A BDD for Linux?

The Knowledge Compilation Challenge for Variability

Thomas Thüm
University of Ulm, Germany
thomas.thuem@uni-ulm.de

Figure 1: Excerpt of a feature model for Linux 2.6.33.3 with 6,467 features and 3,545 cross-tree constraints in FeatureIDE.

ABSTRACT

What is the number of valid configurations for Linux? How to generate uniform random samples for Linux? Can we create a binary decision diagram for Linux? It seems that the product-line community tries hard to answer such questions for Linux and other configurable systems. However, attempts are often not published due to the publication bias (i.e., unsuccessful attempts are not published). As a consequence, researchers keep trying by potentially spending redundant effort. The goal of this challenge is to guide research on these computationally complex problems and to foster the exchange between researchers and practitioners.

CCS CONCEPTS

• Software and its engineering → Formal software verification; **Software testing and debugging**; **Software verification**; **Automated static analysis**; **Consistency**; **Software configuration management and version control systems**; **Preprocessors**; **• Theory of computation → Program verification**; **Program analysis**; **Logic and verification**

.

KEYWORDS

software product line, configurable system, software configuration, product configuration, feature models, decision models, artificial intelligence, satisfiability solving, knownledge compilation, binary decision diagrams

SPLC '20, October 19–23, 2020, Montreal, QC, Canada

ACM ISBN 978-1-4503-7569-6/20/10…$15.00
https://doi.org/10.1145/3382025.3414943

ACM Reference Format:
Thomas Thüm. 2020. A BDD for Linux?: The Knowledge Compilation Challenge for Variability. In *24th ACM International Systems and Software Product Line Conference (SPLC '20), October 19–23, 2020, Montreal, QC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3382025.3414943

1 MOTIVATION

What is the holy grail of the product-line community? A binary decision diagram for Linux.

Linux is an operating system with thousands of configuration options (cf. Figure 1). These options cannot be arbitrarily combined, as every option typically comes with constraints with respect to several other options. Constraints are specified in KConfig [21], but can be translated into feature models or propositional logic [11, 32, 34, 37, 40, 49, 57, 65, 68, 71, 77, 81–83]. Whenever we analyze the Linux kernel for errors, ignoring those constraints would lead to false positives. That is, tools would report errors for invalid configurations, which cannot be used to compile a kernel. Hence, these constraints are crucial for any kind of analysis of Linux.

A *binary decision diagram* (short BDD) is data structure representing a propositional formula. While there are multiple representations of propositional formulas, BDDs can have the advantage of reducing NP-complete problems into more tractable problems (aka. *knowledge compilation*) [14, 31]. For instance, checking whether a formula represented as a BDD is satisfiable is an operation with constant effort. While operations on BDDs might scale well, the downside of BDDs is that their construction can be intractable. The main reason for the scalability challenge is that the variable ordering heavily influences the size of the BDD and they tend to explode for most variable orderings.

Why do we argue that a BDD for Linux is the holy grail of the product-line community?

First, it seems to be a challenging task. To the best of our knowledge, no one has been able to create a BDD for Linux so far. In recent work, we tried to create a BDD for hundreds of large feature models and failed for 98% of them (i.e., 100% of models having more than 550 features) [80]. Furthermore, even though other knowledge compilation techniques scaled to many large feature models, not a single knowledge compilation tool scaled to Linux when we tried to compute the number of valid configurations [80]. Similarly, t-wise sampling algorithms typically do not scale to Linux [69].

Second, there is a considerable amount of research that is based on the translation of the feature model into a BDD. In the past two decades, researchers proposed the use of BDDs to count the number of valid configurations [8, 10, 47, 61, 70, 80], to compute feature-model slices [1] and differences [2], for interactive product configuration [44], to check whether product-line artifacts are consistent [29, 79], to parse preprocessor-based product lines [41], to simplify preprocessor annotations [91, 93], and to lift test-suite generation [15], data-flow analyses [12, 13], or model checking [5, 7, 17, 18, 20, 22–25, 43, 90, 91] to product lines. If we aim to apply that research to Linux or similarly complex configuration spaces, it is an open question whether BDDs can be created for them.

Third, there are many advantages of having a BDD. While it is widely accepted that satisfiability solving scales well for product lines [62, 87], the potential advantage of a BDD is that one-time effort for the construction can pay-off when the BDD is later employed in follow-up analyses. The potential is amplified by several factors. First, a feature model is typically changed less frequently than implementation artifacts [42, 54, 69]. As a consequence, a new BDD only needs to be created when the feature model changes. Second, for every new revision of the product line there are several analyses needed of which each typically is reduced to numerous satisfiability problems. It is likely that a large portion of the satisfiability problems can be solved more efficiently using a BDD.

What is the goal of this challenge?

While our claim about the holy grail is focused on BDDs and Linux, there is a more general challenge behind this specific one. Linux is just one example of a large-scale configuration space. A BDD is just one example of a knowledge compilation technique [31]. The goal of this challenge is to promote the problem of knowledge compilation for large-scale configuration spaces. In particular, this challenge is not only focused on software configuration, but also configurable systems and product configuration. Furthermore, any representation of the configuration space that invests offline computations (i.e., compilation) in favor of faster online computations would fall into the scope of this challenge.

What is the problem of the current situation? Many product-line researchers seem to address very similar problems for various reasons, but largely without documenting their failed attempts. Other researchers are most likely repeating the same mistakes again. The goal of this challenge is to make this problem explicit, to give researchers a forum to discuss attempts, and to exchange ideas on possible solutions.

2 STATE-OF-THE-ART

Knowledge compilation is the process of translating a propositional formula into a target language offline, which is then used online to answer numerous queries more efficiently (i.e., in polytime) [31]. Ideally, such a target language fulfills three properties, namely the target representation is small, many classes of queries run efficiently, and it can be efficiently translated into other target representations. Besides BDDs, there are numerous other target languages of knowledge compilation which are typically variations of conjunctive normal form or disjunctive normal form [31].

Valid combinations of features in product lines or configuration options in configurable software are typically represented by feature models, decision models, or other kinds of variability models [4, 6, 26, 27, 33, 39, 45, 48, 64, 73]. To reason about these constraints, variability models are typically translated into propositional logic [6, 30, 46]. In the past three decades, a myriad of analyses have been proposed that require reasoning about constraints. These include automated analyses of feature models [9, 38] and analyses also incorporating other domain artifacts [86, 92]. In particular, a logical representation has been used for feature-model evolution [67, 87], feature-model interfaces and slicing [1, 74], computation of implicit constraints [3], product configuration [44, 72] including staged configuration [28], parsing [50], dead-code analysis [83], code simplification [93], type checking [85], consistency checking [29], dataflow analyses [56], model checking [19], testing [16] including variability-aware execution [66] and sampling [58, 89], optimization of non-functional properties [78], and variant-preserving refactoring [35]. Each of these analyses may profit from knowledge compilation. As it is likely that numerous of such analyses are combined in practice, it is even more beneficial to invest in some offline computations if those help to speed-up several analyses later on.

Knowledge compilation has been used to reason about configuration spaces for about two decades now. Already in 2004, Hadzic et al. reported on the use of BDDs to speed-up interactive product configurators [44]. However, the algorithms to create BDDs are part of Configit's commercial products and not available to the research community. As their algorithms have never been applied to publicly available benchmarks, the actual scalability of Configit's algorithms is questionable. Czarnecki and Pietroszek used BDDs for consistency checking in model-based product lines and evaluated their approach on an e-commerce platform with about 200 features [29]. Numerous authors used BDDs to compute the number of valid configurations [10, 55, 70, 80]. Benavides et al. employed multiple solvers, such as SAT, CSP, and BDDs for the analysis of feature models in the FaMa framework [8, 10]. Mendonça investigated heuristics to scale BDDs to product lines in his dissertation [60, 63]. Besides randomly generated feature models with up-to 3,000 features, he exercised academic product lines from the SPLOT repository with up-to 200 features. Kübler et al. applied existing knowledge compilation techniques and a proprietary compilation technique to compute the number of valid configurations as well as the relative frequency of components [55]. They evaluated those techniques on automotive product lines from Mercedes-Benz with between 5,000 to 10,000 features. Only the proprietary technique was able to scale to 10,000 features with a runtime of about two hours. Pohl et al. investigated nine knowledge compilation techniques, including BDDs, CSP, and SAT solvers to compute the number of valid configurations [70]. While they only evaluated small academic product lines from the SPLOT repository, we recently extended their study with the largest known feature models from open-source projects and proprietary models [80]. While some knowledge compilation techniques scaled to all systems except Linux and one automotive product line, BDDs did not scale for almost all models. In July 2020, after the preliminary version of this challenge has been published in March 2020, Fernandez-Amoros et al. proposed a special treatment for alternative groups, which helped to scale BDDs to BusyBox, EmbToolKit, and Automotive02 with 604, 2,325, and 17,365 features, respectively [36]. However, all these product lines have under-constrained configuration spaces (i.e., only few features appear in constraints) and are therefore not representative. Even though Fernandez-Amoros et al. were aware of benchmarks with other models, they have not discussed why they excluded those and the most likely reason is that BDDs did not scale. In summary, when knowledge compilation is applied to product lines, significantly smaller or less-constrained product lines than Linux are used and negative results are rarely published.

Nevertheless, BDDs have indeed be used for the analysis of Linux and other large product lines [40, 41, 51, 66, 79, 84, 93]. However, in these cases the BDDs were only used to represent presence conditions or path conditions, which contain only a tiny portion of the product-line features. For those analyses, the feature model has either been ignored or was queried by means of SAT solvers.

3 CALL FOR CONTRIBUTIONS

Knowledge compilation can have a significant positive effect on the performance of hundreds of existing analyses. In particular, a BDD for Linux could help to scale many analyses, including t-wise sampling, uniform random sampling, or even to count the number of configurations. While it seems that numerous researchers have attempted to build a BDD for Linux, these attempts have not been successful so far and are typically not documented in the literature. We call for a community effort to advance the state-of-the-art on knowledge compilation for product lines. Besides a BDD for Linux, we call for related submissions:

- BDDs for parts of Linux or older versions containing fewer features and BDDs for Linux ignoring some constraints.
- BDDs for real-world configuration spaces (cf. existing benchmarks [52, 69]) or randomly generated models with thousands of features (cf. existing generators [62, 75, 76, 87]).
- Application of existing knowledge compilation techniques beyond BDDs [31] to variability or development of new knowledge compilation techniques dedicated to variability.
- Strategies for incremental knowledge compilation to cope with the evolution of configuration spaces.
- Attempts to solve any of the above challenges, as this documentation can prevent others from redundant research.

For comparability of solutions, we recommend using one or several of the following three benchmarks. We have translated the KConfig model of Linux in more than 400 revisions between November 2013 and January 2018 [69]. That is, not every commit of Linux is considered, but only those that actually alter the KConfig model. The models have been translated into several formats using KConfigReader [49] and the FeatureIDE library [53, 59] and are available in an online repository.[1] Nevertheless, the benchmark has two disadvantages. First, as the translation with KConfigReader uses the Tseytin transformation [88], the models contain more than 60,000 variables. Second, the feature models are flat and do not preserve the hierarchy of the features in KConfig. To the best of our knowledge, there is no better translation by now.

With support of Thorsten Berger, we translated more than 100 models from KConfig and the Component Definition Language (CDL) [52].[2] These models have been translated with an extension of the Linux Variability Analysis Tools (LVAT) [11].[3] The advantage over the previous benchmark is that the hierarchy of features is available and that models are available in a version without newly introduced variables. While the benchmark also contains Linux, the feature model represents version 2.6.33.3, which has been released in April 2010 and used for illustration in Figure 1.

In our collaborations with industry, we have been able to publish feature models of commercial models [67, 69, 74].[4] The Automotive02 models [74] and FinancialServices01 models [67] origin from the automotive and financial services industry, respectively. These models are available as monthly snapshots and feature names are obfuscated. The obfuscation algorithm being used ensures that feature names are consistently replaced over the evolution history. In four revisions, Automotive02 grew from 14,010 features and 666 constraints to a version with 18,616 features and 1,339 constraints. Similarly, FinancialServices01 grew from 557 features and 1,001 constraints to a product line with 771 features and 1,080 constraints in ten revisions.

We ask submitters to accompany their solution with a number of metrics for later comparisons. Besides the target product line, we ask about the actual source and its version. If translations were used to translate variability models into another language or into a logical representation, solutions should describe the techniques or tools being used. For the target product lines, statistics should be reported, such as the number of features, constraints, and clauses (i.e., in conjunctive or disjunctive normal form). If only a subset of the feature model is compiled, the percentage of covered features, constraints, and clauses is to be specified. For the compilation, time and memory consumption with respect to the used hardware and software should be specified. In case of success, we also expect metrics on the result of knowledge compilation, such as the number of nodes for a BDD or the number of literals of a normal form.

We explicitly invite researchers *and* practitioners to document their efforts with knowledge compilation for product lines.

ACKNOWLEDGMENTS

In the past years, I have had numerous fruitful discussions on this topic with other researchers. In particular, I would like to thank Tobias Heß, Chico Sundermann, Sebastian Krieter, Tobias Pett, Ina Schaefer, Christian Kästner, Jeffrey Young, Eric Walkingshaw, Eric Bodden, Andrzej Wąsowski, and Leopoldo Teixeira. Special thanks goes to Thorsten Berger and Alexander Knüppel for retrieving real-world feature models with hierarchy. Their efforts resulted in a benchmark published in 2017 [52] and one of those models is illustrated in Figure 1. This work has been supported by the German Research Foundation within the project VariantSync (TH 2387/1-1).

[1] https://github.com/PettTo/Feature-Model-History-of-Linux

[2] https://github.com/AlexanderKnueppel/is-there-a-mismatch/tree/master/Data/LargeFeatureModels

[3] https://code.google.com/archive/p/linux-variability-analysis-tools/

[4] https://github.com/PettTo/SPLC2019_The-Scalability-Challenge_Product-Lines

REFERENCES

[1] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert B. France. 2011. Slicing Feature Models. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. IEEE, Washington, DC, USA, 424–427. https://doi.org/10.1109/ASE.2011.6100089

[2] Mathieu Acher, Patrick Heymans, Philippe Collet, Clément Quinton, Philippe Lahire, and Philippe Merle. 2012. Feature Model Differences. In *Proc. Int'l Conf. on Advanced Information Systems Engineering (CAiSE)* (Gdansk, Poland). Springer, Berlin, Heidelberg, 629–645. https://doi.org/10.1007/978-3-642-31095-9_41

[3] Sofia Ananieva, Matthias Kowal, Thomas Thüm, and Ina Schaefer. 2016. Implicit Constraints in Partial Feature Models. In *Proc. Int'l Workshop on Feature-Oriented Software Development (FOSD)* (Amsterdam, Netherlands). ACM, New York, NY, USA, 18–27. https://doi.org/10.1145/3001867.3001870

[4] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines.* Springer, Berlin, Heidelberg.

[5] Sven Apel, Alexander von Rhein, Philipp Wendler, Armin Größlinger, and Dirk Beyer. 2013. Strategies for Product-Line Verification: Case Studies and Experiments. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (San Francisco, USA). IEEE, Piscataway, NJ, USA, 482–491. https://doi.org/10.1109/ICSE.2013.6606594

[6] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. Springer, Berlin, Heidelberg, 7–20.

[7] Shoham Ben-David, Baruch Sterin, Joanne M. Atlee, and Sandy Beidu. 2015. Symbolic Model Checking of Product-Line Requirements Using SAT-Based Methods. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Florence, Italy). IEEE, Piscataway, NJ, USA, 189–199.

[8] David Benavides. 2007. *On the Automated Analysis of Software Product Lines Using Feature Models - A Framework for Developing Automated Tool Support.* Ph.D. Dissertation. University of Seville, Spain.

[9] 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.

[10] David Benavides, Sergio Segura, Pablo Trinidad, and Antonio Ruiz-Cortés. 2007. FAMA: Tooling a Framework for the Automated Analysis of Feature Models. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Limerick, Ireland). Technical Report 2007-01, Lero, Limerick, Ireland, 129–134.

[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. *IEEE Trans. on Software Engineering (TSE)* 39, 12 (2013), 1611–1640. https://doi.org/10.1109/TSE.2013.34

[12] Eric Bodden, Társis Tolêdo, Márcio Ribeiro, Claus Brabrand, Paulo Borba, and Mira Mezini. 2013. SPLLIFT: Statically Analyzing Software Product Lines in Minutes Instead of Years. In *Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)* (Seattle, Washington, USA). ACM, New York, NY, USA, 355–364. https://doi.org/10.1145/2491956.2491976

[13] Claus Brabrand, Márcio Ribeiro, Társis Tolêdo, Johnni Winther, and Paulo Borba. 2013. Intraprocedural Dataflow Analysis for Software Product Lines. *Trans. Aspect-Oriented Software Development* 10 (2013), 73–108. https://doi.org/10.1007/978-3-642-36964-3_3

[14] Randal E. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers* C-35, 8 (Aug. 1986), 677–691. https://doi.org/10.1109/tc.1986.1676819

[15] Johannes Bürdek, Malte Lochau, Stefan Bauregger, Andreas Holzer, Alexander von Rhein, Sven Apel, and Dirk Beyer. 2015. Facilitating Reuse in Multi-Goal Test-Suite Generation for Software Product Lines. In *Proc. Int'l Conf. on Fundamental Approaches to Software Engineering (FASE)*. Springer, Berlin, Heidelberg, 84–99. https://doi.org/10.1007/978-3-662-46675-9_6

[16] Ivan Do Carmo Machado, John D. McGregor, Yguaratã Cerqueira Cavalcanti, and Eduardo Santana De Almeida. 2014. On Strategies for Testing Software Product Lines: A Systematic Literature Review. *J. Information and Software Technology (IST)* 56, 10 (2014), 1183–1199. https://doi.org/10.1016/j.infsof.2014.04.002

[17] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. 2012. Model Checking Software Product Lines with SNIP. *Int'l J. Software Tools for Technology Transfer (STTT)* 14, 5 (2012), 589–612.

[18] Andreas Classen, Maxime Cordy, Patrick Heymans, Axel Legay, and Pierre-Yves Schobbens. 2014. Formal Semantics, Modular Specification, and Symbolic Verification of Product-Line Behaviour. *Science of Computer Programming (SCP)* 80, Part B, 0 (Feb. 2014), 416–439. https://doi.org/10.1016/j.scico.2013.09.019

[19] Andreas Classen, Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, Axel Legay, and Jean-Francois Raskin. 2013. Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking. *IEEE Trans. on Software Engineering (TSE)* 39, 8 (Aug. 2013), 1069–1089. https://doi.org/10.1109/TSE.2012.86

[20] Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. 2011. Symbolic Model Checking of Software Product Lines. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Waikiki, Honolulu, HI, USA). ACM, New York, NY, USA, 321–330. https://doi.org/10.1145/1985793.1985838

[21] The Kernel Development Community. 2018. KConfig Language. Website. Available online at https://www.kernel.org/doc/html/latest/kbuild/kconfig-language.html; visited on March 13th, 2020.

[22] Maxime Cordy, Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. 2013. ProVeLines: A Product Line of Verifiers for Software Product Lines. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Tokyo, Japan). ACM, New York, NY, USA, 141–146. https://doi.org/10.1145/2499777.2499781

[23] Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. 2012. Behavioural Modelling and Verification of Real-Time Software Product Lines. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Salvador, Brazil). ACM, New York, NY, USA, 66–75. https://doi.org/10.1145/2362536.2362549

[24] Maxime Cordy, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. 2013. Beyond Boolean Product-Line Model Checking: Dealing with Feature Attributes and Multi-Features. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (San Francisco, CA, USA). IEEE, Piscataway, NJ, USA, 472–481.

[25] Maxime Cordy, Marco Willemart, Bruno Dawagne, Patrick Heymans, and Pierre-Yves Schobbens. 2014. An Extensible Platform for Product-Line Behavioural Analysis. In *Proc. Workshop on Software Product Line Analysis Tools (SPLat)* (Florence, Italy). ACM, New York, NY, USA, 102–109. https://doi.org/10.1145/2647908.2655973

[26] Krzysztof Czarnecki and Ulrich Eisenecker. 2000. *Generative Programming: Methods, Tools, and Applications.* ACM/Addison-Wesley, New York, NY, USA.

[27] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wąsowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Leipzig, Germany). ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/2110147.2110167

[28] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. 2005. Staged Configuration through Specialization and Multi-Level Configuration of Feature Models. *Software Process: Improvement and Practice* 10, 2 (2005), 143–169.

[29] Krzysztof Czarnecki and Krzysztof Pietroszek. 2006. Verifying Feature-Based Model Templates Against Well-Formedness OCL Constraints. In *Proc. Int'l Conf. on Generative Programming and Component Engineering (GPCE)* (Portland, Oregon, USA). ACM, New York, NY, USA, 211–220.

[30] Krzysztof Czarnecki and Andrzej Wąsowski. 2007. Feature Diagrams and Logics: There and Back Again. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)*. IEEE, Washington, DC, USA, 23–34.

[31] Adnan Darwiche and Pierre Marquis. 2002. A Knowledge Compilation Map. *J. Artificial Intelligence Research (JAIR)* 17, 1 (Sept. 2002), 229–264.

[32] Christian Dietrich, Reinhard Tartler, Wolfgang Schröder-Preikschat, and Daniel Lohmann. 2012. A Robust Approach for Variability Extraction from the Linux Build System. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Salvador, Brazil). ACM, New York, NY, USA, 21–30. https://doi.org/10.1145/2362536.2362544

[33] Holger Eichelberger and Klaus Schmid. 2015. Mapping the Design Space of Textual Variability Modeling Languages: A Refined Analysis. *Int'l J. Software Tools for Technology Transfer (STTT)* 17, 5 (2015), 559–584. https://doi.org/10.1007/s10009-014-0362-x

[34] Sascha El-Sharkawy, Adam Krafczyk, and Klaus Schmid. 2015. Analysing the KConfig Semantics and its Analysis Tools. In *Proc. Int'l Conf. on Generative Programming: Concepts & Experiences (GPCE)* (Pittsburgh, PA, USA). ACM, New York, NY, USA, 45–54. https://doi.org/10.1145/2814204.2814222

[35] Wolfram Fenske, Jens Meinicke, Sandro Schulze, Steffen Schulze, and Gunter Saake. 2017. Variant-Preserving Refactorings for Migrating Cloned Products to a Product Line. In *Proc. Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, Piscataway, NJ, USA, 316–326.

[36] David Fernández-Amorós, Sergio Bra, Ernesto Aranda-Escolástico, and Ruben Heradio. 2020. Using Extended Logical Primitives for Efficient BDD Building. *Mathematics* 8, 8 (2020), 1253:1–1253:17.

[37] David Fernandez-Amoros, Ruben Heradio, Christoph Mayr-Dorn, and Alexander Egyed. 2019. A KConfig Translation to Logic with One-Way Validation System. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Paris, France). ACM, New York, NY, USA, 303–308. https://doi.org/10.1145/3336294.3336313

[38] 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 (May 2019), 387–433. https://doi.org/10.1007/s00607-018-0646-1

[39] José A. Galindo, Deepak Dhungana, Rick Rabiser, David Benavides, Goetz Botterweck, and Paul Grünbacher. 2015. Supporting Distributed Product Configuration by Integrating Heterogeneous Variability Modeling Approaches. *J. Information and Software Technology (IST)* 62, C (June 2015), 78–100. https://doi.org/10.1016/j.infsof.2015.02.002

[40] Paul Gazzillo. 2017. Kmax: Finding All Configurations of Kbuild Makefiles Statically. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)* (Paderborn, Germany). ACM, New York, NY, USA, 279–290. https://doi.org/10.1145/3106237.3106283

[41] Paul Gazzillo and Robert Grimm. 2012. SuperC: Parsing All of C by Taming the Preprocessor. In *Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)* (Beijing, China). ACM, New York, NY, USA, 323–334. https://doi.org/10.1145/2254064.2254103

[42] Karine Gomes, Leopoldo Teixeira, Thayonara Alves, Márcio Ribeiro, and Rohit Gheyi. 2019. Characterizing Safe and Partially Safe Evolution Scenarios in Product Lines: An Empirical Study. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Leuven, Belgium). ACM, New York, NY, USA, Article Article 15, 9 pages. https://doi.org/10.1145/3302333.3302346

[43] Joel Greenyer, Amir Molzam Sharifloo, Maxime Cordy, and Patrick Heymans. 2013. Features Meet Scenarios: Modeling and Consistency-Checking Scenario-Based Product Line Specifications. *Requirements Engineering* 18, 2 (2013), 175–198. https://doi.org/10.1007/s00766-013-0169-4

[44] Tarik Hadzic, Sathiamoorthy Subbarayan, Rune M. Jensen, Henrik R. Andersen, Jesper Møller, and Henrik Hulgaard. 2004. Fast Backtrack-Free Product Configuration using a Precompiled Solution Space Representation. In *Proc. Int'l Conf. on Economic, Technical and Organisational Aspects of Product Configuration Systems*. Gamez Publishing, Kongens Lyngby, Denmark, 131–138.

[45] Arnaud Hubaux, Dietmar Jannach, Conrad Drescher, Leonardo Murta, Tomi Männistö, Krzysztof Czarnecki, Patrick Heymans, Tien N. Nguyen, and Markus Zanker. 2012. Unifying Software and Product Configuration: A Research Roadmap. In *Proc. Configuration Workshop (ConfWS)* (Montpellier, France). CEUR-WS.org, Aachen, Germany, 31–35. https://doi.org/10.5555/3053577.3053583

[46] Mikoláš Janota, Joseph Kiniry, and Goetz Botterweck. 2008. *Formal Methods in Software Product Lines: Concepts, Survey, and Guidelines.* Technical Report Lero-TR-SPL-2008-02. Lero, University of Limerick.

[47] Martin Fagereng Johansen, Øystein Haugen, and Franck Fleurey. 2011. Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible. In *Proc. Int'l Conf. on Model Driven Engineering Languages and Systems (MODELS)*. Springer, Berlin, Heidelberg, 638–652. https://doi.org/10.1007/978-3-642-24485-8_47

[48] 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.

[49] Christian Kästner. 2017. *Differential Testing for Variational Analyses: Experience from Developing KConfigReader.* Technical Report arXiv:1706.09357. Cornell University Library. http://arxiv.org/abs/1706.09357

[50] Christian Kästner, Paolo G. Giarrusso, Tillmann Rendel, Sebastian Erdweg, Klaus Ostermann, and Thorsten Berger. 2011. Variability-Aware Parsing in the Presence of Lexical Macros and Conditional Compilation. In *Proc. Conf. on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)* (Portland, Oregon, USA). ACM, New York, NY, USA, 805–824. https://doi.org/10.1145/2048066.2048128

[51] Christian Kästner, Alexander von Rhein, Sebastian Erdweg, Jonas Pusch, Sven Apel, Tillmann Rendel, and Klaus Ostermann. 2012. Toward Variability-Aware Testing. In *Proc. Int'l Workshop on Feature-Oriented Software Development (FOSD)* (Dresden, Germany). ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2377816.2377817

[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 *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)* (Paderborn, Germany). ACM, New York, NY, USA, 291–302. https://doi.org/10.1145/3106237.3106252

[53] Sebastian Krieter, Marcus Pinnecke, Jacob Krüger, Joshua Sprey, Christopher Sontag, Thomas Thüm, Thomas Leich, and Gunter Saake. 2017. FeatureIDE: Empowering Third-Party Developers. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Sevilla, Spain). ACM, New York, NY, USA, 42–45. https://doi.org/10.1145/3109729.3109751

[54] Christian Kröher, Lea Gerling, and Klaus Schmid. 2018. Identifying the Intensity of Variability Changes in Software Product Line Evolution. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Gothenburg, Sweden). ACM, New York, NY, USA, 54–64. https://doi.org/10.1145/3233027.3233032

[55] Andreas Kübler, Christoph Zengler, and Wolfgang Küchlin. 2010. Model Counting in Product Configuration. In *Proc. Int'l Workshop on Logics for Component Configuration (LoCoCo)* (Edinburgh, UK). Open Publishing Association, Waterloo, Australia, 44–53. https://doi.org/10.4204/EPTCS.29.5

[56] Jörg Liebig, Alexander von Rhein, Christian Kästner, Sven Apel, Jens Dörre, and Christian Lengauer. 2013. Scalable Analysis of Variable Software. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)* (Saint Petersburg, Russia). ACM, New York, NY, USA, 81–91. https://doi.org/10.1145/2491411.2491437

[57] Rafael Lotufo, Steven She, Thorsten Berger, Krzysztof Czarnecki, and Andrzej Wąsowski. 2010. Evolution of the Linux Kernel Variability Model. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Jeju Island, South Korea). Springer, Berlin, Heidelberg, 136–150.

[58] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. 2016. A Comparison of 10 Sampling Algorithms for Configurable Systems. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Austin, Texas). ACM, New York, NY, USA, 643–654. https://doi.org/10.1145/2884781.2884793

[59] Jens Meinicke, Thomas Thüm, Reimar Schröter, Fabian Benduhn, Thomas Leich, and Gunter Saake. 2017. *Mastering Software Variability with FeatureIDE.* Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-319-61443-4

[60] Marcílio Mendonça. 2009. *Efficient Reasoning Techniques for Large Scale Feature Models.* Ph.D. Dissertation. University of Waterloo, Canada.

[61] Marcílio Mendonça, Moises Branco, and Donald Cowan. 2009. S.P.L.O.T.: Software Product Lines Online Tools. In *Proc. Conf. on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)*. ACM, New York, NY, USA, 761–762.

[62] Marcílio Mendonça, Andrzej Wąsowski, and Krzysztof Czarnecki. 2009. SAT-Based Analysis of Feature Models is Easy. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (San Francisco, California). Software Engineering Institute, Pittsburgh, PA, USA, 231–240.

[63] Marcílio Mendonça, Andrzej Wąsowski, Krzysztof Czarnecki, and Donald Cowan. 2008. Efficient Compilation Techniques for Large Scale Feature Models. In *Proc. Int'l Conf. on Generative Programming and Component Engineering (GPCE)*. ACM, New York, NY, USA, 13–22.

[64] Andreas Metzger, Klaus Pohl, Patrick Heymans, Pierre-Yves Schobbens, and Germain Saval. 2007. Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis. In *Proc. Int'l Conf. on Requirements Engineering (RE)*. IEEE, Washington, DC, USA, 243–253.

[65] 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 *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Paris, France). ACM, New York, NY, USA, 289–301. https://doi.org/10.1145/3336294.3336297

[66] Hung Viet Nguyen, Christian Kästner, and Tien N. Nguyen. 2014. Exploring Variability-Aware Execution for Testing Plugin-Based Web Applications. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Hyderabad, India). ACM, New York, NY, USA, 907–918. https://doi.org/10.1145/2568225.2568300

[67] Michael Nieke, Jacopo Mauro, Christoph Seidl, Thomas Thüm, Ingrid Chieh Yu, and Felix Franzke. 2018. Anomaly Analyses for Feature-Model Evolution. In *Proc. Int'l Conf. on Generative Programming and Component Engineering (GPCE)* (Boston, MA, USA). ACM, New York, NY, USA, 188–201. https://doi.org/10.1145/3278122.3278123

[68] Jeho Oh, Paul Gazzillo, Don Batory, Marijn Heule, and Maggie Myers. 2019. *Uniform Sampling from Kconfig Feature Models.* Technical Report TR-19-02. The University of Texas at Austin, Department of Computer Science.

[69] Tobias Pett, Thomas Thüm, Tobias Runge, Sebastian Krieter, Malte Lochau, and Ina Schaefer. 2019. Product Sampling for Product Lines: The Scalability Challenge. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Paris, France). ACM, New York, NY, USA, 78–83. https://doi.org/10.1145/3336294.3336322

[70] Richard Pohl, Kim Lauenroth, and Klaus Pohl. 2011. A Performance Comparison of Contemporary Algorithmic Approaches for Automated Analysis Operations on Feature Models. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)*. IEEE, Washington, DC, USA, 313–322. https://doi.org/10.1109/ASE.2011.6100068

[71] Valentin Rothberg, Nicolas Dintzner, Andreas Ziegler, and Daniel Lohmann. 2016. Feature Models in Linux: From Symbols to Semantics. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Salvador, Brazil). ACM, New York, NY, USA, 65–72. https://doi.org/10.1145/2866614.2866624

[72] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. 2013. Scalable Product Line Configuration: A Straw to Break the Camel's Back. In *Proc. Int'l Conf. on Automated Software Engineering (ASE)* (Silicon Valley, CA, USA). IEEE, Piscataway, NJ, USA, 465–474. https://doi.org/10.1109/ASE.2013.6693104

[73] Pierre-Yves Schobbens, Patrick Heymans, Jean-Christophe Trigaux, and Yves Bontemps. 2007. Generic Semantics of Feature Diagrams. *Computer Networks* 51, 2 (2007), 456–479.

[74] Reimar Schröter, Sebastian Krieter, Thomas Thüm, Fabian Benduhn, and Gunter Saake. 2016. Feature-Model Interfaces: The Highway to Compositional Analyses of Highly-Configurable Systems. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Austin, Texas). ACM, New York, NY, USA, 667–678. https://doi.org/10.1145/2884781.2884823

[75] 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 *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Leipzig, Germany). ACM, New York, NY, USA, 63–71. https://doi.org/10.1145/2110147.2110155

[76] Sergio Segura, José A. Parejo, Robert M. Hierons, David Benavides, and Antonio Ruiz-Cortés. 2014. Automated Generation of Computationally Hard Feature Models Using Evolutionary Algorithms. *Expert Systems with Applications: An Int'l J. (EXWA)* 41, 8 (June 2014), 3975–3992. https://doi.org/10.1016/j.eswa.2013.12.028

[77] Steven She and Thorsten Berger. 2010. *Formal Semantics of the Kconfig Language.* Technical Report. University of Waterloo, Canada.

[78] 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. *Software Quality Journal (SQJ)* 20, 3-4 (Sept. 2012), 487–517. https://doi.org/10.1007/s11219-011-9152-9
[79] Julio Sincero, Reinhard Tartler, Daniel Lohmann, and Wolfgang Schröder-Preikschat. 2010. Efficient Extraction and Analysis of Preprocessor-Based Variability. In *Proc. Int'l Conf. on Generative Programming and Component Engineering (GPCE)* (Eindhoven, The Netherlands). ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/1868294.1868300
[80] Chico Sundermann, Thomas Thüm, and Ina Schaefer. 2020. Evaluating #SAT Solvers on Industrial Feature Models. In *Proc. Int'l Working Conf. on Variability Modelling of Software-Intensive Systems (VaMoS)* (Magdeburg, Germany). ACM, New York, NY, USA, Article 3, 9 pages. https://doi.org/10.1145/3377024.3377025
[81] Reinhard Tartler, Christian Dietrich, Julio Sincero, Wolfgang Schröder-Preikschat, and Daniel Lohmann. 2014. Static Analysis of Variability in System Software: The 90,000 #Ifdefs Issue. In *Proc. USENIX Annual Technical Conference (ATC)* (Philadelphia, PA). USENIX Association, Berkeley, CA, USA, 421–432.
[82] Reinhard Tartler, Daniel Lohmann, Christian Dietrich, Christoph Egger, and Julio Sincero. 2012. Configuration Coverage in the Analysis of Large-Scale System Software. *ACM SIGOPS Operating Systems Review* 45, 3 (Jan. 2012), 10–14. https://doi.org/10.1145/2039239.2039242
[83] Reinhard Tartler, Daniel Lohmann, Julio Sincero, and Wolfgang Schröder-Preikschat. 2011. Feature Consistency in Compile-Time-Configurable System Software: Facing the Linux 10,000 Feature Problem. In *Proc. Europ. Conf. on Computer Systems (EuroSys)* (Salzburg, Austria). ACM, New York, NY, USA, 47–60. https://doi.org/10.1145/1966445.1966451
[84] Reinhard Tartler, Julio Sincero, Wolfgang Schröder-Preikschat, and Daniel Lohmann. 2009. Dead or Alive: Finding Zombie Features in the Linux Kernel. In *Proc. Int'l Workshop on Feature-Oriented Software Development (FOSD)* (Denver, Colorado, USA). ACM, New York, NY, USA, 81–86.
[85] Sahil Thaker, Don Batory, David Kitchin, and William Cook. 2007. Safe Composition of Product Lines. In *Proc. Int'l Conf. on Generative Programming and Component Engineering (GPCE)*. ACM, New York, NY, USA, 95–104.
[86] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. 2014. A Classification and Survey of Analysis Strategies for Software Product Lines. *Comput. Surveys* 47, 1 (June 2014), 6:1–6:45. https://doi.org/10.1145/2580950
[87] Thomas Thüm, Don Batory, and Christian Kästner. 2009. Reasoning about Edits to Feature Models. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Vancouver, Canada). IEEE, Washington, DC, USA, 254–264. https://doi.org/10.1109/ICSE.2009.5070526
[88] G. S. Tseytin. 1983. *On the Complexity of Derivation in Propositional Calculus*. Springer, Berlin, Heidelberg, 466–483. https://doi.org/10.1007/978-3-642-81955-1_28
[89] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 2018. A Classification of Product Sampling for Software Product Lines. In *Proc. Int'l Systems and Software Product Line Conf. (SPLC)* (Gothenburg, Sweden). ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3233027.3233035
[90] Mahsa Varshosaz and Ramtin Khosravi. 2013. Discrete Time Markov Chain Families: Modeling and Verification of Probabilistic Software Product Lines. In *Proc. Int'l Workshop on Formal Methods and Analysis in Software Product Line Engineering (FMSPLE)* (Tokyo, Japan). ACM, New York, NY, USA, 34–41. https://doi.org/10.1145/2499777.2500725
[91] Alexander von Rhein. 2016. *Analysis Strategies for Configurable Systems*. Ph.D. Dissertation. University of Passau, Germany.
[92] Alexander von Rhein, Sven Apel, Christian Kästner, Thomas Thüm, and Ina Schaefer. 2013. The PLA Model: On the Combination of Product-Line Analyses. In *Proc. Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS)* (Pisa, Italy). ACM, New York, NY, USA, 14:1–14:8. https://doi.org/10.1145/2430502.2430515
[93] Alexander von Rhein, Alexander Grebhahn, Sven Apel, Norbert Siegmund, Dirk Beyer, and Thorsten Berger. 2015. Presence-Condition Simplification in Highly Configurable Systems. In *Proc. Int'l Conf. on Software Engineering (ICSE)* (Florence, Italy). IEEE, Piscataway, NJ, USA, 178–188.