Faster Existential FO Model Checking on Posets

Petr Hliněný

Faculty of Informatics
Masaryk University, Brno, CZ

Jakub Gajarský,
Jan Obdržálek,
Sebastian Ordyniak

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

* Logic on Graphs

- Propositional logic ($\wedge \vee \rightarrow$), graph vertices/edges ($x, y, z, .., e, \dots$);
  - e.g., $\forall x, y \big(\mathit{edge}(x, y) \rightarrow x \in C \vee y \in C \big)$,

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

* Logic on Graphs

- Propositional logic ($\wedge \vee \rightarrow$), graph vertices/edges ($x, y, z, .., e, \dots$);
  - e.g., $\forall x, y \big(\mathit{edge}(x, y) \rightarrow x \in C \vee y \in C\big)$, "$C$ is vertex cover"
  - **FO logic** (first-order): just this $\uparrow$

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

* Logic on Graphs

- Propositional logic ($\wedge \vee \rightarrow$), graph vertices/edges ($x, y, z, .., e, \dots$);
  - e.g., $\forall x, y \big(\mathit{edge}(x, y) \rightarrow x \in C \vee y \in C \big)$, "$C$ is vertex cover"
  - **FO logic** (first-order): just this $\uparrow$
  - **MSO logic** (monadic second-o.): quantifies vertex sets $\exists X, Y$

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

* Logic on Graphs

- Propositional logic ($\wedge \vee \rightarrow$), graph vertices/edges ($x, y, z, .., e, \dots$);
  - e.g., $\forall x, y \big(\textit{edge}(x, y) \rightarrow x \in C \vee y \in C \big)$, "$C$ is vertex cover"
  - **FO logic** (first-order): just this $\uparrow$
  - **MSO logic** (monadic second-o.): quantifies vertex sets $\exists X, Y$

$\uparrow$ MSO_1 vs. MSO_2 $\downarrow$

0 The Aim; Algorithmic Metatheorems

– theor. tools claiming efficient solvability of large classes of problems at once.

* Courcelle's Theorem

- All MSO_2-definable prop. in linear-time FPT for bounded tree-width.
  - perhaps the best known algo. metatheorem on graphs (1988)
  - clique-width + MSO_1 version by [Courcelle–Makowsky–Rotics]

* Logic on Graphs

- Propositional logic ($\wedge \vee \rightarrow$), graph vertices/edges ($x, y, z, .., e, \dots$);
  - e.g., $\forall x, y \big(\mathit{edge}(x, y) \rightarrow x \in C \vee y \in C \big)$, "$C$ is vertex cover"
  - **FO logic** (first-order): just this ↑
  - **MSO logic** (monadic second-o.): quantifies vertex sets $\exists X, Y$

    ↑ MSO_1 vs. MSO_2 ↓
  - or, quantifies vertex and edge sets together $\exists X, Y, E, F$.

How far can one get?

* Not far with MSO

- On monotone graph classes:

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

- FO is always in XP, but we aim for FPT (fixed exponent poly.):

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

- FO is always in XP, but we aim for FPT (fixed exponent poly.):
  - [Seese] on bounded degree graphs (1996)

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

- FO is always in XP, but we aim for FPT (fixed exponent poly.):
  - [Seese] on bounded degree graphs (1996)
  - [Frick–Grohe] locally bounded tree-width

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

- FO is always in XP, but we aim for FPT (fixed exponent poly.):
  - [Seese] on bounded degree graphs (1996)
  - [Frick–Grohe] locally bounded tree-width
  - [Dawar–Grohe–Kreutzer] locally excluding a minor

How far can one get?

* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

- FO is always in XP, but we aim for FPT (fixed exponent poly.):
  - [Seese] on bounded degree graphs (1996)
  - [Frick–Grohe] locally bounded tree-width
  - [Dawar–Grohe–Kreutzer] locally excluding a minor
  - [D.-K. / Dvořák–Král'–Thomas] locally bounded expansion

How far can one get?

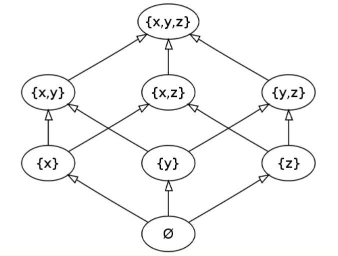
* Not far with MSO

- On monotone graph classes:
  - [Kreutzer–Tazari] not above polylog. tree-width with MSO_2
  - [GHLORS] not above polylog. tree-width with coloured MSO_1
- On hereditary classes, perhaps analogously with clique-width. . . ?

* Better with FO

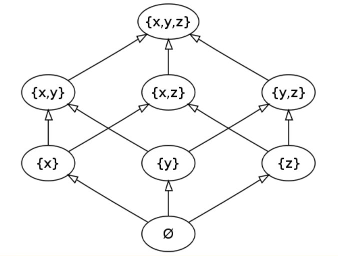
- FO is always in XP, but we aim for FPT (fixed exponent poly.):
  - [Seese] on bounded degree graphs (1996)
  - [Frick–Grohe] locally bounded tree-width
  - [Dawar–Grohe–Kreutzer] locally excluding a minor
  - [D.-K. / Dvořák–Král'–Thomas] locally bounded expansion
  - [Grohe–Kreutzer–Siebertz] **nowhere dense graphs !** (2013)

1 FO Model Checking on Posets



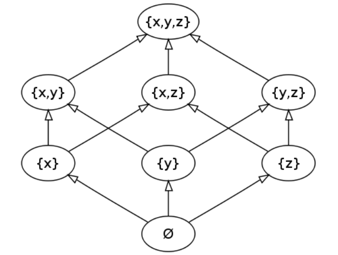
- A poset $\mathcal{P}$ (partially ordered set)

1 FO Model Checking on Posets



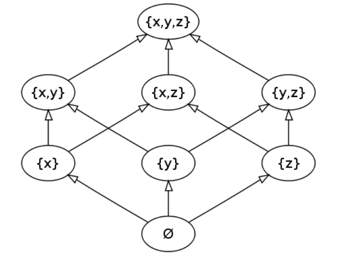
- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.

1 FO Model Checking on Posets



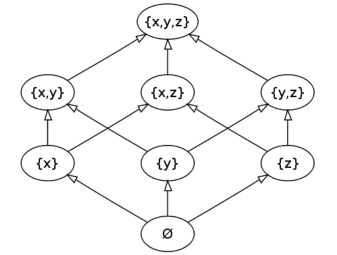
- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.
- FO logic on a poset; e.g.
  - $\forall y\, (y \leq x \rightarrow x \leq y)$,

1 FO Model Checking on Posets



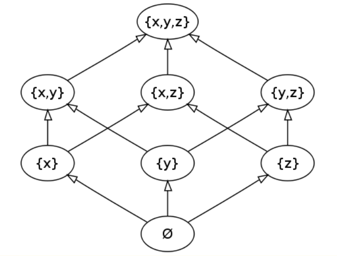
- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.
- FO logic on a poset; e.g.
  - $\forall y \, (y \leq x \rightarrow x \leq y)$, "$x$ is a minimal element",

1 FO Model Checking on Posets



- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.
- FO logic on a poset; e.g.
  - $\forall y\,(y \leq x \to x \leq y)$, "$x$ is a minimal element",
  - $z \leq x \wedge z \leq y \wedge \forall t\left[(t \leq x \wedge t \leq y) \to t \leq z\right]$, "infimum",

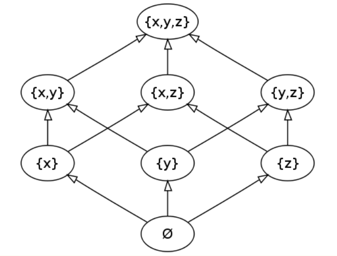
1 FO Model Checking on Posets



- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.
- FO logic on a poset; e.g.
  - $\forall y\,(y \leq x \rightarrow x \leq y)$, "$x$ is a minimal element",
  - $z \leq x \wedge z \leq y \wedge \forall t\left[(t \leq x \wedge t \leq y) \rightarrow t \leq z\right]$, "infimum",
- Model checking (a parameterized formulation)

  Input: A poset $\mathcal{P}$, and an (FO) sentence ϕ.

1 FO Model Checking on Posets



- A poset $\mathcal{P}$ (partially ordered set)
  - a ground set P, and
  - a reflexive, symmetric, transitive bin. relation $\leq$ on P.
- FO logic on a poset; e.g.
  - $\forall y \, (y \leq x \rightarrow x \leq y)$, "$x$ is a minimal element",
  - $z \leq x \wedge z \leq y \wedge \forall t \, \big[(t \leq x \wedge t \leq y) \rightarrow t \leq z\big]$, "infimum",
- Model checking (a parameterized formulation)

  Input: A poset $\mathcal{P}$, and an (FO) sentence ϕ. Param.: $|\phi|$.
  Question: Is $\mathcal{P} \models \phi$?

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;
  - except [Ganian, PH, Král', Obdržálek, Schwartz, Teska] FO model checking on L-interval graphs (2013).

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;
  - except [Ganian, PH, Král', Obdržálek, Schwartz, Teska] FO model checking on L-interval graphs (2013).
- Posets present a very natural example of dense relational structures.

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;
  - except [Ganian, PH, Král', Obdržálek, Schwartz, Teska] FO model checking on L-interval graphs (2013).
- Posets present a very natural example of dense relational structures.
- The research initiated by

[Bova, Ganian, Szeider, LICS 2014]:

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;
  - except [Ganian, PH, Král', Obdržálek, Schwartz, Teska] FO model checking on L-interval graphs (2013).
- Posets present a very natural example of dense relational structures.
- The research initiated by

  [Bova, Ganian, Szeider, LICS 2014]:

  FO model checking on posets is hard (W[1]-hard par. $|\phi|$) for, e.g.
  - bounded depth (the maximum size of a chain),
  - bounded cover-degree (the max. deg. in the Hasse diagram),

Why FO on Posets?

- So far, most nontrivial results obtained on sparse classes of graphs;
  - except [Ganian, PH, Král', Obdržálek, Schwartz, Teska] FO model checking on L-interval graphs (2013).
- Posets present a very natural example of dense relational structures.
- The research initiated by

$$\boxed{\text{[Bova, Ganian, Szeider, LICS 2014]:}}$$

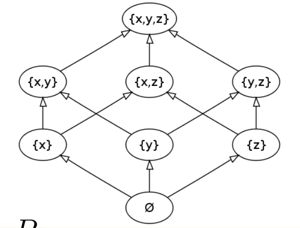
FO model checking on posets is hard (W[1]-hard par. $|\phi|$) for, e.g.

- bounded depth (the maximum size of a chain),
- bounded cover-degree (the max. deg. in the Hasse diagram),

while the problem **becomes FPT** for

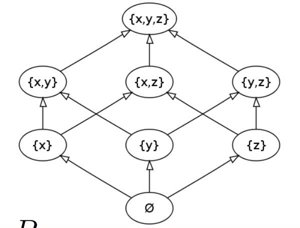
- posets of bounded width (the maximum size of an antichain).

A short detour; Hasse Diagrams



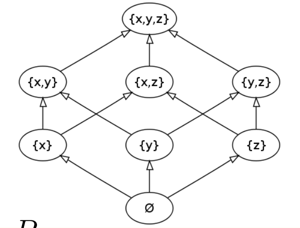
- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.

A short detour; Hasse Diagrams



- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.
- H determines $\mathcal{P}$, and so
  - how FO m.c. can be hard when $\Delta(H)$ (cov. deg.) is bounded?

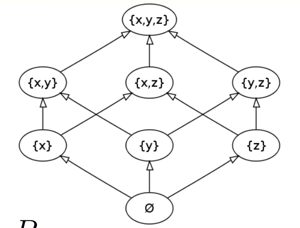
A short detour; Hasse Diagrams



- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.
- H determines $\mathcal{P}$, and so
  - how FO m.c. can be hard when $\Delta(H)$ (cov. deg.) is bounded?

Yes... but $\mathcal{P}$ is the transitive closure of H – not FO-definable!

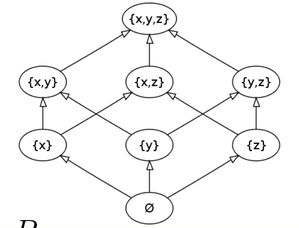
A short detour; Hasse Diagrams



- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.
- H determines $\mathcal{P}$, and so
  - how FO m.c. can be hard when $\Delta(H)$ (cov. deg.) is bounded?

  Yes... but $\mathcal{P}$ is the transitive closure of H – not FO-definable!
- Still, one more try:
  - MSO logic defines the transitive closure, and

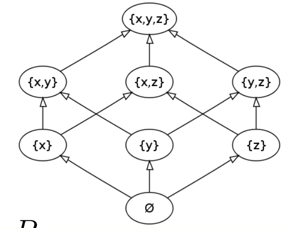
A short detour; Hasse Diagrams



- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.
- H determines $\mathcal{P}$, and so
  - how FO m.c. can be hard when $\Delta(H)$ (cov. deg.) is bounded?

  Yes... but $\mathcal{P}$ is the transitive closure of H – not FO-definable!
- Still, one more try:
  - MSO logic defines the transitive closure, and
  - Hasse d. of bounded width seem to have small tree-width.

A short detour; Hasse Diagrams



- Having a poset $\mathcal{P}(P, \leq)$, the Hasse diagram is the digraph H on $V(H) = P$
  - such that $(u, v) \in E(H)$ iff $u < v$, and no $x \in P$, $u < x < v$.
- H determines $\mathcal{P}$, and so
  - how FO m.c. can be hard when $\Delta(H)$ (cov. deg.) is bounded?

  Yes. . . but $\mathcal{P}$ is the transitive closure of H – not FO-definable!
- Still, one more try:
  - MSO logic defines the transitive closure, and
  - Hasse d. of bounded width seem to have small tree-width.

  NO, [BGS] arbitrarily large grids even for poset width 2!

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \dots$).

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \dots$).
- The main result of [BGS, LICS 2014]:

  POSET ∃-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{f(|\phi|) \cdot |P|^{g(width(\mathcal{P}))}.}$$

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \dots$).
- The main result of [BGS, LICS 2014]:

  Poset ∃-FO model checking on $\mathcal{P} = (P, \leq)$ solvable in time

  $$\boxed{f(|\phi|) \cdot |P|^{g(width(\mathcal{P}))}.}$$

  Note, only an XP algorithm with respect to $width(\mathcal{P})$.

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \ldots$).
- The main result of [BGS, LICS 2014]:

  POSET ∃-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

  $$\boxed{f(|\phi|) \cdot |P|^{g(width(\mathcal{P}))}.}$$

  Note, only an XP algorithm with respect to $width(\mathcal{P})$.
- Step 1. An FPT reduction to many instances of the embedding problem ("induced subposet").

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \dots$).
- The main result of [BGS, LICS 2014]:

  Poset ∃-FO model checking on $\mathcal{P} = (P, \leq)$ solvable in time

  $$\boxed{f(|\phi|) \cdot |P|^{g(width(\mathcal{P}))}.}$$

  Note, only an XP algorithm with respect to $width(\mathcal{P})$.

- Step 1. An FPT reduction to many instances of the embedding problem ("induced subposet").
- Step 2. A further reduction to a family of instances of the homomorphism problem on certain lattice structures.

2 ∃-FO on Posets

*** [Bova, Ganian, Szeider]**

- ∃-FO – the existential fragment of FO (no $\forall x \ldots$).
- The main result of [BGS, LICS 2014]:

  Poset ∃-FO model checking on $\mathcal{P} = (P, \leq)$ solvable in time

  $$\boxed{f(|\phi|) \cdot |P|^{g(width(\mathcal{P}))}.}$$

  Note, only an XP algorithm with respect to $width(\mathcal{P})$.

- Step 1. An FPT reduction to many instances of the embedding problem ("induced subposet").
- Step 2. A further reduction to a family of instances of the homomorphism problem on certain lattice structures.
- Step 3. Solving the homomorphism problem in polytime (using a highly non-trivial theorem).

The Embedding Problem

*** [BGS]**

- EMBEDDING (for posets)

  INPUT: Two poset $\mathcal{Q} = (Q, \leq_Q)$ and $\mathcal{P} = (P, \leq_P)$.

  PARAMETERS: $|Q|$ and $width(\mathcal{P})$.

  QUESTION: Is there an embedding from $\mathcal{Q}$ into $\mathcal{P}$?

The Embedding Problem

*** [BGS]**

- EMBEDDING (for posets)

  INPUT: Two poset $\mathcal{Q} = (Q, \leq_Q)$ and $\mathcal{P} = (P, \leq_P)$.
  PARAMETERS: $|Q|$ and $width(\mathcal{P})$.

  QUESTION: Is there an embedding from $\mathcal{Q}$ into $\mathcal{P}$?

- An embedding from $\mathcal{Q}$ into $\mathcal{P}$
  - an injective mapping $e : Q \to P$ such that $q \leq_Q q'$ iff $e(q) \leq_P e(q')$, for every $q, q' \in Q$.

The Embedding Problem

*** [BGS]**

- EMBEDDING (for posets)

  INPUT: Two poset $\mathcal{Q} = (Q, \leq_Q)$ and $\mathcal{P} = (P, \leq_P)$.
  PARAMETERS: $|Q|$ and $width(\mathcal{P})$.

  QUESTION: Is there an embedding from $\mathcal{Q}$ into $\mathcal{P}$?

- An embedding from $\mathcal{Q}$ into $\mathcal{P}$
  - an injective mapping $e : Q \to P$ such that
    $q \leq_Q q'$ iff $e(q) \leq_P e(q')$, for every $q, q' \in Q$.

- Technically easier (with $width(\mathcal{P})^{|Q|}$ blow-up) to consider:

  COMPATIBLE EMBEDDING

The Embedding Problem

*** [BGS]**

- Embedding (for posets)

  Input: Two poset $\mathcal{Q} = (Q, \leq_Q)$ and $\mathcal{P} = (P, \leq_P)$.
  Parameters: $|Q|$ and $width(\mathcal{P})$.

  Question: Is there an embedding from $\mathcal{Q}$ into $\mathcal{P}$?

- An embedding from $\mathcal{Q}$ into $\mathcal{P}$
  - an injective mapping $e : Q \to P$ such that
    $q \leq_Q q'$ iff $e(q) \leq_P e(q')$, for every $q, q' \in Q$.

- Technically easier (with $width(\mathcal{P})^{|Q|}$ blow-up) to consider:

  Compatible Embedding

  ... $\mathcal{P}$ with a chain partition $(C_1, \ldots, C_w)$, $w = width(\mathcal{P})$,
  and $\mathcal{Q}$ with a mapping $f : Q \to \{1, \ldots, w\}$.

The Embedding Problem

* [BGS]

- EMBEDDING (for posets)

  INPUT: Two poset $\mathcal{Q} = (Q, \leq_Q)$ and $\mathcal{P} = (P, \leq_P)$.

  PARAMETERS: $|Q|$ and $width(\mathcal{P})$.

  QUESTION: Is there an embedding from $\mathcal{Q}$ into $\mathcal{P}$?

- An embedding from $\mathcal{Q}$ into $\mathcal{P}$
  - an injective mapping $e : Q \to P$ such that
    $q \leq_Q q'$ iff $e(q) \leq_P e(q')$, for every $q, q' \in Q$.

- Technically easier (with $width(\mathcal{P})^{|Q|}$ blow-up) to consider:

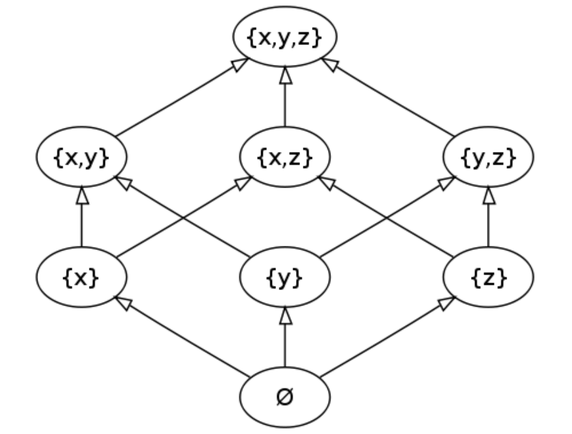
  COMPATIBLE EMBEDDING

  ... $\mathcal{P}$ with a chain partition $(C_1, \ldots, C_w)$, $w = width(\mathcal{P})$,
  and $\mathcal{Q}$ with a mapping $f : Q \to \{1, \ldots, w\}$.

  QUESTION: Is there an embedding $e : Q \to P$ such that
  $e(q) \in C_{f(q)}$, for every $q \in Q$?

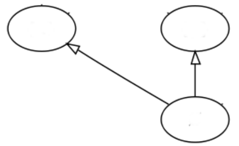
The Embedding Problem; examples

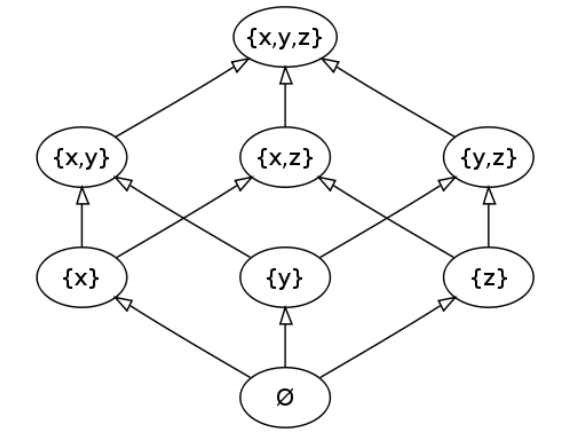
Does it embed?



The Embedding Problem; examples

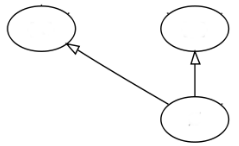
Does it embed?



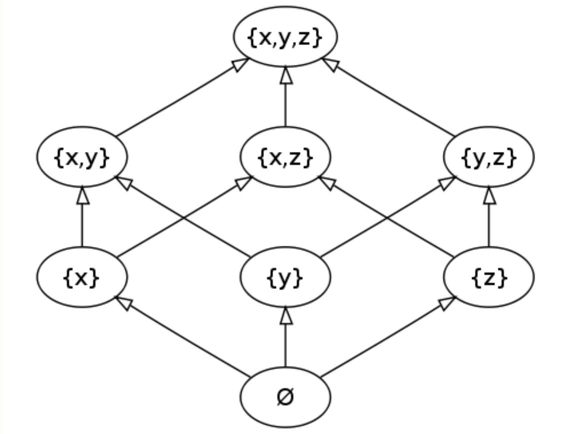


The Embedding Problem; examples

Does it embed?

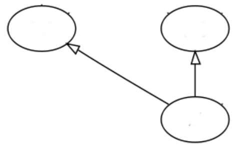


YES

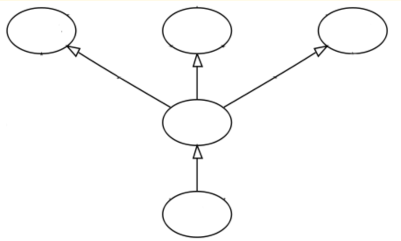


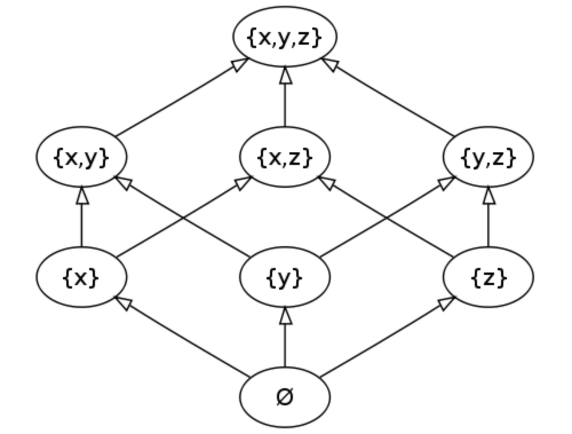
The Embedding Problem; examples

Does it embed?



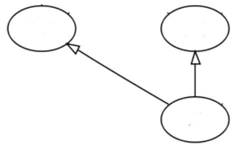
YES



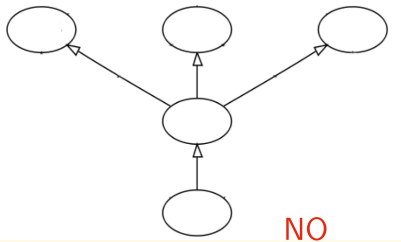


The Embedding Problem; examples

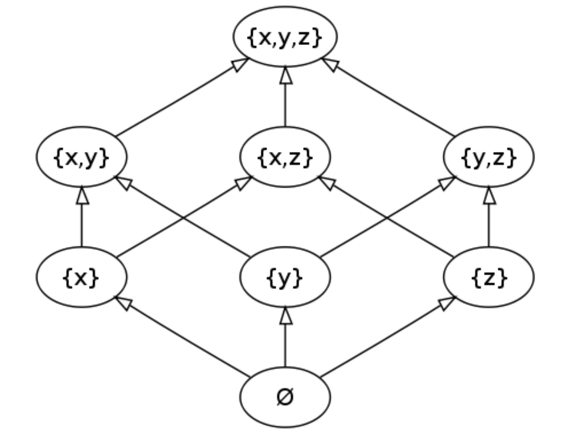
Does it embed?



YES



NO



3 New Result

POSET $\exists$-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

3 New Result

POSET $\exists$-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

Note, this is now an FPT algorithm with respect to both $|\phi|$, $width(\mathcal{P})$.

3 New Result

POSET $\exists$-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

Note, this is now an FPT algorithm with respect to both $|\phi|$, $width(\mathcal{P})$.

- Step 1. The same FPT reduction (as [BGS]) to many instances of COMPATIBLE EMBEDDING.

3 New Result

Poset $\exists$-FO model checking on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

Note, this is now an FPT algorithm with respect to both $|\phi|$, $width(\mathcal{P})$.

- Step 1. The same FPT reduction (as [BGS]) to many instances of Compatible Embedding.
- Step 2. (solution 1) Solving each Comp. Embedding inst. as a CSP instance closed under a min-polymorhpism (poly but slow).

3 New Result

POSET $\exists$-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

Note, this is now an FPT algorithm with respect to both $|\phi|$, $width(\mathcal{P})$.

- Step 1. The same FPT reduction (as [BGS]) to many instances of COMPATIBLE EMBEDDING.
- Step 2. (solution 1) Solving each COMP. EMBEDDING inst. as a CSP instance closed under a min-polymorhpism (poly but slow).

  (solution 2) Further one-to-one reduction to a certain variant of well known MULTICOLOURED CLIQUE.

3 New Result

POSET $\exists$-FO MODEL CHECKING on $\mathcal{P} = (P, \leq)$ solvable in time

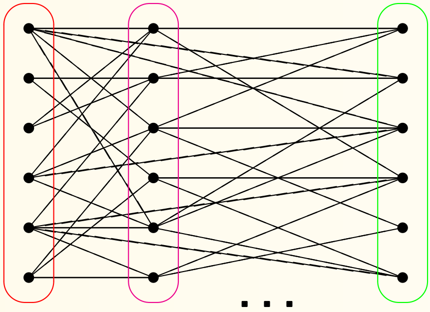
$$\boxed{h(|\phi|, width(\mathcal{P})) \cdot |P|^2.}$$

Note, this is now an FPT algorithm with respect to both $|\phi|$, $width(\mathcal{P})$.

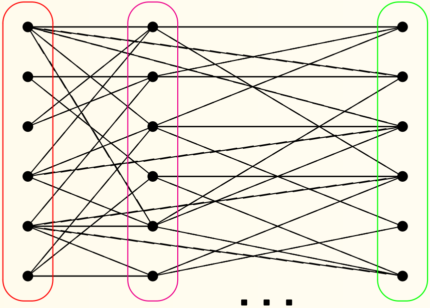
- Step 1. The same FPT reduction (as [BGS]) to many instances of COMPATIBLE EMBEDDING.
- Step 2. (solution 1) Solving each COMP. EMBEDDING inst. as a CSP instance closed under a min-polymorhpism (poly but slow).

  (solution 2) Further one-to-one reduction to a certain variant of well known MULTICOLOURED CLIQUE.
- Step 3. Solving the (interval-monotone) variant of MULTICOLOURED CLIQUE in polynomial time – overall quadratic in $|P|$.

Multicoloured Clique



Multicoloured Clique



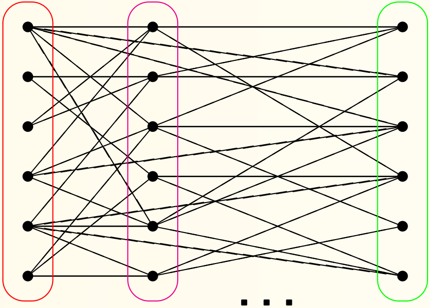
- Multicoloured Clique

  Input: A graph G with a proper k-colouring.

  Parameter: k.

  Question: Is there a clique (complete subgr.) of size k in G?

Multicoloured Clique



- MULTICOLOURED CLIQUE

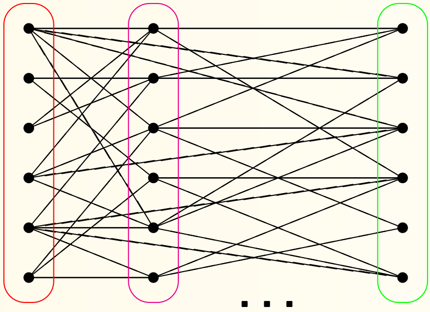
  INPUT: A graph G with a proper k-colouring.

  PARAMETER: k.

  QUESTION: Is there a clique (complete subgr.) of size k in G?

- A traditional "hard problem" in parameterized complexity, though,

Multicoloured Clique



- MULTICOLOURED CLIQUE

  INPUT: A graph G with a proper k-colouring.

  PARAMETER: k.

  QUESTION: Is there a clique (complete subgr.) of size k in G?

- A traditional "hard problem" in parameterized complexity, though, we are going to use the following special (polynomial) variant.

Reduction to Multicol. Clique

- Recall... COMPATIBLE EMBEDDING
  - $\mathcal{P}$ with a chain partition $(C_1, \ldots, C_w)$, $w = width(\mathcal{P})$,
  - and $\mathcal{Q}$ with a "chain" mapping $f : Q \rightarrow \{1, \ldots, w\}$.

Reduction to Multicol. Clique

- Recall… COMPATIBLE EMBEDDING
  - $\mathcal{P}$ with a chain partition $(C_1, \dots, C_w)$, $w = width(\mathcal{P})$,
  - and $\mathcal{Q}$ with a "chain" mapping $f : Q \to \{1, \dots, w\}$.

- $\rightsquigarrow$ our $|Q|$-COLOURED CLIQUE of G defined

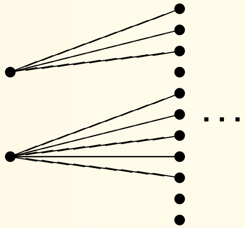
Reduction to Multicol. Clique

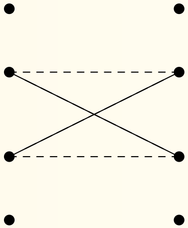
- Recall… Compatible Embedding
  - $\mathcal{P}$ with a chain partition $(C_1, \ldots, C_w)$, $w = width(\mathcal{P})$,
  - and $\mathcal{Q}$ with a "chain" mapping $f : Q \to \{1, \ldots, w\}$.
- $\leadsto$ our $|Q|$-Coloured Clique of G defined
  - $V(G) = V_1 \dot\cup \ldots \dot\cup V_{|Q|}$ where V_i is a copy of $C_{f(i)}$,

Reduction to Multicol. Clique

- Recall... Compatible Embedding
  - $\mathcal{P}$ with a chain partition $(C_1, \ldots, C_w)$, $w = width(\mathcal{P})$,
  - and $\mathcal{Q}$ with a "chain" mapping $f : Q \to \{1, \ldots, w\}$.

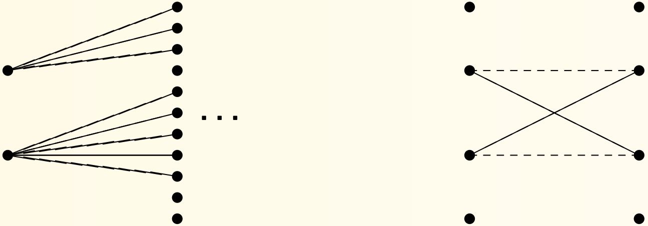
- $\rightsquigarrow$ our $|Q|$-Coloured Clique of G defined
  - $V(G) = V_1 \,\dot\cup \ldots \dot\cup\, V_{|Q|}$ where V_i is a copy of $C_{f(i)}$,
  - for $p \in V_a, q \in V_b$ copies of $p' \in C_{f(a)}, q' \in C_{f(b)}$, $a \neq b$, $pq \in E(G)$ iff $p' \leq_P q' \leftrightarrow a \leq_Q b$ and $p' \geq_P q' \leftrightarrow a \geq_Q b$.





Interval-Monotone Multicol. Clique

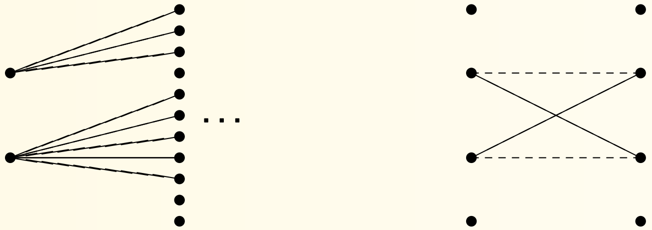
Using special properties of the reduction...



A MULTICOLOURED CLIQUE instance is called interval-monotone if

Interval-Monotone Multicol. Clique

Using special properties of the reduction...

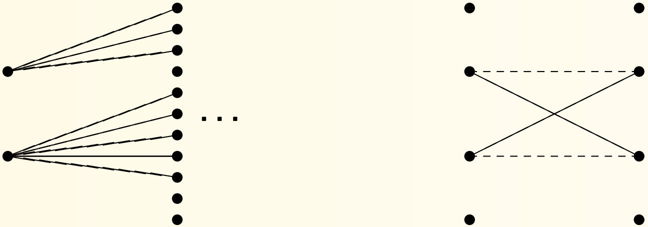


A MULTICOLOURED CLIQUE instance is called interval-monotone if

- the given colour classes are $V(G) = V_1 \cup \cdots \cup V_k$, $E = E(G)$, and
- the vertex set $V(G)$ can be linearly ordered as $\prec$ such that;

Interval-Monotone Multicol. Clique

Using special properties of the reduction...

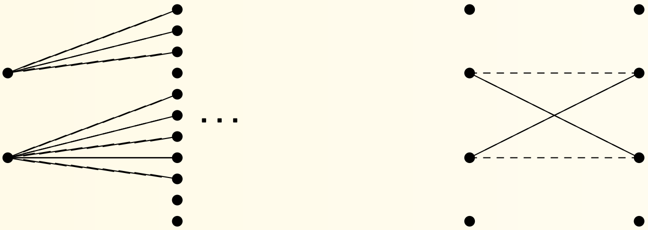


A MULTICOLOURED CLIQUE instance is called interval-monotone if

- the given colour classes are $V(G) = V_1 \cup \cdots \cup V_k$, $E = E(G)$, and
- the vertex set $V(G)$ can be linearly ordered as $\prec$ such that;
  - $\forall p \in V_a,\ \forall q_1 \prec q_2 \prec q_3 \in V_b :\ pq_1, pq_3 \in E \rightarrow pq_2 \in E$.

Interval-Monotone Multicol. Clique

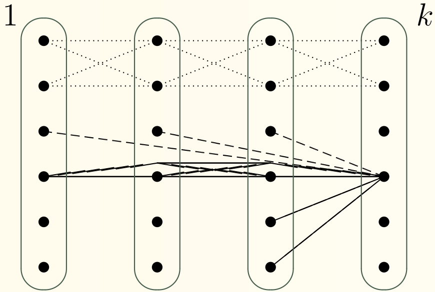
Using special properties of the reduction...



A MULTICOLOURED CLIQUE instance is called interval-monotone if

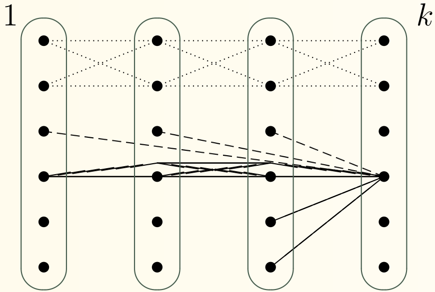
- the given colour classes are $V(G) = V_1 \cup \cdots \cup V_k$, $E = E(G)$, and
- the vertex set $V(G)$ can be linearly ordered as $\prec$ such that;
  - $\forall p \in V_a,\ \forall q_1 \prec q_2 \prec q_3 \in V_b :\ pq_1, pq_3 \in E \rightarrow pq_2 \in E$.
  - $\forall p_1 \prec p_2 \in V_a,\ \forall q_1 \prec q_2 \in V_b :\ p_1q_2, p_2q_1 \in E \rightarrow p_1q_1, p_2q_2 \in E$.

The Algorithm



INPUT: G, coloured $V(G) = V_1 \cup \ldots \cup V_k$, ordered by $\prec$.

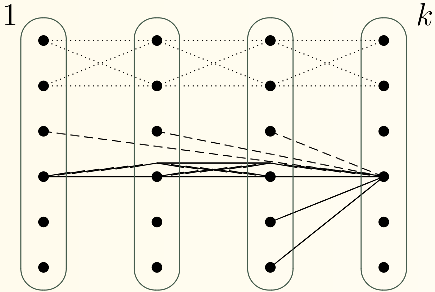
The Algorithm



INPUT: G, coloured $V(G) = V_1 \cup \ldots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \le i \le k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \cdots \cup V_{i-1} \cup \{v\}$.

The Algorithm

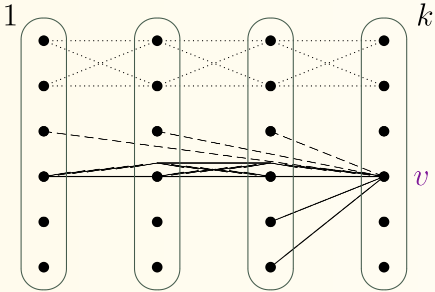


Input: G, coloured $V(G) = V_1 \cup \ldots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \leq i \leq k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \cdots \cup V_{i-1} \cup \{v\}$.
- Dynamically compute this information, for $i = 2, 3, \ldots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i - 1, \ldots, 1$:

The Algorithm

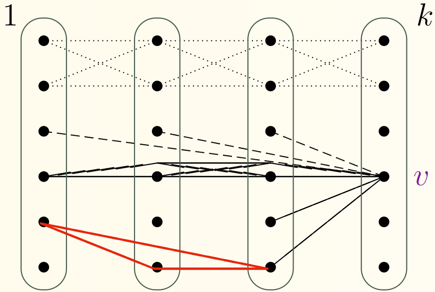


INPUT: G, coloured $V(G) = V_1 \cup \dots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \le i \le k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \dots \cup V_{i-1} \cup \{v\}$.
- Dynamically compute this information, for $i = 2, 3, \dots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i - 1, \dots, 1$:
  - $x :=$ $\prec$-min. neighbour of X in V_j such that $MaxK^j(x) \neq \emptyset$ is in or above the neighbs. of X in each of $V_1, \dots, V_{j-1}$;
  - if x nonexistent, set $X := \emptyset$;

The Algorithm



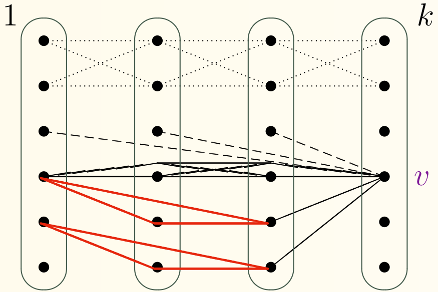
Input: G, coloured $V(G) = V_1 \cup \dots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \le i \le k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \dots \cup V_{i-1} \cup \{v\}$.

- Dynamically compute this information, for $i = 2, 3, \dots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i-1, \dots, 1$:
  - $x :=$ $\prec$-min. neighbour of X in V_j such that $MaxK^j(x) \neq \emptyset$ is in or above the neighbs. of X in each of $V_1, \dots, V_{j-1}$;
  - if x nonexistent, set $X := \emptyset$;

The Algorithm

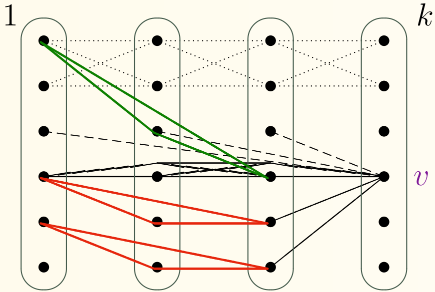


INPUT: G, coloured $V(G) = V_1 \cup \ldots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \le i \le k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \cdots \cup V_{i-1} \cup \{v\}$.
- Dynamically compute this information, for $i = 2, 3, \ldots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i-1, \ldots, 1$:
  - $x :=$ $\prec$-min. neighbour of X in V_j such that $MaxK^j(x) \neq \emptyset$ is in or above the neighbs. of X in each of $V_1, \ldots, V_{j-1}$;
  - if x nonexistent, set $X := \emptyset$;

The Algorithm

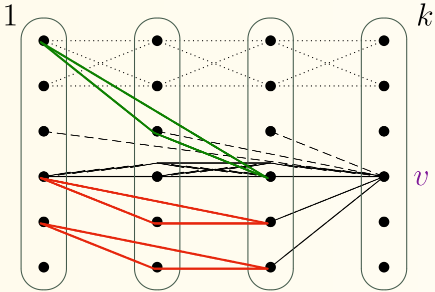


Input: G, coloured $V(G) = V_1 \cup \ldots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \leq i \leq k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \cdots \cup V_{i-1} \cup \{v\}$.
- Dynamically compute this information, for $i = 2, 3, \ldots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i-1, \ldots, 1$:
  - $x :=$ $\prec$-min. neighbour of X in V_j such that $MaxK^j(x) \neq \emptyset$ is in or above the neighbs. of X in each of $V_1, \ldots, V_{j-1}$;
  - if x nonexistent, set $X := \emptyset$;

The Algorithm



INPUT: G, coloured $V(G) = V_1 \cup \dots \cup V_k$, ordered by $\prec$.

- Define $MinK^i(v)$, $MaxK^i(v)$ where $2 \leq i \leq k$ and $v \in V_i$:
  - $MinK^i(v) :=$ the $\prec$-minimum of all the i-cliques which are contained in $V_1 \cup \dots \cup V_{i-1} \cup \{v\}$.
- Dynamically compute this information, for $i = 2, 3, \dots, k$, as follows.

  For every $v \in V_i$, set $X := \{v\}$, and repeat for $j = i-1, \dots, 1$:
  - $x :=$ $\prec$-min. neighbour of X in V_j such that $MaxK^j(x) \neq \emptyset$ is in or above the neighbs. of X in each of $V_1, \dots, V_{j-1}$;
  - if x nonexistent, set $X := \emptyset$;
  - $MinK^i(v) := X$ after finishing the iterations (of j).

4 Summary

- Improvement over [Bova, Ganian, Szeider, LICS 2014]

[BGS14]	$O(f(\phi) \cdot n^{g(w)})$
Algorithm 1 (using CSP)	$O(f'(\phi, w) \cdot n^4)$
Algorithm 2 (using mult. clique)	$O(f''(\phi, w) \cdot n^2)$

4 Summary

- Improvement over [Bova, Ganian, Szeider, LICS 2014]

[BGS14]	$O(f(\phi) \cdot n^{g(w)})$
Algorithm 1 (using CSP)	$O(f'(\phi, w) \cdot n^4)$
Algorithm 2 (using mult. clique)	$O(f''(\phi, w) \cdot n^2)$

  – our algorithms are FPT both in ϕ the width of the poset.

- Straightforward, simpler, and self-contained proofs (Alg. 2).

  Non-complicated algorithm (Alg. 2) – "implementable".

4 Summary

- Improvement over [Bova, Ganian, Szeider, LICS 2014]

[BGS14]	$O(f(\phi) \cdot n^{g(w)})$
Algorithm 1 (using CSP)	$O(f'(\phi, w) \cdot n^4)$
Algorithm 2 (using mult. clique)	$O(f''(\phi, w) \cdot n^2)$

  – our algorithms are FPT both in ϕ the width of the poset.

- Straightforward, simpler, and self-contained proofs (Alg. 2).

  Non-complicated algorithm (Alg. 2) – "implementable".

- Current work:

  – extension to full FO logic.

4 Summary

- Improvement over [Bova, Ganian, Szeider, LICS 2014]

[BGS14]	$O(f(\phi) \cdot n^{g(w)})$
Algorithm 1 (using CSP)	$O(f'(\phi, w) \cdot n^4)$
Algorithm 2 (using mult. clique)	$O(f''(\phi, w) \cdot n^2)$

  – our algorithms are FPT both in ϕ the width of the poset.

- Straightforward, simpler, and self-contained proofs (Alg. 2).

  Non-complicated algorithm (Alg. 2) – “implementable”.

- Current work:

  – extension to full FO logic.

THANK YOU FOR YOUR ATTENTION.