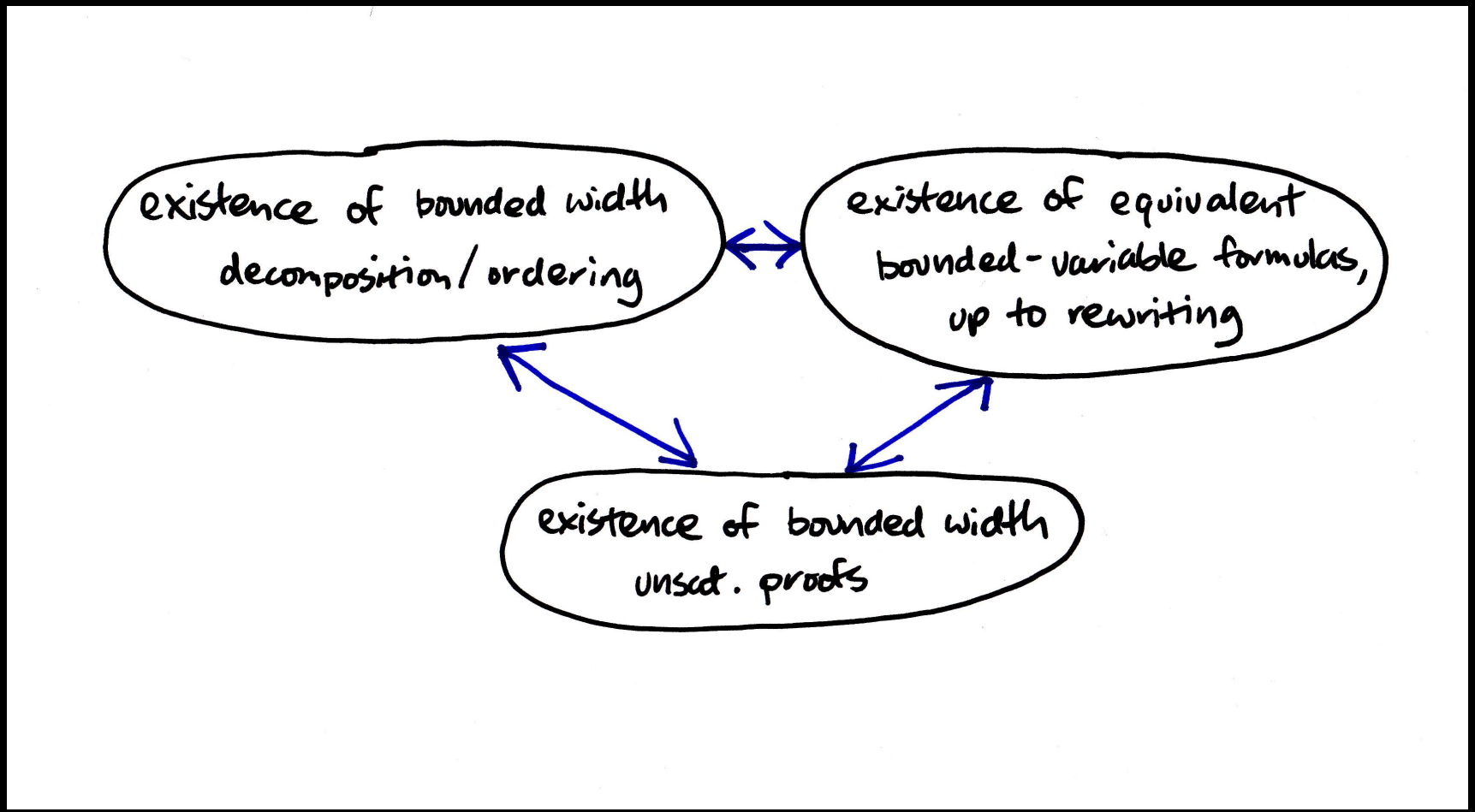
We said: if qcsp-sentence ϕ has Q-width $\leqslant k$,
can syntactically transform into k-variable sentence ψ

"Preservation" Theorem: existence of width k falsity proof
is preserved by the relevant syntactic transformations!

Consequence: if ϕ has Q-width $\leqslant k$ and is false on structure $\mathbf{B}$, then there exists a width k, poly-size falsity proof

Consequence of consequence:
if θ is a false QSAT instance with Q-width $\leqslant k$,
then it has a width k, poly-size Q-resolution proof

Summary


existence of bounded width decomposition / ordering
existence of equivalent bounded-variable formulas, up to rewriting
existence of bounded width unsat. proofs

Act: Proof complexity

QBF solving

Success in SAT solving (last $\approx$2 decades)
$\Rightarrow$ research on solving generalizations of SAT

Note: SAT treated as a black-box oracle by QBF solvers
(e.g. QBF solver *sKizzo* - Benedetti '05)

Rise in study of QBF lead to
algorithmic techniques and proof systems

Q-resolution: example proof system

Example formula: $\exists x_1 \exists x_2 \forall y_3 \exists x_4 (\ldots (x_2 \vee y_3 \vee x_4) \wedge (\overline{x_4} \vee \overline{x_1}) \ldots)$

Q-resolution has two derivation rules...

- Resolve on $\exists$-variable:
  $(\overline{x_1} \vee \overline{x_4}), (x_4 \vee x_2 \vee \overline{y_3}) \rightsquigarrow (\overline{x_1} \vee x_2 \vee \overline{y_3})$
- Eliminate a trailing $\forall$-literal:
  $(\overline{x_1} \vee x_2 \vee \overline{y_3}) \rightsquigarrow (\overline{x_1} \vee x_2)$

QBF proof complexity

QBF proof complexity – study lengths of proofs
in proof systems (for certifying QBF falsity)

Motivations:

- Certify a solver's *no* decision
- Solvers typically generate proofs...

  understanding proof length $\rightsquigarrow$ understanding running time
- Connection to separation of complexity classes

Dilemma

Basic, primary question, on *lower bounds*:

Take a usual QBF proof system, such as *Q-resolution*.

Can it be shown that (exponentially) long proofs are needed?

Answer: YES!

When restricted to SAT instances,
Q-resolution is identical to resolution.

So lower bounds on resolution apply to Q-resolution.

Reaction: This doesn't seem interesting.

We generalize resolution to Q-resolution to handle QBFs and quantifier alternation,
but this argument doesn't address this extra generality.

This also clashes with the QBF view of SAT as an oracle!

Escaping the dilemma

How can we prove lower bounds that are based on alternation?

We present a framework for doing this [Chen, ICALP '16]

- We define a proof system ensemble to be an infinite collection of proof systems,

  where in each, proof checking can be done in the PH (equivalently: with an oracle to bounded-alternation QSAT)

- An ensemble has polynomially bounded proofs if it *contains* a proof system where all false QBFs have polysize proofs

- Result: straightforward to define ensembles that have poly bd proofs on any set of QBFs with bounded alternation

  So, proof size lower bounds address the ability to handle alternation

Relaxing

Def (approximate): A relaxation of a QBF Ψ is
a QBF obtained from Ψ by shifting universal quantifiers left
and/or existential quantifiers right

Example: Consider a QBF $\exists x_1 \exists x_2 \forall y \forall y' \exists x_3 \psi$.
Example relaxations: $\forall y \forall y' \exists x_1 \exists x_2 \exists x_3 \psi$, $\exists x_1 \forall y' \exists x_2 \forall y \exists x_3 \psi$

Prop: If a relaxation of a QBF Ψ is false, then Ψ is false

That is, a relaxation is “more likely” to be true

Hard formulas

$$\Phi_n = \exists x_1 \forall y_1 \ldots \exists x_n \forall y_n \phi_n$$

$$\text{where} \quad \phi_n \text{ is true} \Leftrightarrow \sum_{i=1}^{n} (x_i + y_i) \not\equiv n \text{ (modulo 3)}$$

Φ_n is FALSE: universal can always set y_i to be $1 - x_i$,
so at the end of game, $\sum_{i=1}^{n} (x_i + y_i) = \sum_{i=1}^{n} 1 = n$

On the other hand, a simple swap of quantifiers can make a formula Φ_n TRUE: consider for example

$$\Phi'_3 = \forall y_1 \exists x_1 \exists x_2 \forall y_2 \exists x_3 \forall y_3 \phi_3$$

$\exists$ can play so that sum in the end $\not\equiv 0$:

- $\exists$ can ensure that x_1, x_2 are set so that $y_1 + x_1 + x_2 \equiv 0$,
- $\exists$ can then set $x_3 = 1 - y_2$ so that $y_1 + x_1 + x_2 + y_2 + x_3 \equiv 1$,
- $\Rightarrow$ sum in the end will be $\equiv 1$ or 2

An anecdote

I was talking with some QBF researchers.

Researchers: We need to select some benchmarks to do a comparison of QBF solvers. We want to make sure that there's a good balance of true/false instances.

Me: Why not just include, for every instance, its negation?

Researchers: Oh no, no, no! This would completely affect the instance structure, radically change solver behavior, blah blah...

The negation of a QSAT instance is a QSAT instance!
(or can be made to be, in poly-time)

Meta-research question: Are we even thinking about the right algorithms/proof systems for QSAT?

Should our algorithms'/proof systems' behavior be preserved under negation? ...and other natural transformations?

e.g.: Q-resolution defined on quantified CNFs, used to certify falsity

Act: Closing

Wrap-up

Interplay among treewidth & generalizations,
logic, proof systems

Theorems proved using FO logic have consequences for propositional logic

In terms of FO logic, focused on syntactically defined fragments $\{\exists, \wedge\}$ and $\{\forall, \exists, \wedge\}$

Many open problems, e.g., understand larger fragments of FO!

終

end [*fin*]