1.Formal Verification Problems in a Big Data World: Towards a Mighty Synergy

- In this paper we introduce a distributed approach which exploits techniques typically used by the “big data” community to enable verification of very complex systems using “big data” approaches and cloud computing facilities.
- Problem
  - the development of techniques and tools able to cope with the complexity of models coming from real word examples.
  - the state explosion problem.
- Solution
  - symbolic model checking with ordered binary decision diagrams (especially in the area of hardware verification), partial order reduction techniques, and bounded model checking

- → Demonstration
  - → Distributed CTL Model Checking
  - → Our distributed MapReduce approach is based on the fixed-point characterizations of the basic temporal operators of CTL to evaluate in parallel massive sets of states satisfying a given formula. The basic idea is to apply an iterative MapReduce algorithm, where at each iteration we compute the predicate transformer on the output of the previous iteration until we reach the fixed-point.

2.Dynamic Models for the Formal Verification of Big Data Applications Via Stochastic Model Checking

- → Definition
  - → In this paper we propose a modeling framework for BDAs. A key feature of our framework is that it allows to apply verification techniques to prove properties of the given models. The proven properties are then supposed to hold for the BDAs execution. This approach generates models of the application execution by abstracting only the characteristics of the framework and application that are relevant from the verification perspective.
- → Solution
  - → In this paper we presented a first-principle modeling framework for the execution of BDAs. First-principle modeling provides guarantees on the generality of the model, and most important, allows us to synthesize control strategies (e.g., the resource allocation policy in xSpark) and validate them.

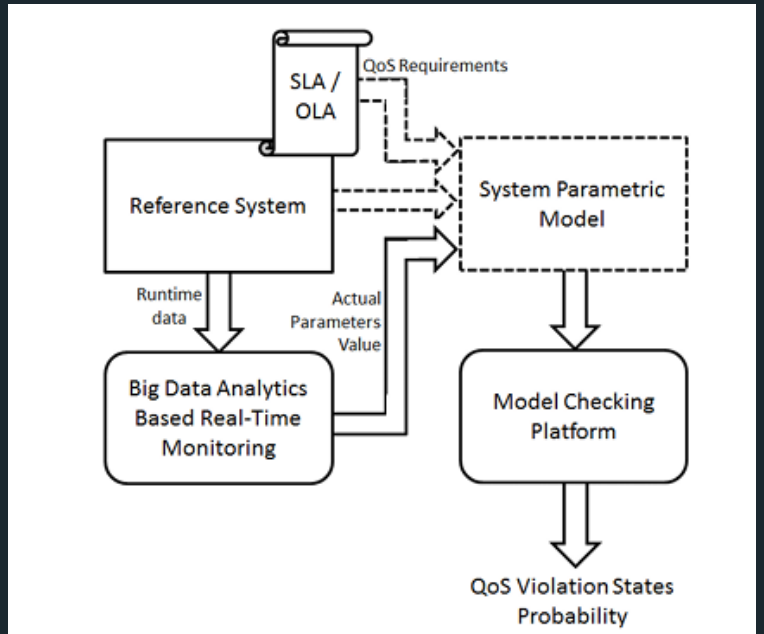
- → Demonstration
  - → We implemented the proposed model family in the stochastically model checker UPPAAL and we used the implementation to show:
    - → (i) that the model captures the behavior of an application running on a cluster of machines;
    - → (ii) that probabilistic queries can be done on the behavior of the application and of the resource allocation policy.

3.Big Data Analytics for QoS Prediction Through Probabilistic Model Checking

- Definition
  - In this paper we propose a model checking based approach to predict QoS(quality of service (QoS) prediction) of a formally described process.
- Solution
  - The continuous model checking is enabled by the usage of a parametrized model of the monitored system, where the actual value of parameters is continuously evaluated and updated by means of big data tools.
    - 1) Specification of the parameterized QoS stochastic model and QoS constraints to monitor
    - 2) Real-time data analysis and parameters synthesis
    - 3) Generation of the internal state-transition Model representation
    - 4) Execution of the Probabilistic Model-Checking to quantify the likelihood of future QoS state
    - 5) QoS Verification

→ Demonstration

→ The proposed QoS Prediction approach has been validated with respect to a Smart Grid (SG) case study. SG is the integration of the IT infrastructure into a traditional power grid in order to continuously exchange and process information to better control the production, consumption and distribution of electricity. For this purpose Smart Meters (SMs) devices are used to measure variations of electric parameters (e.g. voltage, power, etc.) and send such data to a computational environment which, in turn, analyze and monitor it in a real-time fashion. In this case study, our tool performs the remote monitoring on behalf of an Energy Distributor (ED) which purchases electric power from Energy Producers (EPs) and retails it to Energy Consumers (ECs). The primary goal of the ED is to balance the purchased electric power with respect to the variations of power demand.



The QoS Monitoring and Prediction achitecture

4.A Methodology for Real-Time Data Verification exploiting Deep Learning and Model Checking

- → Definition
  - → This paper presents a methodology for real-time data extraction and verification. In particular, considering the lacking of real-time data in sport analytics context, we propose a method to generate a data-set of player positions from soccer game videos, considering deep learning techniques, in order to extract player position, in terms of x-axis and y-axis, related to the accuracy of the detection.
- → Problem
  - → in outdoor situations like on a soccer field, there are different parameters that affect the robustness of the algorithms, such as the lightning, the environmental conditions, the distance of the pitch from the camera, and also the colors of the teams involved in the analysis.

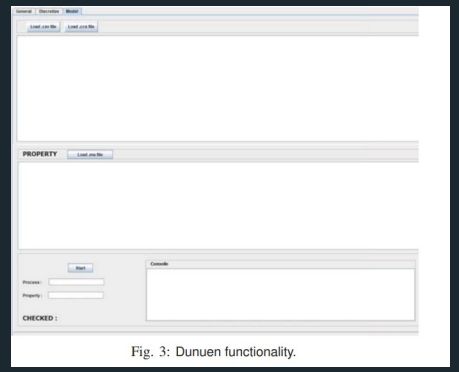


Fig. 3: Dunuen functionality.

→ Solution

• providing a set of pre-processing functionalities such as loading CSV file, deleting of specific rows and saving the modified CSV file;

• providing variable discretization in the loaded CSV, exploiting two different discretization methods: equal frequency or equal width method;

• building of CCS model from the CSV and load the property file;

• performing model checking of the property file loaded on the constructed CCS (Figure 3).


```
'frame','acc_g1s1','x_g1s1','y_g1s1', ...,'acc_g1s2','x_g1s2','y_g1s2', ...
        ...
'5', '95%','1216','1096', ...,'96%','1078','874', ...
'6', '92%','411','131', ...,'97%','578','752', ...
'7', '88%','782','633', ...,'NULL','NULL','NULL', ...
'8', '98%','1113','993', ...,'86%','778','1271', ...
        ...
```

Fig. 4: CSV data structure.

5.Big Data on Linear Temporal Logic Formulas

- → Problem
  - → The existing Linear Temporal Logic (LTL) formula sets are small ones.
- → Solution
  - → an algorithm for generating LTL formulas is proposed in this study, which can precisely fix the length of random formulas. First, we set an external loop controlling the number of generated formulas. Second, an inner loop is set to control the length of formulas. Final, a set consisting of 10 million formulas is generated. The experimental results confirm the comparative advantages of the new method. To the best of our knowledge, this is the first big-data-oriented set of LTL formulas that is expected to become a benchmark one.

Algorithm 1. An algorithm for generating LTL formulas

INPUT:

the length of each formula: n,

the number of randomly generated LTL formulas: m

OUTPUT: all LTL formulas

BEGIN

A:={X,G,F,U,!,/\,\/};

B(0):={p,q,r};

for i=1 to m:

for j=1 to n:

randomly select a logical operator x from A;

if x is an unary operator, then

randomly select a formula y from B(j-1);

new_formula:=xy;

add new_formula to B(j);

else // x is a binary operator

randomly select a formula y1 from B(random(s)); // random(s)∈0<=s<=j-1

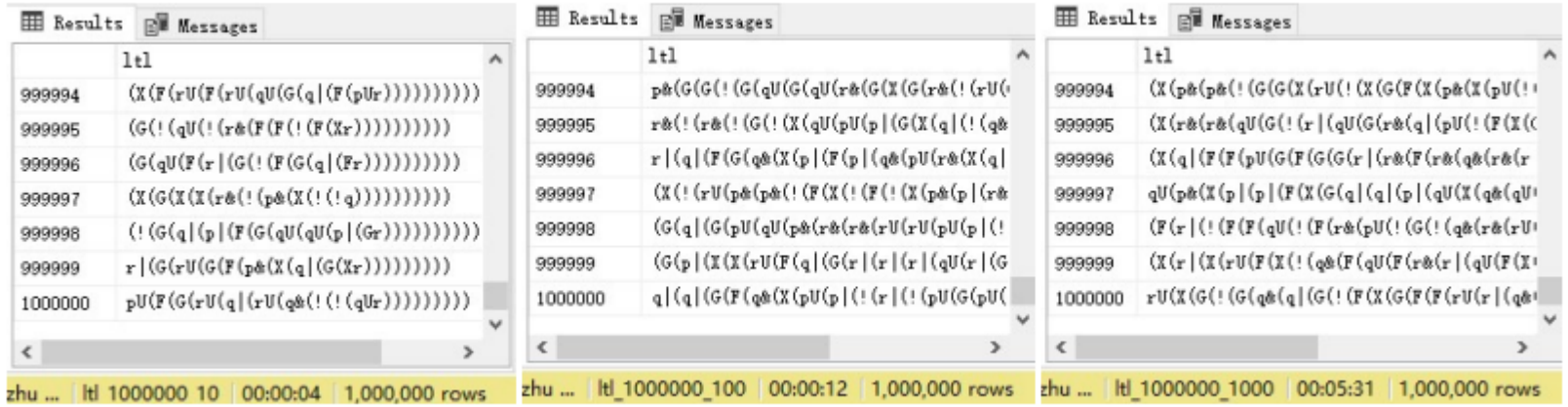
randomly select a formula y2 from B(j-1-random(s));

new_formula:=y1xy2;

add new_formula to B(j);

- Demonstration
- Such works can compare the performance and efficiency among different LTL model checking algorithms on small samples. In contrast, the big data set presented in this work contains ten million LTL formulas, which makes it possible to compare different LTL model checking algorithms and LTL satisfiability checking algorithms on large-scale samples. In addition, with the new method at hand, users can further build a bigger data set consisting of one hundred million formulas within ten days, only on a personal computer. After all, today's LTL has been used widely, making a study on LTL with small samples likes a toy, which can no longer meet the needs of the era of big data. This background can help us understand the potential benefits of using a larger scale set of LTL formulas established in this work.

- To verify the effectiveness of algorithm 1.



6. Recommender systems in model-driven engineering

→ Purpose

→ This study aims to serve as a guide for tool builders and researchers in understanding the MDE tasks that might be subject to recommendations, the applicable recommendation techniques and evaluation methods, and the open challenges and opportunities in this field of research.

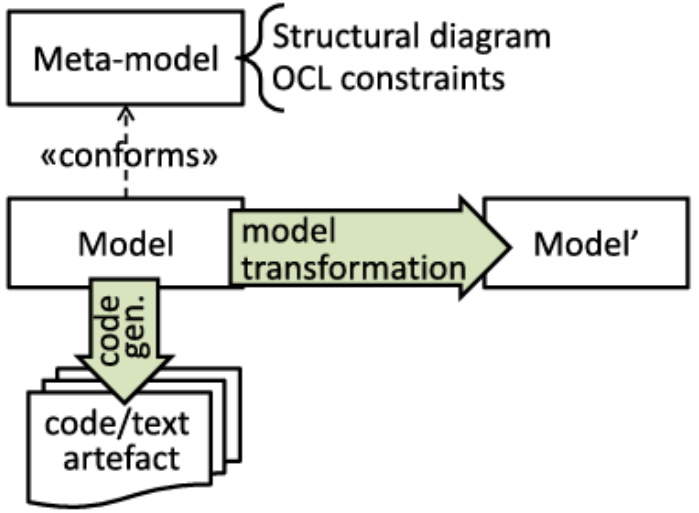


Fig. 1 Main elements of MDE solutions

Summary

→ In this paper , we have presented a systematic mapping review of existing research works on RSs for MDE. We have classified those works along four main dimensions (domain, tooling, recommendation and evaluation) characterized by means of feature models.

→ The review has allowed answering three research questions. First, we have seen that current RSs mainly target model completion and repair. Second, the most used recommendation methods in MDE are knowledge-based and content-based. Finally, we have identified research gaps and opportunities in the area, like implementing RSs to help in developing transformations and code generators, finding and reusing artefacts and creating artefacts from scratch. We encourage the community to pick these challenges to improve the current MDE practice and tooling.

Table 1 Terms used in the formal search query

Recommender systems/purpose	Modelling/MDE
Recommender	Model-driven
Recommendation	Domain-specific language
Model completion	State machine
Model reuse	Model transformation
Model repair	Code generation
Transformation completion	Code generator
Transformation reuse	Unified modelling language
Transformation repair	UML
Generator completion	
Generator reuse	
Generator repair	
Quick fix	
Quick fixes	
Assistant	
Assistance	

7. Automatic B-model repair using model checking and machine learning

- Problem
  - The B-method, which provides automated verification for the design of software systems, still requires users to manually repair faulty models.
- Solution
  - This paper proposes B-repair, an approach that supports automated repair of faulty models written in the B formal specification language. After discovering a fault in a model using the B-method, B-repair is able to suggest possible repairs for the fault, estimate the quality of suggested repairs and use a suitable repair to revise the model. The suggestion of repairs is produced using the *Isolation* method, which suggests changing the pre-conditions of operations, and the *Revision* method, which suggests changing the post-conditions of operations. The estimation of repair quality makes use of machine learning techniques that can learn the features of state transitions. After estimating the quality of suggested repairs, the repairs are ranked, and a best repair is selected according to the result of ranking and is used to revise the model.

- Demonstration
  - This approach has been evaluated using a set of finite state machines seeded with faults and a case study. The evaluation has revealed that B-repair is able to repair a large number of faults, including invariant violations, assertion violations and deadlock states, and gain high accuracies of repair. Using the combination of model checking and machine learning-guided techniques, B-repair saves development time by finding and repairing faults automatically during design.

8. Business Application Modeler: A Process Model Validation and Verification Tool

→ Purpose

→ We present the Business Application Modeler (BAM), which is a modeling and Validation & Verification tool that integrates modeling of processes and formal graphical validation rules.

→ These rules can be automatically applied to process models. In particular, the modeler is supported by visualizations of checking results directly in the process models. Next to highlighting mechanisms this support includes recommendations for the correction of errors.

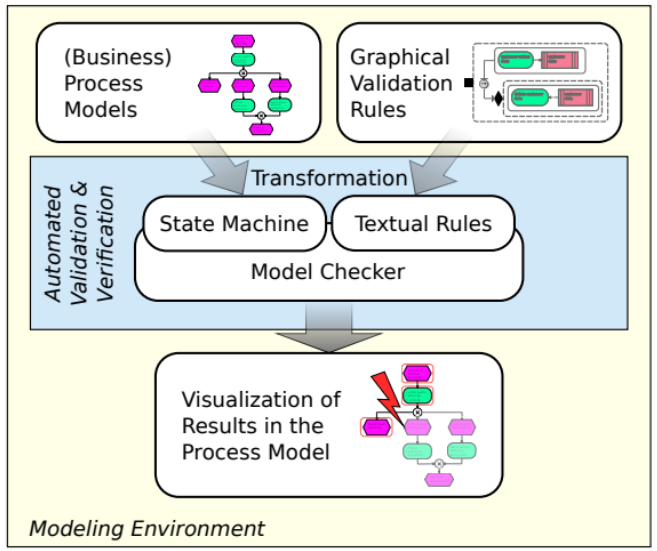


Fig. 1. Structure of BAM and the general validation process.

- Contribution
- Figure 1 depicts BAM's core areas. First, the modeling environment to model processes, graphical validation rules and visualize checking results. Second, the Validation & Verification which enables the automated checking whether the processes models comply to the rules.
- In this contribution we extended BAM with mechanisms for the visualization of results.

BUSINESS APPLICATION MODELER

A.
BAM provides a rule editor that allows the graphical modeling of validation rules. The notation for these rules is the Graphical Computational Tree Logic (G-CTL).
B.
Basically, BAM allows to adapt multiple checking techniques via a plug-in interface.
C.
Additionally, a concept called MultiView is implemented in BAM. MultiViews allow the assignment of specific responsibilities in the process modeling workflow and provide mechanism to handle the increasing complexity of process models.

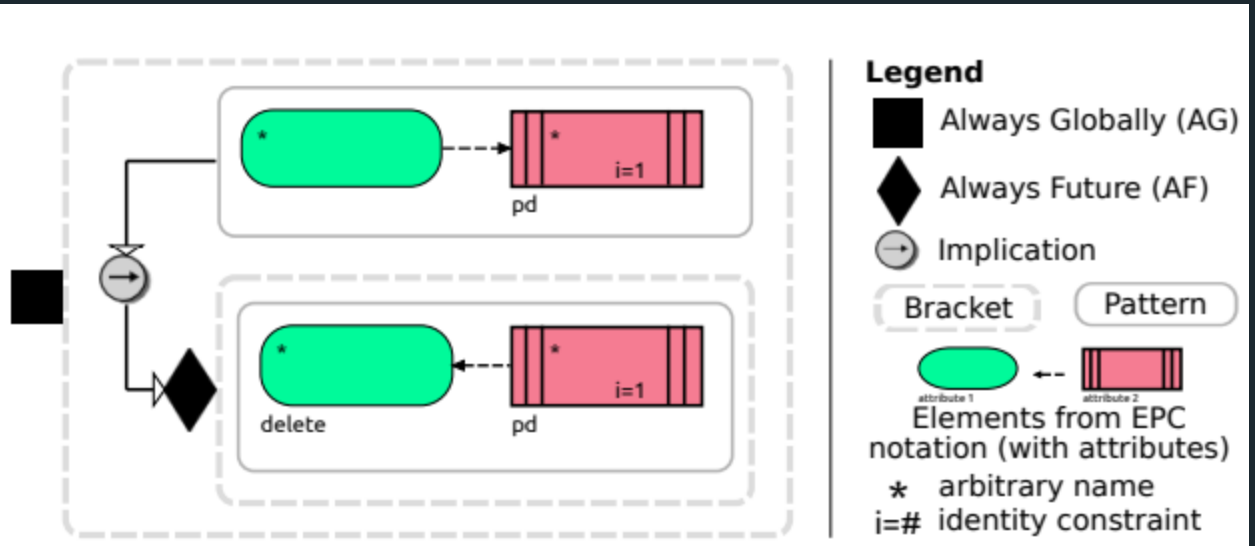


Fig. 2. A G-CTL example rule, requiring the deletion of personal data [5].

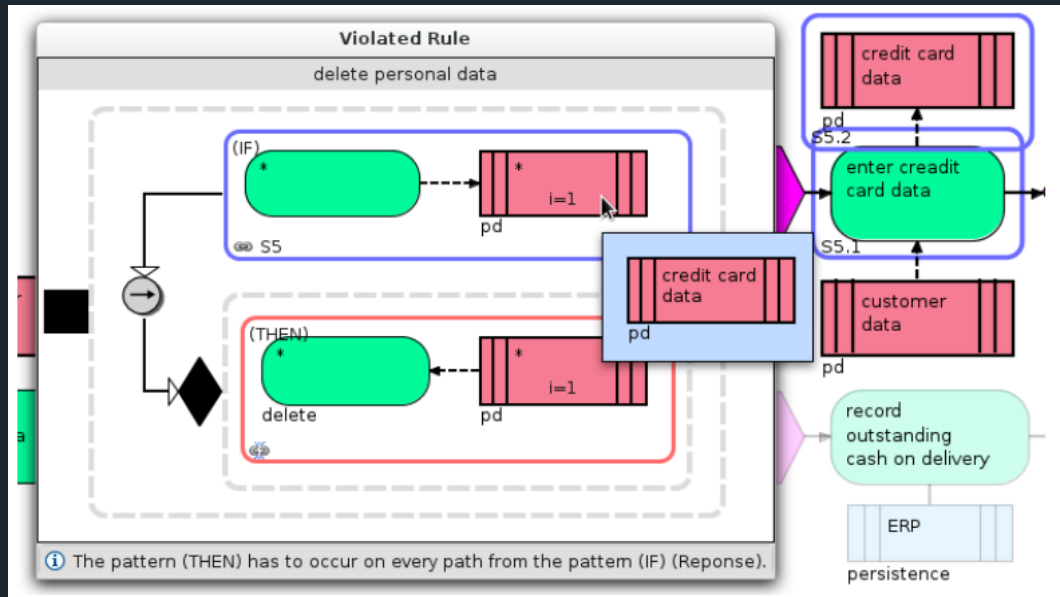


Fig. 3. Visual feedback utilizing the G-CTL rule. Elements matched by the upper pattern are highlighted. A recommendation is presented under the rule.

9. Automated verification of model transformations based on visual contracts

- → Purpose
  - → In this paper we fill this gap by proposing a declarative language for the specification of visual contracts, enabling the verification of transformations defined with any transformation language. The verification is performed by compiling the contracts into QVT(**Query/View/Transformation**) to detect disconformities of transformation results with respect to the contracts.
- → Solution
  - → based on the well-known design by contract paradigm, we have presented a visual, declarative language called PAMOMO to specify behavioral semantic contracts for M2M transformations in an implementation-independent way.

→ Demonstration

→ Using PAMOMO to verify its own translation into QVT-relations

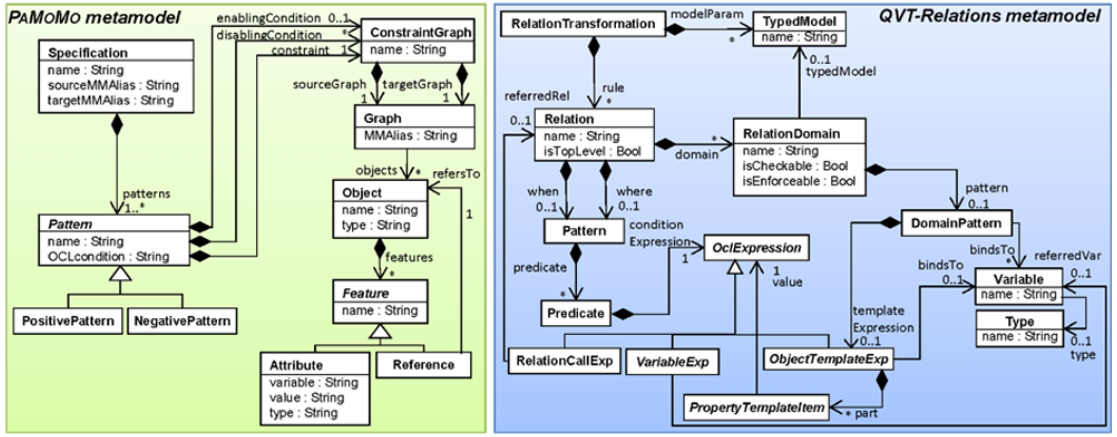


Fig. 32 PAMOMO (*left*) and QVT-R (*right*) metamodels

Fig. 33 A positive invariant for PAMOMO-to-QVT-R

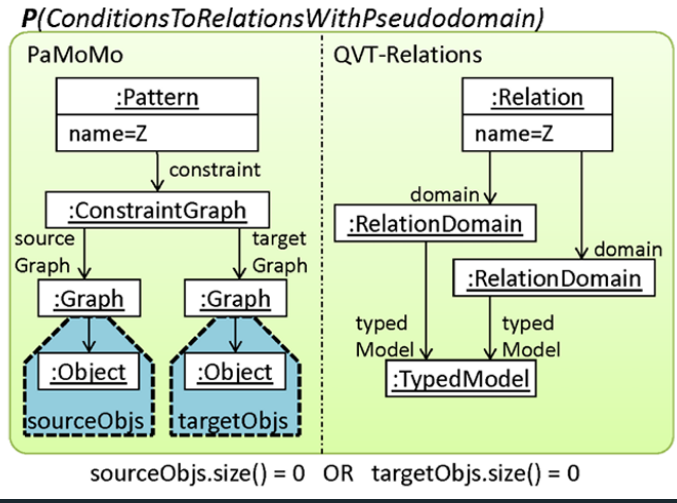
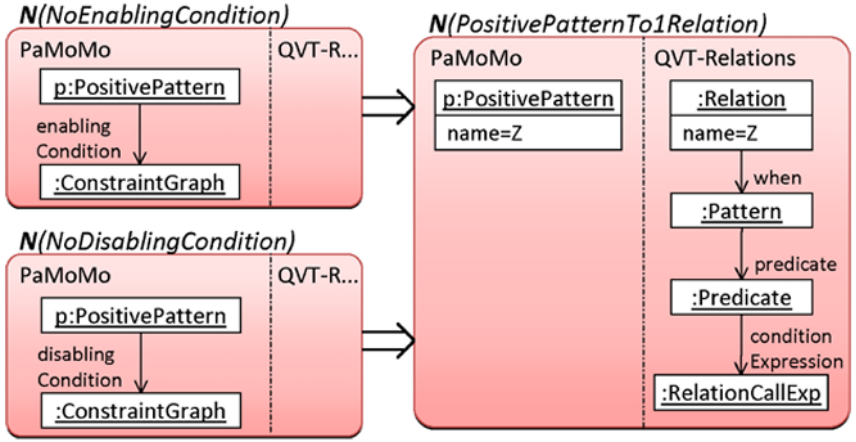


Fig. 34 A negative invariant for PAMOMO-to-QVT-R



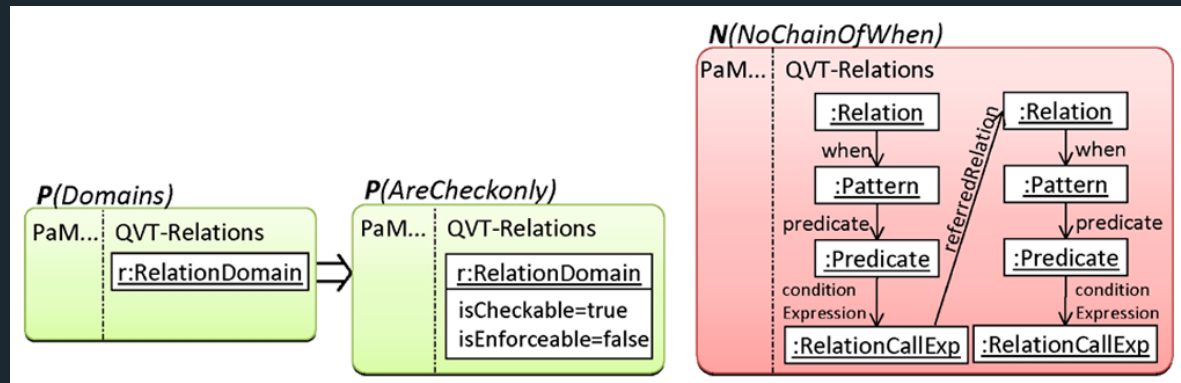


Fig. 35 Two postconditions for PAMOMO-to-QVT-R

10. Patterns in property specifications for finite-state verification

- Purpose
  - In a recent paper, we proposed a pattern-based approach to the presentation, codification and reuse of property specifications for finite-state verification. Since then, we have carried out a survey of available specifications, collecting over 500 examples of property specifications. We found that most are instances of our proposed patterns. Furthermore, we have updated our pattern system to accommodate new patterns and variations of existing patterns encountered in this survey. This paper reports the results of the survey and the current status of our pattern system.

- → Demonstration
  - → An example of a property specification pattern is given in Figure 1 (we use a variant of the “gang-of-four” pattern format).
  - → A pattern comprises a name or names, a precise statement of the pattern’s intent (i.e., the structure of the behavior described); mappings into common specification formalisms, examples of known uses, and relationships to other patterns.

Precedence

Intent

To describe a relationships between a pair of events/states where the occurrence of the first is a necessary pre-condition for an occurrence of the second. We say that an occurrence of the second is enabled by an occurrence of the first. Also known as *Enables*.

Example Mappings

In these mappings S enables the occurrence of P.

CTL S precedes P:

Globally	$\neg E[\neg S\ \mathcal{U}(P \wedge \neg S))]$
Before R	$\neg E[(\neg S \wedge \neg R)\ \mathcal{U}(P \wedge \neg S \wedge \neg R \wedge EF(R))]$
After Q	$\neg E[\neg Q\ \mathcal{U}(Q \wedge \neg E[\neg S\ \mathcal{U}(P \wedge \neg S)])]$
Between Q and R	$AG(Q \rightarrow \neg E[(\neg S \wedge \neg R)\ \mathcal{U}(P \wedge \neg S \wedge \neg R \wedge EF(R))])$
After Q until R	$AG(Q \rightarrow \neg E[(\neg S \wedge \neg R)\ \mathcal{U}(P \wedge \neg S \wedge \neg R)])$

LTL S precedes P:

Globally	$\Diamond P \rightarrow (\neg P\ \mathcal{U}(S \wedge \neg P))$
Before R	$\Diamond R \rightarrow (\neg P\ \mathcal{U}(S \vee R))$
After Q	$\Box \neg Q \vee \Diamond(Q \wedge (\neg P\ \mathcal{U}(S \vee \Box \neg P)))$
Between Q and R	$\Box((Q \wedge \Diamond R) \rightarrow (\neg P\ \mathcal{U}(S \vee R)))$
After Q until R	$\Box(Q \rightarrow ((\neg P\ \mathcal{U}(S \vee R)) \vee \Box \neg P))$

Quantified Regular Expressions Let Σ be the set of all events, let $[-P, Q, R]$ denote the expression that matches any symbol in Σ *except* P, Q, and R, and let $e^?$ denote zero or one instance of expression e.

Event S precedes P:

Globally	$[-P]^* \mid ([-S, P]^*; S; \Sigma^*)$
Before R	$[-R]^* \mid ([-P, R]^*; R; \Sigma^*) \mid ([-S, P, R]^*; S; \Sigma^*)$
After Q	$[-Q]^*; (Q; ([-P]^* \mid ([-S, P]^*; S; \Sigma^*)))^?$
Between Q and R	$[-Q]^*; (Q; ([-P, R]^* \mid ([-S, P, R]^*; S; [-R]^*)); R; [-Q]^*)^*; (Q; [-R]^*)^?$
After Q until R	$[-Q]^*; (Q; ([-P, R]^* \mid ([-S, P, R]^*; S; [-R]^*)); R; [-Q]^*)^*;$ $(Q; ([-P, R]^* \mid ([-S, P, R]^*; S; [-R]^*)))^?$

Examples and Known Uses

Precedence properties occur quite commonly in specifications of concurrent systems. One example is describing a requirement that a resource (e.g., a lock) is only granted in response to a request.

Precedence and response properties often go together. A response property says that when S occurs then an occurrence of P must follow. If we want to restrict P to only follow S then we use a precedence property. Note that these properties do not guarantee a one-to-one correspondance between an occurrence of S and an occurrence of P. Such additional constraints can be added using the *constrained* variations of these patterns.

The mappings given in this pattern do not describe precedence properties where P and S occur simultanously (i.e., S must strictly precede P). To relax this constraint use the *possibly empty* variation of the pattern.

Relationships

A generalization of precedence properties that allows for multiple separate states/events to constitute P and S is called the precedence chain pattern.

Figure 1: Precedence Pattern

Paper link	paper	domain	Keywords	contribution	No.
https://dl.acm.org/doi/abs/10.1145/2591062.2591088	Formal verification problems in a big data world: towards a mighty synergy	Bigdata and model checking	Formal Verification, CTL, Big Data, MapReduce	Our experiments report that our approach can be used effectively to manage and analyze state spaces of different orders of magnitude. In particular, the major is the complexity of the model to be analyzed, the major is the scalability of our distributed algorithms that shown a potential for a super-linear speedup.	1
https://ieeexplore.ieee.org/abstract/document/8511410	Dynamic Models for the Formal Verification of Big Data Applications via Stochastic Model Checking	Bigdata and model checking	model checker, Big Data Applications (BDAs), first-principle	In this paper we presented a first-principle modeling framework for the execution of BDAs. First-principle modeling provides guarantees on the generality of the model, and most important, allows us to synthesize control strategies (e.g., the resource allocation policy in xSpark) and validate them.	2
https://arxiv.org/abs/1405.0327	Big Data Analytics for QoS Prediction Through Probabilistic Model Checking	Qos prediction and model checking	Big Data Analytics, QoS Prediction, Model Checking, SLA compliance monitoring	To support Big Data analysis of QoS information in this paper ,we have proposed a QoS prediction framework which take s advantage of the qualitative and quantitative analysis performed by a probabilistic model-checking technique.	3
https://ieeexplore.ieee.org/abstract/document/9005994	A Methodology for Real-Time Data Verification exploiting Deep Learning and Model Checking	methodology for real-time data extraction and verification	formal methods, real-time data extraction and verification, CSV (comma separated value)	In this work, we proposed a methodology to construct a data-sets from real-time data and to verify behavioral properties exploiting formal methods.	4
https://ieeexplore.ieee.org/abstract/document/9482368	Big Data on Linear Temporal Logic Formulas	Bigdata and model checking	linear temporal logic; big data; model checking; randomly generated formula	In contrast, the big data set presented in this work contains ten million LTL formulas, which makes it possible to compare different LTL model checking algorithms and LTL satisfiability checking algorithms on large-scale samples.	5

Paper link	paper	domain	Keywords	contribution	No.
https://link.springer.com/article/10.1007/s10270-021-00905-x	Recommender systems in model-driven engineering	Recommender systems in MDE research	Model-driven engineering · Recommender systems · Systematic mapping review	This study aims to serve as a guide for tool builders and researchers in understanding the MDE tasks that might be subject to recommendations, the applicable recommendation techniques and evaluation methods, and the open challenges and opportunities in this field of research.	6
https://link.springer.com/article/10.1007/s10515-019-00264-4	Automatic B-model repair using model checking and machine learning	Automatic B-model repair	model checking B-model repair machine learning	This paper proposes B-repair, an approach that supports automated repair of faulty models written in the B formal specification language.	7
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6912283	Business Application Modeler: A Process Model Validation and Verification Tool	modeling and Validation & Verification tool	Business Application Modeler (BAM)	BAM allows the specification of formal, graphical validation rules (G-CTL) on the level of process models. Rules can be specified in a reusable manner, allowing to automatically check the process model for these properties (requirements).	8
https://link.springer.com/article/10.1007/s10515-012-0102-y	Automated verification of model transformations based on visual contracts	Model-to-Model (M2M) transformations	Model-Driven Engineering · Model transformation · Contract-based specification · Verification · QVT-relations	In this paper we fill this gap by proposing a declarative language for the specification of visual contracts, enabling the verification of transformations defined with any transformation language.	9
chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://dl.acm.org/doi/pdf/10.1145/302405.302672	Patterns in Property Specifications for Finite-State Verification*	use of high-level abstractions in writing formal specifications is an important factor in making automated formal methods, specifically finite-state verification tools, more usable	Patterns, finite-state verification, formal specification, concurrent systems	We have described an updated pat- tern system we developed for property specifications in finite-state verification and have collected a large sam- ple of specifications that suggests that most property	10