Acyclic Orders, Partition Schemes and CSPs

Unified Hardness Proofs and Improved Algorithms

Peter Jonsson Victor Lagerkvist George Osipov

Linköping University

IJCAI 2021

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

Short Summary

- Many computational problems can be realized as CSPs with infinite domains over partition schemes.
- Applications in AI include formalisms for qualitative spatial and temporal reasoning: *RCC-8, Allen's Interval Algebra, Rectangle Algebra, etc.*
- Many CSPs over partition schemes are computationally hard.
- We provide sufficient conditions for hardness explaining many results in the literature in a uniform way.
- We show that even restricted to degree-bounded cases, such CSPs are unlikely to admit subexponential time algorithms.
- In special cases (e.g., RCC-8) we show that CSPs over partition schemes admit improved algorithms.

CSP

Let $\mathcal{B}$ be a set of binary relations over a domain D.

CSP($\mathcal{B}$)

INSTANCE: A set of variables V and a set of constraints C of form $R(x, y)$, where $x, y \in V$ and $R \in \mathcal{B}$.
QUESTION: Is there an assignment $f \colon V \to D$ such that $(f(x), f(y)) \in R$ for all constraints $R(x, y)$ in C?

If $(f(x), f(y)) \in R$, we say that assignment f *satisfies* $R(x, y)$.

Partition Schemes

A *partition scheme* $\mathcal{B}$ is a set of binary relations over an infinite domain D such that:

- $\bigcup_{R \in \mathcal{B}} R = D^2$ (jointly exhaustive).
- If $R_1, R_2 \in \mathcal{B}$, then $R_1 \cap R_2 = \emptyset$ (pairwise disjoint).
- $\{(d, d) \mid d \in D\} \in \mathcal{B}$ (equality).
- If $R \in \mathcal{B}$, then $\{(b, a) \mid (a, b) \in R\} = R^{-1} \in \mathcal{B}$ (inverses).

Partition Schemes

A *partition scheme* $\mathcal{B}$ is a set of binary relations over an infinite domain D such that:

- $\bigcup_{R \in \mathcal{B}} R = D^2$ (jointly exhaustive).
- If $R_1, R_2 \in \mathcal{B}$, then $R_1 \cap R_2 = \emptyset$ (pairwise disjoint).
- $\{(d, d) \mid d \in D\} \in \mathcal{B}$ (equality).
- If $R \in \mathcal{B}$, then $\{(b, a) \mid (a, b) \in R\} = R^{-1} \in \mathcal{B}$ (inverses).

Partition Schemes

A *partition scheme* $\mathcal{B}$ is a set of binary relations over an infinite domain D such that:

- $\bigcup_{R \in \mathcal{B}} R = D^2$ (jointly exhaustive).
- If $R_1, R_2 \in \mathcal{B}$, then $R_1 \cap R_2 = \emptyset$ (pairwise disjoint).
- $\{(d, d) \mid d \in D\} \in \mathcal{B}$ (equality).
- If $R \in \mathcal{B}$, then $\{(b, a) \mid (a, b) \in R\} = R^{-1} \in \mathcal{B}$ (inverses).

Partition Schemes

A *partition scheme* $\mathcal{B}$ is a set of binary relations over an infinite domain D such that:

- $\bigcup_{R \in \mathcal{B}} R = D^2$ (jointly exhaustive).
- If $R_1, R_2 \in \mathcal{B}$, then $R_1 \cap R_2 = \emptyset$ (pairwise disjoint).
- $\{(d, d) \mid d \in D\} \in \mathcal{B}$ (equality).
- If $R \in \mathcal{B}$, then $\{(b, a) \mid (a, b) \in R\} = R^{-1} \in \mathcal{B}$ (inverses).

Example 1

$\langle \mathbb{Q}; =, \neq \rangle$.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 1

$\langle \mathbb{Q}; =, \neq \rangle$.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 1

$\langle \mathbb{Q}; =, \neq \rangle$.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 1

$\langle \mathbb{Q}; =, \neq \rangle$.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 2: Point Algebra

$\langle \mathbb{Q}; <, =, > \rangle$ aka Point Algebra.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 2: Point Algebra

$\langle \mathbb{Q}; <, =, > \rangle$ aka POINT ALGEBRA.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 2: Point Algebra

$\langle \mathbb{Q}; <, =, > \rangle$ aka Point Algebra.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

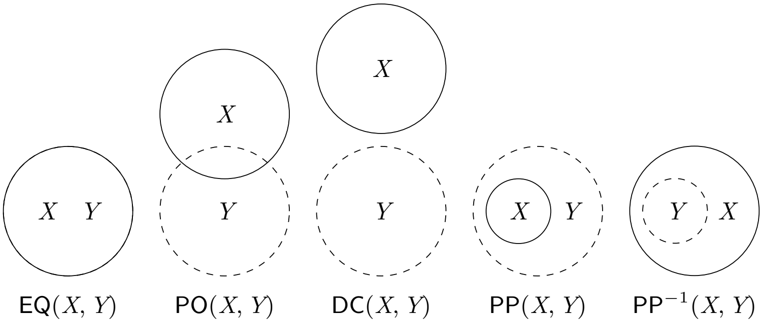
Example 2: Point Algebra

$\langle \mathbb{Q}; <, =, > \rangle$ aka POINT ALGEBRA.

- ✓ jointly exhaustive
- ✓ pairwise disjoint
- ✓ contains equality
- ✓ closed under taking converses

Example 3: RCC-5

Objects of RCC-5 are (not necessarily connected) space regions.



More Expressive Power

Let $\mathcal{B}^{\vee=}$ contain unions of all relations in $\mathcal{B}$.

Example

The relation $\mathsf{DR} \cup \mathsf{PO}$ denoted by $(\mathsf{DR}, \mathsf{PO})$ contains pairs of regions that are either disjoint or overlap. Constraint $(\mathsf{DR}, \mathsf{PO})(X, Y)$ is logically equivalent to $\mathsf{DR}(X, Y) \vee \mathsf{PO}(X, Y)$.

$\mathcal{B}^{\vee=}$ has more expressive power than $\mathcal{B}$. However, CSP($\mathcal{B}^{\vee=}$) is usually computationally hard.

Complexity Questions

1. For which partition schemes $\mathcal{B}$ is $\mathrm{CSP}(\mathcal{B}^{\vee=})$ NP-hard?
2. How much time is required to solve $\mathrm{CSP}(\mathcal{B}^{\vee=})$?
3. Are there better algorithms for *some* partition schemes $\mathcal{B}$?

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \land d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \lor d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \wedge d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \vee d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \wedge d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \vee d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \wedge d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \vee d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \wedge d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \vee d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Orders

An order $\prec \subseteq D^2$ is a binary relation.

- Irreflexive: $\nexists d \in D : d \prec d$.
- Transitive: $\forall d_1, d_2, d_3 : d_1 \prec d_2 \wedge d_2 \prec d_3 \implies d_1 \prec d_3$.
- Acyclic: $\nexists d_1, \ldots, d_k : d_1 \prec d_2 \prec \cdots \prec d_{k-1} \prec d_k \prec d_1$.
- Total: $\forall d_1, d_2 : d_1 \prec d_2 \vee d_2 \prec d_1$.
- Irreflexive + transitive = strict partial $\implies$ acyclic.
- Irreflexive + transitive + total = strict total.

Hardness Conditions

Let $\prec \subseteq D^2$ be an acyclic order and $\sqcap \subseteq D^2$ be a relation.

(C1) (unbounded total orders)
$\forall k \in \mathbb{N}\ \exists L \subset D :\ |L| \geq k$ and $\prec$ is strict total on L.

(C2) (in-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_1 :\ d_1 \sqcap a, d_1 \sqcap b, d_1(\prec, \succ)c.$

(C3) (out-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_2 :\ d_2(\prec, \succ)a, d_2 \sqcap b, d_2 \sqcap c.$

(C4) (no-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \nexists d_3 :\ d_3 \sqcap a, d_3(\prec, \succ)b, d_3 \prec c.$

Hardness Conditions

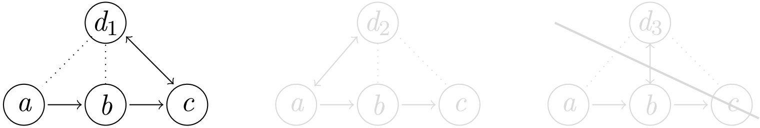
Let $\prec \subseteq D^2$ be an acyclic order and $\sqcap \subseteq D^2$ be a relation.

(C1) (unbounded total orders)
$\forall k \in \mathbb{N}\ \exists L \subset D :\ |L| \geq k$ and $\prec$ is strict total on L.

(C2) (in-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_1 :\ d_1 \sqcap a, d_1 \sqcap b, d_1(\prec, \succ)c.$

(C3) (out-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_2 :\ d_2(\prec, \succ)a, d_2 \sqcap b, d_2 \sqcap c.$

(C4) (no-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \nexists d_3 :\ d_3 \sqcap a, d_3(\prec, \succ)b, d_3 \prec c.$



Hardness Conditions

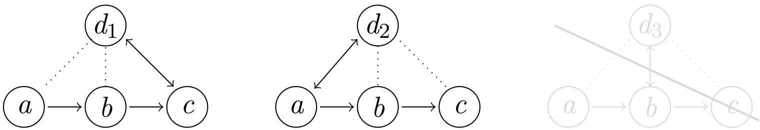
Let $\prec \subseteq D^2$ be an acyclic order and $\sqcap \subseteq D^2$ be a relation.

(C1) (unbounded total orders)
$\forall k \in \mathbb{N}\ \exists L \subset D: |L| \geq k$ and $\prec$ is strict total on L.

(C2) (in-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_1 : d_1 \sqcap a, d_1 \sqcap b, d_1(\prec, \succ)c.$

(C3) (out-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_2 : d_2(\prec, \succ)a, d_2 \sqcap b, d_2 \sqcap c.$

(C4) (no-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \nexists d_3 : d_3 \sqcap a, d_3(\prec, \succ)b, d_3 \prec c.$



Hardness Conditions

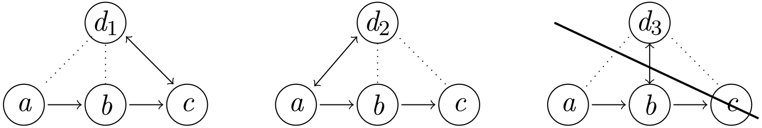
Let $\prec \subseteq D^2$ be an acyclic order and $\sqcap \subseteq D^2$ be a relation.

(C1) (unbounded total orders)
$\forall k \in \mathbb{N}\ \exists L \subset D:\ |L| \geq k$ and $\prec$ is strict total on L.

(C2) (in-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_1:\ d_1 \sqcap a, d_1 \sqcap b, d_1 (\prec, \succ) c.$

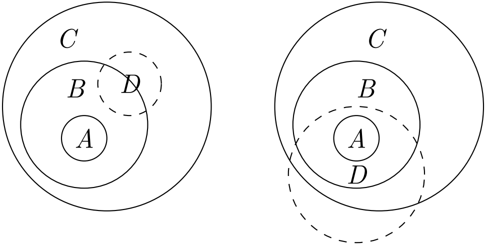
(C3) (out-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \exists d_2:\ d_2 (\prec, \succ) a, d_2 \sqcap b, d_2 \sqcap c.$

(C4) (no-forks)
$\forall a, b, c,\ a \prec b \prec c, a \prec c,\ \nexists d_3:\ d_3 \sqcap a, d_3 (\prec, \succ) b, d_3 \prec c.$



RCC-5

PP is an acyclic order ($\prec$). (DR,PO) is incomparability relation ($\sqcap$).



$A \prec B$, $B \prec C$, $A \prec C$.
In-fork: $D \sqcap A$, $D \sqcap B$, $D \prec C$.
Out-fork: $A \prec D$, $B \sqcap D$, $C \prec D$.

Main Theorem

Definition

Let $\mathcal{H}$ be the set of partition schemes $\mathcal{B}$ such that
(1) CSP($\mathcal{B}$) is in P, and
(2) $\mathcal{B}$ contains acyclic order $\prec$ and relation $\sqcap$ satisfying C1-C4.

Theorem

If $\mathcal{B} \in \mathcal{H}$, then $CSP(\mathcal{B}^{\vee=})$ is NP-complete.

Main Theorem

In CSP($\mathcal{B}^{\vee=}$)-B each variable occurs in at most B constraints.

Theorem

If $\mathcal{B} \in \mathcal{H}$, then CSP($\mathcal{B}^{\vee=}$)-3 is NP-complete.

CSP($\mathcal{B}^{\vee=}$)-2 is in P.

Main Theorem

In CSP($\mathcal{B}^{\vee=}$)-B each variable occurs in at most B constraints.

Theorem

If $\mathcal{B} \in \mathcal{H}$, then CSP($\mathcal{B}^{\vee=}$)-3 is NP-complete.

CSP($\mathcal{B}^{\vee=}$)-2 is in P.

Main Theorem

3-SAT asks whether a Boolean formula in 3-CNF is satisfiable:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4).$$

Exponential Time Hypothesis (ETH)

3-SAT is not solvable in subexponential time.

Theorem

If $\mathcal{B} \in \mathcal{H}$, then CSP($\mathcal{B}^{\vee=}$)-3 is not solvable in subexponential time, unless the ETH fails.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee=})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
   - a. add five variables z, x_1, x_2, x_3, x_4 to I;
   - b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
   - c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee =})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
   a. add five variables z, x_1, x_2, x_3, x_4 to I;
   b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
   c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee=})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
   a. add five variables z, x_1, x_2, x_3, x_4 to I;
   b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
   c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee=})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
   a. add five variables z, x_1, x_2, x_3, x_4 to I;
   b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
   c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee=})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
    a. add five variables z, x_1, x_2, x_3, x_4 to I;
    b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
    c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Proof Idea

3-NAE-SAT asks whether variables can be assigned values $0, 1$ so that no clause contains all equal values.

$$(a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee c \vee d).$$

If $\mathcal{B} \in \mathcal{H}$, we can define a *gadget* $G(a, b, c, z_1, z_2)$ which ensures that $(a \prec b \prec c) \vee (a \succ b \succ c)$. Define instance I of $\mathrm{CSP}(\mathcal{B}^{\vee=})$:

1. Add the variable M to I.
2. For each variable a, add constraints $a(\prec, \succ)M$ to I.
3. For each clause $(a \vee b \vee c)$:
   a. add five variables z, x_1, x_2, x_3, x_4 to I;
   b. add constraints $a(\prec, \succ)b$, $b(\prec, \succ)c$, $a(\prec, \succ)c$;
   c. add constraints $G(a, M, z, x_1, x_2)$ and $G(b, z, c, x_3, x_4)$.

Application: Allen's Interval Algebra

Basic relation		Example	Endpoints
x precedes y	p	xxx	$I^+ < J^-$
y preceded by x	p^{-1}	yyy	
x meets y	m	xxxx	$I^+ = J^-$
y met-by x	m^{-1}	yyyy	
x overlaps y	o	xxxx	$I^- < J^- < I^+$,
y overl.-by x	o^{-1}	yyyy	$I^+ < J^+$
x during y	d	xxx	$I^- > J^-$,
y includes x	d^{-1}	yyyyyyy	$I^+ < J^+$
x starts y	s	xxx	$I^- = J^-$,
y started by x	s^{-1}	yyyyyyy	$I^+ < J^+$
x finishes y	f	xxx	$I^+ = J^+$,
y finished by x	$\mathsf{f}^{-}1$	yyyyyyy	$I^- > J^-$
x equals y	$\equiv$	xxxx yyyy	$I^- = J^-$, $I^+ = J^+$

Let $\prec$ be p and $\sqcap$ be $\mathcal{A} \setminus \{\mathsf{p}, \mathsf{p}^{-1}\}$.

(C1) xxx yyy zzz ...

(C2) uuuuuu

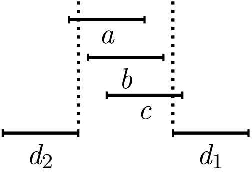
(C3) vvvvvv

(C4) ~~www~~ ~~www~~

Application: Unit Interval Algebra

Basic relation		Example	Endpoints
x precedes y	p	xxx	$I^+ < J^-$
y preceded by x	p^{-1}	yyy	
x meets y	m	xxxx	$I^+ = J^-$
y met-by x	m^{-1}	yyyy	
x overlaps y	o	xxxx	$I^- < J^- < I^+$,
y overl.-by x	o^{-1}	yyyy	$I^+ < J^+$
x equals y	$\equiv$	xxxx	$I^- = J^-$,
		yyyy	$I^+ = J^+$

- Choosing $\prec$ to be p does not work here.
- Instead, let $\prec$ be o and $\sqcap$ be $(\mathsf{p}, \mathsf{p}^{-1})$.
- Conditions C1-C4 hold.



Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, $\text{CSP}(\mathcal{B}^{\vee=})$ is not in $2^{o(n)}$ time.
- Branching solves $\text{CSP}(\mathcal{B}^{\vee=})$-$B$ in $2^{O(n)}$ time for fixed B.
- Branching solves $\text{CSP}(\mathcal{B}^{\vee=})$ in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for $\text{CSP}(\mathcal{B}^{\vee=})$?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?

Upper Bounds

- Under ETH, if $\mathcal{B} \in \mathcal{H}$, CSP($\mathcal{B}^{\vee=}$) is not in $2^{o(n)}$ time.
- Branching solves CSP($\mathcal{B}^{\vee=}$)-B in $2^{O(n)}$ time for fixed B.
- Branching solves CSP($\mathcal{B}^{\vee=}$) in $2^{O(n^2)}$ time.
- For AIA, UIA, RCC-8, RA, there is a $2^{O(n \log n)}$ algorithm.
- Open Question: Is there a tighter lower bound or an improved algorithm for CSP($\mathcal{B}^{\vee=}$)?