Sampling Cardinality-Based Feature Models

Lukas Güthing
lukas.guething@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Mathis Weiß
mathis.weiss@uni-siegen.de
University of Siegen
Siegen, Germany

Ina Schaefer
ina.schaefer@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Malte Lochau
malte.lochau@uni-siegen.de
University of Siegen
Siegen, Germany

ABSTRACT

The goal of sample-based testing of variant-rich software systems is to reduce usually very large configuration spaces to significantly smaller, yet still representative subsets of configurations to be tested for quality assurance. Recent sampling techniques and tools are restricted to finite-dimensional, Boolean configuration spaces specified by a feature model. However, in many modern application domains like cloud computing and cyber-physical systems, customers not only decide about the presence or absence of features in a configuration but also about the multiplicity (number of instances) of configurable resources. Cardinality-based feature models extend Boolean feature models by cardinality annotations and respective constraints to enable multiple, and even potentially a-priori unbounded, copies of features and their respective sub-trees. The resulting infinite and inherently non-convex configuration spaces are no longer tractable by established sampling criteria and corresponding sampling algorithms for Boolean feature models like pairwise feature interaction coverage. In this paper, we first revisit the subtleties of the configuration semantics of cardinality-based feature models. We propose novel sampling criteria explicitly taking multiplicity of feature selections into account. Finally, we present evaluation results gained from applying our tool implementation to a collection of example models, showing applicability of the proposed approach.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

software product lines, software variability, sampling, cardinality-based feature models

VaMoS 2024, February 07–09, 2024, Bern, Switzerland

ACM ISBN 979-8-4007-0877-0/24/02…$15.00
https://doi.org/10.1145/3634713.3634719

ACM Reference Format:
Lukas Güthing, Mathis Weiß, Ina Schaefer, and Malte Lochau. 2024. Sampling Cardinality-Based Feature Models. In *18th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS 2024), February 07–09, 2024, Bern, Switzerland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3634713.3634719

1 INTRODUCTION

Background and Motivation. Many modern software systems are highly configurable to diverse customer needs and application platforms. Feature models provide a graphical modeling language to describe valid configuration spaces from a problem-oriented point of view [7]. Feature models organize the set of features (user-configurable increments of functionality) in a tree-like hierarchy with further notational elements to define configuration constraints. Today, feature models enjoy a rich background theory and mature tool support. For instance, feature models serve as input for many recent tools for *sample-based testing* of variant-rich software [21]. The goal of sampling is to systematically reduce the inherently very large configuration space to significantly smaller, yet still representative subsets (samples) of configurations to be tested for quality assurance. As an effectiveness measure, recent sampling techniques and tools apply combinatorial coverage criteria inspired by functional black-box testing. In this way, fully automated sample selection can be performed. The widely considered k-wise feature interaction coverage criteria requires every possible on/off combination for any subset of k features to be covered in the sample [21].

In its original form, feature models and sampling tools are limited to *finite Boolean* configuration spaces: the number of features is fixed during domain analysis and every feature constitutes an on/off configuration decision of some optional functionality. However, in modern domains like cloud computing and cyber-physical systems, customers may also decide about the *multiplicity* (number of instances) of configurable resources. Cardinality-based feature models (CFM) extend Boolean feature models by cardinality constraints for features [3, 12, 23]. These constraints specify cardinality intervals on the number of feature instances: interval $\langle l, u \rangle$ denotes that feature f must be selected at least l and at most u times. Wildcard $*$ may be used instead of a fixed integer value, allowing a-priori unbounded number of feature instances. Moreover, for every instance of a feature f, the configuration contains an individually configurable copy (clone) of the whole sub-tree of f. This gives rise to *infinite* and *non-convex* configuration spaces of CFMs which are

no more tractable by classical sampling criteria and algorithms of Boolean feature models due to the following reasons.

- Applying classical k-wise feature interaction criteria as-is only covers 0 and 1 in k-wise feature combinations. Multiple copies of features are not sampled. However, a naive generalization of these classical criteria to k-wise combinations of *all* valid feature multiplicities is not feasible leading to very large (or even infinite) samples.
- The sampling problem for Boolean feature models is well-defined as the configuration semantics are clearly understood and canonically formalized. In contrast, no generally agreed formal configuration semantics for CFMs exist so far.
- Most recent sampling tools rely on encodings into propositional formulae, which facilitates applications of off-the-shelf SAT solvers for automated and highly scalable sample generation. This is not feasible for CFMs due to their increased expressiveness, which requires more involved mathematical models like ILP or SMT [15, 22].

Research Questions. We tackle the following research questions to enable sample-based testing for configurable software with configuration spaces specified by CFMs.

(RQ1) How to define reasonable sampling criteria for CFMs?
(RQ2) How to perform automated sample generation for CFMs using existing tools?

Concepts and Contributions. We first revisit syntactic and semantic subtleties of CFMs and precisely characterize the constructs supported by our approach. Our goal is to facilitate expressiveness, yet enable efficient encoding into existing modeling and instance generation tools to perform scalable sample generation using current tools [1, 15].

Second, we propose a generalization of k-wise sampling criteria for CFMs. We employ a normal form representation that makes explicit crucial semantic singularities (interval gaps) relevant for sampling [23]. To handle infinite configuration spaces, we employ the M-boundedness property of CFMs, ensuring the existence of a finite bound M enclosing all singularities. Based on this representation, we utilize principles of boundary-interior testing [4] to sample presumably critical feature multiplicities. Those criteria are based on the empirically approved assumption that faults are often located at the boundary of intervals. We make the following contributions.

We define **novel sampling criteria** for CFMs. To this end, we first clarify the configuration semantics for CFMs considered in our framework. The criteria are well-defined for feature multiplicity intervals, including interval gaps as well as right-open intervals.

We present a **prototype implementation** based on Clafer as textual input language for cardinality-based feature models and Clafer Instance Generator for sample generation.

We provide experimental **evaluation results** gained from applying our tool to examples. Due to the novelty of our approach, we focus our evaluation on applicability and scalability.

2 BACKGROUND

In this section, we introduce a running example to explain the foundations of cardinality-based feature models (CFM), compared to *Boolean* feature models [7], illustrating subtleties of the enhanced configuration semantics of CFM. We derive research challenges in adapting sampling criteria from Boolean configuration spaces to configuration spaces shaped by CFM.

Running Example. Our running example is inspired by a cloud-based multi-player game [13]. Each player of this game interacts with a personal device. A central node administers the game and handles inter-team communication. Players are organized into teams, each team has a leader who bundles communication. Communication is separated into *intra-* and *inter*-team communication, the first bypassing the central node, the second routed via the central node. Each leader can decide between either a scattered or a specific strategy. The first strategy is not available for games with five or fewer teams and requires broadcasts, whereas the second is not available for games with exactly five teams due to game-design-specific reasons. For a few players, Bluetooth is sufficient for intra-team comm., whereas for larger amounts, WiFi is required.

Cardinality-Based Feature Models. The described game is configurable in various ways. Besides *Boolean* configuration options (e.g., choosing between two alternative strategies), most of the variability concerns the *multiplicity* of game components (e.g., number of players or teams). In addition, for each of the feature instances (copies, clones), further instance-specific decisions can be made (e.g., intra-team comm. for each team instance).

Purely Boolean feature models like feature diagrams in FODA notation [7] are not expressive enough to capture feature multiplicity constraints. We instead use a cardinality-based feature model (CFM) as shown in Figure 1 for our running example [2]. A CFM constitutes a conservative extension of Boolean feature diagrams in FODA notation, employing a tree-like hierarchy. The features located below the root node partition the configuration model into two subtrees, where the left sub-tree contains the configuration options for the central node and the right for the separate teams. Based on the tree layout, CFM further comprises different syntactic constructs to restrict feature multiplicity (i.e., the valid number of feature instances within configurations). These restrictions are based on the unifying concept of *cardinality constraints*. For instance, feature *Team* is annotated by a *feature instance cardinality* $\langle 2, * \rangle$ consisting of a lower bound $l = 2$ and an upper bound $u = *$. This *cardinality interval* (l, u) restricts the number of instances of feature *Team*. The lower bound may be any natural number, including 0. The upper bound also includes the distinct symbol $*$ to permit any natural number of instances. For each selected instance of *Team*, the corresponding subtree is cloned to enable instance-specific configurations of every team. For instance, if we select three instances of feature *Team*, the configuration contains instance-specific copies of the sub-features *Members* and *Intra T. Comm.* and their sub-trees.

In this way, a cardinality interval imposed on feature instances generalizes the notions of mandatory features (interval $(1, 1)$) and optional features (interval $(0, 1)$). Cardinality intervals are further used to restrict the selection of feature types in a group, thus generalizing the notions of alternative-groups and or-groups from FODA notation [14]. The *group type cardinality* $[2, 2]$ attached to the bow below the root feature denotes that at least and at most two instances with distinct types of features have to be selected. Hence, at least one feature of type *Central Node* and of type *Team* must

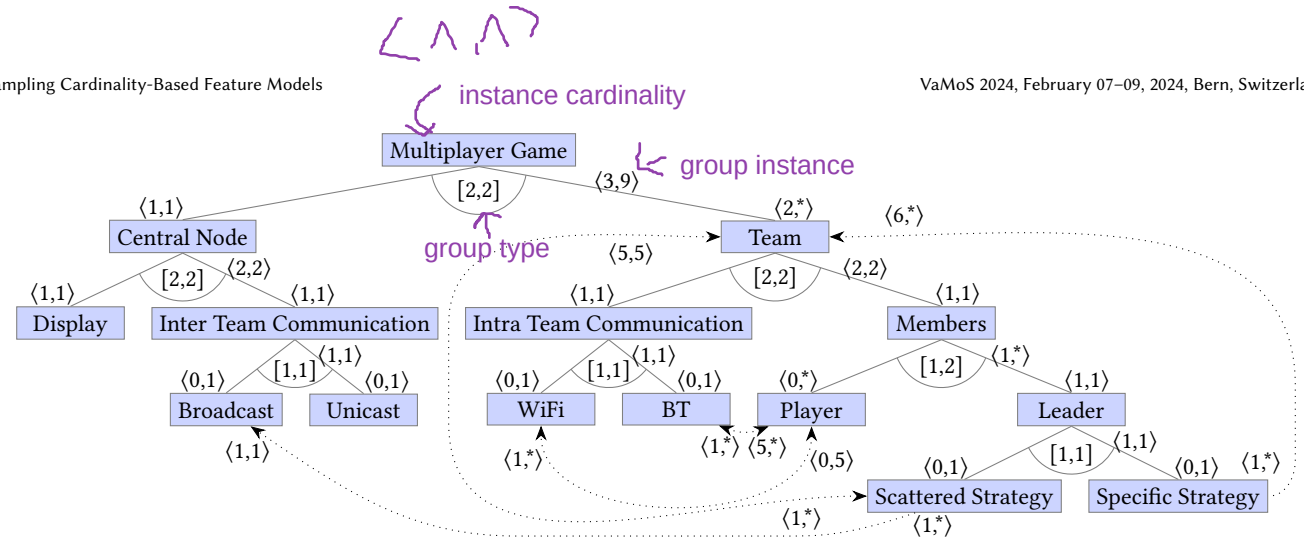

Figure 1: Cardinality-Based Feature Model of a Competitive Mobile Multi-Player Game

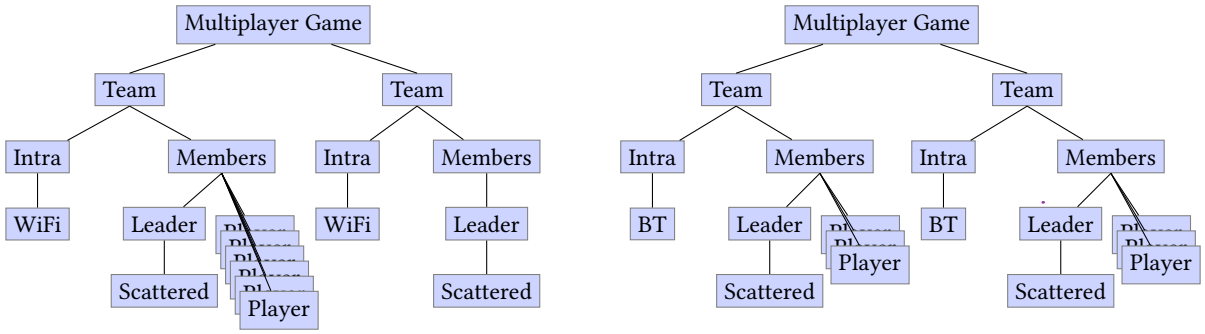

Figure 2: Excerpt from Two Valid Configurations

be selected. In addition, the group below the root feature is annotated with the *group instance cardinality* $\langle 3, 9\rangle$, restricting the total number of instances of any type selected from that group. For instance, selecting two instances of *Central Node* and of *Team* sums up to four instances, satisfying the group instance cardinality. However, this example demonstrates the subtle interplay between cardinality constraints: selecting two times feature *Central Node* as well as feature *Team* would satisfy the group instance cardinality constraint and the group type cardinality constraint $[2, 2]$, whereas the feature instance cardinality $\langle 1, 1\rangle$ does not permit more than one instance of *Central Node*.

Finally, CFMs also include *cross-tree constraint* by means of require- and exclude-edges annotated by cardinality intervals at both ends [12]. For instance, the require-edge from *Specific Strategy* to *Team* is annotated by $\langle 1, *\rangle$ and $\langle 6, *\rangle$. This means that if at least one instance of *Scattered Strategy* is selected, then the number of instances of *Team* must fall into interval $(6, *)$. Exclude-edges are interpreted accordingly.

Modeling Cardinalities in Variability Models. Apart from *Boolean* feature models, there are other variability models, some of which already support cardinalities. *UVL* and *Clafer* are textual modeling languages for variability modeling. UVL, or the Universal Variability Language [18–20], is a project aimed to be a common intermediate language for variability models. UVL uses indents to model the hierarchy of the variability model. Additionally, UVL supports feature instance and group type cardinalities, but no group instance cardinalities (cf. section 2). However, UVL is mainly used to *express* variability, not to *analyze* it. Therefore, there are only a few tools to analyze UVL variability models. This lack of analysis support mainly stems from the sheer expressiveness of UVL models, with complex constraints and $*$-cardinalities leading to potentially undecidability problems. Therefore, there is no general sampling approach for UVL models yet.

Clafer [1, 5] is a modeling language to model software structure, behavior, and variability. Like in UVL, indents represent the hierarchy of the features in Clafer. Clafer also supports feature instance and group type cardinalities and also lacks group instance cardinality. Clafer has some built-in variability analyses: Clafer can detect dead feature intervals, a generalization of dead features in FODA models, false optional and core features, falsely unbounded feature instance intervals, and false upper/lower bounds. Clafer also supports feature attributes and multi-objective optimization. These analyses are based on ILP (integer linear programming), Alloy, and ChocoSolver. Additionally, Clafer also has an instance generator. Due to the high complexity the cardinalities introduce, however, Clafer does not have the same range of analyses that is available for

Boolean feature models, like calculating the number of all possible configurations or sampling.

Cardinality-Based Configuration Spaces. The configuration spaces of *Boolean* feature models comprise all valid assignments of Boolean values *true* (present) or *false* (absent) to each feature such that all configuration constraints are satisfied. The configuration semantics of CFM instead assigns to each feature a natural number k denoting number of instances (0 in case of absence) of that features in a configuration. In this sense, the notion of configuration of CFM generalizes that of Boolean feature models from (sub-)sets of (selected) features to *multisets* of features. However, this notion of CFM configuration might be considered ambiguous as it does not preserve the (potentially relevant) parent-child relationship for each feature instance [23]. To illustrate this issue, consider the excerpts from two configurations of our example in Fig. 2. We abbreviated some feature names and omitted the left subtree. The instance on the left-hand side is a game with two teams. The first team has six players in addition to the leader, whereas the second team only consists of a leader. The instance on the right-hand side is also a game with two teams and an overall number of six players, where the players are equally distributed over both teams (i.e., three players each). Only considering the multiset of both games (total number of players), the composition of the teams is indistinguishable. An instance-based configuration notion contains the number of feature instances and stores the relationship of every instance to its parent instance, distinguishing between both team compositions.

In any case, for a CFM configuration to be valid, it has to satisfy all cardinality constraints of the CFM. Again, depending on the notion of configuration, cross-tree constraints may be interpreted locally or globally [8]. In a local interpretation, the configuration on the left is invalid as the team with 0 players does not satisfy the exclude constraint between *WiFi* and *Players*: an instance of *WiFi* may be only selected if more than five players are part of the team. In contrast, the right configuration is valid. For both teams, an instance of *BT* is selected in compliance with the number of players (3). In case of a global interpretation (multiset-based), just the configuration on the right is valid. Considering the total number of players (6) requires the selection of *WiFi*.

The loss of semantic precision caused by multiset semantics and global interpretation is justified by the improved scalability of automated analyses for CFM due to a significantly reduced impact of the combinatorial explosion problem [23]. However, the configuration space of CFM remains large and is infinite if the CFM contains an unbounded interval (e.g. * for feature *Player*). In addition, the interplay between different kinds of cardinality constraints might lead to non-convex configuration spaces containing singularities (e.g., interval gap of feature *Player* excluding multiplicity value 5).

Research Challenges. For classical Boolean feature models, sampling criteria and corresponding sample-generation techniques and tools are widely considered in practice to counter-act the combinatorial explosion problem for quality assurance of configurable systems [21]. Adapting these concepts to CFM is not straightforward. For instance, uniform random sampling, a basic sampling criterion, may be adapted to CFMs by expanding the random choosing of 0 or 1 instance for each feature, to randomly choose a multiplicity value m from the valid cardinality interval for each feature. Beyond uniform random sampling, k-wise combinatorial feature interaction coverage constitutes the most established sample criterion, guaranteeing that each valid combination of k feature selections is sampled. In this regard, pairwise sampling (i.e., $k = 2$) has been shown to be a promising trade-off between efficiency (sample size) and effectiveness (error coverage) in practice. However, naively generalizing this criterion to CFM is not feasible as treating every valid pair of multiplicity values of any k features as a unique interaction excessively blows up the sample size, already in case of $k = 1$. Moreover, for unbounded cardinality intervals, the sample size would be, by definition, *infinite*. In addition, feature multiplicities may introduce novel types of interactions. Multiple instances of the *same* feature may interact with each other. To summarize, we have to tackle the following research challenges (RC) to enable CFM sampling.

- **(RC1) Infinite configuration spaces.** We have to impose finite bounds on a-priori infinite configuration spaces to make combinatorial coverage criteria applicable for CFM.
- **(RC2) Non-convex configuration spaces.** We have to handle inherently non-convex configuration spaces which obstructs automated sample extraction. In addition, we have to ensure that potentially erroneous realizations of configurations located around intervals gaps are sufficiently covered by the generated samples.
- **(RC3) Excessively large configuration spaces.** We have to define adapted notions of combinatorial coverage criteria to extract samples with reasonable sizes from the now bounded, but still excessively high number of multiplicity combinations.

3 CARDINALITY-BASED FEATURE MODELS

In this section, we provide precise definitions of the abstract syntax and configuration semantics of CFM which are based on Weckesser et al. [23]. This builds the conceptual basis to define novel sampling criteria for CFM in the next section.

3.1 Abstract Syntax

As illustrated in the previous section, CFMs generalize the notion of configurations from Boolean decisions to *multiplicities* of feature instantiations corresponding to non-negative integer values. Hence, a feature is either *absent* (value 0) or *present* k times ($k \geq 1$) in a CFM configuration. In addition to the number of instances per feature, a CFM configuration further consists of a relation defining which child-feature instances are related to which parent-feature instances.

The configuration constraints imposed by a CFM are likewise generalized from propositional formulas (Boolean feature models) to *cardinality constraints*. Cardinality constraints express *cardinality intervals* denoting valid value ranges for the multiplicity of feature instantiations. *Simple cardinality intervals* are defined as pairs (l, u) of non-negative integer values with $l \leq u$, where l denotes the *lower* bound and u denotes the *upper* bound of the interval. Multiplicity value k satisfies (is contained in) an interval (l, u) if $l \leq k$ and $k \leq u$ holds, which we denote by $k \in (l, u)$ in the following. As a possible upper bound of cardinality intervals, we allow the distinguished

symbol *, denoting unbounded (unconstrained, right-open) intervals satisfied by any possible multiplicity (i.e., $k \leq {}^*$ holds for any value k).

Multiple disjoint simple cardinality intervals may be combined into a *compound cardinality interval* to implicitly exclude particular sub-intervals (gaps) from a set of valid multiplicity values. In our running example (Figure 1), each team must select one strategy. However, if the number of teams is exactly 5, neither the feature *Scattered Strategy* nor the feature *Specific Strategy* may be selected. Thus, selecting a strategy for a game with five teams is not possible, leading to an invalid configuration. To make this gap explicit, we refine the feature instance cardinality of *Team* to the compound cardinality interval $\langle 0, 4\rangle\langle 6, *\rangle$. It is reasonable to require a compound interval to be defined as concise as possible, forbidding non-disjoint simple intervals.

Definition 3.1 (Cardinality Interval [23]). The set of cardinality intervals is defined as $\mathcal{I} \subset \mathbb{N}_0 \times (\mathbb{N}_0 \cup \{*\})$, where $(l, u) \in \mathcal{I}$ *iff* $l \leq u$ holds. The set $\mathcal{L} \subset_{fin} 2^{\mathcal{I}}$ of compound cardinality intervals contains all finite subsets $L \in \mathcal{L}$ of $\mathcal{I}$ such that for all pairs $(l_i, u_i) \in L, (l_j, u_j) \in L, i \neq j$, either $l_i > u_j$, or $u_i < l_j$ holds.

We use the notation $k \in \mathcal{L}$ for compound intervals $\mathcal{L}$ in obvious ways (i.e., requiring $k \in (l_i, u_i)$ for exactly one simple interval contained in $\mathcal{L}$).

As illustrated by our running example (Figure 1), cardinality-based feature models provide syntactic constructs to express *cardinality constraints*, which are semantically interpreted as cardinality intervals. Like in Boolean feature models, CFMs define a tree-like hierarchy describing a parent-child relation $\prec_F$ (decomposition relation) on the set of features F. Based on this hierarchy, cardinality intervals occur in five different types of cardinality constraints as defined by the following functions on F.

- Function λ_I^F assigns an *instance cardinality* interval to each feature, constraining the multiplicity of individual features.
- Function λ_I^G assigns a *group instance cardinality* interval to each feature, constraining the sum of multiplicities of its direct child feature instances.
- Function λ_T^G assigns a *group type cardinality* interval to each feature, constraining the number of different types of its direct child feature instances.
- Relation ϕ_R defines *require-edges* between some pairs of features, enforcing certain combinations of multiplicities among these features.
- Relation ϕ_X defines *exclude-constraints* between some pairs of features, forbidding certain combinations of multiplicities among these features.

Summarizing these ingredients, we obtain the following definition of the abstract syntax of cardinality-based feature models.

Definition 3.2 (CFM Abstract Syntax [23]). A cardinality-based feature model (CFM) defined over a non-empty, finite set F is a tuple $(\prec_F, \lambda_I^F, \lambda_I^G, \lambda_T^G, \phi_R, \phi_X)$, where

- $\prec_F \subseteq F \times F$ is a feature decomposition relation,
- $\lambda_I^F : F \to \mathcal{L}$ is a feature instance cardinality function,
- $\lambda_I^G : F \to \mathcal{L}$ is a feature group instance cardinality function,
- $\lambda_T^G : F \to \mathcal{L}$ is a feature group type cardinality function,
- $\phi_R \subseteq F \times \mathcal{L} \times \mathcal{L} \times F$ is a feature instance require-edge cardinality relation, and
- $\phi_X \subseteq F \times \mathcal{L} \times \mathcal{L} \times F$ is a feature instance exclude-edge cardinality relation.

To be *syntactically well-formed*, the feature decomposition relation should form a *finite-rooted tree* on F, meaning that each feature has exactly one predecessor, except for the distinguished root feature f_r. We further assume $\lambda_I^F(f_r) = (1, 1)$ (i.e., there is exactly one instance of the root feature in every configuration) and both kinds of group cardinality intervals assigned to leaf features (feature without child features) must be empty (i.e. (0,0)).

3.2 Configuration Semantics

Based on the abstract syntax, we define a *configuration* of a CFM to be a pair of a *multiset* M over feature set F and a parent-child feature instance relation $\prec_F^M$) on M. Formally, a *multiset* $M : F \to \mathbb{N}_0$ over F is a mapping from F into the non-negative integers, where $M(f)$ denotes the *multiplicity* of $f \in F$. We use the notation $\#f = M(f)$ to refer to the multiplicity of f in multiset M whenever M is clear from context. To refer to the i-th instance, $1 \leq i \leq \#f$, of feature f in multiset M, we write $f^i \in M$. By $\prec_F^M \subseteq M \times M$, we denote the feature instance decomposition relation on multiset M in accordance to the feature decomposition relation $\prec_F$ on F such that: $f^i \prec_F^M f'^j \Rightarrow f \prec_F f'$ holds.

Before we formally define a valid CFM configuration as a pair $(M, \prec_F^M)$, we intuitively explain the configuration semantics using the examples in Figs. 1 and 2, referred to as $(M, \prec_F^M)$ and $(M', \prec_F^{M'})$. Let us consider feature *Members* and its sub-tree. In both configurations, two instances of this feature are present (i.e., $M(Members) = M'(Members) = 2$) and corresponding copies of the sub-tree are related by $\prec_F^M$ and $\prec_F^{M'}$, respectively. Due to the feature instance cardinality constraint $(1, 1)$ for feature *Leader*, each instance of *Member* must be related to exactly one instance of *Leader* which holds in all four cases. In contrast, feature *Player* is annotated with constraint (0,∗) which is satisfied by every number of instances of *Player*. In both configurations, it holds that $M(Player) = M'(Player) = 6$, where in the left one, $\prec_F^M$ relates all six instances to the subtree of the first instance of *Members* and none to the second, whereas $\prec_F^{M'}$ relates three to the first and to the second one. Let us next have a look at the group instance cardinality for feature *Members*: the interval $(1, {}^*)$ requires that at least one instance of some child feature of *Members* is present in a configuration. Due to the mandatory presence of one instance of feature *Leader*, this is satisfied in all sub-trees of both configurations. The group type cardinality $(1, 2)$ *Member* requires that instances from at least one, and at most two, types of child features of *Members* are present. Again, this constraint is always satisfied as the mandatory instance of feature *Leader* ensures that the lower and upper bound cannot be exceeded as *Members* only has two different children.

In contrast to the previously described constraints which are checked locally (i.e., for every feature instance sub-tree in separate), we interpret cross-tree constraints globally (i.e., on the overall number of selected feature instances). For instance, for the require-constraint from *Specific Strategy* to *Team* to be satisfied, it must hold that if the overall number of instances of *Specific Strategy* falls into $(1, *)$, then the overall number of instances of *(Team)* must fall

into $(6, *)$. However, this condition is obviously satisfied as in both configurations, *Specific Strategy* is absent. Similarly, the exclude-constraint between *Player* and *BT* forbids the global number of instances of *Player* to fall into $(5, *)$ if that of feature *BT* falls into $(1, *)$ and vice versa. As the global number of instances of type *Player* is 6 in both configurations, this forbids the selection of any instances of type *BT*, causing the right configuration to be invalid.

We formally define configuration semantics of CFM as follows.

Definition 3.3 (CFM Configuration Semantics). A *configuration* $(M, \prec_F^M)$ of a cardinality-based feature model $(\prec_F, \lambda_I^F, \lambda_I^G, \lambda_T^G, \phi_R, \phi_X)$ is *valid* if the following properties hold.

- $M(f_r) = 1$.
- If $f'^i \prec_F^M f^j$, then $f' \prec_F f$ and $(\prec_F^M)^+$ forms a rooted tree on M.
- If $f'^i \in M$, then for each $f \in F$ with $f' \prec_F f$ it holds that $|\{f^j \in M | f'^i \prec_F^M f^j\}| \in \lambda_F^I(f)$.
- If $f'^i \in M$, then it holds that $|\{f^j \in M | f'^i \prec_F^M f^j\}| \in \lambda_G^I(f')$.
- If $f'^i \in M$, then it holds that $|\{f \in F | \exists f^j \in M : f'^i \prec_F^M f^j\}| \in \lambda_G^T(f')$.
- If for each $f, f' \in F$ with $(f, \mathcal{L}, \mathcal{L}', f') \in \phi_R$, it holds that $\#f \in \mathcal{L}$ implies $\#f' \in \mathcal{L}'$.
- If for each $f, f' \in F$ with $(f, \mathcal{L}, \mathcal{L}', f') \in \phi_X$, it does not hold that $\#f \in \mathcal{L}$ and $\#f' \in \mathcal{L}'$.

It is important to note that different valid configurations $(M, \prec_F^M)$, $(M, \prec_F^{M'})$ may exist that only differ in the relations $\prec_F^M$ and $\prec_F^{M'}$, while their multisets are equal. Concerning our example, the same global number of instances of type *Player* may be selected in two different configurations (e.g., 6), while the composition between teams may vary. However, if we omit the relation $\prec_F^M$ and only consider multiset M, this does not guarantee that every possible other relation $\prec_F^{M'}$ also leads to a valid configuration. For instance, having a configuration with two instances of *Team* and *Leader*, it is crucial for validity that every *Team* instance is related to one *Leader* instance, whereas a configuration in which one *Team* has two *Leaders* and one none is invalid. In other words, one multiset-based configuration may correspond to multiple valid as well as multiple invalid instance-based configurations.

4 SAMPLING CARDINALITY-BASED FEATURE MODELS

In this section, we describe how to adapt Boolean sampling criteria to CFM. We first consider a direct adaption and illustrate why this is, in general, infeasible. We then propose possible solutions to obtain effective and scalable sampling criteria for CFM. We focus on one-wise sampling criteria, considering each feature individually, and briefly describe how to generalize to the k-wise case.

4.1 Valid Configuration Spaces of CFM

For a generally very large *valid configuration space* C, the goal of Boolean sampling criteria is to select a small, yet representative, subset $\mathcal{S} \subseteq C$ of configurations.

The definition of a *valid configuration* $C = (M, \prec_F^M)$ of CFM according to Def. 3.3 *shapes the valid configuration space* C (i.e., the set of all valid configurations) of a given CFM. Similar to Boolean feature models, we define some semantic properties of CFM such as *consistency* (i.e., does C contain at least one valid configuration?). Our running example in Fig. 1 is obviously consistent, as we have already described a valid configuration (see Figs. 1 and 2). In addition, CFMs exhibit semantic properties not present in Boolean feature models. For instance, the occurrence of * in cardinality constraints may lead to *unbounded* models (i.e., C contains an infinite number of valid configurations). Our running example in Fig. 1 is indeed unbounded due to *Player*. Finally, we generalize the notion of *dead features* from Boolean feature models to *dead cardinality* in CFM: Cardinality $k \in \lambda_F^I(f)$ is dead for feature f if C does not contain any configuration $C \in C$ in which $\#f = k$ holds. In our running example, feature *Team* is annotated by feature instance cardinality of $(2, *)$. However, due to group instance cardinality constraint $(3, 9)$ and the group type cardinality constraint $(2, 2)$, at most 8 instances of this feature may be selected. Thus, the whole cardinality interval $(9, *)$ for feature *Team* is dead.

4.2 One-Wise CFM Sampling

In case of Boolean feature models, one of the most basic sampling criteria (besides uniform random sampling) is *one-wise* combinatorial feature interaction coverage. This criterion requires that every valid valuation (i.e., selection (0) and deselection (1)) of every feature $f \in F$ is covered by at least one configuration $C \in S$ in a sample $S \subseteq C$. Adapting this notion for CFM would require covering every possible *multiplicity* value $\#f \in \{0, 1, ..., k\}$ of every feature f across the CFM sample. However, this adaption is not feasible for the following reasons, corresponding to the research challenges described in Sect. 2:

(RC1) If the valid cardinality interval of f is *unbounded*, no maximum value k exists and the resulting samples $\mathcal{S}$ are infinite.

(RC2) If the valid cardinality interval of f contains *dead cardinalities*, then some values between 0 and k cannot be covered in a sample.

(RC3) Even if the valid cardinality interval of f contains only finitely many values, the number of the k non-dead cardinality values of f may still be too large to be completely sampled.

All these issues are present in our running example in Fig. 1. In the following, we propose counter-measures to tackle each of the research challenges.

RC1: Bounded Configuration Spaces. To handle infinite configuration spaces, replace all occurrences of $*$ in feature cardinality constraints $(l, *)$ by proper finite upper bound values. We distinguish between *false unbounded* and *true unbounded* feature instance cardinalities.

An unbounded instance cardinality interval $(l, *)$ of a feature f is factually bounded to a maximum valid multiplicity value m due to other constraints. In this case, we can safely replace $(l, *)$ by (l, m) without changing the valid configuration space. This is actually a special case of what we will discuss for RC2 (see below) as every cardinality $k > m$ of feature f is *dead*.

Otherwise, the instance cardinality interval $(l, *)$ of f is *true unbounded*, as every multiplicity value $k > l$ is indeed valid for f.

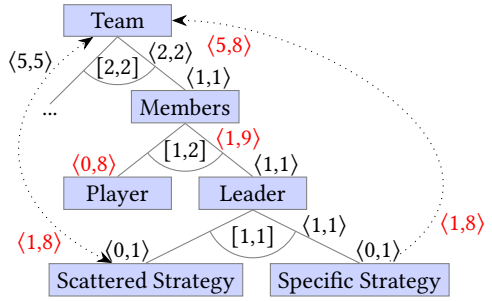

Figure 3: Normalized Excerpt of the Running Example

To handle these cases, we rely on a useful semantic property of CFM [9, 23].

Proposition 4.1. *Given a CFM over F according to Def. 3.2, there exists a finite integer value M such that for each dead cardinality value k of every unbounded feature $f \in F$, it holds that $k \leq M$.*

This so-called Big-M-value can be used as an upper-bound value replacing * in every true unbounded instance cardinality interval $(l, *)$. This alters the valid configuration space to a finite sub-space. However, it is ensured that this sub-space covers all presumably relevant (i.e., non-convex) regions of the whole configuration space.

RC2: Normal Form Representation. To avoid invalid configurations in one-wise samples, we have to make interval gaps explicit caused by dead cardinality values, potentially hidden in simple feature instance cardinality intervals. To this end, we employ another useful property of CFMs [23].

Proposition 4.2. *Given a CFM over F according to Def. 3.2, there exists an equivalent CFM' without dead feature instance cardinalities.*

The transformation of a CFM into its so-called *normal form* CFM' includes replacing (1) false unbounded cardinality constraints $*$ by factual upper bounds m and (2) dead cardinality values within simple intervals by compound intervals excluding these values.

Figure 3 shows an excerpt from the normalized CFM' for our running example CFM, where all transformations are highlighted in red. For instance, the feature instance cardinality of *Team* has been replaced by the compound interval $\langle 0, 4\rangle\langle 6, 8\rangle$ excluding the interval gap 5 and replacing the false unbounded upper bound by 8.

A safe global upper bound value M can be approximated by multiplying the upper bounds of all feature instance cardinalities along all branches containing no * from the root to all leaf nodes, and to pick the maximum value. For our example, the value for M results from the branch *Multiplayer Game*, *Team*, *Members*, *Leader*, *Scattered Strategy* which yields $1 * 8 * 1 * 1 * 1 = 8$. All occurrences of $*$ in cross-tree constraints may be replaced by M, while all group instance cardinalities having upper bound $*$ may be replaced by the sum of the upper bounds of the group members.

After this transformation, we obtain a CFM' with a finite valid configuration space C', for which we are able to calculate the size. For instance, for two teams each having up to 8 possible multiplicity values for feature *Player* (i.e., 0 to 4 and 6 to 8), the number of possible combinations for two teams is $\binom{8+2-1}{2} = 36$. If we sum this up for any valid number of teams, we get a total number of $36 + 120 + 330 + 62,016 + 191,862 + 544,793 = 799,157$ configurations part of the valid configuration space.

We next describe two measures of our one-wise coverage for CFM to significantly reduce the sample size compared to the size of the valid configuration space.

RC3: Multiset Coverage. As described in Sect. 3.2 and illustrated in Fig. 2, the valid configuration space of a CFM may contain a high number of similar configurations $(M, \prec_F^M)$, $(M, \prec_F'^M)$ having the same multisets, differing in the parent-child relations $\prec_F^M$. Our one-wise criterion is only concerned with the number of feature instances to be sampled, such that we can drastically reduce sample sizes by dropping the relation $\prec_F^M$ from the configurations. By $\overline{C}$, we denote the valid configuration space of a CFM using multiset-based configuration semantics.

Reconsidering our example with two teams, we have an overall number between 0 and 16 players, resulting in 17 multiset configurations instead of 36 for two teams. Summing this up for every possible number of teams, we have 17+25+33+384+504+640 = 1603 distinct configurations to be sampled which is considerably smaller than before, but still very large.

RC3: Boundary-Interior Coverage. To further reduce sample sizes, we employ concepts from boundary-interior testing [4]. Boundary interior testing is based on the assumption that boundaries of intervals are particularly error-prone and should be prioritized during test input selection. We adopt this principle to CFM with the normal form representation CFM'. For every feature f, CFM' makes explicit the intervals and corresponding boundaries of all valid multiplicity values.

For instance, in our running example, the global minimum number of instances of feature *Player* is 0, while the global maximum number is 64 (8 teams having 8 players each). Both extremes should be covered as being located at the left-outer and right-outer *boundaries* of the instance cardinality interval of feature *Player*. Concerning the coverage of the *interior* boundaries of cardinality intervals, we utilize the compound interval representation. For instance, to cover the interior boundaries of the compound interval $\langle 2, 4\rangle\langle 6, 8\rangle$ of feature *Team*, we should at least sample configurations with 4 instances and with 6 instances.

To sum up, we define our one-wise CFM sampling criterion.

Definition 4.3 (One-wise CFM Sample). Let $\overline{C}$ be the valid multiset configuration space of a normalized CFM over F. A subset $\mathcal{S} \subseteq \overline{C}$ is a one-wise sample for CFM if for each $f \in F$ the following holds:

(1) if $\exists M' \in \overline{C} : M'(f) = b_l$ and $\forall M'' \in \overline{C} : M''(f) \neq b_l + 1$ then $M' \in \mathcal{S}$ with $M'(f) = b_l$.
(2) if $\exists M' \in \overline{C} : M'(f) = b_r$ and $\forall M'' \in \overline{C} : M''(f) \neq (b_r) - 1$ then $M' \in \mathcal{S}$ with $M'(f) = b_r$.

The criterion requires each global interval gap $b_l][b_r$ of every feature f to be covered at the left boundary b_l (1) and the right boundary b_r. Note that this definition also includes coverage of the left-most boundary (e.g., $b_r = 0$) and the right-most boundary (e.g., $b_l = M$) of the global cardinality interval of f.

	Broa.	Uni.	T.	WiFi	BT	P.	L.	Sca.	Spec.
c_0	1	0	2	0	2	0	2	2	0
c_1	1	0	4	0	4	4	4	4	0
c_2	0	1	6	6	0	6	6	0	6
c_3	0	1	8	8	0	64	8	0	8
c_4	1	0	8	0	8	0	8	8	0

Table 1: Complete and Minimal One-wise Sample

A complete and minimal one-wise sample for our running example containing five configurations is shown in Tab. 1. Features with an instance cardinality of (1,1) are omitted.

The first four configurations cover the boundaries and interiors of *Team* (multiplicities 2, 4, 6 and 8). The additional configuration covers the maximum multiplicity of *BT* and *WiFi* which require eight teams but cannot be covered at the same time.

4.3 Notes on K-Wise CFM Sampling

Similar to the Boolean case, the proposed criterion for one-wise CFM samples gives rise to the definition of k-wise CFM sampling criteria. For instance, for the case $k = 2$, every pair of multiplicity valuations $\#f, \#f'$ according to Def. 4.3 for any two features $f, f' \in F$ is sampled. However, similar to pairwise combinatorial feature interaction coverage for Boolean feature models, we have to check each pair for validity. Considering the feature pair *Player* and *Team* in our running example, the maximum number of 64 instances of type *Player* only builds a valid pair with 8 instances of *Team*, whereas all other valuations selected for one-wise (i.e., 2, 4 and 6) lead to invalid combinations.

There might be critical valid combinations of instance multiplicity values of k different features that are not apparent in the one-wise case. For example, sampling 4 teams with a maximum number of players per team (32 in total) would cover boundary cases depending on a specific number of parent feature instances. Analogously, interval gaps may only appear for specific k-wise combinations of feature multiplicities. While such cases are not contained in our running example, it makes sense to enforce coverage of such globally hidden interval gaps by enhanced k-wise sample criteria for CFMs. However, we assume that this leads to quite involved cases, which we plan to investigate in more depth in future work.

5 IMPLEMENTATION AND FEASIBILITY STUDY

Due to the novelty of the overall approach, our goal is to show feasibility of one-wise CFM sampling. Our second evaluation goal is to assess the run time and sample size.

Our prototypical implementation is based on Clafer[1] and CardyGAn[2] [15, 23]. The CardyGAn boundanalyzer detects cardinality gaps but does not automatically produce the normal form. Furthermore, as neither CardyGAn nor Clafer natively support compound intervals, we are limited to one-wise sampling of CFM with convex valid configuration spaces. Note that this limitation means that we can not tackle RC2 with this implementation yet, but plan on doing so in future work. The input models are encoded as Clafer models. The boundanalyzer analyzes the global minimum/maximum interval bounds per feature. Which are to be covered according to Def. 4.3. To retrieve a valid configuration for each bound, we constrain the feature in the Clafer model to one specific bound and run the Clafer instance generator. The resulting set of generated configurations is a complete one-wise sample. Note that this prototype does not yet include any sample minimization steps.

Our implementation samples CFMs and obtains configurations that cover all one-wise feature interactions of global cardinality bounds. As depicted in Table 2, the overall number of valid instance-based configurations rapidly increases even for simple models. The number of multiset-based configurations, however, increases at a much lower rate. As every feature in CFM with a convex configuration space has exactly one upper and one lower bound that has to be covered, the number of cardinality values to be covered sums up to only $2 * |F|$. Additionally, configurations generated to cover the global upper bound of a child feature also cover the global upper bound of the respective parent feature. This reduces the sample size further. For each configuration space depicted in Fig. Table 2 a sample has at most the size 12, demonstrating the effective reduction of sample size of the proposed sample criterion. The main issue is a lack of scalability to larger models with large global interval bounds. Table 3 depicts sampling of CFMs with increasing bounds and hierarchy depths. The runtime of the Clafer instance generator, and thus our sampling implementation, heavily depends on the maximum global bound. As a result, sampling small models (w.r.t. the number of features) with large bounds takes longer than larger models with smaller bounds. In general, for increasing bounds and hierarchy depth, the run time tends to grow exponentially. We find Clafer not feasible for sampling larger CFMs. We therefore plan to either improve Clafer or to build our own sampling tool for CFMs based on recent SMT, CSP, and ILP solvers [22].

6 RELATED WORK

Sampling criteria and sample-generation techniques have been extensively studied for Boolean feature models in the recent past [21]. In contrast, we are not aware of any existing work on sampling CFM-like models. Basic criteria like uniform random sampling may be useful for CFM sampling, while most other criteria cannot be directly transferred to CFMs. These state-of-the-art criteria include one-wise, pair-wise, and k-wise feature interaction coverage, partially discussed in this paper. In contrast, more involved sample criteria like most-enable-disable, all-one-disable, or one-enable that focus on sampling corner cases might be promising for CFM sampling, where we consider our approach as a proper basis.

Kaltenecker et al. [6] propose a distance-based sampling approach that might be adapted to CFM. This technique requires a metric, assigning a distance value to every configuration. Guided by a predefined probability distribution, configurations with specific distances are generated evenly covering the configuration space.

Concerning sampling feature models beyond Boolean, Munoz et al. [10] propose bit-blasting for feature models with numerical features. While able to generate uniform random samples and to count the number of valid configurations, this approach only works for numerical features, whereas multi-instantiation is out of scope.

[1] https://www.clafer.org/
[2] https://github.com/Echtzeitsysteme/cardygan

Hierarchy level	#valid configurations instance-based 1..3	instance-based 1..5	multiset-based 1..3	multiset-based 1..5	#value schemata
1	3	5	3	5	2
2	19	251	15	65	4
3	1539	$8.8 * 10^9$	151	2885	6
4	$6 * 10^8$	$4.4 * 10^{47}$	3495	492545	8
5	$3.8 * 10^{25}$	$1.4 * 10^{236}$	198503	$3.4 * 10^8$	10
6	$9 * 10^{75}$	$> 1.8 * 10^{308}$	$2.9 * 10^7$	$9.8 * 10^{11}$	12

Table 2: Number of valid configurations of CFM models with increasing nesting level and bounds

Hierarchy level	run time for bound (s) 1..1	1..3	1..5	1..10
1	0.5	0.2	0.2	0.2
2	0.2	0.4	0.4	0.3
3	0.3	0.4	0.6	4.9
4	0.3	0.7	5.7	15460.6
5	0.3	2.7	1852.8	
6	0.4	49.6		

Table 3: Run times for increasing hierarchy level and bounds

There are several works on optimizing non-boolean configurations of software product lines [1, 11, 24]. Again, all these techniques consider numerical features and are thus not transferable to CFM. Sousa et al. [16] highlight the problem of different interpretations of cardinalities in CFMs. However, their work does not mention sampling or sample generation for the different notions of cardinalities.

Another branch of research does not particularly focus on CFM but on instance-based structural modeling formalisms in general. Sullivan et al. [17] propose *AUnit* for *Alloy*, enabling automated generation of coverage-based test suits for *Alloy* models. For a given model, the approach generates test cases (instances) covering specific parts of the model definition. *AUnit* differs from our approach in multiple aspects. The goal of *AUnit* is to test whether the model is correctly defined, which requires a specification of the expected behavior. In contrast, sampling is concerned with testing products, not the configuration model. An *Alloy* model may define an unbounded number of products which is handled by *AUnit* by requiring a manually specified bound (scope). In contrast, due to the structural properties of CFM, we automatically determine a finite bound for a model.

7 CONCLUSION AND FUTURE WORK

In this paper, we tackled the open problem of sampling CFM. To this end, we had to deal with several challenges obstructing a straightforward application of established sampling approaches for Boolean feature models. These challenges include false/true unbounded and non-convex configurations spaces of CFM as well as a drastically increased combinatorial explosion problem. Our proposed criterion is inspired by k-wise combinatorial coverage criteria as well as by boundary interior testing. Using our implementation, we were able to show realizability of our proposed approach and to gather first insights into run-time factors and scalability.

In future work, we plan to develop fully-fledged tool support for CFM modeling and sampling, including proper handling of interval gaps. Building upon the proposed one-wise sample criterion, we want to investigate more complex criteria like pairwise and k-wise. Finally, we want to overcome the lack of example models by conducting case studies.

ACKNOWLEDGMENTS

We thank our reviewers for their constructive feedback. We also want to thank Alexander Lieb and Hendrick Göttmann for providing source code believed lost to support our evaluation. This work has been funded by the German Research Foundation within the project *Co-InCyTe* (SCHA1635/15-1 and LO 2198/4-1).

REFERENCES

[1] Michał Antkiewicz, Kacper Bąk, Alexandr Murashkin, Rafael Olaechea, Jia Hui (Jimmy) Liang, and Krzysztof Czarnecki. 2013. Clafer Tools for Product Line Engineering. In *Proceedings of the 17th International Software Product Line Conference Co-Located Workshops* (Tokyo, Japan) *(SPLC '13 Workshops)*. Association for Computing Machinery, New York, NY, USA, 130–135. https://doi.org/10.1145/2499777.2499779

[2] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Formalizing Cardinality-Based Feature Models and Their Specialization. 10 (2005), 7–29.

[3] Krzysztof Czarnecki and Chang Hwan Peter Kim. 2005. Cardinality-Based Feature Modeling and Constraints: A Progress Report. 16–20.

[4] William E. Howden. 1975. Methodology for the Generation of Program Test Data. *IEEE Trans. Computers* 24, 5 (1975), 554–560. https://doi.org/10.1109/T-C.1975.224259

[5] Paulius Juodisius, Atrisha Sarkar, Raghava Rao Mukkamala, Michal Antkiewicz, Krzysztof Czarnecki, and Andrzej Wasowski. 2018. Clafer: Lightweight Modeling of Structure, Behaviour, and Variability. https://arxiv.org/pdf/1807.08576v1. (2018).

[6] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. 2019. Distance-Based Sampling of Software Configuration Spaces. 1084–1094. https://doi.org/10.1109/ICSE.2019.00112

[7] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical Report CMU/SEI-90-TR-21. Software Engineering Institute.

[8] Raphael Michel, Andreas Classen, Arnaud Hubaux, and Quentin Boucher. 2011. A Formal Semantics for Feature Cardinalities in Feature Diagrams. 82–89. https://doi.org/10.1145/1944892.1944902

[9] Regina Hunter Mladineo. 1994. Model Building in Mathematical Programming (H. P. Williams). *SIAM Rev.* 36, 2 (1994), 313–315. https://doi.org/10.1137/1036082

[10] Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes, and Don Batory. 2019. Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features. 289–301. https://doi.org/10.1145/3336294.3336297

[11] Rafael Olaechea, Steven Stewart, Krzysztof Czarnecki, and Derek Rayside. 2012. Modelling and Multi-Objective Optimization of Quality Attributes in Variability-Rich Software. In *Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages* (Innsbruck, Austria) *(NFPinDSML '12)*. Association for Computing Machinery, New York, NY, USA, Article 2, 6 pages. https://doi.org/10.1145/2420942.2420944

[12] Clément Quinton, Daniel Romero, and Laurence Duchien. 2013. Cardinality-based feature models with constraints: a pragmatic approach. In *17th International Software Product Line Conference, SPLC 2013, Tokyo, Japan - August 26 - 30, 2013*, Tomoji Kishi, Stan Jarzabek, and Stefania Gnesi (Eds.). ACM, 162–166. https://doi.org/10.1145/2491627.2491638

[13] Björn Richerzhagen, Dominik Stingl, Ronny Hans, Christian Gross, and Ralf Steinmetz. 2014. Bypassing the cloud: Peer-assisted event dissemination for augmented reality games. In *14th IEEE International Conference on Peer-to-Peer Computing, P2P 2014, London, United Kingdom, September 9-11, 2014, Proceedings*. IEEE, 1–10. https://doi.org/10.1109/P2P.2014.6934296

[14] Matthias Riebisch, Kai Böllert, Detlef Streitferdt, and Ilka Philippow. 2002. Extending Feature Diagrams with UML Multiplicities.

[15] Thomas Schnabel, Markus Weckesser, Roland Kluge, Malte Lochau, and Andy Schürr. 2016. CardyGAn: Tool Support for Cardinality-Based Feature Models. In *Proceedings of the 10th International Workshop on Variability Modelling of Software-Intensive Systems* (Salvador, Brazil) *(VaMoS '16)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2866614.2866619

[16] Gustavo Sousa, Walter Rudametkin, and Laurence Duchien. 2016. Extending Feature Models with Relative Cardinalities. In *Proceedings of the 20th International Systems and Software Product Line Conference* (Beijing, China) *(SPLC '16)*. Association for Computing Machinery, New York, NY, USA, 79–88. https://doi.org/10.1145/2934466.2934475

[17] Allison Sullivan, Kaiyuan Wang, and Sarfraz Khurshid. 2018. AUnit: A Test Automation Tool for Alloy. In *11th IEEE International Conference on Software Testing, Verification and Validation, ICST 2018, Västerås, Sweden, April 9-13, 2018*. IEEE Computer Society, 398–403. https://doi.org/10.1109/ICST.2018.00047

[18] Chico Sundermann, Kevin Feichtinger, Dominik Engelhardt, Rick Rabiser, and Thomas Thüm. 2021. Yet Another Textual Variability Language? A Community Effort Towards a Unified Language. 136–147. https://doi.org/10.1145/3461001.3471145

[19] Chico Sundermann, Kevin Feichtinger, José A. Galindo, David Benavides, Rick Rabiser, Sebastian Krieter, and Thomas Thüm. 2022. Tutorial on the Universal Variability Language. 260:1. https://doi.org/10.1145/3546932.3547024

[20] Chico Sundermann, Stefan Vill, Thomas Thüm, Kevin Feichtinger, Prankur Agarwal, Rick Rabiser, José A. Galindo, and David Benavides. 2023. UVLParser: Extending UVL with Language Levels and Conversion Strategies. In *Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B* (Tokyo, Japan) *(SPLC '23)*. Association for Computing Machinery, New York, NY, USA, 39–42. https://doi.org/10.1145/3579028.3609013

[21] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 2018. A Classification of Product Sampling for Software Product Lines. 1–13. https://doi.org/10.1145/3233027.3233035

[22] Markus Weckesser, Malte Lochau, Michael Ries, and Andy Schürr. 2018. Mathematical Programming for Anomaly Analysis of Clafer Models. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (Copenhagen, Denmark) *(MODELS '18)*. Association for Computing Machinery, New York, NY, USA, 34–44. https://doi.org/10.1145/3239372.3239398

[23] Markus Weckesser, Malte Lochau, Thomas Schnabel, Björn Richerzhagen, and Andy Schürr. 2016. Mind the Gap! Automated Anomaly Detection for Potentially Unbounded Cardinality-Based Feature Models. https://doi.org/10.1007/978-3-662-49665-7_10

[24] Yi Xiang, Yuren Zhou, Zibin Zheng, and Miqing Li. 2018. Configuring Software Product Lines by Combining Many-Objective Optimization and SAT Solvers. *ACM Trans. Softw. Eng. Methodol.* 26, 4, Article 14 (feb 2018), 46 pages. https://doi.org/10.1145/3176644