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

Lucas Bourneuf

Univ Rennes, France

13 july 2018
Outline

Motivation

Answer Set Programming to dot with Biseau

Reconstruct FCA basics

Build FCA extensions

Discussion & conclusion
Most programs target a use case

<table>
<thead>
<tr>
<th>LatViz</th>
<th>Efficient exploration of Galois lattices</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCA Tools Bundle</td>
<td>Web interface for contexts and (ternary) concept lattices exploration</td>
</tr>
<tr>
<td>In-Close</td>
<td>Fast concept miner</td>
</tr>
</tbody>
</table>

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](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](https://huit.re/biseau)
Outline

Motivation

Answer Set Programming to dot with Biseau

Reconstruct FCA basics

Build FCA extensions

Discussion & conclusion
Answer Set Programming

Fact

\[
a(1) . b(2) . c(1,2).
\]

Rule & variable

\[
a(X) : \neg b(X) \Rightarrow a(2)
\]

Conjunction & negation: \( a \land \neg b \)

\[
p(X) : \neg a(X) ; \textbf{not} \ b(X) . \Rightarrow p(1)
\]

Implication: \( b(X) : a(X) \) holds if \( a(X) \Rightarrow b(X) \ \forall X \)

\[
q(X) : \neg X=1..3 ; b(X) : a(X) . \Rightarrow q(2) q(3)
\]

Potassco, the Potsdam Answer Set Solving Collection \ cf potassco.org
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"]
}
```
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

Motivation

Answer Set Programming to dot with Biseau

Reconstruct FCA basics

Build FCA extensions

Discussion & conclusion
Formal context encoding in ASP

<table>
<thead>
<tr>
<th></th>
<th>adult</th>
<th>child</th>
<th>female</th>
<th>male</th>
<th>boy</th>
<th>woman</th>
<th>man</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bob</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>eve</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>john</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can be encoded as rel/2 atoms:

\[
\text{rel}(\text{alice}, \text{female}). \ \text{rel}(\text{bob}, \text{adult}). \ \text{rel}(\text{eve}, \text{adult}). \ \ldots
\]
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 \(\text{rel}/2\) describes the context:

\[
\text{ext}(X) : \neg \text{rel}(X,\_); \text{rel}(X, Y) : \text{int}(Y).
\]

\[
\text{int}(Y) : \neg \text{rel}(\_, Y); \text{rel}(X, Y) : \text{ext}(X).
\]

ASP enables close-to-specification encoding
Computing the Galois lattice

Formal concepts \((A_n, B_n)\) are ordered:

\[ (A_1, B_1) < (A_2, B_2) \iff A_1 \subset A_2 \]

And they are linked to their greatest subconcept.

A possible encoding in ASP:

```prolog
% 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

Motivation

Answer Set Programming to dot with Biseau

Reconstruct FCA basics

Build FCA extensions

Discussion & conclusion
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:

\[
\begin{align*}
\text{ext}(X): & \quad \text{rel}(X,_,_); \quad \text{rel}(X,A,C): \quad \text{int}(A), \quad \text{cnd}(C). \\
\text{int}(X): & \quad \text{rel}(_,X,_); \quad \text{rel}(O,X,C): \quad \text{ext}(O), \quad \text{cnd}(C). \\
\text{cnd}(X): & \quad \text{rel}(_,_,X); \quad \text{rel}(O,A,X): \quad \text{ext}(O), \quad \text{int}(A). 
\end{align*}
\]

Need adaptation of code for visualizations
Object oriented concepts \((X, Y)\) defined by \(X = Y\diamond\) and \(Y = X\Box\):

\[
Y\diamond = \bigcup_{y \in Y} R_y \quad \text{and} \quad X\Box = \{y \in A | R_y \subseteq X\}
\]

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

Mining of the object-oriented concepts:

```prolog
% 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:

\[ \{ \text{ext} (_) \} < \text{nobj} \times \text{mins supp} / 100. \]

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

Motivation

Answer Set Programming to dot with Biseau

Reconstruct FCA basics

Build FCA extensions

Discussion & conclusion
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

- FCA reconstruction
  - basics (context, concept mining, Galois lattice)
  - extensions (iceberg, ternary,... integer pattern structure)

- Get graphs
  - universal data structure
  - fully customizable (dot)

- 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

\[
\text{link}(a,b).
\text{color}(a,b, \text{green}).
\text{color}(a, \text{red}).
\text{shape}(b, \text{rectangle}).
\]

\[
\text{Digraph graph } \{
\text{node } [\text{shape=ellipse}]
\text{edge } [\text{arrowhead=none}]
a [\text{fillcolor="red"}]
b [\text{shape="rectangle"}]
a \rightarrow b [\text{color="green"}]
\}
\]
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,_).
Other resulting lattices

veil type partial

ring number one
gill attachment free
veil color white
Formal concepts mining

**Definition**

A 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: \(\{b, e, f\} \times \{h, j, k, n\}, \{a, c, d, g\} \times \{i, l, m\}\)