Towards Sampling and Simulation-Based Analysis of Featured Weighted Automata

Maxime Cordy, Axel Legay, Sami Lazreg, Philippe Collet

Towards Sampling and Simulation-Based Analysis of Featured Weighted Automata

Maxime Cordy
SnT, University of Luxembourg
Luxembourg
maxime.cordy@uni.lu

Axel Legay
Université Catholique de Louvain
Belgium
axel.legay@uclouvain.be

Sami Lazreg
Visteon Electronics
Université Côte d'Azur
CNRS, I3S, France
lazreg@i3s.unice.fr

Philippe Collet
Université Côte d'Azur
CNRS, I3S, France
Philippe.Collet@univ-cotedazur.fr

***Abstract*—We consider the problem of model checking Variability-Intensive Systems (VIS) against non-functional requirements. These requirements are typically expressed as an optimization problem over quality attributes of interest, whose value is determined by the executions of the system. Identifying the optimal variant can be hard for two reasons. First, the state-explosion problem inherent to model checking makes it increasingly complex to find the optimal executions within a given variant. Second, the number of variants can grow exponentially with respect to the number of variation points in the VIS. In this paper, we lay the foundations for the application of smart sampling and statistical model checking to solve this problem faster. We design a simple method that samples variants and executions in a uniform manner from a featured weighted automaton and that assesses which of the sampled variants/executions are optimal. We implemented our approach on top of ProVeLines, a tool suite for model-checking VIS and carried out a preliminary evaluation on an industrial embedded system design case study. Our results tend to show that sampling-based approaches indeed holds the potential to improve scalability but should be supported by better sampling heuristics to be competitive.**

***Index Terms*—Variability, Embedded Systems, Model Checking, Sampling**

I. Introduction

Variability-Intensive Systems (VIS) encompass a large class of systems that can be derived into multiple variants, such as software product lines [1]. These systems pose a general challenge: a linear increase in the number of variation points can lead to an exponential increase of variants. Because of that, the analysis of such systems is computationally expensive, if not intractable. A common way to circumvent this consists of factorizing the analyses across multiple variants by exploiting their commonalities. For example, family-based model-checking techniques assume that common behavior between VIS should only be verified once (or as few times as possible) [2], [3].

In this work, we consider the problem of checking VIS against non-functional requirements. Such requirements are typically expressed as a constrained optimisation problem over some quality attributes of the targeted system. For example, the VIS we consider in our industrial case study are dataflow-oriented software systems embedded in automotive vehicles, such as infotainment systems and instrument clusters. Such systems typically consist of an image processing application deployed on a non-programmable platform. As such, they exhibit three main sources of variability: (1) the data processing can be achieved through the execution of alternative sequences of operations; (2) the non-programmable platforms are often developed as product lines for the manufacturers to achieve economies of scale; (3) how operations can be scheduled and allocated onto processors. Overall, all these choices can impact the feasibility of building and deploying a given system, as well as its quality attributes, i.e. in our case: manufacturing cost, computation time and quality of the graphics rendering. This raises the question of how to identify the feasible variants whose executions are optimal wrt. some trade-off between the quality attributes.

Past research has relied on variability-aware formalism to represent and verify the behaviour of VIS efficiently [2]–[9]. In particular, Featured Weighted Automata (FWA) [10] is a recent formalism able to (i) express system behaviour as a set of execution traces (i.e. sequences of transitions between system states), (ii) restrict the executability of a given transition to specific VIS variants, and (iii) associate distinct quality attributes to the execution of transitions by the variants. Intuitively, FWA can be seen as the merging of multiple weighted automata, each of which encodes the behaviour of a single variant, which makes it possible to search for the optimal executions across all variants in a single run. Thus, analyzing an FWA comes down to analyzing the equivalent set of single-variant weighted automata while avoiding going multiple times through an execution that occurs in multiple variants.

While FWA and their related algorithms succeeds in capitalising on common behaviour across multiple variants, their applications to real-world problems are hindered by multiple complexities: the state-explosion problem inherent to model checking, the high number of variants induced by variability, the quantitative computation of quality attributes. Our recent endeavour [9], [11] shows that FWA algorithms are sufficiently efficient to check isolated modules of industrial instrument cluster applications. We believe, however, that this solution will not scale up to systems of larger sizes [12].

Statistical Model Checking (SMC) techniques [13], can be seen as a trade-off between software testing and formal verification. The core idea of the approach is to conduct some

simulations of the system and verify whether they satisfy a given property. The results are then used together with algorithms from the statistical area in order to decide whether the system satisfies the property with some probability. SMC can also be used to drive the search towards rare (quantitative or not) bugs. Of course, in contrast with an exhaustive approach, a simulation-based solution does not guarantee a result with 100% confidence. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory- and time-intensive than exhaustive ones, and are sometimes the only viable option. Over the past years, SMC has been used to (1) assess the absence of errors in various areas from aeronautic to systems biology, (2) measure cost average and energy consumption for complex applications such as nanosatellites and (3) detect rare bugs in concurrent systems. SMC is not only able to estimate a probability distribution, but can also be used to optimize costs as shown in [14]. In some recent work [15], we have showed how to use smart sampling to extend SMC to non-deterministic stochastic systems. There, the main challenge is to sample among the set of schedulers introduced by the non-determinism, and then apply SMC on the resulting system.

We see an analogy between the choice of schedulers and the selection of variants. Indeed, each variant can be encoded with a weighted automaton on which SMC can be applied directly. Thus, in this paper, we propose to use a similar strategy in FWA. More precisely, we present our preliminary attempt to apply variant sampling and simulation-based methods in order to speed up the identification of optimal VIS variants. Our method iteratively samples variants and their executions from an FWA and assesses which of the sampled executions are optimal. We implemented our approach on top of ProVeLines, a tool suite for model-checking VIS and carried out a first evaluation on the instrument cluster design engineering case study of previous orks [9], [11]. Our results tend to show that sampling-based approaches indeed hold the potential to improve scalability, but should be supported by better sampling heuristics to be competitive.

The paper is structured as follows. In Section II, we give some background on FWA. We present our sampling-based method in Section III and report our evaluation results in Section IV. Section V concludes and draws the roadmap of our upcoming research on this topic.

II. Featured Weighted Automata

FWA are an extension of transition systems that capture the variation points in the behaviour of a given VIS, as well as the relationship between this behaviour and the quality attributes of the system. The starting point of this formalism is to model the variability of the considered VIS as features. That is, each feature represents a unit of variability that, if enabled, can affect the structure and the behaviour of the VIS. A variant of the VIS is then defined as the set of its enabled features.

In our case study, a feature can represent, e.g., a design choice between alternative processing operations, an optional hardware resource, a level of image quality, or an allocation of an operation (resp. an image) onto a processor (resp. a memory storage). Obviously, these features will have an impact on the quality attributes of the system. For example, the selected platform resources affect cost and performance, whereas the performed operations and the quality of the processed images determine the quality of the graphic rendering. These impacts are static, in that they depend only on the selected features and not on the behaviour actually executed by the variant. In a recent work [11], we show that this variability and its impact on quality attributes can be model in a Priced Feature Model (PFM).

Quality attributes can also be influenced by the behaviour of the system, e.g. how it will schedule the processing operations and the image store onto the available hardware resources. PFMs alone are not sufficient to represent these dynamic impacts. Instead, those are modelled as additional labels on the transitions of the FWA. These labels describe which variants can execute the transition and, for each such variant, whether and how it modifies the value of each quality attribute when executing the transition. This information can be symbolically encoded in the form of a *weighted feature expression* [11].

Definition II.1. Let t be a transition, F be a set of features and $\{\tau_1 \dots \tau_n\}$ a set of quality attributes. Then a weighted feature expression over t is a (possibly partial) function $\gamma : 2^F \to \mathbb{R}^n_{\geq 0}$ that associates a variant $F' \subseteq F$ with a vector w such that (i) the domain of γ represents the variants able to execute t and (ii) w_i is the value that F' adds to τ_i when it executes t.

Definition II.2. Let $T = \{\tau_1 \dots \tau_n\}$ be an ordered set of real-valued, quality attributes and pfm be a PFM defining a set of features F. A FWA over T and pfm is a tuple $fwa = (S, s_0, s_f, \rightarrow, \gamma_{\rightarrow})$ where S is a set of states, s_0 is the initial state, s_f is the final state, $T \subseteq S \times S$ is the transition relation, and $\gamma_{\rightarrow} : T \to 2^F \to \mathbb{R}^n_{\geq 0}$ associates each transition with a weighted feature expression.

III. Sampling Variants and Execution Paths

A valid path in an FWA is a sequence of transitions starting from the initial state and ending in the final state. Let $\pi = s_0 \to s_1 \to \cdots \to s_{k-1} \to s_f$ be a valid path of length k. Any valid path can be executed by one or more variants, which are given by $F' \in \bigcap_{i=1}^{k} dom(\gamma_{\rightarrow}(s_{i-1}, s_k))$. Then, the semantics of an FWA can be defined as a function that associates each valid variant of the induced pfm to the set of paths that this variant can execute. A weighted automaton modelling the behaviour of a given variant F' can be obtained by computing the *projection* of the FWA onto F'. Intuitively, the projection changes the weighted feature expression in order to impede the other variants to execute any transition.

Definition III.1. Let $fwa = (S, s_0, s_f, \rightarrow, \gamma_{\rightarrow})$ be an FWA over a pfm PFM and let $F' \in dom(pfm)$. Then the projection of fwa onto F' is the FWA $fwa_{|F'} = (S, s_0, s_f, \rightarrow, \gamma'_{\rightarrow})$ such that for all $t = (s, s') \in \rightarrow$, we have $dom(\gamma'_{\rightarrow})(t) = \{F'\}$ and $\gamma'_{\rightarrow}(F') = \gamma_{\rightarrow}(F')$.

Let π be some valid path in an FWA. Let F' be a product able to execute π. Then the weight vector induced by π and associated to F', noted $w_{\pi,F'}$, is obtained by summing the weight vector associated to all transitions of π, that is, $w_{\pi,F'} = \sum_{i=1}^{k} \gamma_{\rightarrow}(s_{i-1}, s_i)(F')$. Let $\zeta : \mathbb{R}^n \rightarrow \mathbb{R} : \{\tau_1 \dots \tau_n\} \rightarrow \theta_1 \times \tau_1 + \dots + \theta_n \times \tau_n$ be some cost function that defines the trade-off between all quality attributes by multipliying each of them with a coefficient θ_i. Then the benefits of executing a path π is given by $\zeta(w_{\pi,F'})$.

A complete model-checking procedure for FWA can find the path yielding an optimal cost, either for each product or across all products [10], [11]. However, in large VIS the number of paths (due to the state-explosion problem inherent to model checking) and the number of products (due to variability) can reach tremendous numbers. Therefore, it may not always be realistic or efficient to apply such procedures.

Facing this issue, we designed a sampling-based procedure to search for optimal paths in an FWA efficiently. This procedure is described in Algorithm 1. The key principles are to allocate a budget of variants N_F and a budget of executions N_π for each variant, and then to sample variants and executions until those budgets are fully consumed. First, N_F distinct variants are iteratively sampled (Line 6). There exist several methods to sample variants from feature models (see [16]–[18] for more details). In our preliminary experiments, we adopt the simplest approach that consists in setting the value of each feature randomly, following a Bernoulli distribution with parameter $p = 0.5$, and repeating this process until we obtain a valid variant according to the PFM. Once a valid variant is obtained, we compute the projection of fwa onto this variant and sample N_π paths from the projection using random walk (Line 11). By the definition of projection, we are guaranteed than all those paths are executable by the considered variant. Then, we compute the cost function of all sampled paths across all variants and return the best path with its associated variant.

Input: $fwa = (S, s_0, s_f, \rightarrow, \gamma_{\rightarrow})$;
$pfm = (F, Q, \Psi_\rho, \eta, \Psi_\tau)$; $\zeta : \zeta(\tau_1, \dots, \tau_n) \in \mathbb{R}^+$;
N_F, the budget of variants;
N_π, the budget of executions per variant;
Output: $\mathcal{F}^*$, the best sampled variants that reach s_f
ζ^*, its associated minimal cost in the sample

```
1  ζ* ← +∞;
2  i_F ← 0;
3  Excluded ← ∅;
4  while i_F < N_F do
5      i_F ← i_F + 1;
6      F' ← Sample(pfm, Excluded);
7      Excluded ← Excluded ∪ {F'};
8      i_π ← 0;
9      while i_π < N_π do
10         i_π ← i_π + 1;
11         π ← RandomWalk(fwa_|F');
12         if w_π,F' < ζ* then
13             ζ* ← w_π;
14             F* ← F';
15         end
16     end
17 end
18 return (F*, ζ*)
```

Algorithm 1: Sampling-based Model Checking for FWA

IV. Experiments

In the context of embedded system design, our recent study [11] shows that FWA can be used as an effective model of computation to identify the optimal design variants. It considers industrial automotive instrument clusters, which consist of an image processing application and a configurable hardware platform. As mentioned in the introduction, such systems can be designed in thousands of ways due to a three-source variability [9]. Finding the optimal design variants is thus inherently hard. To address this problem, we proposed a framework able to transform model instrument clusters specifications into FWA formalism in order to search efficiently and exhaustively for the optimal variant [11] in the resulting model. We implemented our model-checking algorithms in ProVeLines [19] and conducted experiments on both industrial and generated models. Our results showed that this approach exploits behavioral commonalities between system designs to speed-up remarkably the verification process.

In spite of these positive results, we are convinced that exhaustive methods are inapplicable to large-scale industrial systems. In order to evaluate our sampling-based approach against the exhaustive algorithm of our recent work [11], we implemented Algorithm 1 within ProVeLines and experimented it on 10 models reported in [11] as a comparison baseline, numbered from 0 to 9. Model #0 has been reverse-engineered from a real industrial system, while the other models were generated in a realistic way, based on real-world system topology and statistics.

Results are given in Table I. The different models are characterized by their state density (i.e. the average number of reachable states per valid variant) and their number of valid variants. We checked each model using both the exhaustive algorithm [11] and Algorithm 1 parameterized with $N_\pi = 1$ and $N_F = 50$ (except for models #1 and #2 for which we sampled 12,5% of the variants). For each model, we compute the execution time of both algorithms, the total number of states they explored, and the cost of the optimal executions they found across all products.

Unsurprisingly, the results show that sampling can reduce verification time by multiple orders of magnitude compared to the exhaustive algorithm. The downside lies in the fact that the real optimum was never sampled for any of the models. In case #0 – the real-world model – a nearly-optimum was found, yielding an increase in cost function value of a mere 12%. In the other cases, however, the increase is more significant, ranging from 61% to 135%. This calls for two improvements of our procedure: (i) more variants should be sampled and (ii) the sampling method should be smarter than simply selecting

TABLE I
EVALUATION RESULTS WHEN SAMPLING 50 VARIANTS OR 12,5% OF THEM (TIMES ARE IN SECONDS).

			ProVeLines (Exhaustive)			**ProVeLines (Sampling 50)**			**ProVeLines (Sampling 12,5%)**		
case	density	variants	time	explored	ζ^*	time	explored	ζ^* (incr. %)	time	explored	ζ^* (incr. %)
Ins. Cl. (#0)	369	1,878	2.42	224,209	556	0.98	29905	620 (+12%)	2.84	86902	620 (+12%)
Gen. #1	1,561	32	0.38	36,488	2620	0.12	6,264	5192 (+98%)		— idem —	
Gen. #2	2,250	64	1.27	71,063	4462	0.37	18,064	7542 (+69%)		— idem —	
Gen. #3	3,061	243	75.79	414,604	5,098	3.95	153,050	11,010 (+116%)	2.42	91,830	11,010 (+116%)
Gen. #4	1,656	516	8.23	564,360	3,894	1.51	82,968	6,714 (+72%)	1.94	106,259	6,300 (+61%)
Gen. #5	2,256	1,152	29.77	1,637,492	4,712	2.06	114,247	7,936 (+68%)	6.68	328,772	7,412 (+57%)
Gen. #6	1,496	1,280	8.65	642,277	4,196	1.58	74,607	6,896 (+61%)	4.80	239,342	5,608 (+34%)
Gen. #7	2,416	2,187	89.56	3,380,866	7,172	2.81	120,800	16,824 (+135%)	15.26	659,568	16,824 (+135%)
Gen. #8	1,607	12,288	75.97	2,214,258	2,564	1.89	79,607	4,500 (+76%)	47.21	2,470,220	3,976 (+55%)
Gen. #9	1,732	34,560	72.17	1,965,447	2,412	1.97	85,589	5,066 (+110%)	155.57	7,493,328	3,940 (+63%)

features individually and independently). To assess the impact of the former improvement, we repeated the experiments with a variable sample size equal to 12,5% of the total number of valid variants (see right side of Table I. In some cases (#4–6,8–9), we obtained positive effects; in others (#1,3,7), this had no impact.

Nevertheless, sampling size can be a significant factor for the performance of our sampling-based method. The fixed size (50) and the variable size (12,5%) we used have been set arbitrarily in order to carry out our preliminary study. Our future work includes the definition of heuristics to define an appropriate sample size given the considered case.

V. CONCLUSION

Although the results remain encouraging, our experiments raise the need for smarter sampling of variants and executions. A first source of inspiration lies in the methods that statically analyze feature models in an attempt to maximize diversity across the sampled variants. Still, we believe that the solution lies in combining those methods with information on the previously sampled executions. For instance, we might cover a wide diversity of states and transitions. We might favour the features that contributed to the best found executions but, at the same time, we might look for rare feature combinations that may have been ignored in the past samplings. All these heuristics could be build up in an evolutionary approach that would create a population of increasingly better variants. Nevertheless, there are plethora of potential heuristics, and assessing them constitutes our upcoming future work.

REFERENCES

[1] P. C. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, ser. SEI Series in Software Engineering. Addison-Wesley, August 2001.

[2] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. cois Raskin, "Featured transition systems: Foundations for verifying variability-intensive systems and their application to LTL model checking," *Transactions on Software Engineering*, pp. 1069–1089, 2013.

[3] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti, "Modelling and analysing variability in product families: Model checking of modal transition systems with variability constraints," *Journal of Logical and Algebraic Methods in Programming*, vol. 85, no. 2, pp. 287 – 315, 2016.

[4] M. Chechik, B. Devereux, S. M. Easterbrook, and A. Gurfinkel, "Multi-valued symbolic model-checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 12, no. 4, pp. 371–408, 2003.

[5] K. Lauenroth, S. Toehning, and K. Pohl, "Model checking of domain artifacts in product line engineering," in *IEEE/ACM ASE*, 2009, pp. 269–280.

[6] M. Cordy, P. Heymans, P.-Y. Schobbens, and A. Legay, "Behavioural modelling and verification of real-time software product lines," in *SPLC'12*. ACM, 2012.

[7] G. N. Rodrigues, V. Alves, V. Nunes, A. Lanna, M. Cordy, P. Schobbens, A. M. Sharifloo, and A. Legay, "Modeling and verification for probabilistic properties in software product lines," in *16th IEEE International Symposium on High Assurance Systems Engineering, HASE 2015, Daytona Beach, FL, USA, January 8-10, 2015*, 2015, pp. 173–180.

[8] R. Olaechea, U. Fahrenberg, J. M. Atlee, and A. Legay, "Long-term average cost in featured transition systems," in *Proceedings of the 20th International Systems and Software Product Line Conference*, ser. SPLC '16. New York, NY, USA: ACM, 2016, pp. 109–118. [Online]. Available: http://doi.acm.org/10.1145/2934466.2934473

[9] S. Lazreg, P. Collet, and S. Mosser, "Assessing the functional feasibility of variability-intensive data flow-oriented systems," in *Symposium on Applied Computing*, 2018.

[10] U. Fahrenberg and A. Legay, "Featured weighted automata," in *5th IEEE/ACM International FME Workshop on Formal Methods in Software Engineering, FormaliSE@ICSE 2017, Buenos Aires, Argentina, May 27, 2017*, 2017, pp. 51–57. [Online]. Available: https://doi.org/10.1109/FormaliSE.2017.2

[11] S. Lazreg, M. Cordy, P. Collet, P. Heymans, and S. Mosser, "Multi-faceted automated analyses for variability-intensive embedded systems," in *ICSE '19 (to appear)*, 2019.

[12] S. Lazreg, P. Collet, and S. Mosser, "Functional feasibility analysis of variability-intensive data flow-oriented applications over highly-configurable platforms," *ACM SIGAPP Applied Computing Review*, vol. 18, no. 3, pp. 32–48, 2018.

[13] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, 2010, pp. 122–135. [Online]. Available: https://doi.org/10.1007/978-3-642-16612-9_11

[14] A. David, K. G. Larsen, A. Legay, M. Mikucionis, and D. B. Poulsen, "Uppaal SMC tutorial," *STTT*, vol. 17, no. 4, pp. 397–415, 2015. [Online]. Available: https://doi.org/10.1007/s10009-014-0361-y

[15] A. Legay, S. Sedwards, and L. Traonouez, "Estimating rewards & rare events in nondeterministic systems," *ECEASST*, vol. 72, 2015. [Online]. Available: https://doi.org/10.14279/tuj.eceasst.72.1023

[16] F. Medeiros, C. Kästner, M. Ribeiro, R. Gheyi, and S. Apel, "A comparison of 10 sampling algorithms for configurable systems," in *ICSE'16*.

[17] M. Varshosaz, M. Al-Hajjaji, T. Thüm, T. Runge, M. R. Mousavi, and I. Schaefer, "A classification of product sampling for software product lines," in *SPLC '18*, 2018, pp. 1–13.

[18] Q. Plazar, M. Acher, G. Perrouin, X. Devroey, and M. Cordy, "Uniform sampling of sat solutions forconfigurable systems: Are we there yet?" in *ICST '19 (to appear)*, 2019.

[19] M. Cordy, P.-Y. Schobbens, P. Heymans, and A. Legay, "Provelines: A product-line of verifiers for software product lines," in *SPLC'13*. ACM, 2013, pp. 141–146.