Variability-aware Behavioural Learning

Sophie Fortz
sophie.fortz@unamur.be
PReCISE, NaDI, University of Namur
Namur, Belgium

ABSTRACT

Addressing variability proactively during software engineering activities means shifting from reasoning on individual systems to *reasoning on families of systems*. Adopting appropriate variability management techniques can yield important *economies of scale* and quality improvements. Conversely, variability can also be a curse, especially for Quality Assurance (QA), *i.e., verification and testing* of such systems, due to the combinatorial explosion of the number of software variants. Featured Transition Systems (FTSs) were introduced as a way to represent and reason about the behaviour of Variaility-intensive Systems (VISs). By labelling a transition system with feature expressions, FTSs capture multiple variants of a system in a single model, enabling reasoning at the family level. They have shown significant improvements in automated QA activities such as model-checking and model-based testing, as well as guiding design exploration activities. Yet, as most model-based approaches, FTS modelling requires both *strong human expertise and significant effort* that would be unaffordable in many cases, in particular for large legacy systems with outdated specifications and/or systems that evolve continuously.

Therefore, this PhD project aims *to automatically learn FTSs from existing artefacts*, to ease the burden of modelling FTS and support continuous QA activities. To answer this research challenge, we propose a two-phase approach. First, we rely on deep learning techniques to locate variability from execution traces. For this purpose, we implemented a tool called *VaryMinions*. Then, we use these annotated traces to learn an FTS. In this second part, we adapt the seminal L^* algorithm to *learn behavioural variability*. Both frameworks are open-source and we evaluated them separately on several datasets of different sizes and origins (*e.g.,* software product lines and configurable business processes).

CCS CONCEPTS

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

KEYWORDS

Software Product Lines, Featured Transition Systems, Reverse Engineering, Active Automata Learning, Variability Mining

ACM Reference Format:
Sophie Fortz. 2023. Variability-aware Behavioural Learning. In *27th ACM International Systems and Software Product Line Conference - Volume B (SPLC '23), August 28-September 1, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3579028.3609007

1 INTRODUCTION AND MOTIVATION

Variability-Intensive Systems (VISs) are families of systems with different specificities, while sharing a common purpose. Each individual member of this family is referred to as a *variant* or a *product*, which is defined by a combination of *features* (i.e., specific characteristics). VISs encompass a wide range of applications, including Software Product Lines (SPLs), configurable systems or adaptive systems.

For Quality Assurance (QA) activities, such as verification and validation, one of the challenges posed by *Variability-Intensive Systems (VISs)* lies the combinatorial explosion of variants. Even with a relatively modest number of boolean features (*i.e.,* options that can be either activated or deactivated), such as 33, the number of possible variants exceeds the population of the Earth. With 320 features, the number of variants surpasses the number of atoms in the universe. To provide a comparison, the Linux Kernel counts approximately 15, 000 distinct features [30, 32], which are not limited to boolean values but can have different types or attributes. When considering the verification and validation of each product configuration individually, tackling the problem becomes practically infeasible. One solution to mitigate the impact of combinatorial explosion is to reason on *family* models. Model-based approaches facilitate the automation of numerous QA tasks, offering a viable approach to address this challenge.

Several approaches have been defined to model the *behaviour* of VISs. Classen *et al.* [8–10] defined *Featured Transition Systems (FTS)* as a way to represent an entire SPL in a compact way. FTSs represent the behaviour of an SPL by a Labelled Transition System [5, 19], whose labels are called Feature expressions. *Feature Expressions* relate structural variability (*i.e.,* SPL features) to behaviour, specifying which subset of products is able to execute each transition. Thus, FTS take advantage of *shared behaviour* and allow substantial scale economies to perform analysis tasks, such as testing and model checking.

Yet, VIS are rarely shipped with FTSs. Engineers usually provide these models by hand, which is time-consuming, error-prone and does not scale to complex VISs. Current inference approaches [11, 12, 16, 34] either suffer from scalability issues (enumerative, variant-based learning) or lack of automation (manual feature annotation). These existing approaches often overlook the key strength of variability models, which is their capability to express shared behaviours among different products. Hence, the main objective of this research is to tackle these limitations and automate the creation of FTS, to alleviate the burden associated with modelling FTS and enhance continuous QA activities. By harnessing the power of automated learning techniques, this research aims to unleash the full potential of variability models in expressing shared behaviours. This, in turn, will enable more streamlined and effective verification and validation processes, leading to improved software quality.

To address the current *automation* and *scalability* issues identified in the state-of-the-art, we take commonalities between variants into account right from the early stages of learning. As FTS is a fundamental formalism that can serve as a semantics for other VIS modelling languages such as UML State Diagrams (*e.g.,* via flattening [14]), the results are intended to *be generic and therefore have a profound impact on behavioural inference and automation.*

2 RESEARCH QUESTIONS

This PhD project aims to answer the following research questions:

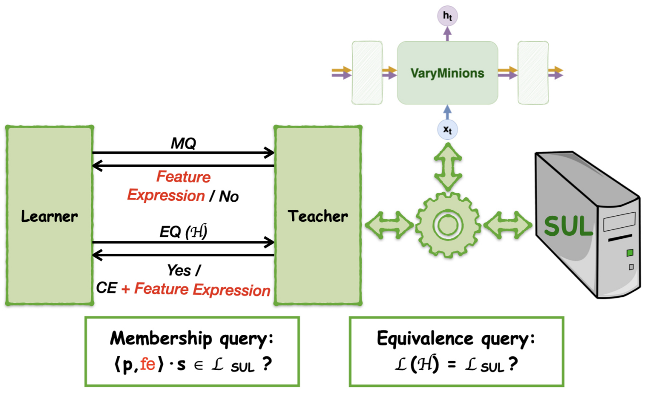
- RQ_1 **How to automatically learn Featured Transition Systems in order to ease the burden of modelling FTS and support continuous VIS QA activities?**
  - $RQ_{1.1}$ *What amount of time does a variability-aware learning approach requires?* This question addresses the feasibility, efficiency and scalability of our approach.
  - $RQ_{1.2}$ *Does variability-aware learning lead to fewer membership queries than state-of-the-art approaches?* By limiting the number of queries, the learning process becomes more efficient and faster, thereby allowing better scaling on large systems.
  - $RQ_{1.3}$ *Does variability-aware learning lead to fewer learning rounds and equivalence queries than state-of-the-art approaches?* Equivalence queries are typically very expensive in a learning algorithm, requiring significant computational resources and time. Therefore, it is crucial to minimise their number whenever possible.
  - $RQ_{1.4}$ *Does variability-aware learning lead to fewer resets than state-of-the-art approaches?* Resets allow a system to come back to its original state, as if we had just restarted it. Resets are difficult to avoid during learning process, but they are time-consuming and should be avoided.
- RQ_2 **How can we classify previously unseen behaviour to multiple variants of a VIS?**
  - $RQ_{2.1}$ *How accurately can we identify process variants based on their traces?* To the best of our knowledge, we propose the first attempt to use recurrent neural networks (RNNs) to learn such a mapping, we cannot compare it with the state of the art. Instead, we expect the RNNs to be at least more accurate than random classifiers (accuracy higher than $> 50\%$).
  - $RQ_{2.2}$ *What is the performance of long-short term memory (LSTM) versus that of gated recurrent unit (GRU) for process traces classification?* We would like to know which model architecture is the most appropriate for this task if any.

Hypothesis. In order to focus on the behavioural aspects, *we assume that the Feature Model either already exists or has been learned in some way.* Various studies have already attempted to learn structural variability, depending on the source of information available. We can highlight different methods, such as natural language analysis [27] or static analysis of variant configurators [33]. Other studies learned FM from variant catalogues (product tables), either using data mining [1, 2] or evolutionary algorithms [28]. These techniques were summarised in a recent systematic literature review [29]. Besides, Ramos-Gutiérrez *et al.* [31] use process mining to retrieve the process of configuring an SPL.

3 METHODOLOGY AND APPROACH

To address these research questions, we propose a two-phase framework, depicted in Figure 1.

In order to learn an FTS, we rely on Dana Angluin's $\mathbf{L}^*$ **learning algorithm** [3]. The L^* algorithm follows a simple metaphor where a Learner component constructs a model iteratively. The Learner make queries to another component known as the Teacher, which acts as a proxy for the system we want to model. If the learned model is incorrect, the Teacher provides counterexamples to guide the learning process. In this PhD, we adapt $\mathbf{L}^*$ algorithm to learn the behaviour of systems in a family-based and symbolic manner, *treating feature expressions as first class citizen.*



h_t
VaryMinions
x_t
MQ
Feature Expression / No
Learner
EQ ($\mathcal{H}$)
Yes / CE + Feature Expression
Teacher
SUL
Membership query: $\langle p, fe \rangle \cdot s \in \mathcal{L}_{SUL}$?
Equivalence query: $\mathcal{L}(\mathcal{H}) = \mathcal{L}_{SUL}$?

Figure 1: LiFTS General Framework

Learning the VIS in a family-based fashion *requires to relate Angluin's queries and counterexamples to configurations.* However, since the Teacher only knows about previously observed variants, the existing mappings are incomplete as they rely on partial observations of the system. All new configurations are considered unknown. Therefore, a configuration prediction technique is required to bridge this gap and accurately relate queries and counterexamples to specific configurations. Being able to locate variations is also an essential part of any re-engineering endeavour [4] and is naturally useful for testing techniques, notably to sample which variants should be tested [23]. Existing variant analysis [35] techniques rather focusing on the differences between identified variants than identifying which variant(s) may have produced a given trace. To address these challenges, we propose to *leverage deep learning techniques to develop an efficient approach for variability localisation.* By harnessing the power of deep learning, we aim to enhance the capability of accurately identifying and localising variations within the VIS, thus contributing to improved understanding and testing of software variants. This second project is called VaryMinions.

4 PRELIMINARY RESULTS

Our results are divided in two parts, each of them answering one of the general research questions.

4.1 Variability-L*

To address RQ$_1$ (and its sub-questions), we introduce a novel model called Featured Deterministic Finite Automaton (FDFA), which serves as a finite and deterministic version of Featured Transition Systems. Transition systems (and by extension FTSs) generally represent infinite behaviours. However, Since our learning approach is based on finite execution trace, it naturally defines final states. We thus defined FDFA as a way to represent finite behaviour in an FTS-like structure. We then present a new algorithm, $Featured-L^*$ (FL^*), that fundamentally differs from previous approaches.

While Tavassoli, Damasceno, *et al.* [11, 12, 34] has contributed valuable insights into learning in behavioural models of SPLs, their approaches relies on a product-based perspective. This limits the applicability to systems with few features. In this PhD project, we aim to learn a unified model, rather than treating each product individually (as in [11, 34]). We propose a more comprehensive and integrated approach, where feature expression are treated as first-class citizens. The notion of software family becomes the core of our approach, emphasising the importance of considering the relationships between features.

We also provide an implementation of FL^* called LiFTS, which successfully learns five distinct case studies within a short time frame ranging from a few seconds to less than two hours. Each case studies has between 5 and 25 features, defining 6 to $4,774$ different variants. Furthermore, we highlight the unique aspects of FL^* compared to previous approaches. Additionally, we define a visualisation aid for FTS/FDFA models.

Variability-L∗ contributions:

(1) A new formalism, FDFA, as a variation FTS, allowing finite behaviour and deterministic by construction;
(2) FL^*, a new variation of L^* algorithm, to automate the behaviour modelling of an SPL, considering variability from the early stage of learning;
(3) LiFTS, an implementation of FL^*, which successfully learns 5 distinct case studies (with 5 to 24 input symbols) within short time frame, ranging from a few seconds to less than two hours for each case study;
(4) a first comparison with the original algorithm and the latest state-of-the-art approaches, demonstrating a significant reduction in the number of queries when variability is handled explicitly;
(5) a procedure to ease FTS and FDFA visualisation, by leveraging the capabilities of a Neo4J graph database.

4.2 VaryMinions

To address RQ$_2$, we draw inspiration from natural language processing (NLP) techniques. We establish an analogy between an NLP sentence and an execution trace, where words or actions are arranged into valid sentences based on a specific grammar. Leveraging this analogy, VaryMinions utilises RNNs to classify VIS behaviours among different variants that can reproduce them. However,in NLP context, classical RNNs tend to suffer from the vanishing or exploding gradient problem [7, 24] when dealing with long sentences. To overcome this issue, we chose more specified RNN architectures: GRUs and LSTMs. We evaluate VaryMinions on six diverse datasets, encompassing both SPL and business process domains. Each dataset consists of up to 50 variants and $5,000$ event traces per variant.

We decided to vary only a few of hyperparameters to try to understand how much impact they may have on learning. In total, we evaluate 20 distinct RNN parameterisations on each of the 6 datasets. Based on our previous experiences [21], we decide to set the number of units to 30 which has shown relatively good performances while limiting the training time. The percentage of the data used for training is set to 66% of the whole dataset which is a common value in the ML community, the remaining traces are used in the test set to assess the generalisation performances of the trained models. We set the batch size to 128, which is adapted to the dataset size. We set the number of epochs to 20 to avoid overfitting. In our preliminary evaluations (evaluated between 10 and 50 epochs), a plateau was reached after approximately 15 epochs. We thus set the number of epochs to 20, to allow for small increases in accuracy. During training, Loss functions are used to optimise the weights of the networks by back-propagating errors. We have used three loss functions already implemented in tensorflow [1], namely Binary Cross-Entropy (with and without logits, respectively named hereafter Bin-CE and Bin-CE logits) and the Mean Squared Error (MSE). Since we represent each variant by an element in a vector, the error can be defined as a difference between two vectors. We thus propose two new custom loss functions: a variant of the Jaccard distance [26] (named Weight_Jaccard hereafter, already used to evaluate trace dissimilarity in VISs, *e.g.,* [18]), and the Manhattan distance between two vectors (sometimes called L1 norm). Finally, we experimented with two common activation functions which are sigmoid and hyperbolic tangent (tanh).

The training and performance evaluation process is done as follows: i) the entire dataset is randomly split into training and test sets. We have used the Keras function *train_test_split* [2] that ensures the data distribution of classes among the two sets are similar. ii) A model is trained using the training set. iii) Its prediction performances are evaluated on the test set. We mitigate biases in our analyses by repeating the whole process ten times, with a new split between train and test sets. In total, the evaluation took approximately 150 days of execution.

VaryMinions contributions:

(1) the first family-based approach, which we called VaryMinions, to map execution traces to variants of a system. We showed empirically that VaryMinions can distinguish 50 variants from $5,000+$ event traces per variant;
(2) a detailed account on the usage of Long Short Term Memory (LSTMs) [25] and Gated Recurrent Units (GRUs) [6], two RNN architectures, on six different datasets, describing business processes and course management system variants, showing that we can identify the variant(s) producing an event trace with high accuracy (> 80%);

[1]https://www.tensorflow.org/api_docs/python/tf/keras/losses
[2]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

(3) four datasets openly available and based on Claroline [13, 15, 17] and containing $2 * 10$ and $2 * 50$ software variants with $5,000$ traces per variants;
(4) a characterisation of the learning difficulty based on the behaviour shared amongst event traces.

The first results of this thesis has been already published in the following contributions:

[20] Sophie Fortz. Lifts: Learning featured transition systems. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B, Leicester, United Kindom*, SPLC '21, page 1–6, New York, NY, USA, 2021. Association for Computing Machinery (Doctoral Symposium) (Doctoral Symposium)

[21] Sophie Fortz, Paul Temple, Xavier Devroey, Patrick Heymans, and Gilles Perrouin. Varyminions: leveraging RNNs to identify variants in event logs. In Apostolos Ampatzoglou, Daniel Feitosa, Gemma Catolino, and Valentina Lenarduzzi, editors, *Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution, Athens, Greece, 23 August 2021*, pages 13–18. ACM, 2021

Variability-aware L^* is currently unpublished, but we have plans to address this and publish it in the near future.

Open Science Policy. We have provided a replication package that includes an implementation of VaryMinions, along with all the related results [22]. In the future, we intend to offer a similar replication package for Variability-L^*, after conducting a thorough evaluation.

5 WORK PLAN

Since the 2021 edition of the Doctoral Symposium [20], we pursued our work on trace classification among multiple variants. We provide a statistical evaluation on 4 new custom datasets from SPL, in addition to our 2 previous business process datasets. We also improved our implementation of VaryMinions.

During this period, we also participated to several scientific events, including:

- the workshop of the Interdisciplinary Centre for Security, Reliability and Trust (SnT group from Luxemburg, event taking place in Namur, 2021);
- the 44th ICSE (Virtual, Pittsburgh, USA, 2022);
- the annual workshop of the EOS project on Verifying Learning Artificial Intelligence Systems (Leuven, Belgium, 2022);
- the annual days of the Programming and Software Engineering Research Group (GDR-GPL Days, Vannes, France, 2022);
- the 26th SPLC (Graz, Austria, 2022);
- the PhD day of the Computer Science faculty (Namur, Belgium, 2022);
- the Women & Girls in Sciences Day (Namur, Belgium, 2022 and 2023 editions);
- the 17th VaMoS (Odense, Denmark, 2023).

A poster was presented at the 2022 edition of Women & Girls in Sciences and short presentations were given at the SnT workshop, the PhD day and the 2023 edition of Women & Girls in Sciences.

Future Work. This PhD is planned to be defended publicly by the end of September 2023. Until then, we pursue the work on Variability-aware L^*, particularly its evaluation. We envision a statistical analysis for a better comparison with the state-of-the-art and assessing the impact of family-based methods versus product-based methods.

Then, we envision several research directions:

- *Combining VaryMinions and Variability-aware L^*:* We propose an integrated approach for the LiFTS project, suggesting the integration of VaryMinions into our implementation of Variability-aware L^* to enhance the capabilities of the Teacher. We discus the joint learning of structural and behavioural aspects, addressing limitations in the current assumption of a pre-existing feature model in LiFTS, and proposing the development of a comprehensive platform for FTS manipulation.
- *Studying variability-aware equivalence queries:* We aim to investigate techniques and approaches to handle equivalence queries in the context of VISs, to improve the scalability, accuracy, and efficiency of equivalence testing in the presence of variability. This research can contribute to the development of more reliable and robust techniques for learning and verifying variability-intensive systems, enabling better understanding and analysis of their behavioural properties.
- *Generalising the framework and improving its scalability:* To tackle scalability challenges in the context of variability, we suggest the use of counterexamples to approximate equivalence, prioritisation techniques for exploring the behaviour space efficiently, and extending the SUL interface to interact directly with a real system for practical evaluation. We also consider memory usage, optimisations through parallelism and integration with existing frameworks.
- *Developing a feature-based approach data representation for VaryMinions:* To overcome the impracticality of enumerating and executing all variants in large-scale VISs, a potential future direction is to shift the data representation. Instead of explicitly considering each variant, system configurations can be described based on their features. This approach enables precise identification of feature combinations that correspond to specific behaviours, offering valuable insights for fault localisation and repair techniques where identifying the specific feature combination causing an issue is crucial.
- *Studying other neural network architectures:* To optimise the classification performance in trace-to-variant mapping, we need to consider factors such as loss functions, activation functions, network complexity, and data labelling. We suggest exploring alternative loss functions, custom activation functions, balancing network complexity and generalisation capability, and leveraging semi-supervised learning techniques to reduce the labelling effort and improve prediction performance.

Overall, we provide perspectives and potential research directions for advancing variability-aware automata learning and improving the association of behaviour with VIS configurations, with a focus on integration, scalability, data representation, and neural network optimisation.

ACKNOWLEDGEMENTS

Sophie Fortz is supported by the FNRS via a FRIA grant.

REFERENCES

[1] Mathieu Acher, Benoit Baudry, Patrick Heymans, Anthony Cleve, and Jean-Luc Hainaut. 2013. Support for reverse engineering and maintaining feature models. In *VaMoS*, Stefania Gnesi, Philippe Collet, and Klaus Schmid (Eds.). ACM, 20.

[2] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. 2012. On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*. 45–54.

[3] Dana Angluin. 1987. Learning regular sets from queries and counterexamples. *Information and computation* 75, 2 (1987), 87–106. https://doi.org/10.1016/0890-5401(87)90052-6

[4] Wesley Klewerton Guez Assunção, Roberto Erick Lopez-Herrejon, Lukas Linsbauer, Silvia Regina Vergilio, and Alexander Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Engineering* 22 (2017), 2972–3016.

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

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

[9] Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, Axel Legay, and Jean-François Raskin. 2010. Model checking lots of systems: efficient verification of temporal properties in software product lines. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. 335–344.

[10] Maxime Cordy, Xavier Devroey, Axel Legay, Gilles Perrouin, Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Jean-François Raskin. 2019. A decade of featured transition systems. In *From Software Engineering to Formal Methods and Tools, and Back*. Springer, 285–312.

[11] Carlos Diego N Damasceno, Mohammad Reza Mousavi, and Adenilso da Silva Simao. 2019. Learning to reuse: Adaptive model learning for evolving systems. In *International Conference on Integrated Formal Methods*. Springer, 138–156.

[12] Carlos Diego Nascimento Damasceno, Mohammad Reza Mousavi, and Adenilso da Silva Simao. 2021. Learning by sampling: learning behavioral family models from software product lines. *Empirical Software Engineering* 26, 1 (2021), 1–46.

[13] Xavier Devroey. 2020. VIBeS Case Studies: Featured Transition Systems and Feature Models. https://doi.org/10.5281/zenodo.4105900. https://doi.org/10.5281/zenodo.4105900

[14] X. Devroey, M. Cordy, P. Schobbens, A. Legay, and P. Heymans. 2015. State machine flattening, a mapping study and tools assessment. In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 1–8. https://doi.org/10.1109/ICSTW.2015.7107408

[15] Xavier Devroey, Gilles Perrouin, Maxime Cordy, Hamza Samih, Axel Legay, Pierre-Yves Schobbens, and Patrick Heymans. 2017. Statistical prioritization for software product line testing: an experience report. *Softw. Syst. Model.* 16, 1 (2017), 153–171. https://doi.org/10.1007/s10270-015-0479-8

[16] Xavier Devroey, Gilles Perrouin, Maxime Cordy, Hamza Samih, Axel Legay, Pierre-Yves Schobbens, and Patrick Heymans. 2015. Statistical prioritization for software product line testing: an experience report. *Software & Systems Modeling* (2015), 1–19. https://doi.org/10.1007/s10270-015-0479-8

[17] Xavier Devroey, Gilles Perrouin, Maxime Cordy, Pierre-Yves Schobbens, Axel Legay, and Patrick Heymans. 2014. Towards Statistical Prioritization for Software Product Lines Testing. In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems* (Sophia Antipolis, France) *(VaMoS '14)*. Association for Computing Machinery, New York, NY, USA, Article 10, 7 pages. https://doi.org/10.1145/2556624.2556635

[18] Xavier Devroey, Gilles Perrouin, Axel Legay, Pierre-Yves Schobbens, and Patrick Heymans. 2016. Search-based similarity-driven behavioural SPL testing. In *Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems*. 89–96.

[19] Dario Fischbein, Sebastian Uchitel, and Victor Braberman. 2006. A foundation for behavioural conformance in software product line architectures. In *Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis*. 39–48.

[20] Sophie Fortz. 2021. LIFTS: Learning Featured Transition Systems. In *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B, Leicester, United Kindom* (Leicester, United Kindom) *(SPLC '21)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3461002.3473066

[21] Sophie Fortz, Paul Temple, Xavier Devroey, Patrick Heymans, and Gilles Perrouin. 2021. VaryMinions: leveraging RNNs to identify variants in event logs. In *Proceedings of the 5th International Workshop on Machine Learning Techniques for Software Quality Evolution, Athens, Greece, 23 August 2021*, Apostolos Ampatzoglou, Daniel Feitosa, Gemma Catolino, and Valentina Lenarduzzi (Eds.). ACM, 13–18. https://doi.org/10.1145/3472674.3473980

[22] Sophie Fortz, Paul Temple, Xavier Devroey, Patrick Heymans, and Gilles Perrouin. 2022. VaryMinions. https://zenodo.org/record/7492126. https://doi.org/10.5281/zenodo.7492126 Sophie Fortz is supported by the FNRS via a FRIA grant. Gilles Perrouin is an FNRS Research Associate..

[23] Axel Halin, Alexandre Nuttinck, Mathieu Acher, Xavier Devroey, Gilles Perrouin, and Benoit Baudry. 2019. Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack. *Empir. Softw. Eng.* 24, 2 (2019), 674–717. https://doi.org/10.1007/s10664-018-9635-4

[24] Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 02 (1998), 107–116. https://doi.org/10.1142/S0218488598000094

[25] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[26] Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37 (1901), 547–579.

[27] Yang Li, Sandro Schulze, and Gunter Saake. 2017. Reverse Engineering Variability from Natural Language Documents: A Systematic Literature Review. In *SPLC'17 - Volume A* (Sevilla, Spain) *(SPLC '17)*. ACM, New York, NY, USA, 133–142. https://doi.org/10.1145/3106195.3106207

[28] Roberto E. Lopez-Herrejon, Lukas Linsbauer, and Alexander Egyed. 2015. A systematic mapping study of search-based software engineering for software product lines. *Information and Software Technology* 61 (2015), 33 – 51. https://doi.org/10.1016/j.infsof.2015.01.008

[29] Jabier Martinez and Ali Parsai. 2018. *D3.1: Identification of relevant state of the art.* Technical Report. ITEA 3 ReVAMP2 Project Consortium.

[30] Johann Mortara and Philippe Collet. 2021. *Capturing the Diversity of Analyses on the Linux Kernel Variability.* Association for Computing Machinery, New York, NY, USA, 160–171. https://doi.org/10.1145/3461001.3471151

[31] Belén Ramos-Gutiérrez, Ángel Jesús Varela-Vaca, José A Galindo, María Teresa Gómez-López, and David Benavides. 2021. Discovering configuration workflows from existing logs using process mining. *Empirical Software Engineering* 26, 1 (2021), 1–41.

[32] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2010. The Variability Model of The Linux Kernel. *VaMoS* 10, 10 (2010), 45–51.

[33] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2011. Reverse engineering feature models. In *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 461–470.

[34] Shaghayegh Tavassoli, Carlos Diego N Damasceno, Ramtin Khosravi, and Mohammad Reza Mousavi. 2022. Adaptive behavioral model learning for software product lines. In *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume A*. 142–153.

[35] Farbod Taymouri, Marcello La Rosa, Marlon Dumas, and Fabrizio Maria Maggi. 2021. Business process variant analysis: Survey and classification. *Knowledge-Based Systems* 211 (2021), 106557. https://doi.org/10.1016/j.knosys.2020.106557