The Fixed Vertex Property, Products and Binary Constraint Satisfaction

Bernd.Schroeder@usm.edu

Fixed Point Properties

Fixed Point Properties

Definition.

A structure
has the fixed point property iff every
structure-preserving map
f *has a fixed point* $x = f(x)$.

Fixed Point Properties

Definition.

A topological space has the fixed point property iff every continuous function from the space to itself f has a fixed point $x = f(x)$.

Fixed Point Properties

Definition.

A topological space has the fixed point property iff every continuous function from the space to itself f has a fixed point $x = f(x)$.

Example.

Fixed Point Properties

Definition.

A topological space has the fixed point property iff every continuous function from the space to itself f has a fixed point $x = f(x)$.

Example. The unit ball in $\mathbb{R}^d$ has the fixed point property.

Fixed Point Properties

Definition.

A topological space has the fixed point property iff every continuous function from the space to itself f has a fixed point $x = f(x)$.

Example. The unit ball in $\mathbb{R}^d$ has the fixed point property. (Brouwer's Fixed Point Theorem.)

Fixed Point Properties

Definition. *A structure has the fixed point property iff every structure-preserving map f has a fixed point $x = f(x)$.*

Fixed Point Properties

Definition.

An ordered set has the fixed point property iff every order-preserving self map f has a fixed point $x = f(x)$.

Fixed Point Properties

Definition.

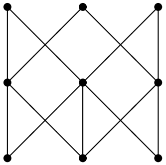
An ordered set has the fixed point property iff every order-preserving self map f has a fixed point $x = f(x)$.

Example.

Fixed Point Properties

Definition. *An ordered set has the fixed point property iff every order-preserving self map* f *has a fixed point* $x = f(x)$.

Example.



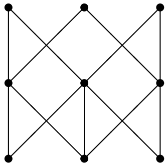
has the fixed point property.

Fixed Point Properties

Definition.

An ordered set has the fixed point property iff every order-preserving self map f has a fixed point $x = f(x)$.

Example.



has the fixed point property.

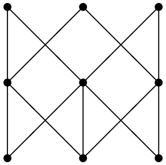
(This is a separate talk

Fixed Point Properties

Definition.

An ordered set has the fixed point property iff every order-preserving self map f has a fixed point $x = f(x)$.

Example.



has the fixed point property.

(This is a separate talk ... as is the connection to analysis.)

Fixed Point Properties

Definition.

A structure has the fixed point property iff every structure-preserving map f has a fixed point $x = f(x)$.

Fixed Point Properties

Definition.

A graph has the **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Fixed Point Properties

Definition.

A graph has the **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Examples.

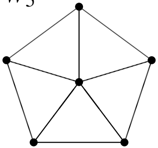
Fixed Point Properties

Definition.

A graph has the **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Examples.

W_5



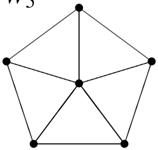
Fixed Point Properties

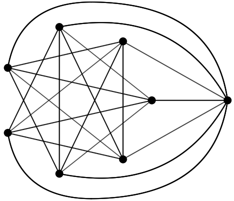
Definition.

A graph has the **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Examples.

W_5





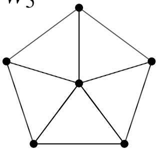
Fixed Point Properties

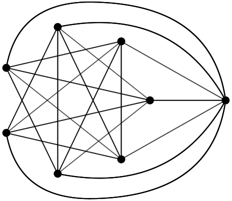
Definition.

A graph has the **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Examples.

W_5





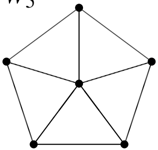
have the fixed vertex property.

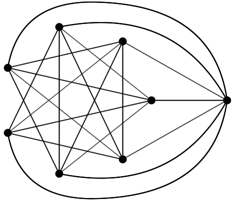
Fixed Point Properties

Definition. *A graph has the* **fixed vertex property** *iff every homomorphism from the graph to itself (every endomorphism) f has a fixed point $x = f(x)$.*

Examples.

W_5





have the fixed vertex property. But why do they?

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges.

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v, w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

A **graph** $G=(V,E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v,w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

Definition.

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v, w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

Definition. *(See, for example,* P. Hell and J. Nešetril (2004), Graphs and homomorphisms, Oxford University Press (Oxford Lecture Series in Mathematics and its Applications 28), Oxford.*)*

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v, w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

Definition. *(See, for example,* P. Hell and J. Nešetril (2004), Graphs and homomorphisms, Oxford University Press (Oxford Lecture Series in Mathematics and its Applications 28), Oxford.*) Let $G = (V, E)$ and $H = (W, F)$ be graphs.*

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v, w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

Definition. *(See, for example,* P. Hell and J. Nešetril (2004), Graphs and homomorphisms, Oxford University Press (Oxford Lecture Series in Mathematics and its Applications 28), Oxford.*) Let $G = (V, E)$ and $H = (W, F)$ be graphs. A function $f : V \to W$ is called a* **homomorphism** *iff, for all $v, w \in V$, if $\{v, w\} \in E$, then $\{f(v), f(w)\} \in F$.*

A **graph** $G = (V, E)$ consists of a set V of points, which are called vertices, and a set E of two-element subsets of E, called edges. The standard visualization is what we had on the previous panel: Points are drawn in the plane and an edge $\{v, w\}$ is indicated by a line or an arc from (the point corresponding to) v to (the point corresponding to) w.

Definition. *(See, for example,* P. Hell and J. Nešetril (2004), Graphs and homomorphisms, Oxford University Press (Oxford Lecture Series in Mathematics and its Applications 28), Oxford.*) Let $G = (V, E)$ and $H = (W, F)$ be graphs. A function $f : V \to W$ is called a* **homomorphism** *iff, for all $v, w \in V$, if $\{v, w\} \in E$, then $\{f(v), f(w)\} \in F$. An* **endomorphism** *is a homomorphism from G to G, an* **isomorphism** *is a bijective homomorphism whose inverse is a homomorphism, too, and an* **automorphism** *is an isomorphism from G to G.*

To Loop or not to Loop ... There is no Question

To Loop or not to Loop ... There is no Question

If $G=(V,E)$ has a vertex b with a loop at b, that is, an edge $\{b\}$, and another edge $\{b,c\}$

To Loop or not to Loop ... There is no Question

If $G=(V,E)$ has a vertex b with a loop at b, that is, an edge $\{b\}$, and another edge $\{b,c\}$, then the function that maps $V\setminus\{b\}$ to b and b to c is an endomorphism that does not fix a single vertex.

To Loop or not to Loop ... There is no Question

If $G = (V, E)$ has a vertex b with a loop at b, that is, an edge $\{b\}$, and another edge $\{b, c\}$, then the function that maps $V \setminus \{b\}$ to b and b to c is an endomorphism that does not fix a single vertex.

(So that's also why we defined all edges as two-element subsets.)

Without Loops, Homomorphisms do not Collapse Edges

Without Loops, Homomorphisms do not Collapse Edges

Observation.

Without Loops, Homomorphisms do not Collapse Edges

Observation. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be graphs (from now on automatically assumed to be without loops)*

Without Loops, Homomorphisms do not Collapse Edges

Observation. *Let $G = (V, E)$ and $H = (W, F)$ be graphs (from now on automatically assumed to be without loops) and let $f : V \to W$ be a homomorphism.*

Without Loops, Homomorphisms do not Collapse Edges

Observation. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be graphs (from now on automatically assumed to be without loops) and let* $f : V \to W$ *be a homomorphism. If* $v \sim w$*, then* $f(v) \neq f(w)$.

Without Loops, Homomorphisms do not Collapse Edges

Observation. *Let* $G=(V,E)$ *and* $H=(W,F)$ *be graphs (from now on automatically assumed to be without loops) and let* $f:V\to W$ *be a homomorphism. If* $v\sim w$*, then* $f(v)\neq f(w)$*.*

This means that graph colorings are homomorphisms into complete graphs, which is why the term “generalized coloring” is sometimes associated with the study of graph homomorphisms.

Speaking of Coloring

Speaking of Coloring

Definition.

Speaking of Coloring

Definition. *Let $G = (V, E)$ be a graph.*

Speaking of Coloring

Definition. *Let $G = (V, E)$ be a graph. A function $c : V \to \{1, \ldots, n\}$ such that $v \sim w$ implies $f(v) \neq f(w)$ is called an n-coloring of G.*

Speaking of Coloring

Definition. *Let $G = (V, E)$ be a graph. A function $c : V \to \{1, \ldots, n\}$ such that $v \sim w$ implies $f(v) \neq f(w)$ is called an n-coloring of G.*

Definition.

Speaking of Coloring

Definition. *Let $G = (V, E)$ be a graph. A function $c : V \to \{1, \ldots, n\}$ such that $v \sim w$ implies $f(v) \neq f(w)$ is called an n-coloring of G.*

Definition. *Let $G = (V, E)$ be a graph.*

Speaking of Coloring

Definition. *Let $G=(V,E)$ be a graph. A function $c:V\to\{1,\ldots,n\}$ such that $v\sim w$ implies $f(v)\neq f(w)$ is called an n-coloring of G.*

Definition. *Let $G=(V,E)$ be a graph. The smallest n such that there is an n-coloring of G is the* **chromatic number** $\chi(G)$.

Speaking of Coloring

Definition. *Let $G=(V,E)$ be a graph. A function $c:V\to\{1,\ldots,n\}$ such that $v\sim w$ implies $f(v)\neq f(w)$ is called an n-coloring of G.*

Definition. *Let $G=(V,E)$ be a graph. The smallest n such that there is an n-coloring of G is the* **chromatic number** $\chi(G)$.

Proposition.

Speaking of Coloring

Definition. *Let $G=(V,E)$ be a graph. A function $c:V\to\{1,\ldots,n\}$ such that $v\sim w$ implies $f(v)\neq f(w)$ is called an n-coloring of G.*

Definition. *Let $G=(V,E)$ be a graph. The smallest n such that there is an n-coloring of G is the* **chromatic number** *$\chi(G)$.*

Proposition. *Let $G=(V,E)$ and $H=(W,F)$ be graphs and let $f:V\to W$ be a homomorphism.*

Speaking of Coloring

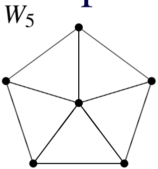
Definition. *Let $G = (V,E)$ be a graph. A function $c : V \to \{1,\ldots,n\}$ such that $v \sim w$ implies $f(v) \neq f(w)$ is called an n-coloring of G.*

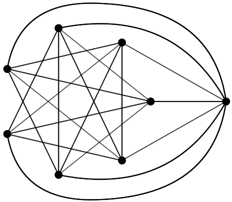
Definition. *Let $G = (V,E)$ be a graph. The smallest n such that there is an n-coloring of G is the* **chromatic number** $\chi(G)$.

Proposition. *Let $G = (V,E)$ and $H = (W,F)$ be graphs and let $f : V \to W$ be a homomorphism. Then $\chi(G) \leq \chi(H[f[V]])$.*

Back to our Examples

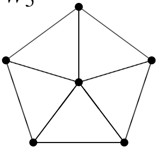
Back to our Examples

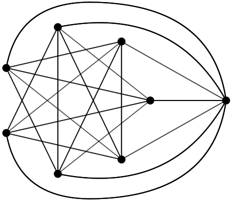




Back to our Examples

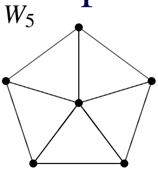
W_5

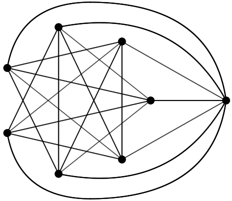




For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left)

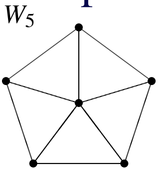
Back to our Examples

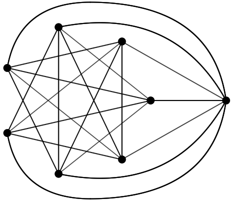




For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left) or 4 (right).

Back to our Examples

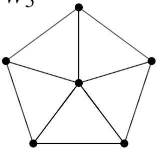


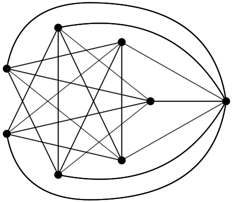


For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left) or 4 (right). The chromatic numbers of the neighborhoods of all other vertices are smaller.

Back to our Examples

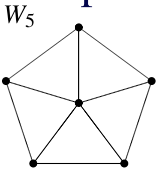
W_5

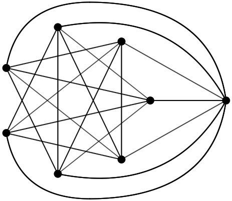




For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left) or 4 (right). The chromatic numbers of the neighborhoods of all other vertices are smaller. Thus the dominating vertices must be mapped to themselves by any endomorphisms.

Back to our Examples

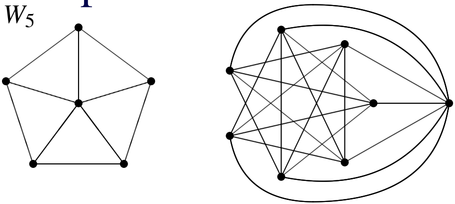




For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left) or 4 (right). The chromatic numbers of the neighborhoods of all other vertices are smaller. Thus the dominating vertices must be mapped to themselves by any endomorphisms.

A dominating vertex does not guarantee the fixed vertex property

Back to our Examples



For the graphs above, the dominating vertex is the only vertex whose neighborhood has chromatic number 3 (left) or 4 (right). The chromatic numbers of the neighborhoods of all other vertices are smaller. Thus the dominating vertices must be mapped to themselves by any endomorphisms.

A dominating vertex does not guarantee the fixed vertex property: Attaching a dominating vertex to the two graphs above produces graphs without the fixed vertex property.

Fixed Point Property for Ordered Sets

Fixed Point Property for Ordered Sets

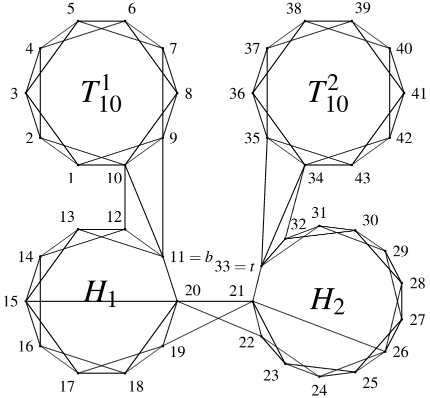
The whole fixed point theory for ordered sets can be embedded into the theory for the fixed vertex property.

Fixed Point Property for Ordered Sets

The whole fixed point theory for ordered sets can be embedded into the theory for the fixed vertex property. “Simply” replace every directed edge in the ordered set (including loops) with

Fixed Point Property for Ordered Sets

The whole fixed point theory for ordered sets can be embedded into the theory for the fixed vertex property. "Simply" replace every directed edge in the ordered set (including loops) with



A Semi-Solved Problem

A Semi-Solved Problem

The product of any two finite ordered sets with the fixed point property has the fixed point property, too. (Roddy, 1994)

Bernd.Schroeder@usm.edu Department of Mathematics, The University of Southern Mississippi

A Semi-Solved Problem

The product of any two finite ordered sets with the fixed point property has the fixed point property, too. (Roddy, 1994)

What about infinite ordered sets?

A Semi-Solved Problem

The product of any two finite ordered sets with the fixed point property has the fixed point property, too. (Roddy, 1994)

What about infinite ordered sets?

Certain tools for finite ordered sets are not available for general infinite ordered sets.

A Semi-Solved Problem

The product of any two finite ordered sets with the fixed point property has the fixed point property, too. (Roddy, 1994)

What about infinite ordered sets?

Certain tools for finite ordered sets are not available for general infinite ordered sets. They are also not available for the fixed vertex property for graphs.

A Semi-Solved Problem

The product of any two finite ordered sets with the fixed point property has the fixed point property, too. (Roddy, 1994)

What about infinite ordered sets?

Certain tools for finite ordered sets are not available for general infinite ordered sets. They are also not available for the fixed vertex property for graphs. So, with the fixed point property (including products of ordered sets with the fixed point property) embedding into the fixed vertex property, maybe an analysis of the fixed vertex property for products of graphs could provide new insights?

Does the Product of Two Graphs with the Fixed Vertex Property have the Fixed Vertex Property?

Does the Product of Two Graphs with the Fixed Vertex Property have the Fixed Vertex Property?

Which product?

Does the Product of Two Graphs with the Fixed Vertex Property have the Fixed Vertex Property?

Which product? There are at least 4 products for graphs.

Does the Product of Two Graphs with the Fixed Vertex Property have the Fixed Vertex Property?

Which product? There are at least 4 products for graphs.

One of them corresponds to the one for ordered sets (with the embedding from the previous panel) and maybe the other ones are interesting, too.

Definition.

Definition. *The* **direct product** *or* **categorical product** *or* **tensor product** *of G and H is the graph $G \times H$ whose vertices are the set $V \times W$ and for which there is an edge between (x,u) and (y,v) iff $\{x,y\} \in E$ and $\{u,v\} \in F$.*

Definition. *The* **direct product** *or* **categorical product** *or* **tensor product** *of G and H is the graph $G \times H$ whose vertices are the set $V \times W$ and for which there is an edge between (x,u) and (y,v) iff $\{x,y\} \in E$ and $\{u,v\} \in F$.*

This is the product that acts like the product for ordered sets on the mentioned embeddings (technical proof).

Definition. *The* **direct product** *or* **categorical product** *or* **tensor product** *of G and H is the graph $G \times H$ whose vertices are the set $V \times W$ and for which there is an edge between (x,u) and (y,v) iff $\{x,y\} \in E$ and $\{u,v\} \in F$.*

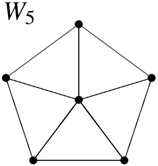
This is the product that acts like the product for ordered sets on the mentioned embeddings (technical proof).

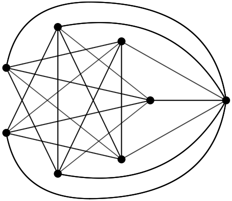
The shock (for me). The direct product of the two graphs below does not have the fixed vertex property.

Definition. *The* **direct product** *or* **categorical product** *or* **tensor product** *of G and H is the graph $G \times H$ whose vertices are the set $V \times W$ and for which there is an edge between (x,u) and (y,v) iff $\{x,y\} \in E$ and $\{u,v\} \in F$.*

This is the product that acts like the product for ordered sets on the mentioned embeddings (technical proof).

The shock (for me). The direct product of the two graphs below does not have the fixed vertex property.





How Do We Know?

How Do We Know?

Definition.

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$,

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$*, (take the vertices of the graph)*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$*, (take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$*, one for each variable,*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$*, (take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$*, one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
   - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
   - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$ *(already incorporated in the choice of the* D_i*),*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
   - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$ *(already incorporated in the choice of the* D_i*),*
   - *Each binary constraint consists of a set of two variables* $\{x_i, x_j\}$ *and a binary relation* $C_{ij} \subseteq D_i \times D_j$

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1,\cdots,x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1,\cdots,D_r$, *one for each variable, (make* $D_i := V\setminus\{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
    - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$ *(already incorporated in the choice of the* D_i*),*
    - *Each binary constraint consists of a set of two variables* $\{x_i, x_j\}$ *and a binary relation* $C_{ij} \subseteq D_i \times D_j$ *(*$x_i \not\sim x_j$*: no constraint*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
   - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$ *(already incorporated in the choice of the* D_i*),*
   - *Each binary constraint consists of a set of two variables* $\{x_i, x_j\}$ *and a binary relation* $C_{ij} \subseteq D_i \times D_j$ *(*$x_i \not\sim x_j$*: no constraint;* $x_i \sim x_j$*:* $C_{ij} = \{(f_i, f_j) \in D_i \times D_j : f_i \sim f_j\}$*),*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables $x_1,\cdots,x_r$, (take the vertices of the graph)*
2. *A set of domains $D_1,\cdots,D_r$, one for each variable, (make $D_i := V\setminus\{x_i\}$)*
3. *A set $\mathscr{C}$ of unary and binary constraints.*
    - *Each unary constraint consists of a variable x_i and a set $C_i \subseteq D_i$ (already incorporated in the choice of the D_i),*
    - *Each binary constraint consists of a set of two variables $\{x_i,x_j\}$ and a binary relation $C_{ij} \subseteq D_i \times D_j$ ($x_i \not\sim x_j$: no constraint; $x_i \sim x_j$: $C_{ij} = \{(f_i,f_j) \in D_i \times D_j : f_i \sim f_j\}$),*
    - *For each set of variables, we have at most one constraint.*

How Do We Know?

Definition. *A* **binary constraint network** *consists of the following.*

1. *A set of variables* $x_1, \cdots, x_r$, *(take the vertices of the graph)*
2. *A set of domains* $D_1, \cdots, D_r$, *one for each variable, (make* $D_i := V \setminus \{x_i\}$*)*
3. *A set* $\mathscr{C}$ *of unary and binary constraints.*
   - *Each unary constraint consists of a variable* x_i *and a set* $C_i \subseteq D_i$ *(already incorporated in the choice of the* D_i*),*
   - *Each binary constraint consists of a set of two variables* $\{x_i, x_j\}$ *and a binary relation* $C_{ij} \subseteq D_i \times D_j$ *(*$x_i \not\sim x_j$*: no constraint;* $x_i \sim x_j$*:* $C_{ij} = \{(f_i, f_j) \in D_i \times D_j : f_i \sim f_j\}$*),*
   - *For each set of variables, we have at most one constraint.*

A **solution** is a set of instantiations $\{(x_1, f_1), \ldots, (x_r, f_r)\}$ such that any subset $\{(x_j, f_j), (x_k, f_k)\}$ is such that $(f_j, f_k) \in C_{jk}$.

(My) Implementation

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices.

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices.
  (All pairs (x_j, f_j) with $x_j \neq f_j$.)

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed (“consistent”).

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)
- Solutions correspond to r-cliques.

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)
- Solutions correspond to r-cliques.

Enforcing (Path) Consistency

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)
- Solutions correspond to r-cliques.

Enforcing (Path) Consistency

- Remove any remaining $\{(x_j, f_j), (x_k, f_k)\}$ such that there is an $x_i \notin \{x_j, x_k\}$ for which there is no triangle of the form $\{(x_j, f_j), (x_k, f_k), (x_i, y)\}$

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)
- Solutions correspond to r-cliques.

Enforcing (Path) Consistency

- Remove any remaining $\{(x_j, f_j), (x_k, f_k)\}$ such that there is an $x_i \notin \{x_j, x_k\}$ for which there is no triangle of the form $\{(x_j, f_j), (x_k, f_k), (x_i, y)\}$ until this is no longer possible.

(My) Implementation

Expanded Binary Constraint Network. (Interpret as a graph.)

- Store all allowed instantiations (x_j, f_j) as vertices. (All pairs (x_j, f_j) with $x_j \neq f_j$.)
- The pair $\{(x_j, f_j), (x_k, f_k)\}$ is an edge iff the assignment of f_j to x_j and of f_i to x_i is allowed ("consistent"). (This is all pairs $\{(x_j, f_j), (x_k, f_k)\}$ with $x_j \neq f_j$, $x_k \neq f_k$, and such that $x_i \sim x_j \Rightarrow f_i \sim f_j$.)
- Solutions correspond to r-cliques.

Enforcing (Path) Consistency

- Remove any remaining $\{(x_j, f_j), (x_k, f_k)\}$ such that there is an $x_i \notin \{x_j, x_k\}$ for which there is no triangle of the form $\{(x_j, f_j), (x_k, f_k), (x_i, y)\}$ until this is no longer possible.
- There are many more consistency enforcing mechanisms.

(My) Implementation

(My) Implementation

Search

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i > k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$.

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i>k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i>k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)
- There are many more search algorithms.

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i > k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)
- There are many more search algorithms.

(This is another separate talk.)

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i > k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)
- There are many more search algorithms.

(This is another separate talk.)

Once the front end for direct products was written, the algorithm worked embarrassingly fast

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i>k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)
- There are many more search algorithms.

(This is another separate talk.)

Once the front end for direct products was written, the algorithm worked embarrassingly fast ... even faster if we enforce $(2,2)$-consistency

(My) Implementation

Search

- Extend any clique $\{(x_1,f_1),\ldots,(x_k,f_k)\}$ of mutually consistent instantiations until there is an $i>k$ such that there is no clique of the form $\{(x_1,f_1),\ldots,(x_k,f_k),(x_i,y)\}$. (Forward checking.)
- There are many more search algorithms.

(This is another separate talk.)

Once the front end for direct products was written, the algorithm worked embarrassingly fast ... even faster if we enforce $(2,2)$-consistency, that is, delete all edges $\{(x_j,f_j),(x_k,f_k)\}$ of the network such that there are 2 more variables $x_i,x_\ell\notin\{x_j,x_k\}$ such that the edge is not part of a 4-clique $\{(x_j,f_j),(x_k,f_k),(x_i,y),(x_\ell,v)\}$.

Back to Products

Back to Products

Definition.

Back to Products

Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs.*

Back to Products

Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs. The* **lexicographic product** $G \bullet H$ *of* G *and* H *is the graph obtained from* $|V|$ *isomorphic copies* H_v $(v \in V)$ *of* H

Back to Products

Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs. The* **lexicographic product** $G \bullet H$ *of* G *and* H *is the graph obtained from* $|V|$ *isomorphic copies* H_v $(v \in V)$ *of* H *such that, if* $v \sim w$*, then all vertices in* H_v *are adjacent to all vertices in* H_w

Back to Products

Definition. *Let* $G = (V,E)$ *and* $H = (W,F)$ *be two graphs. The* **lexicographic product** $G \bullet H$ *of* G *and* H *is the graph obtained from* $|V|$ *isomorphic copies* H_v $(v \in V)$ *of* H *such that, if* $v \sim w$*, then all vertices in* H_v *are adjacent to all vertices in* H_w*, and*

Back to Products

Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs. The* **lexicographic product** $G \bullet H$ *of* G *and* H *is the graph obtained from* $|V|$ *isomorphic copies* H_v $(v \in V)$ *of* H *such that, if* $v \sim w$*, then all vertices in* H_v *are adjacent to all vertices in* H_w*, and, if* $v \not\sim w$*, then no vertex in* H_v *is adjacent to any vertex in* H_w*.*

Back to Products

Definition. *Let $G = (V,E)$ and $H = (W,F)$ be two graphs. The* **lexicographic product** $G \bullet H$ *of G and H is the graph obtained from $|V|$ isomorphic copies H_v $(v \in V)$ of H such that, if $v \sim w$, then all vertices in H_v are adjacent to all vertices in H_w, and, if $v \not\sim w$, then no vertex in H_v is adjacent to any vertex in H_w.*

Lexicographic products of graphs with the fixed vertex property should have the fixed vertex property

Back to Products

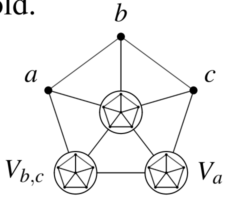
Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **lexicographic product** $G \bullet H$ *of G and H is the graph obtained from $|V|$ isomorphic copies H_v $(v \in V)$ of H such that, if $v \sim w$, then all vertices in H_v are adjacent to all vertices in H_w, and, if $v \not\sim w$, then no vertex in H_v is adjacent to any vertex in H_w.*

Lexicographic products of graphs with the fixed vertex property should have the fixed vertex property ... but for lexicographic *sums*, this does not hold.

Back to Products

Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **lexicographic product** *$G \bullet H$ of G and H is the graph obtained from $|V|$ isomorphic copies H_v $(v \in V)$ of H such that, if $v \sim w$, then all vertices in H_v are adjacent to all vertices in H_w, and, if $v \not\sim w$, then no vertex in H_v is adjacent to any vertex in H_w.*

Lexicographic products of graphs with the fixed vertex property should have the fixed vertex property ... but for lexicographic *sums*, this does not hold.



Back to Products

Back to Products

Definition.

Back to Products

Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs.*

Back to Products

Definition. *Let $G=(V,E)$ and $H=(W,F)$ be two graphs. The* **strong product** *of G and H is the graph $G\boxtimes H$ whose vertices are the set $V\times W$ and for which there is an edge between (x,u) and (y,v) iff ($\{x,y\}\in E$ or $x=y$) and ($\{u,v\}\in F$ or $u=v$).*

Back to Products

Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **strong product** *of G and H is the graph $G \boxtimes H$ whose vertices are the set $V \times W$ and for which there is an edge between (x, u) and (y, v) iff ($\{x, y\} \in E$ or $x = y$) and ($\{u, v\} \in F$ or $u = v$).*

I have little intuition beyond the fact that an endomorphism of the form $f(x, y) = (g(x), h(y))$ must have a fixed point and that many endomorphisms must be of this form.

Back to Products

Definition. *Let $G=(V,E)$ and $H=(W,F)$ be two graphs. The* **strong product** *of G and H is the graph $G\boxtimes H$ whose vertices are the set $V\times W$ and for which there is an edge between (x,u) and (y,v) iff ($\{x,y\}\in E$ or $x=y$) and ($\{u,v\}\in F$ or $u=v$).*

I have little intuition beyond the fact that an endomorphism of the form $f(x,y)=(g(x),h(y))$ must have a fixed point and that many endomorphisms must be of this form. (See below for cartesian products.)

Back to Products

Definition. *Let $G = (V,E)$ and $H = (W,F)$ be two graphs. The* **strong product** *of G and H is the graph $G \boxtimes H$ whose vertices are the set $V \times W$ and for which there is an edge between (x,u) and (y,v) iff ($\{x,y\} \in E$ or $x = y$) and ($\{u,v\} \in F$ or $u = v$).*

I have little intuition beyond the fact that an endomorphism of the form $f(x,y) = (g(x),h(y))$ must have a fixed point and that many endomorphisms must be of this form. (See below for cartesian products.)

I think the strong product operation also corresponds to the product operations for the embeddings of ordered sets.

Back to Products

Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **strong product** *of G and H is the graph $G \boxtimes H$ whose vertices are the set $V \times W$ and for which there is an edge between (x, u) and (y, v) iff ($\{x, y\} \in E$ or $x = y$) and ($\{u, v\} \in F$ or $u = v$).*

I have little intuition beyond the fact that an endomorphism of the form $f(x, y) = (g(x), h(y))$ must have a fixed point and that many endomorphisms must be of this form. (See below for cartesian products.)

I think the strong product operation also corresponds to the product operations for the embeddings of ordered sets. (Should be a similar proof as for the direct product, but more technical.)

Back to Products

Back to Products

Definition.

Back to Products

Definition. *Let $G=(V,E)$ and $H=(W,F)$ be two graphs.*

Back to Products

Definition. *Let $G=(V,E)$ and $H=(W,F)$ be two graphs. The* **cartesian product** *of G and H is the graph $G\Box H$ whose vertices are the set $V\times W$ and for which there is an edge between (x,u) and (y,v) iff $\{x,y\}\in E$ and $u=v$ or $x=y$ and $\{u,v\}\in F$.*

Back to Products

Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **cartesian product** *of G and H is the graph $G \square H$ whose vertices are the set $V \times W$ and for which there is an edge between (x, u) and (y, v) iff $\{x, y\} \in E$ and $u = v$ or $x = y$ and $\{u, v\} \in F$.*

Lemma.

Back to Products

Definition. *Let $G = (V, E)$ and $H = (W, F)$ be two graphs. The* **cartesian product** *of G and H is the graph $G \square H$ whose vertices are the set $V \times W$ and for which there is an edge between (x, u) and (y, v) iff $\{x, y\} \in E$ and $u = v$ or $x = y$ and $\{u, v\} \in F$.*

Lemma. *Any triangle $(x_1, y_1) \sim (x_2, y_2) \sim (x_3, y_3) \sim (x_1, y_1)$ in $G \square H$ satisfies either $x_1 = x_2 = x_3$ or $y_1 = y_2 = y_3$.*

Back to Products

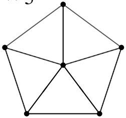
Definition. *Let* $G = (V, E)$ *and* $H = (W, F)$ *be two graphs. The* **cartesian product** *of* G *and* H *is the graph* $G \square H$ *whose vertices are the set* $V \times W$ *and for which there is an edge between* (x, u) *and* (y, v) *iff* $\{x, y\} \in E$ *and* $u = v$ *or* $x = y$ *and* $\{u, v\} \in F$.

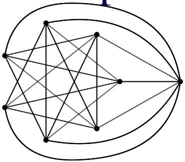
Lemma. *Any triangle* $(x_1, y_1) \sim (x_2, y_2) \sim (x_3, y_3) \sim (x_1, y_1)$ *in* $G \square H$ *satisfies either* $x_1 = x_2 = x_3$ *or* $y_1 = y_2 = y_3$. *Therefore, if* G *and* H *have the fixed vertex property and any two vertices of* G *are connected by a path such that every edge is contained in a triangle, then* $G \square H$ *has the fixed vertex property.*

... so there is no counterexample here

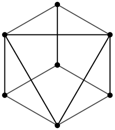
... so there is no counterexample here

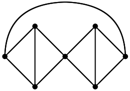
W_5

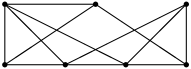




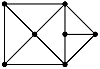
Graph 649 Graph 738 Graph 769

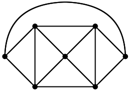


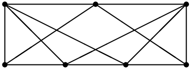




Graph 766 Graph 825 Graph 794

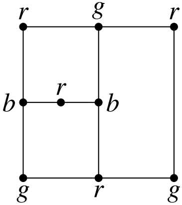


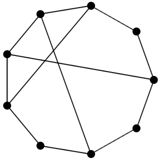


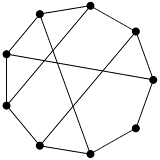


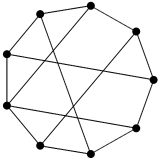
... so far, there also is no counterexample here

... so far, there also is no counterexample here

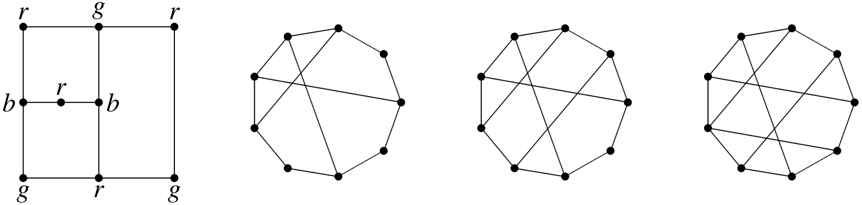






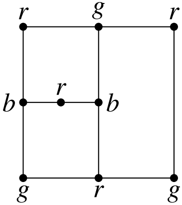


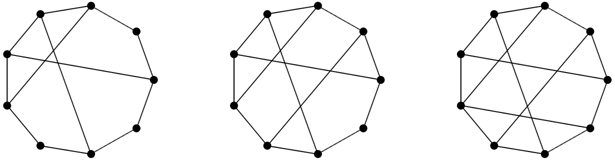
... so far, there also is no counterexample here



... and it naturally brings up the following question

... so far, there also is no counterexample here





... and it naturally brings up the following question: What kinds of graphs have the fixed vertex property and no triangles?

A Universal Tool for Fixed Points

A Universal Tool for Fixed Points

Definition.

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map

$r : P \to P$ is called a **retraction** *iff $r^2 = r$*

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map

$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***).*

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map
$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P*

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map
$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P iff there is a retraction $r : P \to P$ with $r[P] = R$.*

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map
$r : P \to P$ *is called a* **retraction** *iff* $r^2 = r$ *(that is, iff* r *is* **idempotent***). We will say that* $R \subseteq P$ *is a* **retract** *of* P *iff there is a retraction* $r : P \to P$ *with* $r[P] = R$.

Theorem.

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map

$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P iff there is a retraction $r : P \to P$ with $r[P] = R$.*

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction.

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map

$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P iff there is a retraction $r : P \to P$ with $r[P] = R$.*

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

A Universal Tool for Fixed Points

A Universal Tool for Fixed Points

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

A Universal Tool for Fixed Points

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

Proof.

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving.

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$.

A Universal Tool for Fixed Points

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

Proof. Let $f : r[P] \to r[P]$ be

structure-preserving.

Then $f \circ r : P \to P$ is

structure-preserving,

too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]]$

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$

A Universal Tool for Fixed Points

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

Proof. Let $f : r[P] \to r[P]$ be

structure-preserving.

Then $f \circ r : P \to P$ is

structure-preserving,

too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$, which means that x

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$, which means that $x = f(r(x))$

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$, which means that $x = f(r(x)) = f(x)$

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$, which means that $x = f(r(x)) = f(x)$ is a fixed point of f.

A Universal Tool for Fixed Points

Theorem. *Let P be a structure with the fixed point property and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.*

Proof. Let $f : r[P] \to r[P]$ be structure-preserving. Then $f \circ r : P \to P$ is structure-preserving, too, and hence it has a fixed point $x = f(r(x))$. But then $x \in f[r[P]] \subseteq r[P]$ and hence $x = r(x)$, which means that $x = f(r(x)) = f(x)$ is a fixed point of f. ∎

A Universal Tool for Fixed Points

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map $r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P iff there is a retraction $r : P \to P$ with $r[P] = R$.*

A Universal Tool for Fixed Points

Definition. *Let P be a*

structure.

Then

a structure-preserving map

$r : P \to P$ is called a **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq P$ is a* **retract** *of P iff there is a retraction $r : P \to P$ with $r[P] = R$.*

Theorem. *Let P be*

a structure with the fixed point property

and let $r : P \to P$ be a retraction. Then $r[P]$ has the fixed point property.

A Universal Tool used for Fixed Vertices

Definition. *Let G be a graph. Then an endomorphism $r : V \to V$ is called a* **retraction** *iff $r^2 = r$ (that is, iff r is* **idempotent***). We will say that $R \subseteq V$ is a* **retract** *of P iff there is a retraction $r : V \to V$ with $r[V] = R$.*

Theorem. *Let G be a graph with the fixed vertex property and let $r : V \to V$ be a retraction. Then $G[r[V]]$ has the fixed vertex property.*

Searching for Graphs without Triangles and with the Fixed Vertex Property

Searching for Graphs without Triangles and with the Fixed Vertex Property

1. Outerplanar graphs retract onto their shortest odd cycle.

Searching for Graphs without Triangles and with the Fixed Vertex Property

1. Outerplanar graphs retract onto their shortest odd cycle.
2. Cycles with 2 chords do not have the fixed vertex property.

Searching for Graphs without Triangles and with the Fixed Vertex Property

1. Outerplanar graphs retract onto their shortest odd cycle.
2. Cycles with 2 chords do not have the fixed vertex property.
3. Cycles with at most 2 chords crossing in an “outerplanar drawing do not have the fixed vertex property.

Searching for Graphs without Triangles and with the Fixed Vertex Property

1. Outerplanar graphs retract onto their shortest odd cycle.
2. Cycles with 2 chords do not have the fixed vertex property.
3. Cycles with at most 2 chords crossing in an “outerplanar drawing do not have the fixed vertex property.
4. Cycles with ≥ 3 chords can have the fixed vertex property.

Searching for Graphs without Triangles and with the Fixed Vertex Property

1. Outerplanar graphs retract onto their shortest odd cycle.
2. Cycles with 2 chords do not have the fixed vertex property.
3. Cycles with at most 2 chords crossing in an "outerplanar drawing do not have the fixed vertex property.
4. Cycles with ≥ 3 chords can have the fixed vertex property.