Incrementally Synthesizing Controllers from Scenario-Based Product Line Specifications  
(Extended Abstract)

Joel Greenyer\textsuperscript{1}, Christian Brenner\textsuperscript{2}, Maxime Cordy\textsuperscript{3}, Patrick Heymans\textsuperscript{3}, Erika Gressi\textsuperscript{4}
\textsuperscript{1}Software Engineering Group, Leibniz Universität Hannover, Germany  
greenyer@inf.uni-hannover.de  
\textsuperscript{2}Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn, Germany  
cbr@uni-paderborn.de  
\textsuperscript{3}PReCISE Research Center, University of Namur, Belgium  
\{mcr|phe\}@info.fundp.ac.be  
\textsuperscript{4}DEEPSE Group, DEIB, Politecnico di Milano, Italy  
erika.gressi@mail.polimi.it

Abstract: Many software-intensive systems consist of components that interact to fulfill complex functionality. Moreover, often many variants of such systems have to be designed at once. This adds complexity to the design task. Recently, we proposed a scenario-based approach to specify product lines, which combines feature diagrams and Modal Sequence Diagrams. We propose a new game-based technique for checking which product specifications are realizable and which ones are not, due to inconsistencies in the specification. In addition, our technique automatically synthesizes controllers for the realizable product specifications. To increase efficiency, we exploit the fact that many product variants are similar and synthesize product controllers incrementally. We provide a prototype tool and evaluate the efficiency of the approach.
A specification of allowed and forbidden behavior, however, can be unrealizable, i.e., it may be impossible to build a product that is able to react to all possible sequences of environment events in a way that its specification is satisfied. This can require costly iterations or in the end many desired products may be unrealizable or intrinsically flawed.

In this paper [GBC13], we propose a new realizability-checking technique. The technique is based on an on-the-fly game-solving algorithm for synthesizing controllers from the MSD specifications of each product in a PL. This algorithm will not only compute a yes/no answer; instead the synthesis algorithm will produce for each realizable product specification a controller that implements this specification. For unrealizable product specification, it will produce a counter-strategy that shows how the environment can always force the system to violate the specification. The algorithm operates on a game graph that is induced by the MSD specification and represents the possible interactions of the environment and the system. Being on-the-fly, the game-solving algorithm can often find a controller or counter-strategy before the complete game graph is explored. This avoids parts of the effort of constructing the game graph and exploring/backtracking.

We apply the synthesis incrementally and specifically optimized for PLs described by FDs. Typically many products in a PL are very similar, i.e., they share many common features and consequently share many common MSDs. Therefore, if a controller for one product could be successfully synthesized, we can more efficiently, based on this controller, synthesize a controller for a similar product. The efficiency gain can obtained because, by looking at a controller for a similar product, the on-the-fly synthesis algorithm can follow actions that led to a successful execution w.r.t. a similar specification. There exists a chance that these actions will lead to a behavior that satisfies also the MSD specification for the current product. This way, the synthesis algorithm can often more quickly find an admissible behavior and avoid building and exploring large parts of the game graph.

Our technique not only supports MSD specification that formalize requirements for the system; MSD specifications can also describe environment assumptions. We implemented this technique in SCENARIO TOOLS, an Eclipse/UML-based modeling, simulation, and synthesis tool suite for MSD specifications. Evaluations show that the incremental variant of the algorithm could outperform the non-incremental variant in many cases. Altogether, we pave the way for the intuitive and rigorous design of PLs of reactive systems.

In ongoing work, we are exploring an approach where, instead of synthesizing controllers in a sequential-incremental way, we perform the synthesis for all products at once [Gre14].

References
