Western Norway
University of
Applied Sciences

Formalization and analysis of BPMN using graph transformation systems

Tim Kräuter
Harald König, Adrian Rutle, and Yngve Lamo

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

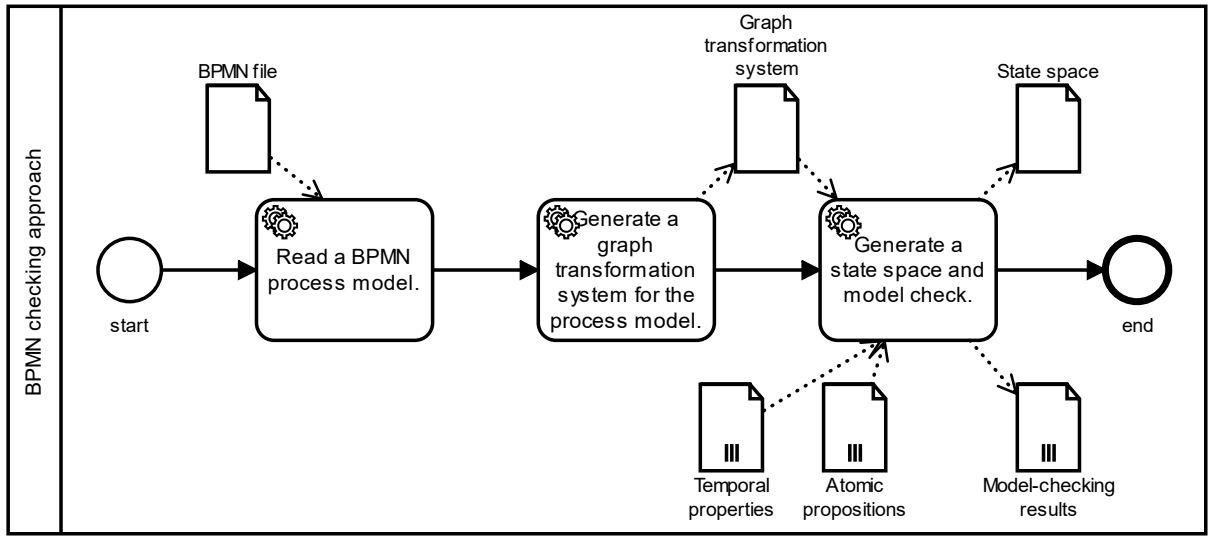
Introduction

Business Process Modeling Notation (BPMN) is a widely used standard notation to define intra- and inter-organizational workflows.

BPMN only has an informal description of its execution semantics [6].

Without a formalization it is difficult to check behavioral properties.

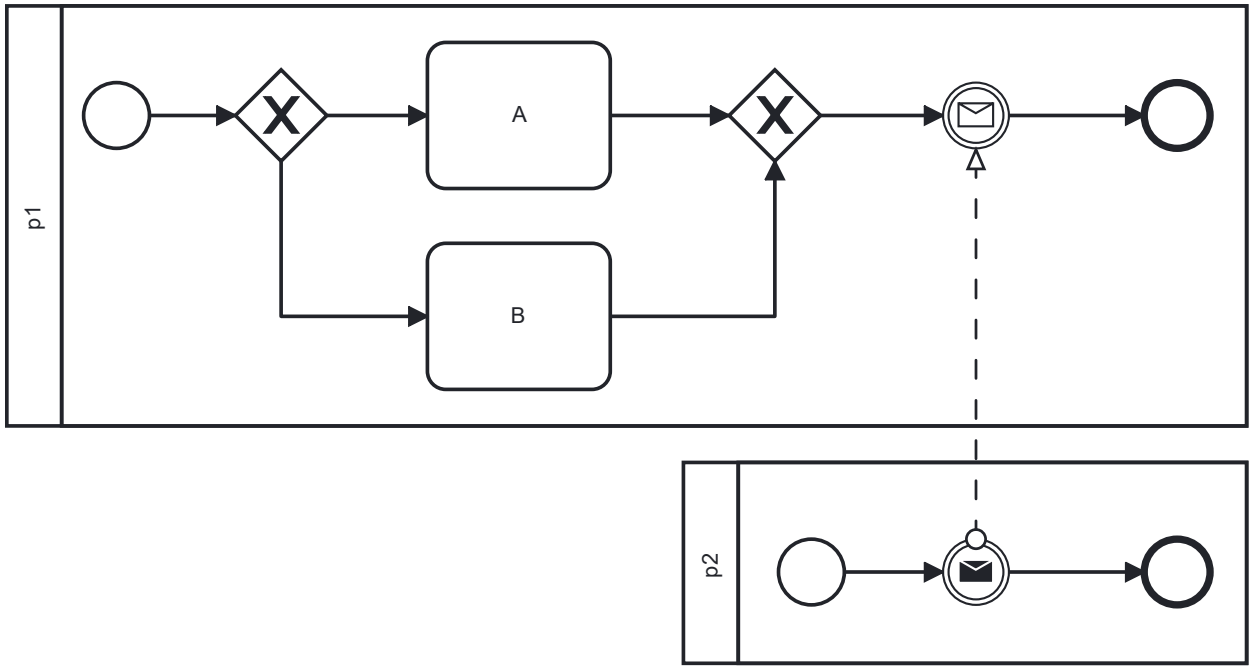
We propose a formalization based on graph rewriting to allow checking of behavioral properties.



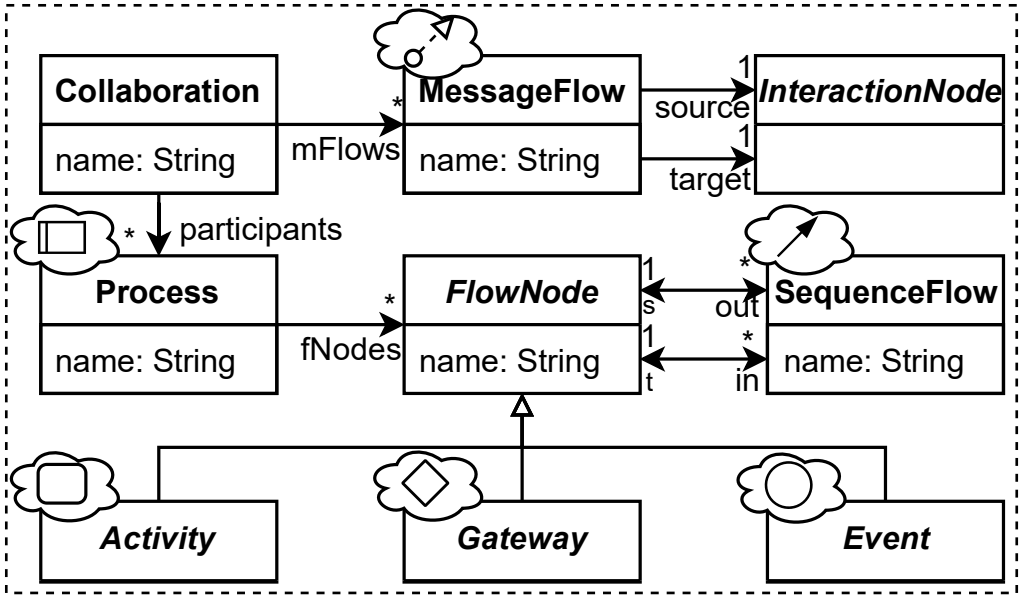
Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

Preliminaries: BPMN structure

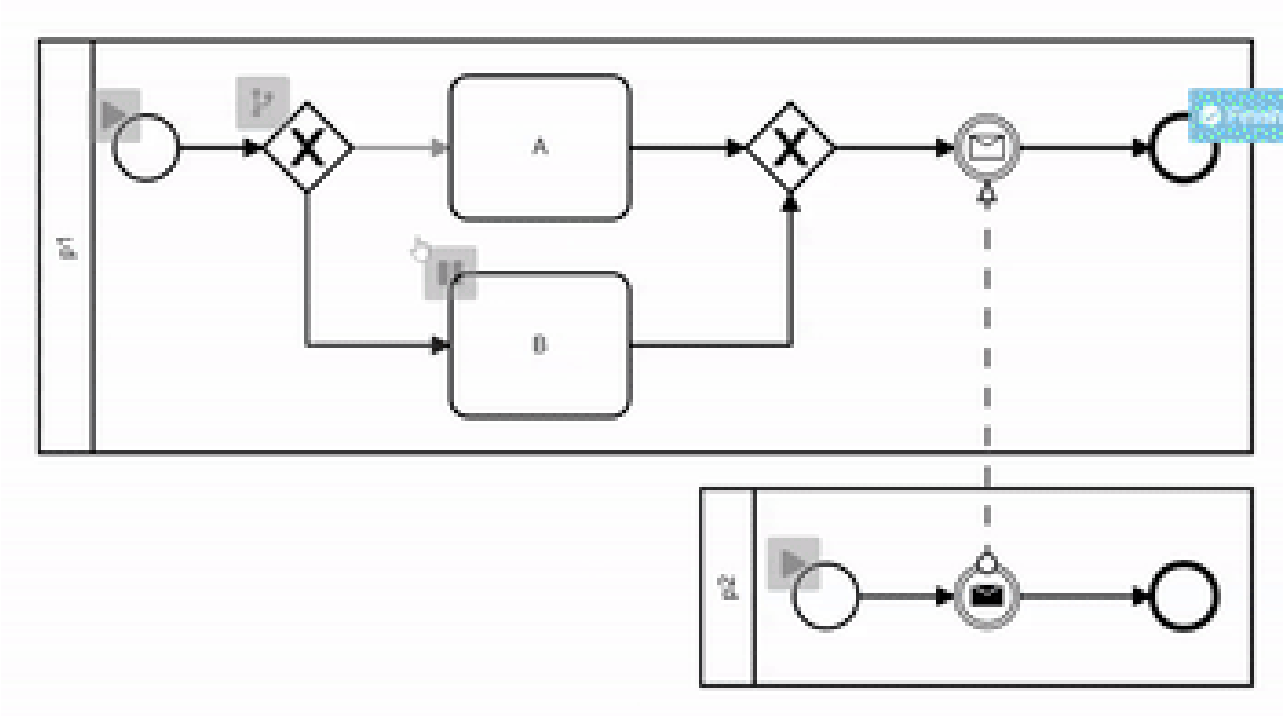


BPMN example



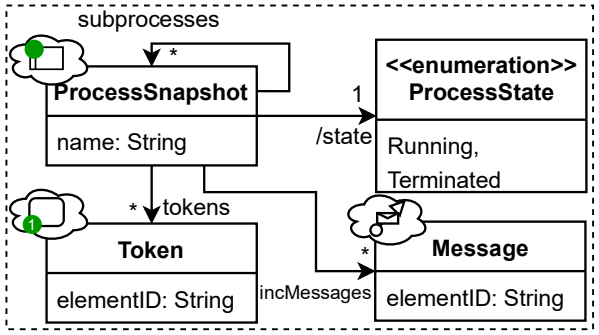
BPMN metamodel [6]

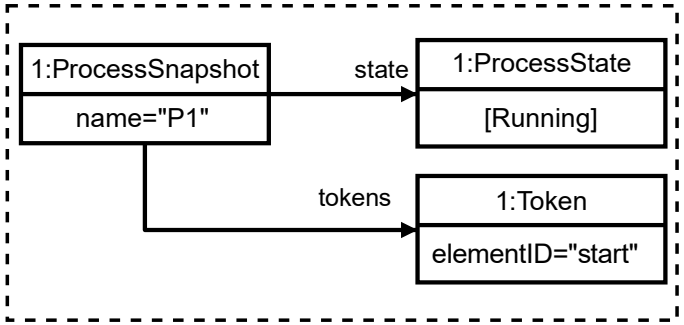
Preliminaries: BPMN semantics

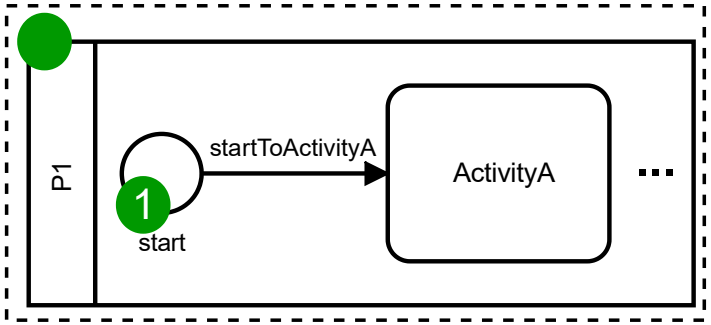


Preliminaries: Theoretical background

Each state/token distribution is represented as a **typed attributed graph**.







BPMN execution type graph | Abstract syntax | Concrete syntax

We are using the single-pushout (SPO) approach with negative application conditions and quantified nested rules [3,7,8].

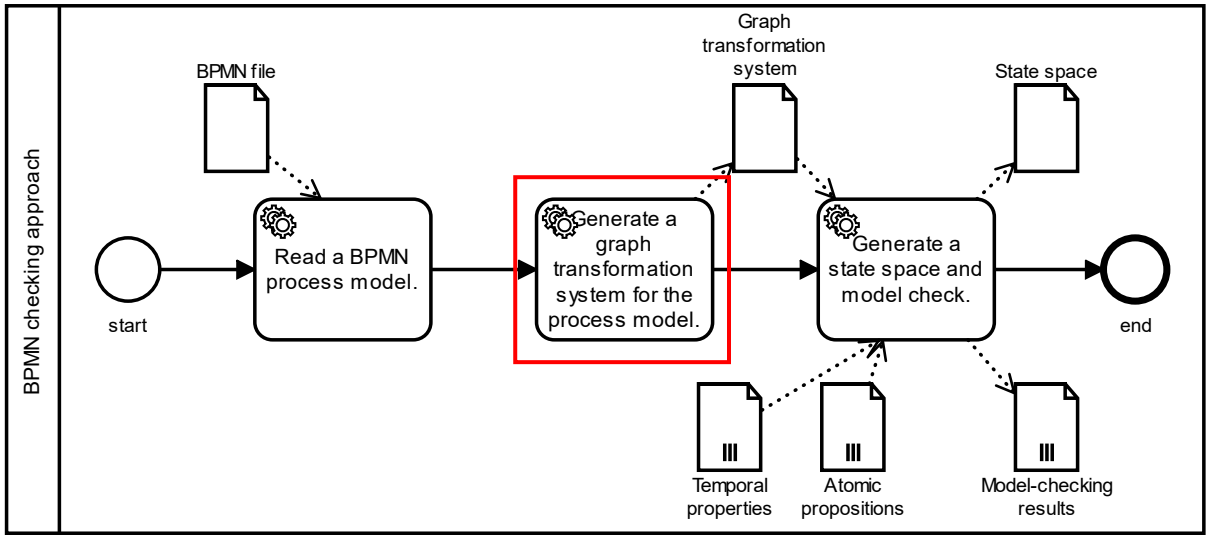
So far, SPO works well to formalize the BPMN execution semantics.

Dangling edge removal by SPO is not a problem.

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

BPMN semantics formalization: Overview



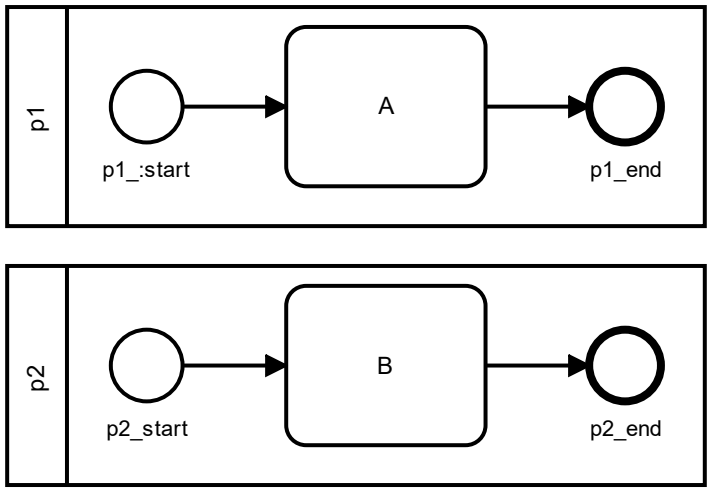
Model transformation from BPMN to graph transformation systems.

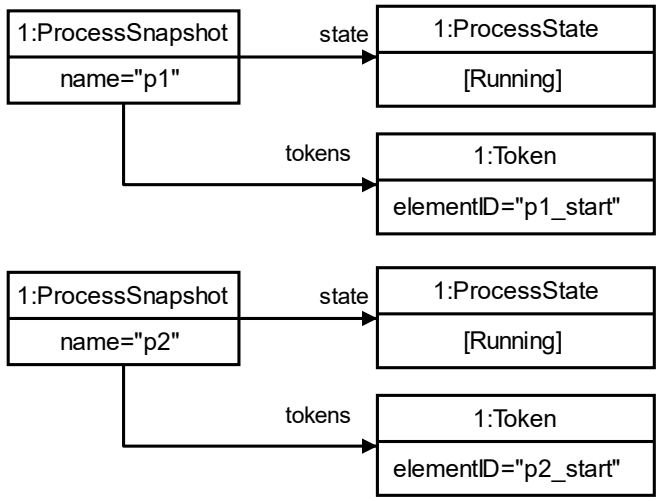
1	Start graph
2	GT-Rules

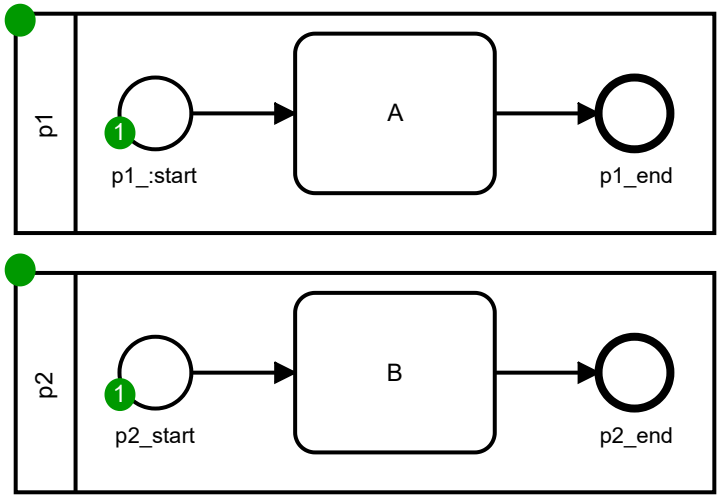
BPMN semantics formalization: Start graph generation

Generate a process snapshot for each process in the BPMN model.

For each start event we add a token to the respective process snapshot.







BPMN model

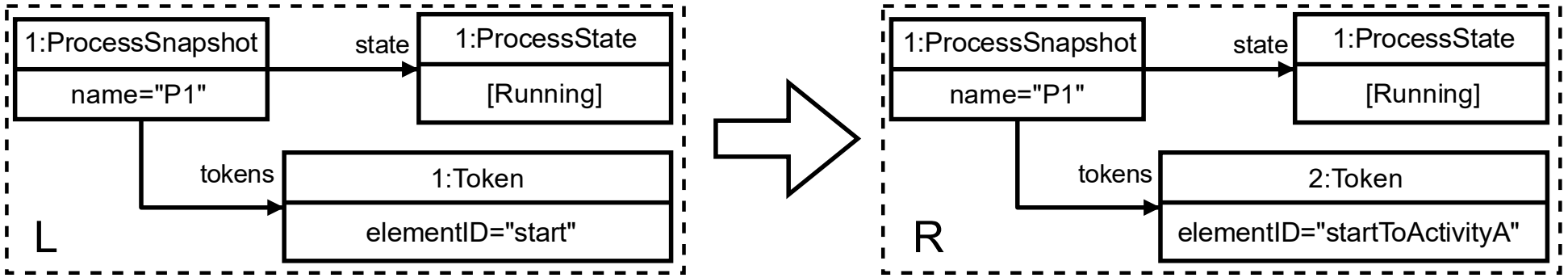
Abstract syntax

Concrete syntax

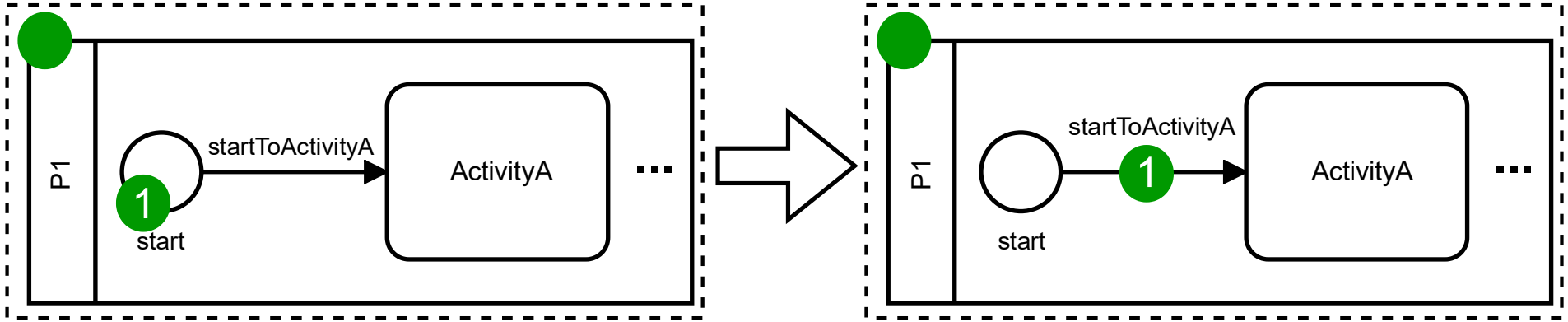
Start graph

BPMN semantics formalization: GT-Rule syntax

Rule conforming to the type graph

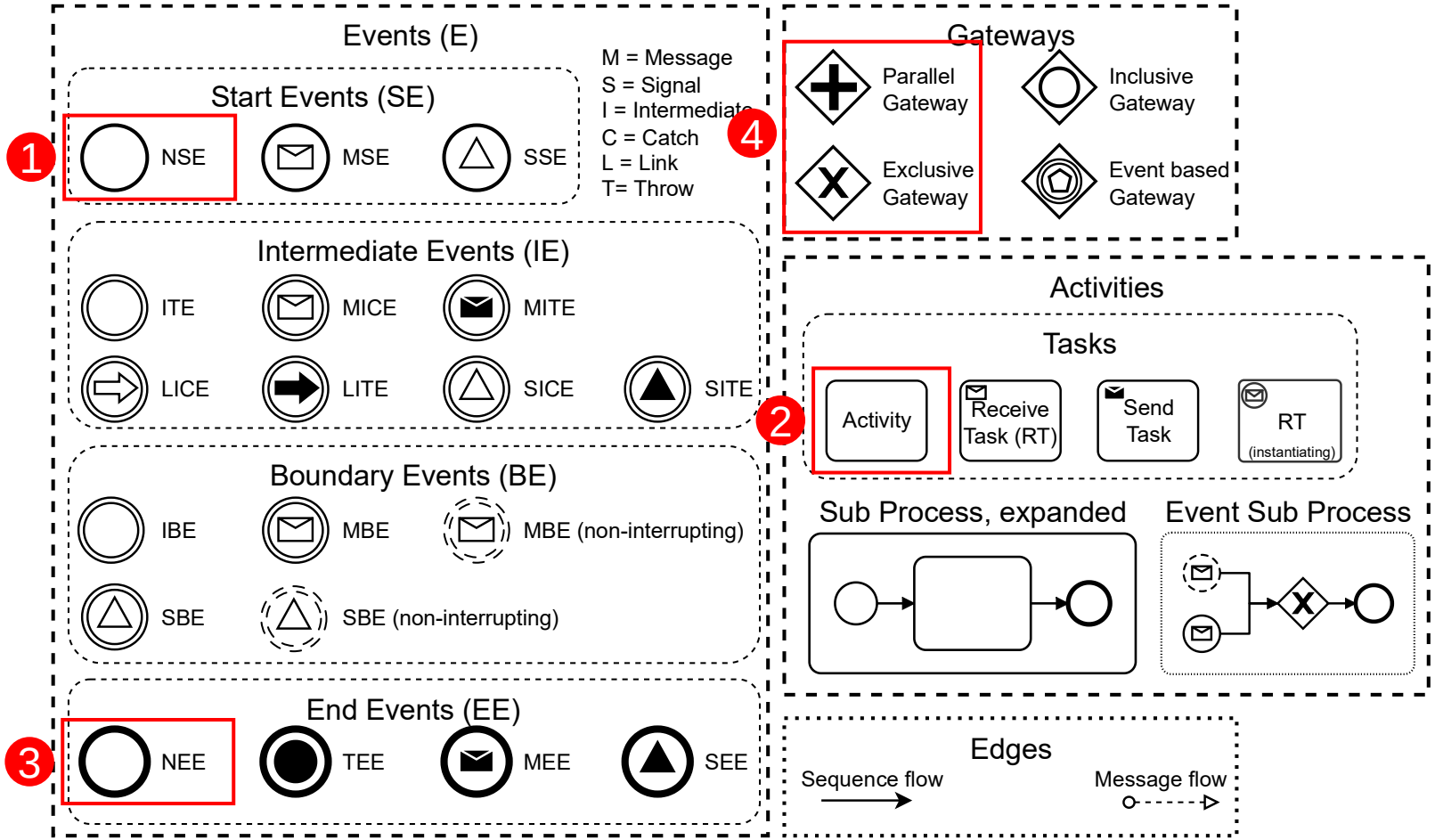


Equivalent rule in concrete syntax

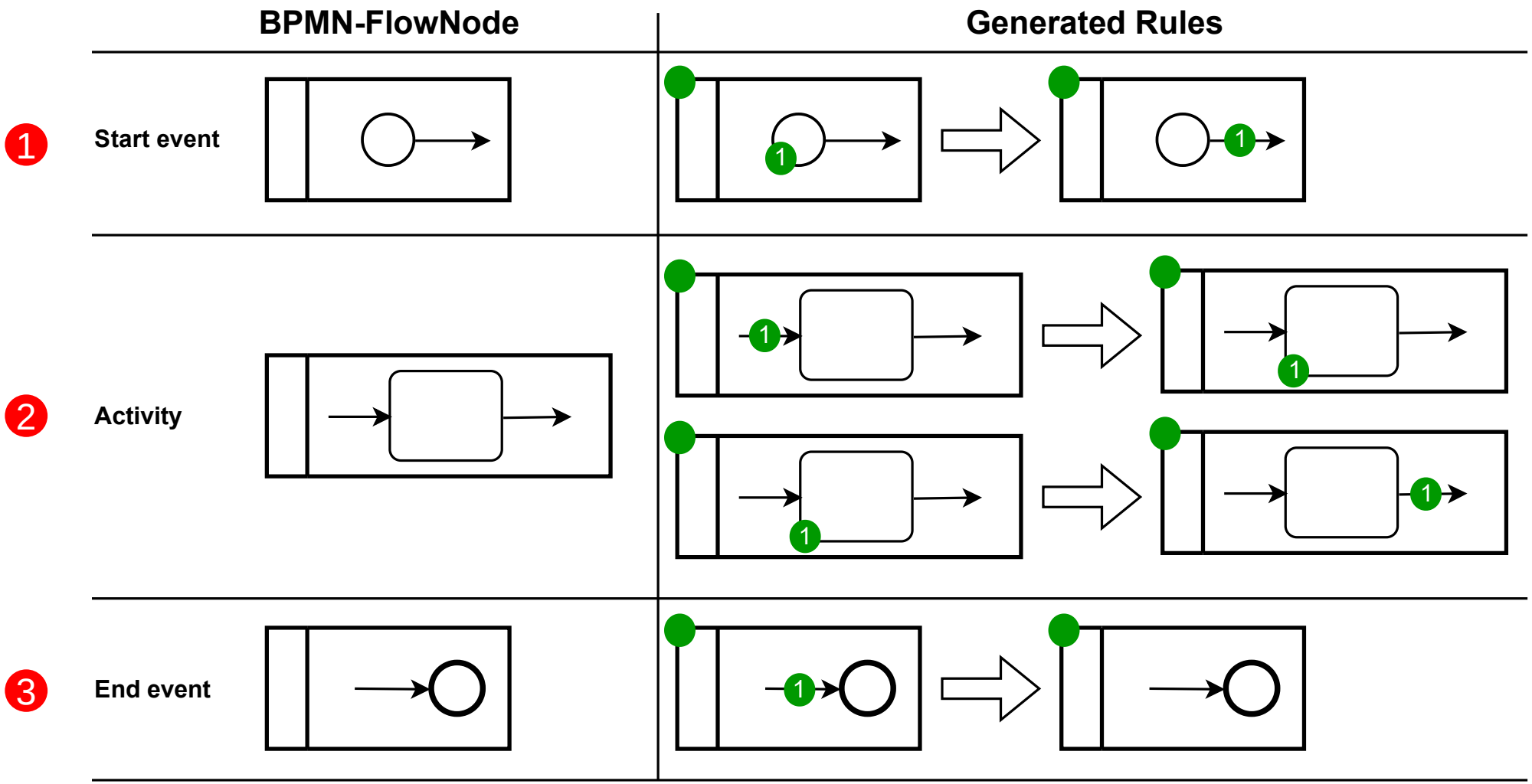


BPMN semantics formalization: GT-Rule generation I

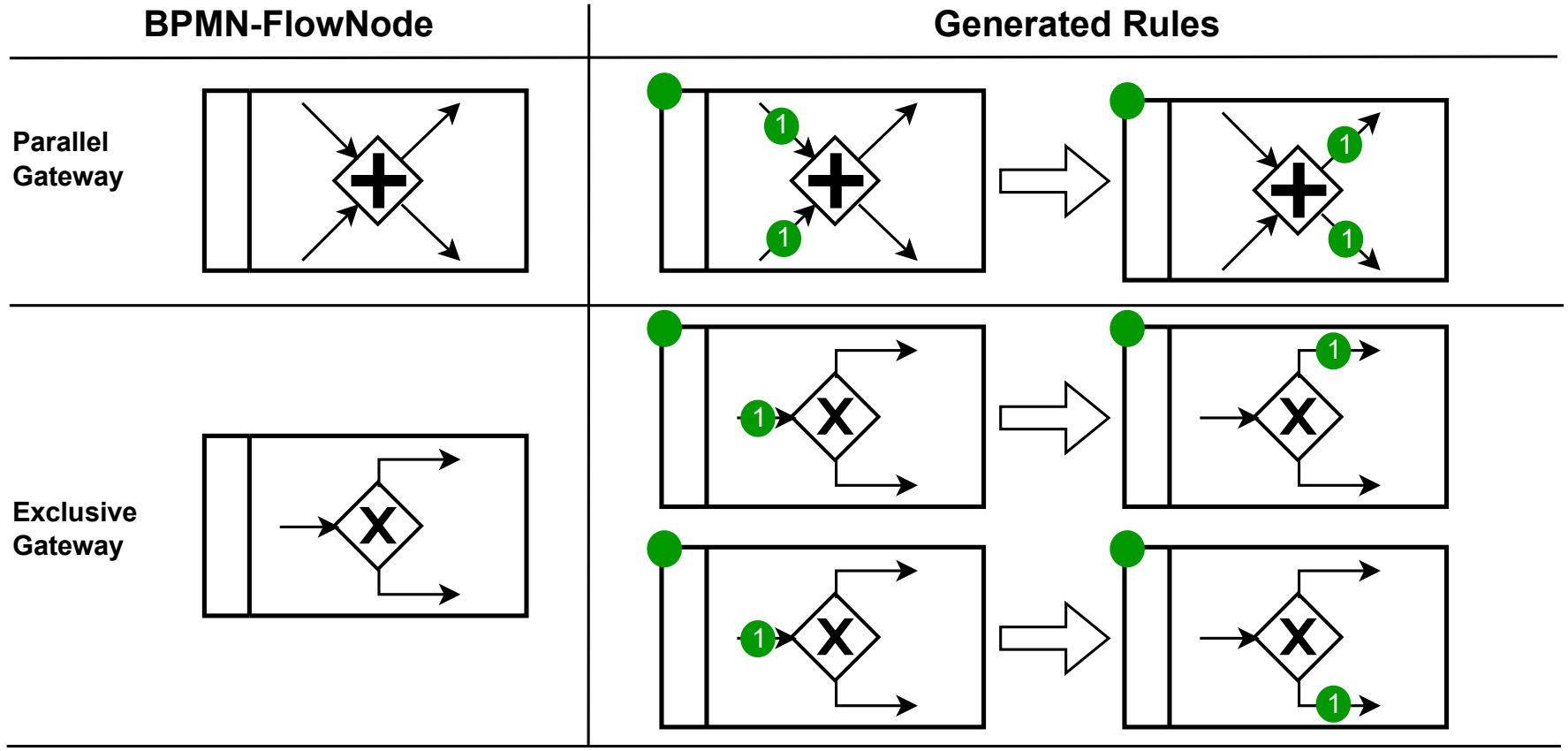
Each BPMN FlowNode is transformed to one or more GT-Rules.



BPMN semantics formalization: GT-Rule generation II



BPMN semantics formalization: GT-Rule generation III



4

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

Model checking BPMN I

Details

BPMN-specific properties [2]	
Safeness [2]	At most one token occurs along the same sequence flow.
Option to complete [2]	Any running process instance must eventually complete.
No dead activities [2]	Any activity can be executed in at least one process instance.

It is possible to check all properties using our approach!

Safeness	Check specific LTL properties on the GTS state space.
Option to complete	
No dead activities	Check activity executions in the GTS state space.

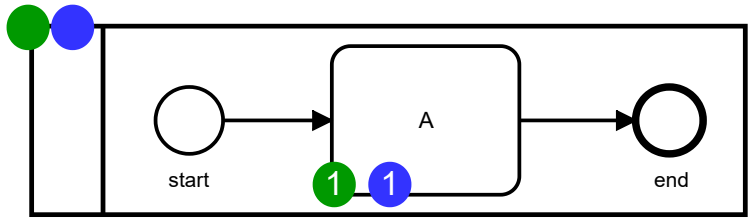
Model checking BPMN II

Custom properties

Domain-specific requirements can be encoded using custom properties.

1. Define atomic propositions.

Use the introduced concrete BPMN syntax.



2. Write a temporal property (for example LTL).

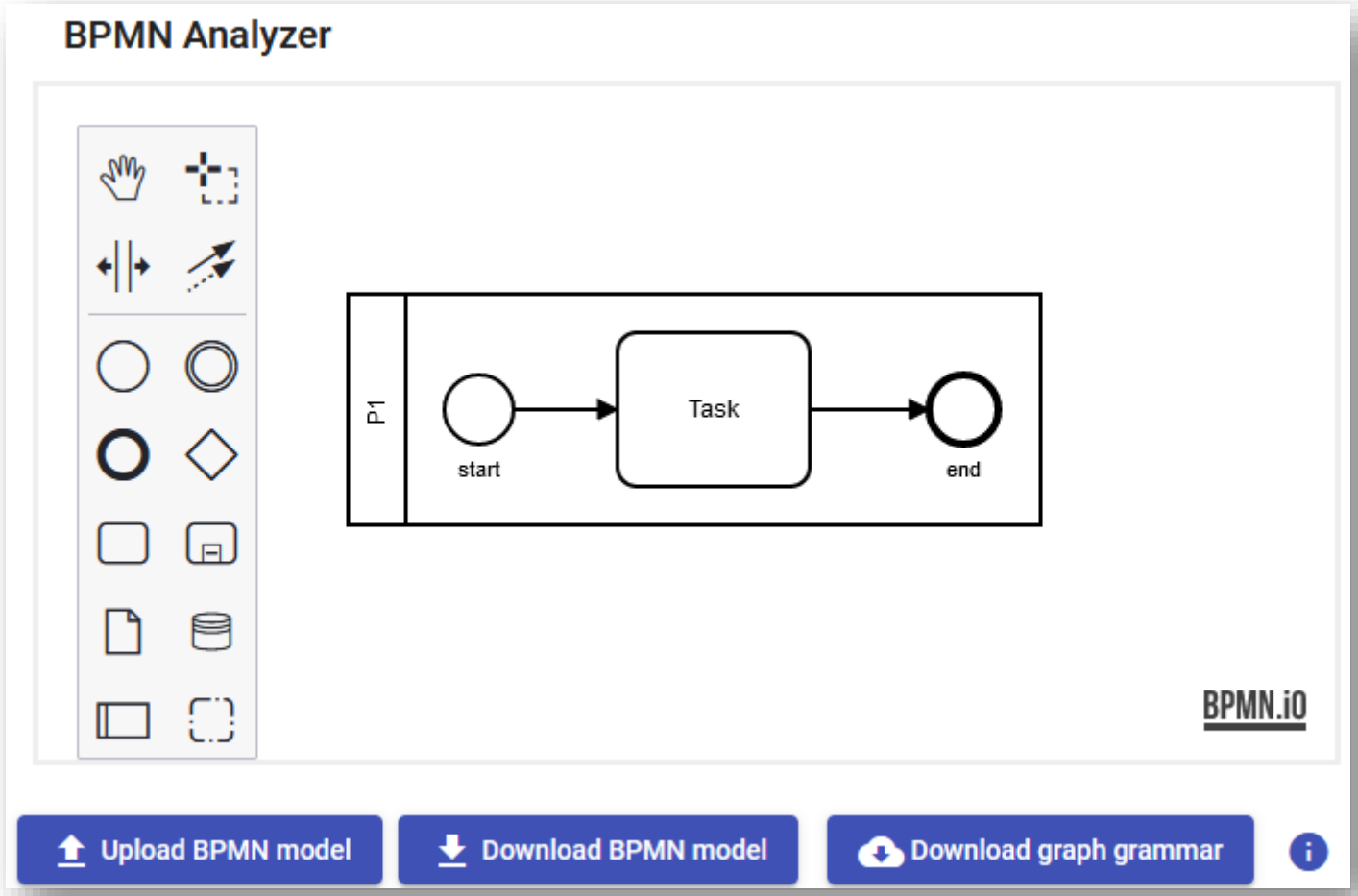
3. Check property on the GTS state space.

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

Implementation & Demonstration I

Web-based tool implementation started.



Upload, download and edit BPMN models.

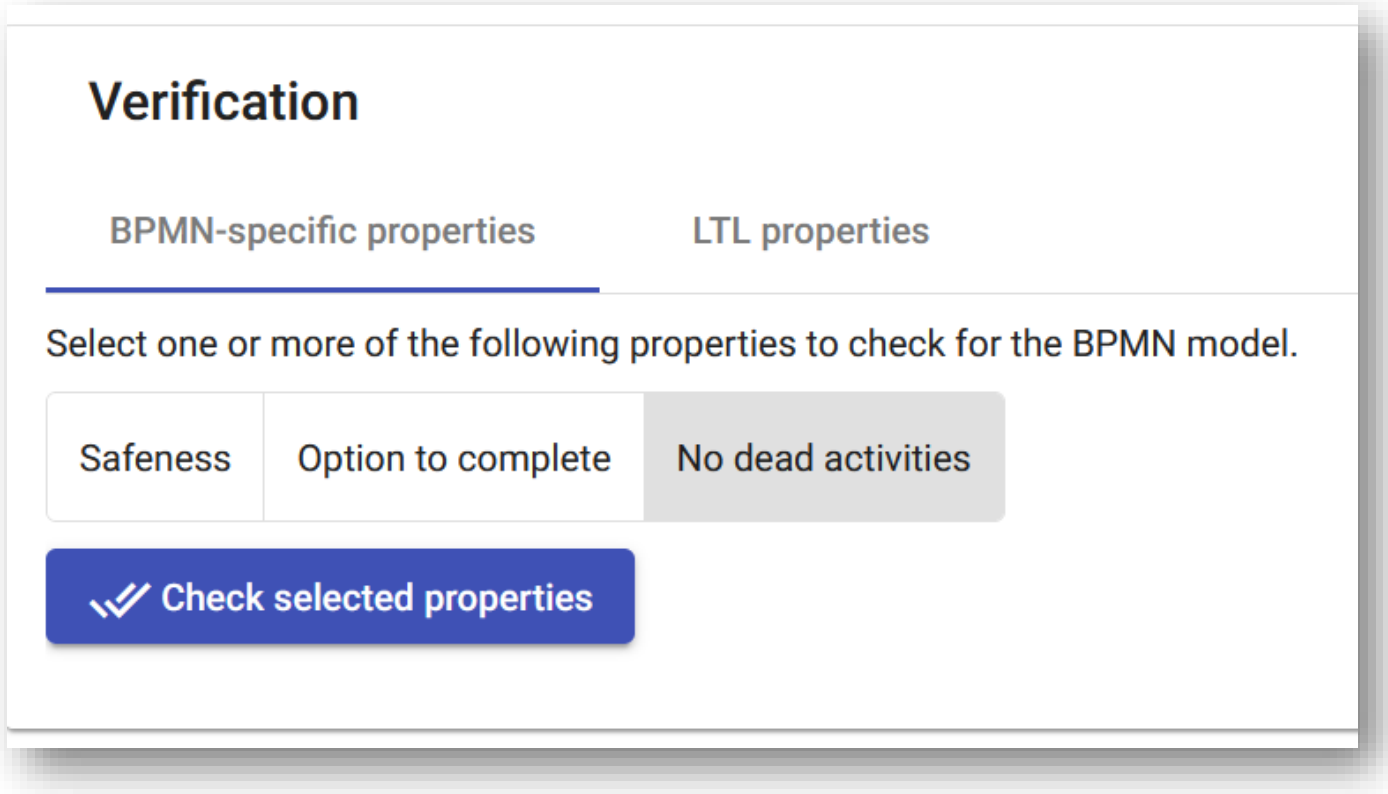
Generate GTS's from BPMN models.

GTS's executable in Groove.

Groove implements SPO with NAC's and nested Rules.

Implementation & Demonstration II

Model-checking/Verification implementation ongoing.



BPMN-specific properties

Safeness | Option to complete | No dead activities

LTL properties

Atomic propositions editor planned.

Dependent on the groove model checker.

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

Related work

Comparison of supported BPMN features

We uniquely support some features.

Support for more event types is planned.

Table 1: Constructs supported by different BPMN formalizations (overview based on [10]).

Feature	Van Gorp et al. [10]	Corradini et al. [1]	Houhou et al. [5]	This paper
Instantiation and termination				
Start event instantiation	X	X	X	X
Exclusive event-based gateway instantiation	X			X
Parallel event-based gateway instantiation				
Receive task instantiation				X
Normal process completion	X	X	X	X
Activities				
Activity	X	X	X	X
Subprocess	X	X	X	X
Ad-hoc subprocesses				
Loop activity	X			
Multiple instance activity				
Gateways				
Parallel gateway	X	X	X	X
Exclusive gateway	X	X	X	X
Inclusive gateway (split)	X	X	X	X
Inclusive gateway (merge)	X		X	X
Event-based gateway		X[1]	X	X
Complex gateway				
Events				
None Events	X	X	X	X
Message events	X	X	X	X
Timer Events			X	
Escalation Events				
Error Events (catch)	X			
Error Events (throw)	X			
Cancel Events	X			
Compensation Events	X			
Conditional Events				
Link Events	X			X
Signal Events	X			X
Multiple Events				
Terminate Events	X	X	X	X
Boundary Events	X[2]		X[3]	X
Event subprocess				X

Agenda

- Introduction
- Preliminaries
- BPMN semantics formalization
- Model checking BPMN
- Implementation & Demonstration
- Related work
- Conclusion & Future work

Conclusion

Model transformation from BPMN to graph transformation systems.

BPMN Formalization is comprehensive.

Model checking is supported.

Prototype implementation in a web-based tool.

Future work

Extend the formalization to support more BPMN features.

Evaluate our approach extensively.

Extend the implementation to include more model checking capabilities.