FCA and an Extension to Graphs

Saullo H. G. de Oliveira

FEEC - University of Campinas - Brazil

Outline

Intro

What is FCA?

- *Formal Concept Analysis* is a field of applied mathematics based on the mathematization of *concept* and *conceptual hierarchy*. It thereby activates mathematical thinking for conceptual data analysis and knowledge processing (Ganter 1999).
- It tries to formalize our use of concepts and specialization of concepts.
- FCA is a way of hierarchically represent a relationship between two sets.

A Toy Example

Table 1: Toy example

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

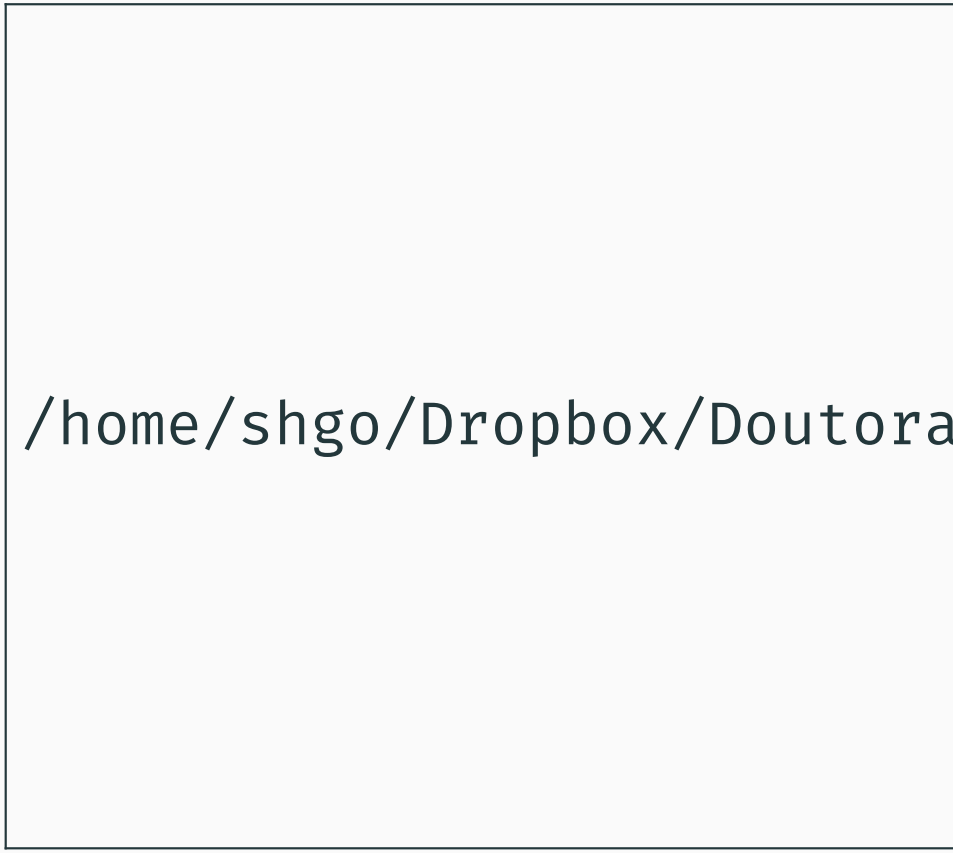


Figure 1: Toy example concept lattice.

Checklist for FCA

- ☐ Find a hierarchical order in a set of elements.
- ☐ Find relationships between two sets.
- ☐ Find a hierarchical order in a relationship between two sets.

Ordered Sets and Lattices

Binary Relation

Definition (Ganter 1999, p. 1)

A **binary relation** R between two sets M and N is a set of pairs (m, n) with $m \in M$ and $n \in N$. Instead of $(m, n) \in R$ we often write mRn.

Definition (Ganter 1999, p. 1)

A binary relation R on a set M is called an **order relation**, if for all $x, y, z \in M$:

- Reflexivity: xRx
- Antisymmetry: xRy and $x \neq y \implies$ not yRx
- Transitivity: xRy and $yRz \implies xRz$

Ordered Sets

Definition (Ganter 1999, p. 1)

An **ordered set** is a pair $(M, \leq)$, with M being a set and $\leq$ an order relation on M.

Examples:

- $\mathbb{R}$ with the usual $\leq$ relation.
- the power set $\mathfrak{B}(X)$ of any set X with set inclusion ($\subseteq$) relation.

Neighbors

Definition (Ganter 1999, p. 2)

a is a **lower neighbor** of b, if $a < b$ and there is no c, such that $a < c < b$. In that case, we write $a \prec b$.

- $1 \prec 2$ in $(\mathbb{N}, \leq)$
- $\{0, 1\} \prec \{0, 1, 2\}$ in $(\mathfrak{B}(\{0, 1, 2, 3\}), \subseteq)$

Infimum and Supremum

Definition (Ganter 1999, p. 5)

Let $(M, \leq)$ be an ordered set and $A \subset M$. A **lower bound** (**upper bound**) of A is an element $s \in M$ such that $s \leq a (s \geq a), \forall a \in A$.

Definition (Ganter 1999, p. 5)

If there is a largest element in the set of all lower bounds of A, it is called the **infimum** of A, denoted by $inf\ A$ or $\bigwedge A$. Dually, a least upper bound is called **supremum**, denoted by $sup\ A$ or $\bigvee A$.

Complete Lattices

Definition (Ganter 1999, p. 5)

An ordered set $\mathbf{V} := (V, \leq)$ is a **lattice**, if for any two elements x and y in V, $x \wedge y$ and $x \vee y$ always exist. $\mathbf{V}$ is called a **complete lattice**, if $\bigwedge V$ and $\bigvee V$ exist for any subset X of V.

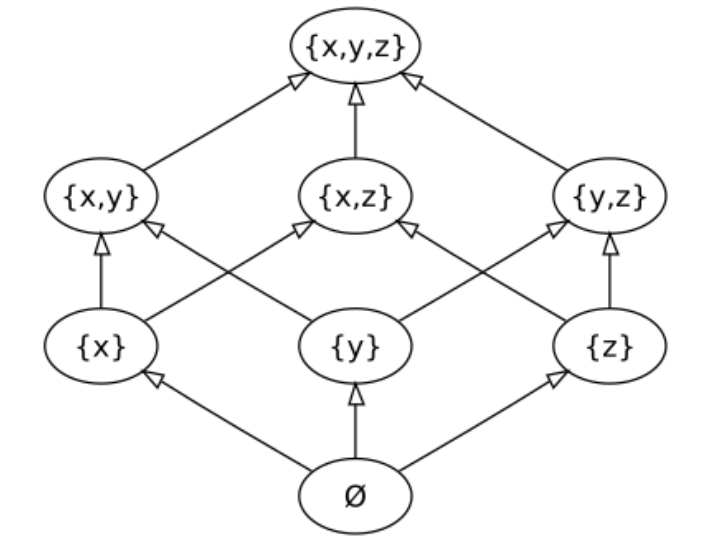


Figure 2: Powerset Lattice of the set $\{x, y, z\}$, ordered by inclusion (KSmrq 2004).

Checklist for FCA

- ☑ Find structure in a set of elements
- ☐ Find relationships between two sets
- ☐ Find a structure in a relationship between two sets

Maps Between Posets

Definition (Smith 2010, p. 6)

Suppose that $\mathbf{P} = (P, \leq)$ and $\mathbf{Q} = (Q, \leq)$ are two posets. Let $f : P \to Q$ be a map between the two sets. Then,

(i) f is monotone *iff*, for all $p, p' \in P$, if $p \leq p'$ then $f(p) \leq f(p')$

(ii) f is order-embedding *iff* for all $p, p' \in P$, $p \leq p'$ iff $f(p) \leq f(p')$

Important observations

- A map can squeeze all elements.
- f is an injective (one-one) map.
- if f is an order-embedding map, and is an onto function, f is an order-isomorphism.

A special case: Galois Connection

Definition (Smith 2010, p. 12)

Let $\mathsf{P} = (P, \leq)$ and $\mathsf{Q} = (Q, \leq)$ be posets. Suppose $f^{\uparrow} : P \to Q$ and $f^{\downarrow} : Q \to P$ are a pair of functions such that for all $p \in P, q \in Q$:

$$f^{\uparrow}(p) \leq q \iff p \leq f^{\downarrow}(q). \tag{1}$$

Then the pair $(f^{\uparrow}, f^{\downarrow})$ form a Galois connection between P and Q.

Galois Connection

Theorem (Alternative definition, Smith 2010, p. 15)

Let $\mathbf{P} = (P, \leq)$ and $\mathbf{Q} = (Q, \leq)$ be posets. Suppose $f^{\uparrow} : P \to Q$ and $f^{\downarrow} : Q \to P$ are a pair of functions between the two sets. Then $(f^{\uparrow}, f^{\downarrow})$ is a Galois connection if and only if:

(i) $f^{\uparrow}, f^{\downarrow}$ *are both monotone, and*

(ii) *for all* $p \in P, q \in Q, p \leq f^{\downarrow}(f^{\uparrow}(p))$ *and* $f^{\uparrow}(f^{\downarrow}(q)) \leq q$.

Closure Operation

Definition (Smith 2010, p. 20)

Let $\mathbf{P} = (P, \leq)$ is a poset; then a closure function for $\mathbf{P}$ is a function c such that, for all $p, p' \in P$,

(i) $p \leq c(p)$;

(ii) if $p \leq p'$, then $c(p) \leq c(p')$, i.e. c is monotone;

(iii) $c(c(p)) = c(p)$ i.e. c is 'idempotent'.

Roughly speaking, then, a closure function c maps a poset "upwards" to a subset which then stays fixed under further applications of c.

A Galois connection is a closure operation!

Checklist for FCA

- ☑ Find a hierarchical order in a set of elements
- ☑ Find relationships between two sets
- ☐ Find a hierarchical order in a relationship between two sets

Formal Concept Analysis

Formal Context

Definition (Ganter 1999, p. 17)

A **formal context** $\mathbb{K} := (G, M, I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the **objects** and the elements of M are called the **attributes** of the context. When an object $g \in G$ is in a relation I with an attribute $m \in M$, we write gIm or $(g, m) \in I$.

Example of a Formal Context

Table 2: A Toy Example

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

Closure Operation

Definition (Ganter 1999, p. 18)

For a set $A \subseteq G$ of objects we define:

$$A^{\uparrow} := \{m \in M | gIm \text{ for all } g \in A\}. \tag{2}$$

Correspondingly, for a set $B \subseteq M$:

$$B^{\downarrow} := \{g \in G | gIm \text{ for all } m \in B\} \tag{3}$$

Example of the Closure Operation

Table 3: Formal Context

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

- $\{\text{Nilson, Cássia}\}^{\uparrow} = \{\text{Tem Filhos, Trabalha}\}$
- $\{\text{Tem Filhos, Trabalha}\}^{\downarrow} = \{\text{Nilson, Cássia, Aline}\}$
- $\{\text{Nilson, Cássia, Aline}\}^{\uparrow} = \{\text{Tem Filhos, Trabalha}\}$
- $\emptyset^{\uparrow} = \{\text{Tem Filhos, Trabalha, Cozinha}\}$
- $\{\text{Tem Filhos, Trabalha, Cozinha}\}^{\downarrow} = \{\text{Nilson}\}$

Formal Concept

Definition (Ganter 1999, p. 18)

A **formal concept** of the context (G, M, I) is a pair (A, B) with $A \subseteq G, B \subseteq M, A^{\uparrow} = B,$ and $B^{\downarrow} = A$. A is the **extent** and B is the **intent** of the concept (A, B). $\mathfrak{B}(G, M, I)$ denotes the set of all concepts of the context (G, M, I).

Example of the Closure Operation

Table 4: Formal Context

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

- $\{\text{Nilson, Cássia, Aline}\}^{\uparrow} = \{\text{Tem Filhos, Trabalha}\}$
- $\{\text{Tem Filhos, Trabalha}\}^{\downarrow} = \{\text{Nilson, Cássia, Aline}\}$
- $(A = \{\text{Nilson, Cássia, Aline}\}, B = \{\text{Tem Filhos, Trabalha}\})$ is a formal concept.

Important Properties of Concept Closure Operations

(Ganter 1999, p. 18)

If (G, M, I) is a context, $A, A_1, A_2 \subseteq G$ and $B, B_1, B_2 \subseteq M$, then:

1) $A_1 \subseteq A_2 \implies A_2^{\uparrow} \subseteq A_1^{\uparrow}$
1') $B_1 \subseteq B_2 \implies B_2^{\downarrow} \subseteq B_1^{\downarrow}$
2) $A \subseteq A^{\uparrow\downarrow}$
2') $B \subseteq B^{\downarrow\uparrow}$
3) $A^{\uparrow} = A^{\uparrow\downarrow\uparrow}$
3') $B^{\downarrow} = B^{\downarrow\uparrow\downarrow}$
4) $A \subseteq B^{\downarrow} \iff B \subseteq A^{\uparrow} \iff A \times B \subseteq I$.

Concept Lattice

Definition (Ganter 1999, p. 19)

If (A_1, B_1) and (A_2, B_2) are concepts of a context, (A_1, B_1) is called a **subconcept** of its **superconcept** (A_2, B_2), provided that $A_1 \subseteq A_2$ ($B_2 \subseteq B_1$). We write $(A_1, B_1) \leq (A_2, B_2)$. $\underline{\mathfrak{B}}(G, M, I)$ is the ordered set of all concepts of (G, M, I), and is called the **concept lattice** of the context (G, M, I).

A Toy Example

Table 5: Toy example

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

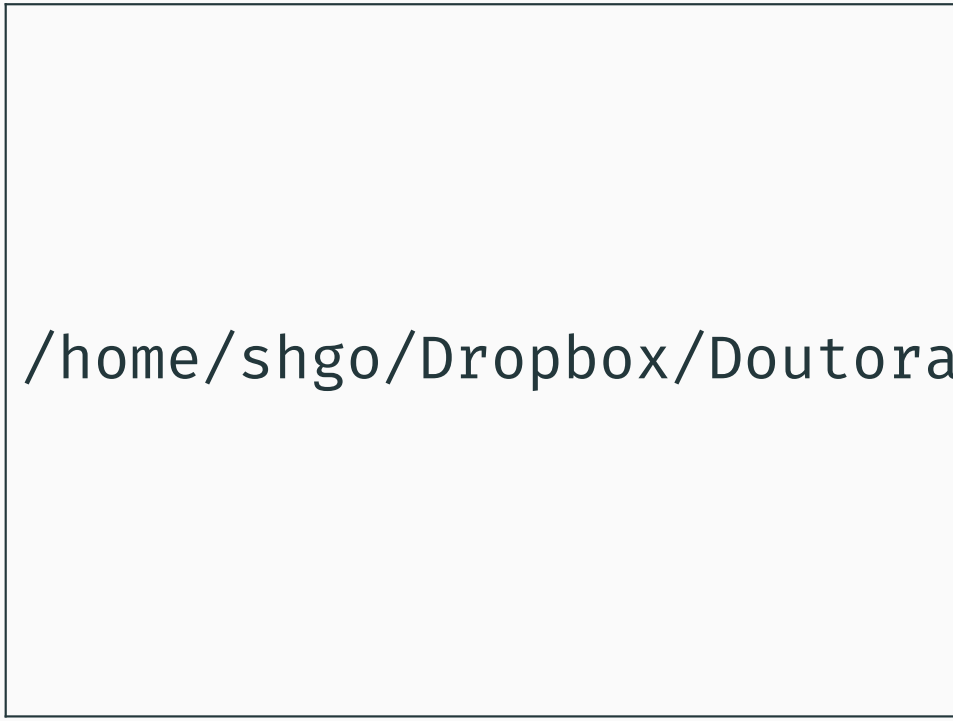


Figure 3: Toy example concept lattice.

$(\{\text{Nilson, Cássia, Aline}\}, \{\text{Tem Filhos, Trabalha, Cozinha}\}) \leq$
$(\{\text{Nilson, Cássia, Aline, Alan}\}, \{\text{Trabalha}\})$

Checklist for FCA

- ☑ Find a hierarchical order in a set of elements
- ☑ Find relationships between two sets
- ☑ Find a hierarchical order in a relationship between two sets

How to compute formal concepts?

Concept Computation

Search Space Characterization

From (Andrews 2015):

- Finding all concepts from a context is equivalent to finding all Galois Connections in the form of concepts (A, B), where $B = A^{\uparrow}, A = B^{\downarrow}$.
- In a formal context (G, M, I), for every possible combination of attributes $B \subseteq M$, there is a concept $(B^{\downarrow}, B^{\downarrow\uparrow})$.
- We will start by generating all possible combinations of attributes, in a depth-first search.

Search Space Exploration: Naive Approach (Andrews 2015)

Algorithm 1 NaiveComputeConceptsOne(J, y)

1: **for** $j \leftarrow y$ **to** $n - 1$ **do**
2: $K \leftarrow J \cup \{j\}$
3: $A \leftarrow K^{\downarrow}$
4: $B \leftarrow A^{\uparrow}$
5: save(A, B)
6: NaiveComputeConceptsOne($K, j + 1$)
7: **end for**

- $J = \emptyset; J = \{0\}; J = \{0, 1\}...$
- Can we improve this algorithm?

Search Space Exploration: Naive Approach (Andrews 2015)

Algorithm 2 NaiveComputeConceptsOne(J, y)

1: **for** $j \leftarrow y$ **to** $n - 1$ **do**
2: $K \leftarrow J \cup \{j\}$
3: $A \leftarrow K^{\downarrow}$ {room for improvement}
4: $B \leftarrow A^{\uparrow}$
5: save(A, B)
6: NaiveComputeConceptsOne$(K, j + 1)$
7: **end for**

Search Space Exploration: Naive Approach II (Andrews 2015)

Algorithm 3 NaiveComputeConceptsTwo(A, y)

1: **for** $j \leftarrow y$ **to** $n - 1$ **do**
2: $\quad C \leftarrow A \cap \{j\}^{\downarrow}$
3: $\quad B \leftarrow C^{\uparrow}$
4: $\quad$ save(C, B)
5: $\quad$ NaiveComputeConceptsTwo($C, j + 1$)
6: **end for**

Search Space Exploration: First Problem

- The same concept can be computed more than once.
- The result will be hard to analyze!

Search Space Exploration: Compute Once (Andrews 2015)

Algorithm 4 ComputeConceptsOnce(A, y)

1: **for** $j \leftarrow y$ **to** $n - 1$ **do**
2: $\quad C \leftarrow A \cap \{j\}^{\downarrow}$
3: $\quad B \leftarrow C^{\uparrow}$
4: $\quad$ **if** NewConcept(C, B) **then**
5: $\quad\quad$ save(C, B)
6: $\quad\quad$ ComputeConceptsOnce($C, j + 1$)
7: $\quad$ **end if**
8: **end for**

Still, the concept is computed before checking if it is a new concept!

Combinatorial Problem!

- # possible combs. $= \binom{1}{n} + \binom{2}{n} + \cdots + \binom{n}{n}$

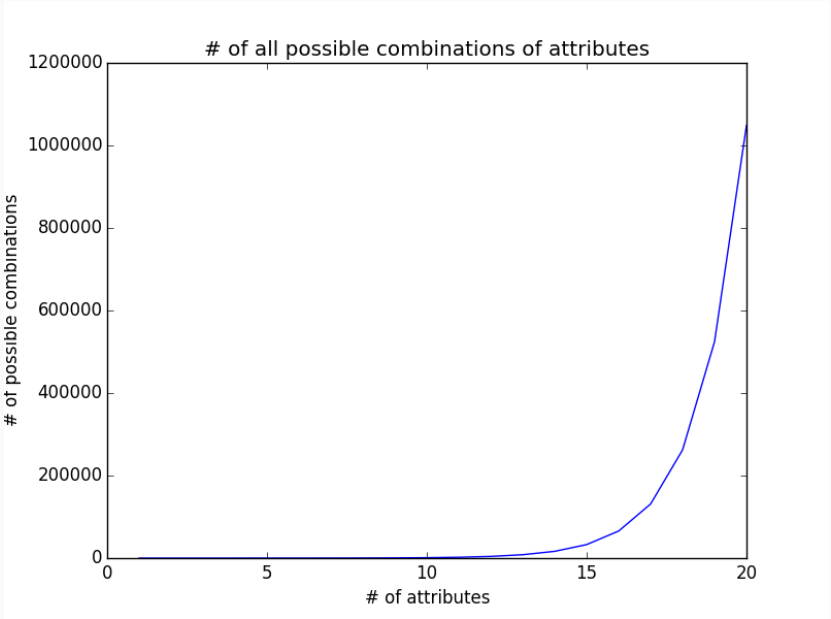


Figure 4: All possible combinations of attributes, by the number of attributes in a Formal Context.

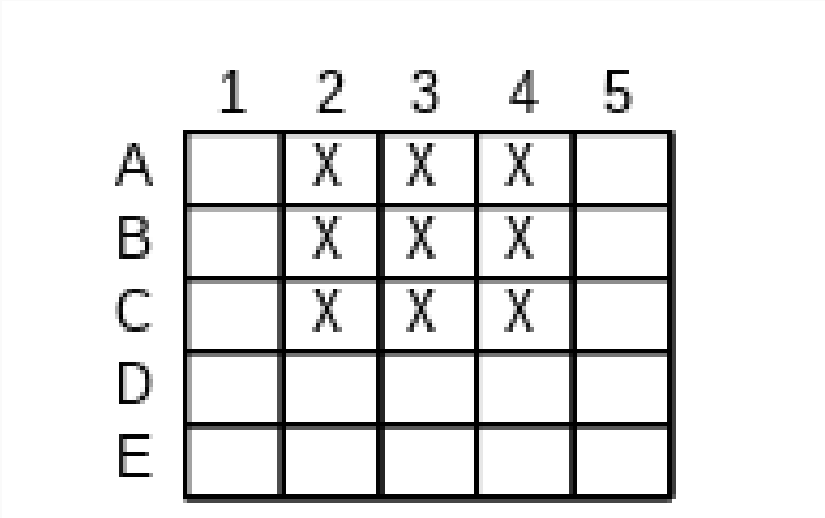
Canonical Order (Andrews 2015)

The canonical ordering

- $\{1\} \leq \{2\}$
- $\{1\} \leq \{1, 2\}$
- $\{1, 2\} \leq \{2\}$

How can that help?

- $Y_j := \{y \in Y | y < j\}$
- $B \cap Y_j = D \cap Y_j$



$$\{2\} \implies (\{A, B, C\}, \{2, 3, 4\})$$
$$\{2, 3\} \implies (\{A, B, C\}, \{2, 3, 4\})$$
$$\{2, 3, 4\} \implies (\{A, B, C\}, \{2, 3, 4\})$$

Search Space Exploration: Close By One (Andrews 2015)

Algorithm 5 ComputeConcepsFrom$((A, B), y)$

1: save$((A, B))$
2: **for** $j \leftarrow y$ **to** $n - 1$ **do**
3: **if** $j \notin B$ **then**
4: $C \leftarrow A \cap \{j\}^{\downarrow}$
5: $D \leftarrow C^{\uparrow}$
6: **if** $B \cap Y_j = D \cap Y_j$ **then**
7: ComputeConceptsFrom$((C, D), j + 1)$
8: **end if**
9: **end if**
10: **end for**

Search Space Exploration: Close By One (Andrews 2015)

Algorithm 6 ComputeConceptsFrom$((A, B), y)$

1: save$((A, B))$
2: **for** $j \leftarrow y$ **to** $n - 1$ **do**
3: **if** $j \notin B$ **then**
4: $C \leftarrow A \cap \{j\}^{\downarrow}$
5: $D \leftarrow C^{\uparrow}$
6: **if** $B \cap Y_j = D \cap Y_j$ **then**
7: ComputeConceptsFrom$((C, D), j + 1)$ {room for improvement}
8: **end if**
9: **end if**
10: **end for**

Avoiding Computation (Andrews 2015)

How can we avoid computing a concept before checking its canonicity?

- Partial closure $A^{\uparrow_j} := \{y \in Y_j | \forall x \in A : xIy\}$
- Partial closure canonicity test $B \cap Y_j = C^{\uparrow_j}$.

Search Space Exploration: In-Close (Andrews 2015)

Algorithm 7 ComputeConceptsFrom$((A, B), y)$

1: **for** $j \leftarrow y$ **to** $n - 1$ **do**
2: $\quad C \leftarrow A \cap \{j\}^{\downarrow}$
3: $\quad$ **if** $A = C$ **then**
4: $\quad\quad B \leftarrow B \cup \{j\}$
5: $\quad$ **else**
6: $\quad\quad$ **if** $B = C^{\uparrow_j}$ **then**
7: $\quad\quad\quad D \leftarrow B \cup \{j\}$
8: $\quad\quad\quad$ ComputeConceptsFrom$((C, D), j + 1)$
9: $\quad\quad$ **end if**
10: $\quad$ **end if**
11: **end for**

Another Possibilities of Improvement

- In-Close2: inherited intents (partial closure)
- FCbO: Failure inheritance + inherited intents (full closure)
- In-Close3: inherited intents + failure inheritance (partial closure)

Comparison By # of Attributes

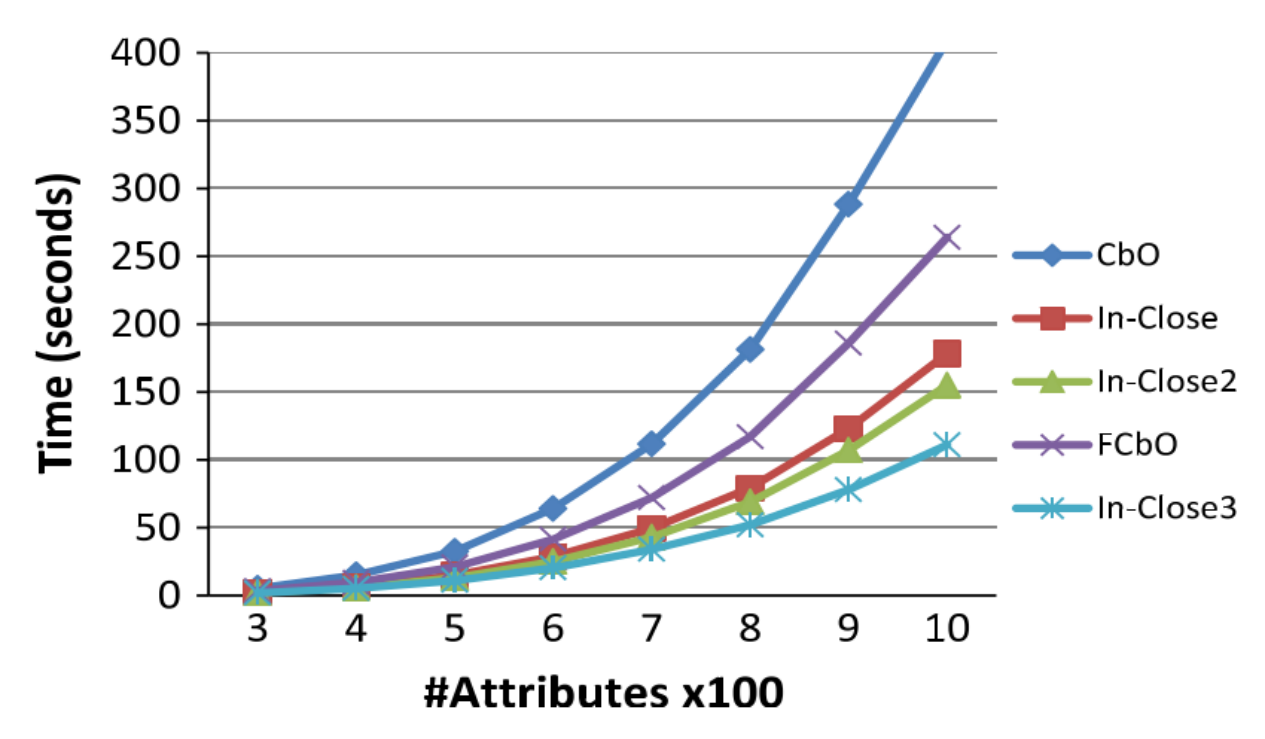


Figure 5: Comparison of performance with varying number of attributes (300-1000). 5% density, 5000 objects. # of concepts approx 1M-22M. From Andrews 2015.

Comparison By Density

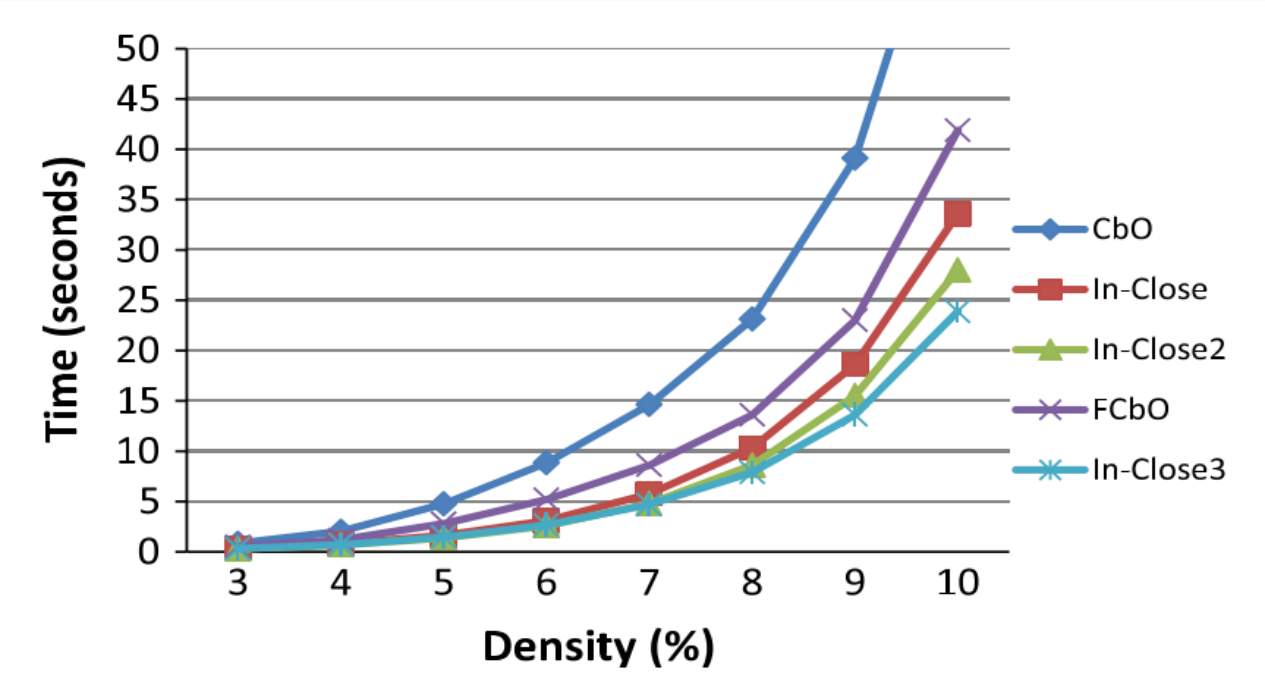


Figure 6: Comparison of performance with varying context density (3%-10%). 200 attributes, 10,000 objects. # of concepts approx 200m-19M. From Andrews 2015.

Comparison By # of Objects

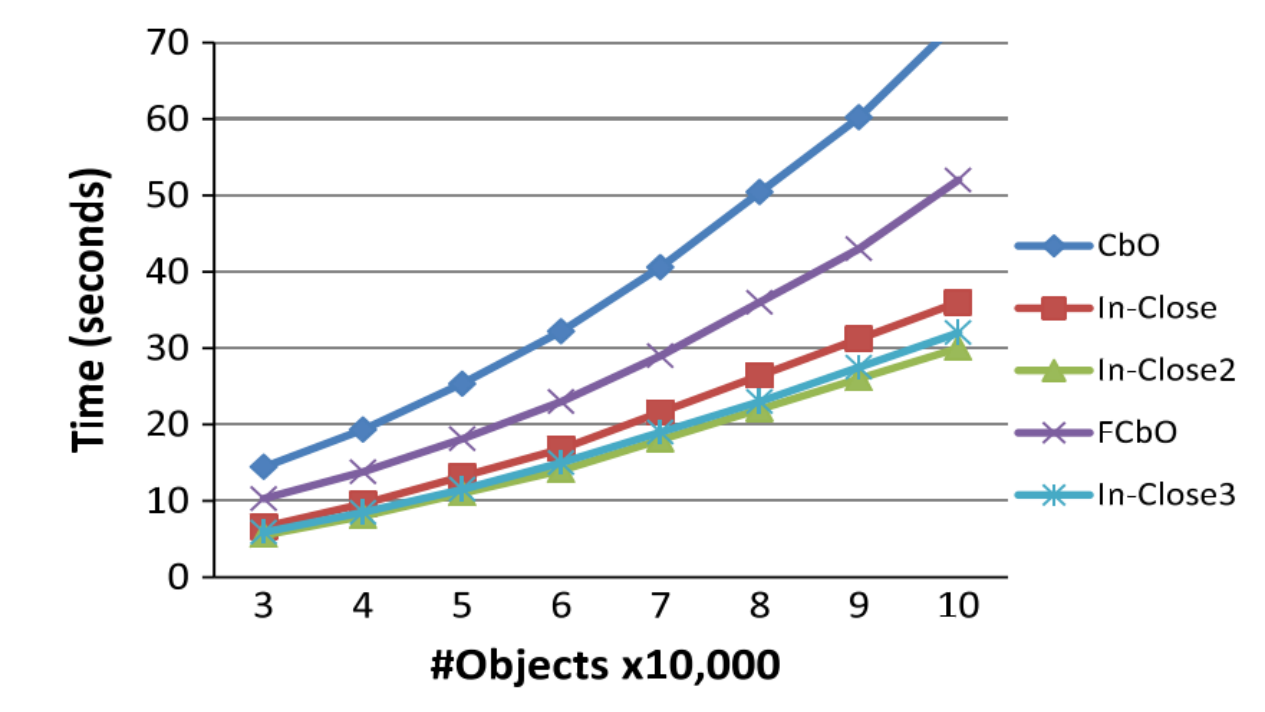


Figure 7: Comparison of performance with varying number of objects (30m-100m). 5% density, 200 attributes. # of concepts approx 4M-22M. From Andrews 2015.

A Toy Example

Table 1: Toy example

	Tem Filhos	Trabalha	Cozinha
Nilson	X	X	X
Cássia	X	X	
Aline	X	X	
Amanda			X
Alan		X	
Beatriz			

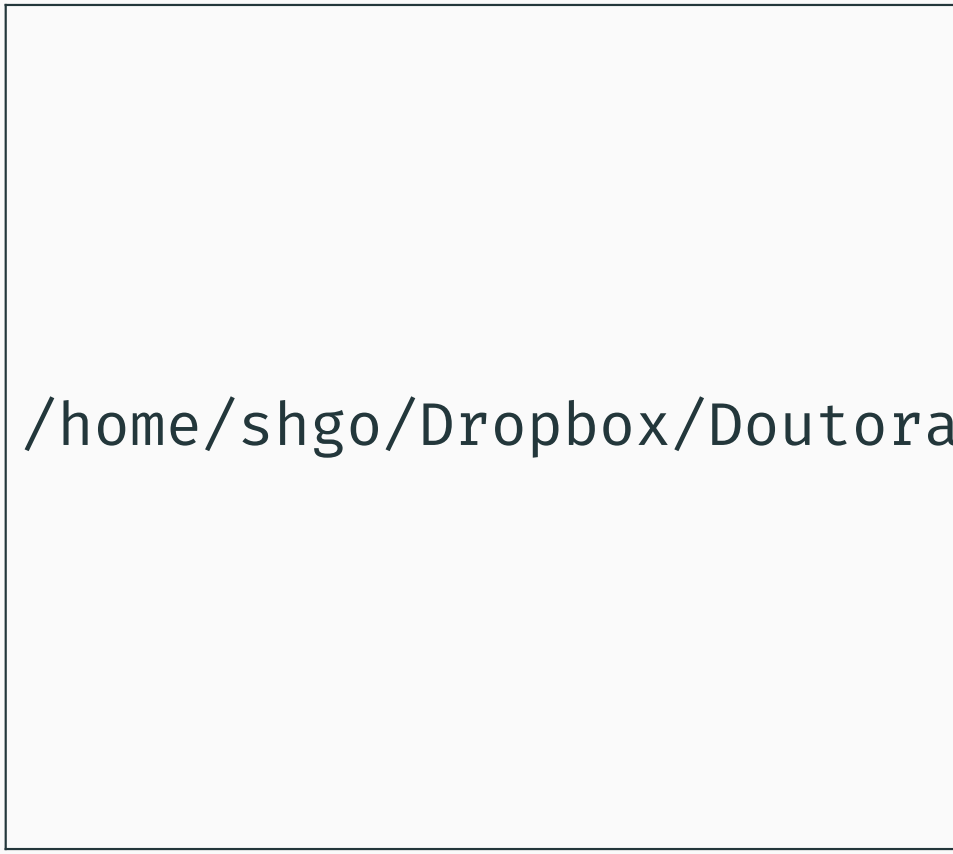
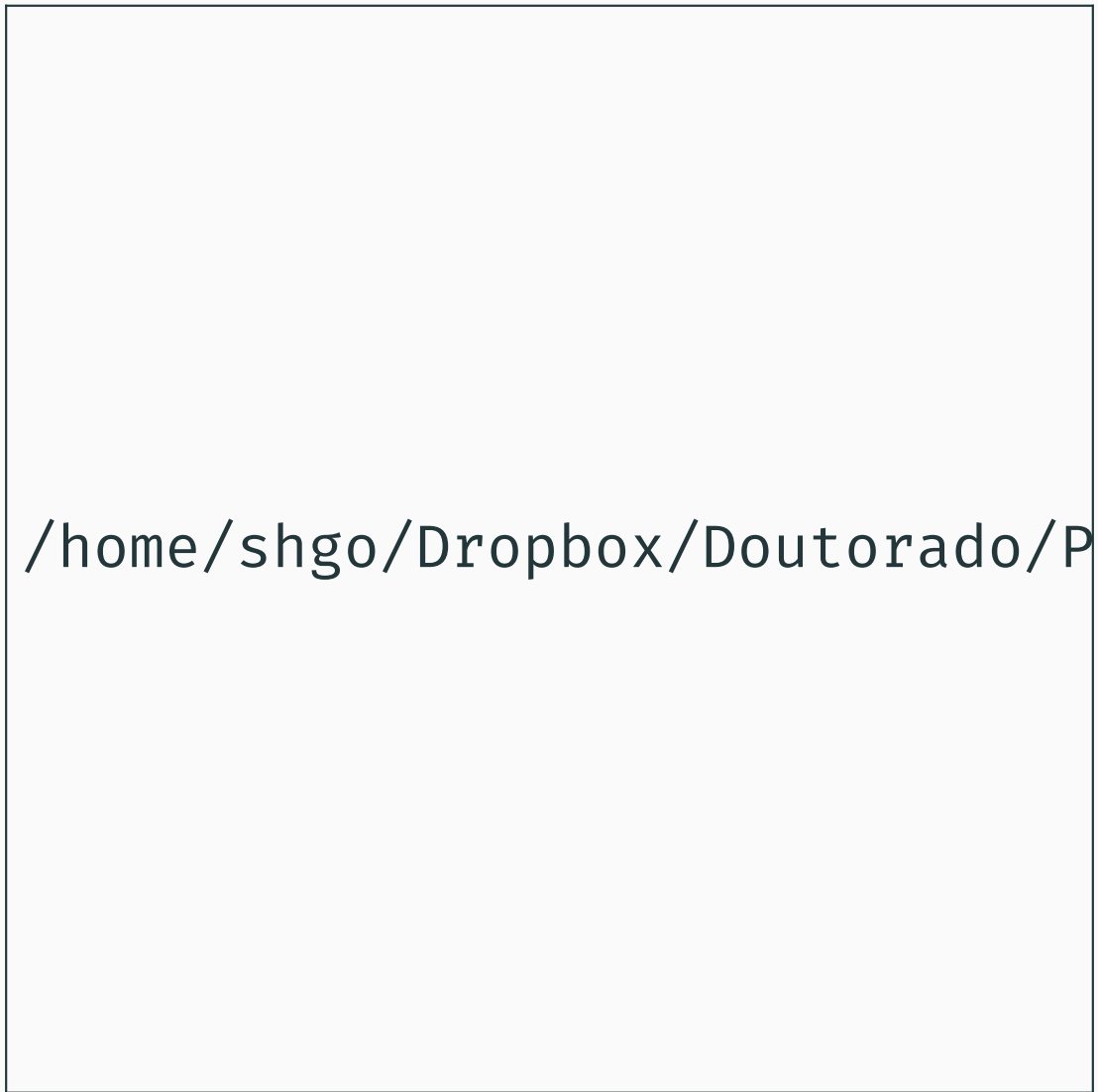


Figure 1: Toy example concept lattice.

An Extension to Graphs

- How can I include object relationships in my analysis?



Graph-FCA

From Dyadic to Graph data

- We can model object dependence structure by graphs.
- Several kinds of dependence: (un)directed, hyper-edges, edge-labels...
- Complex network is a big interdisciplinary research field completely based on graphs.

Can FCA offer good tools for Graph Mining?

- Concepts, extension and intension.
- Generalizing order of graph concepts.
- New applications for the FCA framework, such as molecule structure investigation, social network analysis, etc.

Checklist for Graph-FCA

- ☐ Graph context
- ☐ Graph concept extension/intension
- ☐ Galois connection
- ☐ Graph concept lattice

Graph Context

Definition (Ferré 2015)

A graph formal context is a triple $K = (O, A, I)$, where O is a set of objects, A is a set of attributes, and $I \subseteq O^* \times A$ is an incidence relation between object tuples and attributes. The maximum cardinality of incidences is denoted by $|K|$.

- Objects are nodes, incidence elements are hyper-edges, and attributes are hyper-edge labels.

Graph Context: Example

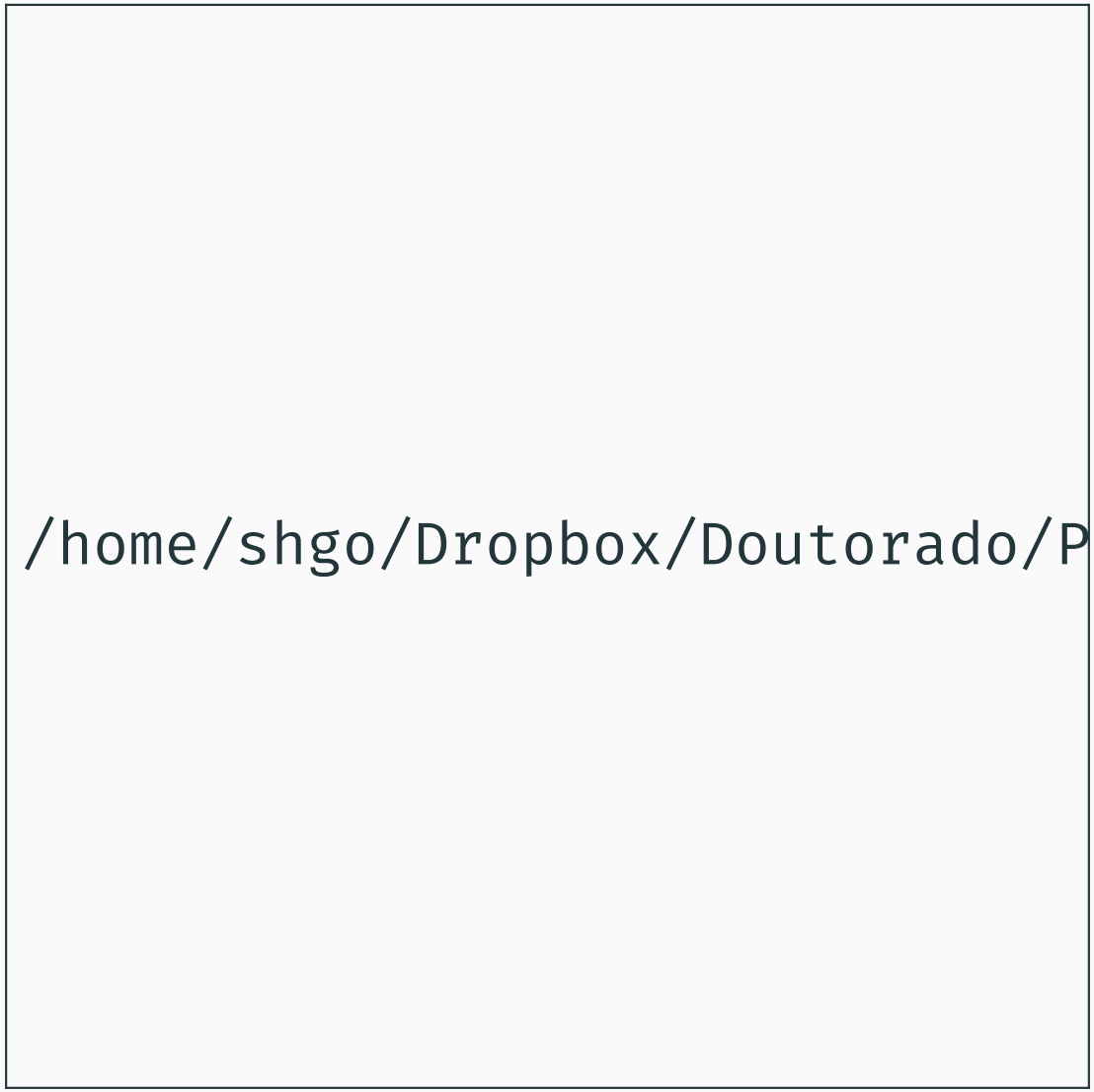


Figure 9: British Royal Family, from Ferré 2015.

Checklist for Graph-FCA

- ☑ Graph context
- ☐ Graph concept extension/intension
- ☐ Galois connection
- ☐ Graph concept lattice

Projected Graph Patterns

Definition (Ferré and Cellier 2016)

A graph pattern $P \subseteq \mathcal{V}^* \times A$ is a set of directed hyperdedges with variables and/or objects as nodes, and attributes as labels.

Definition (Ferré and Cellier 2016)

A projected graph pattern (PGP) is a couple $Q = (\bar{x}, P)$, where P is a graph pattern, and $\bar{x} \in \mathcal{V}^*$ is called the projection tuple. The arity of a PGP is the length of $\bar{x}$. We note $\mathcal{Q}_k$ the set of PGPs having arity k.

Projected Graph Patterns: Example From Ferré and Cellier 2016

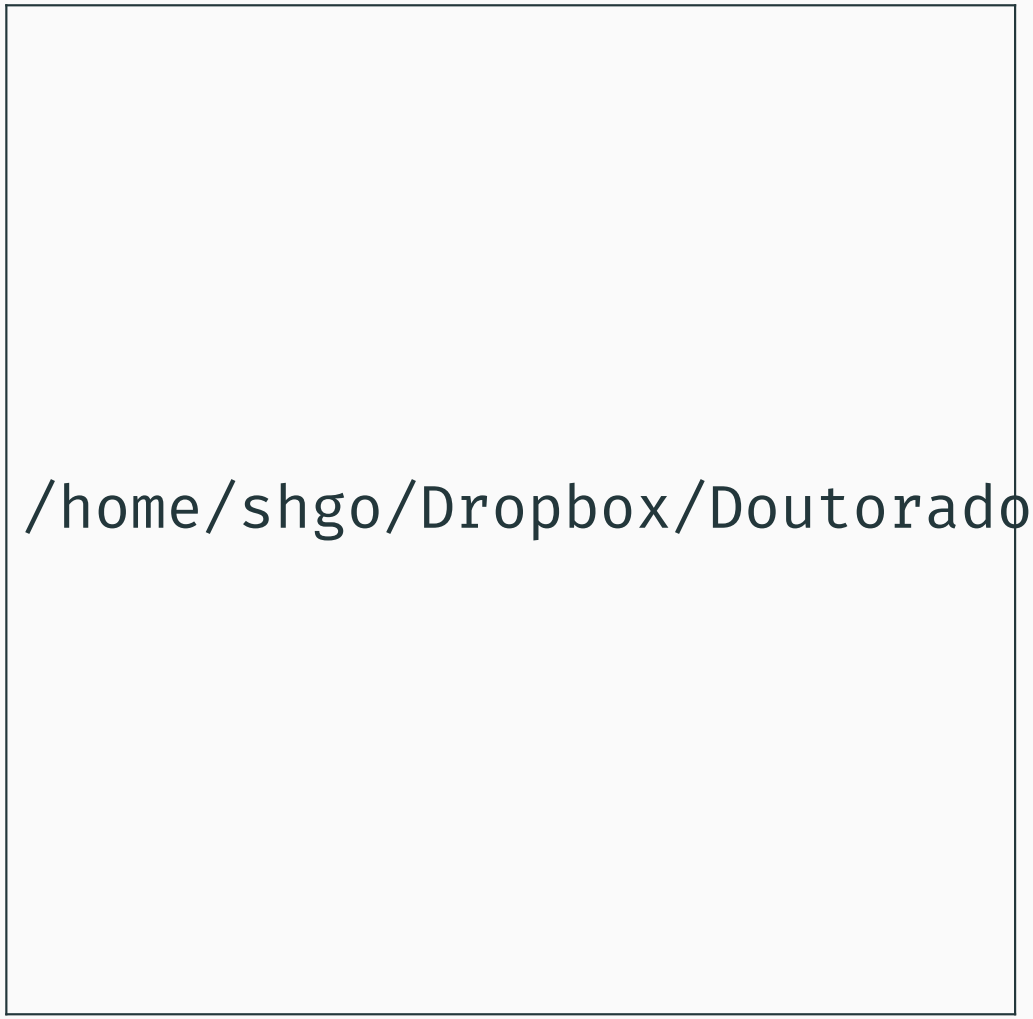


Figure 10: British Royal Family

Figure 11: Sibiling binary relation.

How can we order a set of PGPs?

Definition (Ferré and Cellier 2016)

A k-PGP $Q_1 = (\overline{x_1}, P_1)$ is included in a k-PGP $Q_2 = (\overline{x_2}, P_2)$, denoted by $Q_1 \subseteq_q Q_2$, iff there is a mapping ϕ from P_1-nodes to P_2-nodes s.t. $\overline{\phi(x_1)} = \overline{x_2}$, and for every edge $(\overline{y}, a) \in P_1$, $(\overline{\phi(y)}, a) \in P_2$.

- ϕ is a homomorphism from P_1 to P_2 that preserves the projection tuple and the edges.
- When $Q_1 \subseteq_q Q_2$ and $Q_2 \subseteq_q Q_1$, they are said equivalent ($Q_1 \equiv_q Q_2$).

Graph Homomorphism

Brewster 2006

Graph Homomorphisms

Definition 1 *Let G and H be graphs. A homomorphism of G to H is a*

Graph Homomorphisms

Definition 1 *Let G and H be graphs. A homomorphism of G to H is a function $f : V(G) \to V(H)$ such that*

$$xy \in E(G) \Rightarrow f(x)f(y) \in E(H).$$

Graph Homomorphisms

Definition 1 *Let G and H be graphs. A homomorphism of G to H is a function $f : V(G) \to V(H)$ such that*

$$xy \in E(G) \Rightarrow f(x)f(y) \in E(H).$$

Adjacent vertices receive adjacent images.

Graph Homomorphisms

Definition 1 *Let G and H be graphs. A homomorphism of G to H is a function $f : V(G) \to V(H)$ such that*

$$xy \in E(G) \Rightarrow f(x)f(y) \in E(H).$$

We write $G \to H$ ($G \not\to H$) if there is a homomorphism (no homomorphism) of G to H.

Beyond graphs

Definition of a homomorphism naturally extends to:

- digraphs;
- edge-coloured graphs;
- relational systems.

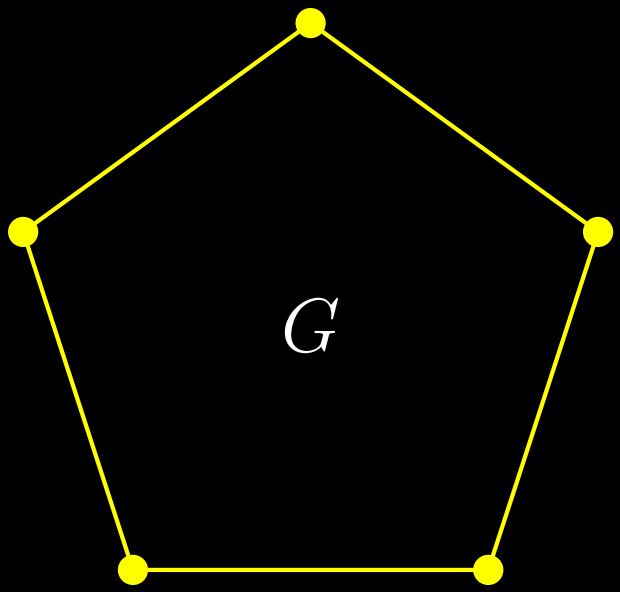
Beyond graphs

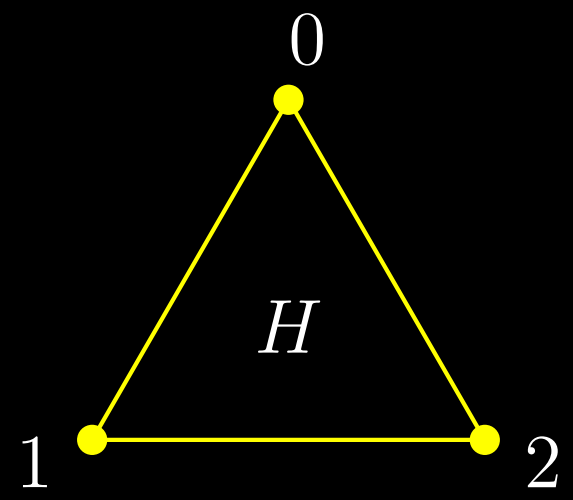
Definition of a homomorphism naturally extends to:

- digraphs;
- edge-coloured graphs;
- relational systems.

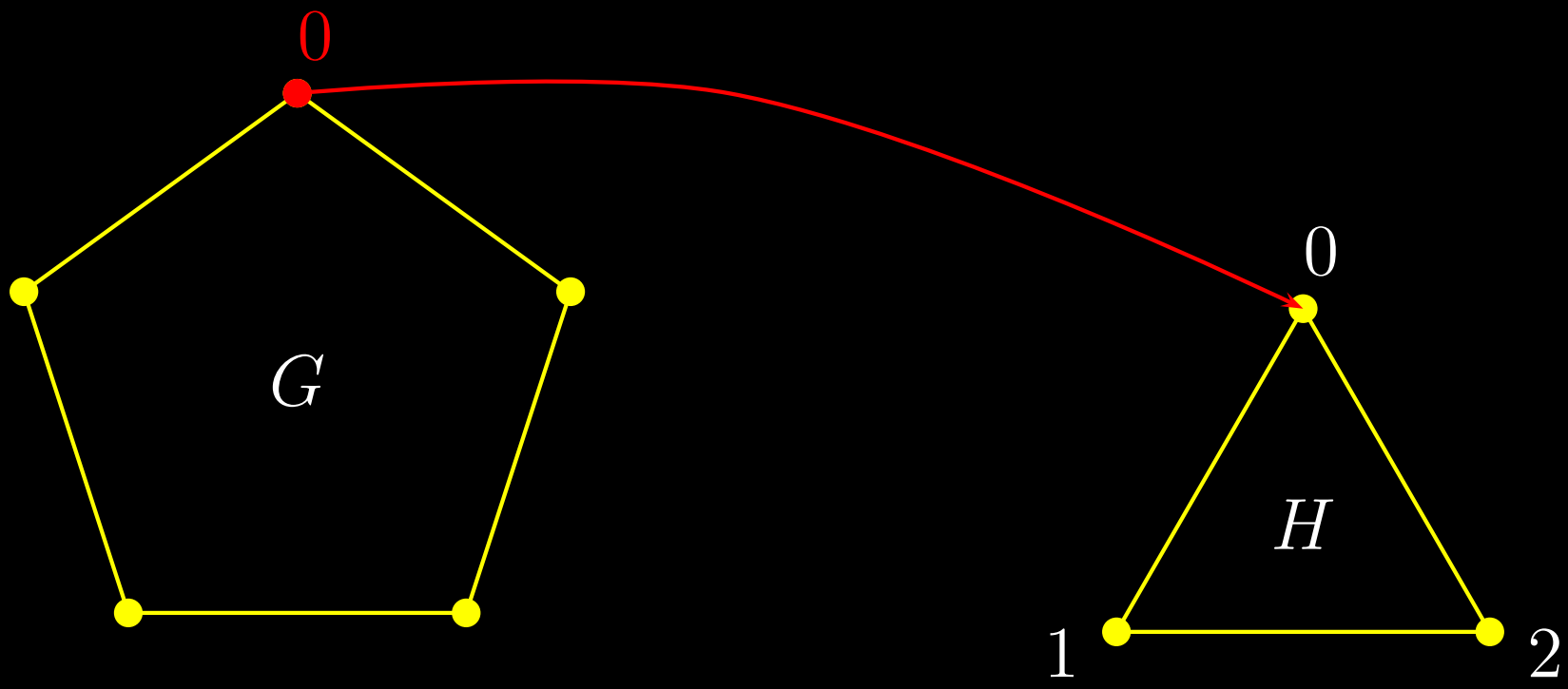
Hot idea: *Constraint Satisfaction Problems* encoded as homomorphisms.

An example

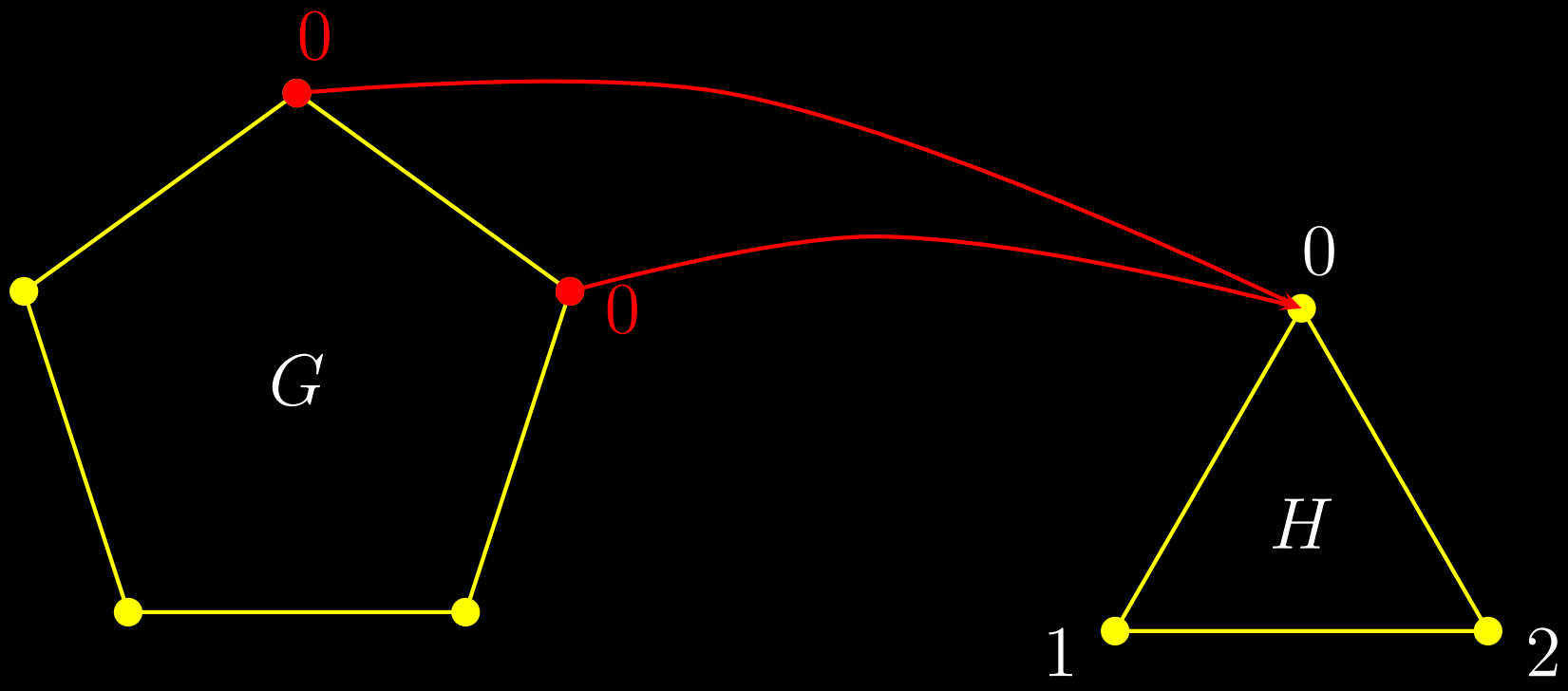




An example

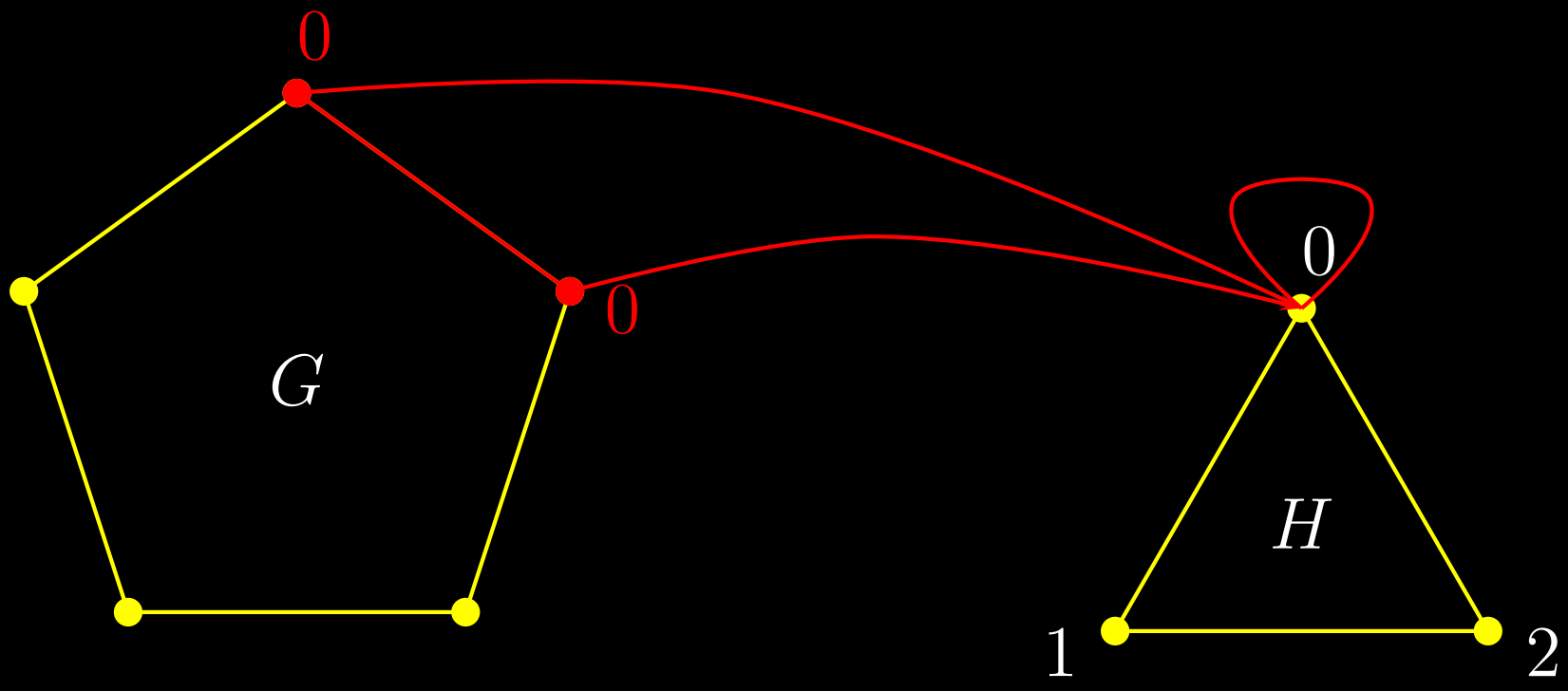


An example



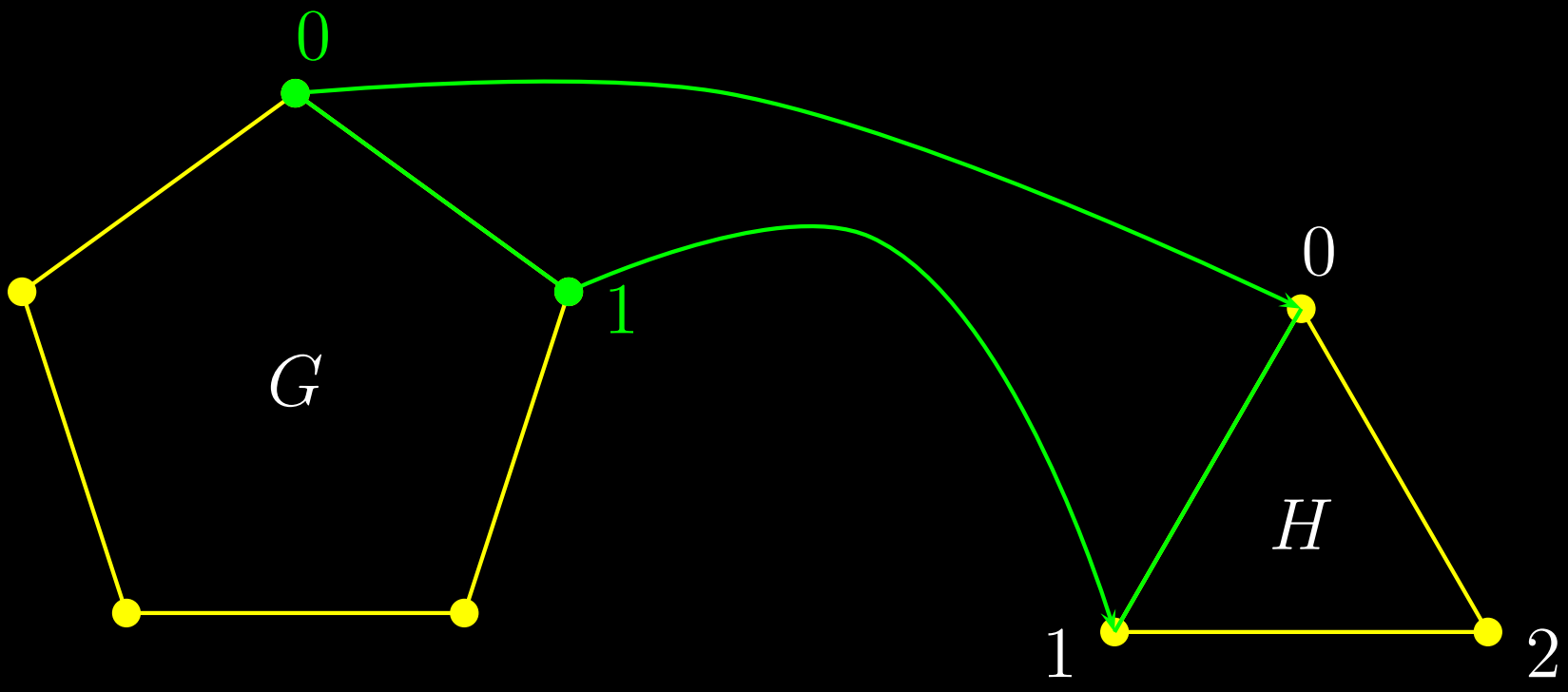
Why is this assignment not allowed?

An example



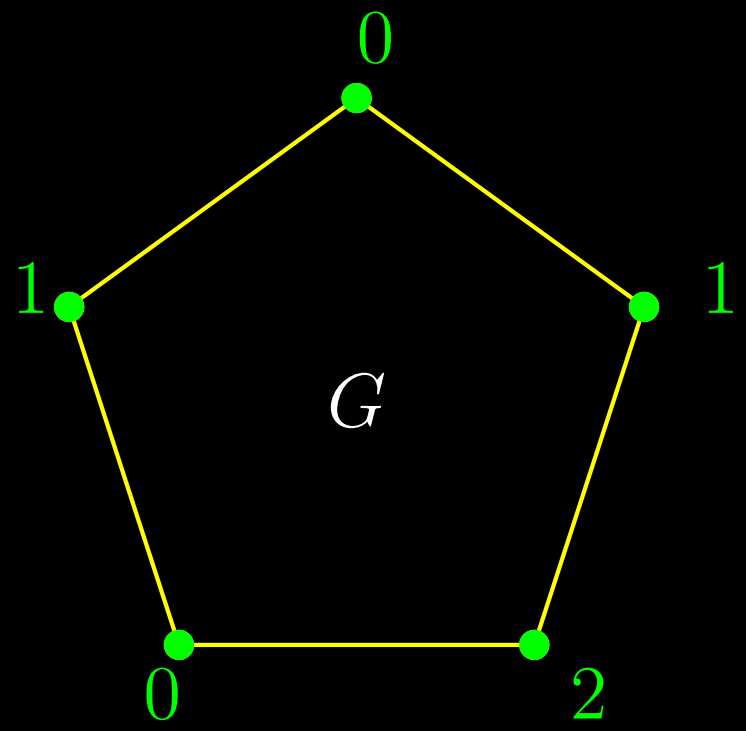
This assignment requires a loop on vertex 0 (in H)

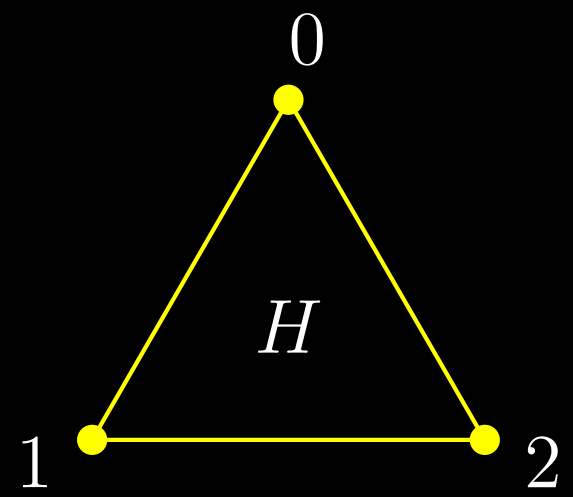
An example



This assignment is allowed.

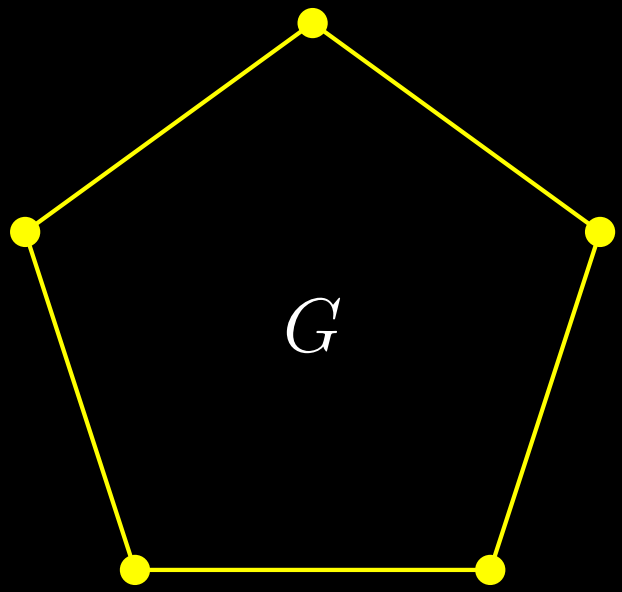
An example

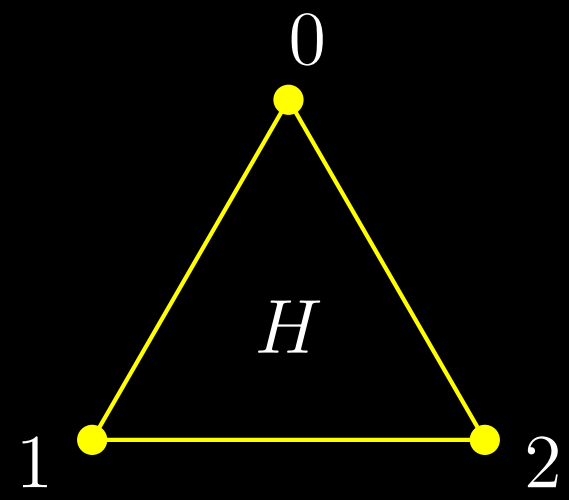




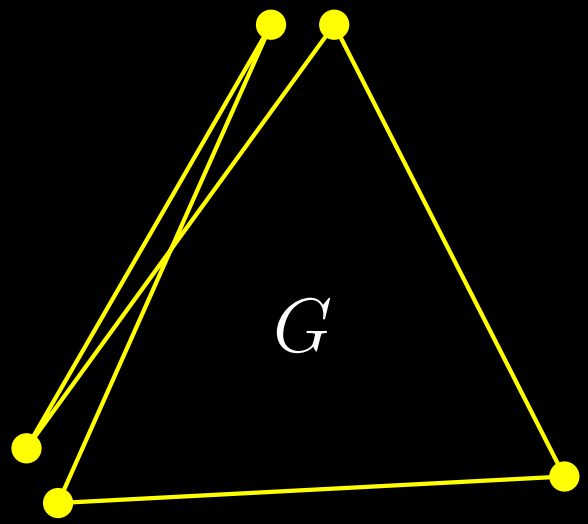
This labeling is a homomorphism $G \to H$.

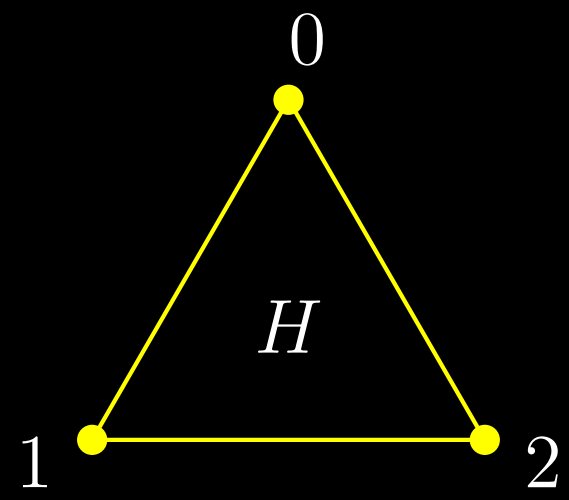
A partitioning problem



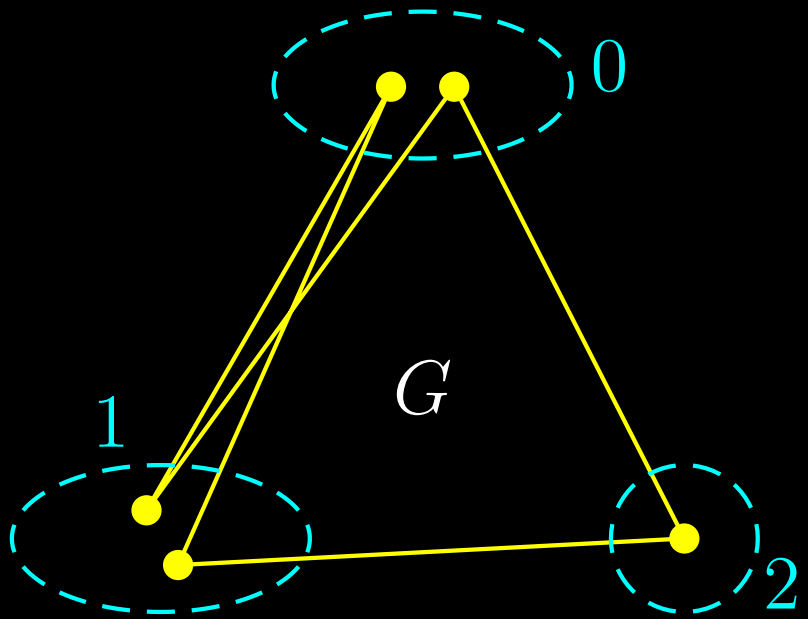


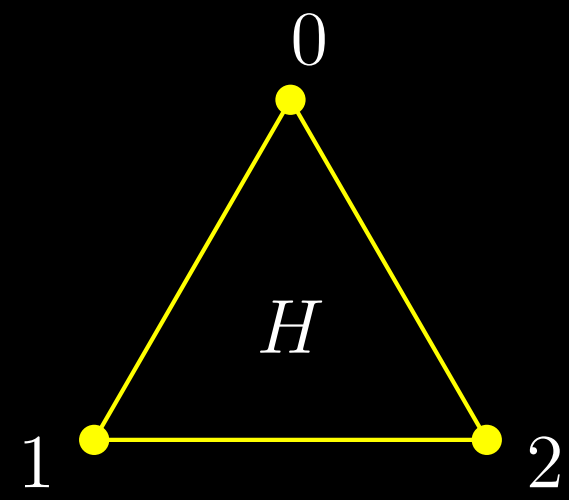
A partitioning problem





A partitioning problem





The *quotient* of the partition is a subgraph of H

The partition is the *kernel* of the map.

How can we order a set of PGPs?

Definition (Ferré and Cellier 2016)

A k-PGP $Q_1 = (\overline{x_1}, P_1)$ is included in a k-PGP $Q_2 = (\overline{x_2}, P_2)$, denoted by $Q_1 \subseteq_q Q_2$, iff there is a mapping ϕ from P_1-nodes to P_2-nodes s.t. $\overline{\phi(x_1)} = \overline{x_2}$, and for every edge $(\overline{y}, a) \in P_1$, $(\overline{\phi(y)}, a) \in P_2$.

- ϕ is a homomorphism from P_1 to P_2 that preserves the projection tuple and the edges.
- When $Q_1 \subseteq_q Q_2$ and $Q_2 \subseteq_q Q_1$, they are said equivalent ($Q_1 \equiv_q Q_2$).

Example of PGP inclusion

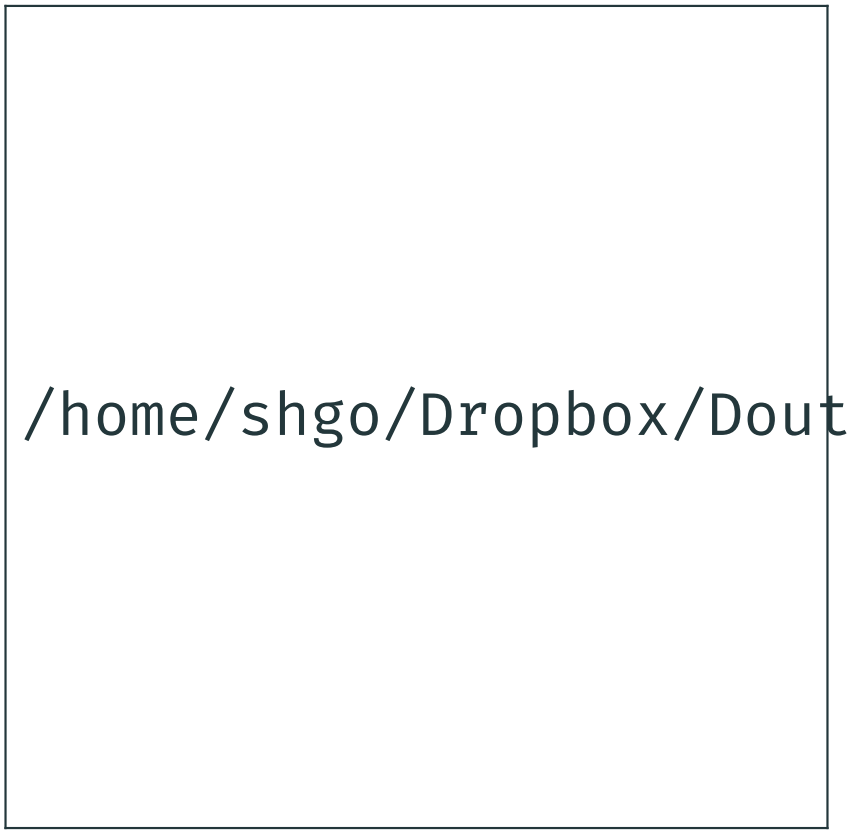


Figure 12: Example of PGP Inclusion, based on Ferré and Cellier 2016.

- $Q_1 \subseteq Q_2$
- $\phi(x_1) = x_2$
- $\phi(y_1) = y_2$

We will also need a PGP intersection $\cap_q$

Definition (Ferré and Cellier 2016)

Let ψ be an injective mapping from pairs of variables to variables. The intersection of two k-PGPs $Q_1 = (\overline{x_1}, P_1)$ and $Q_2 = (\overline{x_2}, P_2)$, denoted by $Q_1 \cap_q Q_2$, is defined as $Q = (\overline{x}, P)$, where $\overline{x} = \overline{\psi(x_1, x_2)}$, and $P = \{(\overline{\psi(y_1, y_2)}, a) | a \in A, (\overline{y_1}, a) \in P_1, (\overline{y_2}, a) \in P_2, |\overline{y_1}| = |\overline{y_2}|\}$.

- Graph alignment.
- $\overline{x_1} = \{a_1, b_1, c_1\}, \overline{x_2} = \{a_2, b_2, c_2\}$
- $\psi(a_1, a_2) = a$;
- $\psi(a_1, b_2) = b$;
- $\psi(a_1, c_2) = c$;

Example of PGP intersection $\cap_q$

- $\psi(x_1, x_2) = x$
- $\psi(y_1, y_2) = y$
- $\psi(z_1, y_2) = z$

Figure 13: Example of PGP Intersection.

Object Relations

Definition (Ferré and Cellier 2016)

An object relation $\mathcal{R} \subseteq O^k$ is a tuple of objects. Its arity is the lenght of the tuple. We note $\mathcal{R}_k$ the set of relations with arity k.

Checklist for Graph-FCA

- ☑ Graph context
- ☑ Graph concept extension/intension
- ☐ Galois connection
- ☐ Graph concept lattice

From Patterns to Relations and Back

Definition (Galois connection, Ferré and Cellier 2016)

Let $K = (O, A, I)$ be a graph context. For every arity k, the following pair of mappings between PGPs $Q \in Q_k$ and object relations $R \in \mathcal{R}_k$ forms a Galois connection.

$$Q' := \{\overline{o} \in O^k | Q \subseteq_q (\overline{o}, I)\} \qquad R' := \cap_q \{(\overline{o}, I)\}_{\overline{o} \in R}$$

- Very informally: $Q^{'} \approx Q^{\downarrow}$ and $R^{'} \approx R^{\uparrow}$.

Formal Graph Concepts

Definition (Ferré and Cellier 2016)

A graph concept with arity k is a pair (R, Q), made of an object relation $R \in \mathcal{R}_k$ (the extent) and a PGP $Q \in \mathcal{Q}_k$ (the intent), such that $R = Q'$ and $Q = R'$.

Figure 14: Sibiling relation PGP, from Ferré and Cellier 2016.

Checklist for Graph-FCA

- ☑ Graph context
- ☑ Graph concept extension/intension
- ☑ Galois connection
- ☐ Graph concept lattice

Supremum and Infimum

- PGP inclusion $\subseteq_q$ is a pre-order over PGPs.
- PGP Union of a finite collection of PGPs gives us a supremum relative to PGP inclusion.
- PGP intersection $\cap_q$ gives us an infimum relative to PGP inclusion.
- Proofs in Ferré 2015.

Checklist for Graph-FCA

- ☑ Graph context
- ☑ Graph concept extension/intension
- ☑ Galois connection
- ☑ Graph concept lattice

Graph-FCA in Practice

Difficulties on Graph-FCA Computation

- Complexity of computing with graphs.
- Graph Homomorphism.
- PGP intersection can lead to a bigger graph.
- How can I find canonical representations of PGPs, like in FCA attributes?
- In this section, from now on the reference is Ferré and Cellier 2016.

Naive Computation

- Generate all combinations of extents (Object Relations).
- For each one, compute the intent (GPG) using the Galois connection $R' := \cap_q \{(\overline{o}, I)\}_{\overline{o} \in R}$.

Algorithm 8 Generation of concepts$((A, B), y)$

Require: $K = (O, A, I)$ is a graph context

1: $Concepts \leftarrow \emptyset$
2: $Patterns \leftarrow \{I\}$
3: **for all** $P \in Patterns$ **do**
4: **for all** new $P_a \in ConnectedComponents(P \cap_q I)$ **do**
5: $P_a \leftarrow removeDuplicateNodes(P_a)$
6: **for all** new $P_b \in Retracts(P_a)$ **do**
7: $X \leftarrow P_b.nodes \setminus \bigcup_{R \in Retracts(P_a) | R \subseteq P_b} R.nodes$
8: **if** $X \neq \emptyset$ **then**
9: **for all** $x \in X$ **do**
10: $Q \leftarrow ((x), P_b); R \leftarrow Q'$
11: $Concepts \leftarrow \{(R, Q) \in P_b\} \cup Concepts$
12: **end for**
13: $Patterns \leftarrow \{P_b\} \cup Patterns$
14: **end if**
15: **end for**
16: **end for**
17: **end for**

ConnectedComponents($P \cap_q I$)

How to get all Graph Patterns

The algorithm starts by aligning I with itself and using the connected components as Graph Patterns. This alignment is done using the Categorical Product.

Definition

Let G and H be two graphs. The Categorical Product $P = G \times H$ is the graph where $N(P) = N(G) \times N(H)$ and $E(P) = \{[(u, x), (v, y)] : [u, v] \in E(G), [x, y] \in E(H)\}$

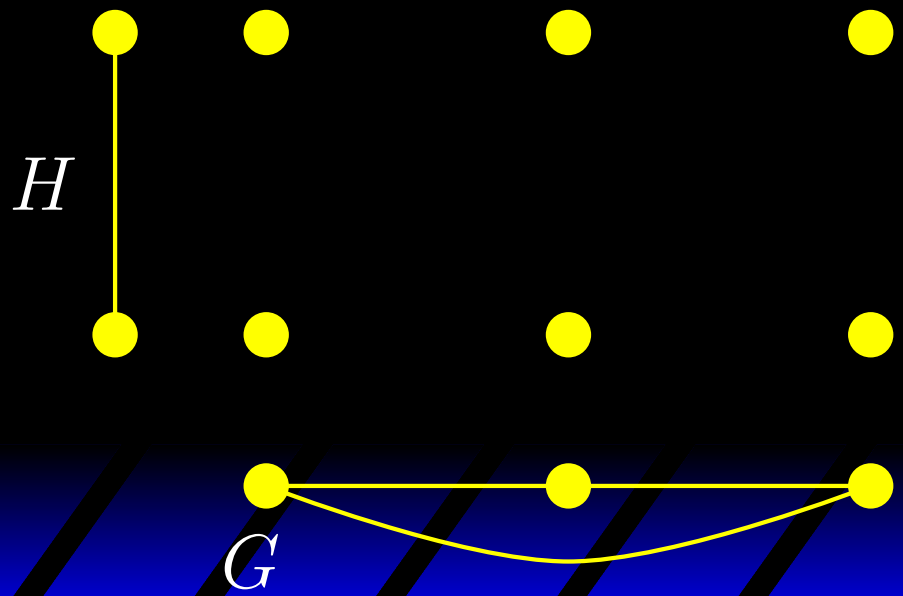
Graph Categorical Product

Brewster 2006

Products

The natural product with homomorphisms is the *categorical product* $G \times H$.

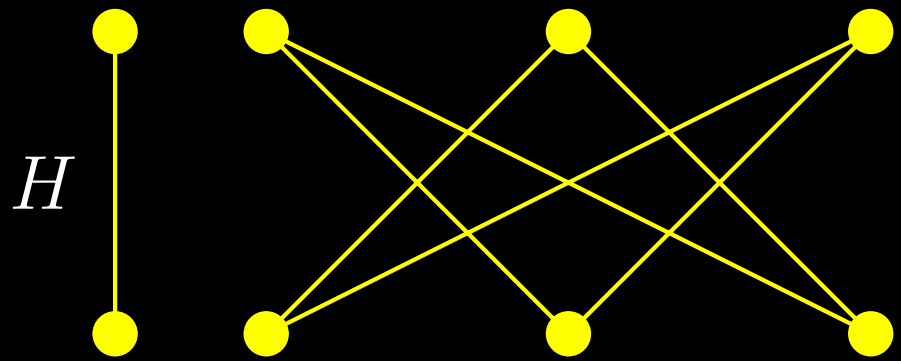
$$(g_1, h_1)(g_2, h_2) \in E(G \times H)$$
$$\Leftrightarrow \quad g_1 g_2 \in E(G) \text{ and } h_1 h_2 \in E(H)$$

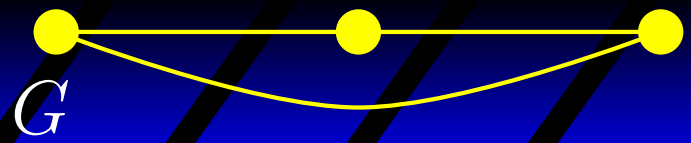


Products

The natural product with homomorphisms is the *categorical product* $G \times H$.

$$(g_1, h_1)(g_2, h_2) \in E(G \times H)$$
$$\Leftrightarrow \quad g_1 g_2 \in E(G) \text{ and } h_1 h_2 \in E(H)$$

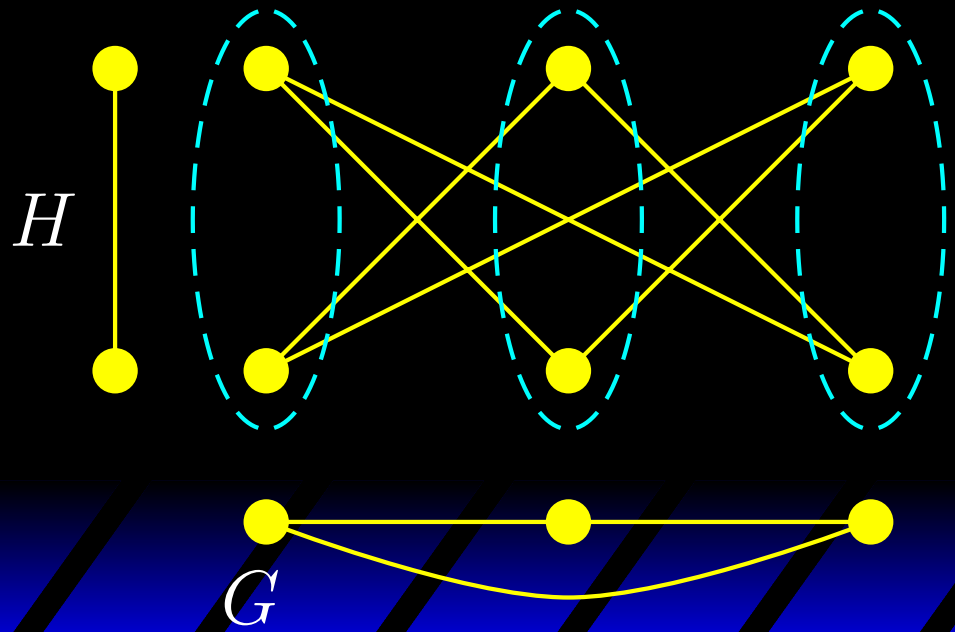




Products

The natural product with homomorphisms is the *categorical product* $G \times H$.

$$(g_1, h_1)(g_2, h_2) \in E(G \times H)$$
$$\Leftrightarrow \quad g_1 g_2 \in E(G) \text{ and } h_1 h_2 \in E(H)$$



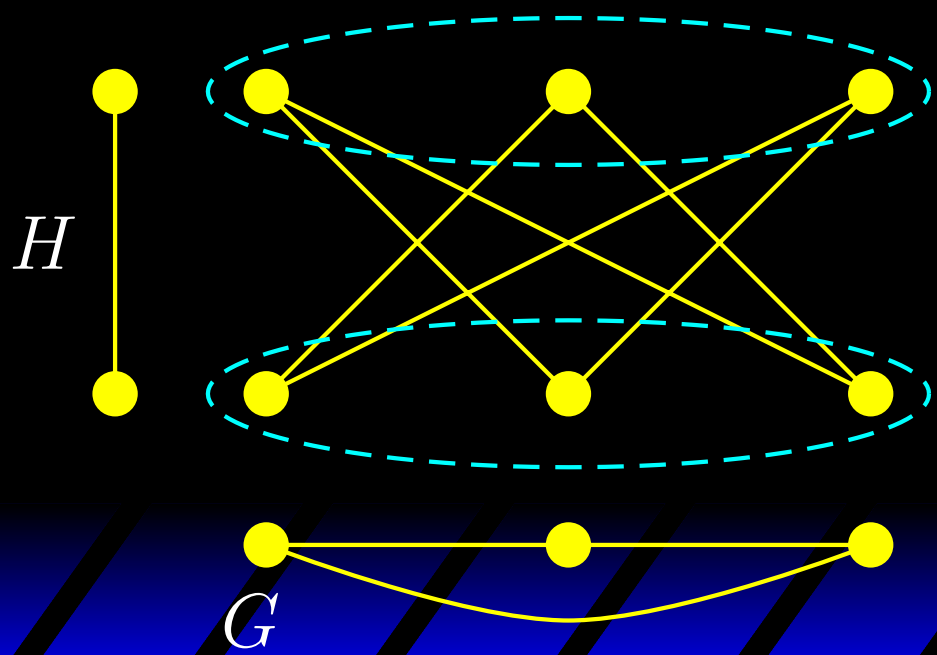
$$\pi_1 : G \times H \to G$$

Projections are homomorphisms

Products

The natural product with homomorphisms is the *categorical product* $G \times H$.

$$(g_1, h_1)(g_2, h_2) \in E(G \times H)$$
$$\Leftrightarrow \quad g_1 g_2 \in E(G) \text{ and } h_1 h_2 \in E(H)$$



$$\pi_2 : G \times H \to H$$

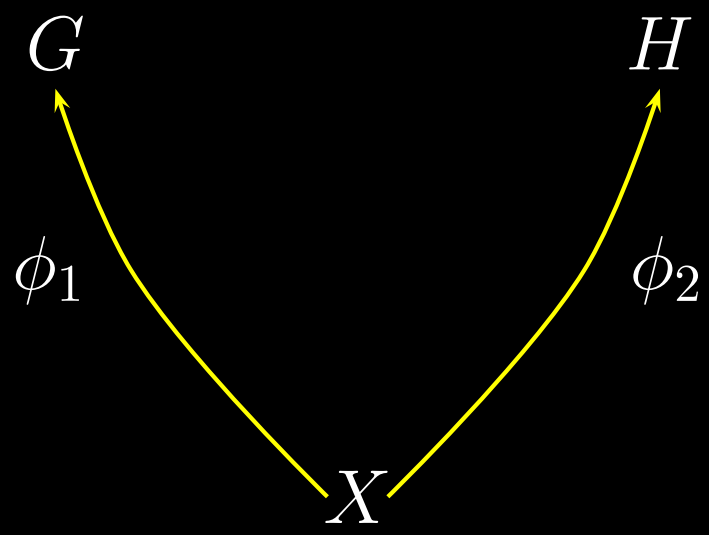
Projections are homomorphisms

Products (2)

Prop 6 $X \to G \times H$ *iff* $X \to G$ *and* $X \to H$

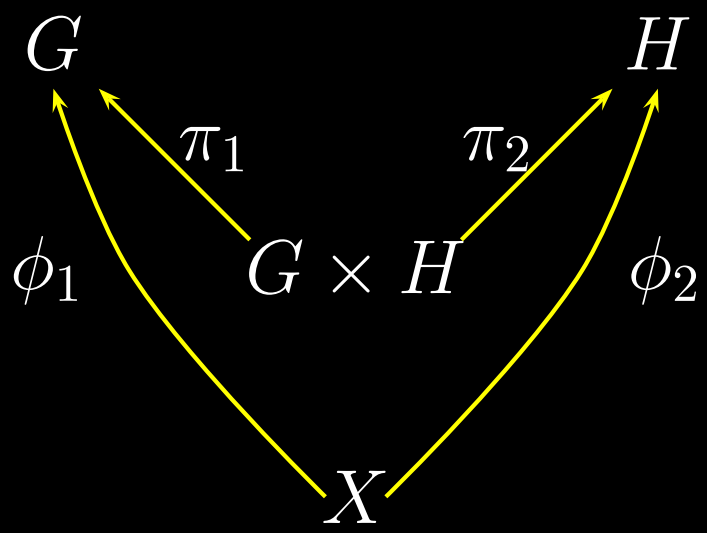
Products (2)

Prop 6 $X \to G \times H$ *iff* $X \to G$ *and* $X \to H$



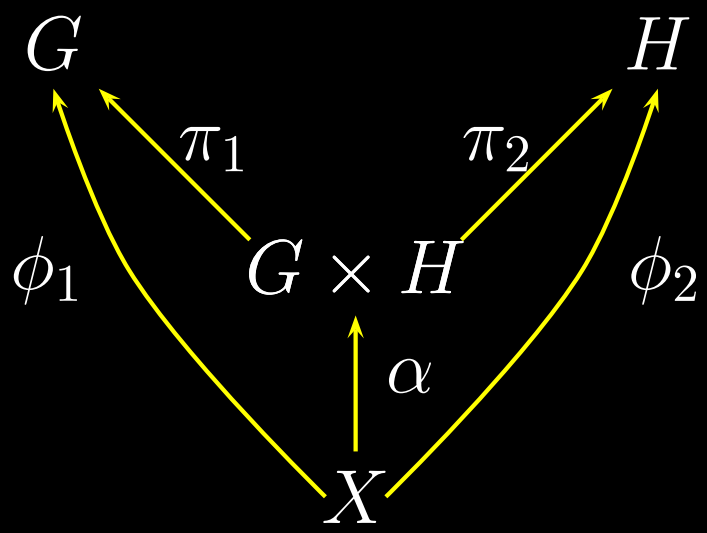
Products (2)

Prop 6 $X \to G \times H$ *iff* $X \to G$ *and* $X \to H$



Products (2)

Prop 6 $X \to G \times H$ *iff* $X \to G$ *and* $X \to H$



$$\alpha(x) := (\phi_1(x), \phi_2(x)) \qquad (= \phi_1 \times \phi_2(x))$$

$$\phi_1 = \pi_1 \circ \alpha \qquad \phi_2 = \pi_2 \circ \alpha$$

Algorithm 9 Generation of concepts$((A, B), y)$

Require: $K = (O, A, I)$ is a graph context

Concepts $\leftarrow \emptyset$
Patterns $\leftarrow \{I\}$
for all $P \in$ *Patterns* **do**
  for all new $P_a \in$ *ConnectedComponents*$(P \cap_q I)$ **do**
    $P_a \leftarrow$ *removeDuplicateNodes*(P_a)
    for all new $P_b \in$ *Retracts*(P_a) **do**
      $X \leftarrow P_b.nodes \setminus \bigcup_{R \in Retracts(P_a) | R \subseteq P_b} R.nodes$
      if $X \neq \emptyset$ **then**
        for all $x \in X$ **do**
          $Q \leftarrow ((x), P_b); R \leftarrow Q'$
          Concepts $\leftarrow \{(R, Q) \in P_b\} \cup$ *Concepts*
        end for
        Patterns $\leftarrow \{P_b\} \cup$ *Patterns*
      end if
    end for
  end for
end for

Retracts and Core Graphs

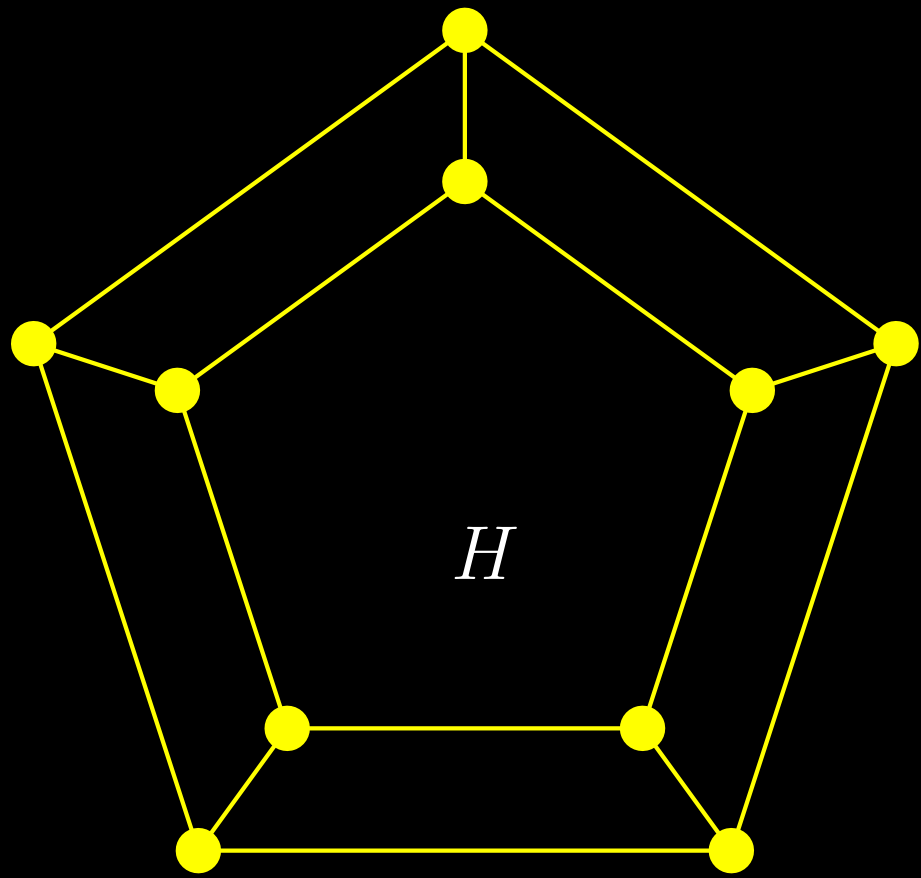
Brewster 2006

The core of a graph

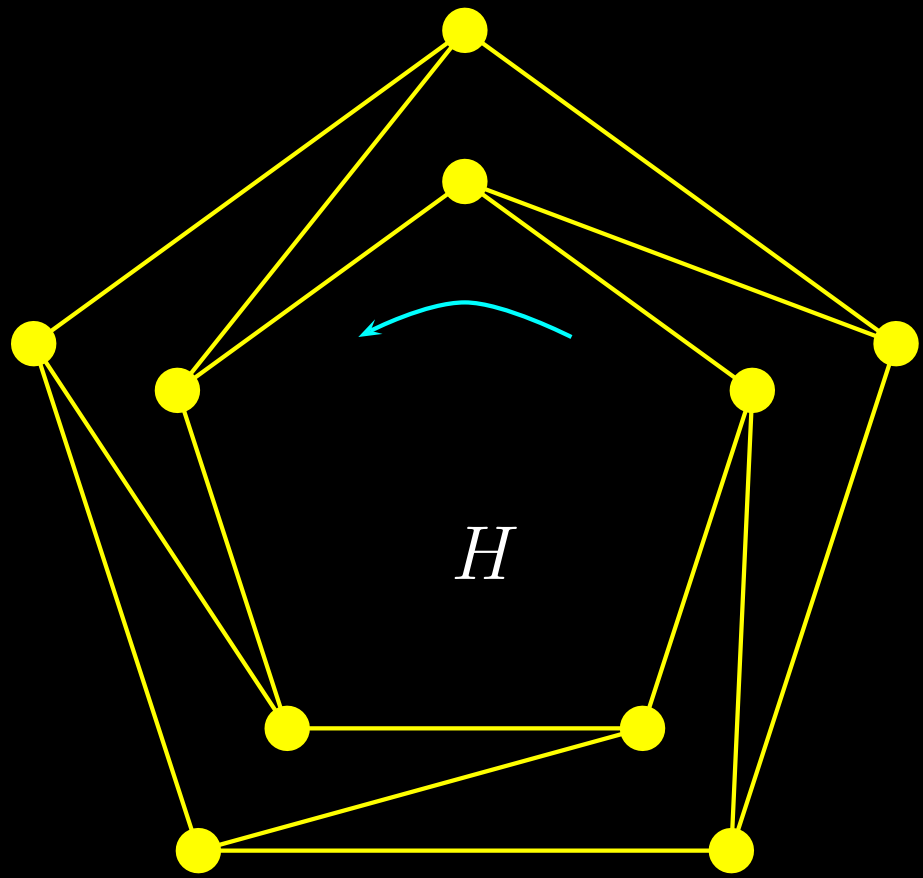
In our example,

- $H \to K_3$ and $K_3 \hookrightarrow H$.
- H and K_3 are *homomorphism equivalent*.
- Every graph has a unique (up to iso) inclusion minimal subgraph to which it is hom-equivalent called the *core* of the graph.

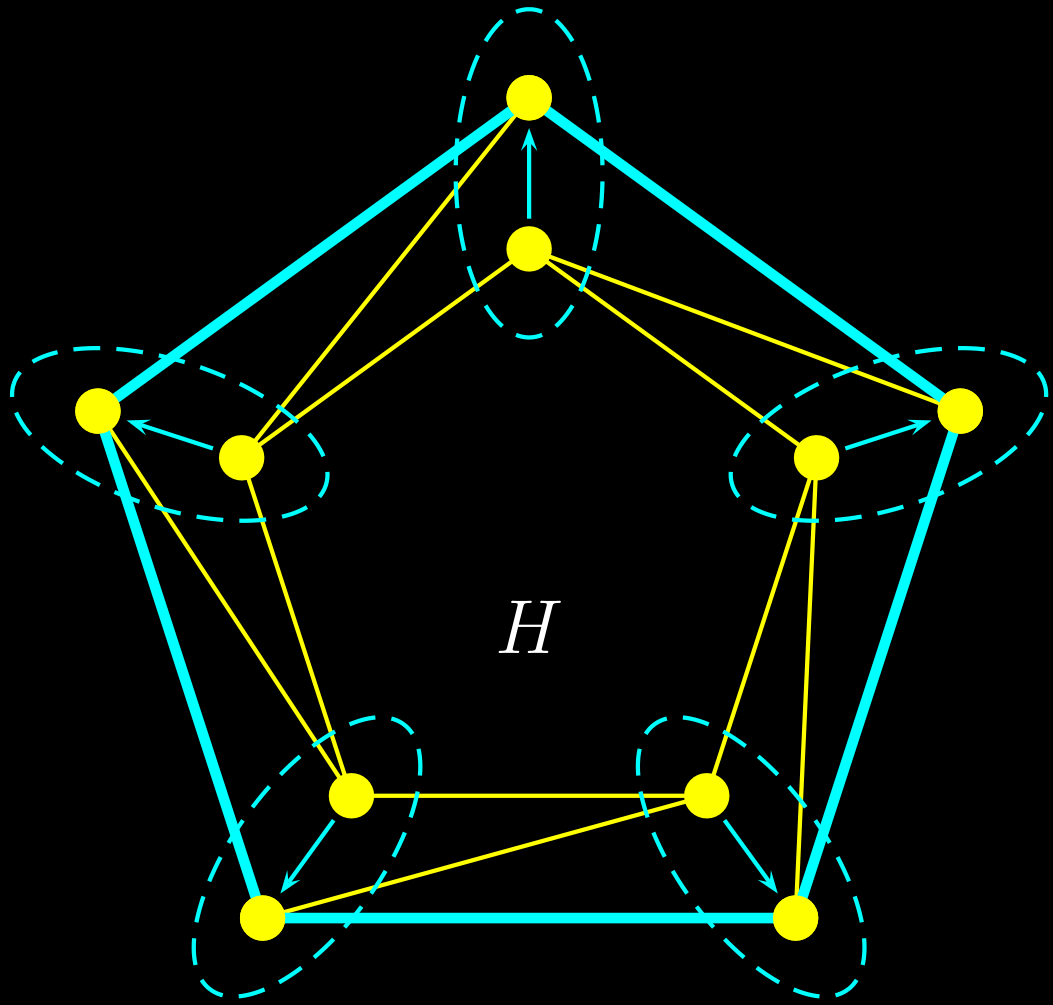
Core examples



Core examples



Core examples

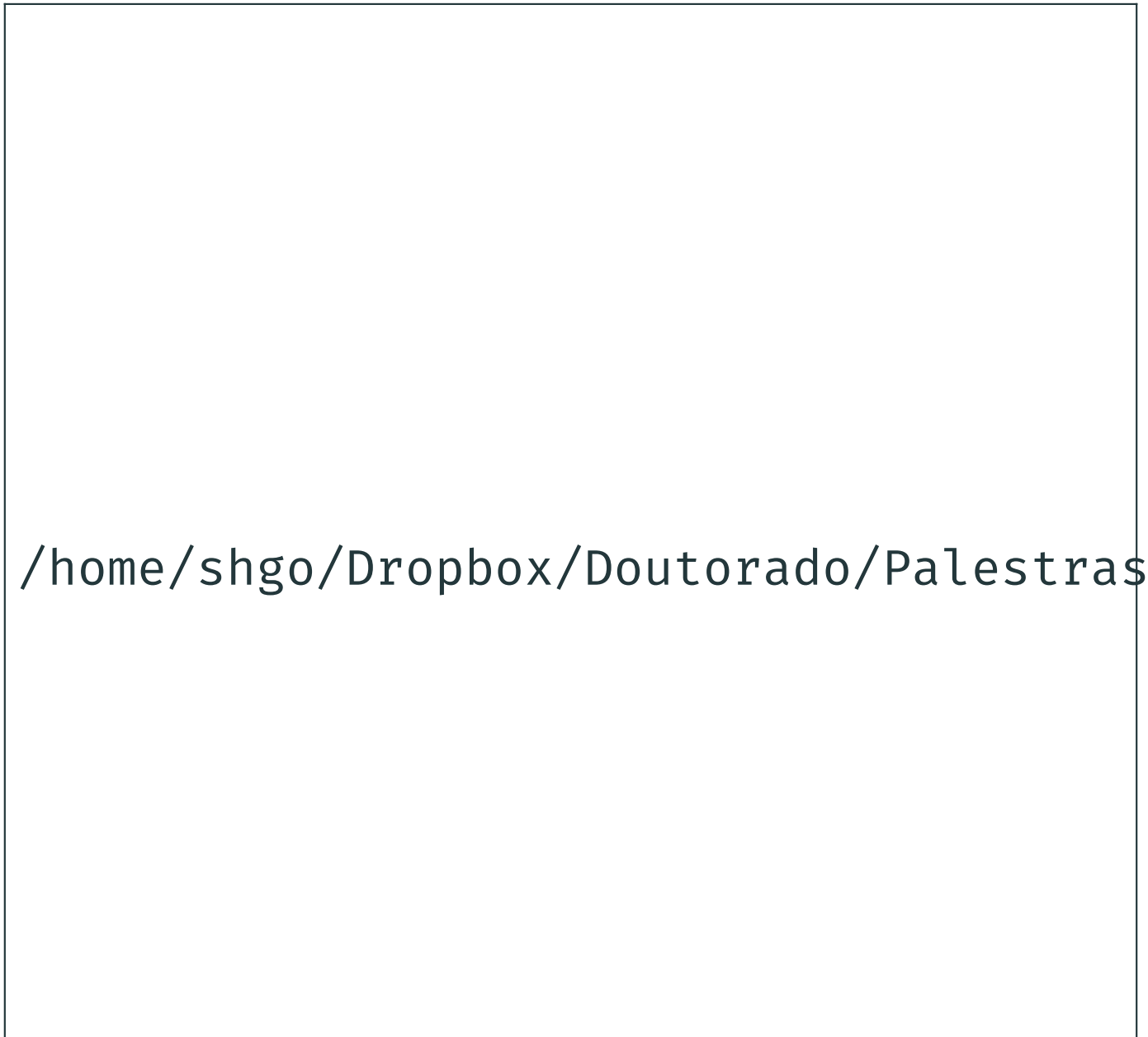


Algorithm 10 Generation of concepts$((A, B), y)$

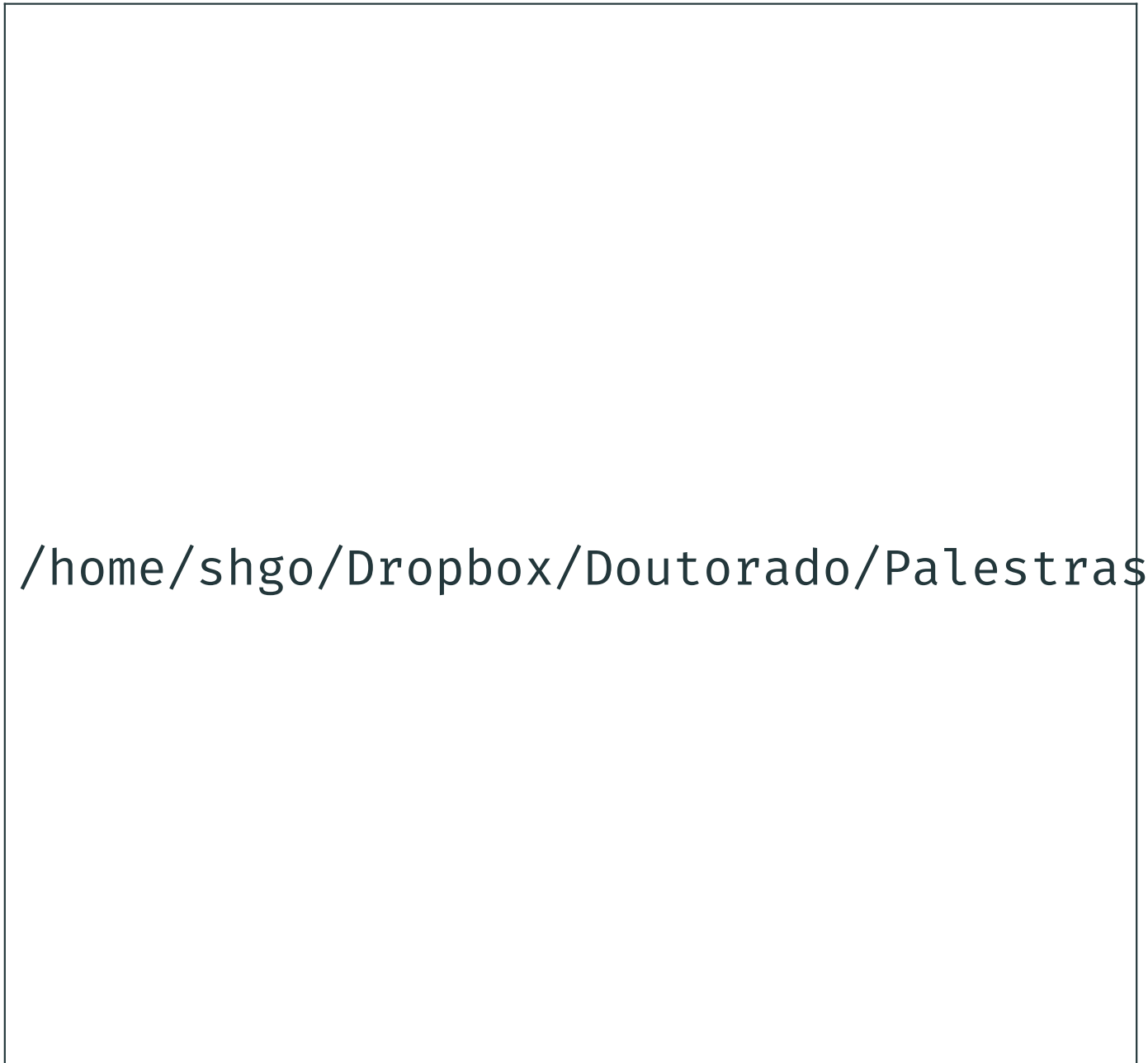
Require: $K = (O, A, I)$ is a graph context

Concepts $\leftarrow \emptyset$
Patterns $\leftarrow \{I\}$
for all $P \in$ *Patterns* **do**
    for all new $P_a \in$ *ConnectedComponents*$(P \cap_q I)$ **do**
        $P_a \leftarrow$ *removeDuplicateNodes*(P_a)
        for all new $P_b \in$ *Retracts*(P_a) **do**
            $X \leftarrow P_b.nodes \setminus \bigcup_{R \in Retracts(P_a) | R \subseteq P_b} R.nodes$
            if $X \neq \emptyset$ **then**
                for all $x \in X$ **do**
                    $Q \leftarrow ((x), P_b); R \leftarrow Q'$
                    Concepts $\leftarrow \{(R, Q) \in P_b\} \cup$ *Concepts*
                end for
                Patterns $\leftarrow \{P_b\} \cup$ *Patterns*
            end if
        end for
    end for
end for

An example



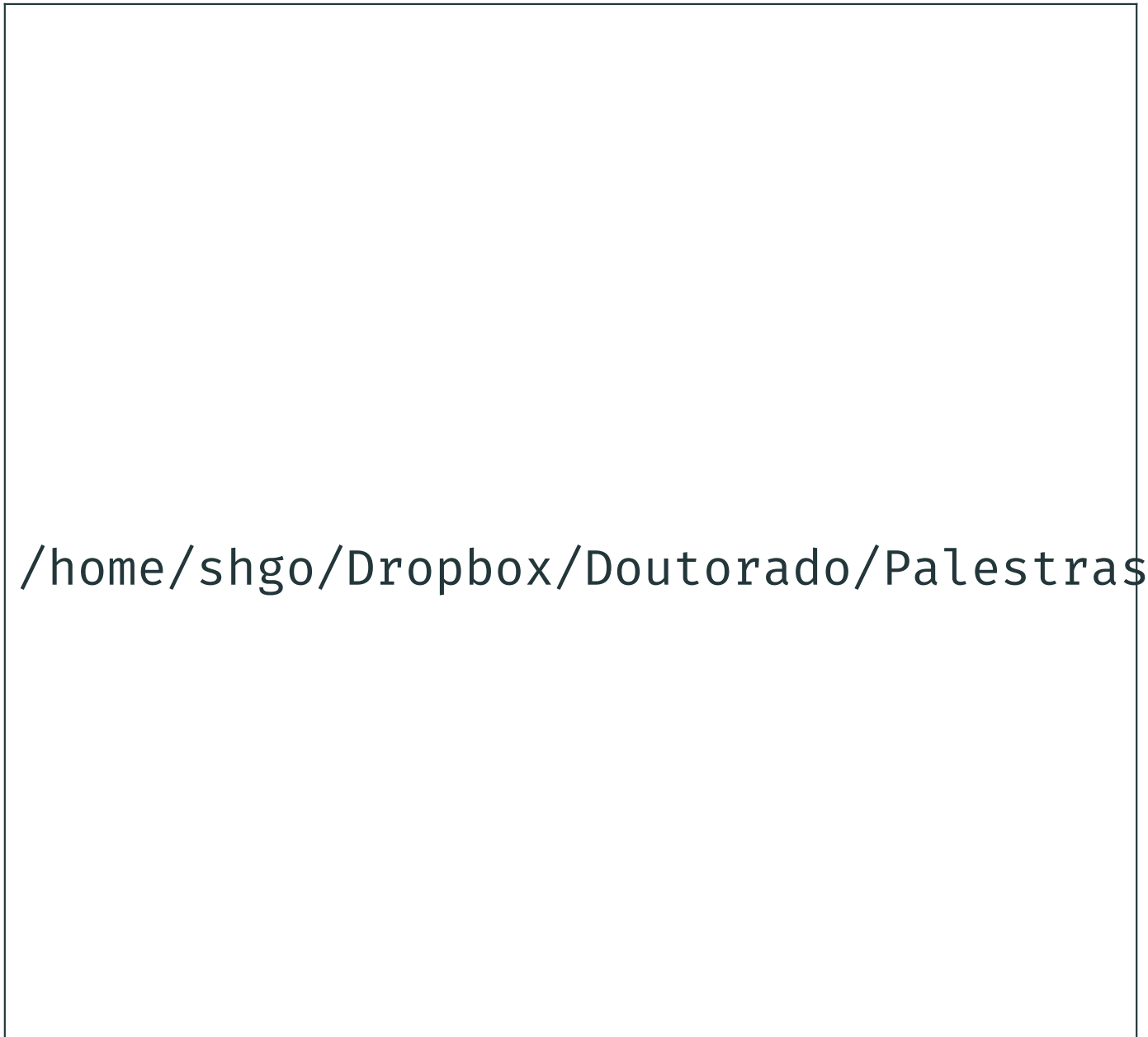
Some Graph Concepts



Experiment

- Chocolate apple pie, strawberry-apple pie, mango-coconut pie and gooseberry pie.
- n-ary relations represent temporal constraints between actions, and entities manipulated by actions. For instance, “put on”: (1) start, (2) end, (3) object, (4) destination.
- Attributes are action descriptions (e.g., “cut”), types of ingredients (e.g., “cream”) or utensils (e.g., “dish”).

Results



Results

- Fig. 17a: ingredients or atomic actions (e.g. “pour on”).
- Fig. 17b: refinements of (a) patterns (e.g. “pour cream on something”).
- Fig. 17c: abstraction of many recipes (e.g. “cut the fruit in order to put it on something, which is put on a dish, which is baked, after preheating the oven, in order to obtain a pie”).

Conclusions

Conclusion

- FCA abstraction can be extended in a creative and rigorous way.
- There are tentatives of generalizing the FCA framework.
- The experiments showed that Graph Patterns can explain sequential behavior and hidden structure in the network.

References

Andrews, Simon (2015). “A ‘Best-of-Breed’ approach for designing a fast algorithm for computing fixpoints of Galois Connections”. In: *Information Sciences* 295, pp. 633 –649. ISSN: 0020-0255. DOI: http://dx.doi.org/10.1016/j.ins.2014.10.011. URL: http://www.sciencedirect.com/science/article/pii/S0020025514009918.

References ii

Brewster, Rick (2006). *Graph Homomorphism Tutorial*. `http://www.site.uottawa.ca/~lucia/CA06/TutorialBrewster.pdf`. Online; accessed 10-August-2016.

References iii

Ferré, Sébastien and Peggy Cellier (2016). “Graph-FCA in Practice”. In: *Graph-Based Representation and Reasoning: 22nd International Conference on Conceptual Structures, ICCS 2016, Annecy, France, July 5-7, 2016, Proceedings*. Ed. by Ollivier Haemmerlé, Gem Stapleton, and Catherine Faron Zucker. Cham: Springer International Publishing, pp. 107–121. ISBN: 978-3-319-40985-6. DOI: 10.1007/978-3-319-40985-6_9. URL: http://dx.doi.org/10.1007/978-3-319-40985-6_9.

References iv

Ferré, Sebastien (2015). “A Proposal for Extending Formal Concept Analysis to Knowledge Graphs”. In: *Formal Concept Analysis: 13th International Conference, ICFCA 2015, Nerja, Spain, June 23-26, 2015, Proceedings*. Springer International Publishing, pp. 271–286. ISBN: 978-3-319-19545-2. DOI: 10.1007/978-3-319-19545-2_17.

Ganter Bernhard; Wille, Rudolf (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-62771-5,978-3-642-59830-2.

References v

KSmrq (2004). *Hasse diagram of powerset of 3*. https://commons.wikimedia.org/wiki/File:Hasse_diagram_of_powerset_of_3.svg/. [Online; accessed 08-August-2016].

Smith, Peter (2010). *The Galois connection between syntax and semantics*. URL: http://www.logicmatters.net/resources/pdfs/Galois.pdf.