Towards Analyzing Variability in Space and Time of Products from a Product Line using Triadic Concept Analysis

Alexandre Bazin, Marianne Huchard, Pierre Martin

Towards Analyzing Variability in Space and Time of Products from a Product Line using Triadic Concept Analysis

Alexandre Bazin
LIRMM, Univ Montpellier, CNRS
Montpellier, France
alexandre.bazin@lirmm.fr

Marianne Huchard
LIRMM, Univ Montpellier, CNRS
Montpellier, France
marianne.huchard@lirmm.fr

Pierre Martin
AIDA, Cirad
Montpellier, France
pierre.martin@cirad.fr

ABSTRACT

In this paper, we report an ongoing work on exploring the ability of Triadic Concept Analysis to provide a framework for analyzing products evolution in time and space, and highlight possible usages in the lifecycle of a product line.

ACM Reference Format:
Alexandre Bazin, Marianne Huchard, and Pierre Martin. 2023. Towards Analyzing Variability in Space and Time of Products from a Product Line using Triadic Concept Analysis. In *27th ACM International Systems and Software Product Line Conference - Volume B (SPLC '23), August 28-September 1, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3579028.3609019

1 INTRODUCTION

Software Product Line Engineering (SPLE) [16] is currently an established paradigm in software engineering. The existing product lines face various challenges, including their evolution and maintenance. Recent work highlighted the importance of proposing conceptual foundations and tools for simultaneously supporting variability in *space* (i.e. modeling and development of alternative products) and variability in *time* (i.e. developing versions) [1]. The focus can be put on the evolution of the product line itself and its main elements (i.e. features, feature model, assets, reference architecture), or on the independent evolution of products once derived from the product line, by selecting features that were implemented with assets in different manners according to the adopted mechanism, i.e. annotative, compositional, or transformational [21].

In this paper, we focus on the independent evolution of products in terms of features, assuming that this evolution remains in the context of an evolving product line. Each product can acquire or lose features along its lifetime, knowing that a feature can result from a revision of other features. We call a snapshot the set of features of a product observed at a specific date. We aim to analyze the evolution of various products through snapshots at several dates, with the objective to formulate recommendations to the product developers (in selecting features during application engineering) or to product line developers (e.g. by suggesting feature constraints not identified in the current feature model, to consolidate domain engineering).

SPLC '23, August 28-September 1, 2023, Tokyo, Japan

ACM ISBN 979-8-4007-0092-7/23/08…$15.00
https://doi.org/10.1145/3579028.3609019

Observing the relationships between products and features has already been made in the context of variability in space. This analysis is, for instance, part of the process of feature model synthesis [3, 14, 19]. It has been done using specialized algorithms highlighting particular relationships, such as binary implications, mutex, and groups [19], or using Formal Concept Analysis (FCA) as a global approach to produce a canonical structure including all the logical relationships [5, 12, 18].

To analyze the variability in date and space of products from a product line, we therefore need to consider not only the two dimensions of products and features, but also a third dimension, i.e. the snapshots' dates. In this paper, we report an ongoing work on exploring the ability of an extension of FCA, i.e. Triadic Concept Analysis (TCA) to provide a framework for analyzing products evolution in the three dimensions, and highlighting possible usages in the lifecycle of a product line.

In Section 2, we introduce basics for FCA and TCA using a short working example, and focus on the implications provided by each of these methods. Then, in Section 3, we elaborate on the interpretation and usage of the obtained triadic implications. We conclude and give future directions of this work in Section 4.

2 POLYADIC CONCEPT ANALYSIS

Formal concept analysis (FCA), also called dyadic concept analysis, is a mathematical framework based on lattice theory that aims at structuring the information contained in the relation between *objects* and their *attributes* [8]. This structuring takes the form of *concepts*, i.e. groupings of objects that share the same attributes, or *implications*, i.e. regularities in the description of objects by attributes. In this paper, we are interested in implications as they encode knowledge on variability into logical relationships.

Binary relations are represented as binary tables called *formal contexts*. For instance, Table 1 illustrates the presence of features (attributes) in software systems, i.e. products (objects).

	f_1	f_2	f_3	f_4	f_5
p_1	×	×			×
p_2	×	×			×
p_3	×		×	×	×
p_4			×		×

Table 1: Example of formal context between products $p_i,\ 1 \leq i \leq 4$ and features $f_j,\ 1 \leq j \leq 5$.

Implications are regularities of the form $A \rightarrow B$, where A and B are sets of attributes, meaning that all objects described by attributes A are also described by attributes B. We name *support of an*

	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
p_1	×					×	×				×	×			
p_2						×	×				×	×			
p_3			×		×			×	×	×	×		×	×	×
p_4								×		×			×		×
p_5			×		×			×		×			×		×
p_6									×				×		×
			d_1					d_2					d_3		

Table 2: Example of triadic context between products p_i, features f_j, and snapshots' dates d_k, where d_1<d_2<d_3.

implication the set of objects described by A (and, thus, by $A \cup B$). For instance, in Table 1, the following implications may be observed:

- $\emptyset \rightarrow \{f_5\}$ (support $\{p_1, p_2, p_3, p_4\}$)
- $\{f_2\} \rightarrow \{f_1\}$ (support $\{p_1, p_2\}$)
- $\{f_4\} \rightarrow \{f_1, f_3, f_5\}$ (support $\{p_3\}$)
- $\{f_1, f_3\} \rightarrow \{f_4, f_5\}$ (support $\{p_3\}$)
- $\{f_3\} \rightarrow \{f_5\}$ (support $\{p_3, p_4\}$)

The first implication, $\emptyset \rightarrow \{f_5\}$, means that all products have the feature f_5. The set of all the implications that can be observed in this formal context is clearly redundant, e.g. $\{f_3\} \rightarrow \{f_5\}$ can be inferred from $\emptyset \rightarrow \{f_5\}$. Thus, one is often interested in subsets of implications called *implication bases* that minimize redundancy while still retaining all the information. One such implication base is made of the implications of the form $A \rightarrow \{b\}$, where A is a cardinality-minimal set of attributes such that the implication is valid. A is then called a *proper premise* [17]. In the SPLE domain, the proper premises implication base has been used in [18] to add missing dependencies to a feature diagram extracted from the attribute concept graph (excerpt of the concept lattice). The proper premises implication base of the Table 1 context is:

- $\emptyset \rightarrow \{f_5\}$
- $\{f_2\} \rightarrow \{f_1\}$
- $\{f_4\} \rightarrow \{f_1\}$
- $\{f_4\} \rightarrow \{f_3\}$
- $\{f_1, f_3\} \rightarrow \{f_4\}$
- $\{f_2, f_3\} \rightarrow \{f_4\}$ (support $\emptyset$)

A natural extension of FCA, TCA [11], is aimed at analyzing ternary relations, such as the one presented in Table 2 depicting a ternary relation between three object kinds: products (software systems), features, and snapshots' dates. As FCA, TCA computes concepts and implications. In TCA, a set of implications holds for each subset of dimensions [2, 7], such as the implications between sets of snapshots' dates, implications between sets of features, and implications between pairs of dimensions e.g. (feature, snapshots' date). An implication $A \rightarrow B$ holds if and only if for every x such that there is a triple $x \cup a$ for all $a \in A$, there is also a triple $x \cup b$ for all $b \in B$. The support of the implication is then the set of such x. Regarding Table 2 for instance, the support of implication $\{f_2\} \rightarrow \{f_1\}$ is $\{(p_1, d_2), (p_1, d_3), (p_2, d_2), (p_2, d_3)\}$ and the support of implication $\{(p_5, d_1)\} \rightarrow \{(p_5, d_2), (p_5, d_3)\}$ is $\{f_3, f_5\}$. Some implications have empty supports, i.e. their premise never appears in the dataset. These are interesting as they can be interpreted as impossibilities or mutual exclusions.

The next section presents examples of such implications together with their interpretation and potential usage by product line developers.

3 INTERPRETATION OF TCA RESULTS

Table 2 contains diverse product evolution patterns. Product p_1 aggregated feature f_2 with f_1 at date d_2. Product p_2 included features f_1 and f_2 at date d_2, and remained unchanged. Product p_3 included features f_3 and f_5 at d_1, then included f_4 at d_2, and f_1 at d_3. Product p_4 included features f_3 and f_5 at date d_2, and remained unchanged. Product p_5 was developed using features f_3 and f_5 at date d_1, and remained unchanged. Finally, Product p_6 included f_4 at d_2, that was replaced by f_3 and f_5 at d_3. These patterns of features adding/losing/remaining unchanged at various dates can be observed in the development of products from a product line.

This section presents some of the implications computed, as the proper premise base for Table 2, per subset of dimensions using TCA. An interpretation and a potential usage for product line developers of some implications is also proposed.

Implications between features.

- Example of implications:

FI1 $\{f_2\} \rightarrow \{f_1\}$

- Interpretation of [FI1]: This implication means that when a product p_i has feature f_2 at date d_j it also has feature f_1. It is held by $\{(p_1, d_2), (p_1, d_3), (p_2, d_2), (p_2, d_3)\}$.
- Potential usage: When the feature $\{f_2\}$ is included in the development of a product, then it can be recommended to add $\{f_1\}$. This recommendation can be strong if the support of this implication is a significant part of pairs (p_i, d_j), and more particularly if they correspond to recent versions of products. In such case, this recommendation could be added as a constraint in the feature model.

Implications between dates.

- Example of implications:

DI1 $\{d_1\} \rightarrow \{d_2, d_3\}$ (support $\{(p_1, f_1), (p_3, f_3), (p_3, f_5), (p_5, f_3), (p_5, f_5)\}$, i.e. all relations in the sub-context d_1 (left part of the table)).

- Interpretation of [DI1]: All features present in products at date d_1 are also present at dates d_2 and d_3.
- Potential usage: if dates d_1, d_2, and d_3 are successive, this may indicate that all product features present at date d_1 are a basis for products already developed, and that d_1 may be

the date of the foundational step of the products or of a main release.

Implications between products.

- Example of implications:

PI1 $\{p_5\} \rightarrow \{p_3\}$
PI2 $\{p_6\} \rightarrow \{p_3\}$
PI3 $\{p_5, p_6\} \rightarrow \{p_4, p_3\}$

- Interpretation of [PI1]: At all dates, if p_5 has some feature, p_3 has also this feature. Thus p_3 is an extension of p_5, i.e. always has more features than p_5.
- Potential usage of [PI1]: This may be used to recommend to developers of p_5 to complete their feature set, to be up to date. Alternatively, this can be used to recommend a lighter product.

Implications between pairs $(product, feature)$.

- Example of implications:

PFI1 $\emptyset \rightarrow \{(p_1, f_1), (p_3, f_3), (p_3, f_5), (p_5, f_3), (p_5, f_5)\}$ (support $\{d_1, d_2, d_3\}$)

- Interpretation of [PFI1]: product p_1 always has feature f_1, and products p_3 and p_5 always have features f_3 and f_5.
- Potential usage of [PFI1]: These implications may indicate what is sometimes called "core features". Here, they are not the core features of the product line, but the core of a product, or of a set of products. E.g. p_3 and p_5 may be considered to have common foundations.

Implications between pairs $(product, date)$.

- Example of implications:

PDI1 $\{(p_5, d_1)\} \rightarrow \{(p_6, d_3), (p_4, d_2), (p_4, d_3)\}$

- Interpretation of [PDI1]: all features, possessed by product p_5 at date d_1, are also possessed by product p_4 at date d_2 and d_3 and by product p_6 at date d_3.
- Potential usage of [PDI1]: This implication may address the relative versioning of products from a product line, where some features are included in specific versions or products (e.g. testing product or early release product) before being integrated in others.

Implications between pairs $(feature, date)$.

- Example of implications:

FDI1 $\{(f_4, d_2)\} \rightarrow \{(f_5, d_3)\}$ (support $\{p_3, p_6\}$)

- Interpretation of [FDI1]: all products that have features f_4 at date d_2 have feature f_5 at date d_3.
- Potential usage of [FDI1]: This implication may inform on specific feature needs induced by the time, either required by the users or technical. In addition, a technical need could be induced by the user need, such as securing exchanges in a version, that requires enabling password management, a feature appearing only from the next version.

Making concrete the example. We assume that the products p_i, $i = 1 \ldots 6$, have been derived using an hypothetical e-commerce product line. Dates are $d_1 = 2020/12/12$, $d_2 = 2021/12/12$ and $d_3 = 2022/12/12$, meaning that the product evolution has been observed at the end of three consecutive years. Table 3 presents the features. Two features allow comment management: *add comment* and *update comment*. Two features support purchase: *manage a basket* and *pay by creditcard*. One last feature (*manage a wishlist*) allows users to keep track of their choices.

f_1	add comment
f_2	update comment
f_3	manage a basket
f_4	manage a wishlist
f_5	pay by credit card

Table 3: Example of e-commerce website features

The implications presented above can therefore be interpreted as follows.

FI1 $\{update\ comment\} \rightarrow \{add\ comment\}$: regardless of the date and version, when a product allows to update comments, then it allows to add comments.

DI1 $\{2020/12/12\} \rightarrow \{2021/12/12, 2022/12/12\}$: this implication highlights that there are persistent features in a product by the time, e.g. features that were present in a product at the end of 2020 have been kept in 2021 and 2022.

PI1 $\{p_5\} \rightarrow \{p_3\}$: p_3 may be (over the years) viewed as an extension of p_5, corresponding to the addition of a wish list management feature to the purchase features.

PFI1 $\emptyset \rightarrow \{(p_1, add\ comment), (p_3, manage\ a\ basket), (p_3, pay\ by\ credit\ card), (p_5, manage\ a\ basket), (p_5, pay\ by\ credit\ card)\}$: this indicates essential features for the mentioned products as they persist in these products over the time.

PDI1 $\{(p_5, 2020/12/12)\} \rightarrow \{(p_6, 2022/12/12), (p_4, 2021/12/12), (p_4, 2022/12/12)\}$: what is provided by p_5 at the end of 2020 appears in p_4 at the end of 2021 and 2022, and in p_6 at the end of 2022. The set of purchase features (i.e. *manage a basket* and *pay by credit card*) was the whole set of features of p_5 as stated in 2020. As they were recognized as so important or evaluated positively by customers, they have later been adopted by two other products.

FDI1 $\{(manage\ a\ wishlist, 2021/12/12)\} \rightarrow \{(pay\ by\ credit\ card, 2022/12/12)\}$: products that provide *manage a wishlist* by the end of 2021 also necessarily provide *pay by credit card* by the end of 2022. This implication suggests that the wish list feature may have created users interest for online purchase, through the features added in the following version. This is true, even if the wish list feature itself disappears, as in p_6.

4 DISCUSSION AND RESEARCH AGENDA

In this paper, we have presented an ongoing work using Triadic Concept Analysis (TCA) to highlight and leverage knowledge on the evolution of features in products of a product line over time. This work is in an early stage, which has led to many questions to be answered.

Exploiting more Information. While we were analyzing the three dimensions (products, features, dates), we observed that additional information could be included in other dimensions such as the

software artifacts implementing the features, the fact that some features are revisions of other features, and the chronology. Adding software artifacts would consist in adding a 4th dimension, which can be addressed using the general polyadic concept analysis (n-dimensional) framework [22]. Another dimension in product line comes when it is relevant to distinguish various user roles to access different features depending on the version, i.e. a role may have access to a feature in a specific version, but not in another one. Chronology implies considering an order in an additional dimension by TCA. Feature revision information needs introducing a particular additional relationship, that we could investigate using a combination of TCA and Relational Concept Analysis [9].

Alternative FCA Knowledge Patterns. We may also consider using other formal concept analysis entities as they could provide different point of views. There are other implication bases that can be extracted from a formal context, each one having its own characteristics. In this paper, we have used the proper premises base. The Duquenne-Guigues base of implications (DGBI) is another base that has the particularity of being a cardinality minimal set of non-redundant implications. DGBI has already been used in the SPLE domain, e.g. in [12] to find additional logical constraints and in [4] to summarize variability in product descriptions. However, it is much more costly to compute DGBI than the proper premises base.

Using the formal concepts, from which the formalism gets its name, is an alternative to implication bases. While an implication base is a logical structure that supports reasoning, formal concepts are groups of objects sharing attributes. In the dyadic formal context of Table 1, a concept is $(\{p_1, p_2, p_3\}, \{f_1, f_5\})$, which means that products p_1, p_2 and p_3 share features f_1, f_5. In the triadic context of Table 2, a concept is a triple made of subsets of each dimension, i.e. a set of products, a set of features, and a set of dates. For example, the concept $(\{p_3, p_5\}, \{f_3, f_5\}, \{d_1, d_2, d_3\})$ means that products p_3 and p_5 have both features f_3 and f_5 at the three dates d_1, d_2, and d_3. In the dyadic case, concepts are ordered in a lattice structure, enabling efficient visualization and study of the content of the data. Concept lattices, or some of their excerpts, have been used in the dyadic case for Feature Model synthesis [5, 12, 18]. In the triadic case, although information is richer, it is more difficult to get good intuitions and how to represent the lattice is an open question. Approaches tried to propose forms of conceptual navigation in these triadic graphical representations [10], that we could leverage. Using such an approach to analyze multidimensional data challenges SPLE to imagine an intuitive graphical representation of the multi-dimensional variability.

Develop a Methodology. Finally, we aim at designing a complete methodology that would guide a product line designer or a product developer using knowledge extracted by TCA from the evolution of the existing products. To achieve this objective, we need to complete our analysis using the lessons learned from the implications in the several subsets of dimensions.

Both implications and formal concepts are costly to compute and the approach does not scale well. This could be a limit when analyzing the variability of product lines with many versions and features such as Linux [15] or the industrial video generator of [6]. An FCA-based approach thus aims to be devoted to medium-size dataset, to identified subsets of large datasets, or be combined with other approaches. FCA, as a symbolic / logic- based method, brings together exact implications, exact data structuring and knowledge patterns with explainable results. Other approaches use machine learning approaches to extract constraints, such as [20]. Finally, a worthwhile track of research would consist in combining both approaches. For assessing and parameterizing the approach, including its scalability and the relevance of results, exploiting the benchmark of [13] sounds promising.

Acknowledgments. This work was supported by the ANR SmartFCA project[1], Grant ANR-21-CE23-0023 of the French National Research Agency.

REFERENCES

[1] Sofia Ananieva, Sandra Greiner, Timo Kehrer, Jacob Krüger, Thomas Kühn, Lukas Linsbauer, Sten Grüner, Anne Koziolek, Henrik Lönn, S. Ramesh, and Ralf H. Reussner. 2022. A conceptual model for unifying variability in space and time: Rationale, validation, and illustrative applications. *Empir. Softw. Eng.* 27, 5 (2022), 101. https://doi.org/10.1007/s10664-021-10097-z

[2] Alexandre Bazin. 2020. On implication bases in n-lattices. *Discret. Appl. Math.* 273 (2020), 21–29. https://doi.org/10.1016/j.dam.2019.02.044

[3] Guillaume Bécan, Mathieu Acher, Benoit Baudry, and Sana Ben Nasr. 2016. Breathing ontological knowledge into feature model synthesis: an empirical study. *Empir. Softw. Eng.* 21, 4 (2016), 1794–1841. https://doi.org/10.1007/s10664-014-9357-1

[4] Jessie Carbonnel, Karell Bertet, Marianne Huchard, and Clémentine Nebut. 2020. FCA for software product line representation: Mixing configuration and feature relationships in a unique canonical representation. *Discret. Appl. Math.* 273 (2020), 43–64. https://doi.org/10.1016/j.dam.2019.06.008

[5] Jessie Galasso and Marianne Huchard. 2023. Extending Boolean Variability Relationship Extraction to Multi-valued Software Descriptions. In *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*, Roberto E. Lopez-Herrejon, Jabier Martinez, Wesley Klewerton Guez Assunção, Tewfik Ziadi, Mathieu Acher, and Silvia Regina Vergilio (Eds.). Springer International Publishing, Berlin, 143–173. https://doi.org/10.1007/978-3-031-11686-5_6

[6] José Angel Galindo, Mauricio Alférez, Mathieu Acher, Benoit Baudry, and David Benavides. 2014. A variability-based testing approach for synthesizing video sequences. In *International Symposium on Software Testing and Analysis, ISSTA '14, San Jose, CA, USA - July 21 - 26, 2014*, Corina S. Pasareanu and Darko Marinov (Eds.). ACM, 293–303. https://doi.org/10.1145/2610384.2610411

[7] Bernhard Ganter and Sergei A. Obiedkov. 2004. Implications in Triadic Formal Contexts. In *Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS 2004 Proceedings (Lecture Notes in Computer Science, Vol. 3127)*, Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach (Eds.). Springer, Huntsville, AL, USA, July 19-23, 2004, 186–195. https://doi.org/10.1007/978-3-540-27769-9_12

[8] Bernhard Ganter and Rudolf Wille. 1999. *Formal Concept Analysis - Mathematical Foundations.* Springer, Berlin. https://doi.org/10.1007/978-3-642-59830-2

[9] Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. 2013. Relational concept analysis: mining concept lattices from multi-relational data. *Ann. Math. Artif. Intell.* 67, 1 (2013), 81–108. https://doi.org/10.1007/s10472-012-9329-3

[10] Levente Lorand Kis, Christian Sacarea, and Diana Troanca. 2016. FCA Tools Bundle - A Tool that Enables Dyadic and Triadic Conceptual Navigation. In *Proceedings of the 5th International Workshop "What can FCA do for Artificial Intelligence"? co-located with the European Conference on Artificial Intelligence, FCA4AI@ECAI 2016 (CEUR Workshop Proceedings, Vol. 1703)*, Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph (Eds.). CEUR-WS.org, The Hague, the Netherlands, August 30, 2016, 42–50. https://ceur-ws.org/Vol-1703/paper6.pdf

[11] Fritz Lehmann and Rudolf Wille. 1995. A Triadic Approach to Formal Concept Analysis. In *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95 Proceedings (Lecture Notes in Computer Science, Vol. 954)*, Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa (Eds.). Springer, Santa Cruz, California, USA, August 14-18, 1995, 32–43. https://doi.org/10.1007/3-540-60161-9_27

[12] Felix Loesch and Erhard Ploedereder. 2007. Restructuring Variability in Software Product Lines using Concept Analysis of Product Configurations. In *11th European Conference on Software Maintenance and Reengineering, Software Evolution in Complex Software Intensive Systems, CSMR 2007, 21-23 March 2007*, René L. Krikhaar, Chris Verhoef, and Giuseppe A. Di Lucca (Eds.). IEEE Computer Society, Amsterdam, The Netherlands, 159–170. https://doi.org/10.1109/CSMR.2007.40

[1] https://www.smartfca.org/

[13] Jabier Martinez, Wesley K. G. Assunção, and Tewfik Ziadi. 2017. ESPLA: A Catalog of Extractive SPL Adoption Case Studies. In *Proceedings of the 21st International Systems and Software Product Line Conference, SPLC 2017, Volume B, Sevilla, Spain, September 25-29, 2017*, Maurice H. ter Beek, Walter Cazzola, Oscar Díaz, Marcello La Rosa, Roberto E. Lopez-Herrejon, Thomas Thüm, Javier Troya, Antonio Ruiz Cortés, and David Benavides (Eds.). ACM, 38–41. https://doi.org/10.1145/3109729.3109748

[14] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2023. Bottom-Up Technologies for Reuse: A Framework to Support Extractive Software Product Line Adoption Activities. In *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*, Roberto E. Lopez-Herrejon, Jabier Martinez, Wesley Klewerton Guez Assunção, Tewfik Ziadi, Mathieu Acher, and Silvia Regina Vergilio (Eds.). Springer International Publishing, Berlin, 355–377. https://doi.org/10.1007/978-3-031-11686-5_14

[15] Sarah Nadi and Richard C. Holt. 2014. The Linux kernel: a case study of build system variability. *J. Softw. Evol. Process.* 26, 8 (2014), 730–746. https://doi.org/10.1002/smr.1595

[16] Klaus Pohl, Günter Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering - Foundations, Principles, and Techniques*. Springer, Berlin. https://doi.org/10.1007/3-540-28901-1

[17] Uwe Ryssel, Felix Distel, and Daniel Borchmann. 2014. Fast algorithms for implication bases and attribute exploration using proper premises. *Ann. Math. Artif. Intell.* 70, 1-2 (2014), 25–53. https://doi.org/10.1007/s10472-013-9355-9

[18] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. 2011. Extraction of feature models from formal contexts. In *Software Product Lines - 15th International Conference, SPLC 2011, Workshop Proceedings (Volume 2)*, Ina Schaefer, Isabel John, and Klaus Schmid (Eds.). ACM, Munich, Germany, August 22-26, 2011, 4. https://doi.org/10.1145/2019136.2019141

[19] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2011. Reverse engineering feature models. In *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011*, Richard N. Taylor, Harald C. Gall, and Nenad Medvidovic (Eds.). ACM, Waikiki, Honolulu , HI, USA, May 21-28, 2011, 461–470. https://doi.org/10.1145/1985793.1985856

[20] Paul Temple, José Angel Galindo, Mathieu Acher, and Jean-Marc Jézéquel. 2016. Using machine learning to infer constraints for product lines. In *Proceedings of the 20th International Systems and Software Product Line Conference, SPLC 2016, Beijing, China, September 16-23, 2016*, Hong Mei (Ed.). ACM, 209–218. https://doi.org/10.1145/2934466.2934472

[21] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.* 47, 1 (2014), 6:1–6:45. https://doi.org/10.1145/2580950

[22] George Voutsadakis. 2002. Polyadic Concept Analysis. *Order* 19, 3 (2002), 295–304. https://doi.org/10.1023/A:1021252203599