An Answer Set Programming environment for high-level specification and visualization of FCA

Lucas Bourneuf

Univ Rennes, France

13 july 2018

Outline

Most programs target a use case

LatViz

Efficient exploration of Galois lattices

FCA Tools Bundle

Web interface for contexts and (ternary) concept lattices exploration

In-Close

Fast concept miner

Each program implements and let user explore a data model

From the point of view of users

A user problem is either solved by:

1. An existing tool
2. A *variant* or a *combination* of existing methods
   - the tool do not (yet) exists
   - need development effort

Development effort **beyond specification** necessary in most cases

Complementary approach: let users define the model

- work on the model, instead of data
- do not target efficiency, but flexibility
- leave room for future optimizations

Specifications as mathematical relations

- most frameworks are defined that way

Get results as graphs

- graph are the most fundamental data structure

Data model prototyping using high-level language and high-level results

Conception with logic programming and graph

Answer Set Programming

- logic programming
- implementation close to specifications

Dot

- graph description language
- high-level output visualizations

Biseau: a proof of concept

- ASP to dot compiler: Write ASP, get graphs
- the user's aim is the proper design of a general model
- data are only support to the model validity
- `https://huit.re/biseau`

Conception with logic programming and graph

Answer Set Programming

- logic programming
- implementation close to specifications

Dot

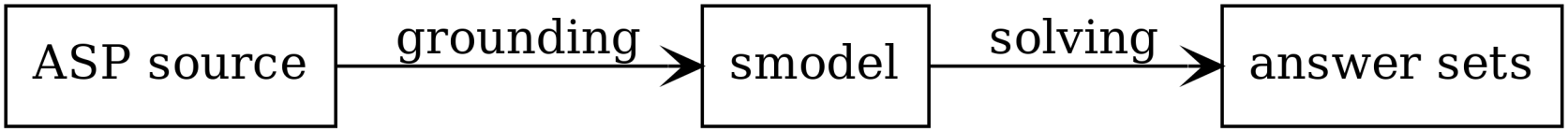
- graph description language
- high-level output visualizations

Biseau: a proof of concept

- ASP to dot compiler: Write ASP, get graphs
- the user's aim is the proper design of a general model
- data are only support to the model validity
- `https://huit.re/biseau`

Outline

Answer Set Programming



Fact

```
a(1). b(2). c(1,2).
```

Rule & variable

```
a(X):- b(X).
```
$\Rightarrow$ a(2)

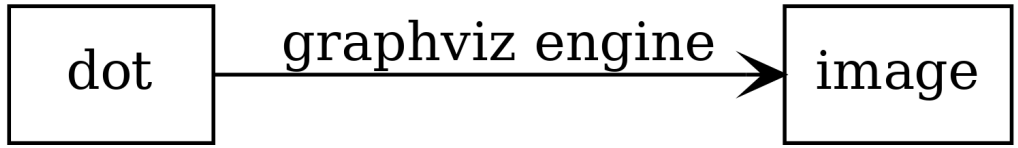
Conjunction & negation: $a \wedge \neg b$

```
p(X):- a(X) ; not b(X).
```
$\Rightarrow$ p(1)

Implication: $b(X) : a(X)$ *holds if* $a(X) \implies b(X) \quad \forall X$

```
q(X):- X=1..3 ; b(X): a(X).
```
$\Rightarrow$ q(2) q(3)

Dot (in one slide)

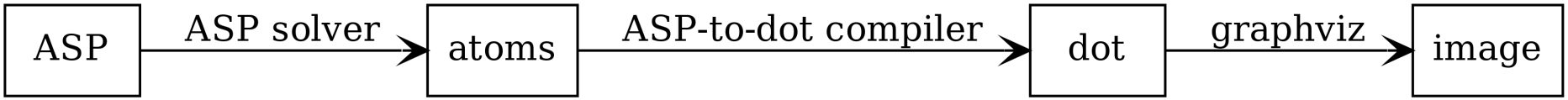


- Graph description language
- Defined by the Graphviz software
- cf `https://www.graphviz.org/`

```
Digraph graph {
        graph [dpi="400" rankdir="LR"];
        node [shape="rectangle" fillcolor="white"];
        edge [arrowhead="vee"];
        dot->image [label="graphviz engine"]
}
```

Drawing graph by description of their content

Biseau: principle




```
link("ASP",atoms). link(atoms,dot). link(dot,image).
label("ASP",atoms,"ASP solver").
label(atoms,dot,"ASP-to-dot compiler").
label(dot,image,graphviz).
obj_property(node,fillcolor,white).
obj_property(node,shape,rectangle).
obj_property(edge,arrowhead,vee).
obj_property(graph,rankdir,"LR").
```

```
Digraph biseau_graph {
        node [shape="rectangle" fillcolor="white"];
        graph [dpi="400" rankdir="LR"];
        edge [arrowhead="vee"];
        "ASP"->atoms [label="ASP solver"]
        atoms->dot [label="ASP-to-dot compiler"]
        dot->image [label="graphviz"]
}
```

Outline

Formal context encoding in ASP

	adult	child	female	male	boy	woman	man
alice			×				
bob	×			×			×
eve	×		×			×	
john		×		×	×		

Can be encoded as rel/2 atoms:

```
rel(alice,female). rel(bob,adult). rel(eve,adult). [...]
```

From standard format to ASP encoding

Formal concepts mining with ASP

Formal concept (A, B) over **objects** X and **attributes** Y:

$$A = \{x \in X \mid r(x, b)\ \forall b \in B\}$$
$$B = \{y \in Y \mid r(a, y)\ \forall a \in A\}$$

In ASP, when atoms rel/2 describes the context:

```
ext(X):- rel(X,_) ; rel(X,Y): int(Y).
int(Y):- rel(_,Y) ; rel(X,Y): ext(X).
```

ASP enables close-to-specification encoding

Computing the Galois lattice

Formal concepts (A_n, B_n) are ordered:

$$(A_1, B_1) < (A_2, B_2) \Leftrightarrow A_1 \subset A_2$$

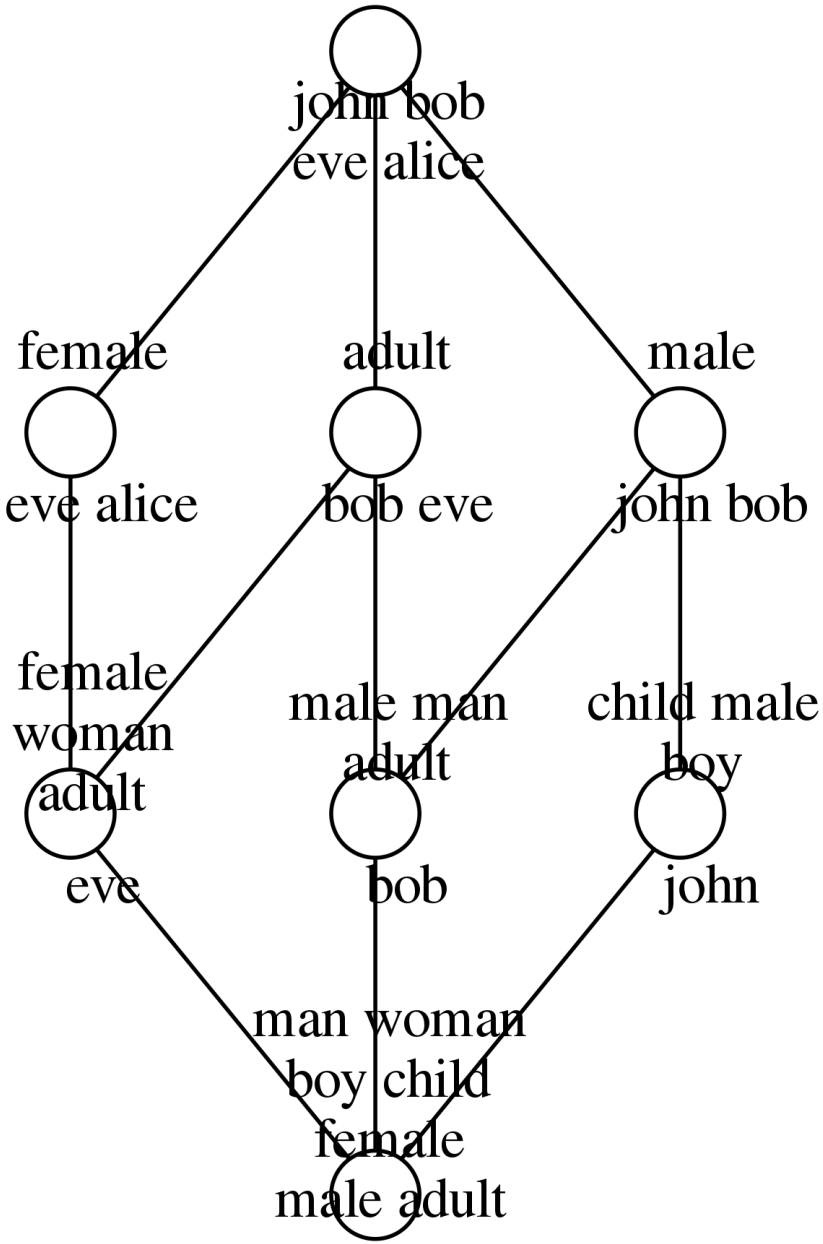
And they are linked to their greatest subconcept.

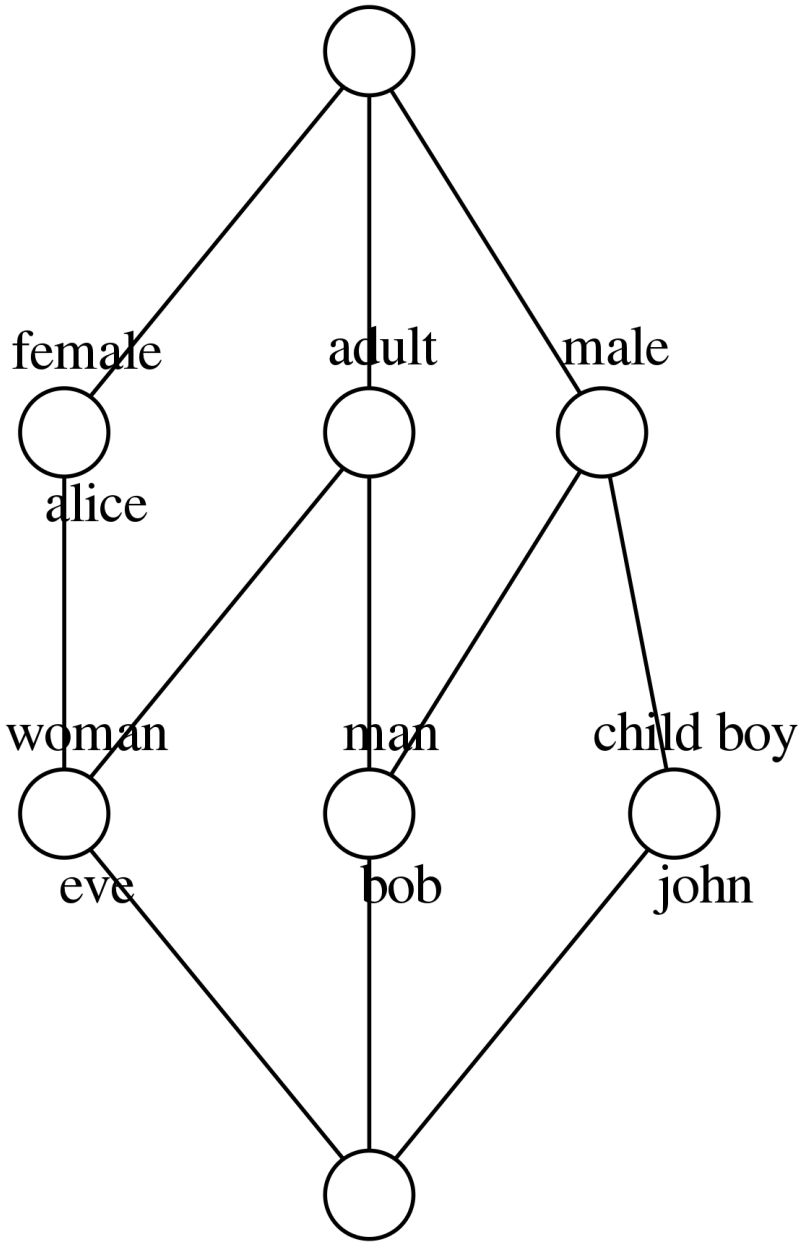
A possible encoding in ASP:

```
% Ordering of two concepts
contains(C1,C2):- c(C1) ; c(C2) ; C1!=C2 ;
                  ext(C1,X): ext(C2,X).
% Concepts linked to another.
link(C1,C3):- contains(C1,C3) ;
              not link(C1,C2): contains(C2,C3).
% Annotate nodes with extent and intent.
annot(upper,X,A):- ext(X,A).
annot(lower,X,B):- int(X,B).
```

ASP enables close-to-specification encoding

Resulting Galois lattices





Outline

3-adic FCA

and obvious generalization to n-adic

Extensions to n-dimensional data requires new concept miners.

When atoms rel/3 describes the context:

```
ext(X):- rel(X,_,_) ; rel(X,A,C): int(A), cnd(C).
int(X):- rel(_,X,_) ; rel(O,X,C): ext(O), cnd(C).
cnd(X):- rel(_,_,X) ; rel(O,A,X): ext(O), int(A).
```

Need adaptation of code for visualizations

Object and Attribute-oriented lattices

Object oriented concepts (X, Y) defined by $X = Y^{\Diamond}$ and $Y = X^{\Box}$:

$$Y^{\Diamond} = \bigcup_{y \in Y} Ry \qquad X^{\Box} = \{y \in A | Ry \subseteq X\}$$

With $Ry = \{x \in O | (x, y) \in R\}$.

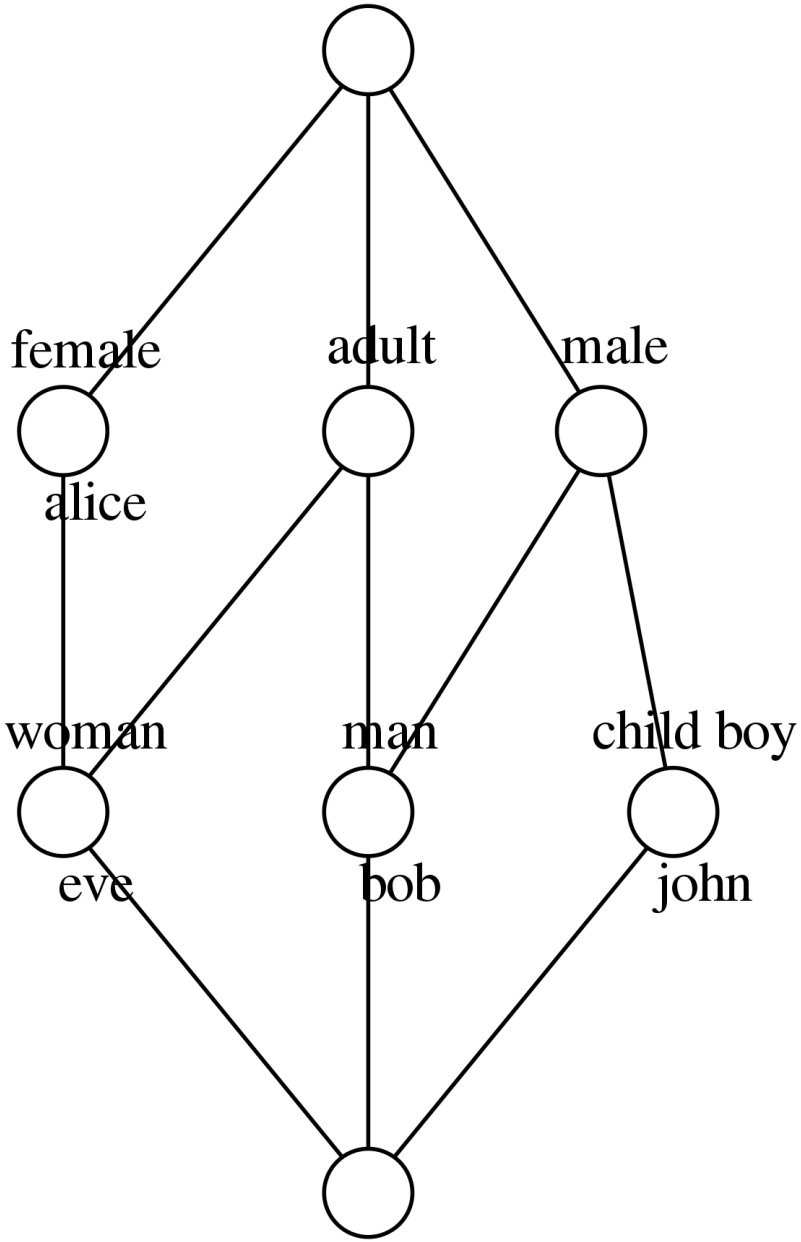
Mining of the object-oriented concepts:

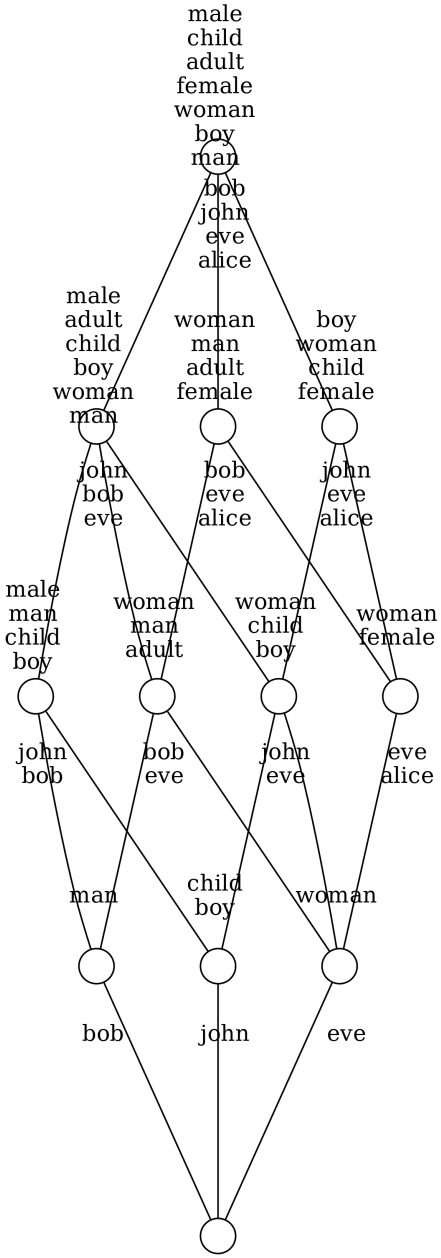
```
% An object linked to an attribute
%  in the intent is in the extent
ext(X):- rel(X,Y) ; int(Y).
% Objects in the complementary set of the extent
not_ext(Nx):- rel(Nx,_) ; not ext(Nx).
% The intent is made of attributes
%  exclusively linked to objects of the extent
int(Y):- rel(_,Y) ; not rel(Nx,Y): not_ext(Nx).
```

Reuse the same Galois lattice generator code

Resulting lattice

With the Galois lattice encoded in Biseau





Iceberg lattice

The Galois lattice stripped of all concepts with a support $<$ minimal.

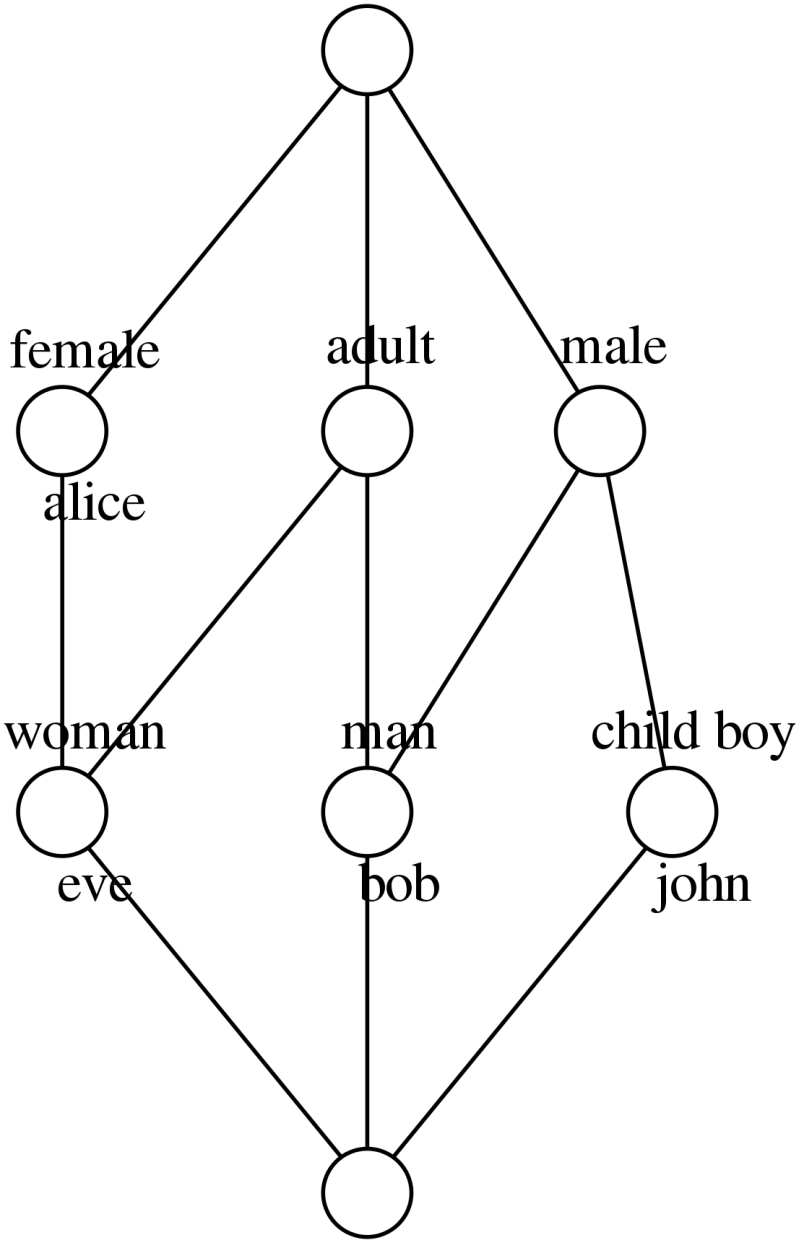
Add the following constraint to discard unwanted concepts:

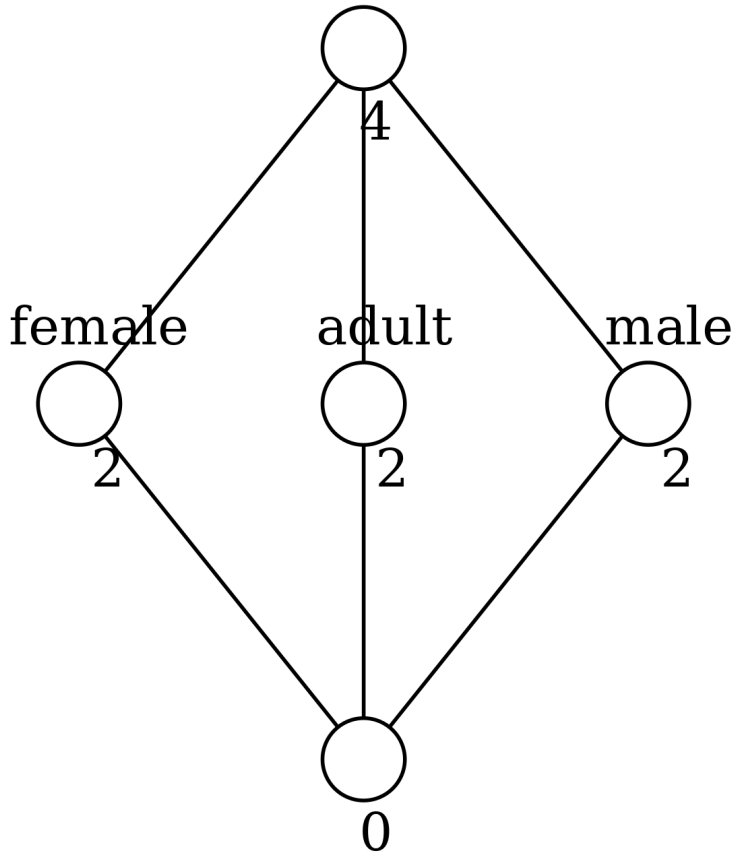
```
:- {ext(_)} < nbobj*minsupp/100.
```

Reuse the same Galois lattice generator code

Resulting lattice

With the Galois encoded in Biseau





Build FCA extension with Biseau

- Other extensions
  - Integer pattern structure
  - Relational concepts
- An extension is a data model
  - Replacing or adding to existing parts

Designing data model by writing ASP

Outline

Discussion: ASP

- High-level language
  - easy encoding of relations
  - malleable, extendable
- Limitations
  - hard to learn (easy to master)
  - scaling problems (total grounding of space search)
  - missing types handling
- A feature-rich language
  - interface to other paradigms
    - imperative (C/python)
    - ILP (cplex)
  - optimizations, heuristic control, propagators
  - Fixed parts can be replaced by other languages/programs

Efficient for prototyping ; extendable ; replaceable

Discussion: Biseau

- A successful experience
  - Simple designing of graph
  - Reproduction of complex models
- Limitations
  - No IDE-like feature to help writing code
  - GUI is too simple
- Scripts
  - Units of code
  - To reproduce all results of the paper
  - To distribute your own model
- Future developments
  - Support for other outputs formats (e.g. GML)
  - Scripts for other domains
  - GUI, CLI, notebook

Conclusion

- **Write ASP**
  - close to specification
  - rich interface
- **Get graphs**
  - universal data structure
  - fully customizable (dot)
- **FCA reconstruction**
  - basics (context, concept mining, Galois lattice)
  - extensions (iceberg, ternary,... integer pattern structure)
- **Other applications**
  - More FCA extensions
  - Other fields: semantic web, bioinformatics
  - ASP extensions

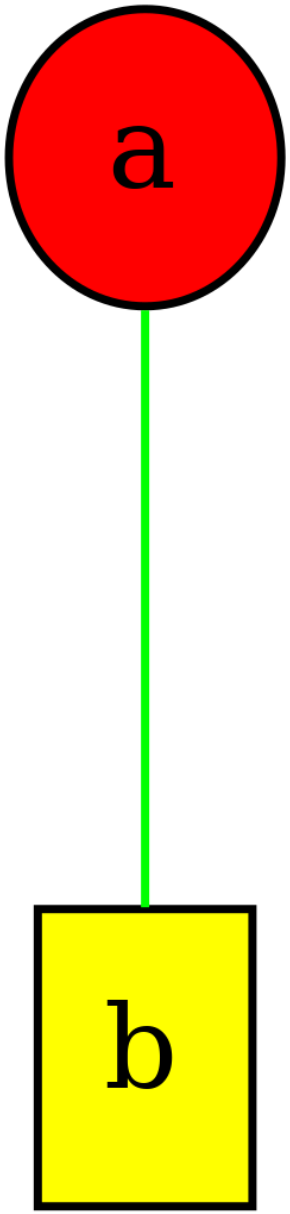
Want to use Biseau ? To participate ?
See `https://huit.re/biseau`
And contact me at `lucas.bourneuf@inria.fr`

Appendix

Styling with dot

```
link(a,b).
color(a,b,green).
color(a,red).
shape(b,rectangle).
```

```
Digraph graph {
        node [shape=ellipse]
        edge [arrowhead=none]
        a [fillcolor="red"]
        b [shape="rectangle"]
        a->b [color="green"]
}
```

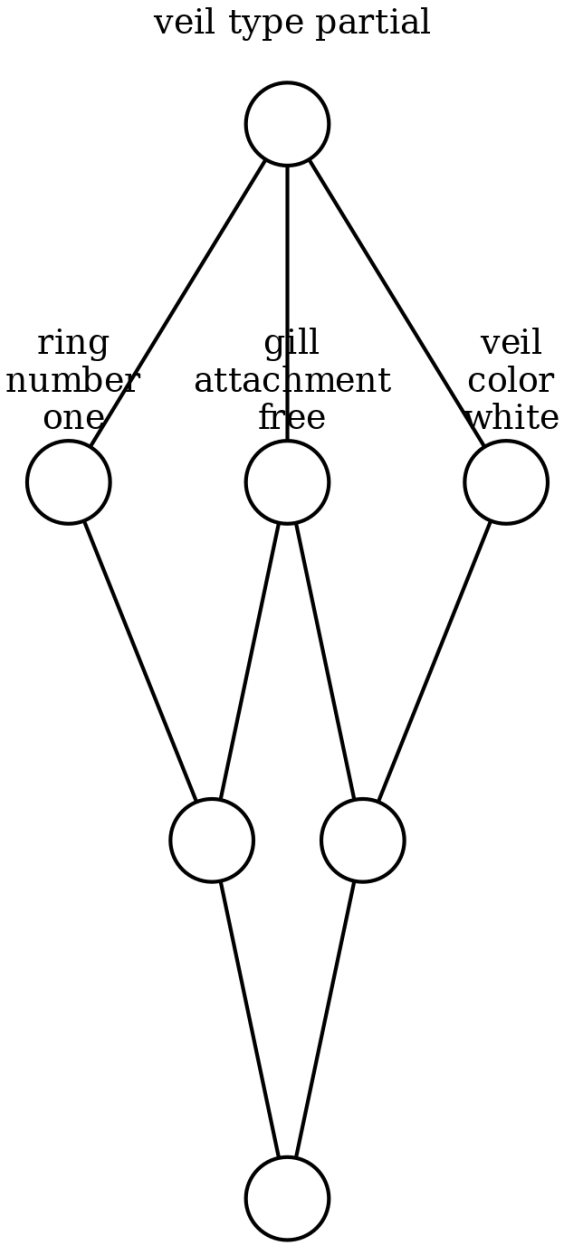


Formal context computation

```
% Facts.
age(john,7). age(eve,71). age(alice,15).
male(john). male(bob). female(alice).
mother(eve,bob).
% Rules.
rel(H,child) :- age(H,A) ; A<12.
rel(H,adult) :- age(H,A) ; A>=18.
rel(H,male)  :- male(H).
rel(H,female):- female(H).
rel(H,man)   :- rel(H,male) ; rel(H,adult).
rel(H,boy)   :- rel(H,male) ; rel(H,child).
rel(H,woman) :- rel(H,female) ; rel(H,adult).
rel(H,girl)  :- rel(H,female) ; rel(H,child).
rel(H,adult) :- rel(H,male) ; not rel(H,boy).
rel(H,female):- mother(H,_).
```

initial extraction of data can also be done in ASP

Other resulting lattices



Formal concepts mining

definition

	h	i	j	k	l	m	n
a		×	×	×	×	×	
b	×		×	×			×
c	×	×			×	×	×
d	×	×			×	×	×
e	×		×	×			×
f	×		×	×			×
g		×	×	×	×	×	

Formal concept (A, B)
over **objects** X and **attributes** Y:

$$A = \{x \in X \mid \forall b \in B,\ r(x, b)\}$$
$$B = \{y \in Y \mid \forall a \in A,\ r(a, y)\}$$

Concepts examples: $\{b, e, f\} \times \{h, j, k, n\}$, $\{a, c, d, g\} \times \{i, l, m\}$