Automated Compliance Verification of Business Processes in Apromore

Heerko Groefsema[1], Nick R.T.P. van Beest[2], and Abel Armas-Cervantes[3]

[1] University of Groningen, The Netherlands
h.groefsema@rug.nl
[2] Data61, CSIRO, Australia
nick.vanbeest@data61.csiro.au
[3] Queensland University of Technology, Australia
abel.armas@qut.edu.au

Abstract. This paper presents the integration of two plugins, a *declarative process specification generator* and a *compliance verifier*, into the Apromore advanced business process analytics platform. The integrated toolchain has a range of applications of interest to both practitioners and researchers. For example, it can be used in the areas of business process compliance, flexibility and variability. The generator can extract a set of formal specifications that declaratively describe a set of business process variants; whereas the verifier can check whether temporal properties over business process models hold. The verifier can use two different model checker tools: NuSMV2 and NuXMV. These plugins allow business analysts to verify if a newly developed process model adheres to rules and regulations or a specification dictated by existing process model variants.

Keywords: Apromore, NuSMV2, formal verification, temporal logic, variability, compliance

1 Overview

Changes in laws and regulations affect the way organizations conduct their businesses, forcing them to achieve higher flexibility and business agility. Oftentimes, an organization manages several variants of the same process, which can emerge from mergers, handling of different products, customer-tailored services, or differences in market segments or legislations. As a result of this increasing number of business processes and their constant evolution, it becomes significantly more complex to continuously ensure their compliance with respect to the laws and regulations in place.

Automated compliance verification of business processes is an emerging field that aims to prove or disprove whether a business process adheres to a set of rules that have been imposed through laws, regulations, business policies, etc. Compliance verification, which applies to a broad set of predefined specifications, should not be confused with soundness verification, which checks a limited set of requirements, such as reachability, termination, and possible proper completion. Thus, compliance verification aims at formally checking whether the behavior of a given business process complies with a set of formal specifications.

[1, 2] and [3] propose a set of techniques for automated verification and automated specification generation, respectively. These techniques serve two related purposes: i) check if a business process is compliant with a set of predefined specifications described as *temporal logic* formulas, e.g., using *Computation Tree Logic (CTL)* [1, 2], and ii) generate a set of specifications – as temporal logic formulas– describing the

commonalities of a family of business process variants [3]. Subsequently, these techniques can be used to automatically check if a new process complies with the rules and regulations prescribed by existing process variants. These techniques use an advanced mapping from business process models to *Kripke structures* and *event structures* such that the information about the dependence (causality) and independence (concurrency) between tasks instances *within* parallel branches is maintained. As a result, the implementation causes limited state explosion in the presence of concurrency and, together with an advanced model reduction, ensures that the approach performs well even for highly complex models [2]. The reduced models are formally verified with respect to their specifications using NuSMV2.[4] An overview of the implemented operations is shown in Fig. 1 below.

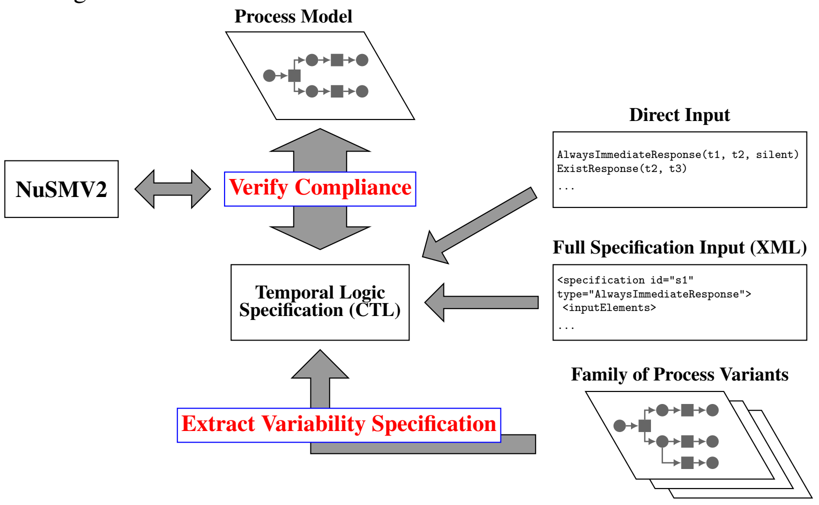

Process Model
Direct Input
AlwaysImmediateResponse(t1, t2, silent)
ExistResponse(t2, t3)
...
NuSMV2
Verify Compliance
Full Specification Input (XML)
<specification id="s1"
type="AlwaysImmediateResponse">
<inputElements>
...
Temporal Logic Specification (CTL)
Family of Process Variants
Extract Variability Specification

Fig. 1: Overview of the approach.

This paper presents the integration of the techniques proposed in [1, 2] and [3] into the Apromore advanced business process analytics platform.[5] Apromore is an open-source and extensible online process analytics platform, comprising state-of-the-art capabilities for managing and analyzing large process model collections. The operations for compliance verification and declarative process variant specification complement a wide range of existing capabilities provided by Apromore, like process model merging, simulation and similarity search. The two techniques are each wrapped into a plugin in Apromore: *Verify Compliance* and *Extract Variability Specification*.

The modeling languages supported are all those available in Apromore (BPMN, CPF, EPCs, Petri nets and YAWL), as Apromore offers conversions from any of these formats to the required Petri Net Markup Language (PNML) format [4]. As such, the integrated verification approach is independent of the process model input-format used.

In addition to a process model, the *Verify Compliance* plugin requires a set of temporal logic rules as input. There are three ways to specify such temporal logic rules:

– *Direct input:* the user can input the specification to be evaluated. The rules can be specified using keywords (column 1 in Table 1) instead of CTL formulas.

[4] NuSMV is a software tool for the formal verification of finite state systems using temporal logics. It is designed to be an open architecture for model checking and has been jointly developed by FBK-IRST and Carnegie Mellon University.

[5] http://apromore.org

- *Full specification input:* the user can input a predefined set of specifications in xml-format using keywords.[6]
- *Family of process variants:* the user can extract a specification in xml-format using the plugin *Extract Variability Specification*. This plugin extracts a specification from a process family, which can immediately be used to verify the compliance of a new process variant to that specification.

Table 1 provides an overview of possible keywords for direct input. The first column presents the keyword that can be used for input. The second column presents an intuitive interpretation of the formula, while the third column shows the corresponding CTL notation. Although any formula in CTL format can be verified, Table 1 shows the different types of temporal logic formulas that are applicable to a variability specification. These keywords are extensible, and user-defined keywords can be added in an xml-file defining the *specification types*.

Keyword	**Description**	**CTL formula**
`AlwaysResponse(p, q)`	*p* is always eventually followed by *q*	`AG(p -> AF q)`
`AlwaysImmediateResponse(p,q,s)`	*p* is always directly followed by *q* (with silent *s*)	`AG(p -> A[(p | s) U q])`
`AlwaysImmediatePrecedence(p,q)`	*q* is always directly preceded by *p*	`!E[!p U q]`
`ExistImmediateResponse(p,q,s)`	there exists a path where *p* is directly followed by *q* (accounting for silent steps *s*)	`AG(p -> E[(p | s) U q])`
`ExistResponse(p,q)`	there exists a path where *p* is eventually followed by *q*	`AG(p -> EF q)`
`AlwaysConflict(p,q)`	*p* and *q* never occur together in the same path	`AG(p -> AG !q)`
`AlwaysParallel(p)`	the activities in group *p* are concurrent	`AG(p)`
`Group(p,q,r,...)`	creates a group *p* containing elements *q*, *r*, ...	–

Table 1: Direct input of specifications.

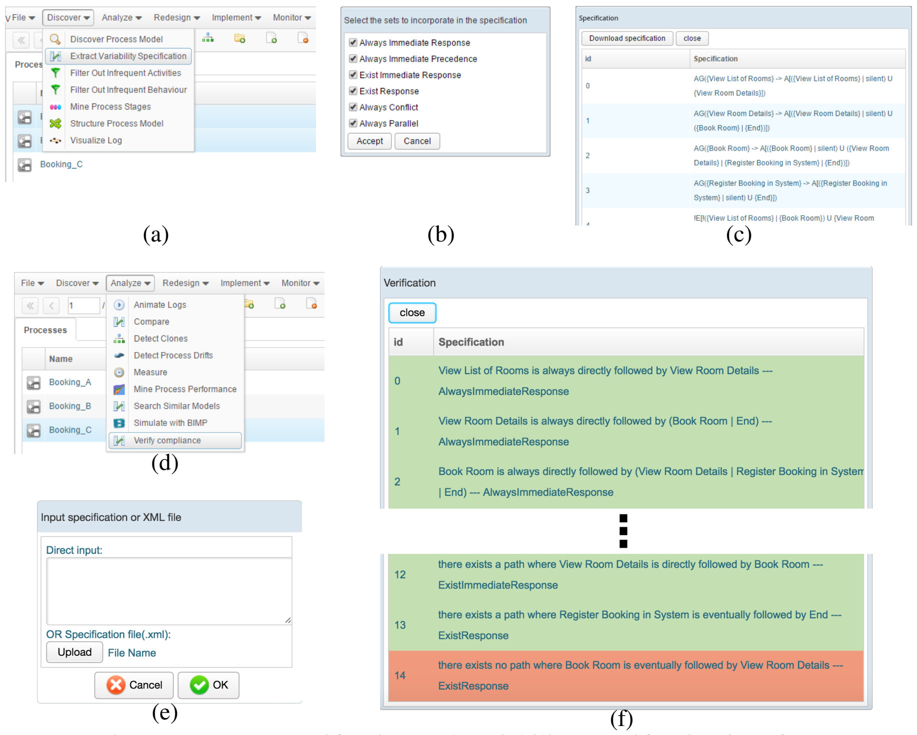
First, the process model is transformed internally into a reduced Kripke Structure [5], by eliminating irrelevant transitions for the temporal logic formulas to be evaluated. Further details on this technique are described in [1, 2]. Subsequently, the Kripke structure and temporal logic formulas are converted into the NuSMV2 model checker input format and passed to the model checker for verification. Once completed, the verification package interprets and returns the results to Apromore for user feedback in natural language, corresponding to the description as provided in Table 1.

Figure 2 shows the different user interfaces in Apromore for *Extract Variability Specification* and *Verify Compliance*. Fig. 2(a) shows the menu where the *Extract Variability Specification* operation is located. Subsequently, the user is allowed to select the types of specifications to be extracted (Fig. 2(b)). The resulting specifications are presented to the user as shown in Fig. 2(c). *Verify Compliance* is located under the menu-item “Analyze”, as shown in Fig. 2(d). The specification-format can be either via direct input, an existing xml-file, or the specification obtained earlier from the *Extract Variability Specification* plugin (Fig. 2(e)). The results of the verification are shown as a list of all formulas represented in natural language, where the specifications that evaluate to `true` are highlighted in green and the specifications that evaluate to `false` are highlighted in red (Fig. 2(f)).

2 Significance and Maturity

Our toolchain features a novel approach to formal design-time verification of business process models. It does not only allow the use of well-known temporal logics and model checking tools, but it also features unique insights into parallel and sequential execution of activities due to its advanced translation techniques. In addition, the approach causes

[6] User defined keywords are supported as well.



Select the sets to incorporate in the specification

- ☑ Always Immediate Response
- ☑ Always Immediate Precedence
- ☑ Exist Immediate Response
- ☑ Exist Response
- ☑ Always Conflict
- ☑ Always Parallel

Specification

id	Specification
0	AG({View List of Rooms} -> A[({View List of Rooms} | silent) U {View Room Details}])
1	AG({View Room Details} -> A[({View Room Details} | silent) U ({Book Room} | {End})])
2	AG({Book Room} -> A[({Book Room} | silent) U ({View Room Details} | {Register Booking in System} | {End})])
3	AG({Register Booking in System} -> A[({Register Booking in System} | silent) U {End}])
4	!E[!({View List of Rooms} | {Book Room}) U {View Room

Input specification or XML file

Direct input:

OR Specification file(.xml):

Verification

id	Specification
0	View List of Rooms is always directly followed by View Room Details --- AlwaysImmediateResponse
1	View Room Details is always directly followed by (Book Room | End) --- AlwaysImmediateResponse
2	Book Room is always directly followed by (View Room Details | Register Booking in Syster | End) --- AlwaysImmediateResponse
⋮	
12	there exists a path where View Room Details is directly followed by Book Room --- ExistImmediateResponse
13	there exists a path where Register Booking in System is eventually followed by End --- ExistResponse
14	there exists no path where Book Room is eventually followed by View Room Details --- ExistResponse

Fig. 2: Apromore Verification and Variability Specification interface.

limited state explosion and includes highly effective reduction techniques to cope with complex models to be verified. This technique comprises a set of fully formalized approaches, which start with a Petri net representation of the business process models, a Kripke structure translation, reduction with respect to stutter equivalence, and the representation of specifications using temporal logics.

The approach can be applied for the verification of temporal properties relevant to business process soundness, compliance, and variability of single business process models and even collaborative business process models. For instance, the reachability of an activity, their exclusive execution, or the absence of concurrent execution can be verified. In addition, the approach can verify whether a business process model is a member of a set of related business process models [3].

The toolchain offers a robust foundation for the formal verification of temporal properties of business process models at design-time in Apromore. Its features and feedback can be interpreted by business analysts to identify soundness, compliance, or other issues related to the possible temporal executions of the business process model through visual feedback. In addition, other modules can use the package for their own verification goals through its extensible specification format and/or its extensible Petri net translation features. At the same time, Apromore provides import, export, and mapping functionalities to and from a large variety of business process modeling languages, including BPMN, XPDL, EPML, ARIS, YAWL, and PNML, allowing for their verification through a PNML mapping.

The approach has been extensively evaluated, including a detailed analysis of expressive power, performance evaluations towards the translation of the Kripke structure and the reduction features, and several case studies in the areas of eGovernment and the telecommunications industry. Evaluations on the expressive power demonstrate that the implemented approach is able to accurately capture differences between different branching constructs that other design-time approaches can not capture. In addition, the approach allows for the evaluation of globally and locally in a branch next activity executions without the need for new or extended temporal logics. The performance evaluations show that Kripke structure generations are nearly instant for business process models with average to little concurrent behavior, while business process models with large amounts of concurrent behavior take seconds to generate, and business process models with extremely large concurrent behavior take mere minutes to generate. At the same time, the state reduction techniques increase in effectiveness with increased concurrent behavior. Evaluations on case studies demonstrate the applicability of the approach in the domains of compliance and variability. Detailed results of these evaluations can be found in [1, 2], and [3].

Apromore is the result of over seven years of ongoing development and is currently in version 5.0. The platform is implemented and deployed as a Software as a Service via a service-oriented architecture. The technologies used in Apromore combine Spring as the Java development framework, Maven as the dependency manager, OSGi as the plugin architecture, EclipseVirgo as the OSGi-based application server, and ZK as the AJAX front end. Together these technologies allow Apromore to be an extensible framework, where new plugins can be easily added to an ecosystem of advanced capabilities for analyzing and managing process model collections. These bundles include presentation capabilities with respect to process model restructuring, filtering of models based on process-related aspects, searching and querying for specific process patterns, advanced design and repair of process models, including configuring and merging of existing models, and evaluation capabilities to assess the quality, correctness and compliance of models, along with simulation and conformance checking techniques for benchmarking.

3 Screencast

A screencast of Apromores compare feature can be found at `https://youtu.be/wTiPgu8G370`. The public release of Apromore is available at `http://apromore.org` and its source code can be downloaded under the GNU LGPL license version 3.0 from `https://github.com/apromore/ApromoreCode`. The verification and variability packages are also available at `http://www.hgroefsema.nl`. The source code can be downloaded from `https://github.com/rug-ds-lab/BPMVerification`.

References

1. Groefsema, H., van Beest, N.R.T.P.: Design-time compliance of service compositions in dynamic service environments. In: Int. Conf. on Service Oriented Computing & Applications. (2015) 108–115
2. Groefsema, H., van Beest, N.R.T.P., Aiello, M.: A formal model for compliance verification of service compositions. IEEE Transactions on Services Computing (2016) To appear.
3. Groefsema, H.: Business Process Variability: A Study into Process Management and Verification. PhD thesis (2016)
4. Billington, J., Christensen, S., van Hee, K.M., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., Weber, M.: The petri net markup language: Concepts, technology, and tools. In: Int. Conf. on Applications and Theory of Petri Nets 2003. (2003) 483–505
5. Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge, Massachusetts and London, UK (1999)