MŰEGYETEM 1782

Budapest University of Technology and Economics
Department of Measurement and Information Systems

Trajectory Set Approximation for Optimization and Verification of IT Systems

PhD Thesis Booklet

Varró-Gyapay Szilvia

Advisor:
Prof. András Pataricza
DSc

Budapest, September, 2014.

1 Preliminaries and Objectives

1.1 Critical Systems Design

With the continuous increase of IT systems both in size and complexity, their quality of service becomes an issue in different application fields. For business critical systems that control the infrastructure of an international company with world–wide locations a service failure may cause serious financial losses. For safety critical systems like automotive or avionics a failure in the delivery of a service may result in critical damages or even casualties.

Certification standards like DO-178C (for avionics) or ISO 26262 (for automotive) prescribe rigorous safety requirements on the entire design process of a critical system including requirements specification, analysis, design verification and validation (V&V), and documentation, in order to assure safe operation throughout the entire lifecycle of such a system. Due to this rigour, the certification costs of a new safety-critical product are horrendously high nowadays. Recent surveys demonstrated that verification and validation costs may contribute as much as 70-80% of the total certification costs. Finding innovative V&V techniques which simultaneously improves the quality of the system while reducing certification costs is a major challenge.

A service delivery compliant with certification standards needs to meet different kind of requirements. A functionally correct service calculates its output for any input exactly as prescribed by its specification (i.e. without side effects). Furthermore, service delivery needs to be completed within certain deadlines with high reliability and availability. Finally, if there are multiple design variants for delivering a service, one needs to select the cheapest option or the one which guarantees the shortest time to market. In this respect, critical systems design distinguishes between functional, extra-functional and optimality requirements.

1.1.1 Model driven engineering of critical systems

Model-driven engineering (MDE) has become a key technique for critical systems design with rapidly increasing number of applications, for instance, in avionics or automotive systems [42, 43]. MDE facilitates the systematic use of models from an early phase of the design cycle. System requirements and design are captured by high-level, visual engineering models. Early systematic formal analysis of design models can be carried out by generating various mathematical models by automated model transformations (MTs) to eliminate conceptual flaws [29]. The source code of the target system is derived by code generator transformations [34, 50, 51].

MDE promises to simultaneously increase productivity and quality while reducing development costs. Early detection of design flaws saves costly re-design (compared to detecting the same problem by traditional testing). Automated (and qualified) code generators improve productivity by synthesizing provenly correct source code of a safety-critical application from a validated model. In addition to source code generation, MDE

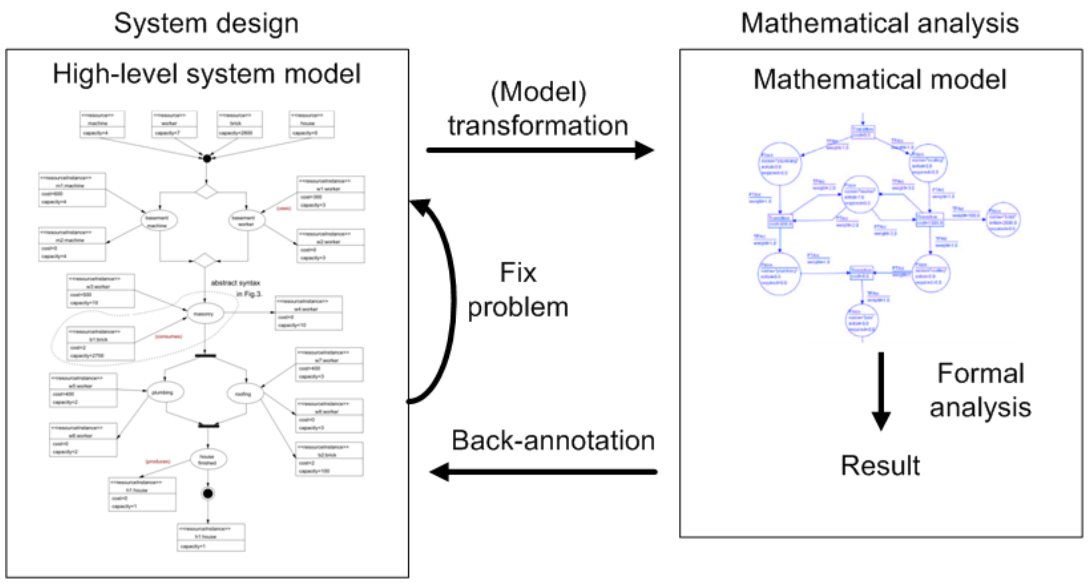

Figure 1: Model driven analysis of critical systems

tools could help synthesize other design artifacts as well (such as configuration tables, fault trees, test cases, documentation) and provide fine-grained support for traceability [26].

1.1.2 Formal methods

Formal methods (such as statecharts, Petri nets, dataflow nets, process algebras, or timed automaton) offer formal models with precisely defined semantics and mathematical algorithms to exhaustively cover all potential executions. They are primary means to carry out early analysis in model-driven engineering by analyzing system models having with mathematical scrutiny.

While traditional testing primarily aims at highlighting flaws in the system (i.e. the presence of faults), in many cases, one can prove the correctness of the system (i.e. the absence of errors) by using formal methods. In this respect, formal methods are accepted verification and validation techniques in most safety-critical certification standards. In real development scenarios, a combined use of traditional testing and formal methods are widely used nowadays, especially, in the avionics domain.

Semi–formal techniques like the Unified Modeling Language (UML) [35], the System Modeling Language (SysML) [23] or Business Process Modeling Notation (BPMN) [24] provide an easy-to-understand and comprehensive language for system engineers with precisely defined structure to model the system at a high-level of abstraction. Using a standard modeling language improves communication between different stakeholders and development teams. However, these languages can be ambiguous due to the lack of formal semantics, therefore, one cannot establish a safety argument directly at this level. Furthermore, certain requirements (such as temporal behavior in UML) cannot easily be captured in these semi-formal languages. In contrast, formal methods always build on precise requirements and mathematical models with formal semantics to specify a system model, which enables formal verification of the model wrt. the requirements (properties). While these formal methods are rigorous and does not allow ambiguity in specifications, they are much more difficult to understand and their use needs a strong mathematical background from the designer. Model-driven engineering closes the gap between semi–formal and formal methods by enabling the precise formal verification of high-level system models while still offering visual notations to system engineers.

Studies [25] have demonstrated that the combined use of formal methods and model driven engineering simultaneously improves quality and reduces V&V costs by one order of magnitude (compared to traditional V&V without formal methods).

Model checking is widely used in the validation and verification process. Model checking means the automatic check of the fulfillment of some properties [27] by an exhaustive traversal of the state space. The aim of model checking is to either prove that a given property is satisfied in a system model or to constructively retrieve a counterexample leading to a violation. The input model of a model checker is a *finite labelled transition system* that captures the behaviour of the underlying system by a directed graph where the nodes are the states of the system and the edges represent the state transitions. The properties that has to be satisfied by the system are formalized as *temporal logical expressions* that are composed of temporal logical operators and state variables. A(n explicit state) model checker (like SPIN [38]) translates a property into an automaton that accepts only those paths where the property is evaluated to true. Then the product automaton of the negation of this property automaton and the automaton of the transition system is analyzed. If there is no execution sequence in the product automaton leading to an accept state (i.e. the product automata accepts the empty language) then there is no sequence in the transition system that satisfies the negation of the property. In other words the original property holds in the system. Otherwise, the corresponding sequence is retrieved as a counterexample.

1.2 Optimization Techniques

In the current thesis, we focus on the research challenge when the system model has to be optimized and verified *simultaneously* according to the system requirements in order to assure an efficient *and* correct operation. For instance, the delivery of a cost– or time–optimal solution is a core problem in business processes or production systems in order to satisfy the prescribed quantitative requirements. The application of *both* formal analysis and optimization can assure that the system will function properly and also fulfills its quantitative constraints in an optimal way (with respect to some objectives).

Linear programming (shortly LP) methods are known as efficient solution techniques for optimization problems [52]. LP problems consist of some variables, linear constraints over these variables and an objective function, which is a linear combination of the variables that has to be either minimized or maximized respecting the constraints.

An LP problem is called integer linear programming (ILP) problem if all variables are constrained to be integers while an LP problem is called mixed integer linear programming (MILP) problem if some (but not necessarily all) variables are forced to be integer.

There were developed several techniques to solve LP problems such as simplex method [52] or interior point methods [39]. A well–known method to solve MILP problems is the so–called *Branch and Bound* (shortly B&B) method [52]. The idea of the B&B algorithm is to apply the steps *branching and bounding* after each other to the current (sub)problem in order to separate and restrict the solution space to subproblems .

Bounding and relaxation The solution of integer problems is much more complex and time consuming than the solution of continuous problems (with arbitrary real values for variables). The idea of relaxation is to use techniques defined for continuous problems to bound the objective value of the integer problem. Usually interior point methods (like the algorithm in [39]) are used to solve LP problems.

A lower bound for the minimum of an optimization problem is derived by the solution of the LP problem that is the same as the ILP problem except that the variables are *relaxed* to continuous ones. On the one hand, if there is no solution for the relaxed LP problem there is also no solution for the integer problem. On the other hand, the objective value of an optimal solution of the relaxed problem is a lower bound (in case of minimization) for all optimal integer solutions since all of those are included in the solution space of the relaxed problem. This way the search space is restricted by the approximation for the objective value of the optimal solution.

There are several engineering problems that can be mapped to (MI)LP problems like process network synthesis in manufacturing, transportation, chemical engineering or VLSI design.

1.3 The Optimal Trajectory Problem: Simultaneous Optimization and Verification

Verification and validation methods aim to assure the correctness of a system model, while optimization may serve to calculate the quantitative performance characteristics of a system by estimating its quantitative boundaries and to minimize operation costs. However, it is a challenging question how to combine the best practices of the two fields.

Combined optimization and V&V problems can frequently be expressed as a reachability problem with quantitative or qualitative measurements, which is called an *optimal trajectory problem*: find an optimal path (if such exists) starting from an initial state to a target state that *satisfies all the given requirements*. The traditional reachability problem is to answer the question whether a given state is reachable from an initial state in the system. However the optimal trajectory problem requires also the generation of a path leading from the initial state to the target state. A target state may be concrete, i.e. all state variables have a desired value in the target state or partial if we focus on the desired value of only a subset of variables.

Similar problems are investigated in planning and scheduling algorithms [30, 44, 49], various strategies in game theory [31], design space exploration in system engineering [28].

1.4 Objectives of the Thesis

The objective of the research presented in this thesis is to propose (i) mathematical models in which the *optimal trajectory problem* can be formally expressed and (ii) strategies and algorithms to solve the optimal trajectory problem by combining verification and optimization techniques.

2 Simultaneous Verification and Optimization of System Models

The combination of efficient verification and optimization techniques requires a framework that is able to model both the crucial aspects and the metrics of the underlying system. Due to their modeling expressiveness and rich mathematical background I use an extension of Petri nets (PNs) and graph transformation systems (GTSs) to model optimization and validation problems.

An optimal trajectory of a Petri net or a graph transformation system is a path in the state space that is optimal with respect to a given metric and satisfies the given (logical) requirements. The main challenge of finding an optimal path is that the state space may be infinite which cannot directly be traversed exhaustively. However, the search space can be reduced to be finite by filtering certain paths or taking into account upper bounds for the metrics.

2.1 Petri Nets

The current thesis focuses on Petri nets, which is a well-known and widely used formal method introduced by Carl Adam Petri in 1962 [47] for the modeling the communication of processes. Petri nets provide an intuitive graphical yet mathematically expressive language to capture nondeterminism and concurrency in system models.

Petri nets are directed bipartite graphs with nodes places and transitions where places represent the state variables of the system and may contain tokens while transitions stand for the tasks or activities in the system describing the system control. The system dynamics is modeled by the token transfer between places by firing transitions: a transition is enabled and it may fire if there are enough tokens on its input places. During its firing the transition removes these tokens and produces as many tokens to its output places as required. The number of removed and produced tokens are defined by a weight function interpreted on the edges of the graph. The potential behavior of the system is identified by trajectories in the form of transition firing sequences in the system state space.

Several extensions of the core Petri net formalism exist to precisely capture time, cost, dependability or other properties of systems. The thesis focuses on the optimal trajectory problem of Petri nets with cost and duration. Cost parameters are assigned to the transitions of the Petri net to denote the cost of the execution of a transition. Then the optimal trajectory problem can be formulated as follows.

2.1.1 Cost-optimal trajectory of Petri nets

Let a Petri net with cost parameters, a partial target marking and a cost limit be given. Then the cost–optimal trajectory problem is to find a trajectory s starting from the initial marking such that (i) it covers the partial target marking, (ii) the cost of all other trajectory that starts from the initial marking and covers the partial target marking is not smaller than the cost of s and (iii) the cost of the trajectory s does not exceed the cost limit.

The token change in the Petri net can be described by the so–called state equation. The state equation is based on the incidence matrix of the Petri net that describes the token change in the Petri net. In order to restrict the search space of the Petri net an integer linear programming problem (ILP) is formulated as an abstraction of the optimal trajectory problem. This abstraction gives an overapproximation for the solution of the optimal trajectory problem since (i) each trajectory in the Petri net has a corresponding solution of the state inequality but (ii) it may exist a solution of the state inequality such that there exists no corresponding executable trajectory in the Petri net.

Such an ILP problem can be solved by traditional methods like the Branch and Bound technique. However, their use focuses only on the ILP problem and ignores the structure of the Petri net. In order to incorporate the structure of the Petri net into the ILP problem Process Network Synthesis algorithms are customized to the Petri net optimal trajectory problem.

2.1.2 Process Network Synthesis

Process Network Synthesis (PNS) solutions were elaborated by Friedler et. al. [36, 37] to deliver an optimal solution for production of some materials in chemical engineering. A process network is represented by a Process-graph (P-graph). A P-graph describes a production problem that aims at producing some products from given raw materials using operating units. Operating units transform their input materials into their output materials.

Friedler et. al. proposed efficient algorithms that exploit the combinatorial properties of the problems. They showed that the P-graph structure of a solution has to conform to five axioms. Such a P–graph is called a combinatorially feasible solution structure that forms a *recipe* describing what materials (ingredients) and operating units (tools) are necessary to “cook” the desired products. The Accelerated Branch and Bound algorithm delivers an optimal solution for the PNS problem using the traditional Branch and Bound algorithm *over* the set of these recipes.

Petri nets and P-graphs have similar graphics: both structures are designated as bipartite graphs with nodes places and transitions, and materials and operating units, respectively. In addition, transitions have the same task as operating units: they transform their input objects into their output objects. Also the aim of the Petri net optimal trajectory problem and PNS problems resembles: how to reach a given state starting from an initial state in an optimal way.

However, the set of solutions for a Petri net optimal trajectory problem is a subset of the set of solutions for the PNS problem. One reason is that the Petri net definition is restricted to integer token flow and integer firing number of transitions. The other reason is that a solution in the Petri net has to be executable from the given initial state while a solution for the PNS problem is required to be executable starting from a stationary state. This way the corresponding Petri net optimal trajectory problem is a restriction of the PNS problem such that each solution for the Petri net optimal trajectory problem is also a solution for the PNS problem.

I customize PNS algorithms in my thesis to deliver a candidate solution for the Petri net optimal trajectory problem. The candidate solution is delivered by an ILP solver that gives a solution for the abstraction ILP problem of the Petri net using PNS algorithms. The candidate solution is a transition occurrence vector or Parikh vector that counts the number of firing of the individual transitions in a corresponding trajectory (if it exists). If there exists no solution for the ILP problem then there is no solution for the optimal trajectory problem. However, if there exists an optimal solution Parikh vector for the ILP problem it may happen that it does not have a corresponding executable (fireable) trajectory in the Petri net.

The executability of the candidate Parikh vector from the given initial marking is checked afterwards in two steps. Fig. 2 shows the workflow of the approach with the *Reachability check*, the *Fireability check* and the trajectory generation.

Reachability check The first subsequent check is performed using the reachability function of Petri nets to check the reachability of a marking compliant with the candidate solution Parikh vector. The reachability function encodes the (finite) transitive closure of the one-step transition function in the form of *Binary Decision Diagrams (BDDs)*. The function is parameterized by the initial and the target marking, i.e. the function does not have to generated again from scratch if only these parameters are changed. This approach provides a reusable filtering of candidate solutions.

Fireability check and trajectory retrieval If the marking compliant with the candidate vector is proven to be reachable, its fireability (executability) still needs to be checked. I propose to use the SPIN model checker tool for fireability checks, completing additional verification tasks and the derivation of an optimal trajectory.

If either the reachability check or the fireability check fails the next best solution vector of the ILP problem is generated and it is checked again until a solution is found or all branches are pruned and there is no solution.

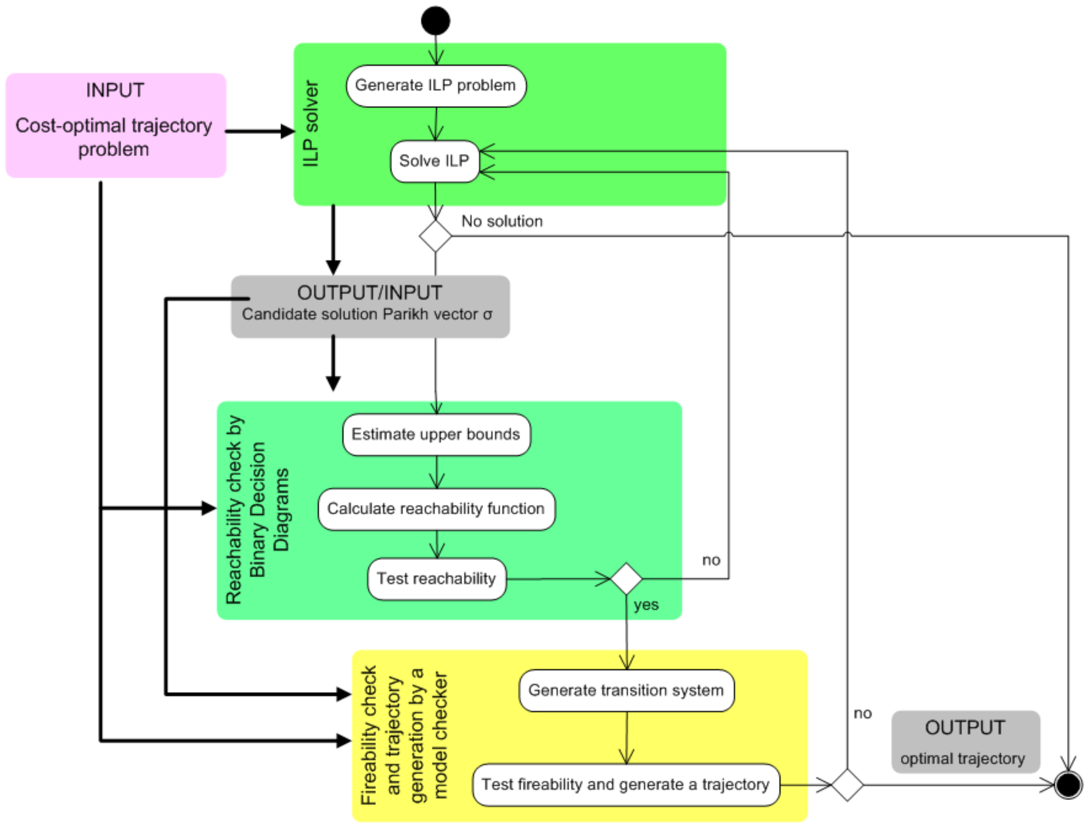
INPUT
Cost-optimal trajectory problem
ILP solver
Generate ILP problem
Solve ILP
No solution
OUTPUT/INPUT
Candidate solution Parikh vector σ
Reachability check by Binary Decision Diagrams
Estimate upper bounds
Calculate reachability function
Test reachability
no
yes
Fireability check and trajectory generation by a model checker
Generate transition system
Test fireability and generate a trajectory
no
OUTPUT
optimal trajectory

Figure 2: Workflow of the approach

2.1.3 Direct encoding approach

I give a second algorithm for the cost–optimal trajectory problem that directly encodes the optimal trajectory problem of bounded Petri nets into the SPIN model checker with on-the-fly optimization during verification. In this technique, the numerical criteria of the Branch and Bound algorithm is also embedded into the linear temporal logic formula of the verification condition, thus state space exploration simultaneously checks both the logical and numerical conditions.

2.1.4 Time-optimal trajectory of Petri nets

Similarly to the cost-optimal case, I give two solutions for the time-optimal trajectory problem. The first one calculates an upper bound for the maximal duration of an optimal Petri net trajectory and delivers a fireable trajectory as a solution for the ILP model that (1) represents the token change in the places in any time instant within this upper bound and (2) encodes the enabledness condition of the transitions in each time instant.

The second solution is a direct method for the time-optimal trajectory problem that uses the same upper bound to constrain the verification run of the SPIN model checker in order to deliver an optimal trajectory on-the-fly by encoding the optimality criteria into an LTL expression.

2.2 Graph Transformation Systems

Graph transformation [48] provides another mathematical model where the system is modeled by typed and attributed graph models while its behaviour is represented by rule and pattern based manipulation. The application of a rule, also known as a rewriting step, consists of the matching of an instance graph against the left hand side graph pattern of the rule and there placing of that part in accordance with the right hand side pattern.

Due to their expressiveness, graph transformation systems are widely used in compiler construction, database management, software engineering, visual modeling, logic programming and in many other fields [33]. The graph based representation and the use of types, labels and attributes facilitate an easy-to-understand graphical modeling of complex systems. Furthermore, the formal analysis of graph transformation systems has a strong theoretical background that enables the check of typical properties in graph transformation systems.

2.2.1 Optimal trajectory problem in graph transformation systems

Since there is a strong correspondence between Petri nets and graph transformation systems (shortly GTS) [32, 40, 41], I generalized the optimal trajectory problems of Petri nets over graph transformation systems with cost and time. I introduced both cost and time into graph transformation systems as the cost and duration of a rule application and I defined formal semantics for evaluating graph transformation rules with cost and time.

The optimal trajectory problem in GTSs is solved using the Petri net based solutions: the GTS is transformed into a Petri net such that all the valid trajectories of the GTS are simulated by a corresponding Petri net trajectory. However, it is not a bisimulation, i.e. trajectories may exist in the Petri net that have no compliant trajectory in the GTS. Therefore the executability of the Petri net solution has to be checked in the original GTS after back-annotation.

3 Main Results

The solution of the so–called optimal trajectory problem that is introduced in the thesis generates an optimal trajectory for *general* Petri nets. This solution is novel to previous ideas such that

- it can be used for general Petri nets and not only for a specific subclass of Petri nets.
- It delivers an optimal trajectory and not a suboptimal one that increases the design quality of the system.
- The sequence of semi–decision methods provides reusable techniques in case for the further investigation of the same system. The introduced methods are reusable in the sense that the underlying mathematical models remain the same or are changed slightly in the next iteration of the search. In other words, the search is continued and is not started again from scratch.

The Petri net based optimal trajectory problem also can be used in the analysis of graph transformation systems (GTS) with quantitative metrics: the graph transformation system representing the evolution of a system is transformed into a Petri net model. Then an optimal solution of the Petri net is generated, and is back–annotated into the original GTS.

Figure 3 shows the novel results in the thesis where C1-C4 denote the number of the corresponding contribution.

3.1 Petri Nets and Process Network Synthesis Problems

Contribution 1. *I analyzed the similarities and differences between Petri nets and PNS problems in order to characterize the application conditions of PNS algorithms for optimal trajectory problems over Petri nets [4, 7–9].*

1/1 *I defined a structural mapping from any Petri net to a P–graph representation. I defined a mapping, which derives a PNS problem from Petri net optimal trajectory problem.*

1/2 *I provided a structurally valid Petri net interpretation of the five PNS axioms that characterize the combinatorially feasible networks of a PNS problem.*

1/3 *I elaborated an algorithm to reduce the size of the PNS problem by adapting the Petri net reduction rules. I proved that the application of the reduction algorithm to the PNS problem preserves the reachability property and the cost of the production.*

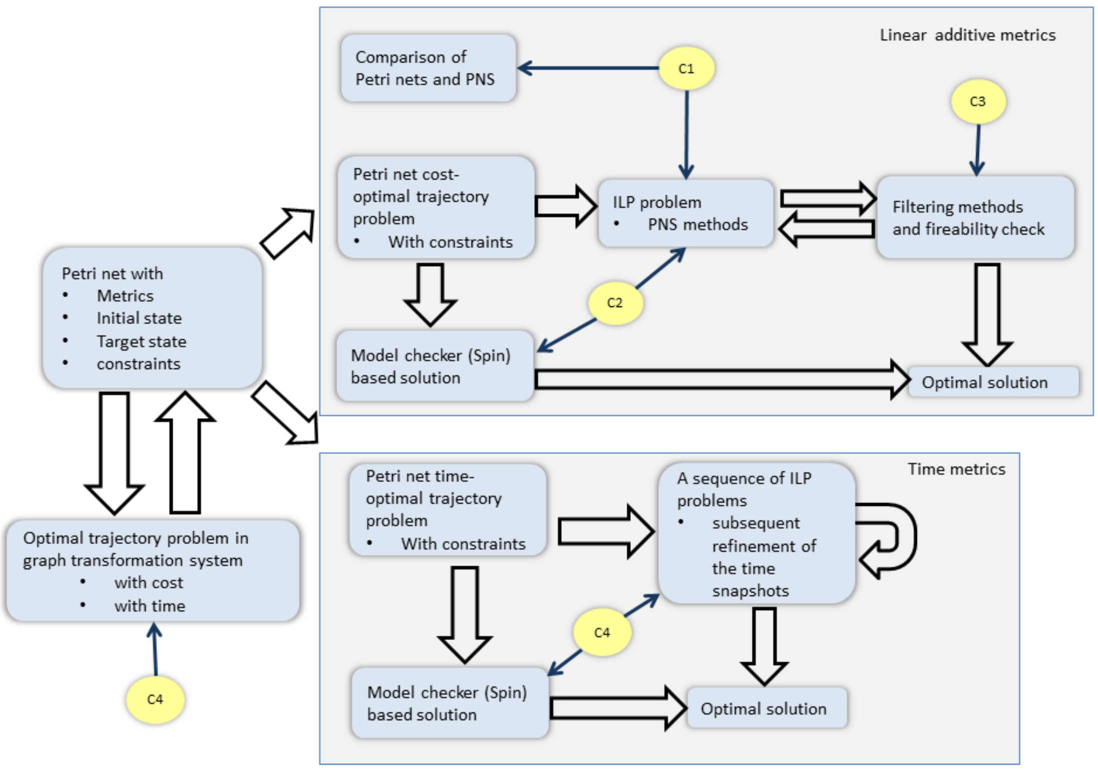
Linear additive metrics
Comparison of Petri nets and PNS
C1
C3
Petri net cost-optimal trajectory problem
• With constraints
ILP problem
• PNS methods
Filtering methods and fireability check
C2
Model checker (Spin) based solution
Optimal solution
Petri net with
• Metrics
• Initial state
• Target state
• constraints
Time metrics
Petri net time-optimal trajectory problem
• With constraints
A sequence of ILP problems
• subsequent refinement of the time snapshots
Optimal trajectory problem in graph transformation system
• with cost
• with time
C4
Model checker (Spin) based solution
Optimal solution
C4

Figure 3: Overview of the contributions

3.2 Customization of PNS Algorithms for the Petri Net Cost–optimal Trajectory Problem

Contribution 2. *I adapted various PNS algorithms to derive candidate Parikh vectors for the optimal trajectory problem over Petri nets.*

2/1 *I customized the PNS MSG algorithm to generate the maximal structure of a Petri net optimal trajectory problem. I interpreted the PNS SSG algorithm for Petri nets to generate the structurally valid solutions of a Petri net optimal trajectory problem. The corresponding publications are [4, 12, 13, 15–17].*

2/2 *I elaborated an algorithm which takes the set of all structurally valid solutions of a Petri net as input and calculates a candidate for an optimal Parikh vector as output. I elaborated another algorithm that generates the same output from a minimal basis of all structurally valid solutions of a Petri net as input. The corresponding publication is [4].*

2/3 *By adapting the PNS ABB algorithm to the Petri net optimal trajectory problem, I elaborated an algorithm which takes the maximal structure of a Petri net as input and calculates a candidate for an optimal Parikh vector as output by continuously checking the structural validity of the candidates. The corresponding publications are [4, 12, 13, 15–17].*

3.3 Reachability and Fireability Check of the Candidate Parikh Vector

Contribution 3. *I proposed different state space exploration and reduction techniques for Petri nets to deliver fireable trajectories relying on various upper bounds (such as candidate Parikh vectors or explicit limits on exploration depth) on the structure of the state space used as exploration hints.*

3/1 *I proposed the use of reachability function of Petri nets to check the reachability of a marking compliant with a candidate optimal Parikh vector. Reachability function encodes the (finite) transitive closure of the one-step transition function parametrized by the initial state in the form of* Binary Decision Diagrams (BDDs)*, thus providing a fast and reusable filtering of candidate Parikh vectors. The results of this thesis were published in [4, 12–16].*

3/2 *I elaborated an algorithm using the* SPIN *model checker tool for the generation of a fireable trajectory from an initial marking that is compliant with (1) a candidate optimal Parikh vector, (2) the reachability function of the Petri net and (3) additional temporal logic expressions. The results of this thesis were published in [4, 12–14, 16, 19].*

3/3 *I proposed a direct encoding of the optimal trajectory problem of bounded Petri nets into the* SPIN *model checker with on-the-fly optimization during verification. In this technique, the numerical criteria of the Branch and Bound algorithm is also embedded into the linear temporal logic formula of the verification condition, thus state space exploration simultaneously checks both the logical and numerical conditions. The corresponding publications are [3, 4].*

3.4 Time-Optimal Trajectory Problem

Contribution 4. *I proposed two solutions for the time-optimal trajectory problem of Petri nets relying on the upper bound taken from the cost-optimal ILP problem and the state equations evaluated at dedicated time instants.*

4/1 *I proposed an algorithm for the time-optimal trajectory problem that calculates an upper bound for the maximal duration of an optimal Petri net trajectory and delivers a fireable trajectory as a solution for the ILP model that (1) represents the token change in the places in any time instant within this upper bound and (2) encodes the enabledness condition of the transitions in each time instant.*

4/2 *I proposed a direct method for the time-optimal trajectory problem that uses the same upper bound to constrain the verification run of the* SPIN *model checker in order to deliver an optimal trajectory on-the-fly by encoding the optimality criteria into an LTL expression.*

The results of this thesis were published in [6].

3.5 Optimization in Graph Transformation Systems

Contribution 4. *I generalized the optimal trajectory problems of Petri nets over graph transformation systems with cost and time.*

4/3 *I introduced cost into graph transformation systems as the cost of a rule application. I defined formal semantics for evaluating graph transformation rules with cost. The corresponding publications are [5, 21, 22].*

4/4 *I introduced time into graph transformation systems as the a time attribute (time stamp) of graph elements. I defined formal semantics for evaluating graph transformation rules with duration. The results of this contribution were published in [2, 3, 10, 11].*

4/5 *I defined the optimal trajectory problem for graph transformation systems by generalizing its Petri net based definition. The corresponding publication is [6].*

The mapping of a GTS into Petri nets is the contribution of my co–author Dániel Varró. The introduction of time to graph transformation systems was a result of an international collaboration under the supervision of Prof. Reiko Heckel (at Univ. Paderborn) and Daniel Varró (at TU Budapest). The adaptation of Petri net based optimization techniques to GTS optimization is my own contribution both in case of GTS with cost and GTS with time.

4 Applications and Future Work

Optimal trajectory problem in high-level modeling languages. Since Petri nets can be generated automatically from high-level modeling languages (like UML, or BPMN) [46, 53] the introduced solutions can be used in the simultaneous optimization and verification of high-level modeling languages.

Maximum likelihood diagnostics. Another direction is to utilize the elaborated solutions in Petri net related fields. For instance, Petri nets describe the fault propagation in some models that can be extended by the likelihood of the corresponding fault in diagnostic problems [45]. The likelihood of the reachability of a state, i.e. the likelihood of a trajectory can be calculated from the likelihood parameters assigned to the transitions. Then the search for a path between two states with maximum likelihood forms an optimal trajectory problem and the optimal trajectory solution algorithms can be adapted to solve the problem.

BDD–guided state space traversal. Both the solution structures of the PNS problem and the graph structure of the Petri net OT problem can be described as a boolean function, i.e. it can be represented by a binary decision diagram. Since the logical representation of the connection between places and transitions is similar to the neutral extension of a decision mapping an interesting issue is to analyze how the BDD of the Petri net OT problem can be used in the guidance of the branching in the adaptation of the ABB algorithm.

List of own publications

Number of publications: 22
Number of peer-reviewed publications: 13
Approximate number of independent citations: 107

Book chapter and journal papers

[1] A. Pataricza, T. Bartha, G. Csertán, S. Gyapay, I. Majzik, and D. Varró, *Formális módszerek az informatikában*. Typotex, 2004. In Hungarian.

[2] S. Gyapay, D. Varró, and R. Heckel, "Graph transformation with time," *Fundam. Inform.*, vol. 58, no. 1, pp. 1–22, 2003.

[3] S. Gyapay, A. Schmidt, and D. Varró, "Joint optimization and reachability analysis in graph transformation systems with time," *Electronic Notes in Theoretical Computer Science*, vol. 109, pp. 137–147, 2004.

[4] S. Gyapay and A. Pataricza, "Optimal trajectory generation for Petri nets," *Acta Cybernetica*, vol. 17, pp. 225–245, January 2005.

[5] S. Varró-Gyapay and D. Varró, "Optimization in graph transformation systems using Petri net based techniques," *Electronic Communications of the EASST (ECEASST)*, vol. 2, 2006. Selected papers of Workshop on Petri Nets and Graph Transformations.

[6] S. Varró-Gyapay, "Optimization in graph transformation systems with time using Petri net based techniques," *Electronic Communications of the EASST*, vol. 51, pp. 1–12, 2012.

Conferences and workshops

[7] S. Gyapay, "Operation research methods in Petri net–based models of IT systems," in *Mini–Symposium 2002*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 38–39, IEEE Hungary Section (BUTE Student Branch), February 4–5 2002.

[8] S. Gyapay, "Operation research methods in Petri net–based analysis of IT systems," in *The Third Conference of PhD Students in Computer Science* (T. Csendes, ed.), (Szeged, Hungary), p. 36, July 1–4 2002.

[9] S. Gyapay, A. Pataricza, J. Sziray, and F. Friedler, "Petri net-based optimization of production systems," in 6th *IEEE International Conference on Intelligent Engineering Systems* (A. Lovrenčić and I. J. Rudas, eds.), pp. 465–469, Organized and published by Faculty of Organization end Informatics, University of Zagrev, Croatia, May 26–28 2002.

[10] S. Gyapay, R. Heckel, and D. Varró, “Graph transformation with time: Causality and logical clocks,” in *Proc. ICGT 2002: 1st International Conference on Graph Transformation* (H.-J. Kreowski and P. Knirsch, eds.), LNCS, (Barcelona, Spain), pp. 120–134, Springer-Verlag, October 7–12 2002.

[11] S. Gyapay and R. Heckel, “Towards graph transformation with time,” in *Proc. AGT 2002: Workshop on Applied Graph Transformation* (H.-J. Kreowski and P. Knirsch, eds.), (Grenoble, France), pp. 131–140, 2002.

[12] S. Gyapay, “Process Network Synthesis-based analysis of Petri net models,” in *Mini–Symposium 2003*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 20–21, IEEE Hungary Section (BUTE Student Branch), February 4–5 2003.

[13] S. Gyapay and A. Pataricza, “A gradual filtering method for simultaneous verification and optimization of Petri nets,” in *The John von Neumann PhD Conference*, (Budapest University of Technology and Economics, Faculty of Electrical Engineering and Informatics), pp. 27–30, October 2 2003.

[14] S. Gyapay and A. Pataricza, “A combination of Petri nets and Process Network Synthesis,” in *2003 IEEE International Conference on Systems, Man & Cybernetics, Invited Sessions/Track on "Petri Nets and Discrete Event Systems"*, (Washington, D.C., USA), pp. 1167–1174, IEEE Press, October 5-8 2003.

[15] S. Gyapay and A. Pataricza, “Optimization methods for reachability analysis of Petri net models,” in *Formal Methods for Railway Operation and Control Systems (Proceedings of Symposium FORMS-2003, Budapest, Hungary, May 15-16)* (G. Tarnai and E. Schneider, eds.), pp. 53–60, L’ Harmattan, Budapest, May 17–23 2003.

[16] S. Gyapay and A. Pataricza, “Eine Kombination von Petrinetzen und Optimierun,” in *Wissenschaftlichen Mitteilungen der 15. Frühlingsakademie, Balatonfüred, Hungary*, pp. 24–29, 2003.

[17] S. Gyapay, “Petri háló modellek folyamatszintézis–alapú analízise,” in *Fiatal Műszakiak Tudományos Ülésszaka*, (Cluj-Napoca, Romania), March 2003.

[18] S. Gyapay, “Solving the optimal trajectory problem using SPIN,” in *Mini–Symposium 2004*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 18–19, IEEE Hungary Section (BUTE Student Branch), February 2004.

[19] S. Varró-Gyapay, A. Pataricza, and Á. Nagy, “Optimization of production nets under temporal constraints,” in *Veszprém Optimization Conference: Advanced Algorithms*, (Veszprém, Hungary), December 13-15 2004.

[20] S. Gyapay, “Model–based optimization and verification of IT systems,” in *Conference of PhD Students in Computer Science* (T. Csendes, ed.), (Szeged, Hungary), p. 52, July 1–4 2004.

[21] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, and S. Varró-Gyapay, “Termination criteria for model transformation,” in *Proc. FASE 2005: Internation Conference on Fundamental Approaches to Software Engineering* (M. Cerioli, ed.), vol. 3442 of *LNCS*, (Edinburgh, UK,), pp. 49–63, Springer, April 2005.

[22] D. Varró, S. Varró-Gyapay, H. Ehrig, U. Prange, and G. Taentzer, “Termination analysis of model transformations by Petri nets,” in *Proc. Third International Conference on Graph Transformation (ICGT 2006)* (A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, eds.), vol. 4178 of *LNCS*, (Natal, Brazil), pp. 260–274, Springer, 2006.

Acknowledgement

I am very much grateful to my supervisor Professor András Pataricza for his support, ideas, and help during my PhD studies and later on. I am also very grateful to all of my colleagues and earlier supervisors at TU Budapest, TU Berlin, University of Paderborn and ELTE for their valuable help, and comments according to my work, especially to Reiko Heckel who was the supervisor of my master thesis.

I would like to thank my husband for his continuous support both in the professional activities and in the management of our family. I thank my parents, my parents in law and my children for their tolerance and help during writing my dissertation. I would like to thank all the people around me for their continuous encouragement to write this thesis.

My work was supported by the OTKA T038027 Hungarian National Research Foundation Grants, SEGRAVIS European Research Training Network, DECOS (IST-511764, FP6) and SENSORIA (IST-3-016004, FP6) European research projects.

Bibliography

[23] OMG Systems Modeling Language, 2007. http://www.omgsysml.org.

[24] Business Process Model and Notation, 2011. http://www.bpmn.org.

[25] A. Pataricza and O. Dobán and Á. Szőke. Costs/benefits of using formal methods. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 104–105, 2004.

[26] N. Aizenbud-Reshef, B. T. Nolan, and Y. S.-G. J. Rubin. Model traceability. *IBM Systems Journal*, 45:515–526, 2006.

[27] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[28] G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, Z. Ujhelyi, and D. Varró. Implementing efficient model validation in EMF tools: Tool demonstration. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, Lawrence, Kansas, USA, 11/2011 2011. IEEE Computer Society.

[29] A. Bondavalli, M. D. Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML-based system design. *Comput. Syst. Sci. Eng.*, 16(5):265 – 275, 2001.

[30] E. Brinksma and A. Mader. Verification and optimization of a PLC control schedule. In *SPIN Model Checking and Software Verification*, volume 1885 of *Lecture Notes in Computer Science*, pages 73–92, Berlin, Germany, 2000. Springer.

[31] C. F. Camerer. Behavioral game theory. experiments in strategic interaction. *The Journal of Socio-Economics*, 32(6):717–720, 2003.

[32] A. Corradini and U. Montanari. Specification of concurrent systems: from Petri nets to graph grammars. *Quality of Communication-Based Systems*, pages 35–52, 1995.

[33] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook on Graph Grammars and Computing by Graph Transformation*, volume 2: Applications, Languages and Tools. World Scientific, 1999.

[34] F. Fleurey, B. Baudry, P.-A. Muller, and Y. L. Traon. Qualifying input test data for model transformations. *Software and Systems Modeling*, pages 185–203, 2009.

[35] M. Fowler and K. Scott. *UML Distilled. Applying the Standard Object Modeling Language*. Addison-Wesley, 1997.

[36] F. Friedler, K. Tarjan, Y. W. Huang, and L. Fan. Combinatorial algorithms for Process Synthesis. *Computers Chemical Engineering*, 16:313–320, 1992.

[37] F. Friedler, J. B. Varga, E. Feher, and L. T. Fan. Combinatorially accelerated Branch–and–Bound method for solving MIP model of Process Network Synthesis, nonconvex optimization and its applications. *State of the Arts in Global Opimization, Computational Methods and Applications*, pages 609–626, 1996.

[38] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[39] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[40] H. G. Knehler. *Transformation von Graph-Grammatiken in Petri-Netze*. TU Berlin, FB Informatik, Bericht Nr. 88-02, September 1987. NewsletterInfo: 30.

[41] H.-J. Kreowski. A comparison between Petri-nets and graph grammars. In *Proceedings of the International Workshop on Graph theoretic Concepts in Computer Science*, WG '80, pages 306–317, London, UK, 1981. Springer-Verlag.

[42] S. P. Miller. Certification issues in model based development. Technical report, Advanced Technology Center, Rockwell Collins, 2006.

[43] S. P. Miller. Proving the shalls: Requirements, proofs, and model-based development. In *14th IEEE Int. Requirements Engineering Conference (RE'06)*, page 266, 2006.

[44] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., 2004.

[45] A. Pataricza. Model based dependability analysis, 2008. DSc Thesis.

[46] A. Pataricza, A. Balogh, and L. Göczy. Verification and validation of nonfunctional aspects in enterprise modeling. *Enterprise Modeling and Computing with UML, Idea Group*, pages 261–303, 2006.

[47] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt University of Technology, 1962.

[48] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations: Foundations*. World Scientific, 1997.

[49] T. C. Ruys. Optimal scheduling using Branch-and-Bound with SPIN 4.0. In *Proc. 10th International SPIN Workshop*, volume 2648 of *LNCS*, pages 1–17, Portland, Oregon, USA, May 9–10 2003. Springer.

[50] I. Stürmer, M. Conrad, H. Dörr, and P. Pepper. Systematic testing of model-based code generators. *IEEE Trans. Software Eng.*, pages 622–634, 2007.

[51] D. Varró. Design and analysis techniques for precise model transformations in model-driven development, 2013. DSc Thesis.

[52] W. L. Winston. *Operációkutatás, módszerek és alkalmazások*, volume 1. Aula Kiadó, 2003.

[53] M. Wirsing, M. Hölzl, L. Acciai, F. Banti, A. Clark, A. Fantechi, S. Gilmore, S. Gnesi, L. Gönczy, N. Koch, et al. Sensoria patterns: Augmenting service engineering with formal analysis, transformation and dynamicity. In *Leveraging Applications of Formal Methods, Verification and Validation*, pages 170–190. Springer, 2009.