Collecting Feature Models from the Literature: A Comprehensive Dataset for Benchmarking

Chico Sundermann
University of Ulm
Germany

Vincenzo Francesco Brancaccio
University of Ulm
Germany

Elias Kuiter
University of Magdeburg
Germany

Sebastian Krieter
Paderborn University
Germany

Tobias Heß
University of Ulm
Germany

Thomas Thüm
Paderborn University
Germany

Abstract

Feature models are widely used for specifying the valid configurations of product lines. Many automated analyses on feature models have been considered, but they often depend on computationally complex algorithms (e.g., solving satisfiability problems). To identify and develop efficient reasoning engines, it is necessary to compare their performance on practically relevant feature models. However, empirical evaluations on feature-model analysis often suffer from the limitations of available feature-model datasets in terms of transferability. A major problem is the accessibility of relevant feature models as they are scattered over numerous publications. In this work, we perform a literature survey on empirical evaluations that target the performance of feature-model analyses to examine common evaluation practices and collect feature models for future evaluations. Furthermore, we examine the suitability of the derived collection for benchmarking performance. To improve accessibility, we provide a repository including all 2,518 identified feature models from 13 application domains, such as system software.

CCS Concepts

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

Keywords

feature model, product line, survey, evaluation, benchmark

ACM Reference Format:
Chico Sundermann, Vincenzo Francesco Brancaccio, Elias Kuiter, Sebastian Krieter, Tobias Heß, and Thomas Thüm. 2024. Collecting Feature Models from the Literature: A Comprehensive Dataset for Benchmarking. In *28th ACM International Systems and Software Product Line Conference (SPLC '24), September 2–6, 2024, Dommeldange, Luxembourg*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3646548.3672590

1 Introduction

Feature models are commonly used to specify the valid configurations of a product line [6, 8, 51]. Typically, a feature model consists of a set of features and dependencies between those features [51]. A product of the product line is identified by a configuration of features that adheres to all specified dependencies. Manually analyzing feature models is generally infeasible [8, 102]. Even simpler problems, such as checking whether a given configuration satisfies all imposed constraints, require all dependencies of a feature model to be considered. Thus, many automated analyses have been proposed to support various activities in product-line engineering, such as modeling [8], configuring [55, 84], and testing [54, 79].

Many feature-model analyses depend on solving computationally complex problems [106]. In particular, feature models are typically translated to propositional formulas, so their analyses can be reduced to Boolean satisfiability (SAT) [8, 63] or model-counting (#SAT) [36, 56, 107] problems, which are computationally complex [34, 106, 113]. As feature models are often analyzed in interactive settings [1, 9, 70], short response times of reasoning engines (e.g., SAT solvers) are in demand. The efficiency (w.r.t. resource consumption) of tools is particularly relevant when dealing with very complex configurable systems, such as the Linux kernel [66, 109].

Building efficient reasoning engines requires empirical evaluations that are transferable to other instances of the targeted scope. Transferability depends on representative datasets [28, 45], as the performance of reasoning engines highly depends, amongst others, on the analysis and model instance [57, 63, 105]. Without comparisons on feature models that reasonably reflect tool performance in practice, it is difficult for practitioners to select the most promising tool and for tool developers to optimize their algorithms.

Using non-representative datasets for benchmarking performance may lead to empirical evaluations that lack external [117] and ecological [3, 50] validity and cannot be transferred into practice [4, 42] due to several aspects. First, it has been observed that runtimes of reasoning engines, such as SAT solvers, often differ vastly between artificial and real-world formulas [4, 52, 63]. Second, when an empirical evaluation includes only few feature models, it is unclear whether any of its conclusions are transferable to other feature models. This transferability is particularly questionable if the considered feature models are all similar regarding certain characteristics, such as structure (e.g., number of features or number of constraints) and application domain (e.g., automotive or operating system). Third, comparing empirical evaluations is difficult if they use varying datasets, even if those datasets contain multiple real-world instances. Currently, different authors typically use varying datasets for empirical evaluations [10, 14, 38, 54, 56, 81, 89, 97, 105] with only small or even no overlap to models from other work.

SPLC '24, September 2–6, 2024, Dommeldange, Luxembourg

ACM ISBN 979-8-4007-0593-9/24/09
https://doi.org/10.1145/3646548.3672590

Identifying relevant feature models for an empirical evaluation is a high-effort task. Publicly available feature models are scattered over various publications [2, 11, 46, 52, 54, 78, 81, 84, 86, 87, 97]. Also, available feature-model repositories (e.g., SPLOT [72]) focus on exchange instead of benchmarking [29, 70, 72] and contain a large share of artificial and small feature models [70, 72], which are less suitable for benchmarking performance [28]. Other available collections are limited to few [52, 76] or even exactly one domain [11, 53, 86, 114].

In this work, we aim to collect feature models used across existing empirical evaluations of feature-model analyses and make them more accessible for researchers and practitioners. To this end, we provide a comprehensive dataset including available feature models from previous publications. We envision that our collection improves the external and ecological validity of future empirical evaluations. Our contributions towards that goal are the following:
Systematic Literature Survey (Section 3). We performed a survey examining overall 4,475 publications to identify work that performs empirical evaluations on feature-model analyses.
Analysis of Common Practices (Section 4). We gather insights on publishing behavior, used feature models, and employed reasoning engines to analyze common practices in feature-model evaluations. Our observations confirm that having an easily accessible, comprehensive collection of feature models yields benefits.
Feature-Model Collection (Section 5.1). We gather feature models from replication packages or similar data repositories provided by identified publications. Overall, our collection contains 2,518 feature models from 13 application domains.
Accessible Repository (Section 5.2). We make each feature model publicly available [21] in (1) the originally published format, (2) the Universal Variability Language (UVL) [103], and (3) as a propositional formula in conjunctive normal form (CNF) expressed in the DIMACS format [90]. For each feature model, we provide various information on its structure, semantics, and origin. For easy usage, our repository comes with several functionalities for extracting partial datasets (e.g., from specific domains, formats, or sizes).
Evaluation Against Requirements (Section 6). We evaluate our dataset regarding its suitability for empirical evaluations considering requirements specified for benchmarks in other domains [45].

2 Background

In the following, we give a brief overview on feature models, representations for feature models, and analyses on feature models. For a more detailed context, we refer to other relevant literature for definition of feature models [6, 51], representations [7, 24] and analyses on feature models [8, 69, 106].

2.1 Feature-Model Representations

A feature model specifies the set of valid configurations for a product line [6, 8]. Typically, a feature model $FM = (F, C)$ is defined as a set of features F and constraints C over those features. The constraints implicitly specify the set of valid configurations for the feature model. Various representations for feature models have been proposed, differing mostly in their syntax and visualization [6, 48, 51, 92, 103]. However, some languages also support different language constructs with varying levels of expressiveness [48, 108]. Often, constraints and features are limited to Boolean logic (i.e., features are selected or deselected) [8, 54, 54, 73, 80, 105]. Some feature-model extensions enrich the domain of possible values by supporting numerical features [76, 108] or provide more complex constraints [102, 108].

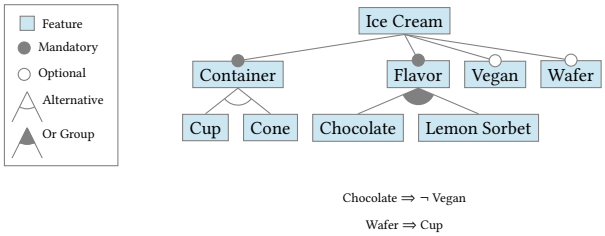

Figure 1: Running Example as a Feature Diagram

Feature Diagrams A feature diagram consists of a hierarchical tree structure over the features and additional cross-tree constraints [92], typically in propositional logic [51, 70, 103]. Figure 1 shows a feature diagram representing a simplified ice cream product line. Here, ice cream requires a `Container` and `Flavor` as denoted by their *mandatory* state. Ice cream may be `Vegan` and may contain a `Wafer` as both are *optional* features. The `Container` can be exactly one of `Cup` and `Cone` as indicated by the *alternative* relationship. Furthermore, the *or* relationship denotes that one or more flavors can be selected. In addition to the tree hierarchy, two cross-tree constraints further limit the set of valid configurations (i.e., ice cream variants). `Chocolate` cannot be selected with `Vegan`. An additional `Wafer` can only be selected with a `Cup`.

Universal Variability Language The Universal Variability Language (UVL) is a textual representation for feature models [103]. Several recent and state-of-the-art tools for feature modeling support UVL [23, 31, 41, 44, 65, 101, 104]. In Listing 1, we show the running example in UVL notation. While the core level of the language supports only propositional constraints [103], recent extensions support more expressive constraints, such as arithmetic expressions over numerical values or non-Boolean features [108].

Conjunctive Normal Form Feature models can be represented as logical formulas. In particular, all feature models with only Boolean variables and constraints can be translated into propositional logic [18]. Furthermore, there have been efforts to also translate more complex feature models (e.g., with numerical features) to propositional logic [76]. Formulas in conjunctive normal form (CNF) are commonly used as input for reasoning engines, such as SAT [12, 73, 75] or #SAT [98, 105, 112] solvers. Hence, feature models are often expressed in CNF to simplify automated reasoning [11, 80]. A CNF is a conjunction of clauses, with each clause being a disjunction of literals [58]. CNFs are typically exchanged in the DIMACS format [90]. In Listing 2, we show an excerpt of a CNF representing our running example in DIMACS format. The line starting with `p cnf` indicates that there are 9 variables (one per feature) and 16 clauses. Each successive line describes a clause. For instance, -2 1 0 describes the clause ($\neg$ `Ice Cream` $\vee$ `Container`).

Listing 1: UVL

```
features
    "Ice Cream"
        mandatory
            Container
                alternative
                    Cup
                    Cone
            Flavor
                or
                    Chocolate
                    "Lemon Sorbet"
        optional
            Vegan
            Wafer
constraints
    Chocolate => !Vegan
    Wafer => Cup
```

Listing 2: DIMACS

```
c 1 Ice Cream
c 2 Container
...
c 8 Wafer
c 9 Vegan
p cnf 9 16
1 0
-2 1 0
-5 1 0
1 -8 0
...
6 -5 7 0
-6 -9 0
3 -8 0
```

2.2 Feature-Model Analyses

Various analyses for feature models have been proposed [8, 25, 53–55, 69, 77, 79, 97, 107]. These analyses can be employed for the development of configurable systems, to extract metrics (e.g., for economical estimations), or to support users during the configuration process. Many analyses depend on numerous complex computations, such as SAT [8, 54, 73] or #SAT [80, 107] problems.

Anomalies An *anomaly* corresponds to a potentially unintended behavior of the feature model [68, 77]. For example, a contradiction in constraints of the feature model may lead to a *void* feature model [8], from which no valid configuration can be derived. Other anomalies considered in the literature are *dead* features (i.e., features that appear in no valid configuration), which may require solving numerous SAT problems [106].

Sampling With *sampling*, a set of valid configurations following a specific goal is derived. With *t-wise* sampling, a set of valid configurations is created that covers all t-wise interactions [47, 54]. For instance, a 2-wise sample covers all interactions between each pair of features. State-of-the-art t-wise samplers rely on repetitive SAT calls to derive a valid sample [47, 54]. Another popular strategy is *uniform random sampling* [37, 80, 88, 97]. To ensure a uniform distribution, state-of-the-art random samplers typically rely on solving #SAT problems [37, 81, 88, 97].

Feature-Model Counting Many analyses depend on *feature-model counting* (i.e., computing the number of valid configurations for a given feature model) [40, 56, 106, 107]. For example, the number of valid configurations that contain a certain feature can be used to prioritize features during development [107]. Such analyses often rely on numerous #SAT invocations.

3 Literature Survey

With our literature survey, we aim to (1) collect a comprehensive dataset of feature models used as test instances in empirical evaluations on feature-model analysis and (2) analyze common practices in evaluations to further inspect the demand for such a collection. In the following, we present our methodology to identify work of interest, collect feature models, and extract relevant information.

3.1 Goals

The main goal of our survey is collecting a feature-model dataset that improves ecological [50] (i.e., transferability to real world) and external validity for benchmarking performance of reasoning engines. Following considerations for selecting benchmark instances from other domains, namely graphs [45], machine learning [61], face recognition [35], and logic [26, 28], we consider three properties relevant for feature models in the context of benchmarking.

P1 *Heterogeneity.* A homogeneous (regarding domain and structure) dataset may limit the transferability [45]. Feature models are used to model variability in various domains, such as automotive [56], system software [11], and finances [27]. As results cannot be necessarily transferred between domains [4, 28, 61], we aim to collect feature models from *many domains*. Also, instances in benchmark datasets should be *heterogeneous considering structural properties* [45], such as number of features or tree hierarchy.

P2 *Suitability for Performance Tests.* There are various aspects of datasets that might be relevant for performance tests depending on the use case. We consider two dimensions here that we found relevant for the evaluation of reasoning engines for own evaluations and for evaluations many we examined during our survey. First, to effectively select and improve algorithms, we need *sufficiently hard* instances (here: feature models) that show the impact of optimizations [28, 45]. Feature models that are easy to analyze with unoptimized algorithms may, thus, be less relevant for a collection that focuses on benchmarking. While the hardness depends on various properties, existing evaluations suggest that feature models with very few features are typically not hard for most feature-model analyses [25, 39, 89, 105]. As feature models with fewer than 100 features appear to be computationally easy for popular automated reasoning engines [89, 102, 105, 106], we only consider feature models with *at least 100 features*. Collecting all feature models available, including very small ones, would considerably increase the required effort and various smaller feature models are already available in the widely used SPLOT repository [72]. We apply this filter of feature models with at least 100 features only for the collection. In the *survey*, we also examine work with smaller feature models. Second, a common goal of performance tests is to *accurately identify performance differences of tools* which requires a *sufficient number* of input instances [35], which are currently laborious to collect.

P3 *Transferability.* The results of evaluations should be transferable into practice [45, 115]. It has been shown that reasoning engines often perform substantially different on artificial instances compared to real-world instances [4, 52, 63]. While empirical evaluations on carefully designed artificial models can still be useful, we only consider *real-world feature models* to ensure ecological validity [50].

In addition to collecting feature models, we aim to use our survey to gather insights on the usage of feature models and reasoning engines in empirical evaluations. We envision that our observations help in understanding the demands and practices in evaluations on feature-model analysis and provide more evidence for problems addressed in this work. Note that while we only consider feature models satisfying **P1**-**P3** for our feature-model collection, we also include empirical evaluations using other feature models in the survey to reduce the bias in our observations on common practices.

Table 1: Search String

	("feature model" OR "feature models" OR "feature modeling" OR "feature modelling")
AND	("product line" OR "product lines" OR "product family")
AND	(analy* OR evaluat* OR solver OR benchmark* OR configura* OR sampl* OR anomal* OR tool*)

Table 2: Inclusion & Exclusion Criteria

IC1	Feature models describe a product line
IC2	Feature models are in a standardized format
IC3	Evaluation with an automated reasoning engine
IC4	Full text of publication is available
IC5	Publication was subject to peer-review
EC1	Publication is not in English
EC2	Paper was published prior to 1990

We provide insights on the considered feature models, the publishing behavior, and evaluated reasoning engines with the following three research questions:

RQ1 How often are feature models reused across evaluations?

RQ2 What kinds of feature models are evaluated?

RQ3 Which reasoning engines are used in empirical evaluations?

3.2 Literature Search

Our literature survey for identifying relevant publications consists of two phases. In the first phase, we perform a keyword search on popular databases to identify publications including relevant empirical evaluations and, thus, feature models. In the second phase, we perform a partial snowballing procedure on the identified papers.

Keyword Search In Table 1, we show the search string we used to identify relevant literature. The search string is inspired by related work also surveying automated analysis on feature models [30, 36]. The search string consists of three main parts aiming to identify work on feature models in the product-line domain that targets a kind of analysis or include an empirical evaluation. Using the search string shown in Table 1, we performed a search on the following literature databases: *ACM Digital Library*, *IEEE Xplore*, *Springer Link*, and *Web of Science*, as we expect a reasonable coverage of relevant publications from these databases.

After collecting the publications matching our keyword search, we apply the inclusion and exclusion criteria shown in Table 2. Here, we aim to only keep publications that cover an empirical evaluation on automated reasoning for feature models in the product line domain. As Kang et al. [51] introduced the notion of feature models in 1990, we exclude earlier publications.

Snowballing After collecting the initial set of publications, we aim to provide additional insights on the included feature models. First, we aim to identify the original publications of feature models we found. Second, we try to collect work that uses feature models originally published in work we identified.

To identify the original publications of found feature models, we perform partial *backward snowballing*. Hereby, we consider feature models included in an empirical evaluation, but originally published in other work. For each publication including such a feature model, we follow the corresponding reference to identify the original work.

To identify work that reuses publicly available feature models, we perform *forward snowballing*. Hereby, we consider each work in our literature set that originally published at least one original feature model with 100 or more features for forward snowballing. We extend our literature set with identified publications that include one of those feature models in their empirical evaluation.

For both directions of snowballing, we collect the publications using the *cited by* function from Google Scholar. For each identified paper, we documented each decision for comprehensibility [22].

Table 3: Data Extraction Form

ID	Data	Purpose
D1	Feature-model publication metric	**RQ1**
D2	Feature-model artificiality metric	**RQ2**
D3	Feature-model size	**RQ2**
D4	Feature-model format	**RQ2**
D5	Feature-model domain	**RQ2**
D6	Solver identification metric	**RQ3**

3.3 Data Extraction

For each identified work satisfying our inclusion and exclusion criteria, we gather insights on the published feature models, how they are published, and with which solvers the feature models were analyzed. In addition to documentary information, such as authors, title, and links, we gathered the information depicted in Table 3. To extract the respective information, the second author searched abstract, introduction, evaluation chapters, and respective artifacts. If a decision about inclusion or the data extraction was unclear, a second author also reviewed the respective work [13].

With the publication metric (D1) shown in Table 4, we examine how the feature models are published. The metric provides insights on the reproducibility and transferability to practice. A score **0** implies that results cannot be reproduced at all. Publications with a score of **1** use generated, artificial models which is out of our scope. We use scores **2** and **3** for forward and backward snowballing (cf. Section 3.2), respectively, to identify additional publications covering real-world feature models.

The artificiality metric (D2) indicates whether a feature model represents a real-world system, is artificial, or of unknown origin. We based the classifications of artificiality on the descriptions in the manuscript. Hereby, we consider the feature model to be real-world if (1) the authors explicitly state that it is a real-world feature model or (2) the descriptions clearly imply that a real-world system is considered. An example for the latter is that the authors state that they used a Linux feature model for their evaluation. If the information does not clearly imply the artificiality, we consider it as *origin unclear*. We later use the metric D2 for identifying feature models satisfying **P3** for our dataset.

With D3-D5, we provide further insights on the feature models by extracting their size (regarding number of features and constraints), original textual format, and domain (e.g., system software or automotive). Note that we select the domain depending on the available information by the following order (1) information by the original authors (2) from other feature models we found with

Table 4: Publication Metric D1

Public. Score	Availability of included feature models
3	Unique feature models publicly available
2	Feature models from other publications
1	Feature models from generator
0	Feature models not published

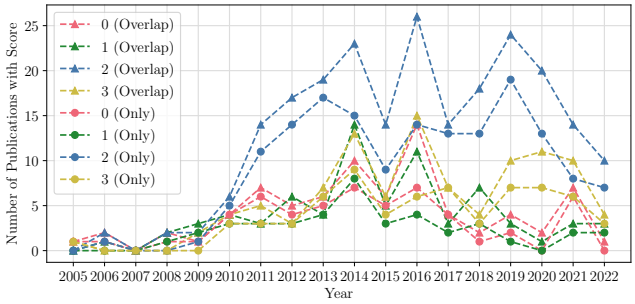

Figure 2: Publication Metric over Time

domain information available, or (3) classification of the authors of this work based on content and description.

Orthogonal to analyzing the feature models, we examine the usage of reasoning engines in empirical evaluations. With D6, we analyze the level of information on the reasoning engine provided by authors. We classify the level of detail in: providing a direct link to the used version, specifying the solver name and version, only giving the name of the solver, or not providing sufficient information to identify the solver at all.

4 Survey Insights

With our initial keyword search, we found 3,382 publications in total that conform to our search string. The databases were accessed in May 2022 and, thus, include publications between January 1990 and May 2022. After applying the inclusion and exclusion criteria and filtering duplicates, we identified a set of 345 papers relevant for our survey. With our strategy of forward and backward snowballing (cf. Section 3.2), we further examined 1,093 publications. Here, we identified 53 additional papers, resulting in an overall set of 398 publications. For comprehensibility, sheets that indicate our decisions for each publication regarding inclusion/exclusion criteria and data extraction are publicly available [22].

4.1 RQ1: Publishing of Feature Models

In Figure 2, we show the results for the publication metric of the 398 publications over time. The *(only)*-lines indicate how many publications satisfy exactly that score. The *(overlap)*-lines show how many publications satisfy that score but possibly also others. Overall, 39.8% of the publications contain only feature models from other publications and 56% at least one. 17.1% of the publications include only original feature models and 25.6% at least one. 55 (13.8%) publications published none of the feature models used in their empirical evaluation, further 19.6% do not publish all.

In Figure 3, we show the number of times different publications or repositories have been used in empirical evaluations. Feature

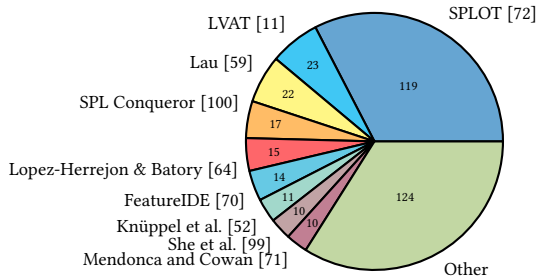

Figure 3: Feature Model Sources

models from the SPLOT [72] repository are by far the most often used with 119 different publications explicitly stating usage of SPLOT models in their empirical evaluation.

RQ1: 19.6% of publications include unpublished feature models in their evaluation. Hence, the feature models cannot be reused in other empirical evaluations and comparisons are more difficult. Results are also not fully reproducible if the feature models are not available. 43.5% of publications do not reuse any previously publicly available feature models.

4.2 RQ2: Feature-Model Characteristics

For 29.4% of the publications, we were not able to identify the origin of all feature models. More than half (53.5%) of the publications include at least one real-world feature model, but also possibly artificial ones. 29.4% of the publications include at least one feature model from generators. Feature models from the following generators are used most often: BeTTy [95] (29 times), the FeatureIDE generator [111] (15 times), and the SPLOT generator [72] (14 times).

We identified a large variety of domains for which feature models are available. However, 102 (25.6%) publications also contain at least one feature model where we were not able to extract the domain. Only 84 (21.1%) of evaluations include feature models from more than three different domains. Feature models from the system software (appearing in 82 publications), e-commerce (80), and automotive (72) domains are included in the largest number of publications. The numbers of occurrences for these domains mainly come from few popular systems, such as Linux for system software, E-Shop [72] for e-commerce, and Automotive02 [52] for automotive, that are included in many publications.

In Figure 4, we show the smallest (x-axis) and largest (y-axis) feature models considering their number of features for each publication. The largest considered feature model includes 1,048,576 features and is artificially generated. The smallest feature model contains only one feature. Overall, we could extract both the minimum and maximum number of features for 324 publications. 100 publications (30.8%) contain only feature models with fewer than 100 features (bottom left square). 233 publications (71.9%) contain at least one feature model with fewer than 100 features (upper squares). 87 publications (26.9%) contain only feature models with at least 100 features (upper right square).

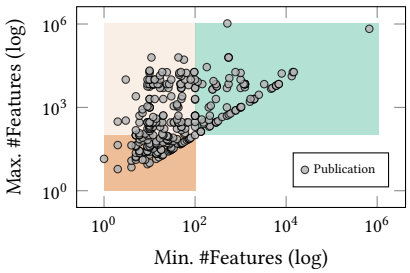

Figure 4: Minimum and Maximum Number of Features

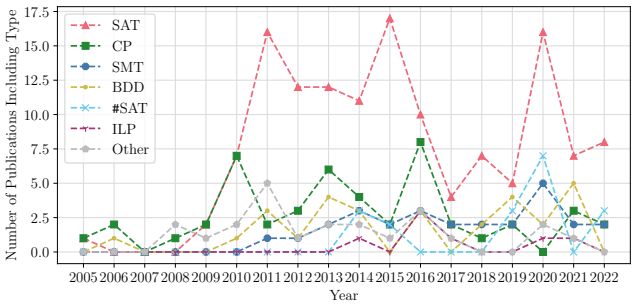

Figure 5: Usage of Solver Classes Over Time

Table 5: Results on Solver Identification

Provided Information	Absolute	Relative
No Information	18	9.7%
Only Name	72	38.7%
Name & Version	22	11.8%
Name & Link	56	30.1%
Name & Version & Link	18	9.7%

RQ2: Real-world feature models seem to be generally more popular than artificial ones for empirical evaluations. However, information on the origin of feature models is often missing, which makes it hard to identify real-world models. While feature models from many domains have been used overall, only 21.1% of the empirical evaluations consider more than three domains. 30.8% of the empirical evaluations are limited to feature models with < 100 features.

4.3 RQ3: Reasoning on Feature Models

In Table 5, we show the type of information documenting the usage of solvers for the 186 publications (46.7%) in our literature set that explicitly report the use of a solver. The remaining publications use ad-hoc solutions, provide no information on a possibly employed reasoning engine, or do not consider automated reasoning at all. The latter were added by the backward snowballing, because they contain models used by included work. Of these 186 publications, 18 (9.7%) do not provide sufficient information to extract even the name of the solver used. Full information (i.e., name, version, and link) is only provided in 18 papers (9.7%).

Figure 5 shows the occurrences of the different solver classes in evaluations over time. SAT solvers are used most in almost every year with overall 149 publications. Solving techniques, such as CP (48), BDDs (28), SMT (27), and #SAT (18), are also often used. The popularity of CP dropped over time, being the second most often used solver 2005–2016. In the last years of the considered time period (2017–2022), CP is only the fifth most popular solver class. 13 of 18 publications considering #SAT solvers were published in the last five years. Overall, the Java-based SAT solver Sat4J [60] is used most (56 publications). The constraint programming (CP) solver Choco [49] and the satisfiability modulo theories (SMT) solver Z3 [19] are used in 30 and 20 publications, respectively. Unidentifiable SAT solvers are also used (17). The most used binary decision diagram (BDD) tool is JavaBDD [116] (13). For model counting (#SAT), sharpSAT [112] is the most often used solver (8).

RQ3: SAT solvers are generally the most popular type of solvers for feature-model analysis with Sat4J [60] being the most used SAT solver. Other popular classes of reasoning engines are CP solvers, BDDs, SMT solvers, and #SAT solvers. 48.4% of publications miss relevant information for reproducibility (e.g., version of solvers).

4.4 Threats to Validity

Validation of Selected Papers The majority of publications in the full dataset (i.e., before applying inclusion and exclusion criteria) were only evaluated by one author. To reduce the effect, each time this author was not fully certain on whether a publication should be included, another author also checked the paper. All decisions and reasoning behind them, in terms of fulfilled or violated inclusion/exclusion criteria, are publicly available [22].

Dependency on Other Author's Information For identifying properties of the feature model (e.g., the artificiality), we rely on the information provided by the authors. We tried to verify some properties, such as domain and artificiality, from provided artifacts. However, as of November 2022, the feature models for many publications are not publicly available (anymore).

Interpretation of Authors In several cases, the classification of feature models, regarding their artificiality and their domain, required interpretation from the authors of this work. Hence, some classifications possibly would have been decided differently by other authors. To reduce the impact of our decisions, we tried to base the decision as much as possible on information provided by the publishers of the feature models. Further, each unclear feature model was examined by at least two authors.

Limited Scope Our literature survey focuses on feature models that have previously been used in evaluations. As a consequence, we might have missed work that published relevant feature models but does not evaluate them for any kind of feature-model analysis. We still expect that the feature models we collected are particularly relevant for better comparability to previous empirical evaluations. Also, as we include feature models from many publications with differing domains and structures, we expect that our dataset is beneficial for practitioners and researchers with regard to the properties outlined in Section 3.1. Nevertheless, we consider extending our collection in the future as valuable.

5 Feature-Model Dataset

The main goal of this work is to derive a feature-model collection that can be used for comparable empirical evaluations that can be transferred into practice. We limit our dataset to feature models that represent a real-world system (cf. **P3** in Section 3.1) and have at least 100 features (cf. **P2**). Overall, we found 2,518 feature models that match our criteria. In the following, we give an overview on the feature models in our collection and the functionalities of the public repository we provide [21].

5.1 Feature Models

To identify the feature models we included in our repository, we further filtered the publications according to the properties **P1**-**P3** defined in Section 3.1. After filtering, we identified 20 papers with relevant, original, and publicly available feature models resulting in overall 2,518 feature models from 41 different systems (e.g., Linux) and 13 application domains (e.g., system software). Table 6 gives an overview on the collected feature models. Since multiple system software feature models were only available in CNF format, we show the number of clauses for better comparability between the formats. Also, for feature models only available in CNF, we consider the number of variables as number of features in Table 6.

While we identified 102 papers with original feature models, we excluded the majority of those, as they only included small or artificial original models or their repositories were not available anymore. We also collected seven feature-model histories, in which each feature model represents an evolution step of the corresponding system. In some cases, the feature model did not change after an evolution step [85, 86], here we discarded all duplicates but the first occurring. Without counting each feature model of histories as different instances, our collection contains 1,617 feature models. The high number of security feature models comes from the AMADEUS-dataset [114] where each feature model describes configurations that might be affected by a cybersecurity vulnerability.

5.2 Repository

Collection of Feature Models We provide each identified feature model in the format of the original publication, in UVL [103], and CNF (DIMACS).[1] We added UVL to facilitate standardized feature-model analyses, which may also rely on the feature hierarchy of the model. The DIMACS format allows users the quick use of off-the-shelf solvers, such as SAT [12, 73] or #SAT [105, 112] solvers. If a feature model's history was made publicly available, we provide all feature models representing the history.

To enable translations from every identified format, we use the transformation approach TraVarT [23], with UVL acting as pivot language (i.e., intermediate language for transformations). The most recent version of TraVarT[2] was lacking support for FeatureIDE [70], AFM [9], DIMACS, and SXFM [72], so we extended TraVarT with respective transformations.[3] For Clafer [48], we could not identify a transformation to one of the supported formats. However, every feature model we found in Clafer was also available in a format supported by TraVarT [11, 52], and we used those instead. For translation to CNF, our TraVarT extension internally uses a distributive transformation implemented in FeatureIDE [70] (v3.9.2), which preserves logical equivalence and introduces no artificial variables [58]. As some formats have different levels of expressiveness, the conversions to UVL or CNF may result in information loss [24]. In particular, our collection includes feature diagrams as presented in Section 2.1, extended feature models which contain non-functional attributes [8], and cardinality-based feature models [17]. However, in our survey we did not identify a single feature model that included constructs that cannot be semantically equivalently (i.e., represents same configuration space) represented in CNF or UVL.[4] The UVL parser [108] (also used in TraVarT) supports semantic equivalence preserving conversions to Boolean logic, which enables converting more complex expressions into CNF. In some cases, SXFM models included the same feature name twice which is not allowed in UVL [72]. Here, we resolve the duplicate feature by introducing a new feature and a respective cross-tree constraint (i.e., for duplicate feature f rename duplicate to f' with $f \iff f'$). Also, CNF does not support storing attributes, so they are discarded. In both cases, we have syntactical loss (i.e., same configuration space but different representation) [24].

Meta-Data on Feature Models For each feature model included in our repository, we provide a variety of different metrics and general information. Table 7 shows an excerpt of the different types of metrics we include in the repository separated into three categories. First, we provide structural metrics, such as the number of features. Second, we include metrics which require some semantic analyses of the feature model (e.g., number of valid configurations). We compute the structural and semantic metrics with a script based on the FeatureIDE library [70].[5] Third, we present information on the origin of the feature model, such as references (DOI) to the original work or the tooling used for conversion.

Dataset Extraction As the repository already contains 2,518 feature models today, using all of them for an empirical evaluation may be too costly. An analysis may also be only relevant for certain domains (e.g., software). In this case, users of our feature-model dataset may want to use only a subset of the feature models. To ease extracting datasets depending on the use case, we provide a Python-based script that supports to filter feature models based on the provided meta-data. Users can filter based on various properties, such as name, tags, structure, domain, or whether they are part of a history. For instance, a user of the repository may derive all feature models with more than *1,000 features* from the *software* or *automotive* domain. The extractor then provides a separate instance with the feature models matching the criteria, statistics over this dataset, and a configuration file for reusing the exact same dataset.

Dataset Configurations The extractor allows the user to derive a feature-model dataset based on configuration files. Mainly, those configurations are used to describe datasets employed in other empirical evaluations to enable easier comparability. The configurations include information on applied filters (e.g., domain), list

[1] Few feature models violated the respective language specification and could not be fully translated to UVL and DIMACS.
[2] https://github.com/SECPS/TraVarT
[3] https://zenodo.org/doi/10.5281/zenodo.11654485
[4] Note that the feature models we found may be already simplified. For instance, tri-state features from KConfig [15] may be simplified to Boolean features [82].
[5] https://zenodo.org/doi/10.5281/zenodo.11653333

Table 6: Feature Models Included in Our Repository

Domain	Feature Models	Systems	Histories	Features	Clauses	Origins
Automotive	5	2	1	2,513–18,253	666–2,833	[52, 53]
Business Software	1	1	0	1,920	59,044	[84]
Cloud	1	1	0	131	14	[33]
Database	1	1	0	117	282	[43]
Deep Learning	2	1	0	3,296–6,867	9–76	[32]
E-Commerce	2	2	0	17–2,238	0	[59, 84]
Finance	13	4	1	142–774	4–1,148	[2, 72, 77]
Games	1	1	0	144	0	[94]
Hardware	2	2	0	172–364	0–12	[72, 102]
Navigation	2	2	0	103–145	2–13	[72]
Security	1,464	2	0	101–4,351	1–8,138	[72, 114]
System Software	1,025	21	5	179–80,258	26–767,040	[2, 11, 52, 72, 78, 81, 83, 85–87]
DSL Development	1	1	0	137	1	[72]
Overall	**2,518**	**41**	**7**	**101–80,258**	**0–767,040**	

Table 7: Excerpt of Provided Metrics

Structure	Semantic†	Origin
#Features	#Valid Configurations	Publication (DOI)
#Constraints	Void	Source (URL)
Tree Depth	#Dead Features	Conversion Tool
#Leaf Features	#Atomic Sets	Domain
Constraints Density	#Core Features	Keywords

† For few feature models, metrics are missing due to scalability issues.

of used feature models, and information on the performed experiments. For now, we include some manually created configurations that describe datasets in existing empirical evaluations. We created configurations for different types of analyses, namely t-wise sampling [54], uniform random sampling [81], validity checks [63], and feature-model counting [105]. We envision that future publications can publish their configuration and reference it.[6]

Dataset Maintenance In the future, we aim to maintain and update the collection regarding the following aspects. First, we ensure that the collection is long-term publicly available. Second, we continue to add feature models that meet our criteria. Third, we review proposals for new additions. Fourth, we update the metadata about the models after new additions.

6 Evaluating Against Requirements

In this section, we aim to provide insights on the suitability of our dataset for empirical evaluations on performance of feature-model analyses. In particular, we examine different indicators on how well our dataset satisfies our requirements **P1**–**P3** which we adopted from benchmark datasets from other domains (cf. Section 3.1).

6.1 P1: Heterogeneity

In this section, we examine the heterogeneity of our dataset regarding structural properties of the feature models. Figure 6 shows the following metrics for each feature model: the number of features, number of clauses, the connectivity[7], share of features appearing in cross-tree constraints (ECR [74]), share of features that are optional (i.e., neither core nor dead), and the number of configurations. The colors of the markers indicate the domain of the feature model. Overall, the collection covers a wide range for every metric with, for instance, 101–800,258 features, 0–767,040 clauses, 10^1–10^{1534} valid configurations. Checking the similarity over various metrics, feature models from different systems structurally differ in most cases. However, feature models from similar systems are often more similar regarding structural metrics. In particular, feature models within the same history from the systems FinancialServices, uClibc, BusyBox, Fiasco, and Linux, respectively, are very similar based on various extracted metrics. Those similarities are already observable on the example metrics shown in Figure 6, as the histories form clusters in the diagrams. For instance, the standard deviation of the number of features over all feature models normalized by the mean is 4.5, while in a given history it is at most 0.32. Similarly, for constraints the overall value is 5.4 and for the different histories at most 0.44. Furthermore, the CDL feature models [52], which are not a history but different variants with a shared core, are also highly similar.

P1: Overall, our dataset covers a wide range of structural properties and most models are considerably different regarding those properties. Still, there are some clusters of feature models with very similar structural properties. Hence, it may yield benefits to only use some candidates of those similar models for evaluations.

6.2 P2: Benchmark Suitability

Here, we examine the suitability of our dataset for benchmarking regarding the two aspects for **P2** introduced in Section 3.1. First, we check if there are hard (i.e., induce long runtimes for some solvers) instances that can be used for effectively improving existing algorithms. Second, we evaluate if our dataset is sensitive enough to detect significant differences between the performances of different tools. To this end, we measure runtimes of differently performing tools, apply common statistical tests, and examine the resulting p-values. Since many feature model analyses rely on SAT and #SAT

[6] Example dataset from Liang et al. [63]: https://github.com/SoftVarE-Group/feature-model-benchmark/blob/master/paper_configs/Liang2015.json

[7] The connectivity describes the average number of literals a feature is connected to via constraints.

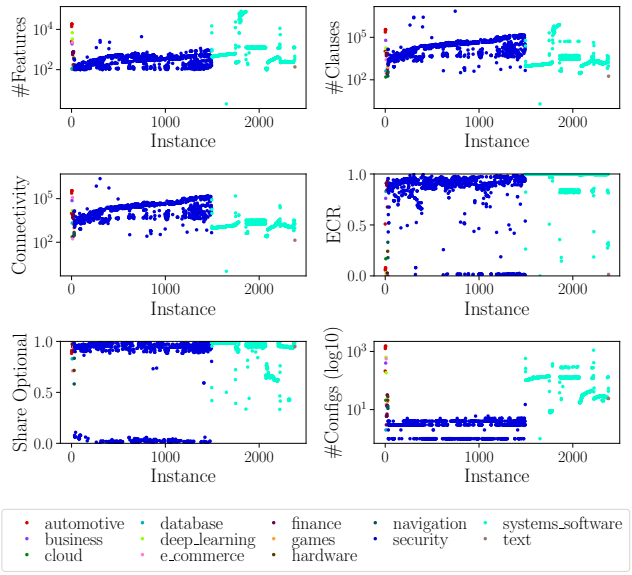

Figure 6: Structural Properties of Dataset

solving [106], we analyze solvers of both categories. In particular, for SAT, we use `Kissat` and `SBVA-Cadical` which performed best at the latest (2023) SAT competition [28] and `Glucose` which is another popular SAT solver employed in Sat4J [60] and in relevant variability modelling tools [31, 70]. For #SAT, we use the three best performing #SAT solvers from the model counting competition 2023.[8] The replication package for the runtime measurements is publicly available.[9]

Figure 7 shows the median[10] runtimes on our feature-model dataset of the SAT solvers in the first row and of the #SAT solvers in the second row. `Kissat` is the only solver which did not hit a single timeout. Hence, for all solvers but `Kissat` there is considerable potential for further optimizations and instances to evaluate potential improvements. Even for `Kissat`, there are several instances with between 0.1s and 1s of runtime which could still be used for tweaking its performance. As many feature-model analyses rely on numerous SAT invocations [106], even the short runtimes for a single `Kissat` invocation can lead to long response times.

To check the significance of differences between tools, we applied a Friedman test with a post-hoc Conover [16] respectively for both types of solvers. For each pair, we found clearly significant differences with p-values smaller than 10^{-120}. We conclude that the dataset enables significant runtime comparisons even for only smaller differences. For instance, the close difference between overall runtimes of `d4` (96.3 minutes) and `Ganak` (103 minutes) is still clearly significant ($p < 10^{-193}$).

P2: Our dataset appears to satisfy both specified requirements for being suitable for benchmarking. First, there are several hard instances that can be used for further optimization. Second, the dataset can be used to identify differences between solvers with high statistical power (i.e., very small p-values).

[8] https://mccompetition.org/assets/files/2023/MC2023_awards.pdf
[9] https://zenodo.org/doi/10.5281/zenodo.11654446
[10] We performed three repetitions per instance for each solver.

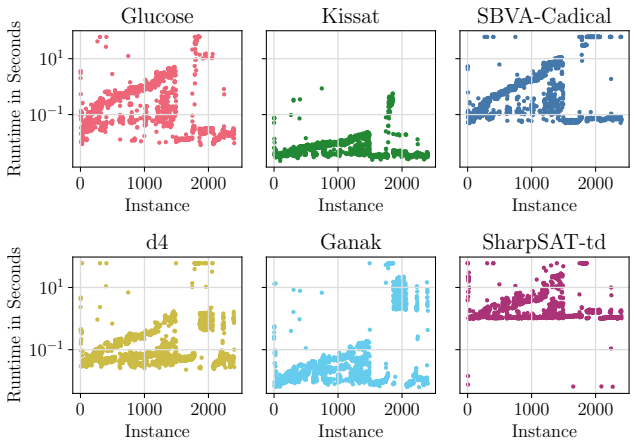

Figure 7: Runtime of Algorithms for SAT and #SAT

6.3 P3: Transferability

We only considered real-world feature models for our dataset. As described in Section 3.3, we chose a rather conservative approach where we only included feature models which we could clearly identify as real-world models or the authors explicitly declared as real-world. Furthermore, our dataset is a superset of publicly available datasets from other empirical evaluations. As we found in our survey, our resulting collection contains more large, real-world feature models and more domains compared to datasets used in previous evaluations, which increases the probability of results being transferable. In addition, the previous usage in automated analyses is an indicator that the feature models in our collection may be relevant instances for future evaluations.

P3: There are several indicators that using feature models in our dataset may improve transferability compared to previously used datasets. However, the representativeness of our dataset for other feature models in practice requires further examination. Still, since we have a superset of feature models used in previous evaluations, we assume a higher coverage of instances relevant in practice compared to previously used datasets. Thus, we expect a practical impact of our collection on future evaluations.

6.4 Discussion

Overall, we found several indicators regarding our specified requirements **P1**–**P3** that our dataset can be beneficial for empirical evaluations. First, the dataset is a superset of existing datasets used in existing empirical evaluations and covers a wide range of structural properties and domains. Second, the included feature models enable comparisons of the performance of different tools with high statistical significance. Third, due to our search criteria, we only included real-world feature models that were already used in other empirical evaluations. Still, some insights on the heterogeneity suggest that using a subset of the collection may be beneficial. We found some clusters of variants and versions of feature models that are structurally similar. As a consequence, we provide filters that support extracting collections of only one feature model per version/variant of the same cluster.

7 Related Work

Ecological Validity for Feature-Model Analysis Ecological validity describes the transferability of studies or evaluations to the real world [50], which is a major motivation for our work. Kamali et al. [50] also target ecological validity by analyzing the usage of variability models in the literature. They use the literature set collected by Galindo et al. [30] who targeted feature-model analysis. Hence, their and our survey have a similar scope. However, Kamali et al. [50] limit their elaborations on explicit open-source projects. As a consequence, they identify around[11] 80 feature models with more than 100 features, while we identified 1,617 models, not counting histories. In contrast to our work, Kamali et al. [50] also do not provide an accumulated collection of feature models.

Feature-Model Collections Several repositories providing a collection of feature models are available [11, 29, 70, 72, 91, 103]. According to the insights from our literature survey, feature models from the SPLOT [72] repository are most commonly used. Still, SPLOT contains rather small real-world feature models with 98.3% (17 out of 997) having less than 100 features. The largest real-world SPLOT feature model contains 451 features. In contrast to our collection, none of the available repositories target datasets that reasonably reflect performance in practice, but target the exchange of all kinds of feature models, including academic models.

In several publications, the authors published considerable collections including real-world feature models [5, 11, 52, 53, 76, 86, 114]. However, these datasets are limited to only few [52, 76] or even exactly one domain [11, 53, 86, 114]. As our collection is a superset of existing datasets, we cover various domains (13) and feature model sizes (101-80,528 features). Additionally, our collection comes with tool support to simplify extracting a dataset tailored to the requirements of an empirical evaluation.

Benchmarking Efforts Several publications consider benchmarking in the product-line domain [20, 95]. However, neither of the available benchmarks targets real-world feature models from multiple domains. The benchmarks either (1) use generated feature models [95], (2) focus on entirely different scopes [20, 67, 93, 118], and/or (3) are limited to a single domain or even system [67, 118].

Surveys on Feature-Model Analysis Several other surveys target work on feature-model analyses [8, 30, 36, 62, 96, 107, 110]. These surveys focus on the analyses and provide, amongst others, formal specifications [8, 36, 107], possible algorithms [36, 107] or categorizations of analyses [62, 96, 107, 110]. In contrast, we focus on the feature models by identifying properties of the feature model, examining the reuse of feature models across empirical evaluations, and gathering the feature models for our collection.

Benchmarks in Other Domains Automated reasoning engines are typically evaluated on collections of logical formulas [12, 26, 28, 112]. In particular, the SAT competition [28] and the model-counting competition [26] provide large datasets to compare solvers. These collections consist of randomly generated instances and SAT encodings of popular complex problems, such as graph coloring [26, 28]. However, these collections do not include instances representing feature models or configurable systems. Our work could be a stepping stone towards including feature-model instances in future contests. Collections for benchmarking are also used in other domains, such as graphs [45], machine learning [61], and face recognition [35]. While the respective instances are not closely related to feature models, we examined their work to derive benchmarking requirements also relevant for our scope (cf. Section 3.1).

[11]The number is not exact, as information for few feature models is missing.

8 Conclusion

Identifying relevant feature models for benchmarking is tedious as they are scattered across numerous publications. Empirical evaluations often include only small feature models and are typically limited to few domains. Also, the sets of feature models used in different evaluations often do not intersect much, which limits external validity and hinders comparability. In this work, we perform a literature survey to gather a collection of real-world feature models for benchmarking. Our dataset contains overall 2,518 real-world feature models with at least 100 features each, made available in the original format, UVL, and DIMACS. We also provide various insights on structure, semantics, origin, and suitability for benchmarking of these feature models. Our observations on the latter suggest that our dataset may be valuable for future evaluations.

In the future, multiple authors plan to maintain the dataset and extend it with new feature models. We also invite researchers and practitioners to contribute models. Extending the repository with data beyond the feature model (e.g., source code) may also be valuable for related analyses. To simplify access, we aim to add our dataset to an existing repository (e.g., UVLHub [91]). Furthermore, we plan on extending recommendations for feature models to include for specific use-cases based on model quality, general user ratings, heterogeneity, and domain.

Acknowledgments

This work is based on the master's thesis of Brancaccio [13].

References

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2013. Familiar: A Domain-Specific Language for Large Scale Management of Feature Models. *SCP* 78, 6 (2013), 657–681.
[2] Mustafa Al-Hajjaji, Thomas Thüm, Malte Lochau, Jens Meinicke, and Gunter Saake. 2019. Effective Product-Line Testing Using Similarity-Based Product Prioritization. *SoSyM* 18, 1 (2019), 499–521.
[3] Chittaranjan Andrade. 2018. Internal, External, and Ecological Validity in Research Design, Conduct, and Evaluation. *Indian Journal of Psychological Medicine* 40, 5 (2018), 498–499.
[4] Carlos Ansótegui, María Luisa Bonet, and Jordi Levy. 2009. On the Structure of Industrial SAT Instances. In *CP*. Springer, 127–141.
[5] Eduard Baranov, Axel Legay, and Kuldeep S. Meel. 2020. Baital: An Adaptive Weighted Sampling Approach for Improved t-Wise Coverage. In *ESEC/FSE*. ACM, 1114–1126.
[6] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *SPLC*. Springer, 7–20.
[7] Maurice H. ter Beek, Klaus Schmid, and Holger Eichelberger. 2019. Textual Variability Modeling Languages: An Overview and Considerations. In *SPLC*. ACM, 151–157.
[8] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. 2010. Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Information Systems* 35, 6 (2010), 615–708.
[9] David Benavides, Sergio Segura, Pablo Trinidad, and Antonio Ruiz-Cortés. 2007. FAMA: Tooling a Framework for the Automated Analysis of Feature Models. In *VaMoS*. Technical Report 2007-01, Lero, 129–134.
[10] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortés. 2005. Using Constraint Programming to Reason on Feature Models. In *SEKE*. 677–682.
[11] Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wąsowski, and Krzysztof Czarnecki. 2013. A Study of Variability Models and Languages in the Systems Software Domain. *TSE* 39, 12 (2013), 1611–1640.

[12] Armin Biere. 2008. PicoSAT Essentials. *JSAT* 4 (2008), 75–97.
[13] Vincenzo Brancaccio. 2022. *A Systematic Literature Review Towards a Representative Feature-Model Benchmark.* Master's Thesis. University of Ulm.
[14] Sheng Chen and Martin Erwig. 2011. Optimizing the Product Derivation Process. In *SPLC*. IEEE, 35–44.
[15] The Kernel Development Community. 2018. KConfig Language. Website: https://www.kernel.org/doc/html/latest/kbuild/kconfig-language.html. Accessed: 2024-01-30.
[16] William J Conover and Ronald L Iman. 1981. Rank Transformations as a Bridge Between Parametric and Nonparametric Statistics. *The American Statistician* 35, 3 (1981), 124–129.
[17] Krzysztof Czarnecki and Chang Hwan Peter Kim. 2005. Cardinality-Based Feature Modeling and Constraints: A Progress Report. In *SF*. 16–20.
[18] Krzysztof Czarnecki and Andrzej Wąsowski. 2007. Feature Diagrams and Logics: There and Back Again. In *SPLC*. IEEE, 23–34.
[19] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *TACAS*. Springer, 337–340.
[20] Sebastian Erdweg, Tijs van der Storm, Markus Völter, Laurence Tratt, Remi Bosman, William R. Cook, Albert Gerritsen, Angelo Hulshout, Steven Kelly, Alex Loh, Gabriël Konat, Pedro J. Molina, Martin Palatnik, Risto Pohjonen, Eugen Schindler, Klemens Schindler, Riccardo Solmi, Vlad Vergu, Eelco Visser, Kevin van der Vlist, Guido Wachsmuth, and Jimi van der Woning. 2015. Evaluating and Comparing Language Workbenches: Existing Results and Benchmarks for the Future. *Comput. Lang. Syst. Struct.* 44 (2015), 24–47.
[21] Sundermann et al. 2024. Repository for Feature-Model Dataset. https://zenodo.org/doi/10.5281/zenodo.11652924.
[22] Sundermann et al. 2024. Reproducibility Package for Literature Survey. https://zenodo.org/doi/10.5281/zenodo.11653969.
[23] Kevin Feichtinger, Johann Stöbich, Dario Romano, and Rick Rabiser. 2021. TRAVART: An Approach for Transforming Variability Models. In *VaMoS*. ACM, Article 8, 10 pages.
[24] Kevin Feichtinger, Chico Sundermann, Thomas Thüm, and Rick Rabiser. 2022. It's Your Loss: Classifying Information Loss During Variability Model Roundtrip Transformations. In *SPLC*. ACM, 67–78.
[25] David Fernández-Amorós, Ruben Heradio, José Antonio Cerrada, and Carlos Cerrada. 2014. A Scalable Approach to Exact Model and Commonality Counting for Extended Feature Models. *TSE* 40, 9 (2014), 895–910.
[26] Johannes K. Fichte, Markus Hecher, and Florim Hamiti. 2021. The Model Counting Competition 2020. *JEA* 26, Article 13 (2021), 26 pages.
[27] Claudia Fritsch, Richard Abt, and Burkhardt Renz. 2020. The Benefits of a Feature Model in Banking. In *SPLC*. ACM, Article 9, 11 pages.
[28] Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda. 2021. SAT Competition 2020. *AIJ* 301 (2021), 103572.
[29] José A. Galindo and David Benavides. 2019. Towards a New Repository for Feature Model Exchange. In *MODEVAR*. ACM, 170–173.
[30] José A. Galindo, David Benavides, Pablo Trinidad, Antonio-Manuel Gutiérrez-Fernández, and Antonio Ruiz-Cortés. 2019. Automated Analysis of Feature Models: Quo Vadis? *Computing* 101, 5 (2019), 387–433.
[31] José A. Galindo, José Miguel Horcas, Alexander Felfernig, David Fernández-Amorós, and David Benavides. 2023. FLAMA: A Collaborative Effort to Build a New Framework for the Automated Analysis of Feature Models. In *SPLC*. ACM, 16–19.
[32] Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2019. Automated Search for Configurations of Convolutional Neural Network Architectures. In *SPLC*. ACM, 119–130.
[33] Oscar González-Rojas and Juan Tafurth. 2019. Multi-cloud Services Configuration Based on Risk Optimization. In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences*, Hervé Panetto, Christophe Debruyne, Martin Hepp, Dave Lewis, Claudio Agostino Ardagna, and Robert Meersman (Eds.). Springer, 733–749.
[34] Jun Gu, Paul W Purdom, John Franco, and Benjamin W Wah. 1996. *Algorithms for the Satisfiability (SAT) Problem: A Survey.* Technical Report. Cincinnati University.
[35] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. 2016. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In *Computer Vision – ECCV 2016*. Springer, 87–102.
[36] Ruben Heradio, David Fernández-Amorós, José Antonio Cerrada, and Ismael Abad. 2013. A Literature Review on Feature Diagram Product Counting and Its Usage in Software Product Line Economic Models. *IJSEKE* 23, 08 (2013), 1177–1204.
[37] Ruben Heradio, David Fernández-Amorós, José Antonio Galindo, and David Benavides. 2020. Uniform and Scalable SAT-Sampling for Configurable Systems. In *SPLC*. ACM, 17:1–17:11.
[38] Ruben Heradio, David Fernández-Amorós, Christoph Mayr-Dorn, and Alexander Egyed. 2019. Supporting the Statistical Analysis of Variability Models. In *ICSE*. IEEE, 843–853.
[39] Ruben Heradio, Héctor José Pérez-Morago, David Fernández-Amorós, Roberto Bean, Francisco Javier Cabrerizo, Carlos Cerrada, and Enrique Herrera-Viedma. 2016. Binary Decision Diagram Algorithms to Perform Hard Analysis Operations on Variability Models. In *SOMET*. IOS Press, 139–154.
[40] Rubén Heradio-Gil, David Fernández-Amorós, José Antonio Cerrada, and Carlos Cerrada. 2011. Supporting Commonality-Based Analysis of Software Product Lines. *IET Software* 5, 6 (2011), 496–509.
[41] Tobias Heß, Tobias Müller, Chico Sundermann, and Thomas Thüm. 2022. ddueruem: A Wrapper for Feature-Model Analysis Tools. In *SPLC*. ACM, 54–57.
[42] Tobias Heß, Chico Sundermann, and Thomas Thüm. 2021. On the Scalability of Building Binary Decision Diagrams for Current Feature Models. In *SPLC*. ACM, 131–135.
[43] Robert M. Hierons, Miqing Li, Xiaohui Liu, Jose Antonio Parejo, Sergio Segura, and Xin Yao. 2020. Many-Objective Test Suite Generation for Software Product Lines. *TOSEM* 29, 1, Article 2 (2020), 46 pages.
[44] Jose M. Horcas, Jose A. Galindo, Mónica Pinto, Lidia Fuentes, and David Benavides. 2022. FM Fact Label: A Configurable and Interactive Visualization of Feature Model Characterizations. In *MODEVAR (SPLC)*. ACM, 42–45.
[45] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), 22118–22133.
[46] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. 2011. Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible. In *MODELS*. Springer, 638–652.
[47] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. 2012. An Algorithm for Generating T-Wise Covering Arrays From Large Feature Models. In *SPLC*. ACM, 46–55.
[48] Paulius Juodisius, Atrisha Sarkar, Raghava Rao Mukkamala, Michal Antkiewicz, Krzysztof Czarnecki, and Andrzej Wasowski. 2018. Clafer: Lightweight Modeling of Structure, Behaviour, and Variability. *CoRR* (2018).
[49] Narendra Jussien, Guillaume Rochart, and Xavier Lorca. 2008. Choco: An Open Source Java Constraint Programming Library. In *OSSICP*. CCSD-HAL.
[50] Seiede Reyhane Kamali, Shirin Kasaei, and Roberto E. Lopez-Herrejon. 2019. Answering the Call of the Wild? Thoughts on the Elusive Quest for Ecological Validity in Variability Modeling. In *MODEVAR*. ACM, 143–150.
[51] 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.
[52] Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. 2017. Is There a Mismatch Between Real-World Feature Models and Product-Line Research?. In *ESEC/FSE*. ACM, 291–302.
[53] Matthias Kowal, Sofia Ananieva, and Thomas Thüm. 2016. Explaining Anomalies in Feature Models. In *GPCE*. ACM, 132–143.
[54] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Gunter Saake, and Thomas Leich. 2020. YASA: Yet Another Sampling Algorithm. In *VaMoS*. ACM, Article 4, 10 pages.
[55] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Reimar Schröter, and Gunter Saake. 2018. Propagating Configuration Decisions With Modal Implication Graphs. In *ICSE*. ACM, 898–909.
[56] Andreas Kübler, Christoph Zengler, and Wolfgang Küchlin. 2010. Model Counting in Product Configuration. In *LoCoCo*. Open Publishing Association, 44–53.
[57] Elias Kuiter, Tobias Heß, Chico Sundermann, Sebastian Krieter, Thomas Thüm, and Gunter Saake. 2024. How Easy Is SAT-Based Analysis of a Feature Model?. In *VaMoS*. ACM, 149–151.
[58] Elias Kuiter, Sebastian Krieter, Chico Sundermann, Thomas Thüm, and Gunter Saake. 2022. Tseitin or not Tseitin? The Impact of CNF Transformations on Feature-Model Analyses. In *ASE*. ACM, 110:1–110:13.
[59] Sean Quan Lau. 2006. *Domain Analysis of E-Commerce Systems Using Feature-Based Model Templates.* Master's thesis. University of Waterloo.
[60] Daniel Le Berre and Anne Parrain. 2010. The Sat4j Library, Release 2.2. *JSAT* 7, 2-3 (2010), 59–64.
[61] Tai Le Quy, Arjun Roy, Vasileios Iosifidis, Wenbin Zhang, and Eirini Ntoutsi. 2022. A Survey On Datasets For Fairness-Aware Machine Learning. *WIREs Data Mining and Knowledge Discovery* 12, 3 (2022).
[62] Jihyun Lee, Sungwon Kang, and Danhyung Lee. 2012. A Survey on Software Product Line Testing. In *SPLC*. ACM, 31–40.
[63] Jia Hui Liang, Vijay Ganesh, Krzysztof Czarnecki, and Venkatesh Raman. 2015. SAT-Based Analysis of Large Real-World Feature Models Is Easy. In *SPLC*. Springer, 91–100.
[64] Roberto E. Lopez-Herrejon and Don Batory. 2001. A Standard Problem for Evaluating Product-Line Methodologies. In *GCSE*. Springer, 10–24.
[65] Jacob Loth, Chico Sundermann, Tobias Schrull, Thilo Brugger, Felix Rieg, and Thomas Thüm. 2023. UVLS: A Language Server Protocol for UVL. In *SPLC*. ACM, 43–46.
[66] Rafael Lotufo, Steven She, Thorsten Berger, Krzysztof Czarnecki, and Andrzej Wąsowski. 2010. Evolution of the Linux Kernel Variability Model. In *SPLC*. Springer, 136–150.
[67] Jabier Martinez, Tewfik Ziadi, Mike Papadakis, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. Feature Location Benchmark for Software

Families Using Eclipse Community Releases. In *ICSR*. Springer, 267–283.
[68] Jacopo Mauro, Michael Nieke, Christoph Seidl, and Ingrid Chieh Yu. 2017. Anomaly Detection and Explanation in Context-Aware Software Product Lines. In *SPLC*. ACM, 18–21.
[69] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. 2016. A Comparison of 10 Sampling Algorithms for Configurable Systems. In *ICSE*. ACM, 643–654.
[70] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability With FeatureIDE*. Springer.
[71] Marcilio Mendonca and Donald Cowan. 2010. Decision-Making Coordination and Efficient Reasoning Techniques for Feature-Based Configuration. *SCP* 75, 5 (2010), 311–332.
[72] Marcílio Mendonça, Moises Branco, and Donald Cowan. 2009. S.P.L.O.T.: Software Product Lines Online Tools. In *OOPSLA*. ACM, 761–762.
[73] Marcílio Mendonça, Andrzej Wąsowski, and Krzysztof Czarnecki. 2009. SAT-Based Analysis of Feature Models Is Easy. In *SPLC*. Software Engineering Institute, 231–240.
[74] Marcílio Mendonça, Andrzej Wąsowski, Krzysztof Czarnecki, and Donald Cowan. 2008. Efficient Compilation Techniques for Large Scale Feature Models. In *GPCE*. ACM, 13–22.
[75] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an Efficient SAT Solver. In *DAC*. ACM, 530–535.
[76] 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. In *SPLC*. ACM, 289–301.
[77] Michael Nieke, Jacopo Mauro, Christoph Seidl, Thomas Thüm, Ingrid Chieh Yu, and Felix Franzke. 2018. Anomaly Analyses for Feature-Model Evolution. In *GPCE*. ACM, 188–201.
[78] Michael Nieke, Gabriela Sampaio, Thomas Thüm, Christoph Seidl, Leopoldo Teixeira, and Ina Schaefer. 2022. Guiding the Evolution of Product-Line Configurations. *SoSyM* 21 (2022), 225–247. Issue 1.
[79] Jeho Oh, Don Batory, Margaret Myers, and Norbert Siegmund. 2017. Finding Near-Optimal Configurations in Product Lines by Random Sampling. In *FSE*. 61–71.
[80] Jeho Oh, Paul Gazzillo, Don Batory, Marijn Heule, and Maggie Myers. 2019. *Uniform Sampling From Kconfig Feature Models*. Technical Report TR-19-02. University of Texas at Austin, Department of Computer Science.
[81] Jeho Oh, Paul Gazzillo, Don Batory, Marijn Heule, and Margaret Myers. 2020. *Scalable Uniform Sampling for Real-World Software Product Lines*. Technical Report TR-20-01. University of Texas at Austin, Department of Computer Science.
[82] Jeho Oh, Necip Fazıl Yıldıran, Julian Braha, and Paul Gazzillo. 2021. Finding Broken Linux Configuration Specifications by Statically Analyzing the Kconfig Language. In *ESEC/FSE*. ACM, 893–905.
[83] Leonardo Passos, Marko Novakovic, Yingfei Xiong, Thorsten Berger, Krzysztof Czarnecki, and Andrzej Wąsowski. 2011. A Study of Non-Boolean Constraints in Variability Models of an Embedded Operating System. In *FOSD (SPLC)*. ACM, Article 2, 8 pages.
[84] Juliana Alves Pereira, Pawel Matuszyk, Sebastian Krieter, Myra Spiliopoulou, and Gunter Saake. 2016. A Feature-Based Personalized Recommender System for Product-Line Configuration. In *GPCE*. ACM, 12 pages.
[85] Tobias Pett, Tobias Heß, Sebastian Krieter, Thomas Thüm, and Ina Schaefer. 2023. Continuous T-Wise Coverage. In *SPLC*. ACM, 87–98.
[86] Tobias Pett, Sebastian Krieter, Tobias Runge, Thomas Thüm, Malte Lochau, and Ina Schaefer. 2021. Stability of Product-Line Sampling in Continuous Integration. In *VaMoS*. ACM, Article 18, 9 pages.
[87] Tobias Pett, Thomas Thüm, Tobias Runge, Sebastian Krieter, Malte Lochau, and Ina Schaefer. 2019. Product Sampling for Product Lines: The Scalability Challenge. In *SPLC*. ACM, 78–83.
[88] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. 2019. Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?. In *ICST*. IEEE, 240–251.
[89] Richard Pohl, Kim Lauenroth, and Klaus Pohl. 2011. A Performance Comparison of Contemporary Algorithmic Approaches for Automated Analysis Operations on Feature Models. In *ASE*. IEEE, 313–322.
[90] Steven David Prestwich. 2009. CNF Encodings. *Handbook of Satisfiability* 185 (2009), 75–97.
[91] David Romero-Organvídez, José A Galindo, Chico Sundermann, Jose-Miguel Horcas, and David Benavides. 2024. Uvlhub: A Feature Model Data Repository Using Uvl and Open Science Principles. *JSS* (2024). To appear.
[92] Pierre-Yves Schobbens, Patrick Heymans, and Jean-Christophe Trigaux. 2006. Feature Diagrams: A Survey and a Formal Semantics. In *RE*. IEEE, 136–145.
[93] Alexander Schultheiß, Paul Maximilian Bittner, Sandra Greiner, and Timo Kehrer. 2023. Benchmark Generation With VEVOS: A Coverage Analysis of Evolution Scenarios in Variant-Rich Systems. In *VaMoS*. ACM, 13–22.
[94] Sandro Schulze, Thomas Thüm, Martin Kuhlemann, and Gunter Saake. 2012. Variant-Preserving Refactoring in Feature-Oriented Software Product Lines. In *VaMoS*. ACM, 73–81.
[95] Sergio Segura, José A. Galindo, David Benavides, José A. Parejo, and Antonio Ruiz-Cortés. 2012. BeTTy: Benchmarking and Testing on the Automated Analysis of Feature Models. In *VaMoS*. ACM, 63–71.
[96] Samuel Sepúlveda and Ania Cravero. 2022. Reasoning Algorithms on Feature Modeling – A Systematic Mapping Study. *Applied Sciences* 12 (2022).
[97] Shubham Sharma, Rahul Gupta, Subhajit Roy, and Kuldeep S Meel. 2018. Knowledge Compilation Meets Uniform Sampling. In *Proc. Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning*. EasyChair, 620–636.
[98] Shubham Sharma, Subhajit Roy, Mate Soos, and Kuldeep S Meel. 2019. GANAK: A Scalable Probabilistic Exact Model Counter. In *IJCAI*, Vol. 19. AAAI Press, 1169–1176.
[99] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wąsowski, and Krzysztof Czarnecki. 2011. Reverse Engineering Feature Models. In *ICSE*. ACM, 461–470.
[100] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2012. SPL Conqueror: Toward Optimization of Non-functional Properties in Software Product Lines. *SQJ* 20, 3-4 (2012), 487–517.
[101] Stefan Sobernig and Olaf Lessenich. 2021. V1e: A Kernel for Domain-Specific Textual Variability Modelling Languages. In *VaMoS*. ACM, Article 4, 7 pages.
[102] Joshua Sprey, Chico Sundermann, Sebastian Krieter, Michael Nieke, Jacopo Mauro, Thomas Thüm, and Ina Schaefer. 2020. SMT-Based Variability Analyses in FeatureIDE. In *VaMoS*. ACM, Article 6, 9 pages.
[103] Chico Sundermann, Kevin Feichtinger, Dominik Engelhardt, Rick Rabiser, and Thomas Thüm. 2021. Yet Another Textual Variability Language? A Community Effort Towards a Unified Language. In *SPLC*. ACM, 136–147.
[104] Chico Sundermann, Tobias Heß, Dominik Engelhardt, Rahel Arens, Johannes Herschel, Kevin Jedelhauser, Benedikt Jutz, Sebastian Krieter, and Ina Schaefer. 2021. Integration of UVL in FeatureIDE. In *MODEVAR*. ACM, 73–79.
[105] Chico Sundermann, Tobias Heß, Michael Nieke, Paul Maximilian Bittner, Jeffrey M. Young, Thomas Thüm, and Ina Schaefer. 2023. Evaluating State-of-the-Art #SAT Solvers on Industrial Configuration Spaces. *EMSE* 28, 29 (2023), 38.
[106] Chico Sundermann, Elias Kuiter, Tobias Heß, Heiko Raab, Sebastian Krieter, and Thomas Thüm. 2023. On the Benefits of Knowledge Compilation for Feature-Model Analyses. *AMAI* (2023).
[107] Chico Sundermann, Michael Nieke, Paul Maximilian Bittner, Tobias Heß, Thomas Thüm, and Ina Schaefer. 2021. Applications of #SAT Solvers on Feature Models. In *VaMoS*. ACM, Article 12, 10 pages.
[108] 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 *SPLC*. ACM, 39–42.
[109] Thomas Thüm. 2020. A BDD for Linux? The Knowledge Compilation Challenge for Variability. In *SPLC*. ACM, Article 16, 6 pages.
[110] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *CSUR* 47, 1 (2014), 6:1–6:45.
[111] Thomas Thüm, Don Batory, and Christian Kästner. 2009. Reasoning About Edits to Feature Models. In *ICSE*. IEEE, 254–264.
[112] Marc Thurley. 2006. sharpSAT - Counting Models With Advanced Component Caching and Implicit BCP. In *SAT*. Springer, 424–429.
[113] Leslie G Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SICOMP* 8, 3 (1979), 410–421.
[114] Ángel Jesús Varela-Vaca, Rafael M. Gasca, Jose Antonio Carmona-Fombella, and María Teresa Gómez-López. 2020. AMADEUS: Towards the AutoMAteD SecUrity TeSting. In *SPLC*. ACM, Article 11, 12 pages.
[115] E.J. Weyuker and F.I. Vokolos. 2000. Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study. *TSE* 26, 12 (2000), 1147–1156.
[116] John Whaley. 2007. JavaBDD. https://javabdd.sourceforge.net/.
[117] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. 2012. *Experimentation in Software Engineering*. Springer.
[118] Zhenchang Xing, Yinxing Xue, and Stan Jarzabek. 2013. A Large Scale Linux-Kernel Based Benchmark for Feature Location Research. In *ICSE*. IEEE, 1311–1314.