Semilattice Polymorphisms on Reflexive Graphs

Mark Siggers (joint work with Pavol Hell)

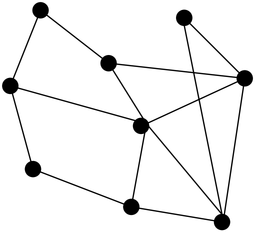
Kyungpook National University

August 21, 2009

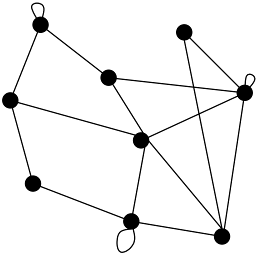
Outline

- Polymorphisms
  - why we care about them
  - what are they
- Reflexive Graphs
- Semilattice Polymorphisms
- Semilattice Polymorphisms on Reflexive Graphs
- Chordal Reducible Graphs

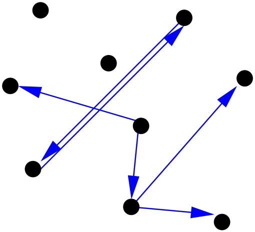
Polymorphisms (why we care about them)



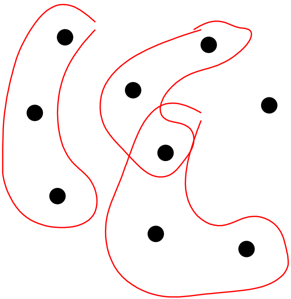
For every relational structure H



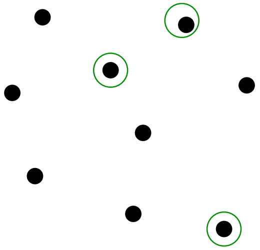
For every relational structure H



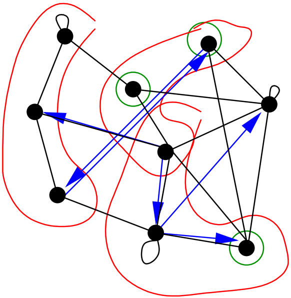
For every relational structure H



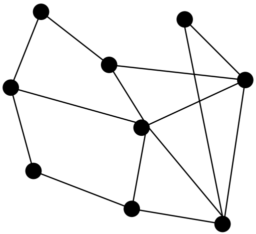
For every relational structure H



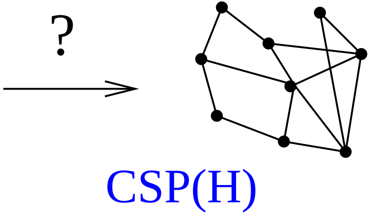
For every relational structure H



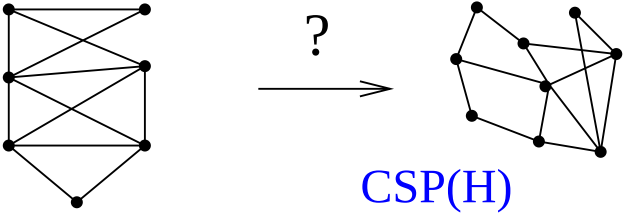
For every relational structure H



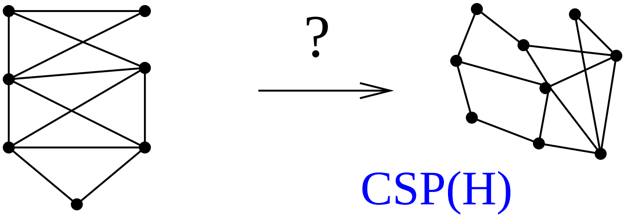
For every relational structure H



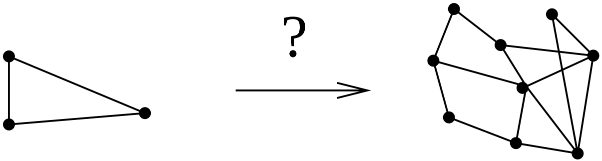
For every relational structure H there is a computational problem $\mathrm{CSP}(H)$.



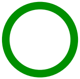
For every relational structure H there is a computational problem $\mathrm{CSP}(H)$.

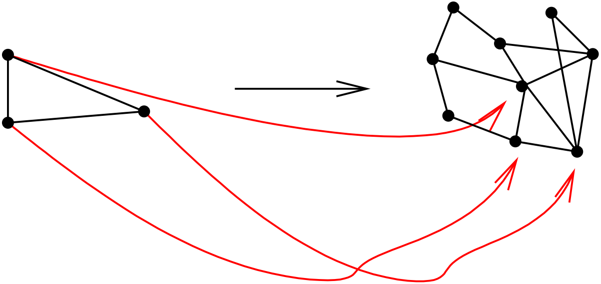


A homomorphism $\phi : G \to H$ is a vertex map that preserves relations.

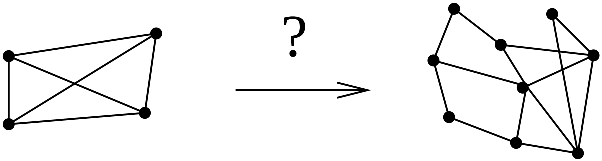


A homomorphism $\phi : G \rightarrow H$ is a vertex map that preserves relations.

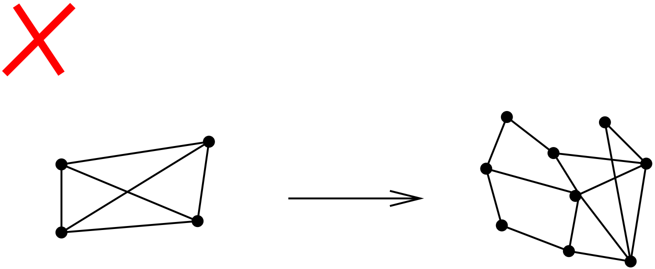




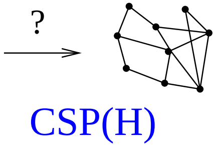
A homomorphism $\phi : G \to H$ is a vertex map that preserves relations.



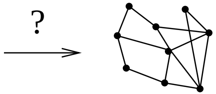
A homomorphism $\phi : G \to H$ is a vertex map that preserves relations.



A homomorphism $\phi : G \rightarrow H$ is a vertex map that preserves relations.

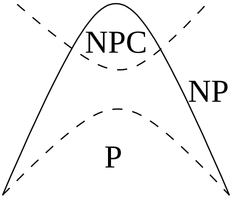


We are interested in the computational complexity of $\mathrm{CSP}(H)$.

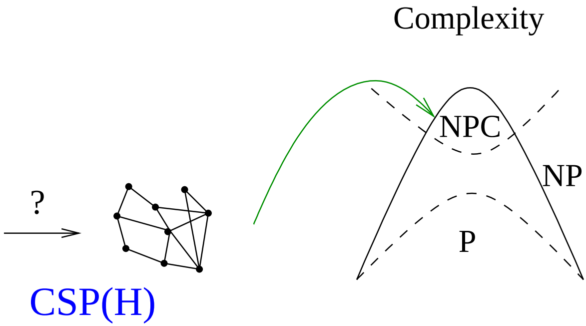


CSP(H)

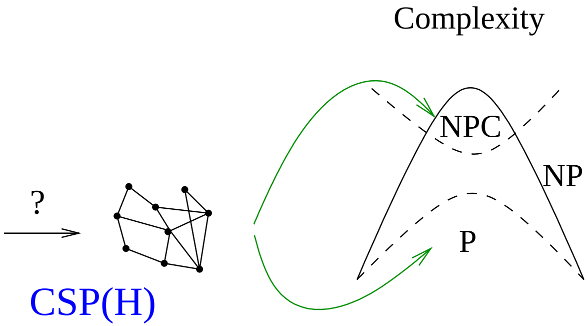
Complexity



We are interested in the computational complexity of $\mathrm{CSP}(H)$.

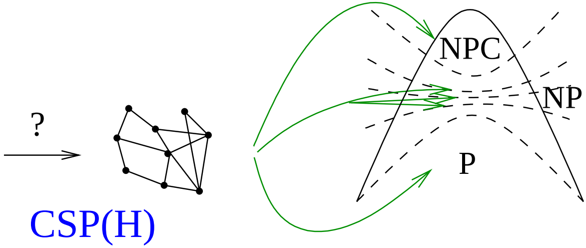


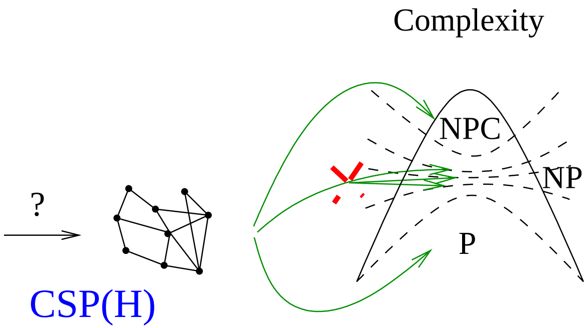
We are interested in the computational complexity of $\mathrm{CSP}(H)$.



We are interested in the computational complexity of $\mathrm{CSP}(H)$.

Complexity





CSP Dichotomy Conjecture [Feder, Vardi '99]

For any H, $\mathrm{CSP}(H)$ is in either P or NPC.

The CSP Dichotomy Conjecture is true ...

- for structures on two vertices. Schaefer '78
- for graphs. Hell, Nešetřil '92
- for structures on three vertices. Bulatov '02
- for conservative structures (list-colouring). Bulatov '06
- for digraphs without sources or sinks. Barto, Kozik, Niven '09

Theorem Jeavons '00

The complexity of $\mathrm{CSP}(H)$ is determined by the polymorphisms of H.

Polymorphisms (what are they)

Definition: Polymorphism

A *polymorphism of* H is d-ary operation on $V(H)$ that is compatible with relations of H.

$\phi : V(H) \times \cdots \times V(H) \to V(H)$

Definition: Polymorphism

A *polymorphism of H* is d-ary operation on $V(H)$ that is compatible with relations of H.

$\phi : V(H) \times \cdots \times V(H) \to V(H)$

$$\phi : \quad (u_1, \ldots, u_d) \quad \mapsto \quad \phi(u_1, \ldots, u_d)$$

Definition: Polymorphism

A *polymorphism of H* is d-ary operation on $V(H)$ that is compatible with relations of H.

$$\phi : V(H) \times \cdots \times V(H) \to V(H)$$

$$\phi : \quad \begin{matrix} (u_1, \ldots, u_d) \\ (v_1, \ldots, v_d) \end{matrix} \quad \mapsto \quad \begin{matrix} \phi(u_1, \ldots, u_d) \\ \phi(v_1, \ldots, v_d) \end{matrix}$$

Definition: Polymorphism

A *polymorphism of H* is d-ary operation on $V(H)$ that is compatible with relations of H.

$$\phi : V(H) \times \cdots \times V(H) \to V(H)$$

$$\phi : \quad \begin{matrix} (\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}, \ldots, \begin{bmatrix} u_d \\ v_d \end{bmatrix}) \end{matrix} \quad \mapsto \quad \begin{matrix} \phi(u_1, \ldots, u_d) \\ \phi(v_1, \ldots, v_d) \end{matrix}$$

Definition: Polymorphism

A *polymorphism of H* is d-ary operation on $V(H)$ that is compatible with relations of H.

$\phi : V(H) \times \cdots \times V(H) \to V(H)$

$$\phi : \quad \begin{matrix} (\begin{bmatrix} u_1 \\ v_1 \end{bmatrix}, \ldots, \begin{bmatrix} u_d \\ v_d \end{bmatrix}) \end{matrix} \quad \mapsto \quad \begin{bmatrix} \phi(u_1, \ldots, u_d) \\ \phi(v_1, \ldots, v_d) \end{bmatrix}$$

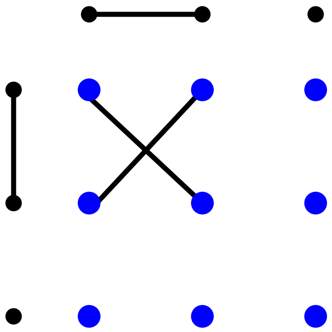
Equivalent Definition: Polymorphism

A *polymorphism of* H is a homomorphism of H^d to H.

Equivalent Definition: Polymorphism

A *polymorphism of H* is a homomorphism of H^d to H.

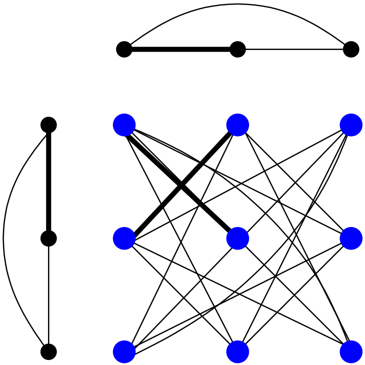
The categorical product H^2:



Equivalent Definition: Polymorphism

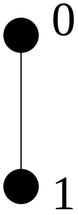
A *polymorphism of H* is a homomorphism of H^d to H.

The categorical product H^2:



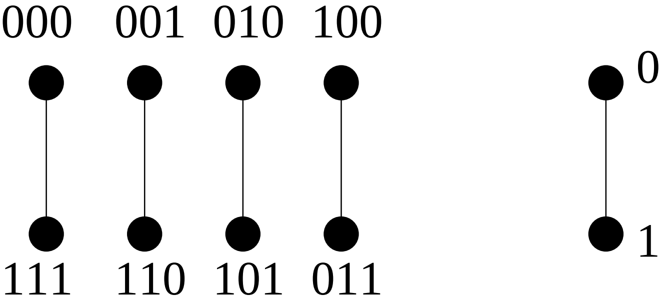
Example: The 3-ary polymorphisms of K_2

$\mathrm{Pol}(K_2)$

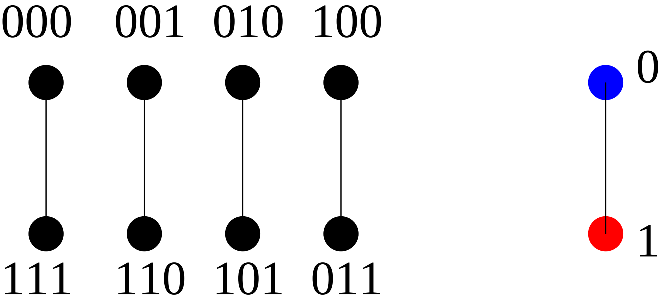


Example: The 3-ary polymorphisms of K_2

Pol(K2)

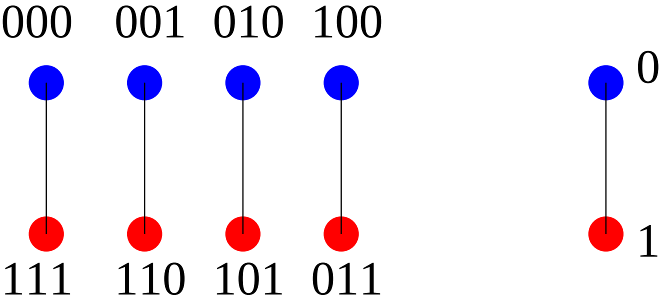


Example: The 3-ary polymorphisms of K_2

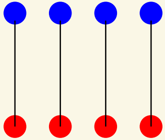


Pol(K_2)

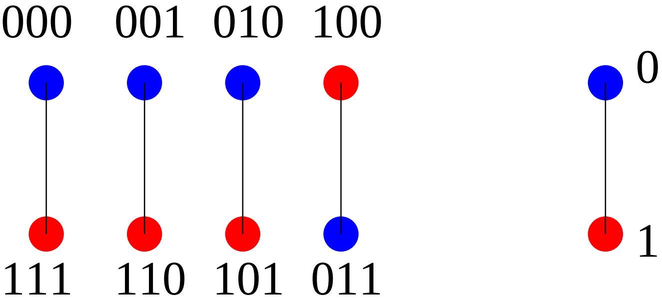
Example: The 3-ary polymorphisms of K_2

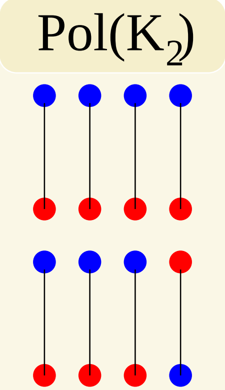


Pol(K_2)

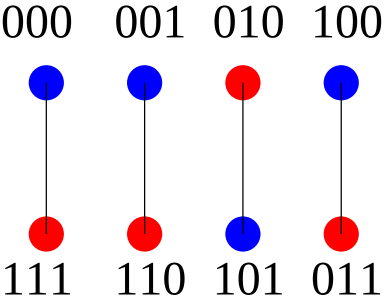


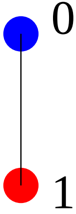
Example: The 3-ary polymorphisms of K_2



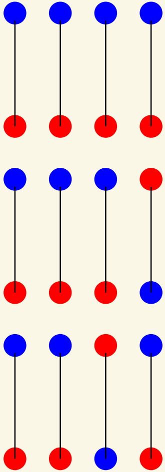


Example: The 3-ary polymorphisms of K_2

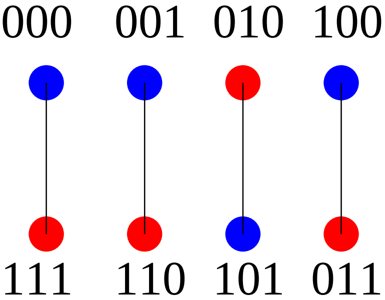


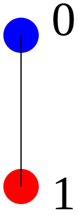


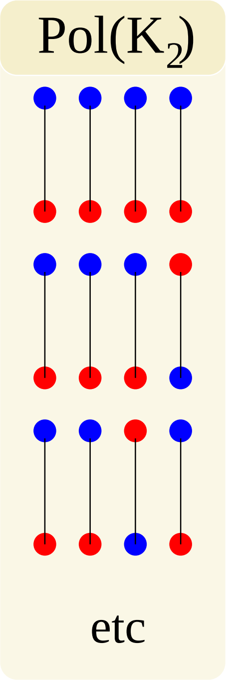
Pol(K_2)



Example: The 3-ary polymorphisms of K_2

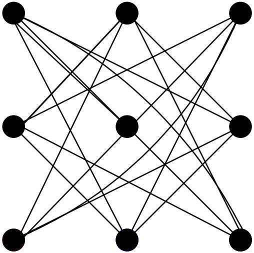


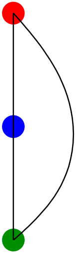


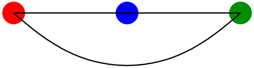


Example: The 2-ary polymorphisms of K_3

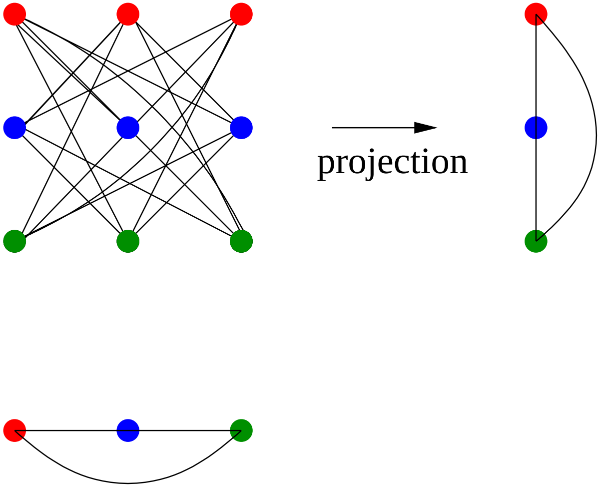
Pol(K_3)

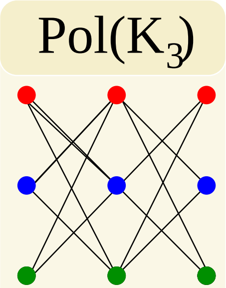




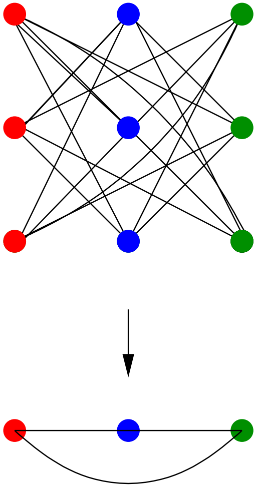


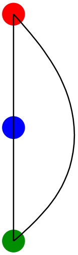
Example: The 2-ary polymorphisms of K_3



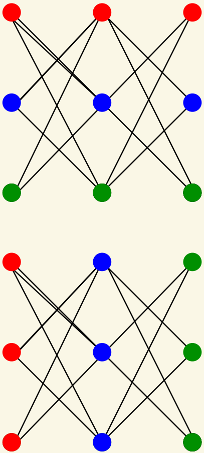


Example: The 2-ary polymorphisms of K_3





Pol(K_3)



Theorem Jeavons '00

If $\mathrm{Pol}(H)$ contains only projections, then $\mathrm{CSP}(H)$ is in NPC.

A polymorphism $\phi : H^d \to H$ is

WNU (weak near-unanimity)

if

$$\begin{aligned} \phi(x, x, \ldots, x, y) &= \phi(x, x, \ldots, y, x) = \ldots \\ &= \phi(y, x, \ldots, x, x) \end{aligned}$$

for all $x, y \in V(H)$.

A polymorphism $\phi : H^d \to H$ is

WNU (weak near-unanimity)

if

$$\begin{aligned} \phi(x, x, \ldots, x, y) &= \phi(x, x, \ldots, y, x) = \ldots \\ &= \phi(y, x, \ldots, x, x) \end{aligned}$$

for all $x, y \in V(H)$.

Conjecture: [BJK'02; MM'08]

CSP(H) is in NPC if H admits no WNU polymorphisms, and is otherwise polynomial time solvable.

BJK: Bulatov, Jeavons, Krokhin
MM: Maroti, McKenzie

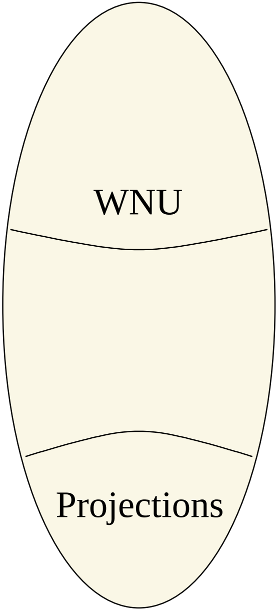
A polymorphism $\phi : H^d \to H$ is

WNU (weak near-unanimity)

if

$$\begin{aligned}\phi(x,x,\ldots,x,y) &= \phi(x,x,\ldots,y,x) = \ldots \\ &= \phi(y,x,\ldots,x,x)\end{aligned}$$

for all $x, y \in V(H)$.



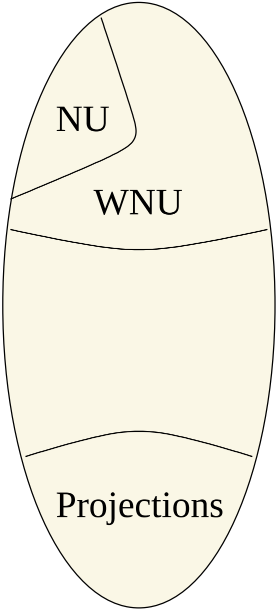
A polymorphism $\phi : H^d \to H$ is

NU (near-unanimity)

if

$$\begin{aligned}\phi(x,x,\ldots,x,y) &= \phi(x,x,\ldots,y,x) = \ldots \\ &= \phi(y,x,\ldots,x,x) = x\end{aligned}$$

for all $x, y \in V(H)$.



A polymorphism $\phi : H^d \to H$ is

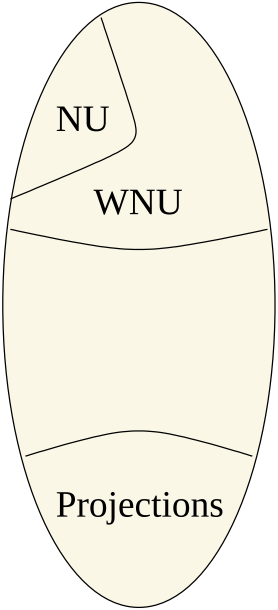
NU (near-unanimity)

if

$$\begin{aligned}\phi(x, x, \dots, x, y) &= \phi(x, x, \dots, y, x) = \dots \\ &= \phi(y, x, \dots, x, x) = x\end{aligned}$$

for all $x, y \in V(H)$.

If H admits an NU polymorphism, then CSP(H) is polynomial time solvable.



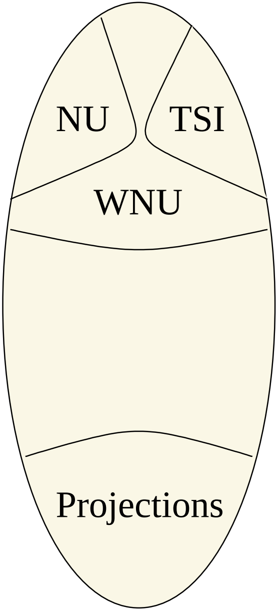
A polymorphism $\phi : H^d \to H$ is

TSI (totally symmetric idempotent)

if

$$\phi(u_1, \ldots, u_d) = \phi(v_1, \ldots, v_d)$$

whenever $\{u_1, \ldots, u_d\} = \{v_1, \ldots, v_d\}$ as sets.



A polymorphism $\phi : H^d \to H$ is

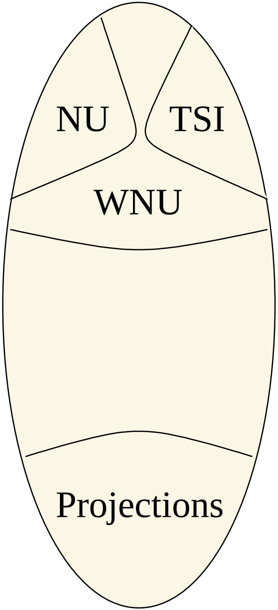
TSI (totally symmetric idempotent)

if

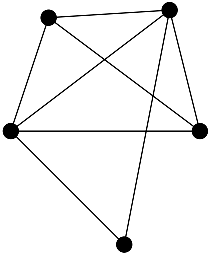
$$\phi(u_1, \ldots, u_d) = \phi(v_1, \ldots, v_d)$$

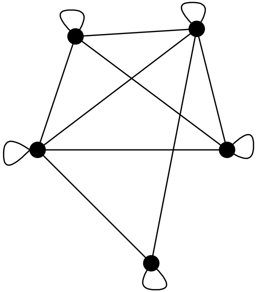
whenever $\{u_1, \ldots, u_d\} = \{v_1, \ldots, v_d\}$ as sets.

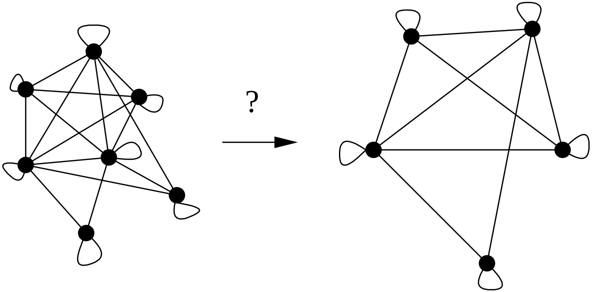
If H admits a TSI polymorphism, then CSP(H) is polynomial time solvable.

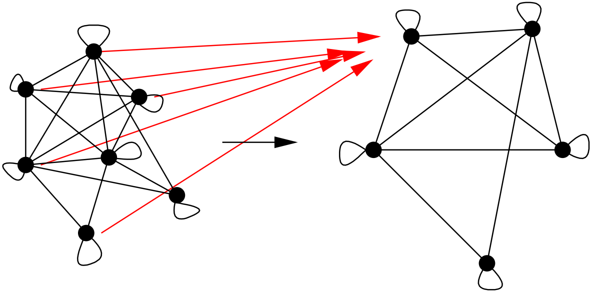


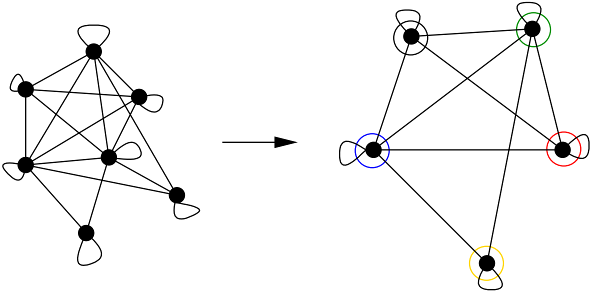
Reflexive Graphs

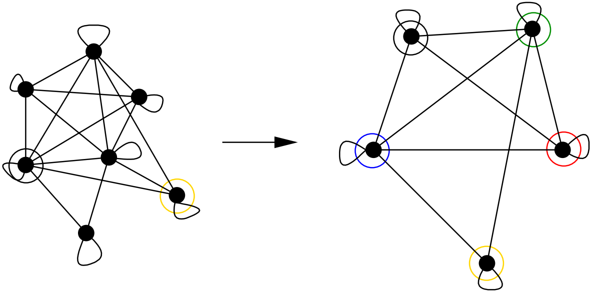


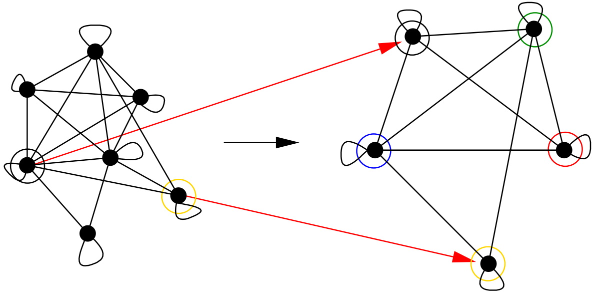






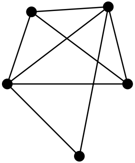




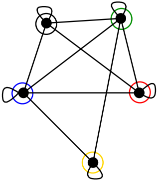


Assume all graphs are connected, reflexive and have all singleton unary relations.

We draw



to mean



Why Reflexive Graphs?

- Dichotomy is done for irreflexive graphs, and hard for digraphs. Reflexive graphs are somewhere in between.
- Dichotomy is done for MinHOM of reflexive graphs. [GHRY '07]. (Infact for digraphs with possible loops.)
- Reflexive graphs admitting NU polymorphisms have been characterised. [BFHHM '06; LLT '06].

Why Reflexive Graphs?

- Dichotomy is done for irreflexive graphs, and hard for digraphs. Reflexive graphs are somewhere in between.
- Dichotomy is done for MinHOM of reflexive graphs. [GHRY '07]. (Infact for digraphs with possible loops.)
- Reflexive graphs admitting NU polymorphisms have been characterised. [BFHHM '06; LLT '06].

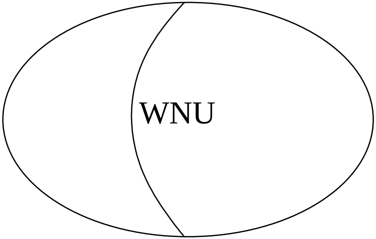
GHRY: Gutin Hell Rafiey Yeo

Why Reflexive Graphs?

- Dichotomy is done for irreflexive graphs, and hard for digraphs. Reflexive graphs are somewhere in between.
- Dichotomy is done for MinHOM of reflexive graphs. [GHRY '07]. (Infact for digraphs with possible loops.)
- Reflexive graphs admitting NU polymorphisms have been characterised. [BFHHM '06; LLT '06].

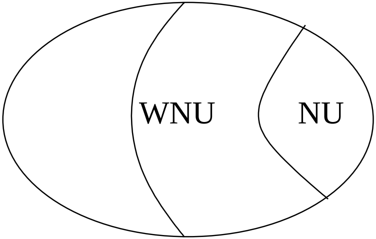
BFHHM: Brewster Feder Hell Huang MacGillivray;
LLT: Larose Loten Tardif

Reflexive Graphs



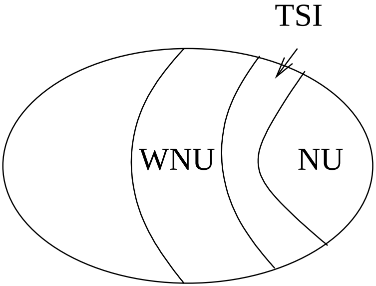
Towards dichotomy on reflexive graphs, we want to know what graphs admit WNU.

Reflexive Graphs



[LLT06] characterised those admitting NU.

Reflexive Graphs

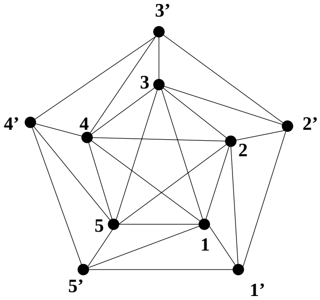


[LLT06] characterised those admitting NU.

Goals

- Characterise reflexive graphs admitting TSI of all arities.
- Characterise reflexive graphs admitting TSI.
- Characterise reflexive graphs admitting WNU.
- Prove Dichotomy for reflexive graphs.

Semilattice Polymorphisms



Let ϕ be defined by

- idempotence.
- maximality on non-primed vertices (ties t
  min label)
- for mix of primed and non-primed entries
  ignore the primed entries and do as in th
  previous step.
- If all entries are primed then
  - if they are i' and $(i+1)'$, go to $i +$
  - if they are $(i-1)'$ and $(i+1)'$, go t
    i
  - if they are $(i-1)'$, $(i)'$ and $(i+1)'$
    go to i
  - otherwise, remove their primes (ie,
    read i' as i) and go to the min entr

Definition

A 2-ary polymorphism $\phi : H^2 \to H$ is SL (semilattice) if it is idempotent, associative and commutative.

Definition

A 2-ary polymorphism $\phi : H^2 \to H$ is SL (semilattice) if it is idempotent, associative and commutative.

Such an operation is called semilattice because the partial ordering

$$u < v \text{ if } \phi(u, v) = u$$

of $V(H)$ is a meet semilattice .

Definition

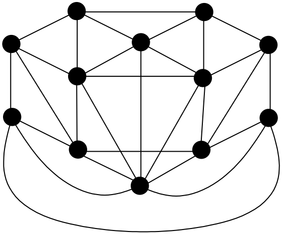
A 2-ary polymorphism $\phi : H^2 \to H$ is SL (semilattice) if it is idempotent, associative and commutative.

$$u < v \text{ if } \phi(u, v) = u$$

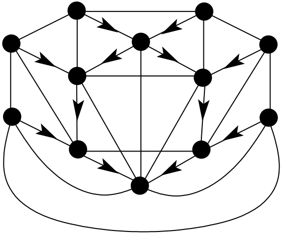
Where $\wedge$ is the associated meet, we have

$$\phi(u, v) = u \wedge v,$$

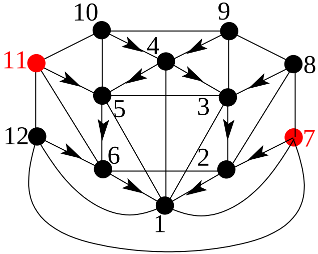
so we will denote SL polymorphisms by $\wedge$.



Semilattice Polymorphisms are easy to represent.

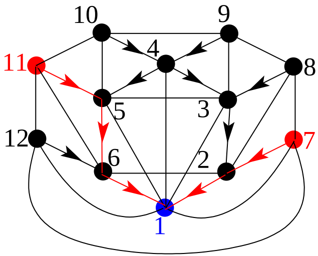


Semilattice Polymorphisms are easy to represent.



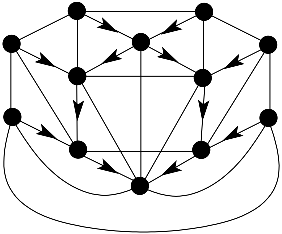
$11 \wedge 7$

Semilattice Polymorphisms are easy to represent.



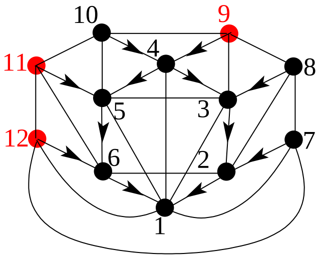
$11 \wedge 7 = 1$

Semilattice Polymorphisms are easy to represent.



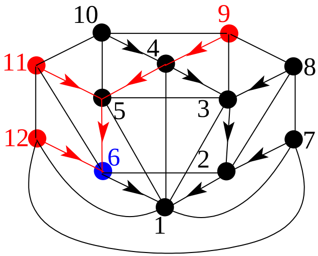
$$\phi : (v_1, \ldots, v_d) \mapsto v_1 \wedge \cdots \wedge v_d$$

And define TSI of every arity.



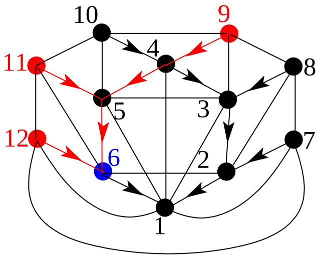
$\phi(9, 11, 12)$

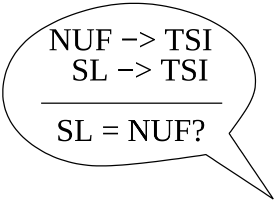
And define TSI of every arity.



$\phi(9, 11, 12) = 9 \wedge 11 \wedge 12 = 6$

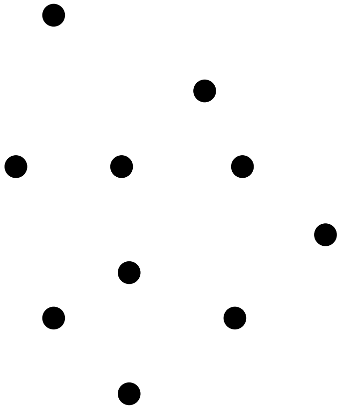
And define TSI of every arity.



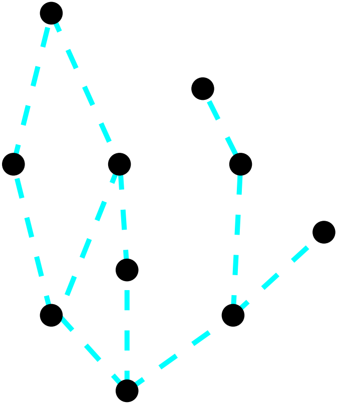


And define TSI of every arity.

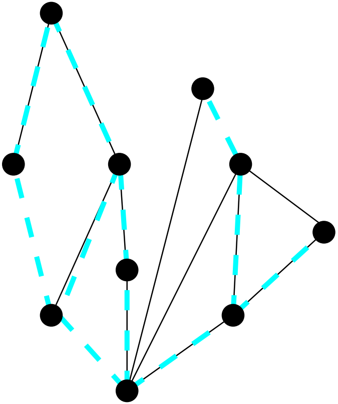
Semilattice Polymorphisms on Reflexive Graphs



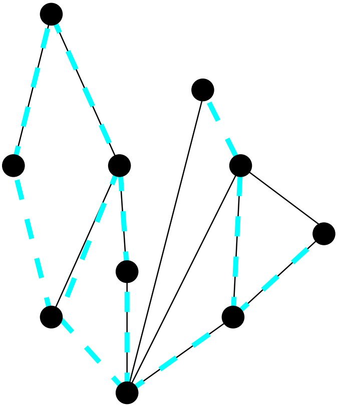
Given some vertices,



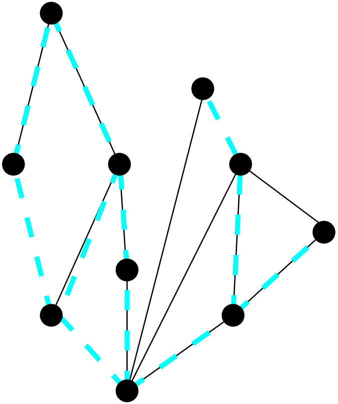
Given some vertices, a semilattice ordering,



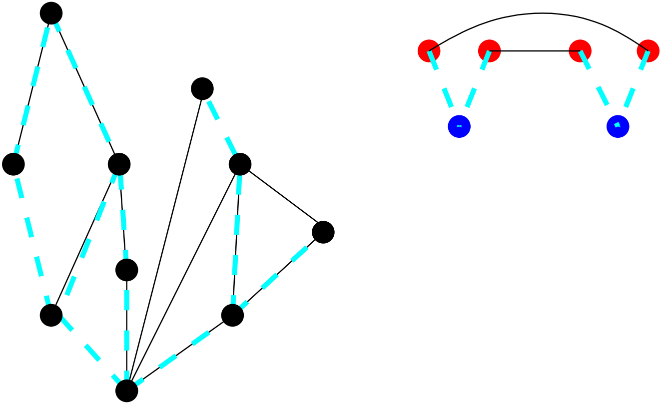
Given some vertices, a semilattice ordering, and a reflexive graph on the vertices,



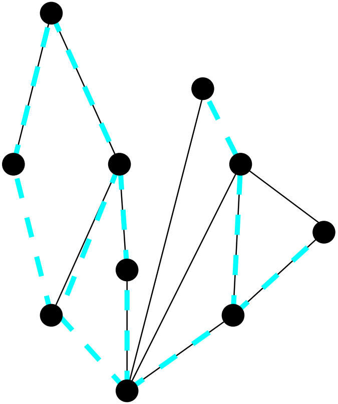
Given some vertices, a semilattice ordering, and a reflexive graph on the vertices, Is the semilattice *polymorphic?*

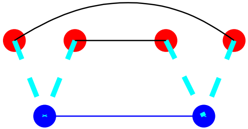


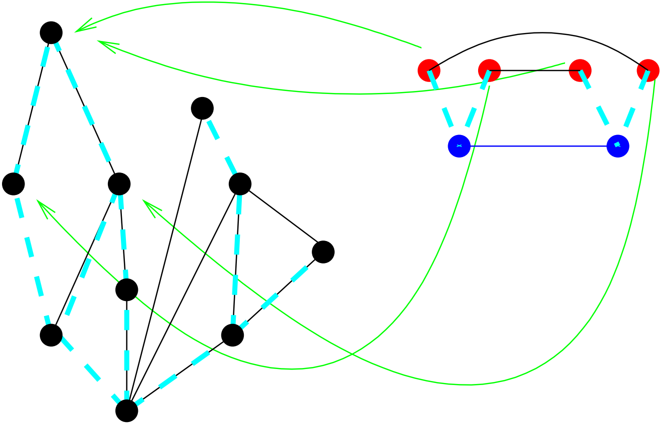
Polymorphism: $u \sim u', v \sim v' \Rightarrow u \wedge v \sim u' \wedge v'$

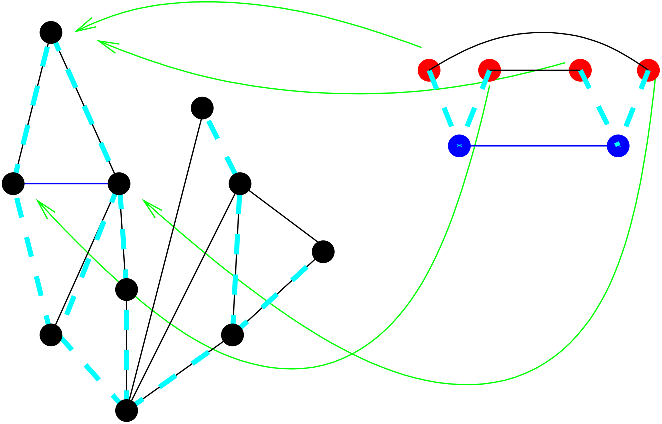


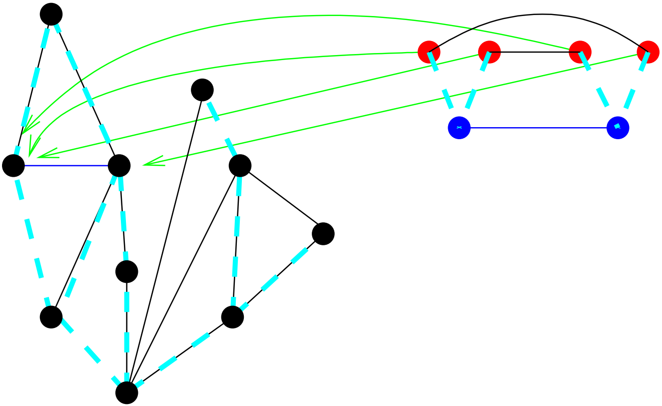
Polymorphism: $u \sim u', v \sim v' \Rightarrow u \wedge v \sim u' \wedge v'$

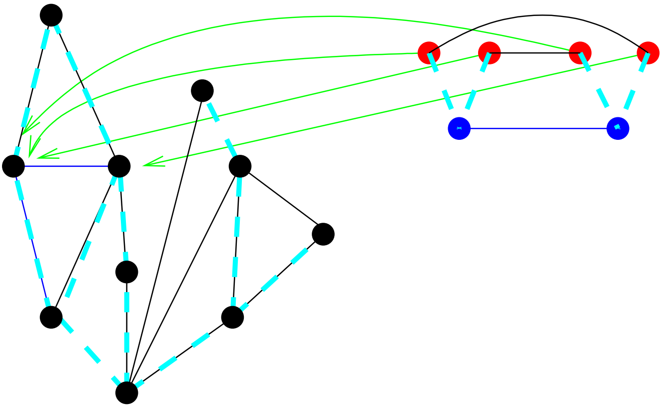


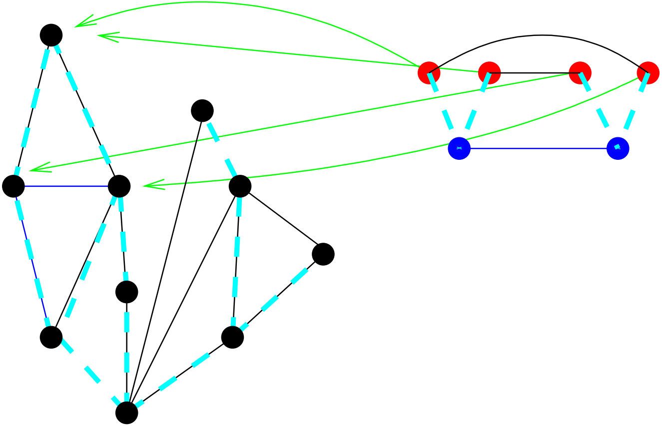


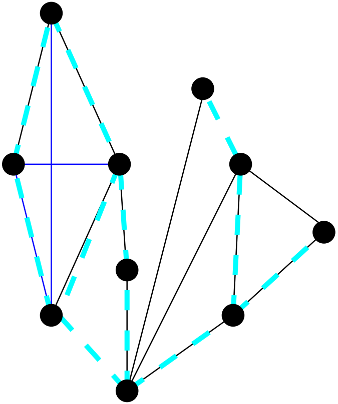


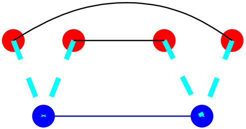


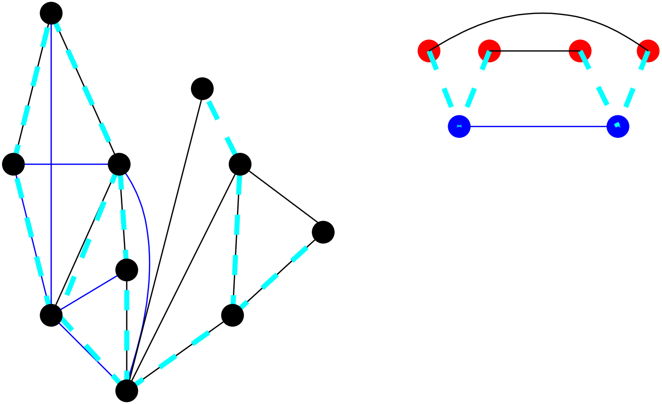


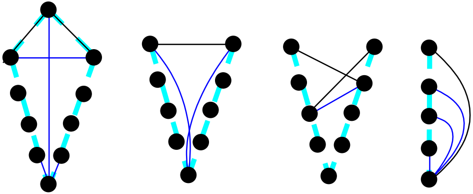




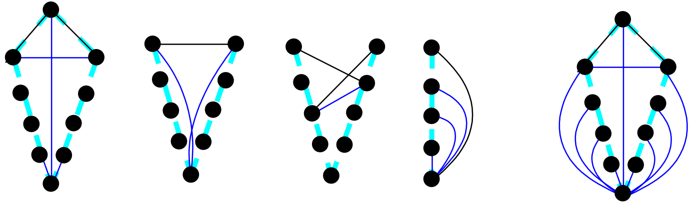




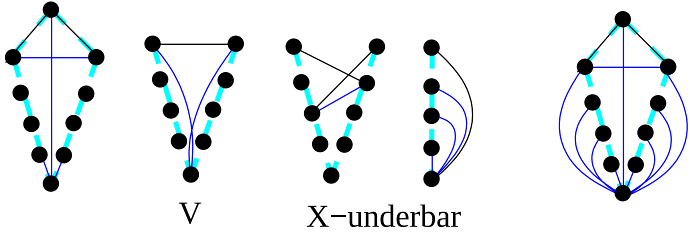




Consequential identities.

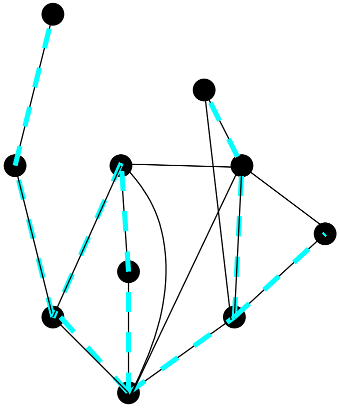


Consequential identities.



Consequential identities.

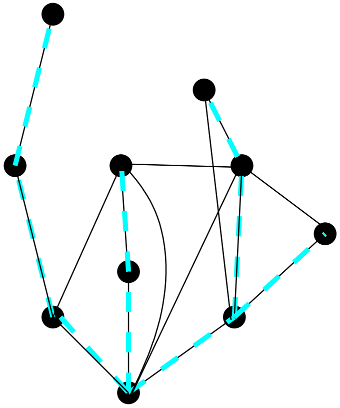
Types of Semilattice Polymorphisms



A semilattice polymorphism is ...

- embedded if every *Hasse* edge (blue edge) is a graph edge.

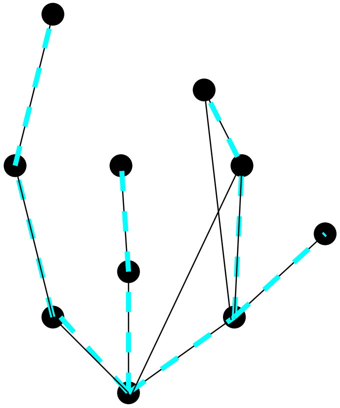
Types of Semilattice Polymorphisms



A semilattice polymorphism is ...

- embedded if every *Hasse* edge (blue edge) is a graph edge.
- tree if the Hasse edges induce a tree.

Types of Semilattice Polymorphisms

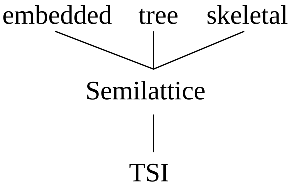


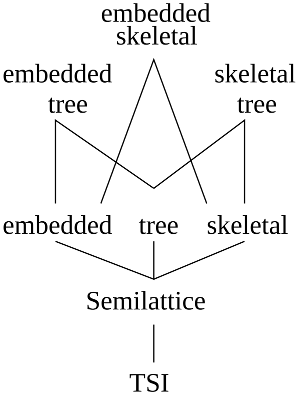
A semilattice polymorphism is ...

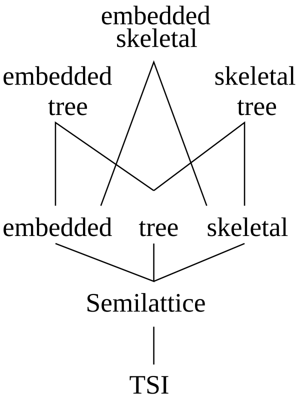
- embedded if every *Hasse* edge (blue edge) is a graph edge.
- tree if the Hasse edges induce a tree.
- skeletal if all graph edges are between comparible vertices.

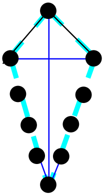
Semilattice

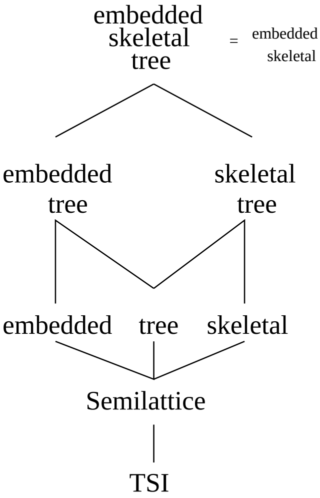
TSI

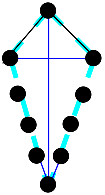

embedded
tree
skeletal
Semilattice
TSI

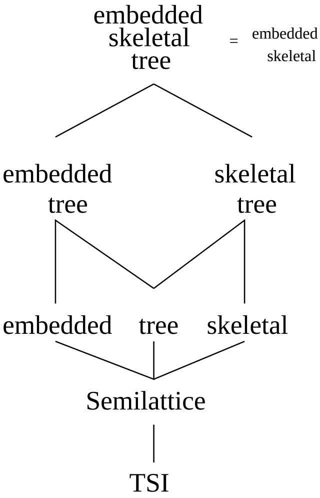

embedded
skeletal
embedded
tree
skeletal
tree
embedded
tree
skeletal
Semilattice
TSI


embedded
skeletal
embedded
tree
skeletal
tree
embedded
tree
skeletal
Semilattice
TSI




embedded
skeletal
tree
=
embedded
skeletal
embedded
tree
skeletal
tree
embedded
tree
skeletal
Semilattice
TSI

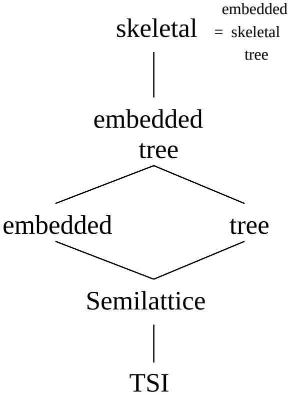




H admits a skeletal SL

$\Rightarrow$

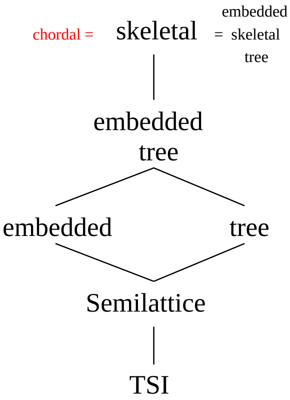
H admits an embedded skeletal tree SL



H admits a skeletal SL

$\Rightarrow$

H admits an embedded skeletal tree SL



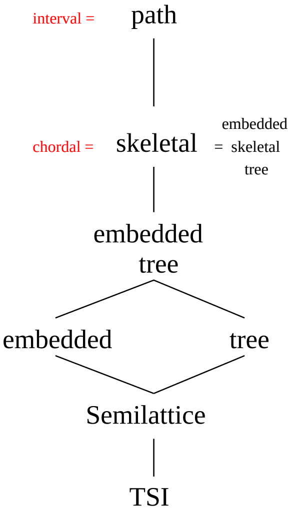
H admits a skeletal SL

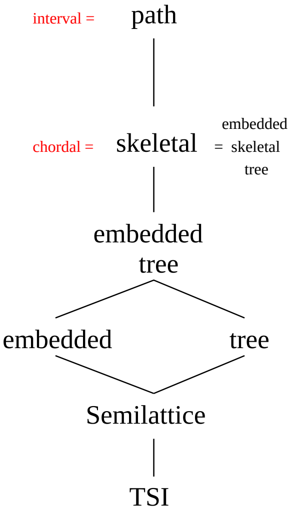
$\Rightarrow$

H is chordal

$\Rightarrow$

H admits an embedded skeletal tree SL


interval =
path
chordal =
skeletal
embedded
= skeletal
tree
embedded
tree
embedded
tree
Semilattice
TSI

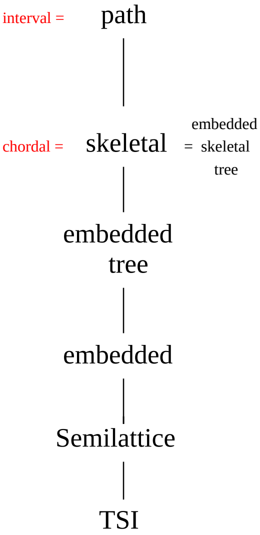


Proposition

H admits a tree SL,

$$\Rightarrow$$

H admits an embedded tree SL.

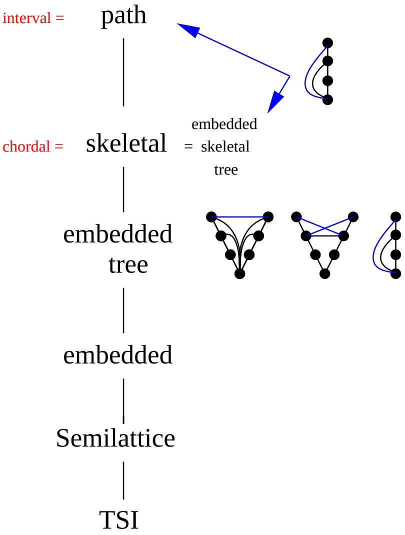


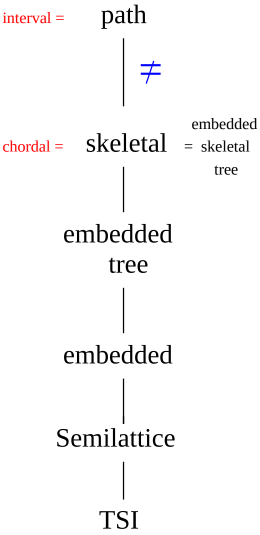
Proposition

H admits a tree SL,

$$\Rightarrow$$

H admits an embedded tree SL.


interval =
path
chordal =
skeletal
embedded
= skeletal
tree
embedded
tree
embedded
Semilattice
TSI


interval =
path
≠
chordal =
skeletal
= embedded skeletal tree
embedded tree
embedded
Semilattice
TSI

interval = path

$\neq$

chordal = skeletal = embedded skeletal tree

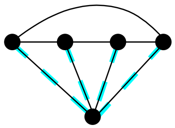
$\neq$

embedded tree

embedded

Semilattice

TSI



interval = path

≠

chordal = skeletal = embedded skeletal tree

≠

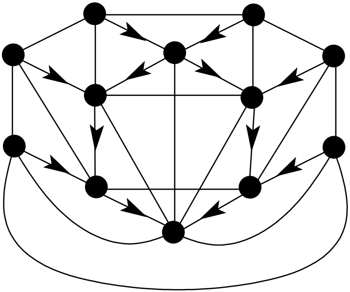
embedded tree

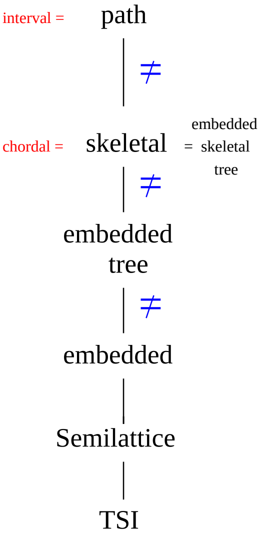
≠

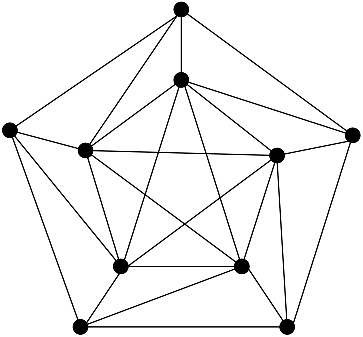
embedded

Semilattice

TSI




interval =
path
≠
chordal =
skeletal
embedded
= skeletal
tree
≠
embedded
tree
≠
embedded
Semilattice
TSI



interval = path

≠

chordal = skeletal = embedded skeletal tree

≠

embedded tree

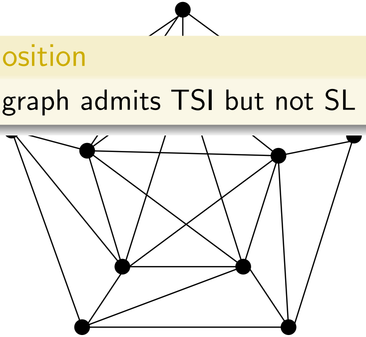
≠

embedded

Semilattice

≠

TSI



Proposition

This graph admits TSI but not SL

interval = path

$\neq$

chordal = skeletal = embedded skeletal tree

$\neq$

embedded tree

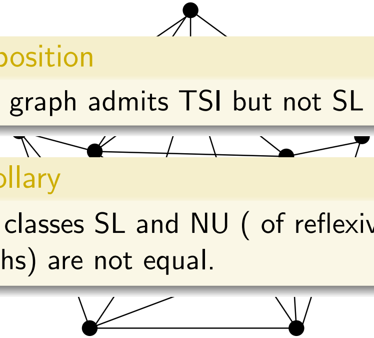
$\neq$

embedded

Semilattice

$\neq$

TSI

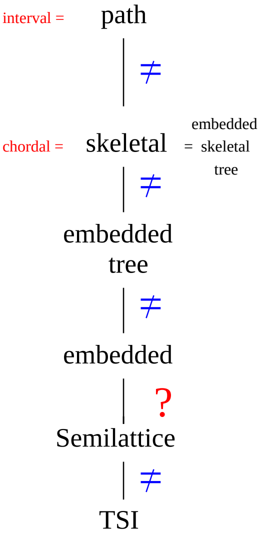


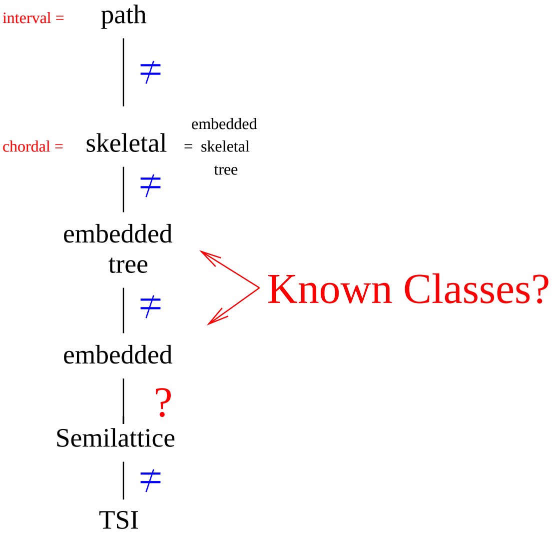
Proposition

This graph admits TSI but not SL

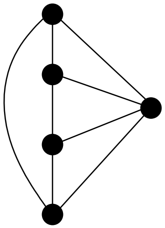
Corollary

The classes SL and NU (of reflexive graphs) are not equal.

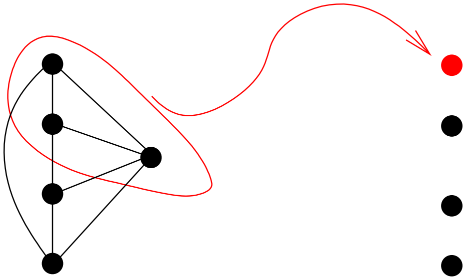

interval =
path
≠
chordal =
skeletal
= embedded skeletal tree
≠
embedded tree
≠
embedded
?
Semilattice
≠
TSI


interval =
path
≠
chordal =
skeletal
= embedded skeletal tree
≠
embedded tree
≠
embedded
?
Semilattice
≠
TSI
Known Classes?

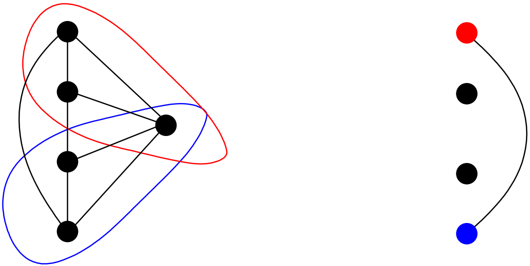
Chordal Reducible Graphs



Given a graph H,

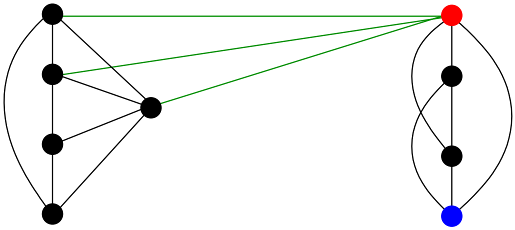


Given a graph H, take its clique graph $\mathrm{CL}(H)$,

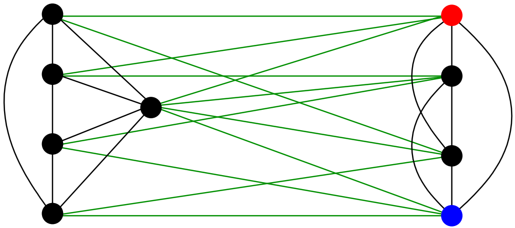


Given a graph H, take its clique graph $\mathrm{CL}(H)$,

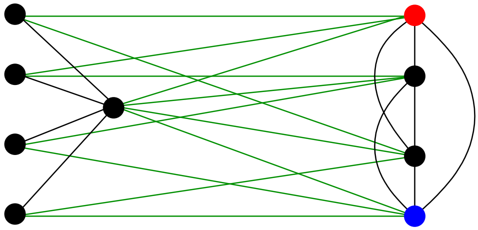
Given a graph H, take its clique graph $\mathrm{CL}(H)$,



Given a graph H, take its clique graph $\mathrm{CL}(H)$, and add edges between them accoring to incidence: $\mathrm{CR}(H)$.



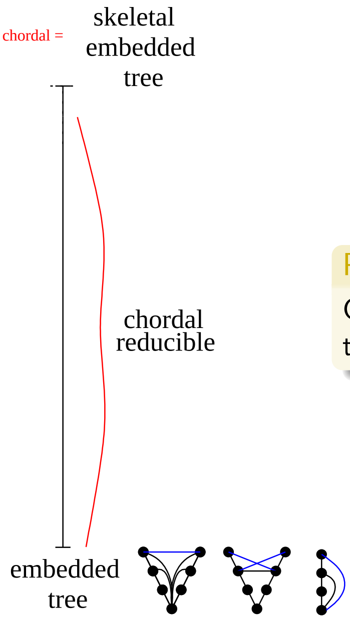
Given a graph H, take its clique graph $\mathrm{CL}(H)$, and add edges between them accoring to incidence: $\mathrm{CR}(H)$.



If we can remove edges from H such that it remains connected, and the full graph $\mathrm{CR}^*(H)$ is chordal, then H is chordal reducible .

- Chordal graphs are chordal reducible.
- Graphs with a universal vertex are chordal reducible.
- Chordal reducible graphs have NU of some arity.

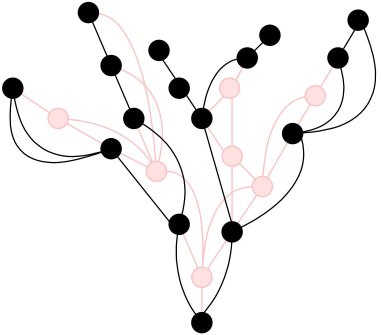
- Chordal graphs are chordal reducible.
- Graphs with a universal vertex are chordal reducible.
- Chordal reducible graphs have NU of some arity.
- Is there a poly time algorithm for recognising chordal reducible graphs?
- Are all graphs with 4-NU chordal reducible?
- Do chordal reducible graphs fit into our heirarchy?



Proposition

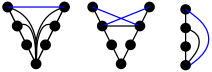
Chordal reducible graphs admit embedded tree polymorphisms.

chordal = skeletal embedded tree

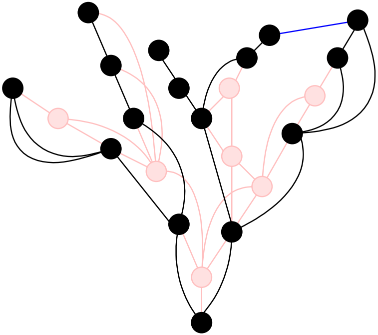


chordal reducible

embedded tree

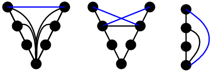


chordal = skeletal embedded tree



chordal reducible

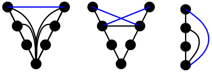
embedded tree

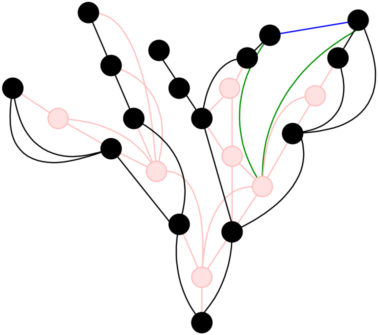


chordal = skeletal embedded tree

chordal reducible

embedded tree

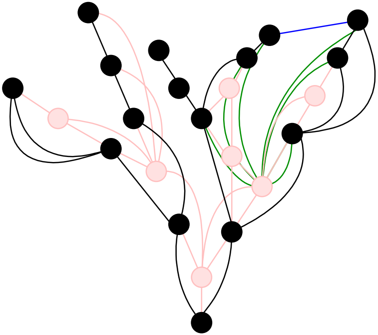


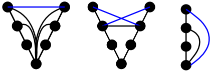


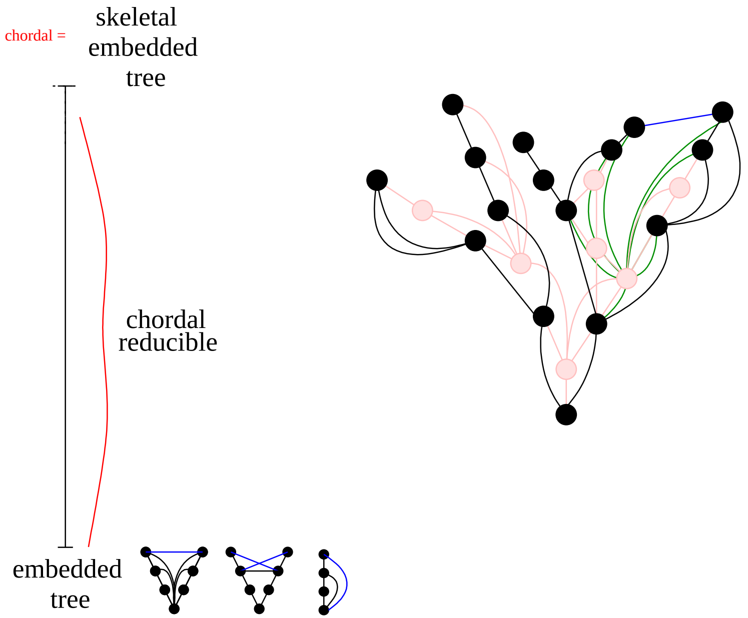
chordal = skeletal embedded tree

chordal reducible

embedded tree



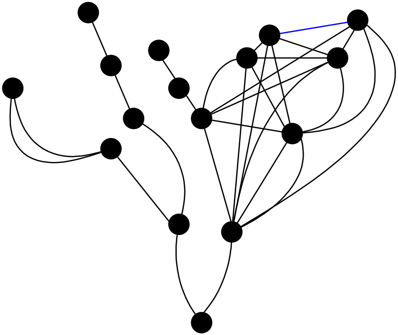


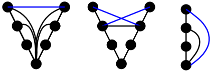

chordal =
skeletal embedded tree
chordal reducible
embedded tree

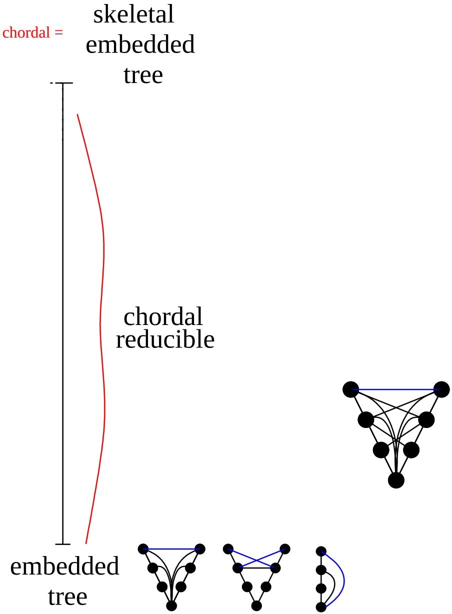
chordal = skeletal embedded tree

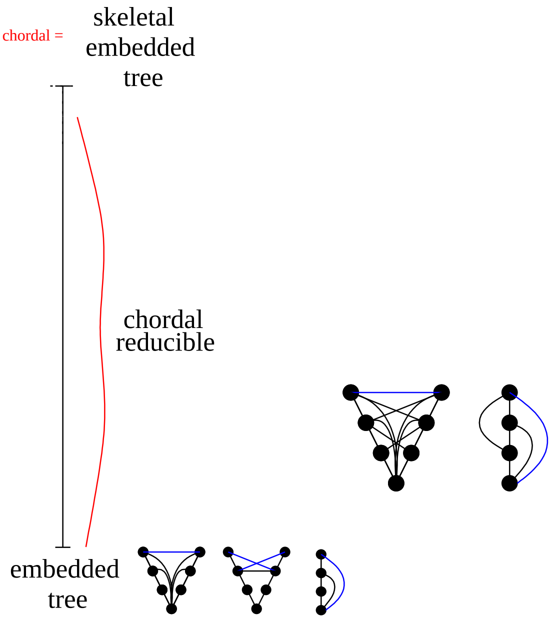
chordal reducible

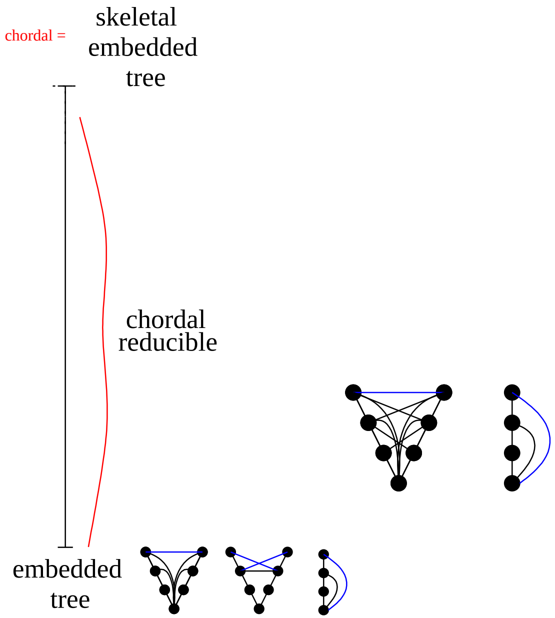
embedded tree

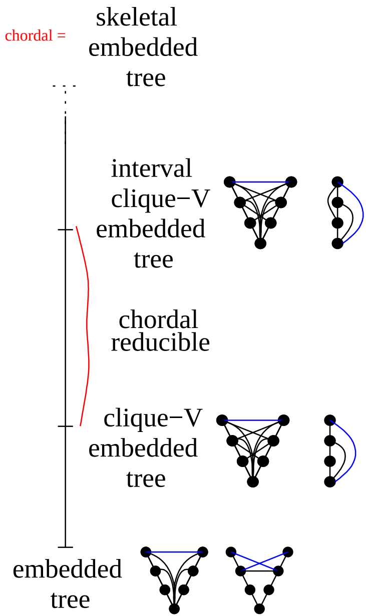




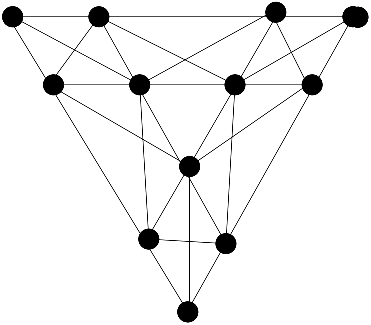

chordal =
skeletal embedded tree
chordal reducible
embedded tree


chordal =
skeletal embedded tree
chordal reducible
embedded tree


chordal =
skeletal embedded tree
chordal reducible
embedded tree


chordal =
skeletal embedded tree
interval clique−V embedded tree
chordal reducible
clique−V embedded tree
embedded tree

A Consequence



This graph has a 4-NU but no clique-V embedded tree polymorphism, so is not chordal reducible.

What we did

- Defined heirarchy of graph classes, generalising 'chordal' according to the type of SL polymorphism admitted.
- SL $\neq$ NU.
- 4-NU $\not\Rightarrow$ Chordal Reducible

Questions

- Does admitting a clique-V SL imply a graph is Chordal Reducible?
- Does 'SL' imply 'embedded SL'?
- Is there a poly-time algorithm for recognising graphs admitting
  - SL
  - clique-V SL
- Find a class of obstructions to SL that aren't obstructions to TSI.

Proof that NU $\neq$ SL

1. For a reflexive graph H let U_H be the structure defined
   - $V(U_H) = \mathrm{Powerset}(V(H))$
   - $(S, T) \in E(U_H)$ if for each $s \in S$ there is $t \in T$ with $(s, t) \in E(H)$, and vice versa.

   H has a *TSI* if and only if U_H retracts to copy of H induce by singleton vertices.
2. *NU* is preserved by retraction (NU is a variety).
3. U_H is in SL for any H: the semilattice $T < S$ if $S \subset T$ is polymorphic.

If H has a NU poly, then $H \in NU \setminus SL$ and we are done. Otherwise H has no NU poly. Since H has *TSI*, U_H retracts to H by (1), and so by (2) U_H has no NU poly. Thus $U_H \in SL \setminus NU$.