

Validating Extended Feature Model Configurations using Petri Nets

Rakshit Mittal^{1,2}, Dominique Blouin², and Soumyadip Bandyopadhyay^{1,3}

¹ BITS Pilani, KK Birla Goa Campus, Goa, 403726 India

² Telecom Paris, Institut Polytechnique de Paris, 91120 France

³ Hasso-Platner Institute, Potsdam, 14482 Germany

Today’s systems are often highly configurable requiring to model them as product lines. For this, feature modeling has been developed to capture the aspects of commonality and variability of systems, which are then represented as configurations of feature models. Validating such configurations is therefore an important topic. Yet, existing approaches have several limitations such as scalability issues for large systems and system families and cannot handle the different feature modeling language variants or extensions that have been proposed.

We present our ongoing work on a novel approach to validate feature model configurations by translating them to Petri Nets. The approach can handle extended models of extended feature modeling languages and proposes an incremental validation scheme.

We detail the major steps of our approach using a simple example Feature Model depicted in Fig. 1(a). The example Feature Model has 8 features from A to H, with A as the root feature. Other than the feature-relationships that are apparent from the figure, there exists a cross-tree constraint ($B \implies C$) and a cardinality constraint $\langle 1, 3, A, D \rangle$. B, E are dead-features and D, H are false-optional.

Transformation: The transformation from Feature Model to Petri net has been developed to construct a constraint satisfaction problem from the constraints of the feature model. Features are mapped to unique places and feature-relations/constraints are mapped to unique transitions. The functions associated with the out-going edges (in yellow) are the key take-away from this section. These functions consist of a multiplication of two terms. The multiplier is a Boolean condition (within $[]$) type-cast to integer. The Boolean conditions, are designed so that “when a transition is fired, if the configuration does not satisfy the constraint represented by the transition, the values of the tokens in all the corresponding post-places are modified”⁴. The multiplicand has been designed to preserve the cardinality of the feature and corresponding token. In a configuration, for each feature, there is a token in the corresponding place with the same corresponding value as the cardinality in the configuration.

Validation: The verification of a feature configuration follows from the logic used in defining the functions associated with the output-arcs. To do so, all

Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

⁴ derivation methodology for the out-going function corresponding to each feature-relation and constraint available at <https://github.com/raks0009/FM2PN>

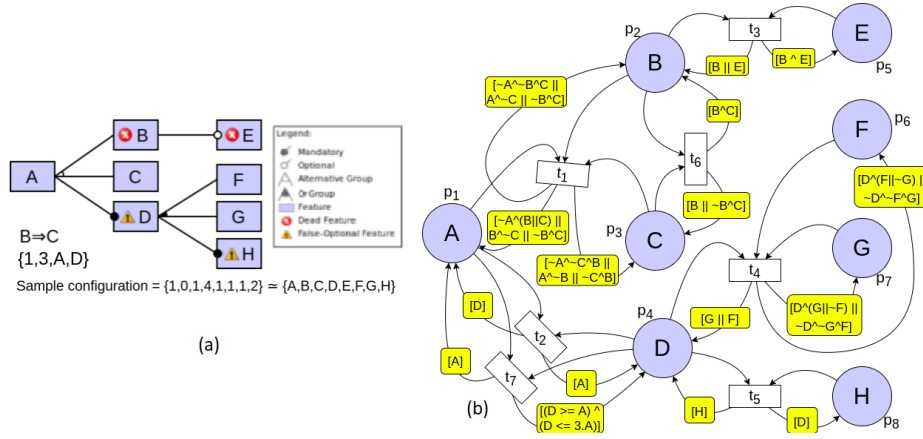


Fig. 1. The motivating example. Only the Boolean multipliers of the out-going functions are described in (b). Complete model available at <https://github.com/raks0009/FM2PN>

corresponding transitions (constraints) are fired (tested). If there is a change in the marking of the Petri net, the particular transition (firing which results in a change in the marking) is associated with a constraint that is not satisfied by the configuration. The transition is flagged for the user to rectify. If the values associated with all features in the configuration are compliant, the algorithm terminates without any flags, and the configuration is validated.

In the motivating example, the described approach tells the user that the firing of transitions t_3 and t_7 results in a change in the marking. Transition t_3 corresponds to the ‘AND-optional’ feature relation between B and E . In the configuration, E is selected but B is not, which is incorrect. Similarly, transition t_7 corresponds to the cardinality constraint between A and D which specifies that for every instance of A there can be minimum 1 and maximum 3 instances of D . A has the cardinality 1 and D has cardinality 4 in the configuration which does not follow this constraint, hence the transition is flagged.

Advantages and Limitations: The proposed approach can handle validation of feature configurations that adhere to extensions of feature models, something that is relatively unexplored in the literature [1]. The model transformation is also much more scalable than the Petri net based method described in [2]. Future work is aimed at formalizing, and implementing the approach. We are also working on a staged configuration approach for incremental validation.

References

- Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.* **35**(6), 615–636 (Sep 2010)
- Mennicke, S., Lochau, M., Schroeter, J., Winkelmann, T.: Automated verification of feature model configuration processes based on workflow petri nets. SPLC ’14, Association for Computing Machinery, New York, NY, USA (2014)