Conexp-Clj – A Research Tool for FCA

Tom Hanika[1] and Johannes Hirth[1]

Knowledge & Data Engineering Group, University of Kassel, Germany
tom.hanika@cs.uni-kassel.de, jhi@cs.uni-kassel.de

Abstract There is a plenitude of software programs to analyze data sets using notions from formal concept analysis (FCA). For example, there are 64 FCA related projects listed on GitHub. Those are developed in ten different programming languages and provide tools and libraries for computing formal concepts and alike. The research tool `conexp-clj` sticks out in this list. It is the only application in the FCA realm developed using the programming language Clojure, which is a modern, dynamic, and functional dialect of the Lisp programming language running on the Java platform. We summarize in this work the extensive set of notions from FCA that are covered by `conexp-clj`, show recent developments, present simple examples on a real world data set and depict a timeline for our next goals.

Keywords: Formal Concept Analysis, Clojure, Functional Programming

1 Introduction

With the increase of computing power as well as its broad availability the field of *experimental mathematics* [2] has flourished in the last decades. This has led to the development of uncountable many applications, libraries, and highly sophisticated tools for computing in various realms of pure and applied mathematics. There are at least two kinds of user for those: The first kind are users employing this kind of software to other research fields, e.g., sociology, physics, or biology. The second kind applies the developed algorithms and tools within the field they were developed in. The goal here is to discover new insights and to advance the research field itself.

Formal concept analysis (FCA) is no exception to this categorization. There is a plenitude of tools for computing *formal concepts*, *implications* (functional dependencies) in data, etc. Many tools are extensive in their features, but limit the user through their user interfaces. Hence, formulating and expressing new ideas is bound to the expressiveness of the particular interface. Furthermore, getting your own new feature functions implemented and linked into the interface is costly and therefore not common in the academic realm. The software `conexp-clj`,[1] developed by Daniel Borchmann as part of the DFG project (GA 216/10-1), aims in a different direction.

[1] https://github.com/exot/conexp-clj

It embeds an extensive amount of FCA-functionality into a highly expressive programming language – Clojure. In this work we demonstrate this expressiveness on a real world data set and provide an overview about recent and future developments.

2 Basic Usage of `conexp-clj`

In this work we use notions from formal concept analysis (FCA) as introduced in [6]. A formal context is a triple $\mathbb{K}=(G,M,I)$ with G being a set of object, M being a set of attributes and $I\subseteq G\times M$ an incidence relation between them. For `conexp-clj` use-cases we often refer with the symbol `ctx` to a formal context and with the symbol `cpt` to a formal concept. The declarative nature of functional programming languages, like Clojure, makes the computation of FCA knowledge acquisition tasks intuitive and adaptive. Being a Lisp dialect, Clojure uses prefix notation and puts statements into parenthesis. Basic collections and their notation in Clojure are *sets* `#{}`, *vectors* `[ ]`, *lists* `'( )`, and *hash-maps* `{ }`. To create a context in `conexp-clj` one can use the `make-context-from-matrix` function, which creates a context from a binary matrix. The function `def` assigns the context to the symbol `ctx`:

```
(def ctx  (make-context-from-matrix
           ["platypus" "duck" "dog"]
           ["eggs" "mammal" "venomous"]
           [1 1 1
            1 0 0
            0 1 0]))
```

There are plenty of other ways to input formal contexts. For example, using its incidence relation one can call `(make-context #{1 2} #{1 2} #{[1 1] [1 2]})`. Furthermore, by providing a function `f` as incidence one can input a context implicitly by `(make-context objectset attributeset f)`, e.g., using the less or equal comparability `(make-context [1 2 3] [1 2 3] <=)`. Note that Clojure treats functions as *first-class* citizens, i.e., the language supports passing functions as arguments to other functions. Following the functional paradigm, Clojure functions are idempotent (often called pure) and return when called twice the same output for a given input. The object/attribute set of a context `ctx` accessed through `(attributes ctx)` and `(objects ctx)` respectively. For attributes we call by `(attribute-derivation ctx #{"mammal"})` the derivation operator and by `(context-attribute-clojure ctx #{"mammal"})` the closure operator. The operators for objects are named accordingly. The set of all concepts can be computed with `(concepts ctx)` and we obtain the canonical base (stem base) with `(canonical-base ctx)`. In terms of data exchange `conexp-clj` supports a variety of file formats. Among those are Burmeister, FCAalgs, Colibri, Conexp, CSV, and Galicia. In addition to binary contexts there is support for many-valued as well.

3 Unique Feature Coverage

Besides the choice of a functional, and therefore very expressive, programming language, `conexp-clj` is equipped with abundance of features. This enables the

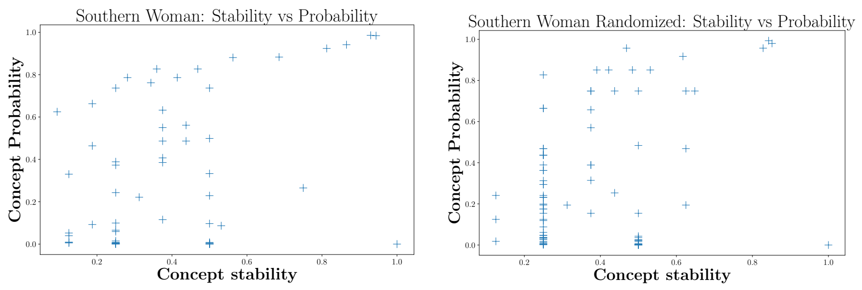

Figure 1. Stability vs probability for concepts from the Southern Woman data set.

researcher to easily implement new ideas and to experiment on them. First of all, `conexp-clj` tries to provide a diverse selection of algorithms for standard tasks like computing the set of concepts. For example, computing the set of concepts is possible through NEXT CLOSURE [5], KRAJCA [10], IN-CLOSE [1], and others. Furthermore, employing external binaries such as CbO or PCbO and interfacing with them is possible.

Interestingness of Concepts. There is a plenitude of measures to express how interesting, relevant, useful or similar concepts or sets of concepts are in a formal context. A good overview is presented in [12], from which the majority is implemented in `conexp-clj`. For example, there is *stability* [11] of formal concepts. The stability of a concept (A,B) indicates how likely B can be generated using uniformly drawn object subsets from A. Another is *robustness* [13], which is a measures for how likely a concept remains closed after removing an object from the concept's extent with probability $1-\alpha$. A different approach is taken by *concept probability* [9] that is the likelihood for an attribute set (object set) being closed. All those notions can be computed in `conexp-clj` for any formal context `ctx` in the following way:

- `(concept-stability ctx cpt)`
  computes $Stab(A,B) = \frac{|\{C \subseteq A \mid C' = B\}|}{2^{|A|}}$
- `(concept-robustness cpt (concepts ctx)` α`)`
  computes $r(c=(A,B),\alpha) = \sum_{d \leq c} (-1)^{|B_d|-|B_c|}(1-\alpha)^{|A_c|-|A_d|}$
- `(concept-probability ctx cpt)`
  computes $p(B=B'') = \sum_{k=0}^{n} p(|B'|=k, B=B'')$

We may remark that `conexp-clj` also includes `weighted similarity` for formal concepts. This function employs different measures for set similarity like the Jaccard Index, the Sorensen coefficient or symmetric set difference.

Expressing New Ideas. In the following we demonstrate how novel ideas can easily be expressed in `conexp-clj`. For this we compute two examples using the well known Southern Woman data set [14]. The example computations were chosen

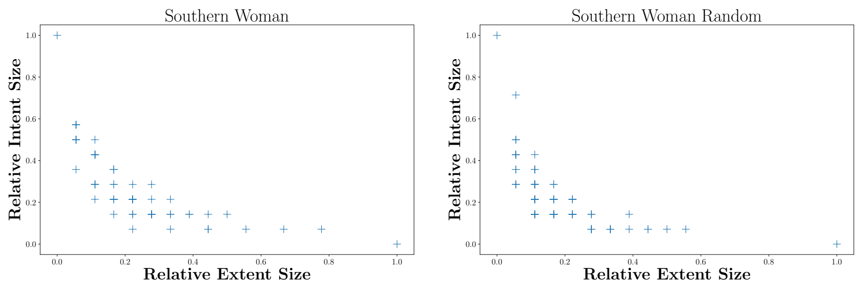

Figure 2. Stability vs probability for concepts from the Southern Woman data set.

for our demonstration purpose only. We do not claim that there is deeper insight to grasp from those. Imagine one wants to investigate possible correlations between the probability and stability of a set of formal concepts in some formal context. This idea can be expressed in simple terms using `conexp-clj` as shown below for context `ctx`:

```
(map (fn [x] [(concept-stability ctx x)
              (concept-probability ctx x)])
     (concepts ctx)))
```

This function call computes at first the set of all concepts for the given context. Secondly, it maps the functions concept stability as well as the probability to the beforehand computed formal concepts. Lastly, it returns a (lazy) list of two-element-vectors which then can be plotted using a suitable tool. We depicted the results in Figure 1, which does also include a plot for a random formal context which exhibits the same statistical properties as the Southern Woman data set, i.e., number of objects, number of attributes and density. The Clojure code shown above can be incorporated in a new function by `(defn somename [ctx] ...)`. Another idea could be the investigation of the relation between the sizes of a concept's intent and extent for a given formal context `ctx`, as shown in the next code example, also see Figure 2:

```
(map (fn [x] [(/ (count (first x))  (count (objects ctx)))
              (/ (count (second x)) (count (attributes ctx)))])
     (concepts ctx)))
```

Probably Approximately Correct Methods. When applying classical notions from FCA to large data sets one might encounter computationally infeasible problems. To date there are two approaches to cope with this implemented in `conexp-clj`. There is the *probably approximately correct (PAC) canonical base*, as investigated in [3]. Furthermore, there is a PAC version of the classical exploration algorithm present in the codebase of `conexp-clj`, which was introduced in [4].

Social Network Analysis and Graphs. Employing FCA for social network analysis has found an increasing attention in the last decade. Hence, various notions like *average shortest path*, *average local clustering coefficient* and *degree distribution* were added recently to `conexp-clj`. More functionality is developed continuously.

4 Outlook and Future Work

We presented the research tool `conexp-clj`and its comprehensiveness. Yet, there are many new features to come. For example, enhanced lattice diagram drawing through novel developed algorithms, interfacing with Wikidata through the SPARQL endpoint, motivated by [8], and dimension reduction methods like attribute selection [7].

References

[1] S. Andrews. “In-Close, a Fast Algorithm for Computing Formal Concepts.” In: *Proc. ICCS 2009*. Ed. by Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov. Vol. 483. CEUR Workshop Proceedings. CEUR-WS.org, 2009.

[2] D. H. Bailey and J. M. Borwein. “Experimental mathematics: Examples, methods and implications.” In: *Notices of the AMS* 52.5 (2005), pp. 502–514.

[3] D. Borchmann, T. Hanika, and S. Obiedkov. “On the Usability of Probably Approximately Correct Implication Bases.” In: *ICFCA*. Ed. by K. Bertet et al. Vol. 10308. Lecture Notes in Computer Science. Springer, 2017, pp. 72–88.

[4] D. Borchmann, T. Hanika, and S. Obiedkov. “Probably approximately correct learning of Horn envelopes from queries.” In: *Accepted for publication: Journal Discrete Applied Mathematics* abs/1807.06149 (2018). Accepted for publication: Journal Discrete Applied Mathematics.

[5] B. Ganter. “Two Basic Algorithms in Concept Analysis.” In: *Formal Concept Analysis*. Ed. by Léonard Kwuida and Barış Sertkaya. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 312–340.

[6] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, 1999, pp. x+284.

[7] T. Hanika, M. Koyda, and G. Stumme. “Relevant Attributes in Formal Contexts.” In: *Accepted for ICCS’19* abs/1812.08868 (2018).

[8] T. Hanika, M. Marx, and G. Stumme. *Discovering Implicational Knowledge in Wikidata*. cite arxiv:1902.00916. 2019.

[9] M. Klimushkin, S. Obiedkov, and C. Roth. “Approaches to the Selection of Relevant Concepts in the Case of Noisy Data.” In: *Formal Concept Analysis*. Ed. by Léonard Kwuida and Barış Sertkaya. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 255–266.

[10] P. Krajca, J. Outrata, and V. Vychodil. “Parallel Recursive Algorithm for FCA.” In: *Proc. CLA 2008*. Ed. by Radim Belohlavek and Sergei O. Kuznetsov. Vol. 433. CEUR Workshop Proceedings. CEUR-WS.org, 2008, pp. 71–82.

[11] S. O. Kuznetsov. “On stability of a formal concept.” In: *Annals of Mathematics and Artificial Intelligence* 49.1 (Apr. 2007), pp. 101–115.

[12] S.O. Kuznetsov and T. Makhalova. “On interestingness measures of formal concepts.” In: *Information Sciences* 442-443 (2018). Ed. by W. Pedrycz, p. 202.

[13] N. Tatti, F. Moerchen, and T. Calders. “Finding Robust Itemsets Under Subsampling.” In: *ACM Trans. Database Syst.* 39.3 (Oct. 2014), 20:1–20:27.

[14] S. Wasserman and K. Faust. *Social Network Analysis. Methods and Applications.* Structural Analysis in the Social Sciences. New York, USA: Cambridge University Press, 1994.