A MODEL-DRIVEN FRAMEWORK FOR GUIDED DESIGN SPACE EXPLORATION[*]

Ábel HEGEDÜS
Advisor: Dániel VARRÓ

I. Introduction

Design space exploration (DSE) is a the analysis of several "functionally equivalent" implementation alternatives, which meet all design constraints in order to identify the most suitable design choice (solution) based on quality metrics such as cost or dependability. Design space exploration is a challenging problem in application areas (such as dependable embedded systems and IT system management), where model-driven engineering (MDE) techniques are already popular. DSE can be performed either during the design process to find optimal designs or during runtime to help dynamic reconfigurations.

In traditional DSE problems, the design constraints and quality metrics are numeric attributes to express cost, time or memory limits, etc. However, systems with modular software and hardware architectures (like AUTOSAR [2] in the automotive domain or large reconfigurable architectures) led to the introduction of complex restrictions on the graph-based model of the system.

Existing DSE approaches usually apply model checking complemented with exhaustive state space exploration or solve finite domain constraint satisfaction problems that cannot effectively handle structural constraints and dynamic manipulation of elements. In order to alleviate these issues, designers often provide additional information (hints) about the system (e.g. from earlier experience or by some analysis) that can reduce the design space to a more feasible size.

Guided model-driven design space exploration aims to explore alternative system designs efficiently by making use of advanced model-driven techniques (e.g. incremental model transformations) and hints (obtained by analysis tools or provided by the designer). These hints are interpreted during the exploration to continue along promising search paths (using selection criteria) and to avoid the traversal of unpromising designs (by cut-off criteria). Additionally, the use of incremental techniques leads to exploration strategies that are able to find additional (alternative) solutions.

In this paper, a model-driven framework for guided design space exploration is described, where the system states are graphs, operations are defined as graph transformation rules, while goals and constraints are defined as graph patterns.

II. Overview of the Approach

The schematic overview of the framework for guided design space exploration is illustrated in Figure 1. First, the design problem description specifies the domain where the exploration takes place to produce solutions. It includes: (1) the initial state of the system at the start of the exploration, (2) the set of manipulation operations (called labeling or exploration rules) defined on the system, (3) goals described as structural or numerical constraints, which need to be satisfied by solution states found by the exploration, and (4) global constraints, which are satisfied by the initial and solution states and all intermediate states on the trajectory between them.

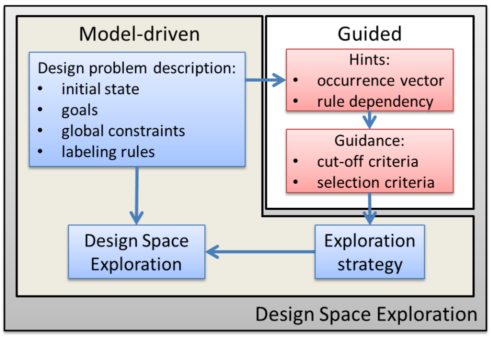

Figure 1: Model-driven Guided DSE

[*]This summary is based on the ASE'11 paper with the same title [1]
This work was partially supported by the grant TÁMOP (4.2.2.B-10/1–2010-0009).

The design space exploration performs the search for solutions by exploring the design (or state) space of the problem description. It starts from the initial state and traverses reachable states by applying the operations on the system. In order to find a solution quickly, exploration is often aided by an exploration strategy. A simple strategy (as proposed in [3]) may use random selection in a depth first search or statically assign priority levels to operations. However, a more advanced strategy should also determine whether a given state will never lead to a valid solution (i.e. it is a dead end) and states reachable from it should not be traversed. In a guided approach, the exploration strategy relies on guidance, which uses hints for driving the traversal and identifying dead ends.

Hints are information originating from the designer or (as in [4]) from some automated analysis carried out using formal methods that often abstract the design problem description. For example, the result of such analysis can be information regarding the number of operation applications (called occurrence vector). The guided approach uses occurrence vectors and dependency relations between rules, computed from pre- and postconditions, as hints (see [1]).

Finally, the guidance calculates and interprets hints and provides decision support for the exploration strategy (see details in [1]). In this approach, guidance is defined as the evaluation of cut-off and selection criteria based on the current state and the hints (as defined in [4]). Cut-off criteria identify dead end states and bound the exploration, while selection criteria prioritize available rules in a state by their likelihood of leading to a final (solution) state.

III. Exploration Strategy

Guided exploration strategies can be categorized by the used hints and guidance. Here, two guided strategies are presented, the first uses occurrence vectors only as hints (occurrence), while the other uses rule dependency as well (full guidance). Note that the full guidance strategy uses rule priorities only if two labeling rules were evaluated as equal by the guidance. These strategies are compared to the fixed priority strategy.

Figure 2 illustrates the design space exploration for these techniques on a simple example. The circles denote the traversed states which are numbered according to the traversal order, while the applicable rules are listed beside them.

Downward arrows illustrate rule applications, while dotted arrows represent backtracking from invalid or cut-off states. The same rule can be applied multiple times at a given state if more than one applicable match is found in the graph (see state 2 on the right side). The exploration terminates when an optimal solution is found. A solution is optimal if the path leading to it contains the least number of rule applications (i.e. it is the shortest trajectory to a solution model).

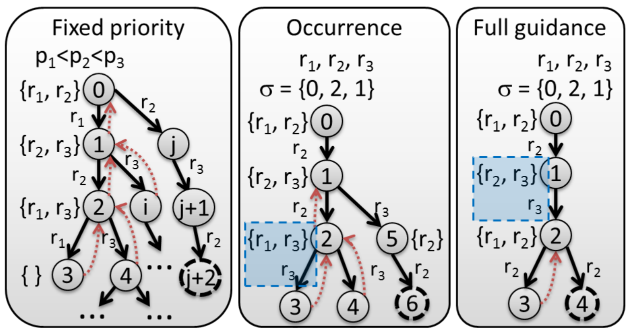

Figure 2: Comparison of exploration strategies

In the case of the fixed priority strategy, the next applied operation is the one with the highest priority among the applicable ones. As the depth-first technique is used in the fixed priority exploration strategy, the first solution found by that strategy is often several times longer than the optimal, suboptimal solutions are used iteratively as depth limits to force the exploration to find shorter solutions.

The *occurrence* strategy applies operations based on the occurrence vector provided by the system analysis. Figure 2 shows that the hint states the r_2 should be applied twice and r_3 once. Therefore, r_1 is not applied in state 0 or 2 (highlighted) in order to be compliant to the occurrence vector.

The full guidance exploration strategy (illustrated in the right side of Figure 2) takes the dependency relations between rules into account in addition to the occurrence vector. Therefore, in state 1 (highlighted) it selects r_3 for the next application. Rule r_2 is applicable on two matches in state 2, the first leading to a dead-end state, while the second application leads to a solution in state 4. Note that the selection in state 1 leads to a reduced traversed design space compared to the occurrence strategy.

IV. Implementation Details

Figure 3 gives an overview of the implemented guided design space exploration framework. The implementation uses the VIATRA2 model transformation framework [5], which provides metamodeling capabilities and supports model transformations based on the concepts of graph transformations and abstract state machines. Its incremental pattern matcher is used as a powerful query engine.

The design space exploration is performed by the constraint satisfaction engine, CSP(M), presented in [3], where rules, goals and constraints (specified using graph transformation rules and patterns) are used in solving constraint satisfaction problems over the input model (both included in the design problem description).

The abstraction of graph transformation rules into Petri nets (PN) and Integer Linear Programming (ILP) problems are also automated. The industry leading IBM CPLEX [6] optimization tool is used, which supports the calculation of alternate solutions (occurrence vectors used for initializing the dependency graph G_d). The edges of G_d are computed from the transformation rules using the Condor [7] dependency analyzer tool, while the graph itself is built and stored as an Eclipse Modeling Framework (EMF) instance model. The criteria definitions and the criteria evaluation algorithm (guidance) are implemented in Java as separate components, and are connected to the guided design space exploration strategy.

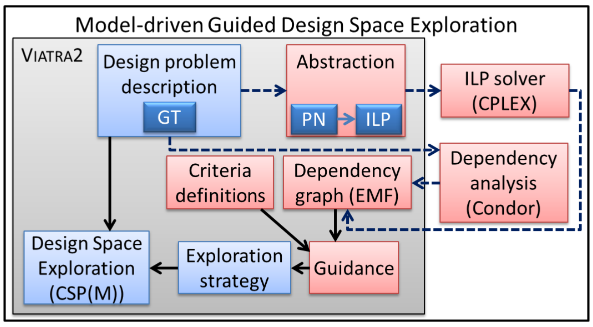
Model-driven Guided Design Space Exploration
VIATRA2
Design problem description
GT
Abstraction
PN
ILP
ILP solver (CPLEX)
Criteria definitions
Dependency graph (EMF)
Dependency analysis (Condor)
Design Space Exploration (CSP(M))
Exploration strategy
Guidance

Figure 3: Overview of the guided DSE framework

V. Evaluation of the Approach

The aim of the evaluation was to demonstrate that the full guidance strategy is more efficient than the other strategies (namely, fixed priority and occurrence strategy, which were used for previous measurements in [3]) as it traverses considerably fewer states and does not introduce significant overhead, thus provides better runtime in the other approaches for most of the experiments.

Cases used in the Evaluation

For evaluation, the cloud case study presented in [1] and a service configuration case study (presented in [8]) were used. These cases are relevant in the context of model-driven DSE as they represent both design and runtime exploration problems and make comparison with previous results [8, 3] possible.

Both case studies included multiple cases (see Figure 4). PowerOn cases deal with empty initial models, while Reconfigure (RC) cases deal with existing models which must be modified to satisfy goals. Finally, the Clustered DB case requires databases to be deployed on clusters.

Evaluation Environment and Method

The evaluation was carried out 5 times for each test case and strategy in the following way:

(1) the initial model is loaded into VIATRA2, (2) the goals, constraints and operations are added to the framework, (3) the exploration component is initialized and runtime measurement is started (using wall time with OS-level nanotime precision). Next, (4) the design space exploration framework computes solutions. Finally, (5) the runtime measurement is stopped and the results are saved. The exploration is limited to 1 million visited states.

Evaluation of Results

The graph in Figure 4 shows the results of measurements using the case study models. For each case, the length of the shortest solution trajectory (the number of applied rules) is given in parenthesis with the average number of visited states during the design space exploration illustrated for each strategy.

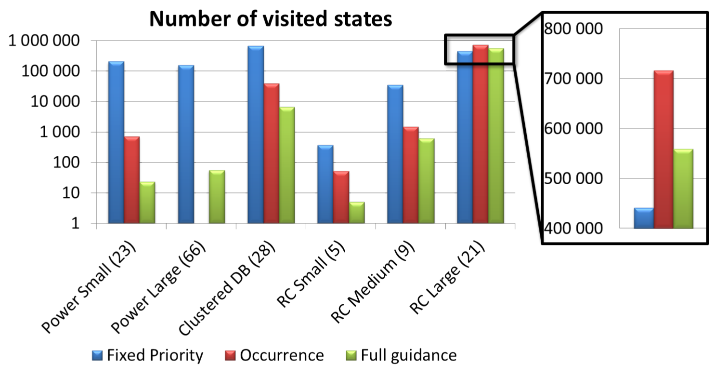

Figure 4: Results for exploration

The following observations were made based on the results from the experiments: (1) The combined use of occurrence vectors and rule dependency for cut-off and selection criteria based guidance outperforms our previously published strategies; (2) The added computation required for criteria evaluation does not significantly increase runtime; and (3) The under-approximation of the occurrence vector based analysis ensures that guided strategies always find optimal solutions first (similarly to the A* algorithm). Note that in the last experiment the fixed priority strategy visits less states than the guided strategies because of a high number of infeasible occurrence vectors.

VI. Conclusion and Future Work

Guided DSE exploration uses hints to reduce the number of states traversed when searching for solutions. Hints are used (i) to identify dead-end states (cut-off criteria) and (ii) to order applicable rules in a given state (selection criteria). In this paper, I summarized the results of developing a model-driven framework for guided DSE, which uses rule dependency and occurrence vectors as hints for the exploration strategy. Evaluation of the exploration strategies using a cloud configuration problem showed that the criteria-driven approach reduces the design space further thus increasing efficiency. The framework was also successfully applied for generating quick fixes for domain specific modeling languages [9], evaluation with BPMN business process models showed that it can work as a good design time development assistance tool.

Future work aims to improve the framework by introducing incremental techniques for identifying states, handling guidance and space exploration. There are also plans for extending the framework to handle EMF models.

References

[1] Á. Hegedüs, Á. Horváth, I. Ráth, and D. Varró, “A model-driven framework for guided design space exploration,” in *26th IEEE/ACM Intl. Conf. on Automated Software Engineering (ASE 2011)*. IEEE Computer Society, 11/2011 2011.

[2] AUTOSAR Consortium, *The AUTOSAR Standard.*, `http://www.autosar.org/`.

[3] Á. Horváth and D. Varró, “Dynamic constraint satisfaction problems over models,” *Software and Systems Modeling*, 2011, 10.1007/s10270-010-0185-5.

[4] Á. Hegedüs, Á. Horváth, and D. Varró, “Towards guided trajectory exploration of graph transformation systems,” *ECEASST*, 40, 2010, Petri Nets and Graph Transformation 2010.

[5] A. Balogh and D. Varró, “Advanced model transformation language constructs in the VIATRA2 framework,” in *ACM Symp. on Applied Computing (SAC 2006)*, p. 1280–1287, Dijon, France, 2006. ACM Press.

[6] IBM, *ILOG CPLEX Optimizer*, `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`.

[7] ROOTS, *Condor*, `http://roots.iai.uni-bonn.de/research/condor/`.

[8] S. Varró-Gyapay and D. Varró, “Optimization in Graph Transformation Systems Using Petri Net Based Techniques,” *ECEASST*, 2, 2006, Petri Nets and Graph Transformation 2006.

[9] Á. Hegedüs, Á. Horváth, I. Ráth, M. C. Branco, and D. Varró, “Quick fix generation for dsmls,” in *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2011*. IEEE Computer Society, 09/2011 2011.