

# Faster Concept Analysis

GQ Zhang

Department of Electrical Engineering and Computer Science  
Case Western Reserve University  
Cleveland, Ohio, USA

Joint work with

Adam Troy   Devin Tian

# Motivation

Algorithms for Formal Concept Analysis (FCA) are important for many applications

Kuznetsov and Obiedkov (2002) provide an extensive survey and comparative experimental evaluation of algorithms for FCA in the literature, e.g.

- Bordat
- Chein
- Norris
- Valtchev et al.
- ...

# Key Features of Multistage Concept Analysis (MCA)

- Separates the computation of diagram graph as a different phase so we can take advantage of sophisticated sub-quadratic partial ordering algorithms proposed by Pritchard;
- Maintains a non-redundant set of concepts and uses an auxiliary search tree for quick concept lookup;
- Keeps only the attribute (dually, object) set, or the intent of concepts throughout the algorithm to eliminate the overhead of extent maintenance;
- Forms the concept system by a **multistage** intersection operation from the initial concept set consisting of all object concepts.

# Plan of this talk

- Notation and example
- MCA
- Proof of correctness
- Experimental results
- Prichard's algorithm

## Notation (Ganter & Willer 1999)

Let  $\mathbf{K} = (G, M, I)$  be a formal context. Then its concept lattice  $\mathcal{BK}$  is isomorphic to the closure system generated by the set  $\{\{g\}' \mid g \in G\}$  and dually,  $\mathcal{BK}$  is inverse-isomorphic to the closure system generated by the set  $\{\{m\}' \mid m \in M\}$ .

Object concepts and attribute concepts are called *primitive concepts* in the rest of the talk.

## Example

To illustrate how MCA works, we applied it to the “Living Beings” context from Ganter & Wille. The context is:

	a	b	c	d	e	f	g	h	i
1	×	×					×		
2	×	×					×	×	
3	×	×	×				×	×	
4	×		×				×	×	×
5	×	×		×		×			
6	×	×	×	×		×			
7	×		×	×	×				
8	×		×	×		×			

## Example

We have the following concepts, generated in stages:

Stage 0		
	0	abcdefghi
	1	abg
	2	abgh
	3	abcgh
	4	acghi
	5	abdf
	6	abcdf
	7	acde
	8	acdf

# Example

Stage 0		
	0	abcdefghi
	1	abg
	2	abgh
	3	abcgh
	4	acghi
	5	abdf
	6	abcdf
	7	acde
	8	acdf

Stage 1			
	1	9	ag
		10	ab
		11	a
	2	12	agh
	3	13	acgh
		14	abc
		15	ac
	4		
	5	16	ad
		17	adf
	6	18	acd

# Basic Algorithm

For collections of subsets  $S$  and  $T$ , define

$$S \pitchfork T := \{s \cap t \mid s \in S, t \in T\}.$$

With respect to a given formal context  $(G, M, I)$ , define

$$\begin{aligned} L_0 &:= \{M\} \\ L_1 &:= \{\{g\}' \mid g \in G\} \\ L_i &:= L_{i-1} \pitchfork L_{i-1} \quad (i > 1) \end{aligned}$$

*Proposition* [Chein] With respect to a given formal context  $(G, M, I)$ ,

$$L = \bigcup_{0 \leq i \leq |G|} L_i,$$

where  $L$  is the set of concepts of  $(G, M, I)$ .

# Intuition

Any concept can be obtained as the intersection of a number of primitive concepts.

Rewrite an intersection  $A$  of  $i + 1$  primitive concepts as the pairwise intersection of two concepts  $B$  and  $C$  in  $L_i$  as  $A = B \cap C$ , where  $B$  is the intersection of the first  $i$  primitive concepts used for  $A$ , and  $C$  is the intersection of the primitive concepts omitting the first one:

$$\bigcap_{1 \leq k \leq i+1} \{g_k\}' = \bigcap_{1 \leq k \leq i} \{g_k\}' \cap \bigcap_{2 \leq k \leq i+1} \{g_k\}'.$$

(Then  $B, C \in L_i$  implies  $A \in L_{i+1}$ )

# Multistage Algorithm: too good to be true?

Our MCA algorithm uses a different sequence of sets, defined as follows:

$$\begin{aligned} S_0 &:= \{M\}, \\ S_1 &:= \{\{g\} \mid g \in G\}, \text{ and} \\ S_{i+1} &:= (S_i \cap S_i) - \bigcup_{1 \leq k \leq i} S_k \quad (i > 1) \end{aligned}$$

(In contrast,  $L_j := L_{i-1} \cap L_{i-1}$ )

The distinction from Chain lies in the removal of all existing concepts  $\bigcup_{1 \leq k \leq i} S_k$  when forming  $S_{i+1}$ .

**Correctness.** With respect to a given formal context  $(G, M, I)$ ,

$$L = \bigcup_{0 \leq i \leq |G|} S_i.$$

## ... Yes

“Multistage” does not work in general – example from “resolution theorem proving”. For a set of clauses  $T$ , define

$$\nabla(T) := \{c \mid c \text{ is a resolvent of two clauses in } T\}$$

$$\mathcal{R}^0(S) := S, \quad \mathcal{R}^{i+1}(S) := \mathcal{R}^i(S) \cup \nabla(\mathcal{R}^i(S))$$

**Robinson resolution principle:**  $\{ \} \in \mathcal{R}^*(S)$  iff  $S$  is unsatisfiable.

**Example** that “multistage” does not work:

$$\{\{A, B\}, \{\neg B\}, \{\neg A\}\} \longrightarrow \{\{A\}, \{B\}\}$$

$(\{\{A, B\}, \{\neg B\}, \{\neg A\}\})$  is unsatisfiable, but  $\{ \} \notin \{\{A\}, \{B\}\}$

... No

Let  $(G, M, I)$  be a formal context. Two sets  $X, Y \subseteq G$  are **equivalent** if

$$\cap\{\{g\}' \mid g \in X\} = \cap\{\{g\}' \mid g \in Y\}.$$

When  $X$  and  $Y$  are equivalent, we write  $X \equiv Y$ .

A subset of  $G$  is **irreducible** if it is not equivalent to any of its proper subsets.

## Proof of Correctness: Setup and Intuition

It suffices to show by (course of value) induction in  $i$  that

if  $|X| = i$  and  $X$  is **irreducible**,

then  $n\{\{g\}' \mid g \in X\}$  belongs to  $S_t$ , where  $t = 1 + \lceil \log_2 i \rceil$ .

## Proof of Correctness: Setup and Intuition

It suffices to show by (course of value) induction in  $i$  that

if  $|X| = i$  and  $X$  is **irreducible**,

then  $n\{\{g\}' \mid g \in X\}$  belongs to  $S_t$ , where  $t = 1 + \lceil \log_2 i \rceil$ .

$i$	0	1	2	3	4	5	6	7	8	9	10	.	16	17
$t$	0	1	2	3	3	4	4	4	4	5	5	.	5	6

# Proof of Correctness: Setup and Intuition

It suffices to show by (course of value) induction in  $i$  that

if  $|X| = i$  and  $X$  is **irreducible**,

then  $\cap\{\{g\}' \mid g \in X\}$  belongs to  $S_t$ , where  $t = 1 + \lceil \log_2 i \rceil$ .

$i$	0	1	2	3	4	5	6	7	8	9	10	.	16	17
$t$	0	1	2	3	3	4	4	4	4	5	5	.	5	6

$$\begin{aligned} 17 &=_{1} 8 + 9 \\ &=_{2} (4 + 4) + (4 + 5) \\ &=_{3} ((2 + 2) + (2 + 2)) + ((2 + 2) + (2 + 3)) \\ &=_{4} (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 1))) + \\ &\quad (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 2))) \\ &=_{5} (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 1))) + \\ &\quad (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + (1 + 1)))) \end{aligned}$$

# Proof of Correctness: Setup and Intuition

It suffices to show by (course of value) induction in  $i$  that

if  $|X| = i$  and  $X$  is **irreducible**,

then  $\cap\{\{g\}' \mid g \in X\}$  belongs to  $S_t$ , where  $t = 1 + \lceil \log_2 i \rceil$ .

$i$	0	1	2	3	4	5	6	7	8	9	10	.	16	17
$t$	0	1	2	3	3	4	4	4	4	5	5	.	5	6

$$\begin{aligned} 17 &=_{1} 8 + 9 \\ &=_{2} (4 + 4) + (4 + 5) \\ &=_{3} ((2 + 2) + (2 + 2)) + ((2 + 2) + (2 + 3)) \\ &=_{4} (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 1))) + \\ &\quad (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 2))) \\ &=_{5} (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + 1))) + \\ &\quad (((1 + 1) + (1 + 1)) + ((1 + 1) + (1 + (1 + 1)))) \end{aligned}$$

If  $|X| = 17$ , then  $\cap\{\{g\}' \mid g \in X\} \in S_6$ .

## Proof of Correctness: Inductive Step ( $k = 17$ )

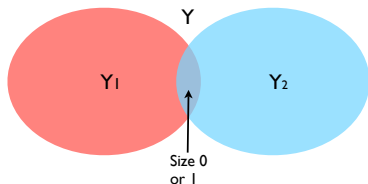
Assume that for some  $17 > 0$ , all irreducible subsets  $X$  with  $|X| = j \leq 17$  determine concepts  $\cap\{\{g\}' \mid g \in X\}$  belonging to  $S_{1+\lceil \log_2 j \rceil}$ .

Show that for an irreducible set  $Y$  with  $|Y| = 17 + 1$ ,  $\cap\{\{g\}' \mid g \in Y\}$  belongs to  $S_6 = S_{1+\lceil \log_2 (17+1) \rceil}$ .

Let  $Y \subseteq G$  be an irreducible set with  $|Y| = 17 + 1$ . We have

$$\cap\{\{g\}' \mid g \in Y\} = \cap\{\{g\}' \mid g \in Y_1\} \cap \cap\{\{g\}' \mid g \in Y_2\},$$

where  $Y_1, Y_2$  are subsets of  $Y$  with sizes  $\leq \lceil (17 + 1)/2 \rceil = 9$ , and  $|Y_1 \cap Y_2| \leq 1$ , and  $Y = Y_1 \cup Y_2$ .



## Proof of Correctness: Two Subcases

When  $17 + 1$  is even,  $Y_1$  and  $Y_2$  are themselves irreducible. We have, by induction hypothesis,  $\{\{g\}' | g \in Y_i\} \in \mathcal{S}_{1+\lceil \log_2(k+1)/2 \rceil} = \mathcal{S}_5$  for  $i = 1, 2$ . Therefore,  $\cap\{\{g\}' | g \in Y\} \in \mathcal{S}_{1+\lceil \log_2(k+1) \rceil} = \mathcal{S}_6$  by the definition of  $\mathcal{S}_i$ .

## Proof of Correctness: Two Subcases

When  $17 + 1$  is even,  $Y_1$  and  $Y_2$  are themselves irreducible. We have, by induction hypothesis,  $\{\{g\}' | g \in Y_i\} \in \mathcal{S}_{1+\lceil \log_2(k+1)/2 \rceil} = \mathcal{S}_5$  for  $i = 1, 2$ . Therefore,  $\cap\{\{g\}' | g \in Y\} \in \mathcal{S}_{1+\lceil \log_2(k+1) \rceil} = \mathcal{S}_6$  by the definition of  $\mathcal{S}_i$ .

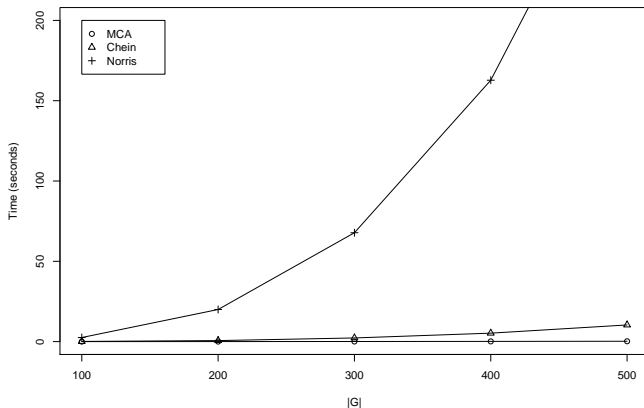
When  $k + 1$  is odd,  $Y_1$  and  $Y_2$  must also be irreducible since any subset of an irreducible set is irreducible. By induction hypothesis, we have  $\{\{g\}' | g \in Y_i\} \in \mathcal{S}_{1+\lceil \log_2(k+2)/2 \rceil}$  for  $i = 1, 2$ .

However,  $\lceil \log_2(k + 2) \rceil = \lceil \log_2(k + 1) \rceil$  when  $k$  is even. Therefore,  $\cap\{\{g\}' | g \in Y\} \in \mathcal{S}_{1+\lceil \log_2(k+1) \rceil}$  again.

# Comparative Experiment I

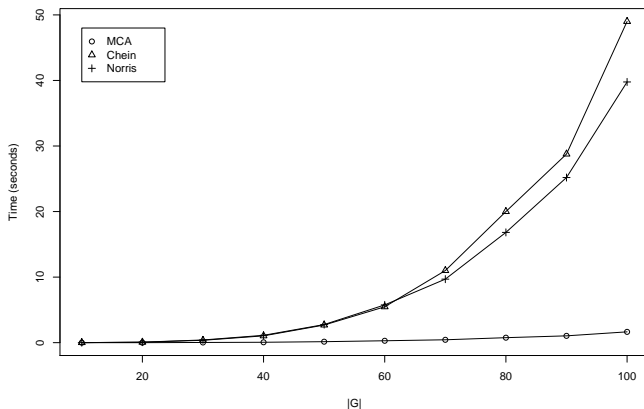
(Ref. Kuznetsov & Obiedkov 2002)

Comparison of time to compute concept set only – MCA vs. Chein and Norris for contexts with  $|M| = 100$ ,  $|g'| = 4$ :



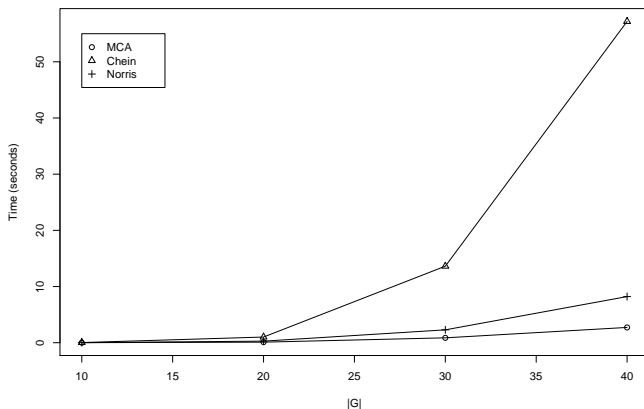
## Comparative Experiment II

Comparison of time to compute concept set only – MCA vs. Chein and Norris for contexts with  $|M| = 100$ ,  $|g'| = 15$ :



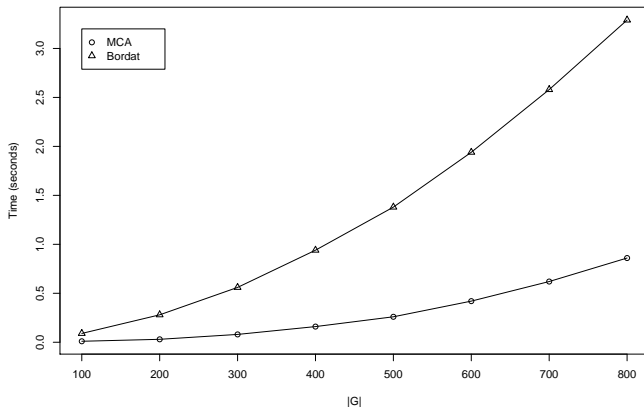
## Comparative Experiment III

Comparison of time to compute concept set only – MCA vs. Chein and Norris for contexts with  $|M| = 100$ ,  $|g'| = 30$ :



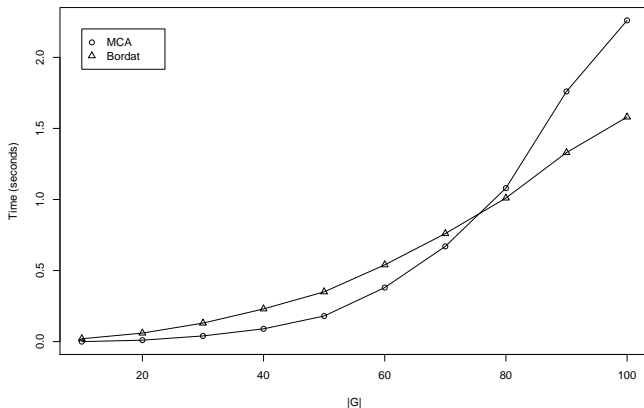
## Comparative Experiment IV

Comparison of time to compute concept set and lattice diagram – MCA+Pritchard vs. Bordat for contexts with  $|M| = 100$ ,  $|g'| = 4$  and  $|G|$  between 100–800.



# Comparative Experiment V

Comparison of time to compute concept set and lattice diagram – MCA+Pritchard vs. Bordat for contexts with  $|M| = 100$ ,  $|g'| = 15$  and  $|G|$  between 10–100.



# Pritchard's Algorithm

Given a collection  $\mathcal{F} = \{S_1, S_2, \dots, S_k\}$ , where  $S_i \subseteq D$  for a fixed set  $D$ , compute the *complete subset graph*.

For any subset  $y$  of  $D$ , let

$$\mathcal{F}.y := \{x \in \mathcal{F} \mid y \subseteq x\}.$$

Then  $\mathcal{F}.d$  stands for the sub-collection of all subsets in  $\mathcal{F}$  containing  $d$ , and  $x \in \mathcal{F}.y$  iff  $y \subseteq x$  iff  $x \in \bigcap_{d \in y} \mathcal{F}.d$ . Therefore,

*Lemma* [Pritchard] For any  $y \in \mathcal{F}$ ,  $\mathcal{F}.y = \bigcap_{d \in y} \mathcal{F}.d$ .

Hence, the intersection  $\bigcap_{d \in y} \mathcal{F}.d$  contains all supersets of  $y$ .

# Pritchard's Algorithm: Example

Context:

	female	child	adult	male
girl	x	x		
woman	x		x	
boy		x		x
man			x	x

Step 1: Build (attribute  $\rightarrow$  concept) membership sets:

- .female: B, D, G, J
- .child: A, B, E, J
- .adult: C, D, F, J
- .male: A, C, H, J

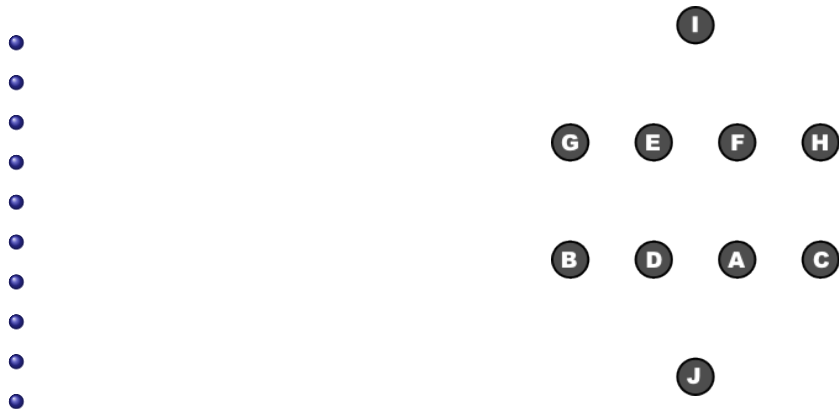
upper neighbors of  $A = \{child, male\}$ :  
.child  $\cap$  .male =  $\{A, J\}$

Concepts:

- A- boy: child, male
- B- girl: child, female
- C- man: adult, male
- D- woman: adult, female
- E- boy, girl: child
- F- man, woman: adult
- G- girl, woman: female
- H- boy, man: male
- I- boy, girl, man, woman:
- J- : child, male, adult, female

## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:



## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$$.child \cap .male = \{A, B, E, J\} \cap \{A, C, H, J\}$$

●  $P_A$ - A, J

●

●

●

●

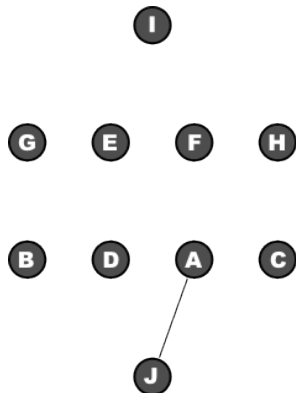
●

●

●

●

●



# Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$$.child \cap .female = \{A, B, E, J\} \cap \{B, D, G, J\}$$

●  $P_A$ - A, J

●  $P_B$ - B, J

●

●

●

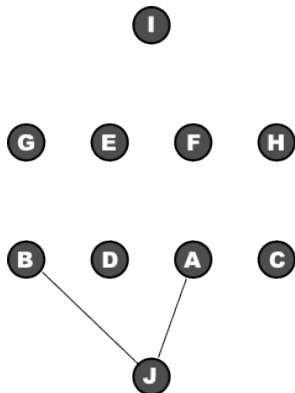
●

●

●

●

●



## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$$.adult \cap .male = \{C, D, F, J\} \cap \{A, C, H, J\}$$

●  $P_A$ - A, J

●  $P_B$ - B, J

●  $P_C$ - C, J

●

●

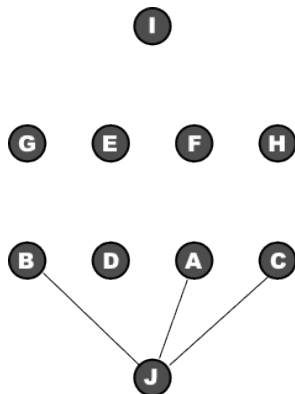
●

●

●

●

●



# Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$$.adult \cap .female = \{C, D, F, J\} \cap \{B, D, G, J\}$$

●  $P_A$ - A, J

●  $P_B$ - B, J

●  $P_C$ - C, J

●  $P_D$ - D, J

●

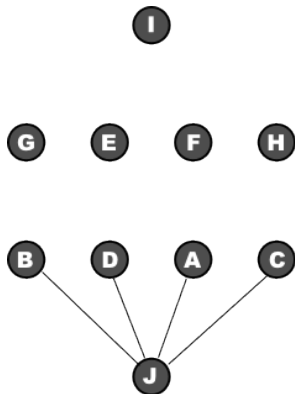
●

●

●

●

●



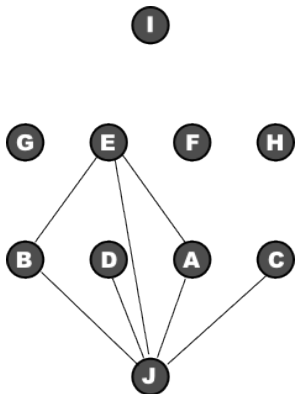
## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$.child = \{A, B, E, J\}$

- $P_A - A, J$
- $P_B - B, J$
- $P_C - C, J$
- $P_D - D, J$
- $P_E - A, B, E, J$

- 
- 
- 
- 
- 
- 

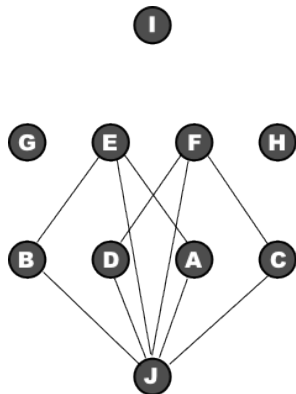


## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$.adult = \{C, D, F, J\}$

- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- 
- 
- 
- 

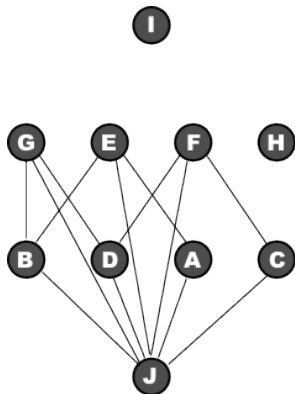


## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$.female = \{B, D, G, J\}$

- $P_A - A, J$
- $P_B - B, J$
- $P_C - C, J$
- $P_D - D, J$
- $P_E - A, B, E, J$
- $P_F - C, D, F, J$
- $P_G - B, D, G, J$
- 
- 
- 

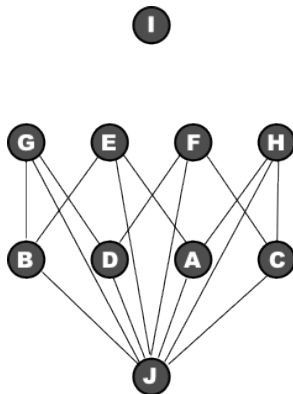


# Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$.male = \{A, C, H, J\}$

- $P_A - A, J$
- $P_B - B, J$
- $P_C - C, J$
- $P_D - D, J$
- $P_E - A, B, E, J$
- $P_F - C, D, F, J$
- $P_G - B, D, G, J$
- $P_H - A, C, H, J$
- 
- 

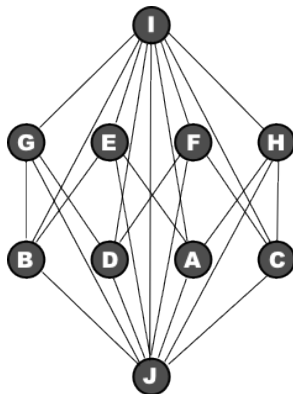


## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$\emptyset = *$

- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- $P_G$ - B, D, G, J
- $P_H$ - A, C, H, J
- $P_I$ - A, B, C, D, E, F, H, I, J
- 

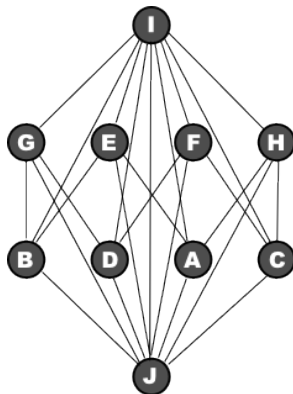


## Pritchard: Example

Step 2: Identify ancestors by computing intersections ( $P_i$ ) of (attribute  $\rightarrow$  concept) sets for the attributes of each concept A-J:

$$* = \{A, B, E, J\} \cap \{A, C, H, J\} \cap \{C, D, F, J\} \cap \{B, D, G, J\} = \emptyset$$

- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- $P_G$ - B, D, G, J
- $P_H$ - A, C, H, J
- $P_I$ - A, B, C, D, E, F, H, I, J
- $P_J$ - J



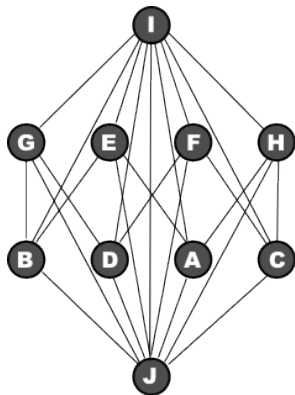


## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

- $P_J - J$
- $P_A - A, J$

- 
- 
- 
- 
- 
- 
- 
- 
- 

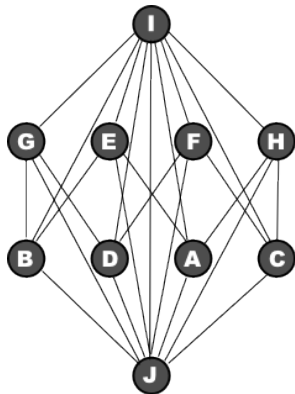


## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J

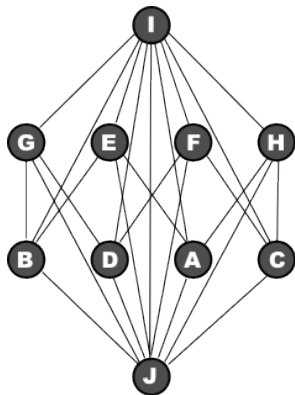
- 
- 
- 
- 
- 
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

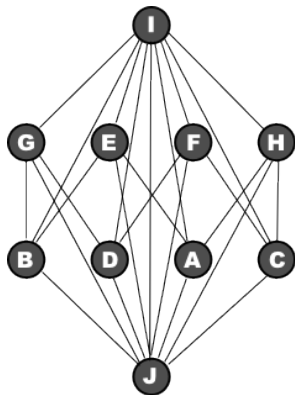
- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- 
- 
- 
- 
- 
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

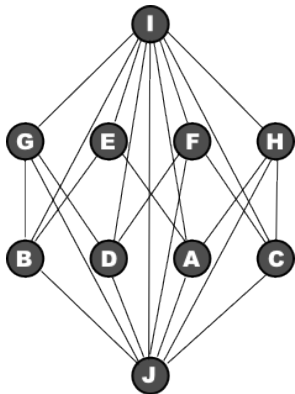
- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- 
- 
- 
- 
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

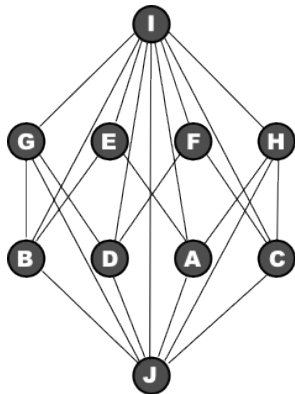
- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- 
- 
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

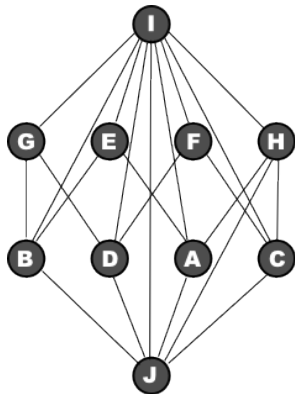
- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- 
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

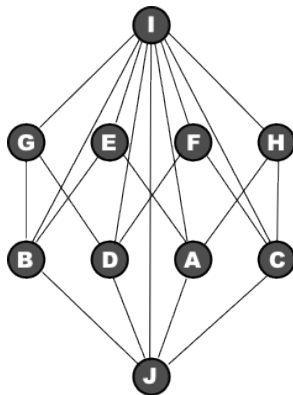
- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- $P_G$ - B, D, G, J
- 
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- $P_G$ - B, D, G, J
- $P_H$ - A, C, H, J
- 



## Pritchard: Example

Step 3: Remove **grandparents** (etc.) from ancestor lists by calculating the complement of each parent set with the union of the parent sets of its ancestors (and removing **self links**), in order of increasing set size:

- $P_J$ - J
- $P_A$ - A, J
- $P_B$ - B, J
- $P_C$ - C, J
- $P_D$ - D, J
- $P_E$ - A, B, E, J
- $P_F$ - C, D, F, J
- $P_G$ - B, D, G, J
- $P_H$ - A, C, H, J
- $P_I$ - A, B, C, D, E, F, G, H, I, J

