Constraint-based analysis of business process models

Changizi, B.

1 Introduction

The term *business process modeling* is first introduced by S. Williams [Wil67] where he argues that the techniques for modeling physical control systems could be applied to business processes [DM03]. However, it took until the 1990s for the term *business process* to become popular [Hoo11].

At the time, companies started to think in terms of processes rather than functions and procedures [Rol95]. Process thinking ensures the right development direction by analyzing the chain of events in an organization. Examples include the events occurring from purchase to supply or from receiving orders to sales.

A *business process* is a set of related and structured activities, which serves a specific goal for a customer [Rol95]. The de-facto standard in the field of business process modeling is the Business Process Model and Notation (BPMN).

Business Process Model and Notation (BPMN) [Gro11], previously referred to as Business Process Modeling Notation, is a graphical representation of business process models based on flowcharting techniques. The main goal of BPMN is to provide an understandable notation for both technical experts and business users.

Similar to many modeling languages, it is possible for a BPMN model to contain errors. Syntactical errors are created by connecting the modeling elements in an

invalid manner. In general, syntactical errors can be detected simply by parsing the model [AP08]. A number of BPMN designing tools such as Eclipse BPMN Modeler [BPM], ARIS Express [ari], and Yaoqiang [Yao] can detect syntactical errors in models.

However, a model may contain behavioral errors, which are more complicated to detect. For instance, a model may represent a process that is not *sound*. A process is sound when every reachable state from an initial state has a way to reach a final state [GPR+07]. A process may contain *deadlock* or *livelock*. Deadlocks occur when a process can reach a non-final state that it cannot leave. Livelocks happen when a process ends in one path, but some states are still active with no progress possible. Detecting behavioral errors requires investigating the runtime behavior of a process. An informal approach to finding cases of deadlock and livelock in BPMN models has been proposed in [AP08] [TJ10], which is based on finding the pre-defined patterns of such errors in the model. Although this approach has low computational costs, it is not complete in term of finding other forms of errors.

Formalizing semantics of a BPMN process enables automated model checking of the process in order to detect behavioral errors. Similar to a variety of BPM systems on the market [DRMR13], the foundation of BPMN is based on Petri nets [vdA04]. The choice of Petri nets as foundation for BPM system implementation over other formal methods, often more expressive or specialized [RBM05, BHF05], is not surprising: hardly any model is as simple, intuitive, and naturally supports task traceability.

While undoubtedly Petri nets based models enable automated process analysis within BPM systems, they lack few desirable characteristics: i) They lack compositionality, which means that they cannot deal with large and complex systems. Ideally, we would like to plug semantic models for individual components to the semantic models of existing processes in a compositional way. ii) The classical Petri nets are not expressive enough and often are extended (e.g., with colors, reset and inhibitor arcs, priority transitions) to enable meaningful process analysis. Such extensions change the operational semantics of the model and generate incompatible dialects of process-specification languages adopted by various tools.

An alternative theory for coordinating concurrent components is called the Reo coordination language [Arb04]. Reo has been used to formalize semantics of Business Process Modeling Notation (BPMN) [AKM08b], UML Activity and Sequence Diagrams [CKA10], map BPEL fragments [STK+10], represent transactional workflows [KA13], implement service orchestrations [JSS+12] and service choreographies [MA07b].

In this dissertation, we propose formal semantics for Business Process Modeling Notation (BPMN) models in terms of Reo. The mapping of BPMN to Reo is implemented as a plugin in the Reo analysis tool-set in a model-driven paradigm. Our mapping completes the proposed mapping of BPMN to Reo in [AKM08b] by covering not only basic BPMN constructs, but also advanced structures such as BPMN transactions. In addition, our proposed mapping rules are expressed formally in a dedicated language for model to model transformation.

Since synchronization propagates through composition in Reo, it allows composition of components and services in an intuitive way, and addresses the issue (i) mentioned above. Reo is easily extensible to support more advanced process models, such as timed [MA07a] or stochastic workflows networks [MSKA10], via defining new channels. However, the open-ended nature of Reo channels makes it necessary to extend the formal semantics of Reo in order to include some new concepts.

Several dozen variations of semantic models for Reo have been proposed [JA12]. They vary from rather simple that cover basic Reo behavior (e.g., constraint automata [BSAR06]) to more complex models that capture specific behavioral aspects, e.g., context-sensitivity [CCA07]. In some of these semantic models, computing the overall semantics of a system given automata-based semantics for its parts (components, services or glue code) is computationally expensive. This hampers using the language for analyzing large real-world business processes.

In this dissertation, we present a constraint-based framework, which unifies various formal semantics of Reo. In this framework, the behavior of a Reo network is described as a constraint satisfaction problem (CSP). A CSP is a problem whose solutions must satisfy some limitations also known as constraints. The constraint-based nature of our approach allows simultaneous coexistence of several semantics in a simple fashion. The behavior of a Reo network is determined by the solutions to its CSP. Since any solution must satisfy all the encoded formal semantics, the framework eliminates any behavior inconsistent with (an aspect of) a formal semantics of Reo.

Another advantage of our proposed constraint-based approach compared to the existing approaches of deriving formal semantics of Reo is its efficiency due to efficient constraint solving methods and optimization techniques used in the off-the-shelf constraint solvers. We support this claim with a case study.

Among the behavioral aspects required to model a business process is *priority*. The notion of priority is necessary for modeling behaviors such as transaction and exception handling, where the data flow representing the error or exception should interrupt the normal flow. A formal semantics to model priority in Reo, named

Constraint Automata with Priority (CAP), has been proposed in [ABS15]. CAP provides means to model propagation and stopping the propagation of priority. Despite its comprehensive approach in modeling priority, the proposed semantics is computationally expensive for direct implementation.

Inspired by CAP, in this dissertation, we present an alternative approach to model priority in Reo by extending our constraint-based framework with priority-aware premises. Further, we extend our priority-aware formal model to support not only a binary notion of priority, as modeled in CAP, but also numeric priorities.

1.1 Contributions

The contributions of this dissertation are as follows:

- We present a model-driven mapping of business process models specified in BPMN into Reo networks. Such transformations enable application of automated analysis and model checking on business processes. We have implemented our proposed mapping in a rule-based fashion using a dedicated transformation language, which makes the implementation of the mapping concise, readable, and easy to maintain. We have integrated our mapping into the Extensible Coordination Tool-set (ECT), the integrated development environment for Reo. This makes it easier for business process models to be fed to various tools in ECT. Figure 1.1.1 shows an example of a BPMN model with the option to be converted to Reo using our BPMN to Reo plugin. Figure 1.1.2 depicts the generated Reo network.

- We provide an extensible constraint-based approach to unify various semantic models of Reo networks. We represent a problem of computing semantics for a complex Reo network by encoding semantics of individual channels as constraints and solving the corresponding constraint satisfaction problem. This approach bridges the expressiveness gaps and incompatibility among different Reo semantics. In addition, using a constraint-based approach replaces direct implementations of algorithms for calculating different Reo formal semantics.

- We extend our constraint-based framework to support the priority-aware behavior of Reo connectors. Priority is an important concept in modeling transactions. Our work makes it more straight-forward and less complicated to obtain Constraint Automata with Priority (CAP) formal semantics for Reo. Our framework is the only existing approach that integrates various behav-

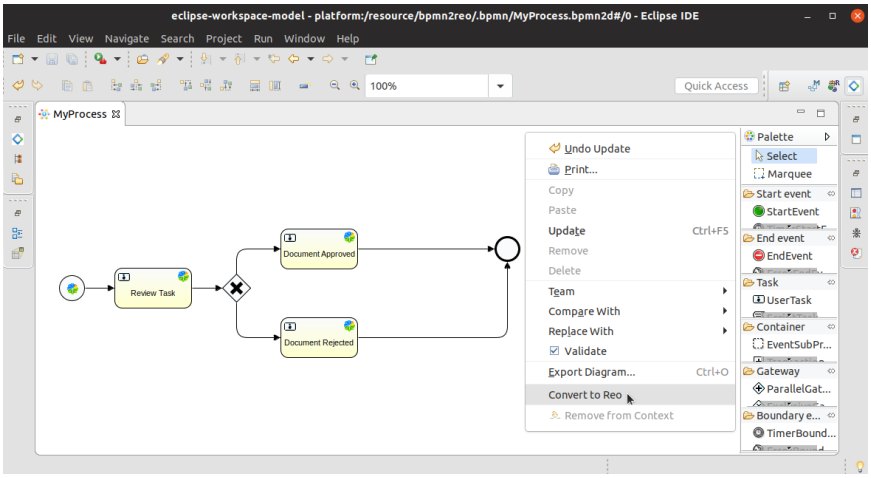

Figure 1.1.1: The BPMN to Reo converter menu in ECT

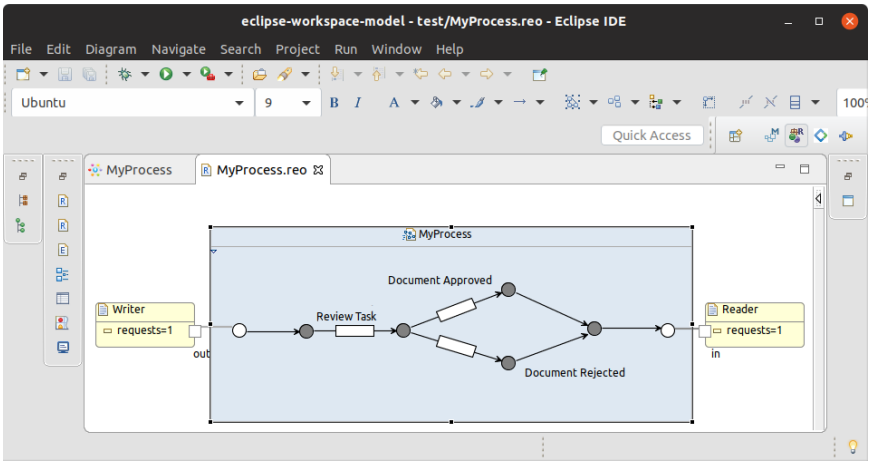

Figure 1.1.2: The mapping of Figure 1.1.1

ioral aspects of a Reo network (e.g. data-dependency, context-sensitivity, priority-awareness) under one umbrella.

1.2 Outline

The rest of this dissertation is organized as follows:

- In Chapter 2, we introduce BPMN 2 modeling elements and introduce an example of BPMN with problems. This chapter is based on the technical

content of [1], listed in Section 1.3.

- Chapter 3 provides an overview of the Reo coordination language. There, we describe the behavior of Reo primitives in an informal style.

- Chapter 4 contains an overview of several formal semantic models proposed for describing behavior of a Reo connector. The definitions of the semantics that are relevant to this work are given in details.

- Chapter 5 describes our rule-based model-driven approach in transforming BPMN models to Reo connectors. The transformation handles advanced BPMN elements, namely, transaction and compensation.

  An obstacle in computing execution semantics of some BPMN models with high-level elements such as transaction is that their behavior is too complicated and elaborated to directly be mapped to constructions of a language used for verification. To tackle this issue, we suggest a refinement procedure to substitute such high-level constructs with a set of simpler elements that together deliver the same functionality. This chapter is partially based on the technical content of [1], listed in Section 1.3.

- In Chapter 6, we introduce our constraint-based framework to capture the formal semantics of Reo networks, given by two different formal semantics namely, Constraint Automata with State Memory (CASM) and Connector Coloring (CC) [CCA07]. CASM is an extension of Constraint Automata (CA), which is one of the most popular semantics for Reo. We favor using CASM over CA, which is a simpler semantics, because CASM provides a mechanism to model the state values. This helps in treating the states symbolically. Therefore, unlike CA every data-item entering a buffer does not lead to a new state.

  To capture context sensitivity, a behavioral aspect that CA and some of its extensions miss, we use CC, which models context sensitivity in a Reo connector using graph coloring techniques.

  We present a tool to generate CASMs from Reo networks in a compositional manner, where the part of behavior that is not compliant with CC is ruled out.

  We employ highly optimized off-the-shelf constraint solvers instead of straightforward custom algorithms for computing the semantics [CKA12]. We provide formal arguments to show the correctness of our approach. Then, we present

an evaluation on the performance of our framework through a case study. The technical content of [4], listed in Section 1.3 is the basis of this chapter.

- In Chapter 7, we extend our framework to support priority and its propagation through a Reo connector. We propose a constraint-based solution to replace the custom algorithm to calculate the priority-aware behavior of a Reo connector [CKA19]. We first introduce a binary model of priority and show how it can be encoded in our constraint-based framework. Subsequently, we extend this solution to numeric priorities. We show the application of our model in a case study. This chapter is based on the technical content of [5], listed in Section 1.3.

- Chapter 8 concludes this thesis and outlines future research directions.

1.3 Publications

1. Behnaz Changizi and Natallia Kokash and Farhad Arbab. A Unified Toolset for Business Process Model Formalization. 7th International Workshop on Formal Engineering approaches to Software Components and Architectures, pages 147-156. ENTCS, 2010.

2. Behnaz Changizi and Natallia Kokash and Farhad Arbab. A Semantic Model for Service Composition with Coordination Time Delays. International Conference on Formal Engineering Methods, pages 106-121. 2010.

3. Behnaz Changizi and Natallia Kokash and Farhad Arbab. Input-Output Conformance Testing for Channel-based Service Connectors. In: Proceedings of PACO, pages 19–35. 2011.

4. Behnaz Changizi and Natallia Kokash and Farhad Arbab. A Constraint-based Method to Compute Semantics of Channel-based Coordination Models. International Conference on Software Engineering Advances. IARA, 2012.

5. Behnaz Changizi and Natallia Kokash and Farhad Arbab. Service Orchestration with Priority Constraints. International Conference on Fundamentals of Software Engineering, pages 194-209. LNCS, 2019.