Mind the Gap! Automated Anomaly Detection for Potentially Unbounded Cardinality-Based Feature Models

Markus Weckesser[1](✉), Malte Lochau[1], Thomas Schnabel[1], Björn Richerzhagen[2], and Andy Schürr[1]

[1] Real-Time Systems Lab, TU Darmstadt, Darmstadt, Germany
markus.weckesser@es.tu-darmstadt.de
[2] Multimedia Communications Lab, TU Darmstadt, Darmstadt, Germany

Abstract. Feature models are frequently used for specifying variability of user-configurable software systems, e.g., software product lines. Numerous approaches have been developed for automating feature model validation concerning constraint consistency and absence of anomalies. As a crucial extension to feature models, cardinality annotations and respective constraints allow for multiple, and even potentially unbounded occurrences of feature instances within configurations. This is of particular relevance for user-adjustable application resources as prevalent, e.g., in cloud computing. However, a precise semantic characterization and tool support for automated and scalable validation of cardinality-based feature models is still an open issue. In this paper, we present a comprehensive formalization of cardinality-based feature models with potentially unbounded feature multiplicities. We apply a combination of ILP and SMT solvers to automate consistency checking and anomaly detection, including novel anomalies, e.g., interval gaps. We present evaluation results gained from our tool implementation showing applicability and scalability to larger-scale models.

Keywords: Software product lines · Cloud-based systems · Cardinality-based feature models · Integer Linear Programming (ILP)

1 Introduction

Feature models become more and more established for specifying *variability* of highly-configurable software, e.g., software product lines [11]. Feature models are used during domain engineering to tailor configuration spaces of product lines in terms of available configuration parameters (*features*) and respective *constraints*, restricting their combinations within valid *configurations*. Each feature constitutes a user-visible (Boolean) configuration option from the problem domain, being mapped onto variable implementation artifacts within the solution space. This way, customer-tailored products are derivable from a common code base during application engineering. The FODA feature diagram notation

P. Stevens and A. Wąsowski (Eds.): FASE 2016, LNCS 9633, pp. 158–175, 2016.
DOI: 10.1007/978-3-662-49665-7_10

is a frequently used graphical representation for feature models [6,22]. FODA feature diagrams organize features as nodes in a tree-like layout to denote a parent-child hierarchy. This feature tree is enriched with constructs to describe logical dependencies among features. Semantically, a feature model specifies a set of valid product configurations, i.e., those feature combinations satisfying all constraints. Recent approaches to formalizing feature model semantics either use algebraic representations [19,34], or transformations into equivalent constraint problems, e.g., propositional formulas (SAT) [5,25], and CSP [7]. The latter approach allows for applying off-the-shelf constraint-solvers for automatically validating desirable semantic properties of feature models such as constraint consistency and absence of anomalies, e.g., dead features [6].

However, FODA feature diagram notation is, in many cases, not expressive enough for capturing all user-configurable properties of real-world applications. In particular, two major extensions to feature models have been proposed, usually summarized under the term *extended feature models* (EFM), namely (1) non-Boolean feature attributes and respective constraints to denote *extra-functional properties of features*, and (2) UML-like feature multiplicities [32] in terms of cardinality annotations and respective constraints to allow selections of *multiple feature instances* (also referred to as *copies*), including (recursive) *clones* of their corresponding sub-trees [14]. Semantically, both concepts impose extensions to the notion of product configurations by means of (1) feature types beyond Boolean, and (2) multi-sets of selected feature instances. Both extensions complicate feature model semantics, thus automated consistency checking and anomaly detection becomes even more important for their applicability in practice. Concerning (1), various promising approaches have been proposed for analyzing non-Boolean configuration constraints [7,9,20,23]. In contrast, concerning (2), only preliminary attempts exist so far [12,14,26,29,30], although cardinality-based variability modeling is emerging in nowadays applications and, therefore, recently found its way into novel modeling approaches like CVL [16] and Clafer [3]. As a prominent example, for cloud-based systems, not only the *type*, but also the *amount* of available resources is explicitly configurable by the user [28], especially including (virtually) *unrestricted* resources [35]. The resulting *compound* cardinality intervals lead to novel kinds of anomalies by means of *dead cardinality*, *cardinality interval gaps* and *false unbounded cardinality*.

In this paper, we present a comprehensive formalization and automated validation technique for cardinality-based feature models (CFM). We support cardinality annotations including compound cardinality intervals and unbounded cardinality for singleton features, feature groups, as well as cross-tree constraints. Our approach is motivated by a real-world cloud-based application [31]. We further introduce a *normal form* for cardinality constraints and enhance established notions of feature model consistency and anomaly to explicitly take feature cardinality constraints into account. Our tool implementation, presented in full detail in an accompanying tool paper [33], combines ILP solvers for interval-bound analysis and SMT solvers for interval-gap analysis to automate validation of cardinality-based feature models. We provide evaluation results from experiments investigating applicability and scalability of our validation approach for input models of varying sizes and complexity.

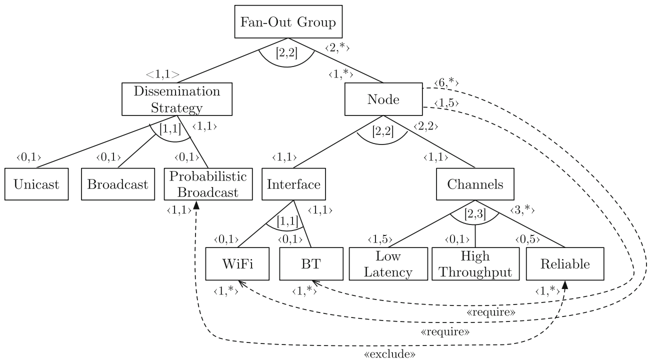


Fig. 1. CFM for fan-out group configuration of the event dissemination system

2 Cardinality-Based Feature Models

2.1 Background

Our running example is part of a cloud-based mobile augmented reality (AR) multi-player game scenario [31]. During a game, players (*nodes*) move and carry devices according to a predefined goal. Players communicate via cellular connections with a cloud-based service provider which delivers relevant game data and disseminating events. Players interact with the physical environment and other players located nearby. For this purpose, an Area of Interest (AoI) virtually surrounds each player's physical location, where overlapping AoI may form *Fan-Out Groups* to establish decentralized ad-hoc connections. This *bypassing* of the service provider may reduce latency of the cellular network.

All components of an AR game are highly configurable, including dynamic reconfigurations for run-time adaptation. Configuration decisions not only comprise *presence* or *absence* of functionality, but also the *available amount* of particular resources. Thus, CFM provide a suitable formalism to capture all relevant configuration choices and respective constraints of AR games. Figure 1 shows the CFM for configuring the *Dissemination Strategy*, the communication *Interface* and *Channel* properties of a (potentially unbounded) number of *Nodes* forming a *Fan-Out Group*. Similar to FODA notation [22], configuration parameters (*features*) reside in a tree-like diagram denoting a feature *decomposition* hierarchy. As a crucial extension, CFM differentiate between selectable/deselectable feature *types* as usual and, additionally, for each selected feature type, the *multiplicity* of occurrences of *feature instances* together with copies of their corresponding subtrees within configurations [14]. Restrictions on selections of both feature types and instances are specified by *cardinality intervals* (l, u), where l denotes the *lower* bound and u denotes the *upper* bound for the number of feature types or instances [32]. In particular, the CFM language considered in this paper provides the following constructs.

– *Feature instance cardinality*, annotated as $\langle l, u\rangle$ on the left-most position on top of each feature rectangle, restricts the minimum and maximum number of feature instances selectable from the sub-tree clone of respective parent feature instances. In our example, $\langle 1, 1\rangle$ denotes that exactly one *Dissemination Strategy* is selectable, whereas $\langle 1, *\rangle$ denotes that arbitrary many, but at least one *Node* must be part of a *Fan-Out Group*.
– *Feature group type cardinality*, annotated as $[l, u]$, restricts the minimum and maximum number of types of feature instances selectable from the set of all immediate sub-features of a selected feature instance. In our example, $[1, 1]$ denotes that either instances of *WiFi*, or of *BT* must be selected for the *Interface*, whereas $[2, 3]$ denotes that at least two types of *Channels* from the given three options must be instantiated in a *Fan-Out Group*.
– *Feature group instance cardinality*, annotated as $\langle l, u\rangle$ at the right-hand side of each group arc, restricts the minimum and maximum number of feature instances of any type selectable from the set of all immediate sub-feature types. In our example, $\langle 3, *\rangle$ denotes that arbitrary many, but at least three *Channel* instances are required for each *Node*.
– *Cross-tree edges* by means of require- and exclude-edges annotated with $\langle l, u\rangle$ constraints at both the source and target feature rectangles [30], define constraints on the number of instances of arbitrary pairs of features. In our example, if at least one instance of *Reliable* is selected in a sub-tree clone, then no instance of *Probabilistic Broadcast* is allowed in the *Fan-Out Group* and vice versa. In addition, if between 1 and 5 *Nodes* are selected in a *Fan-Out Group*, then *BT* is used for all *Nodes* and *WiFi*, otherwise.

Combining different cardinality annotations in one CFM may lead to complicated dependencies among feature types and their possible number of instances. In order to provide a precise characterization of CFM configuration semantics, we provide a CFM formalization in the following. We first define the abstract syntax of CFM. Therefore, we introduce an *interval language* to express cardinality intervals (l, u) as pairs of *lower* and *upper* cardinality bounds, both given by natural numbers, or, in case of upper bounds, also by the special symbol $*$ denoting *unbounded* cardinality. By convention, $k < *$ holds for any $k \in \mathbb{N}_0$. *Compound cardinality intervals* are defined as the *union* of multiple (non-overlapping) intervals $(l_1, u_1), (l_2, u_2), \ldots, (l_n, u_n)$.

Definition 1 (Cardinality Interval). *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 further require compound intervals $L \in \mathcal{L}$ to be defined as concise as possible, e.g., $\{(1, 4)\}$ instead of $\{(1, 2), (3, 4)\}$. Intervals $L \in \mathcal{L}$ are used for all kinds of cardinality annotations in a CFM as described above. A CFM consists of a finite set F of features together with a hierarchy relation $\prec_F$ defining the tree hierarchy on F such that $f \prec_F f'$ denotes f to be the *parent feature* of f'.

In addition, a feature instance cardinality interval $\lambda_I^F(f) \in \mathcal{L}$ is assigned to every feature $f \in F$ by a function λ_I^F, as well as a group type cardinality interval $\lambda_T^G(f) \in \mathcal{L}$ by a function λ_T^G, and a group instance cardinality interval $\lambda_I^G(f) \in \mathcal{L}$ by a function λ_I^G. Both $\lambda_T^G(f)$ and $\lambda_I^G(f)$ define cardinality intervals on the set of *direct sub-features* of feature f with respect to $\prec_F$, hence we do not allow multiple direct sub-groups below one feature node. Furthermore, we require for every non-leaf feature $f \in F$ $\lambda_I^F(f)$, as well as $\lambda_T^G(f)$ and λ_I^G to be properly defined, even if f only contains a singleton sub-feature f', e.g., by assuming default group cardinality constraints $\lambda_T^G(f) = (0, 1)$ and $\lambda_I^G(f) = (0, *)$. Cross-tree edges consist of four components, i.e., the source feature and the target feature and corresponding cardinality annotations restricting the number of feature instances. Due to the binary nature of cross-tree edges, cardinality intervals referring to feature types are meaningless and, therefore, not supported.

Definition 2 (CFM). *A* cardinality-based feature model (CFM) *defined over a non-empty, finite set* F *is a tuple* $(\prec_F, \lambda_I^F, \lambda_T^G, \lambda_I^G, \Phi_R, \Phi_X)$*, where*

- $\prec_F \subseteq F \times F$ *is a* feature decomposition *relation,*
- $\lambda_I^F : F \rightarrow \mathcal{L}$ *is a* feature instance cardinality *function,*
- $\lambda_T^G : F \rightarrow \mathcal{L}$ *is a* feature group type cardinality *function,*
- $\lambda_I^G : F \rightarrow \mathcal{L}$ *is a* feature group instance cardinality *function,*
- $\Phi_R \subseteq F \times \mathcal{L} \times \mathcal{L} \times F$ *is a* feature instance require-edge cardinality *relation,*
- $\Phi_X \subseteq F \times \mathcal{L} \times \mathcal{L} \times F$ *is a* feature instance exclude-edge cardinality *relation.*

For a CFM to be *syntactically well-formed*, it must satisfy further properties.

- $\prec_F$ forms a *finite rooted tree* on F, i.e., $\prec_F^+$ is a *strict partial order* on F with root feature $f_r \in F$ as unique minimal element, and for each $f \in F$, $f \neq f_r$, there is exactly one direct predecessor node $f' \in F$ with $f' \prec_F f$.
- Root feature f_r is a mandatory single-instance feature, i.e., $\lambda_I^F(f_r) = (1, 1)$.
- Leaf nodes have empty group cardinality intervals, i.e., for each $f \in F$ with $\nexists f' \in F : f \prec_F f'$, $\lambda_G^I(f) = \lambda_G^T(f) = (0, 0)$ holds.

Further well-formedness criteria may be imposed, e.g., forbidding $*$ as upper bound for feature group type cardinality. However, these and far more complicated cases are comprehensively treated by the normal form in Definition 6.

Obviously, CFM syntax constitutes a conservative extension to FODA feature diagrams [14,30]. However, concerning CFM semantics, the structure of valid CFM configurations essentially differs from FODA configurations. In particular, a CFM configuration not only contains information about the *presence*, or *absence* of features, but also the *number of instances* selected for each feature, as well as their memberships to the cloned sub-tree related to its parent feature instance. In this regard, one crucial semantic consideration for CFM concerns the interpretation of cardinality intervals restricting the number of feature instances. As already pointed out by Michel et al. in [26], one may either apply a *local*, or a *global* interpretation. For illustration purposes, we use the artificial CFM in Fig. 2 with sample configurations C_1, C_2, C_3, and C_4. Each feature instance constitutes the root of a (recursively) cloned sub-tree which can be configured

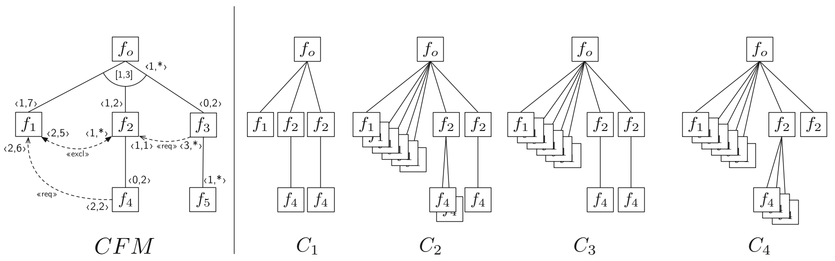


Fig. 2. CFM with sample configurations

individually for that instance. Considering, e.g., the require-edge from f_4 to f_1, a global interpretation would require this constraint to hold for the entire set of selected feature instances of f_4 and f_1, whereas in case of a local interpretation, the constraint must hold for every individual sub-tree clone. As a result, C_1 is invalid in case of a global interpretation, as the overall number of instances of f_4 is 2, but there is only one instance of f_1 in C_1. Hence, C_2 is valid as the overall number of instances of f_4 is 3 and, therefore, the precondition of the require-edge does not hold. C_3 is also valid as a sufficient number of instances of f_1 is selected. In contrast, in case of a local interpretation, C_1, C_2, and C_3 are all valid as either the precondition of the require-edge is not satisfied by any sub-tree clone of f_2 (C_1 and C_3), or the number of instances of f_1 is sufficient (C_2). Finally, although C_2 and C_4 have the same *number* of instances of each feature type, C_2 is valid for both interpretations, whereas C_4 is invalid in both cases as the feature instance cardinality of f_4 is violated. This example shows that the membership of feature instances to their corresponding parent feature instance sub-tree clones is a crucial part of CFM configuration semantics.

Here, we apply the global interpretation, constituting – in our opinion – the more intuitive and graspable CFM semantics. *CFM configuration semantics* characterizes those valid feature sub-tree copies with corresponding parent-child feature instance dependencies satisfying all cardinality constraints. Our CFM semantics is based on multi-sets M over set F to denote the number of feature instances selected in a configuration. A multi-set $M : F \to \mathbb{N}_0$ over set F defines a mapping from each element $f \in F$ onto a natural number $k = M(f)$, defining the *multiplicity* of f, where $k = 0$ denotes *absence* of f in M. We write $f_i^k \in M$, $1 \leq k \leq M(f_i)$ for short to refer to the kth instance of feature $f_i \in F$ within multi-set M with $M(f_i) > 0$. Furthermore, given a compound interval $L = \{(l_1, u_1), (l_2, u_2), \ldots, (l_n, u_n)\} \in \mathcal{L}$ and $k \in \mathbb{N}_0$, we write $k \sqsubseteq L$ if $(l_i, u_i) \in L$ such that $l_i \leq k \leq u_i$ holds. We further denote a relation $\prec_F^M \subseteq M \times M$ on multi-set M, relating child feature instances to parent feature instances.

Definition 3 (CFM Configuration). *A* configuration *of a cardinality-based feature model $(\prec_F, \lambda_I^F, \lambda_I^G, \lambda_T^G, \Phi_R, \Phi_X)$ defined over a set F is a pair $(M, \prec_F^M)$. A configuration $(M, \prec_F^M)$ is* valid *iff*

- *$M(f_r) = 1$,*
- *if $f_i^k \prec_F^M f_j^l$ then $f_i \prec_F f_j$ and $(\prec_F^M)^+$ forms a rooted tree on M,*

- *if* $f_i^k \in M$*, then for each* $f_j \in F$ *with* $f_i \prec_F f_j$ *it holds that* $|\{f_j^l \in M | f_i^k \prec_F^M f_j^l\}| \sqsubseteq \lambda_F^I(f_j)$,
- *if* $f_i^k \in M$*, then it holds that* $|\{f_j^l \in M | f_i^k \prec_F^M f_j^l\}| \sqsubseteq \lambda_G^I(f_i)$,
- *if* $f_i^k \in M$*, then it holds that* $|\{f_j \in F | \exists f_j^l \in M : f_i^k \prec_F^M f_j^l\}| \sqsubseteq \lambda_G^F(f_i)$,
- *if* $(f_i, L_i, L_j, f_j) \in \Phi_R$ *and* $M(f_i) \sqsubseteq L_i$ *then* $M(f_j) \sqsubseteq L_j$*, and*
- *if* $(f_i, L_i, L_j, f_j) \in \Phi_X$ *and* $M(f_i) \sqsubseteq L_i$ *then* $M(f_j) \not\sqsubseteq L_j$ *and vice versa.*

By $[\![CFM]\!]$, we refer to the set of all valid configurations of *CFM*.

2.2 Analysis of Cardinality-Based Feature Models

We are now able to characterize fundamental validity properties of CFM. In particular, we define *consistency* of CFM in terms of the absence of inconsistent cardinality constraints. By including $*$ as cardinality bound, CFM allow to select an a-priori unbounded number of feature instances and, therefore, a potentially infinite number of configurations.

Definition 4 (Consistent and Bounded CFM). *A CFM is* consistent *iff it holds that* $[\![CFM]\!] \neq \emptyset$. *A CFM is* bounded *iff* $*$ *does not occur in a cardinality annotation. A CFM is* false unbounded *iff* $*$ *occurs in at least one cardinality annotation and* $|[\![CFM]\!]| < \infty$ *holds, and CFM is* unbounded*, else.*

False unboundedness is one example for an undesirable CFM property going beyond syntactic well-formedness criteria. To generalize, we recall the notion of *anomaly* to summarize undesirable semantic CFM properties. For FODA feature models, several types of anomalies and accompanying validation techniques have been proposed, e.g., dead features and false optional features [6]. First proposals exist to lift the anomaly notion also to CFM, e.g., *dead cardinality* anomaly [30].

Definition 5 (Dead Feature Instance Cardinality). $k \sqsubseteq \lambda_F^I(f_i)$ *is a* dead feature instance cardinality *of* $f_i \in F$*, if no* $(M, \prec_F^M) \in [\![CFM]\!]$ *with* $f_j^k \in M$ *and* $f_j \prec_F f_i$ *exists such that* $|\{f_i^l \in M | f_j^k \prec_f^M f_i^l\}| = k$ *holds.*

For other kinds of cardinality intervals of a CFM, the notion of dead cardinality can be defined, accordingly. Hence, for a feature f to be *dead* in a CFM, every cardinality $k \sqsubseteq \lambda_F^I(f_i)$ must be dead, thus the *actual* feature cardinality instance interval of f is $(0, 0)$, and a CFM is inconsistent if all features are dead.

The example in Fig. 2 exhibits several subtle cases of CFM anomalies. For example, the group instance cardinality $\langle 1, * \rangle$ of f_0 is *false unbounded* as the maximum number of possible child-feature instances is 11. The same holds for the interval $\langle 1, * \rangle$ on the right-hand side of the exclude-edge between f_1 and f_2 whose upper bound is actually limited to 2. In contrast, feature f_5 is *truly unbounded* thus making the entire CFM unbounded. Besides (false) unbounded intervals, this CFM contains further anomalies concerning bounded cardinality intervals. The lower bound 1 of the group instance cardinality interval $\langle 1, * \rangle$ of f_0 is a *dead cardinality*, as at least one instance of both f_1 and f_2 must be selected. Thus, lower bound 1 of group type cardinality $[1, 3]$ of f_0 is also *dead*. In addition, the lower bound of the target feature node cardinality interval $\langle 2, 6\rangle$ of the require-edge from f_4 to f_1 is actually 6 instead of 2. Besides CFM anomalies affecting upper and/or lower bounds of cardinality intervals, a dead cardinality might be also located *within* intervals, thus imposing *interval gaps*. For example, the group instance cardinality of f_0 contains a gap at $(6, 6)$ as no valid combination of feature instances of f_1, f_2, and f_3 with an overall number of 6 is possible. As an even more subtle case, feature instance cardinality interval $\langle 1, 7\rangle$ of f_1 contains the interval gap $(2, 5)$.

Due to the predominant role of cardinality constraints in CFM, any kind of potential semantic inconsistency can be explained through dead cardinality. To this end, we define a *normal form* for any given CFM by narrowing its declared cardinality intervals down to the *actual* ones, while preserving its feature-tree layout and configuration semantics. In case of gaps, closed interval declarations can be replaced by *compound* intervals, e.g., replacing group instance interval $(1, *)$ of f_0 in Fig. 2 by $\{(2, 5), (7, 11)\}$. In this way, a normal form $\overline{CFM}$ characterizes all dead cardinality anomalies compared to the original model CFM by means of those (sub-)ranges of feature cardinality intervals being removed from CFM to obtain $\overline{CFM}$. Hence, a CFM with $*$ occurring in some cardinality interval, but having no $*$ in its normal form is *false unbounded*. Furthermore, if a given CFM is *inconsistent*, all feature cardinality intervals of $\overline{CFM}$ are narrowed down to $(0, 0)$ (if we permit $\lambda^I_F(f_r) = (0, 0)$). Finally, to handle *redundant* cross-tree edges, we have to allow removals of edges from CFM to obtain a semantically equivalent normal form $\overline{CFM}$. For example, the precondition of the require-edge leading from f_3 to f_2 in Fig. 2 is not satisfiable thus making this edge redundant in $\overline{CFM}$. To formalize CFM normal form, we define an *inclusion hierarchy relation* $\precsim \subseteq \mathcal{L} \times \mathcal{L}$ as

$$L \precsim L' :\Leftrightarrow \forall k \in \mathbb{N}_0 : k \sqsubseteq L \Rightarrow k \sqsubseteq L'$$

thus requiring L to be a sub-range of L'.

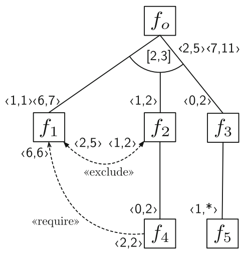
Definition 6 (CFM Normal Form). *$\overline{CFM}$ is a* normal form *of CFM if*

- $[\![\overline{CFM}]\!] = [\![CFM]\!]$,
- $\overline{F} = F$, $\overline{\prec}_F = \prec_F$, $\overline{\Phi}_R \subseteq \Phi_R$, $\overline{\Phi}_X \subseteq \Phi_X$, *and*
- *for each* $f_i, f_j \in \overline{F}$, $\overline{\lambda}^F_I(f_i)$, $\overline{\lambda}^G_T(f_i)$, $\overline{\lambda}^G_I(f_i)$, *as well as* L_i *and* L_j *in each* $(f_i, L_i, L_j, f_j) \in \overline{\Phi}_R$ *and* $(f_i, L_i, L_i, f_j) \in \overline{\Phi}_X$ *are* minimal *with respect to* $\precsim$.

Applied to the CFM in Fig. 2, the resulting normal form is shown in Fig. 3(a). The following property is a direct consequence of Definitions 5 and 6.

Theorem 1. *For any CFM according to Definition 2, a normal form $\overline{CFM}$ exists and $\overline{CFM}$ contains no dead cardinality.*

In contrast, a normal form is, in general, not *unique* as removals of (mutually depending) redundant cross-tree edges may yield ambiguous results. A procedure for computing normal forms would allow for automatically consolidating



(a) Normalized $\overline{CFM}$

$$\boxed{1}\quad \max\ \mathfrak{f}_1 + \mathfrak{f}_2 + \mathfrak{f}_3$$

subject to:

$$\boxed{2}\quad \begin{aligned} &\mathfrak{t}_0 \le \mathfrak{f}_0 \le M\mathfrak{t}_0 \\ &\mathfrak{t}_1 \le \mathfrak{f}_1 \le M\mathfrak{t}_1 \\ &\dots \end{aligned}$$

$$\boxed{3}\quad \begin{aligned} &\mathfrak{f}_0 = 1 \\ &\mathfrak{f}_0 \le \mathfrak{f}_1 \le 7\mathfrak{f}_0 \\ &\mathfrak{f}_0 \le \mathfrak{f}_2 \le 2\mathfrak{f}_0 \\ &0 \le \mathfrak{f}_3 \le 2\mathfrak{f}_0 \\ &0 \le \mathfrak{f}_4 \le 2\mathfrak{f}_0 \\ &\mathfrak{f}_3 \le \mathfrak{f}_5 \end{aligned}$$

$$\boxed{4}\quad \mathfrak{f}_0 \le \mathfrak{f}_1 + \mathfrak{f}_2 + \mathfrak{f}_3$$

$$\boxed{5}\quad \mathfrak{t}_0 \le \mathfrak{t}_1 + \mathfrak{t}_2 + \mathfrak{t}_3 \le 3\mathfrak{f}_0$$

$$\boxed{6.1}\quad \begin{aligned} &\mathfrak{f}_4 \le \mathfrak{r}_0 + 2\mathfrak{r}_1 + 4\mathfrak{r}_2 \\ &\mathfrak{f}_4 \ge \mathfrak{t}_4 - \mathfrak{r}_0 + \mathfrak{r}_1 + 2\mathfrak{r}_2 \\ &\mathfrak{t}_4 = \mathfrak{r}_0 + \mathfrak{r}_1 + \mathfrak{r}_2 \end{aligned}$$

$$\boxed{6.2}\quad 2\mathfrak{r}_1 \le \mathfrak{f}_1 \le 7 - \mathfrak{r}_1$$

$$\boxed{7.1}\quad \begin{aligned} &\mathfrak{f}_3 \ge \mathfrak{t}_3 - \mathfrak{r}_3 + 2\mathfrak{r}_4 \\ &\mathfrak{f}_3 \le 2\mathfrak{r}_3 + 2\mathfrak{r}_4 \\ &\mathfrak{t}_3 = \mathfrak{r}_3 + \mathfrak{r}_4 \end{aligned}$$

$$\boxed{7.2}\quad \mathfrak{r}_4 \le \mathfrak{f}_2 \le 2 - \mathfrak{r}_4$$

$$\boxed{8.1}\quad \begin{aligned} &\mathfrak{f}_1 \le \mathfrak{r}_5 + 5\mathfrak{r}_6 + 7\mathfrak{r}_7 \\ &\mathfrak{f}_1 \ge \mathfrak{t}_1 - \mathfrak{r}_5 + \mathfrak{r}_6 + 5\mathfrak{r}_7 \\ &\mathfrak{t}_1 = \mathfrak{r}_5 + \mathfrak{r}_6 + \mathfrak{r}_7 \end{aligned}$$

$$\boxed{8.2}\quad \mathfrak{t}_2 = \mathfrak{r}_8$$

$$\boxed{8.3}\quad \mathfrak{r}_6 + \mathfrak{r}_8 \le 1$$

(b) ILP Encoding of CFM

Fig. 3. Sample CFM normal and ILP encoding of CFM semantics

and validating CFM, e.g., during domain analysis. However, constraint-solvers for SAT and CSP, usually used for validating FODA feature models, are not applicable for CFM validation due to the potentially unbounded search space.

3 Automated Anomaly Detection for CFM

We observe two potential causes for anomalies in CFM during normal form computation due to faulty declarations of cardinality intervals: (1) unsatisfiable lower/upper bounds (including false unbounded), and (2) unsatisfiable subranges (gaps). For (1), we encode CFM semantics in an ILP representation and use a respective ILP-solver for bound analysis, whereas for (2), we apply an SMT-solver to find interval gaps. To keep the presentation concise, we focus our considerations on input models *CFM* with non-compound cardinality intervals $L \in \mathcal{I}$.

Analysis of Interval Bounds. An ILP consists of a set of linear inequalities on a set of k integer-valued *decision variables*. The resulting convex hull forms the *feasible region* within a k-dimensional search space. An *objective function* states that either lower (minimum), or upper (maximum) boundary integer values for decision variables should be found by an ILP-solver. Encoding CFM semantics as ILP thus enables automated detection of dead cardinality potentially located at the *boundary* of cardinality intervals.

The ILP encoding of the CFM from Fig. 2 is given in Fig. 3(b). As decision variables, we introduce for each feature $f_i \in F$ a *feature multiplicity variable* $\mathfrak{f}_i \in \mathbb{N}_0$, denoting the *number* $M(f_i)$ of instances of type f_i being selected, and a *feature selection variable* $\mathfrak{t}_i \in \{0, 1\}$, denoting whether at least one instance of f_i is selected in a CFM configuration. Consistency between variables $\mathfrak{f}_i$ and corresponding variables $\mathfrak{t}_i$ is enforced by constraints $M \cdot \mathfrak{t}_i \ge \mathfrak{f}_i$ and $\mathfrak{t}_i \le \mathfrak{f}_i$ for all $f_i \in F$, (cf. $\boxed{2}$ in Fig. 3(b)). Here, we incorporate a coefficient M, frequently referred to as *big M* in the literature [37], by means of a sufficiently large number for coupling binary variables $\mathfrak{t}_i$ to integer variables $\mathfrak{f}_i$. Coefficient M is conservatively approximated by multiplying the maximum upper bounds of cardinality intervals occurring in each branch of the feature tree and choosing the overall maximum value. The upper bound is derived from the syntactic context of the cardinality interval under consideration. Occurrences of $*$ are replaced in the same way. Due to monotonicity of aggregated cardinality interval bound values imposed by the CFM tree structure (cf. Definition 3), the restriction of the ILP search space to M, therefore, yields correct analysis results also for unbounded CFM.

To encode CFM semantics of feature instance cardinality intervals and subtree cloning, we introduce inequalities $l \cdot \mathfrak{f}_i \leq \mathfrak{f}_j \leq u \cdot \mathfrak{f}_i$ for all parent-child pairs $f_i \prec_F f_j$ and $(l, u) = \lambda_F^I(f_j)$ for child feature $f_j \in F$ (cf. $\boxed{3}$ in Fig. 3(b)). The inequality restricting the upper bound u is only introduced if u is bounded which does not hold, e.g., for feature f_5 in our example. For root feature f_r (denoted f_0), we have a special constraint $\mathfrak{f}_0 = 1$. For group instance cardinality intervals, we introduce inequalities

$$l \cdot \mathfrak{f}_i \leq \sum_{f_i \in F : f_i \prec_F f_j} \mathfrak{f}_j \leq u \cdot \mathfrak{f}_i$$

for all parent-child pairs $f_i \prec_F f_j$ and $(l, u) = \lambda_G^I(f_i)$ for parent feature $f_i \in F$. Again, in the unbounded case, we only restrict the lower bound. Semantics of group type cardinality intervals can be encoded, accordingly. The resulting group constraints for our example are depicted at $\boxed{4}$ and $\boxed{5}$ in Fig. 3(b), where the constraint at $\boxed{4}$ for f_0 only contains one inequality due to unboundedness.

Finally, cross-tree edges constitute the most complicated part potentially obstructing linearity of the ILP constraint set. To handle those cases, we use additional decision variables by means of fresh *interval selection variables* $\mathfrak{r}_k \in \{0, 1\}$ denoting a particular interval being selected or not. For each cross-tree edge $(f_i, L_i, L_j, f_j)_k \in \Phi_Y$, $Y \in \{R, X\}$, we define inequalities for source and target feature node intervals. For the source feature node f_i, we introduce three *interval selection variables* $\mathfrak{r}_{k-1}$, $\mathfrak{r}_k$, and $\mathfrak{r}_{k+1}$ to encode selection conditions for $L_i = (l_i, u_i)$. We encode the lower bounds of matching conditions of interval selection variables by

$$\mathfrak{f}_i \geq \mathfrak{r}_{k-1} + (l_i + 1) \cdot \mathfrak{r}_k + (u_i + 2) \cdot \mathfrak{r}_{k+1} - 1$$

and, for the upper bounds, by

$$\mathfrak{f}_i \leq (l_i - 1) \cdot \mathfrak{r}_{k-1} + u_i \cdot \mathfrak{r}_k + M \cdot \mathfrak{r}_{k+1},$$

respectively. To this end, $\mathfrak{r}_{k-1}$ indicates that the value of $\mathfrak{f}_i$ is below l_i, $\mathfrak{r}_k$ indicates that the value of $\mathfrak{f}_i$ is within interval L_i, and $\mathfrak{r}_{k+1}$ indicates that value of $\mathfrak{f}_i$ is above u_i. If the source feature node cardinality interval is either unbounded, or its lower bound equals 1, the inequality is adapted, accordingly. In addition, the constraint $\mathfrak{t}_i = \mathfrak{r}_{k-1} + \mathfrak{r}_k + \mathfrak{r}_{k+1}$ ensures the interval not be selected and deselected at the same time if f_i is present. Applied to our example, the resulting encoding of source feature node cardinality intervals of the four cross-tree edges is shown in Fig. 3(b) at $\boxed{5.1}$, $\boxed{6.1}$, $\boxed{7.1}$, and $\boxed{8.1}$. Due to symmetry of exclude-edge semantics, target feature node cardinality intervals can be encoded in the same way as shown at $\boxed{8.2}$. To ensure mutual exclusion, an inequality such as at $\boxed{8.3}$ is added for each exclude-edge. For encoding target feature node cardinality intervals of require-edges $(f_i, L_i, L_j, f_j) \in \Phi_R$ with $(l_j, u_j) = L_j$, we introduce the constraint

$$l_j - M \cdot (1 - \mathfrak{r}_k) \leq \mathfrak{f}_j \leq u_j + M \cdot (1 - \mathfrak{r}_k)$$

to ensure that if the source node condition holds ($\mathfrak{r}_k = 1$), then f_j is within L_j (cf. $\boxed{5.2}$, $\boxed{6.2}$ and $\boxed{7.2}$ in Fig. 3(b)).

Based on this ILP encoding, CFM bound analysis can be performed for intervals $(l, u) \in \mathcal{I}$ by employing a corresponding ILP objective function, i.e., either minimization for lower bound analysis, or maximization for upper bound analysis. In Fig. 3(b), we analyze the upper bound of the group instance cardinality interval of f_0 by using the objective function *max* $f_1 + f_2 + f_3$ (cf. $\boxed{1}$), which returns 11. Considering unbounded cardinality intervals, we have two cases. In case of false unbounded intervals, e.g., the upper bound of the group instance cardinality of f_0, the solver run returns a *bounded* result with an objective value less than M. In case of a truly unbounded cardinality interval, e.g., the upper bound of feature instance cardinality of f_5, the solver either reports *unbounded but feasible*, or returns a value equal to M. To sum up, ILP-based interval bound analysis is *sound* in the sense that bounds of the search space reported feasible do not contain any dead cardinality. Similarly, the technique is *complete* in the sense that any dead cardinality at the bounds of the search space is detectable.

Detection of Interval Gaps. The ILP-based approach for interval-bound analysis is not directly applicable for interval-gap analysis as gaps are, by definition, not located at minima/maxima locations of the search space. For example, for detecting the group instance cardinality interval gap at $(6, 6)$ of f_0 in Fig. 2, we have to check whether $(6, 6)$ is a feasible value for the corresponding feature multiplicity variables. Hence, detecting interval gaps does not constitute an *optimization* problem, but rather a *constraint satisfaction* problem incorporating integer inequalities. To this end, an SMT-solver is applicable, being capable of interpreting first-order logics equipped with linear Integer arithmetics theory according to our ILP encoding of CFM semantics (cf. Fig. 3). For gap analysis, every sub-range of all cardinality intervals of a CFM has to investigated, where in case of unbounded intervals, analysis has to be performed up to M.

Normal Form Computation. We can now combine interval-bound analysis and interval-gap analysis to compute CFM normal forms. By `ILP(CFM,interval)` we denote ILP-solver calls to investigate a particular cardinality `interval` of `CFM`. The call returns the *actual* lower and upper bound of that interval to potentially replace the declared intervals within the normal form. For lower bounds of cardinality intervals defined by λ_I^F, λ_I^G and λ_T^G, the result is either greater than, or equal to the declared lower bound. For upper bounds, the result is either lower than, or equal to the declared upper bound. In case of unbounded intervals, the call either returns a concrete value in case of false unboundedness, or reports unboundedness. In case of infeasible intervals, the call returns $(0, 0)$. For interval-gap analysis, we denote `SMT(CFM,interval,range)` for respective SMT-solver calls, where `range` is a sub-range of `interval` to be investigated. For reducing the search space for gap detection, parameter `range` can be obtained from ILP-based bound analysis. The SMT call reports invalid sub-ranges within `range` leading to compound intervals within the normal form. Finally, for cardinality intervals L_i, L_j of cross-tree edges $(f_i, L_i, L_j, f_j) \in \Phi_Y$, $Y \in \{R, X\}$, bound and gap analysis is, in general, performed as described above. In contrast, infeasibility of source and/or target feature node intervals imposes incremental removals of the corresponding edges from Φ_Y during normal form computation.

4 Experimental Evaluation

We implemented CFM bound analysis and gap detection in a tool providing textual syntax for specifying input CFM models [33]. Here, we present evaluation results gained from several experiments performed with our tool. We address the following research questions.

(RQ1) Is CFM normal form computation *applicable* to real-world input models?
(RQ2) How does the size and complexity of CFM affect *scalability* of CFM analysis?
(RQ3) How does the ILP-based feasibility check perform on FODA feature models compared to a SAT-based satisfiability check?

To address **(RQ1)**, we applied our tool to the real-world CFM in Fig. 1. To address **(RQ2)** and **(RQ3)**, we used synthetically generated CFM models by extending the *BeTTy* tool [36] with cardinality interval generation capabilities including adjustable *maximum feature instance cardinality* and *unbounded interval probability.* We generated CFM by randomly varying all CFM generation criteria using uniformly distributed random variables. Experiments were performed on a Unix machine with Intel Core i5 (2,3 GHz, 8 GB RAM). For bound analysis, we employed as ILP-solvers *CPLEX* [21], *Gurobi* [18], and *GLPK* [17]. For gap detection, we used SMT-solver *Z3* [27] and for **(RQ3)**, we utilized *Sat4j* [24].

For **(RQ1)**, we computed the normal form for the AR game CFM which includes bound analysis for 27 intervals, thus requiring 54 ILP-solver calls. The *CPLEX* ILP-solver took about 10 ms per call. Gap analysis included 27 intervals which took about 15.71 s per call. The resulting normal form exposed a false unbounded group instance interval anomaly for the *Channels* group, thus the unbounded interval symbol `*` is replaced by 11.

Concerning **(RQ2)**, we performed regression analysis to estimate influences of model characteristics on CFM analysis performance metrics. To identify significant coefficients, we applied multiple linear regression analysis on input data sets by randomly varying all generation criteria. We applied *t-tests* to check

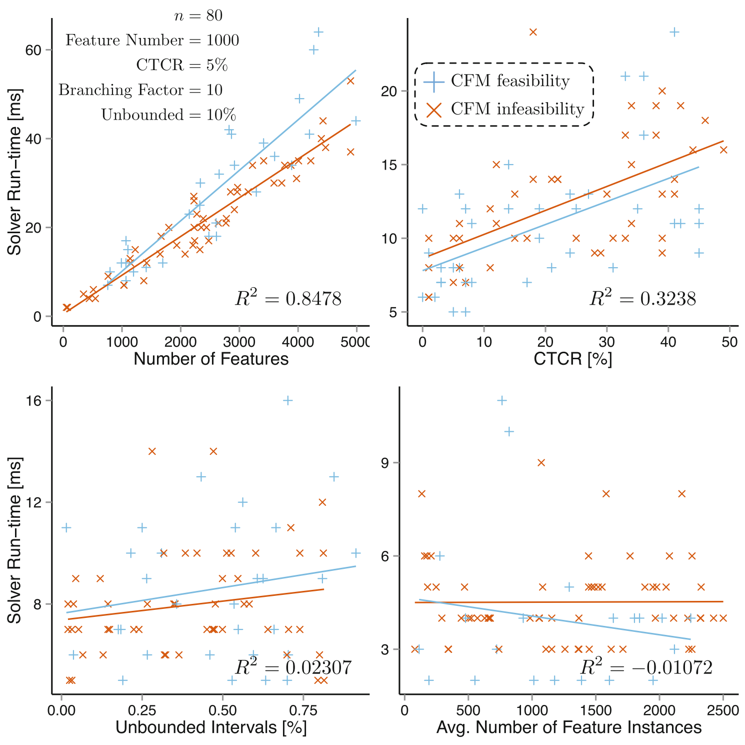


Fig. 4. Evaluation results for **(RQ2)**

significance of regression coefficients. With significance level $p < 0.05$, we identified (a) *number of features*, and (b) *cross-tree constraint ratio (CTCR)*, (c) *ratio of unbounded cardinality intervals*, as well as (d) *CFM feasibility* as coefficients with potentially high influences on run-time of ILP-based bound analysis. In contrast, the influence of *average number of feature instances* is not significant. Figure 4 contains the results of one bound analysis run for individual variation of coefficients (a)–(d). The plots show that run-time of ILP-based bound analysis is dominated by (a) and (b), as the size of the feature tree and the number of cross-tree edges directly affects the number of decision variables and constraints. The results show that ILP-based analysis of one particular bound for CFM with 5,000 features takes about 50 ms and thus about 21 min. for complete bound analysis. This can be considered industrial strength. In contrast, for SMT-based gap analysis, we were only able to obtain run-time analysis results for small-sized (and mostly bounded) CFM up to at most 200 features. As expected, run-time of SMT-based gap analysis tends to show exponential growth with increasing average size of cardinality intervals. For **(RQ3)** we conducted multiple linear regression to estimate influences of FODA feature model characteristics, i.e., with CFM restricted to cardinality intervals between 0 and 1, for comparing run-time of satisfiability checks using SAT and ILP-solvers. We identified coefficients

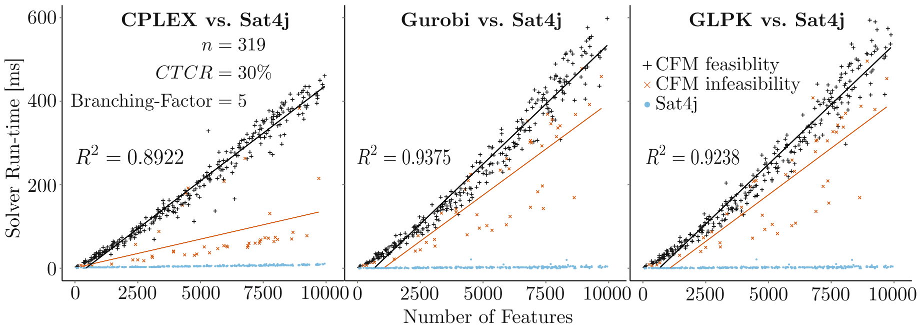


Fig. 5. Evaluation results for **(RQ3)**

number of features, *CTCR* and *CFM feasibility* as highly significant ($p < 0.01$). For *CPLEX*, the *maximum branching factor* has no significant influence. As shown in Fig. 4, the SAT-solver exhibits lower run-time metrics with increasing model size compared to ILP. Nevertheless, ILP-solvers perform remarkably well, with differences in run-time metrics by means of a constant factor only up to models with 5,000 features (Fig. 5).

Threats to Validity. Threats to validity may arise from our experimental input data selection. Concerning **(RQ1)**, the cloud-based AR game is part of a major research project and has already been used for experimental evaluation [31]. Similarly, our design choices for CFM syntax and semantics are derived from requirements of cloud-domain experts. Concerning synthetic data for **(RQ2)** and **(RQ3)**, we employed the well-established *BeTTy* tool for generating FODA-like feature trees, additionally augmented with cardinality intervals. The cardinality interval test data is dimensioned according to characteristics of our case study in order to obtain realistic models. To the best of our knowledge, there does neither exist a fully-fledged CFM generator, nor related approaches for comprehensive CFM analysis as in our approach. Hence, neither a qualitative, nor a quantitative comparison to existing other approaches has been possible so far.

5 Related Work

Formalization of Cardinality-Based Feature Models. Riebisch et al. first propose to extend FODA notation with UML-like multiplicities by means of feature group cardinality [32]. Czarnecki et al. extend feature models with group and feature cardinality, but forbid combinations of both [13]. Thereupon, Czarnecki et al. define CFM semantics based on sub-tree clones and propose their translation into a context-free grammar [14]. They also permit unbounded cardinality but do no investigate their semantic impact. Quinton et al. introduce source and target cardinality for require-edges [30]. However, their approach does neither consider exclude-edges, nor combinations of feature instance and group cardinality. Quinton et al. also mention unbounded cardinality, but neither address it in

CFM semantics, nor as part of CFM analysis. Michel et al. investigate semantic ambiguities due to combinations of feature and group cardinality and distinguish local clone-based from global feature-based interpretation of group type cardinality intervals, being similar to our notion of group instance and group type cardinality intervals [26]. However, they only consider global feature-based interpretation being similar to our notion of group type cardinality intervals. Cordy et al. allow combinations of feature and group cardinality, but for the latter only consider group type cardinality intervals [12]. Again, neither Michel et al., nor Cordy et al. handle unboundedness semantically and during CFM analysis.

Automated Analysis of Cardinality-Based Feature Models. Quinton et al. define inconsistent CFM similar to our notion of dead cardinality anomaly and perform inconsistency detection using CSP [28–30]. Cordy et al. in [12] and Zhang et al. in [38] present BDD-based CFM consistency analysis. However, neither of these approaches is able to handle unbounded configuration spaces and/or interval gaps, nor provide a normal form for CFM.

Analyzing Models with Unbounded Cardinality. Other modeling languages also employ the concept of cardinality to restrict instance multiplicities of model entities. CVL [16] provide *iterators* to mimic cardinality in feature diagrams including unbounded intervals, and the specification language Clafer combines concepts from UML and feature modeling including group and feature instance cardinality [2]. However, no systematic analysis of unbounded cardinality is provided yet. In addition, several approaches have been proposed for analyzing multiplicities in UML class diagrams using Alloy [1], CSP [10], and ILP [15] but none of them explicitly handles unboundedness. Balaban et al. present a graph-based algorithm for tightening multiplicities in UML class diagrams [4]. However, the approach essentially differs from CFM normal form computation as no (recursively) cloned sub-tree hierarchy, cross-tree edges and multiple cardinality constraints per entities occur in class diagrams. Amongst others, Boufares et al. consider inconsistency in cardinality constraints of data-base schema definitions including unbounded cardinality, but do not take interval gaps into account [8].

6 Conclusion

We presented a comprehensive formalization of CFM configuration semantics including unbounded cardinality intervals. We further presented evaluation results gained from experiments conducted with our tool implementation for computing normal forms of CFM. The results show the general applicability and scalability of ILP-based bound analysis. For scalable gap analysis, we aim at replacing the SMT-solver also by an ILP-solver in our future work. We also plan to conduct further experiments including real-world case studies and alternative CFM semantics [26]. For integrating CFM into a fully-fledged engineering process with accompanying tool support, we plan to develop a methodology for mapping feature instances to solution space artifacts as, e.g., propagated by CVL [16].

Acknowledgment. This work was partially supported by the DFG (German Research Foundation) as part of projects B01 and C02 within CRC 1053 – MAKI and under SPP 1593: Design For Future – Managed Software Evolution.

References

1. Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: On challenges of model transformation from UML to Alloy. Softw. Syst. Model. **9**(1), 69–86 (2010)
2. Bąk, K., Czarnecki, K., Wąsowski, A.: Feature and meta-models in Clafer: mixed, specialized, and coupled. In: Malloy, B., Staab, S., Brand, M. (eds.) SLE 2010. LNCS, vol. 6563, pp. 102–122. Springer, Heidelberg (2011)
3. Bak, K., Diskin, Z., Antkiewicz, M., Czarnecki, K., Wasowski, A.: Clafer: unifying class and feature modeling. Softw. Syst. Model. 1–35 (2014)
4. Balaban, M., Maraee, A.: Simplification and correctness of UML class diagrams – focusing on multiplicity and aggregation/composition constraints. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 454–470. Springer, Heidelberg (2013)
5. Batory, D.: Feature models, grammars, and propositional formulas. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 7–20. Springer, Heidelberg (2005)
6. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. Inf. Syst. **35**(6), 615–636 (2010)
7. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated reasoning on feature models. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 491–503. Springer, Heidelberg (2005)
8. Boufares, F., Bennaceur, H.: Consistency problems in ER-schemas for database systems. Inf. Technol. **163**(4), 263–274 (2004)
9. Bürdek, J., Lity, S., Lochau, M., Berens, M., Goltz, U., Schürr, A.: Staged configuration of dynamic software product lines with complex binding time constraints. In: VaMoS 2014, pp. 16: 1–16: 8 (2014)
10. Cadoli, M., Calvanese, D., De Giacomo, G., Mancini, T.: Finite model reasoning on UML class diagrams via constraint programming. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 36–47. Springer, Heidelberg (2007)
11. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co., Inc, Boston (2001)
12. Cordy, M., Schobbens, P.Y., Heymans, P., Legay, A.: Beyond boolean product-line model checking: dealing with feature attributes and multi-features. In: ICSE 2013, pp. 472–481 (2013)
13. Czarnecki, K., Helsen, S.: Staged configuration using feature models. In: Nord, R.L. (ed.) SPLC 2004. LNCS, vol. 3154, pp. 266–283. Springer, Heidelberg (2004)
14. Czarnecki, K., Helsen, S., Eisenecker, U.W.: Formalizing cardinality-based feature models and their specialization. Softw. Process Improv. Pract. **10**(1), 7–29 (2005)
15. Falkner, A., Feinerer, I., Salzer, G., Schenner, G.: Computing product configurations via UML and integer linear programming. Int. J. Mass Customisation **3**(4), 351–367 (2010)
16. Fleurey, F., Haugen, Ø., Møller-Pedersen, B., Svendsen, A., Zhang, X.: Standardizing variability – challenges and solutions. In: Ober, I., Ober, I. (eds.) SDL 2011. LNCS, vol. 7083, pp. 233–246. Springer, Heidelberg (2011)

17. GNU Linear Programming Kit, Version 4.55. http://www.gnu.org/software/glpk/glpk.html
18. Gurobi Optimization, I.: Gurobi Optimizer Reference Manual (2015). http://www.gurobi.com
19. Heymans, P., Schobbens, P.Y., Trigaux, J.C., Bontemps, Y., Matulevicius, R., Classen, A.: Evaluating formal properties of feature diagram languages. IET Softw. **2**(3), 281–302 (2008)
20. Hubaux, A., Heymans, P., Schobbens, P.-Y., Deridder, D.: Towards multi-view feature-based configuration. In: Wieringa, R., Persson, A. (eds.) REFSQ 2010. LNCS, vol. 6182, pp. 106–112. Springer, Heidelberg (2010)
21. IBM ILOG CPLEX V12.6 User's Manual for CPLEX. IBM Corp. (2015). http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
22. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, S.A.: Feature oriented domain analysis (FODA). Technical report, CMU (1990)
23. Karataş, A.S., Oğuztüzün, H., Doğru, A.: Mapping extended feature models to constraint logic programming over finite domains. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 286–299. Springer, Heidelberg (2010)
24. Le Berre, D., Parrain, A.: The Sat4j Library, Release 2.2. J. Satisfiability Boolean Model. Comput. **7**, 59–64 (2010)
25. Mendonça, M., Wasowski, A., Czarnecki, K.: SAT-based analysis of feature models is easy. In: 13th SPLC, pp. 231–240 (2009)
26. Michel, R., Classen, A., Hubaux, A., Boucher, Q.: A formal semantics for feature cardinalities in feature diagrams. In: VaMoS 2011, pp. 82–89 (2011)
27. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
28. Quinton, C., Romero, D., Duchien, L.: Automated selection and configuration of cloud environments using software product lines principles. In: IEEE Cloud 2014, pp. 144–151 (2014)
29. Quinton, C., Pleuss, A., Berre, D.L., Duchien, L., Botterweck, G.: Consistency checking for the evolution of cardinality-based feature models. In: SPLC 2014, pp. 122–131 (2014)
30. Quinton, C., Romero, D., Duchien, L.: Cardinality-based feature models with constraints: a pragmatic approach. In: SPLC 2013, pp. 162–166 (2013)
31. Richerzhagen, B., Stingl, D., Hans, R., Groß, C., Steinmetz, R.: Bypassing the cloud: peer-assisted event dissemination for augmented reality games. In: P2P 2014, pp. 1–10 (2014)
32. Riebisch, M., Böllert, K., Streitferdt, D., Philippow, I.: Extending feature diagrams with UML multiplicities. In: 6th World Conference on Integrated Design & Process Technology (IDPT) (2002)
33. Schnabel, T., Weckesser, M., Kluge, R., Lochau, M., Schürr, A.: CardyGAn: tool support for cardinality-based feature models. In: VaMoS 2016 (2016) (to appear)
34. Schobbens, P.Y., Heymans, P., Trigaux, J.C.: Feature diagrams: a survey and a formal semantics. In: Proceedings of RE 2006, pp. 139–148 (2006)
35. Schroeter, J., Mucha, P., Muth, M., Jugel, K., Lochau, M.: Dynamic configuration management of cloud-based applications. In: SPLC 2012, pp. 171–178 (2012)
36. Segura, S., Galindo, J., Benavides, D., Parejo, J., Ruiz-Cortés, A.: BeTTy: benchmarking and testing on the automated analysis of feature models. In: VaMoS 2012, pp. 63–71 (2012)

37. Williams, H.P.: Model Building in Mathematical Programming. John Wiley & Sons, Hoboken (2013)
38. Zhang, W., Yan, H., Zhao, H., Jin, Z.: A BDD-based approach to verifying clone-enabled feature models' constraints and customization. In: Mei, H. (ed.) ICSR 2008. LNCS, vol. 5030, pp. 186–199. Springer, Heidelberg (2008)