Exploring the variability of interconnected product families with relational concept analysis

Jessie Carbonnel, Marianne Huchard, Clémentine Nebut

Exploring the Variability of Interconnected Product Families with Relational Concept Analysis

Jessie Carbonnel, Marianne Huchard and Clémentine Nebut
LIRMM, Université de Montpellier, CNRS, Montpellier, France

October 21, 2021

Abstract

Among the various directions that SPLE promotes, extractive adoption of complex product lines is especially valuable, provided that appropriate approaches are made available. Complex variability can be encoded in different ways, including the feature model (FM) formalism extended with multivalued attributes, UML-like cardinalities, and references connecting separate FMs. In this paper, we address the extraction of variability relationships depicting connections between systems from separate families. Because Formal Concept Analysis provides suitable knowledge structures to represent the variability of a given system family, we explore the relevance of Relational Concept Analysis, an FCA extension to take into account relationships between different families, to tackle this issue. We investigate a method to extract variability information from descriptions representing several inter-connected product families. It aims to be used to assist the design of inter-connected FMs, and to provide recommendations during product selection.

1 Introduction

As families of similar software systems grow in size and complexity, managing them by adopting an approach based on software product line (SPL) engineering becomes more and more relevant. When various software systems of a same family have been individually developed in an undisciplined way, migrating to an SPL may be done through an extractive approach, by analyzing their commonalities and differences, and by identifying the reusable assets and a reference architecture. Complex variability extraction may result in Feature Models (FMs) [KCH+90, CW07] extended with multivalued attributes, UML-like cardinalities, or references connecting separate FMs [CBUE02].

Previous research work has studied variability extraction in the boolean case [LP07, HLE11, RPK11, ACP+12, AMHS+14, MZB+17, SSS17], and in

presence of multi-valued attributes and cardinalities [BBGA15, CHN19b]. More specifically, in [CHN19b], we show how Formal Concept Analysis (FCA) and its extension to Pattern Structures help in extracting variability relationships from a product family described by multi-valued attributes and cardinalities. It leads to binary implications, groups and mutex involving boolean features as well as attribute values and cardinalities.

In this paper, we address the problem of *cross-family variability relationship extraction* from descriptions representing several inter-connected product families (see Figure 1). These families may correspond to various concerns or product pieces, and may be connected by different relationships, e.g., *use*, *part-of*, *compatible-with*, *provide*. We introduce a method based on Relational Concept Analysis (RCA), an FCA extension to take into account these relationships between the product families. This method produces concept lattice families from which are extracted *cross-family features* and *cross-family relationships* that can be binary implications, mutex, co-occurrences or groups involving both boolean features and cross-family features. This is part of a general approach we envisage, thus we also discuss how in the future this method can assist the design of FMs with new kinds of references, and the exploration of existing configurations for the sake of recommendation.

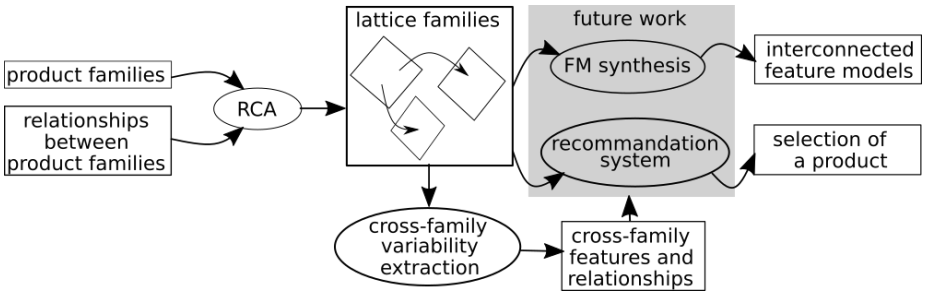


Figure 1: Schema of the envisaged approach.

In Section 2, we introduce the foundations of variability extraction with FCA. Addressing cross-family variability extraction with Relational Concept Analysis is developed in Section 3. Then, we discuss how the extracted cross-family variability can help in inter-connected FMs design and in configuration exploration (Section 4). We develop related work in Section 5, before concluding and drawing perspectives of this work in Section 6.

2 Background: Assisted variability extraction with FCA

Variability modeling. Variability modeling appears in the earliest steps for extractive adoption of a software product line from a family of product variants. To elaborate a variability model in the context of a feature-oriented mod-

eling approach [KCH+90], the main existing approaches need to identify (a) representative and discriminating *features* and (b) relationships between these features (e.g. implications, mutually exclusive features or groups). Then the extracted information is represented through *feature models* [KCH+90, CW07, SHTB07], propositional logics [Bat05], description logics [BEGB11], or constraints [SDM+11]. Relationships between features are usually extracted from the product variant descriptions depicting variants along with their features. For example, these descriptions may take the form of tabular descriptions, such as Product Comparison Matrices that can be found in Wikipedia[1] and extracted thanks to API such as OpenCompare[2], produced by random or manual sampling from generators, such as using the web application generator JHipster[3][HNA+17], or by analysing software developed by communities, such as Robocode[4]. These tabular descriptions generally expose characteristics (as columns) and their values (in cells) for the different product variants (as rows). They need to be cleaned, manually, or automatically [BSA+14] before automated exploitation. From the cleaned tabular descriptions, several methods extract relationships for boolean features, leading to FM synthesis or more simply logical relationships extraction [LP07, HLE11, RPK11, ACP+12, AMHS+14, MZB+17, SSS17, CHN19a]. A few approaches [BBGA15, CHN19b] address the problem of extracting more complex variability information to take into account multi-valued attributes and cardinalities. Several of these approaches [LP07, RPK11, AMHS+14, SSS17, CHN19b, CHN19a] are based on Formal Concept Analysis, which can be seen as a structuring framework for variability analysis and representation, in which some of the other approaches can be embedded, as shown in [AMdSH+14, CHN19a].

Formal Concept Analysis (FCA) Formal Concept Analysis (FCA) [GW99] can be summarized by the equation: "objects + attributes = concept hierarchy". In other words, based on a set of objects (entities, individuals) described by a set of attributes (properties, characteristics), FCA helps to: (1) group a maximal set of objects sharing a maximal set of attributes into a concept, and (2) hierarchically organize the set of concepts.

In the simplest form, FCA considers as input a *formal context* $K = (O, A, J)$, where O is a set of objects, A is a set of attributes and $J \subseteq O \times A$ is a binary relationship, where $(o, a) \in J$ when "o possesses a". Table 1 represents in a tabular form a formal context KET describing (in a simplified way) Knowledge Engineering (KE) tools (left-hand-side), and a formal context KEC describing basic Knowledge Engineering (KE) components (right-hand-side). The first column of a formal context presents the objects (here in the form of an identifier representing KE tools or components), and the first row gathers the attributes (here the boolean features). KE tools are described by their import/export

[1] E.g. comparisons on software systems: https://en.wikipedia.org/wiki/Category: Software_comparisons, last accessed in March 2019

[2] https://github.com/OpenCompare

[3] https://www.jhipster.tech/

[4] https://github.com/but4reuse/RobocodeSPL_teaching

formats, their distribution mode (commercial or open source) and if they are delivered in SaaS mode (Software as a Service). Basic KE components are dedicated to resource creation (ontology, rules or raw file) or to implement a machine learning algorithm from the categories: rule extraction, decision tree or neural network. The strategy can be supervised or unsupervised. The components can be classified as a symbolic method or as a statistical method.

Table 1: Product descriptions of Knowledge Engineering Tools KET (lhs), and Knowledge Engineering Components KEC (rhs).

KET	knowledge engineering tool (ket)	import export (ie)	rdf	json	proprietay format (pf)	distribution (di)	commercial (co)	open source (os)	saas (sa)
ket0	×	×	×			×		×	×
ket1	×	×	×			×		×	×
ket2	×	×	×	×		×		×	
ket3	×	×	×	×		×		×	
ket4	×	×		×	×	×	×		
ket5	×	×		×	×	×	×		
ket6	×	×		×		×		×	×
ket7	×	×			×	×	×		×
ket8	×	×			×	×	×		×

KEC	knowledge engineering component (kec)	resource creation (rc)	ontology definition (od)	rule definition (rd)	raw file (rf)	learning (le)	algorithm (al)	rule extraction (re)	decision tree (dt)	neural network (nn)	strategy (str)	supervised (su)	unsupervised (un)	symbolic (sy)	statistical (sta)
od1	×	×	×											×	
od2	×	×	×											×	
rd1	×	×		×										×	
f1	×	×			×									×	
re1	×					×	×	×			×	×		×	
re2	×					×	×	×			×		×	×	
dt1	×					×	×		×		×	×			×
nn1	×					×	×			×	×	×			×

FCA extracts a set of *formal concepts*. Each concept $C = (E, I)$ is a maximal group of objects E associated with a maximal group of attributes I these objects share. $E = \{o \in O \mid \forall a \in I, (o, a) \in J\}$ is the concept extent and $I = \{a \in A \mid \forall o \in E, (o, a) \in J\}$ is the concept intent. For example (using short names), objects $E_{sy} = \{od1, od2, rd1, f1, re1, re2\}$ share attributes $I_{sy} = \{sy, kec\}$. As these sets cannot be extended, (E_{sy}, I_{sy}) is a concept.

Extent inclusion (and intent containment) induces a specialization order $\leq_{CL}$ between concepts: for two concepts $C_1 = (E_1, I_1)$ and $C_2 = (E_2, I_2)$, $C_1 \leq_{CL} C_2$ if and only if $E_1 \subseteq E_2$ (and equivalently $I_1 \supseteq I_2$). For example, if we consider the concept (E_{rc}, I_{rc}) with $E_{rc} = \{od1, od2, rd1, f1\}$ and $I_{rc} = \{rc, sy, kec\}$, $(E_{rc}, I_{rc}) \leq_{CL} (E_{sy}, I_{sy})$. The *concept lattice* is the set of all concepts C_K of the formal context K, provided with the order $\leq_{CL}$. *Attribute-concepts* (resp. *Object-concepts*) are particular concepts that contain at least an attribute (resp. an object) which is not present in a super-concept (resp. sub-concept). An AOC-poset (for *Attribute-Object-Concept partially ordered set*) is the restriction of the concept lattice to these specific concepts. The AOC-poset which classifies the KE tools (resp. components) of the left-hand-side (resp. right-hand-side) of Table 1 is shown in Figure 2 (resp. Figure 3). A concept is shown as a 3-part box containing: (1) the concept identifier, (2) its intent deprived from the attributes inherited from the super-concepts, and (3) its extent deprived from the objects that appear in the sub-concepts.

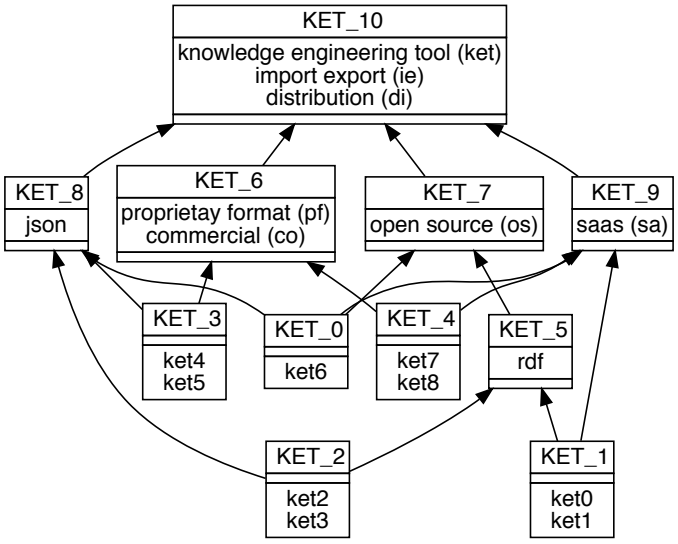


Figure 2: AOC-poset of Knowledge Eng. Tools KET.

Variability information extraction with FCA Variability information can be extracted from the AOC-poset and used to guide the FM synthesis [CHN19a]. Table 2 explains how variability information can be extracted from AOC-posets and translated into FM constructs. The first row considers the situation in which a concept introducing a feature f_2 is a sub-concept of a concept introducing f_1 (denoted by $C_{f_2} <_{CL} C_{f_1}$). In this case, all configurations having f_2 also have f_1, that leads to the propositional formula. Such an implication can be translated into 3 different FM constructs: a refinement relationship ①, an optional relationship ② or a requires constraint ③.

The second row shows how logical equivalences can be read in AOC-posets. When f_1 and f_2 are introduced in the same concept, this means that features f_1 and f_2 are always present together ($f_1 \leftrightarrow f_2$). In a feature diagram, this can be represented: with a mandatory relationship ④ or with two requires constraints ⑤.

The third row shows how mutual exclusions can be read in an AOC-poset. Two features f_1 and f_2 (respectively introduced in C_{f_1} and C_{f_2}) are mutually exclusive if the common sub-concepts of C_{f_1} and C_{f_2} (denoted $C_{f_1} \sqcap C_{f_2}$) do not introduce any object.

The fourth row presents the mapping of or-groups ⑦ into AOC-posets. We consider $\{f_1, ..f_k\}$ the features involved in an or-group, and f_0 the parent-feature of this group. All configurations having one of the features in $\{f_1, ..f_k\}$ should also have f_0, and conversely, configurations having the parent-feature f_0 have at least one feature of the or-group. Thus, the union of the extents of concepts introducing features of $\{f_1, ..f_k\}$ is equal to the extent of the concept introducing f_0. Moreover, the concept introducing the parent-feature of an or-group is not

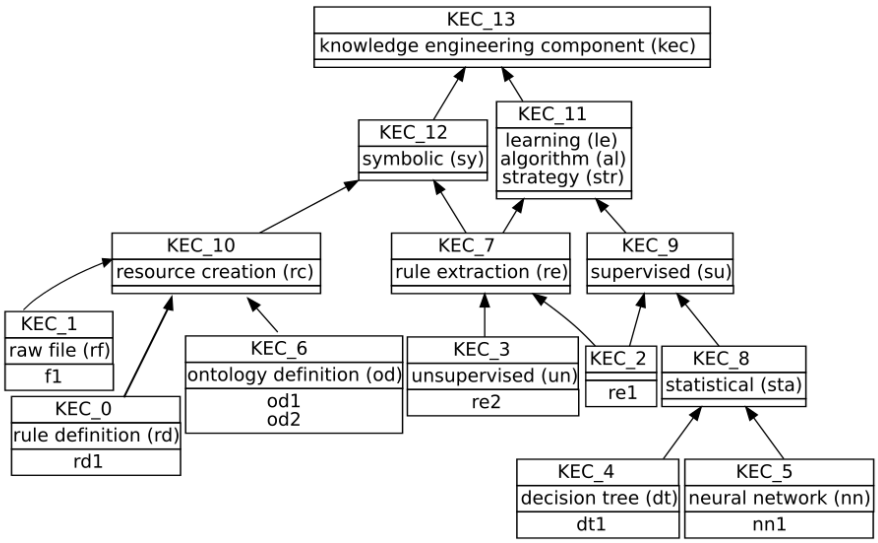


Figure 3: AOC-poset of Knowledge Eng. Components KEC.

an object-concept, as at least one feature of $\{f_1, ..f_k\}$ has to be selected. It is denoted by $C_{f_0} \notin \mathcal{OC}$, C_{f_0} being the concept introducing the parent-feature, and $\mathcal{OC}$ the set of object-concepts of the AOC-poset. Besides, we consider that there always exists a root feature, which appears in the top concept containing all configurations.

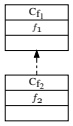
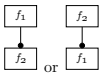
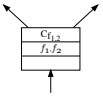
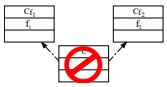
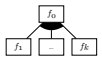
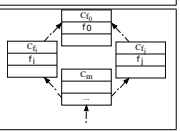
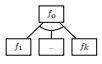
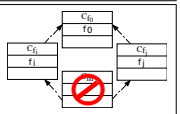
The last row presents the mapping of xor-groups ⑧ into AOC-posets. Xor-groups are like or-groups, but concepts introducing the features of $\{f_1, ..f_k\}$ do not have a common sub-concept introducing an object. Indeed, features involved in a xor-group and features involved in an *exclude* cross-tree constraint have a similar behaviour, as they are mutually exclusive in both situations.

3 Cross-product family variability extraction with RCA

Product line engineering faces inescapable issues related to the growth in size and complexity of software systems and to their evolution [HGR12]. Decomposing systems and considering separated concerns or gradual construction are solutions that have been investigated, e.g. in multi-product lines [HGR12, Bot13, RSKuR08]. Extending the variability extraction from a product family to a set of interconnected product families means that we need to consider both intra-family and inter-family (cross-family) variability information.

Relational Concept Analysis Cross-family variability extraction can be assisted by Relational Concept Analysis (RCA) [RHHNV13] which considers sev-

Table 2: Mapping between AOC-posets and FMs [CHN19a]. The extent of a context C is denoted by $EXT(C)$

FMs	Prop. form.	AOC-posets	
①  ②  ③ $f_2 \rightarrow f_1$	$f_2 \rightarrow f_1$	$C_{f_2} \leq_{CL} C_{f_1}$	
④  or ⑤ $f_1 \rightarrow f_2$ $f_2 \rightarrow f_1$	$f_1 \leftrightarrow f_2$	$C_{f_1} =_{CL} C_{f_2}$	
⑥ $f_1 \rightarrow \neg f_2$	$f_1 \rightarrow \neg f_2$ or $f_2 \rightarrow \neg f_1$	$Ext(C_{f_1} \sqcap C_{f_2}) = \emptyset$	
⑦ 	$f_0 \leftrightarrow (f_1 \vee \ldots \vee f_k)$	$\forall f_i \in \{f_1, ..f_k\}, C_{f_i} \leq_{CL} C_{f_0}$. $\{C_{f_1}, ...C_{f_k}\}$ is the greatest antichain [JRJ94] verifying: $Ext(C_{f_1}) \cup ... \cup Ext(C_{f_k}) = Ext(C_{f_0})$. $C_{f_0} \notin \mathcal{OC}$	
⑧ 	$f_0 \leftrightarrow (f_1 \oplus \ldots \oplus f_k)$	$\forall f_i \in \{f_1, ..f_k\}, C_{f_i} \leq_{CL} C_{f_0}$. $\forall f_i, f_j \in \{f_1, ..f_k\} \mid f_i \neq f_j$, $Ext(C_{f_i} \sqcap C_{f_j}) = \emptyset$. $\{Ext(C_{f_1}), ..., Ext(C_{f_k})\}$ is a partition of $Ext(C_{f_0})$. $C_{f_0} \notin \mathcal{OC}$	

eral object categories (one per product family). While FCA groups objects sharing commonalities in their intrinsic attributes, RCA additionally groups objects sharing commonalities in their similar relations to other objects having themselves commonalities. To that aim, RCA iteratively applies FCA to propagate similarities from objects in one category to objects in (possibly) another category through relations. To follow up our illustrative example, identifying the group of statistical KE components can be propagated to KE tools through a *provides* relation, leading to identify the group of KE tools dedicated to statistical analyses.

The considered dataset in RCA is a Relational Context Family (RCF), which is a set of object-attribute contexts (objects described by intrinsic attributes) and a set of object-object contexts (relations between objects of the different categories). For our example, the RCF is composed of the object-attribute contexts KEC and KET (Table 1), and the object-object relation *provides* $\subseteq KET \times KEC$ (Table 4). In the general case, an RCF contains n object-attribute contexts $K_i = (O_i, A_i, J_i), i \in \{1, ..., n\}$ and m object-object contexts $R_j = (O_k, O_l, r_j)$, $j \in \{1, ..., m\}$, with $r_j \subseteq O_k \times O_l$ is a binary relation such that $k,\ l \in \{1, ..., n\}$, O_k is the domain of the relation, and O_l is the range of the relation.

To consider the *provides* relation, information is added to the description of objects of its domain (KET). A straightforward integration scheme could

be to simply add to the object-attribute context KET attributes of the form $(provides, kec)$, and to add a cross when an object of KET provides the corresponding object of KEC. For example, $(provides, od1)$ could be added to the description of $ket0$ in a new column of KET. This would allow to group KETs that own the same KECs, but we would loose the valuable knowledge given by the KEC concepts. Remember that these concepts indicate what are the shared characteristics of KECs, e.g. $dt1$ and $nn1$ are grouped in KEC_8 because both are statistical-based KECs (see Fig. 3). Now if we consider the way *provides* describes $ket7$ and $ket8$ (resp. by $(provides, dt1)$ and $(provides, nn1)$), these two KETs do not share any KEC, whereas they share the fact that they allow statistical-based learning. To acquire such information, we propagate the knowledge highlighted in the KEC AOC-poset to the KET AOC-poset through *relational attributes*. These relational attributes are formed using scaling quantifiers inspired by constructors used in descriptions logics [BCM$^+$03], such as $\exists$ and $\exists\forall$[5], as illustrated in Table 3. For example, $ket7$ and $ket8$ descriptions will be enhanced by a relational attribute $\exists provides(KEC\_8)$ to indicate that both $ket7$ and $ket8$ have a link to at least one object of the KEC_8 extent.

In the RCA process, a scaling quantifier q is associated with each relation r, and systematically applied to form all the relational attributes of the form $qr(C)$ with C a concept formed on the objects of the range of r. These relational attributes are added to the context of the domain of r and a new extended AOC-poset can be built. For example, Fig. 4 shows the AOC-poset built for KET extended with the relational attributes formed with the quantifier $\exists\forall$ applied to *provides*. In this AOC-poset, KET_4 now highlights new shared information about $ket7$ and $ket8$ which is the fact that both provide **only** ($\exists\forall$) statistical-based learning components (the components they provide are all in KEC_8 extent).

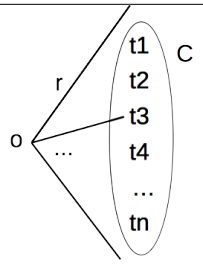
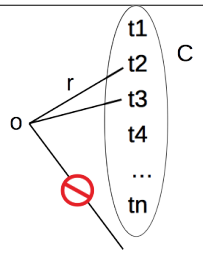
RCA can consider complex and possibly cyclic entity-relationship models. A non-trivial modeling phase is central in this perspective, to determine which are the object-attribute contexts (separate product families) and which are the object-object contexts (interconnections between the families), and the adequate scaling quantifiers.

Cross-family features Table 3 shows two examples of the new features that this framework makes available to describe cross-family variability. These features are abstractions of links between an object o (a product configuration of the domain of r in our context) and a concept C (a group of product configurations of the range of r). Existential features ($\exists r(C)$) indicate if a configuration o is linked by r to at least one configuration in the extent of Concept C. In our example, $\exists provides(C)$ is the feature "provides a KEC with the features of C". Feature $\exists provides(KEC\_8)$ can be read "provides a statistical learning component". Universal features ($\exists\forall r(C)$) indicate if all r links of a configuration o are towards configurations in the extent of Concept C ; for our example, it can

[5] $\exists\forall$ is used instead of $\forall$ to avoid assigning to an object o a relational attribute formed on r when $r(o)$ is empty (and would be included in any concept extent).

be summarized as "provides only KEC with the features of C". As presented previously, $\exists\forall provides(KEC\_8)$ can be read "provides only statistical learning components".

Table 3: Examples of cross-family features

Object links vs. concept extent	Schema	Relational attribute Cross-family Feature
$r(o) \cap Extent(C) \neq \emptyset$		existential $\exists r(C)$
$r(o) \subseteq Extent(C) \quad r(o) \neq \emptyset$		universal $\exists\forall r(C)$

Cross-family variability relationships Variability information can be extracted from the AOC-poset resulting from RCA, using the same mappings as introduced in Table 2. The found relationships also involve cross-family features. Three examples are given below using the AOC-poset of Figure 4.

Example of co-occurrence. By the second row of Table 2, *rdf* and $\exists\forall provides(KEC\_12)$ are co-occurrent, since both are introduced in the same KET_5. In other words, since KEC_12 corresponds to symbolic components, that means that tools exporting in rdf only use symbolic methods.

Example of implication. By the first row of Table 2, KET_1 is below KET_9, thus $\exists\forall provides(KEC\_6) \rightarrow Saas$: the knowledge engineering tools providing components that only deal with ontology definition (KEC_6) are delivered in SaaS mode. This can be translated through several ways in a feature model (see cases ①, ② and ③ in Table 2).

Example of exclusion. KET_5 and KET_6 do not have a common subconcept introducing a configuration. By the third row of Table 2, $\exists\forall provides(KEC\_6) \rightarrow \neg pf$. We can deduce that none of the KETs providing

provides	od1	od2	rd1	fl	re1	re2	dt1	nn1
ket0	×							
ket1	×	×						
ket2	×	×			×			
ket3		×	×			×		
ket4	×	×					×	×
ket5				×	×	×	×	×
ket6				×				×
ket7							×	
ket8								×

Table 4: Relation *provides* between Knowledge Engineering Tools KET (rows) and Knowledge Engineering Components KEC (columns).

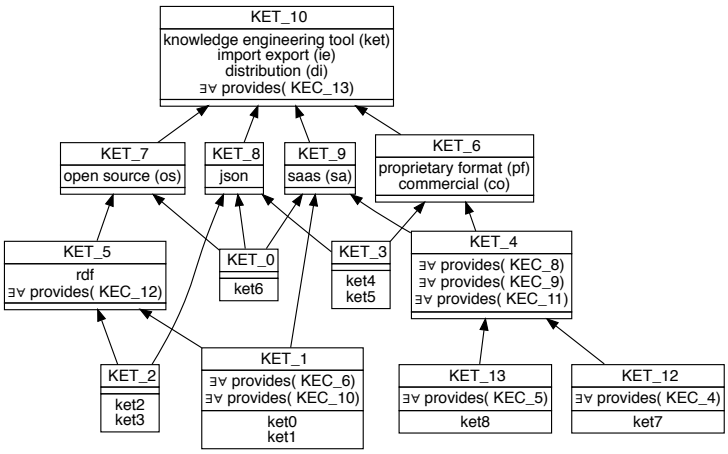


Figure 4: AOC-poset of Knowledge Engineering Tools KET after integration of information about *provides*, using $\exists\forall$ quantifier.

components based only on symbolic methods (KEC_12) is based on a proprietary format.

4 Exploiting cross-family variability

In this section, we draw tracks of future research to exploit cross-family variability information.

Extended (semi-automated) FM synthesis Intra-family information has been used to assist FM synthesis [CW07, SLB+11, HLE11, RPK11, ACP+12, HLE13, AMdSH+14, SRA+14, CHN19b]. The methods that use FCA to assist FM synthesis [RPK11, AMdSH+14, CHN19b] exploit a concept structure to derive automatically an FM or to provide user guidance by reducing their choices during FM construction. For example, the FM of KETs in Fig. 5 (resp. of KECs

in Fig. 6) can be extracted from the AOC-poset of Fig. 2 (resp. Fig. 3) by an FM designer, assisted by the rules presented in Sect. 2. Cross-family variability information can in turn be used to assist the synthesis of interconnected FMs, either by writing constraints between two different FMs (as shown in Sect. 3), or to enhance an FM from one family by features coming from knowledge about relations in real configurations. We illustrate this by using information extracted from the AOC-poset of KET (Fig. 4). First, we introduce a reference `kec` below the FM root, with cardinality *1-many (*)*. Then the AOC-poset states that in observed real configurations (as shown in KET_4), some KETs propose only supervised learning algorithms ($\exists\forall provides(KEC\_9)$), thus if a user chooses that option when he designs a KET, he should be guided afterwards to configure only components implementing supervised learning algorithms. The AOC-poset also shows (in KET_5) that a group of KETs is dedicated to symbolic approaches only ($\exists\forall provides(KEC\_12)$) and a subgroup of them is dedicated to ontology definitions only ($\exists\forall provides(KEC\_6)$). From these observations, the FM designer could decide to introduce a feature `supervised ket (sup)`, a feature `symbolic ket (sy)` with a sub-feature `ontology dedicated (o)`. These choices are represented as an extension of the FM KET, as shown in Fig. 7 in the grey rectangular box. The introduced features in the KET FM should be used in configuration tools to reduce the choices in the KEC FM when they are selected. It can be tricky for an FM designer to choose to add features to the KET FM depending on the relations that are observed between KET configurations and KEC configurations. This indeed propagates structuring information from the target of *provides* (KEC) to the source (KET), and in many cases, this could break the separation of concerns. In this case, it should be preferred to use cross-family constraints. Using cross-family constraints to write cross-tree constraints nevertheless requires that the concept carrying the constraint corresponds to a feature. If this is not the case the constraint should be redistributed on the existing children if any exists.

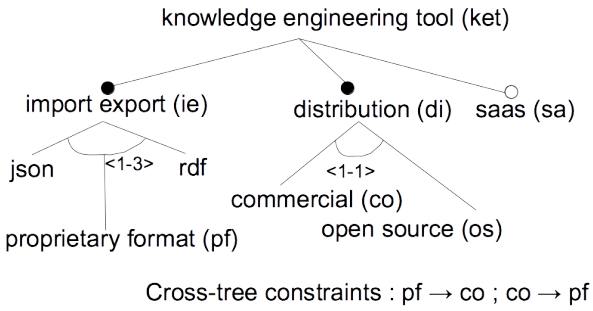


Figure 5: an FM extracted from the AOC-poset of Figure 2.

Introducing in a feature model FM_1 cross-family features that are derived from another feature model FM_2 can be seen as a way to achieve feature model composition, and especially the union operation. Traditional methods merge

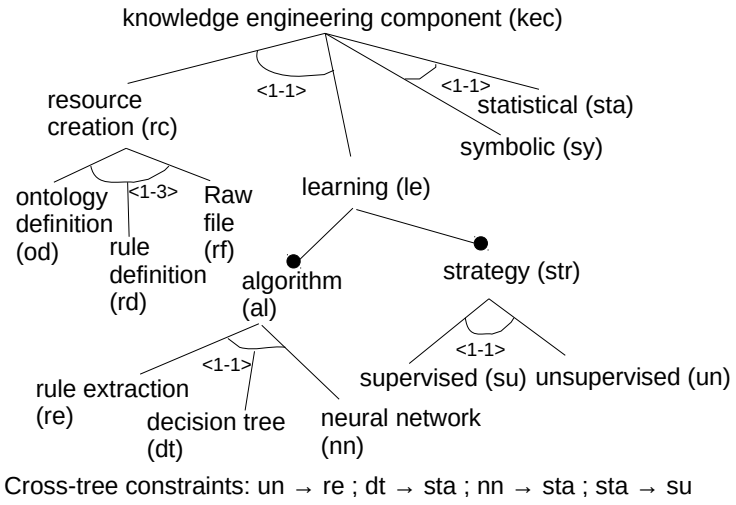
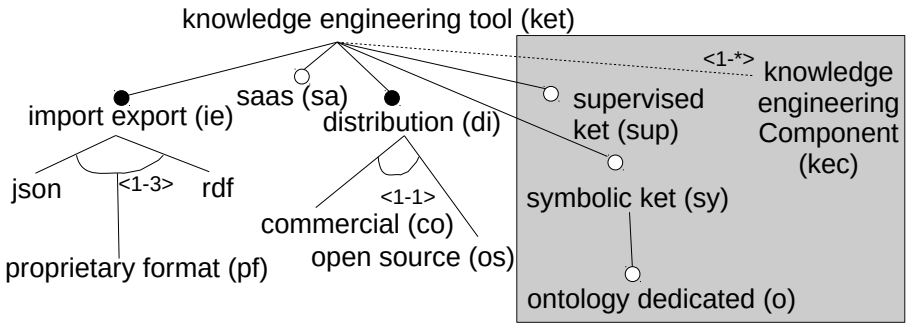


Figure 6: an FM extracted from the AOC-poset of Figure 3.

the common features of two models and keep the specific features of both initial FMs. It produces a single output FM gathering features from two different concerns into one model [ACLF10, CHMN17]. Cross-family features group features of interest from FM_2 into more "abstract features" that will enrich FM_1 with external concerns.

Recommendation Information embodied in AOC-posets can be used to guide the users in their choices when selecting a valid configuration. Concepts represent maximal groups of valid configurations sharing a maximal set of features, that may be intrinsic or cross-family. If we consider that the conjunction of a concept's features forms a query, then the concept's set of configurations can be seen as the result of this query. The concepts kept in an AOC-poset represent the maximal queries that can be formulated by a user, i.e., the maximal feature conjunctions representing each configuration subset. For instance, if a user wants to retrieve all KETs supporting the RDF format, the corresponding maximal query in Figure 4 corresponds to the concept introducing the feature *rdf* (KET_5), and the result is *ket*0-3. This can be applied for cross-family features too: it enables to query the KET configurations depending on their relationships to KECs. For instance, if a user wants to retrieve KETs providing only neural network KECs (KEC_5), this query corresponds to the concept KET_13 of Figure 4 introducing $\exists\forall provides(KEC\_5)$ and it results to the unique configuration *ket*8. The querying can benefit from the variety of quantifiers and their different semantics [BDHB18]. If, this time, a user wants to retrieve KETs providing at least one neural network KEC, we should consider the AOC-poset built with the existential quantifier.

The different quantifiers are ordered by generalization [BDHB18] thus a con-



Cross-tree constraints:
pf → co ; co → pf ; **sy → rdf ; rdf → sy ; o → saas ; sup → co**

Figure 7: FM that can be extracted from the AOC-poset of Figure 4, and cross-family information.

cept formed in an AOC-poset with a quantifier q_s can be projected in the AOC-poset formed with q_g, if q_g is more general than q_s. For instance, $\exists$ is more general than $\exists\forall$. Choosing the right quantifier can be useful to tune the degree of observed variability and the precision of the recommendation systems.

Concepts' position in the AOC-poset and to each other reveals other useful information about the structured family. Concepts on top of the AOC-poset usually possess less features and more configurations and therefore represent groups of features shared by most of the considered configurations. Dually, concepts at the bottom generally possess more features and less configurations, and represent more specialised features, i.e., the ones shared by few configurations. For instance Figure 3 reveals that there are more symbolic KECs (KEC_12) than statistical ones (KEC_8): this information may be used in recommendation systems to propose, for instance, popular features to a user. It is also noteworthy that the closer two concepts are in the structure, the more similar are their sets of features and configurations. Therefore, navigating from one concept to one of its neighbours represents modifications that can be applied on the corresponding query. This navigation has different properties depending the used conceptual structure (e.g. AOC-poset, concept lattice). Retrieving a concept based on a set of features can be seen as *querying* information from the relation concept family. Navigating from a concept to another in the structure enables a user to *explore* the set of available configurations by progressively refining the query. Let us imagine that a user wants to retrieve all the available open source KETs: this query corresponds to KET_7 of Figure 4. If the user wants to refine this query to find a more specific KET, the AOC-poset can be used to recommend the smallest modifications that may be applied: here, it proposes to either explore KETs having features $json$ and $saas$ (i.e., moving to KET_0) or explore KETs supporting rdf format and providing only symbolic KECs (i.e., moving to KET_5). Cross-family features allow to switch between AOC-posets: here, the user can "jump" to the concept referenced by $\exists\forall provides(KEC\_12)$ in the KECs' AOC-poset of

Figure 3 and then refine the corresponding query. For instance, they can choose to focus on *symbolic* KECs specialised on *ressource creation* (KEC_10) and then jump back to the previous KETs' AOC-poset, but this time in the concept introducing $\exists\forall provides(KEC\_10)$ (i.e., KET_1) with $ket0$ and $ket1$.

5 Related Work

Interconnected variability models Several tracks have been followed to represent modularity, and FMs extensions have been proposed, among which we can notice *Feature Models with References* (FMR) [CBUE02, CHE04] and *Modular Feature Models* (MFM) [BEGB11]. These two extensions enable the separation of a single FM into several FMs dedicated to sub-product lines or to separate concerns. In FMRs, a reference is a feature of an FM representing the root of another FM. Cross-FM constraints may be established between features of the two FMs. In MFMs, the authors encourage the definition of FMs modules, and the features in different FM modules are connected through cross-FM implications in *module bridges*. The approach is developped using the description logic $\mathscr{ALCH}$ for describing the FMs and the cross-tree and cross-FM constraints.

Friess et al. [FSSP07] work on modelling composition rules between different feature models representing independent product lines that can be combined. The authors define what they call a *feature configuration*, which represents a subset of an FM valid configurations satisfying some constraints. Then, they define composition rules (e.g., *uses*, *parts-of*) between feature configurations of different FMs. Feature configurations permit to finely tune the set of configurations involved in a composition rule. Similarly, Rosenmuller et al. [RSKuR08] work on reinforcing constraint expressiveness between different yet connected FMs by relying on *product line specialisation* [CHE04]. The latter was introduced by Czarnecki et al. [CHE04] as a way to prune the set of valid configurations of an FM by partially configuring this FM. Rosenmuller et al. argue that, when modelling complex product line, relying on domain constraints at the domain level is not sufficient, and that modelling constraints involving product line instances (i.e., valid configurations) is necessary. They use UML class diagrams to model these constraints on configurations, where a class represents an FM, a sub-class represents an FM specialisation and relations are used to define constraints at the configuration level. These "instance models" are used in association with classic domain models such as FMs. Urli et al. [UBC14] propose a domain model regrouping several FMs and relations between these FMs. In this approach, the domain model defines the different abstract concepts of the modelled complex product line, and each FM characterises the variability of one of these concepts. Contrarily to approaches presented before, constraints between FMs involved features and not subsets of configurations. Dhungana et al. [DSB+11] propose an approach to ease the integration of different kinds of variability models into a unique infrastructure. They present `Invar`, a framework allowing to manage a repository of different variability models. When

a new model is loaded , constraints between the new model and the existing ones need to be specified. These constraints take the form of "if *condition* then *action*". Conditions involve the selection or deselection of a feature. Actions may be to include another variability model (similar to model modularisation of [CHE04]), or to select or deselect a feature in another variability model.

Extraction of interconnected variability models Most of the existing methods for FM reverse engineering exclusively focus on boolean FMs [CW07, RPK11, ACP+12, DDH+13, HLE13, LLE14]. Becan et al [BBGA15] were the first to propose a reverse engineering method for more complex FMs taking the form of FMs with attributes. During previous work [CHN19b], we introduced the usage of Pattern Structures [GK01] to extract complex variability information in the form of logical relationships corresponding to the logical semantics of FMs extended with both multivalued attributes and UML-like cardinalities. Here we elaborate on RCA, both being possibly combined. To the best of our knowledge, there is still no work about the extraction of "cross-family" variability information.

6 Conclusion

As software systems grow and the software product line engineering is spreading, collaborative design of huge product lines which combine several concerns will become more and more critical. Complex variability can take various forms, including variability among inter-connected software families. In this paper, we address the aim to extract such *cross-family variability* from a set of inter-connected product configurations. We propose to employ Relational Concept Analysis (RCA), an extension of Formal Concept Analysis to assist this extraction. We introduce *cross-family features* and *cross-family relationships* that take the form of binary implications, mutex, co-occurrences or groups that involve both boolean features and cross-family features. Extracting such information has the inherent complexity of related data-mining methods. We expect reasonable complexity for exact extraction of binary implications, mutex, co-occurrences, and probably difficulties, if not unfeasibility, for groups, requiring approximate methods. We then discuss the possibility to use this variability information to assist the design of FMs with new kinds of references, or to explore a set of inter-connected configurations. As future work, we plan to apply the method to inter-connected product descriptions, such as connected PCMs, or by dividing large tabular descriptions, like the one produced for JHipster in [HNA+17], into separate concerns. Previous work that applied FCA and RCA to Wikipedia's or synthetic PCMs [CHG15] gave us some preliminary results on the feasibility. Exploring the effects of the different RCA scaling quantifiers on these datasets will be very informative to tune the method. We also are interested by the possibilities of composing different inter-connected FMs with the support of our method. Finally, we will study the way this cross-family variability has to be taken into account to guide the users during a product

selection/construction.

References

[ACLF10] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. Comparing approaches to implement feature model composition. In *Proc. of the 6th Europ. Conf. on Modelling Foundations and Applications (ECMFA'10)*, pages 3–19, 2010.

[ACP+12] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. On extracting feature models from product descriptions. In *Proc. of the 6th Int. Works. on Variability Modelling of Soft.-Intensive Syst. (VaMoS'12)*, pages 45–54, 2012.

[AMdSH+14] Ra'Fat Al-Msie 'deen, Abdelhak-Djamel Seriai, Marianne Huchard, Christelle Urtado, Sylvain Vauttier, and Ahmad Al-Khlifat. Concept lattices: A representation space to structure Soft. variability. In *Proc. of the 5th Int. Conf. on Information and Communication Syst. (ICICS'14)*, pages 1–6, 2014.

[AMHS+14] Ra'Fat Al-Msie'deen, Marianne Huchard, Abdelhak Seriai, Christelle Urtado, and Sylvain Vauttier. Reverse Eng. Feature Models from Soft. Configurations using Formal Concept Analysis. In *Proc. of the 11th Int. Conf. on Concept Lattices and Their Applications (CLA'14)*, pages 95–106, 2014.

[Bat05] Don S. Batory. Feature models, grammars, and propositional formulas. In *Proc. of the 9th Int. Soft. Product Line Conf. (SPLC'05)*, pages 7–20, 2005.

[BBGA15] Guillaume Bécan, Razieh Behjati, Arnaud Gotlieb, and Mathieu Acher. Synthesis of attributed feature models from product descriptions. In *Proc. of the 19th Int. Conf. on Soft. Product Line (SPLC'15)*, pages 1–10, 2015.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge Univ. Press, Cambridge, UK, 2003.

[BDHB18] Agnès Braud, Xavier Dolques, Marianne Huchard, and Florence Le Ber. Generalization effect of quantifiers in a classification based on relational concept analysis. *Knowl.-Based Syst.*, 160:119–135, 2018.

[BEGB11] Ebrahim Bagheri, Faezeh Ensan, Dragan Gasevic, and Marko Boskovic. Modular feature models: Representation and configuration. *Journal of Research and Practice in Information Technology*, 43(2):109–140, 2011.

[Bot13] Goetz Botterweck. Variability and Evolution in Systems of Systems. In *Proc. of the 1st Workshop on Advances in Systems of Systems (AiSoS'13)*, pages 8–23, 2013.

[BSA+14] Guillaume Bécan, Nicolas Sannier, Mathieu Acher, Olivier Barais, Arnaud Blouin, and Benoit Baudry. Automating the formalization of product comparison matrices. In *Proc. of the ACM/IEEE Int. Conf. on Aut. Soft. Eng., (ASE'14)*, pages 433–444, 2014.

[CBUE02] Krzysztof Czarnecki, Thomas Bednasch, Peter Unger, and Ulrich W. Eisenecker. Generative Programming for Embedded Soft.: An Industrial Experience Report. In *Proc. of the 1st ACM SIGPLAN/SIGSOFT Conf. on Generative Programming and Component Eng. (GPCE'02)*, pages 156–172, 2002.

[CHE04] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Staged Configuration Using Feature Models. In *Proc. of the 3rd Int. Soft. Product Line Conf. (SPLC'04)*, pages 266–283, 2004.

[CHG15] Jessie Carbonnel, Marianne Huchard, and Alain Gutierrez. Variability representation in product lines using concept lattices: Feasibility study with descriptions from wikipedia's product comparison matrices. In *Proc. of ws. FCA&A 2015, co-loc. 13th Int. Conf. on Formal Concept Analysis (ICFCA)*, pages 93–108, 2015.

[CHMN17] Jessie Carbonnel, Marianne Huchard, André Miralles, and Clémentine Nebut. Feature model composition assisted by formal concept analysis. In *Proc. of the 12th Int. Conf. on Evaluation of Novel App. to Soft. Eng. (ENASE'17)*, pages 27–37, 2017.

[CHN19a] Jessie Carbonnel, Marianne Huchard, and Clémentine Nebut. Modelling equivalence classes of feature models with concept lattices to assist their extraction from product descriptions. *Journ. of Syst. and Soft.*, 152:1 – 23, 2019.

[CHN19b] Jessie Carbonnel, Marianne Huchard, and Clémentine Nebut. Towards complex product line variability modelling: Mining relationships from non-boolean descriptions. *Journ. of Syst. and Soft. (doi:10.1016/j.jss.2019.06.002)*, 2019.

[CW07] Krzysztof Czarnecki and Andrzej Wasowski. Feature Diagrams and Logics: There and Back Again. In *Proc. of the 11th Int. Soft. Product Line Conf. (SPLC'07)*, pages 23–34, 2007.

[DDH+13] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. Feature model extraction from large collections of informal product descriptions. In *Proc. of the 9th Joint Meeting of the Europ. Soft.*

Eng. Conf. and the ACM SIGSOFT Symposium on the Foundations of Soft. Eng. (ESEC/FSE'13), pages 290–300, 2013.

[DSB+11] Deepak Dhungana, Dominik Seichter, Goetz Botterweck, Rick Rabiser, Paul Grünbacher, David Benavides, and José A. Galindo. Configuration of multi product lines by bridging heterogeneous variability modeling app. In *Soft. Product Lines - 15th Int. Conf., SPLC 2011*, pages 120–129, 2011.

[FSSP07] Wolfgang Friess, Julio Sincero, and Wolfgang Schroeder-Preikschat. Modelling compositions of modular embedded soft. product lines. In *Proc. of the 25th Conf. on IASTED Int. Multi-Conf.: Soft. Eng.*, pages 224–228. ACTA Press, 2007.

[GK01] Bernhard Ganter and Sergei O. Kuznetsov. Pattern Struct. and Their Projections. In *Proc. of the 9th Int. Conf. on Conceptual Struct. (ICCS'01)*, pages 129–142, 2001.

[GW99] Bernhard Ganter and Rudolf Wille. *Formal concept analysis - mathematical foundations*. Springer, 1999.

[HGR12] Gerald Holl, Paul Grünbacher, and Rick Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Information & Soft. Technology*, 54(8):828–852, 2012.

[HLE11] Evelyn Nicole Haslinger, Roberto E. Lopez-Herrejon, and Alexander Egyed. Reverse Eng. Feature Models from Programs' Feature Sets. In *Proc. of the 18th Work. Conf. on Reverse Eng. (WCRE'11)*, pages 308–312, 2011.

[HLE13] Evelyn Nicole Haslinger, Roberto Erick Lopez-Herrejon, and Alexander Egyed. On Extracting Feature Models from Sets of Valid Feature Combinations. In *Proc. of the 16th Int. Conf. on Fundamental App. to Soft. Eng. (FASE'13)*, pages 53–67, 2013.

[HNA+17] Axel Halin, Alexandre Nuttinck, Mathieu Acher, Xavier Devroey, Gilles Perrouin, and Patrick Heymans. Yo Variability! JHipster: A Playground for Web-Apps Analyses. In *11th Int. Works. on Variability Modelling of Soft.-intensive Syst.*, pages 44 – 51, Eindhoven, Netherlands, February 2017.

[JRJ94] Guy-Vincent Jourdan, Jean-Xavier Rampon, and Claude Jard. Computing on-line the lattice of maximal antichains of posets. *Order*, 11(3):197–210, 1994.

[KCH+90] Kyo Kang, Sholom Cohen, James Hess, William Novak, and A. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-021, Soft. Eng. Institute, 1990.

[LLE14] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. Feature Model Synthesis with Genetic Programming. In *Proc. of the 6th Int. Symposium on Search-Based Soft. Eng. (SSBSE'14)*, pages 153–167, 2014.

[LP07] Felix Loesch and Erhard Ploedereder. Restructuring Variability in Soft. Product Lines using Concept Analysis of Product Configurations. In *Proc. of the 11th Europ. Conf. on Soft. Maintenance and ReEng. (CSMR'07)*, pages 159–170, 2007.

[MZB+17] Jabier Martinez, Tewfik Ziadi, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Bottom-up technologies for reuse: Automated extractive adoption of soft. product lines. In *Comp. Proc. of the 39th Int. Conf. on Soft. Eng. (ICSE'17)*, pages 67–70, 2017.

[RHHNV13] Mohamed Rouane-Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Math. and Artificial Intelligence*, 67(1):81–108, 2013.

[RPK11] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. Extraction of feature models from formal contexts. In *Works. Proceedings (Volume 2) of the 15th Int. Conf. on Soft. Product Lines (SPLC'11)*, pages 4:1–4:8, 2011.

[RSKuR08] Marko Rosenmüller, Norbert Siegmund, Christian Kästner, and Syed Saif ur Rahman. Modeling dependent software product lines. In *Proc. of the GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering (McGPLE'08)*, pages 13–18, 2008.

[SDM+11] Camille Salinesi, Olfa Djebbi, Raúl Mazo, Daniel Diaz, and Alberto Lora-Michiels. Constraints: The core of product line eng. In *Proc. of the Fifth IEEE Int. Conf. on Research Challenges in Information Science (RCIS'11)*, pages 1–10, 2011.

[SHTB07] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007.

[SLB+11] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. Reverse eng. feature models. In *Proc. of the 33rd Int. Conf. on Soft. Eng., (ICSE'11)*, pages 461–470, 2011.

[SRA+14] Steven She, Uwe Ryssel, Nele Andersen, Andrzej Wasowski, and Krzysztof Czarnecki. Efficient synthesis of feature models. *Information & Soft. Technology*, 56(9):1122–1143, 2014.

[SSS17] Anas Shatnawi, Abdelhak-Djamel Seriai, and Houari A. Sahraoui. Recovering soft. product line architecture of a family of object-oriented product variants. *Journal of Syst. and Soft.*, 131:325–346, 2017.

[UBC14] Simon Urli, Mireille Blay-Fornarino, and Philippe Collet. Handling complex configurations in software product lines: a tooled approach. In *18th Int. Soft. Product Line Conf. (SPLC '14)*, pages 112–121, 2014.