Declarative Process Mining for Software Processes: The RuM Toolkit and the Declare4Py Python Library

Anti Alman *University of Tartu*
Ivan Donadello *Free University of Bozen-Bolzano*
Fabrizio M. Maggi *Free University of Bozen-Bolzano*

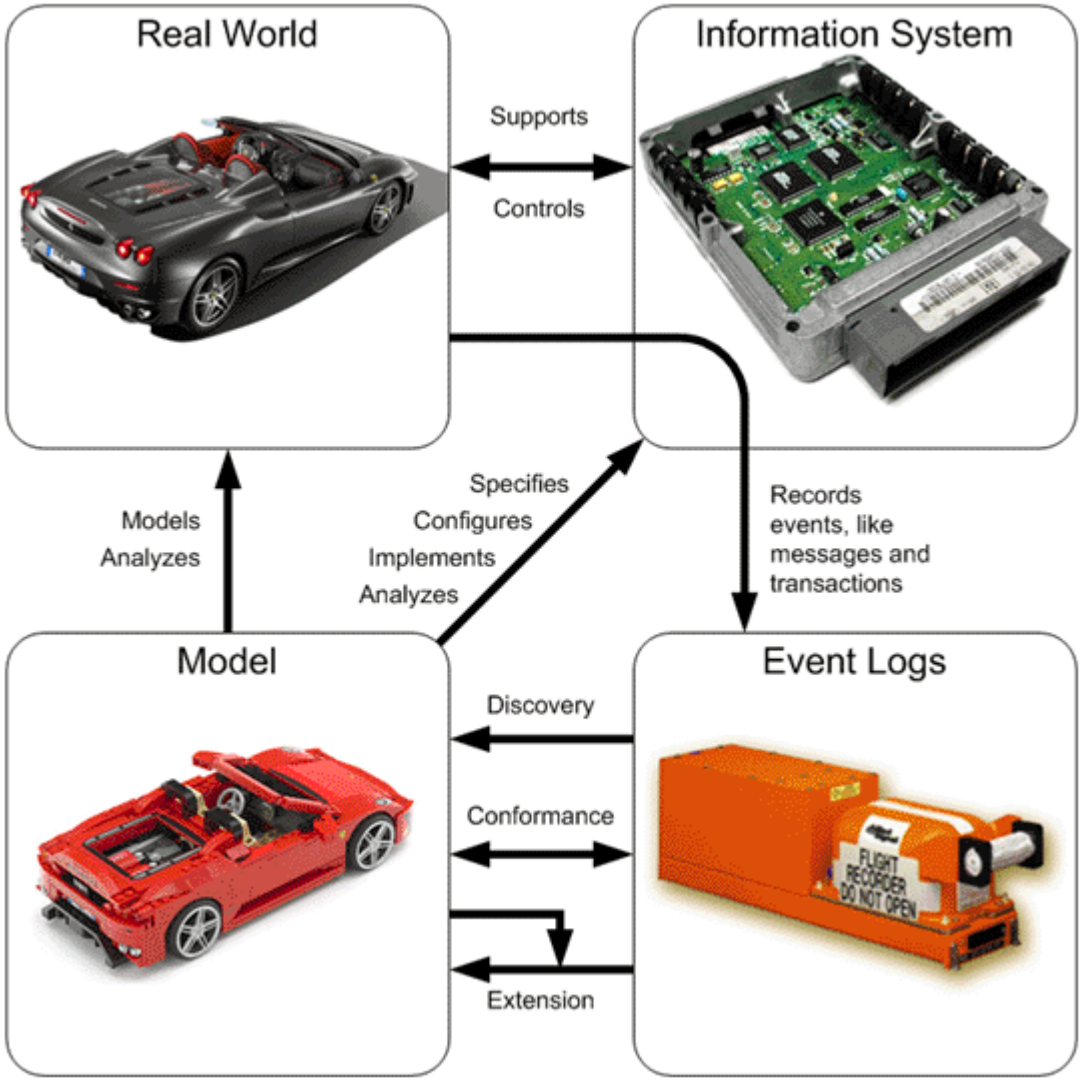
Declarative Process Mining for Software Processes: The RuM Toolkit and the Declare4Py Python Library

UNIFIED MODELING LANGUAGE™

Special Thanks to:
Pasquale Ardimento
University of Bari

Anti Alman *University of Tartu*
Ivan Donadello *Free University of Bozen-Bolzano*
Fabrizio M. Maggi *Free University of Bozen-Bolzano*

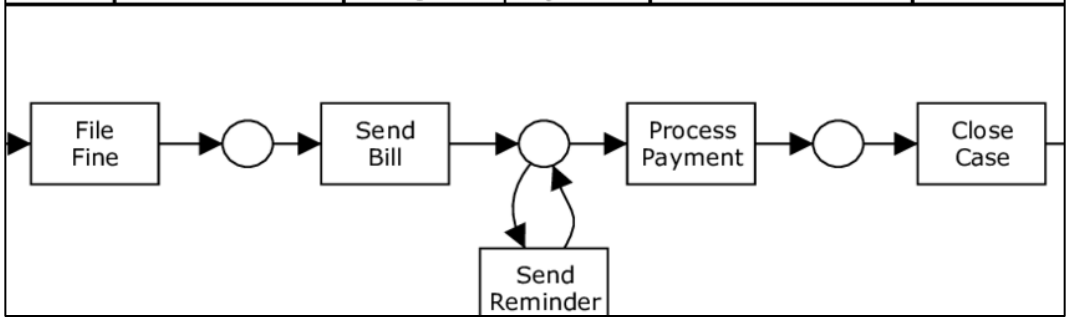
Process Mining



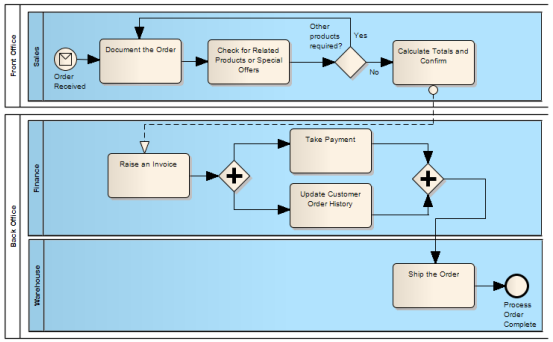
Wil M. P. van der Aalst: ***Process Mining - Data Science in Action, Second Edition***. Springer 2016, ISBN 978-3-662-49850-7

Event Logs

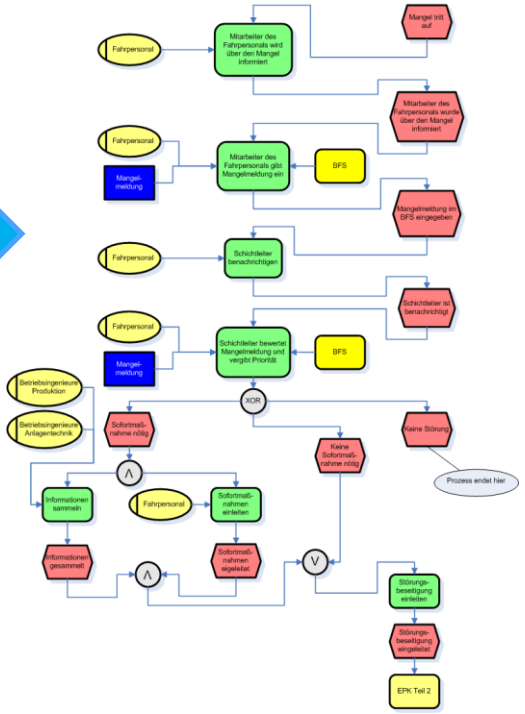
Case ID	Task Name	Event Type	Originator	Timestamp	Extra Data
1	File Fine	Completed	Anne	20-07-2004 14:00:00	...
2	File Fine	Completed	Anne	20-07-2004 15:00:00	...
1	Send Bill	Completed	system	20-07-2004 15:05:00	...
2	Send Bill	Completed	system	20-07-2004 15:07:00	...
3	File Fine	Completed	Anne	21-07-2004 10:00:00	...
3	Send Bill	Completed	system	21-07-2004 14:00:00	...
4	File Fine	Completed	Anne	22-07-2004 11:00:00	...
4	Send Bill	Completed	system	22-07-2004 11:10:00	...
1	Process Payment	Completed	system	24-07-2004 15:05:00	...
1	Close Case	Completed	system	24-07-2004 15:06:00	...
2	Send Reminder	Completed	Mary	20-08-2004 10:00:00	...
3	Send Reminder	Completed	John	21-08-2004 10:00:00	...
2	Process Payment	Completed	system	22-08-2004 09:05:00	...
2	Close case	Completed	system	22-08-2004 09:06:00	...
4	Send Reminder	Completed	John	22-08-2004 15:10:00	...
4	Send Reminder	Completed	Mary	22-08-2004 17:10:00	...
4	Process Payment	Completed	system	29-08-2004 14:01:00	...
4	Close Case	Completed	system	29-08-2004 17:30:00	...
3	Send Reminder	Completed	John	21-09-2004 10:00:00	...
3	Send Reminder	Completed	John	21-10-2004 10:00:00	...
3	Process Payment	Completed	system	25-10-2004 14:00:00	...
3	Close Case	Completed	system	25-10-2004 14:01:00	...

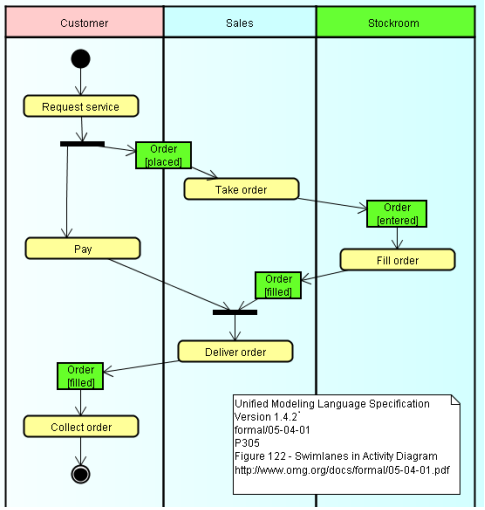


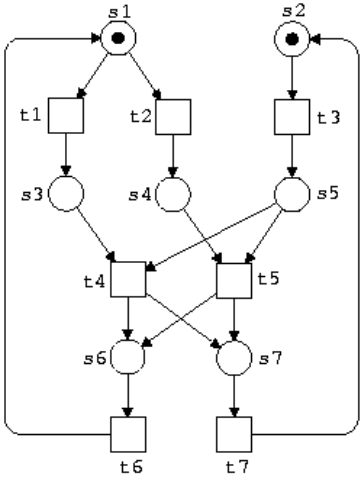
Procedural Process Models



Process Models







The Declarative Approach in a Nutshell

Describes a process in terms of

… *what* rules are followed by the process

Allows **flexibility** in deciding

… *how* these rules are fulfilled

For example: X and Y can only occur after A has occurred

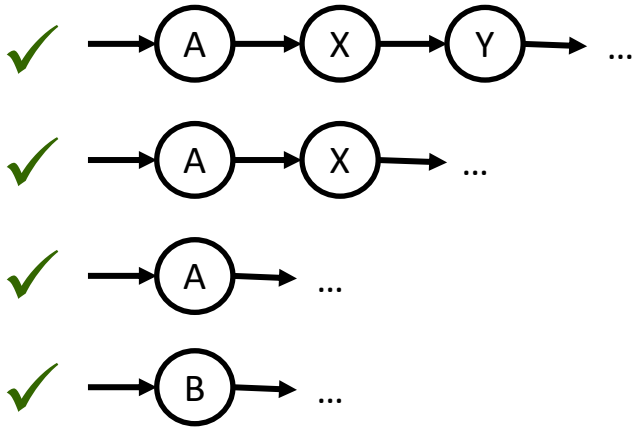
Note that the exact order and possibility of repetitions of X and Y is left open

The Declarative Approach in a Nutshell

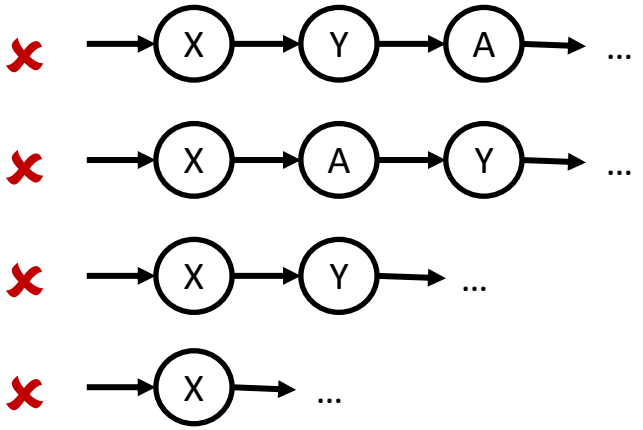
For example: X and Y can only occur after A has occurred

Note that the exact order and possibility of repetitions of X and Y is left open

Positive examples:



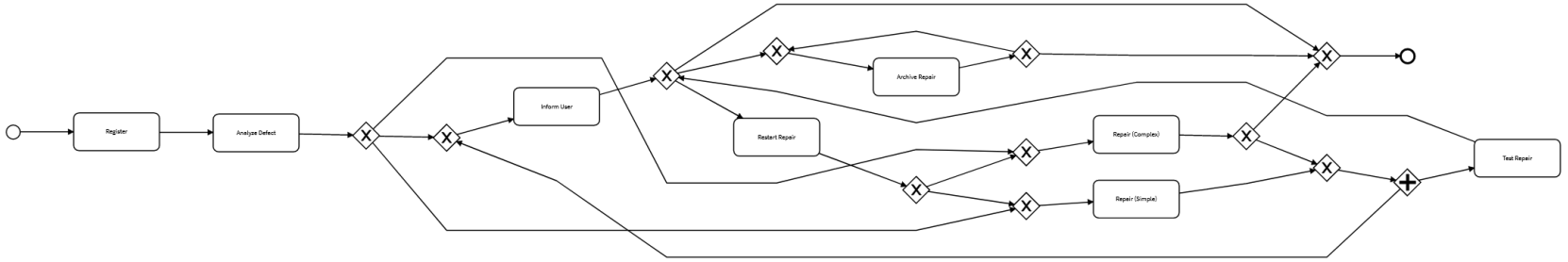
Negative examples:



Motivation for the Declarative Approach

Processes are different

- Some are simple and can be modeled both fully and also in an understandable way

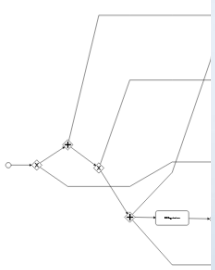


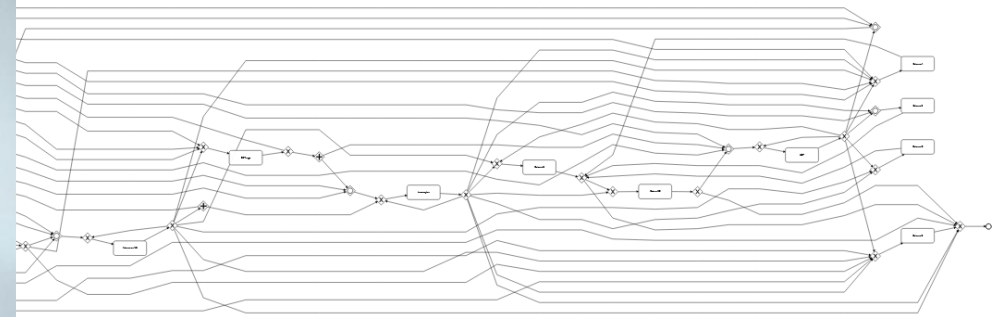
Motivation for the Declarative Approach

Business processes are different

- Some are simple and can be modeled both fully and also in an understandable way
- **And some are inherently too flexible to be modeled fully**
  - For example, treatment process of some disease
  - … combined with every potential set of allergies each individual patient may have
  - … and multiplied by different severity levels

Knowledge Intensive Processes!





Process Mining for Software

UNIFIED MODELING LANGUAGE™

Paradigms

Procedural (imperative) paradigm

- Suitable for predictable processes
  - limited number of exceptions and deviations
- Focus on **how** things must be done

Declarative paradigm

- Suitable for unpredictable processes
  - high number of exceptions and deviations
- Focus on **what** must be accomplished

Pasquale Ardimento, Mario Luca Bernardi, Marta Cimitile, Fabrizio Maria Maggi: ***Evaluating coding behavior in software development processes: a process mining approach.*** ICSSP 2019: 84-93

Pasquale Ardimento, Lerina Aversano, Mario Luca Bernardi, Vito Alessandro Carella, Marta Cimitile, Michele Scalera: ***UML Miner: a tool for mining UML diagrams***. MODELS-C 2023

Procedural Paradigm

Let's take a blank model

- *“Model” is in the box below*

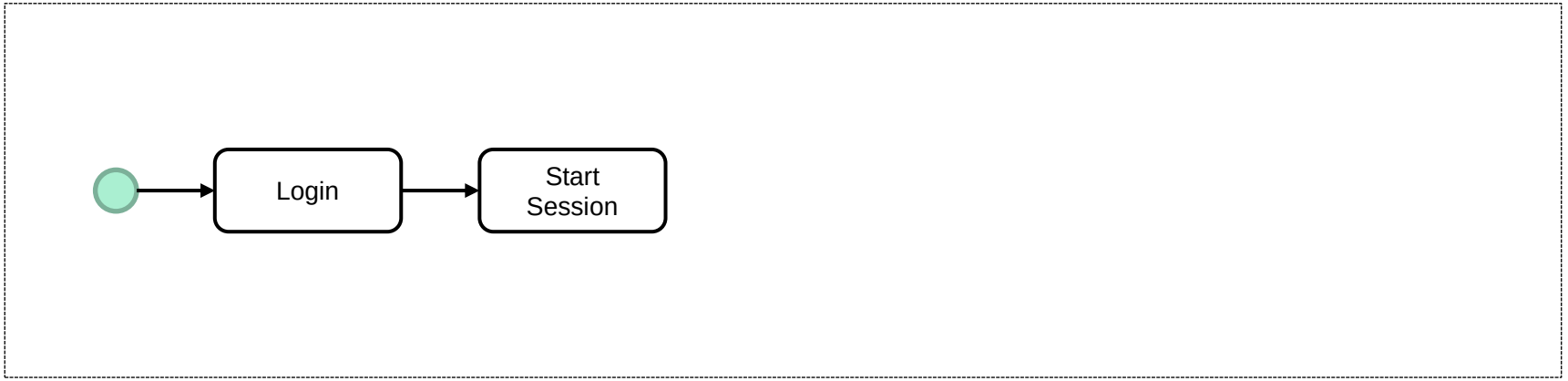
Nothing can happen

- As restrictive as possible
- Also, the model is invalid – we are even missing a place to start in

Procedural Paradigm

Let’s add something

- Login must occur
- Login must occur first
- Start Session occurs immediately after Login

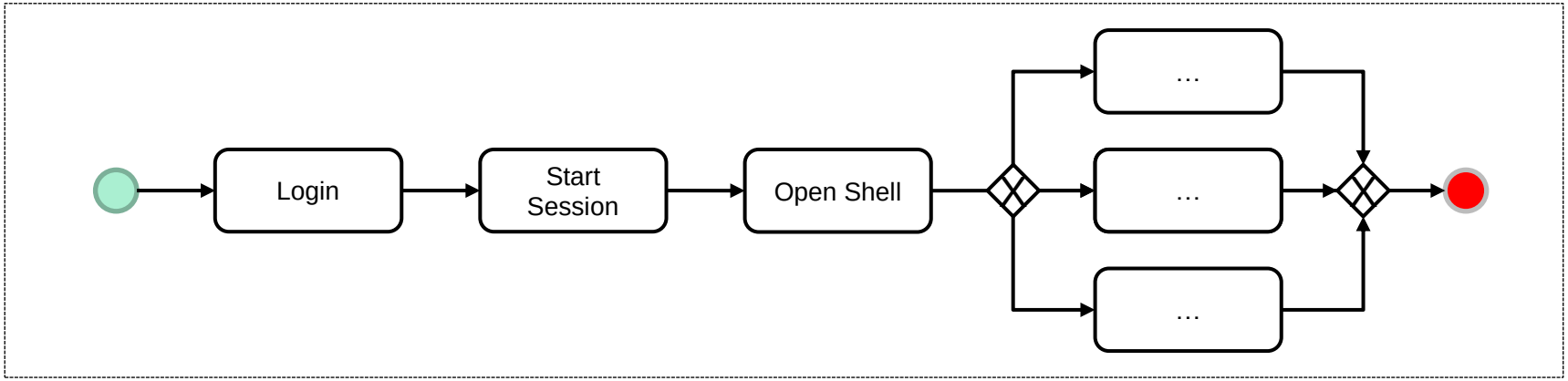


Procedural Paradigm

We need at least one full path from start to end

Supporting more process behaviour means adding more paths

- How to we cope with an ever-increasing model?
- Can we avoid overspecifying the process?
- Do we also add a path for something that occurs once in a million executions?



Theory vs. Practice

In theory, a procedural model will include all allowed execution paths

… while **also excluding all prohibited execution paths**

In practice, we very rarely follow the theory

… we might represent only the main flows

… we might exclude rare execution paths

… we might make simplifications even in the main flow

omitting intricacies of each decision, upper and lower bounds on repetition, etc.

Declarative Paradigm

Let’s start again with a blank model

- *“Model” is in the box below*

The crucial difference – everything can happen

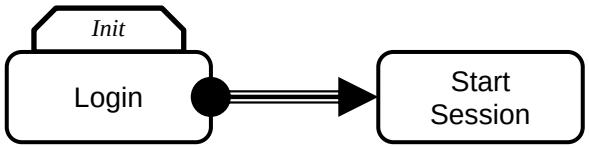
- As flexible as possible
- And could be seen as a valid model, although not a very useful one

15

Declarative Paradigm

Let’s add something

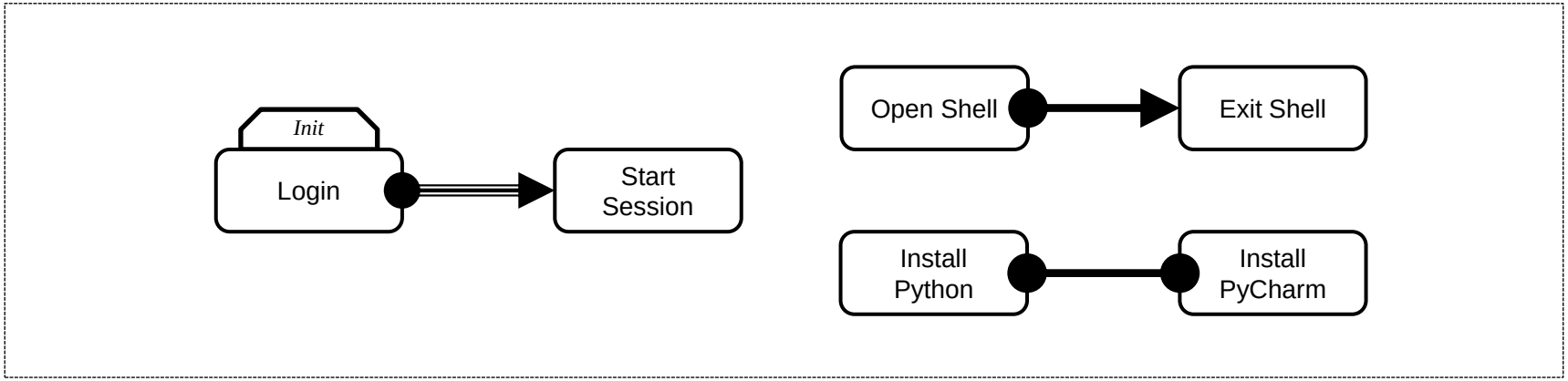
- Login can occur
- Login must occur first
- Start Session occurs immediately after Login -



Declarative Paradigm

But we are not required to go in the execution order

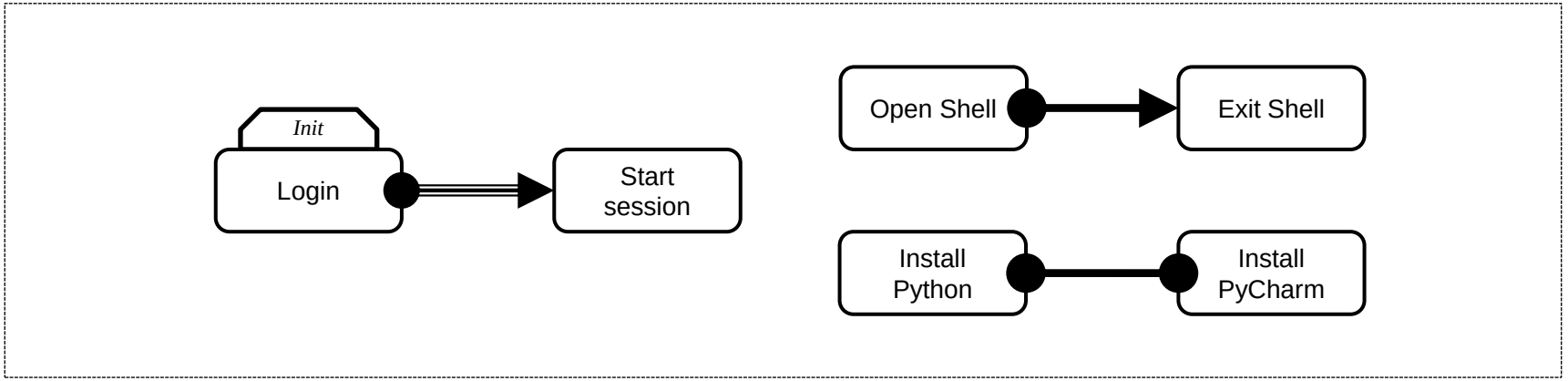
- If Open Shell occurs, then it is eventually followed by Exit Shell
- If Install Python occurs, then Install PyCharm also occurs and vice versa



Declarative Paradigm

In fact, we can go in order of importance

- Describing the most important rules first, one by one, moving closer to the level of detail we wish to achieve
- And whatever is left undescribed, is up to the process workers to decide



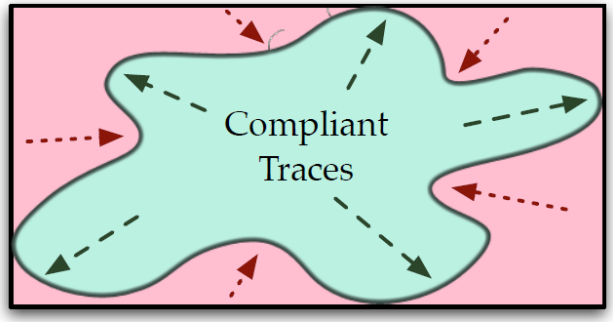
Closed World vs Open World

Closed world – Procedural

- If it is not in the model, then it is not allowed
- Always explicit control flow

Open world – Declarative

- If it does not contradict the model, then it is allowed
- Mostly implicit control flow
- Sometimes tricky to understand



Declare Process Modeling Language

One of the earliest and most popular declarative languages

A Declare model consist of constraints

- Each constraint defines some important aspect of the process

A constraint consists of…

- Template – The semantic meaning of the constraint
- Activity reference(s) – Activity or activities to which this meaning applies

An example of a simple constraint

- Template – The activity must exist
- Activity reference – ‘Login’

Maja Pesic, Helen Schonenberg, Wil M. P. van der Aalst: ***DECLARE: Full Support for Loosely-Structured Processes***. EDOC 2007: 287-300

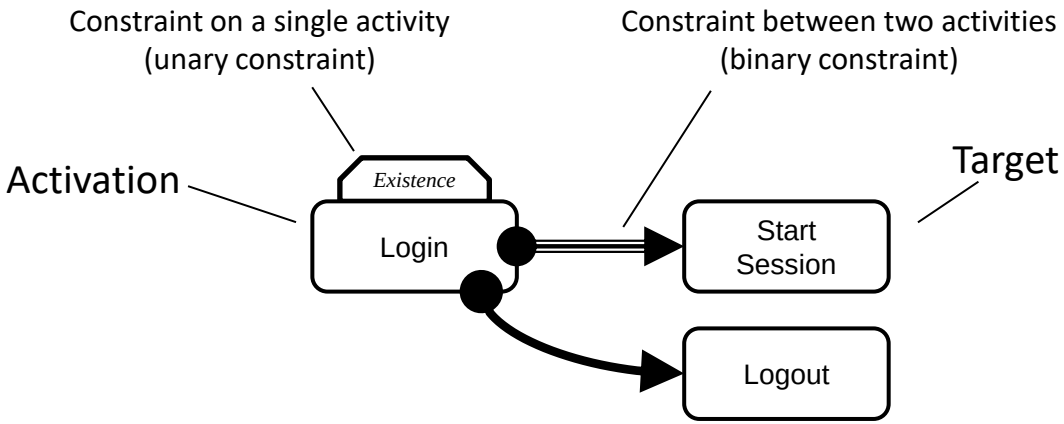
Visual Notation of Declare

Graph-based like BPMN, **but not a flowchart**

- Already used in previous parts of the tutorial

Activity nodes are not duplicated

- For example, if we also had: ‘Login’ is eventually followed by ‘Logout’



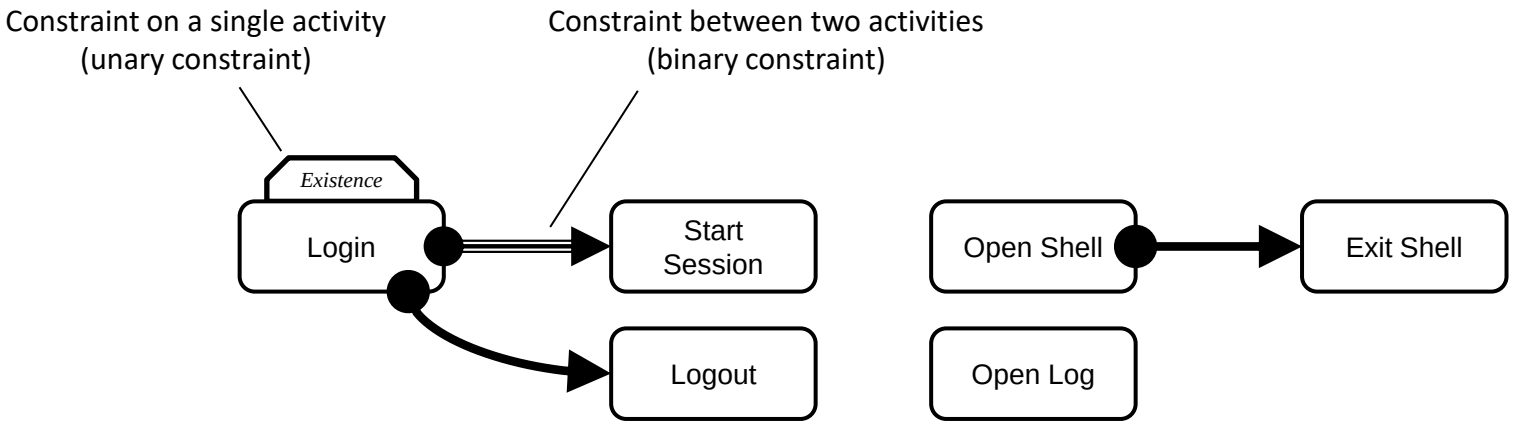
Visual Notation of Declare

Furthermore

- Model can be disconnected
- Activities can appear without constraints
  - Mostly for informative purposes, but can also be otherwise useful

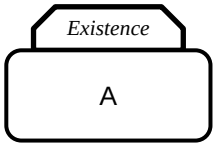
Vacuous Satisfaction

- When the constraint is not activated: *Login -> Start Session -> Logout*

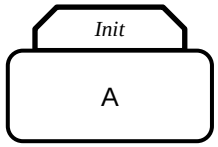


Unary Declare Templates

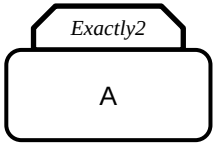
Used to define the cardinality or position of an activity



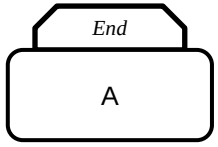
A occurs at least once



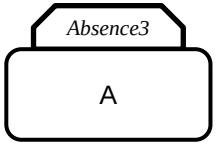
A occurs first



A occurs exactly twice



A occurs last

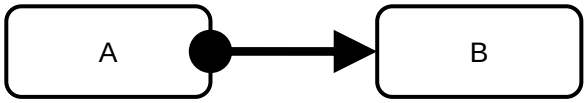


A occurs at most twice

Binary Declare Templates

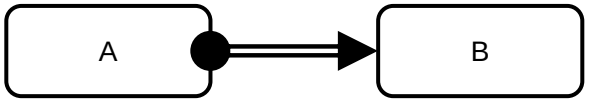
Used to define a relation between two activities

Response(A, B)



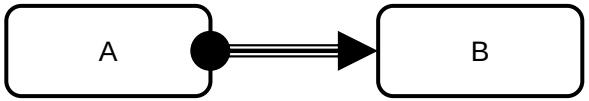
If A occurs, then B occurs after A

Alternate Response(A, B)



Each time A occurs, then B occurs afterwards before A recurs

Chain Response(A, B)

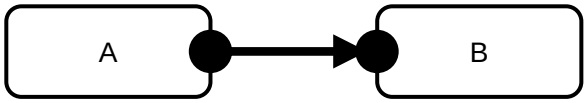


Each time A occurs, then B occurs immediately afterwards

Binary Declare Templates

Additional examples (non-exhaustive)

Succession (A, B)



A occurs if and only if it is followed by B
(Alternate and Chain Succession also possible)

Not Succession (A, B)

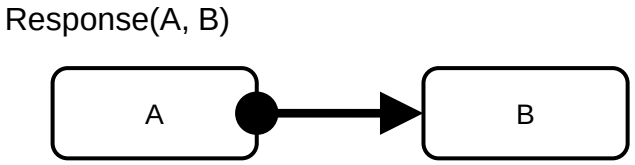
A can never occur before B
(All binary templates can be negated)

Choice (A, B)

A or B must occur at least once

Formal Semantics of Declare

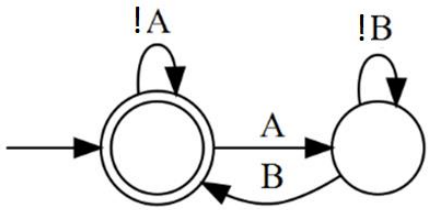
- Graphical representation



- Semantics specified through LTL (for finite traces)

$$\Box(A \Rightarrow \Diamond B)$$

- LTL rules can be translated into automata



Giuseppe De Giacomo, Moshe Y. Vardi: ***Linear Temporal Logic and Linear Dynamic Logic on Finite Traces***. IJCAI 2013: 854-860

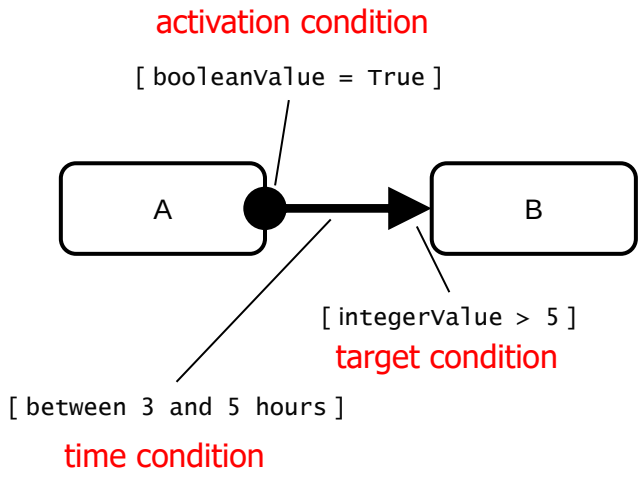
Multi-Perspective Declare (MP-Declare)

Data-Aware extension of Declare

Supports

- Conditions on event data
- Temporal conditions

Andrea Burattin, Fabrizio M. Maggi, Alessandro Sperduti Wil M. P. van der Aalst: ***Conformance checking based on multi-perspective declarative process models.*** Expert Syst. Appl. 65: 194-211 (2016)



If A occurs with ‘booleanValue = True’, then within 3 to 5 hours
B occurs with ‘integerValue > 5’ after A

Roadmap

Automated ***Process Discovery***

Compliance Monitoring

Conformance Checking

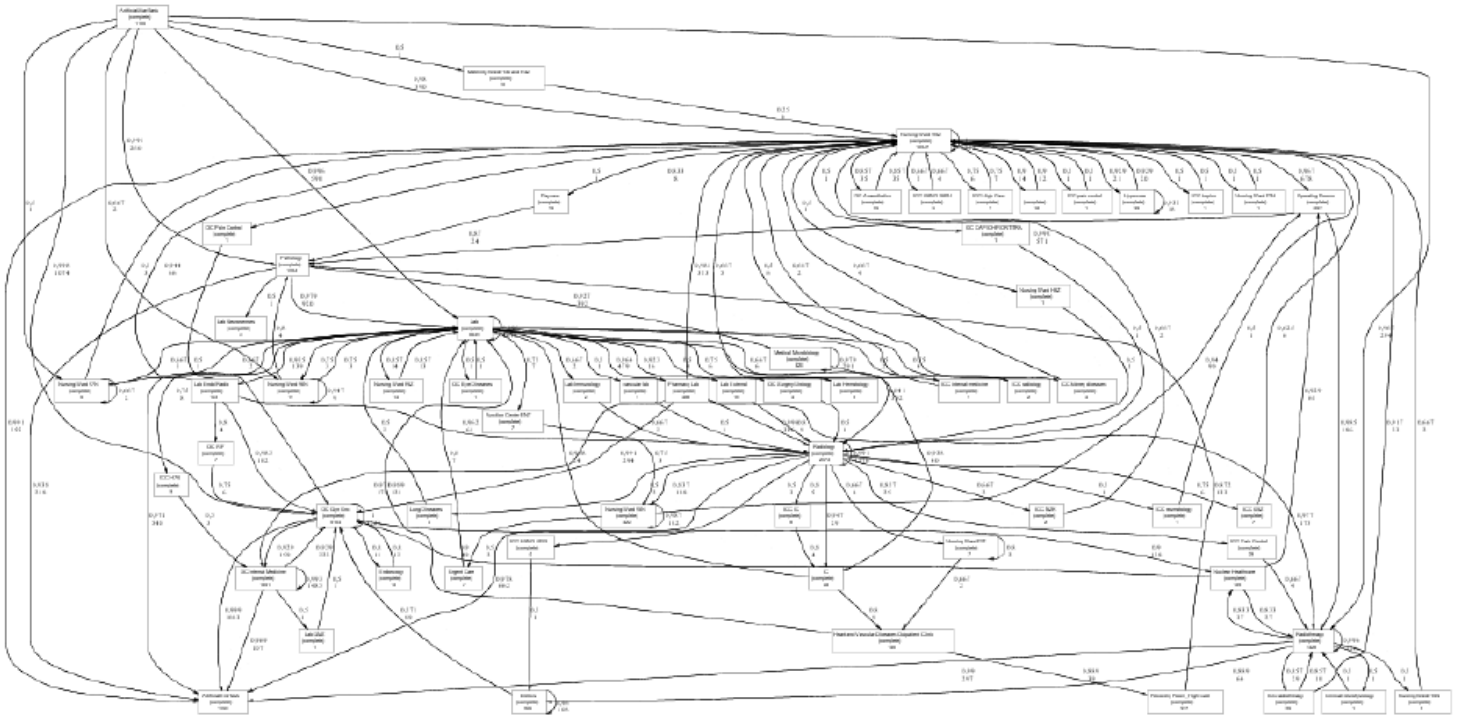
Log Generation

Process Discovery from knowledge intensive data

Process Discovery from knowledge intensive data

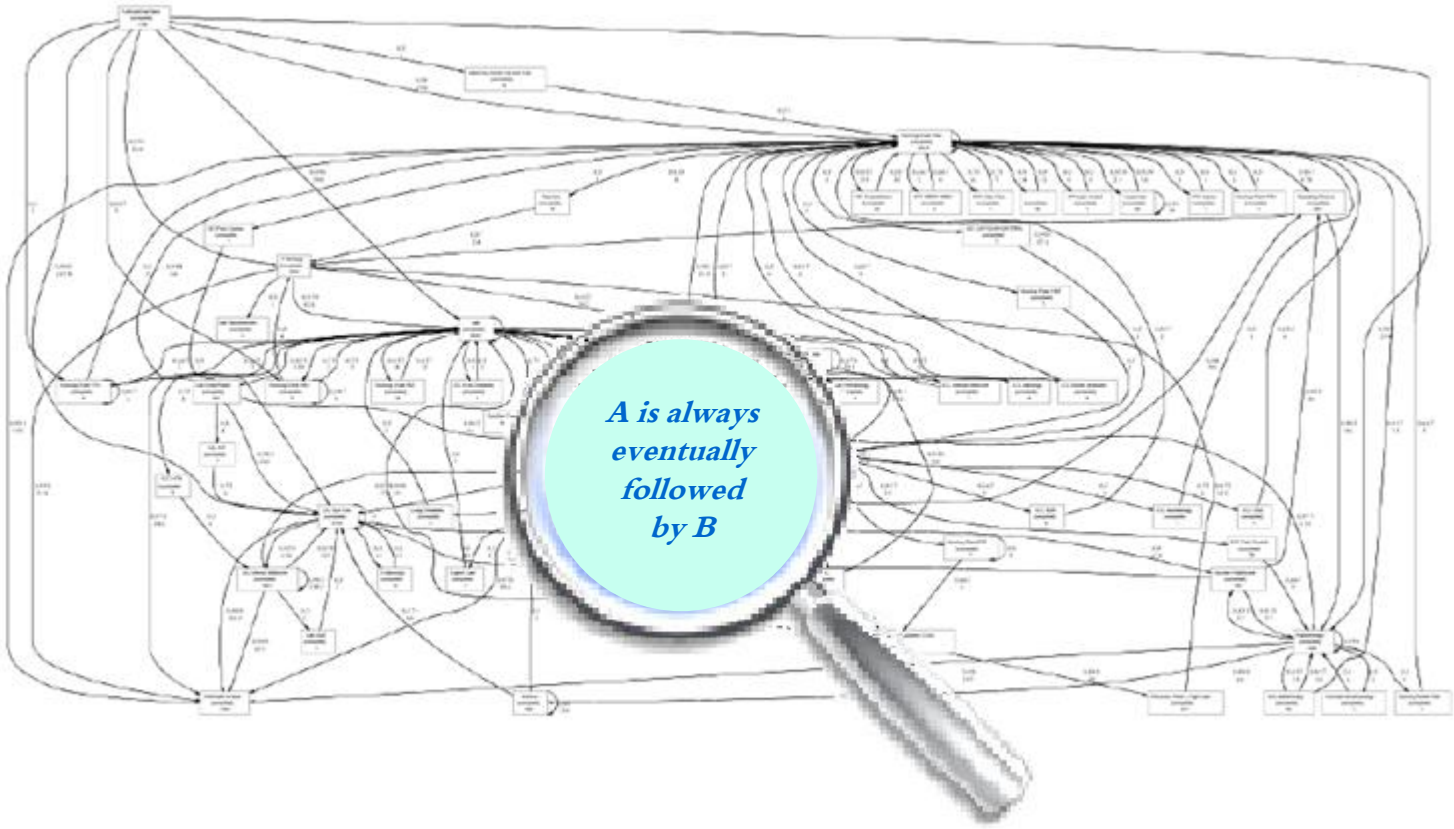
Process Discovery from knowledge intensive data

Procedural Process Discovery

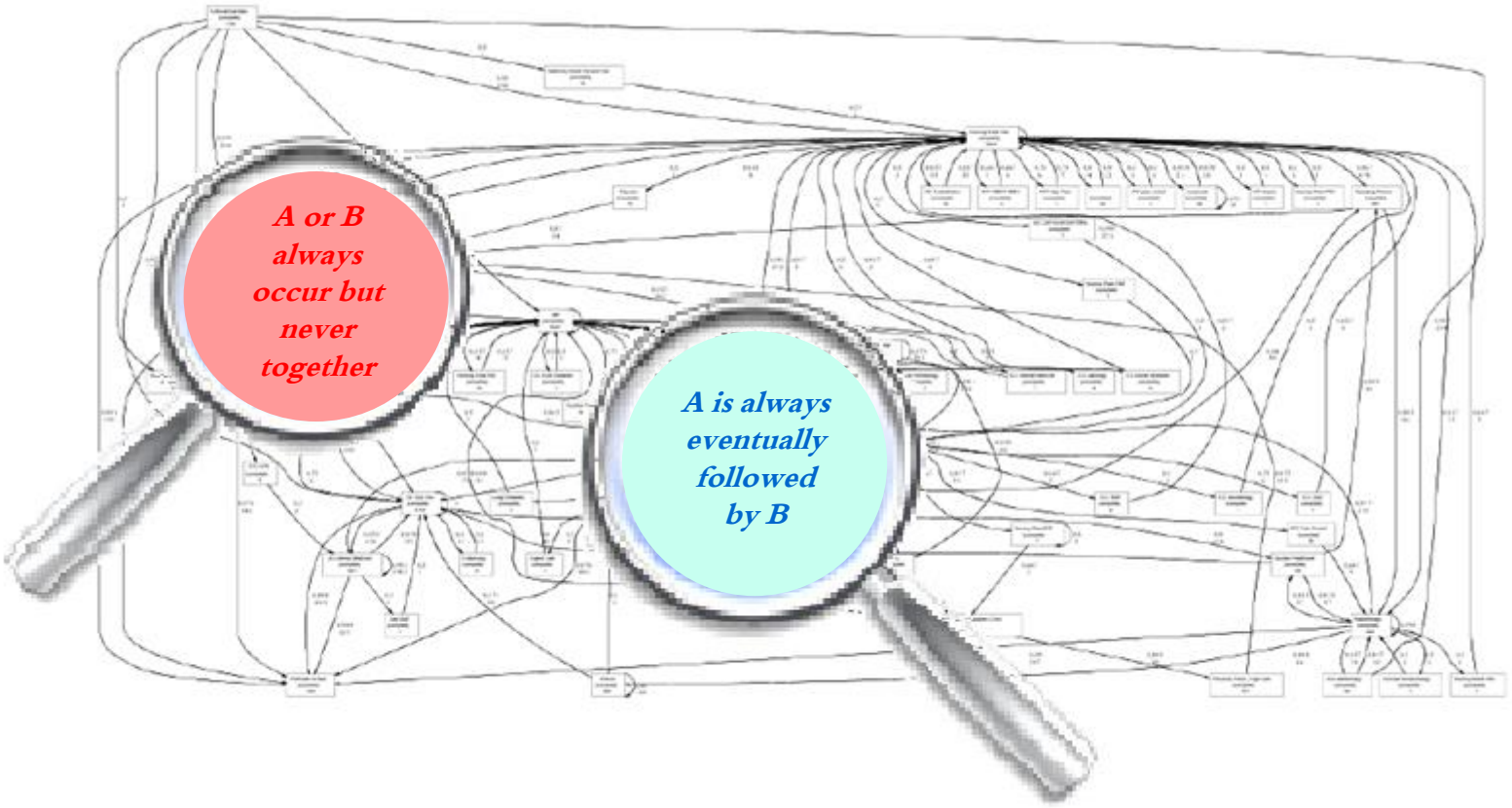


Procedural Process Discovery

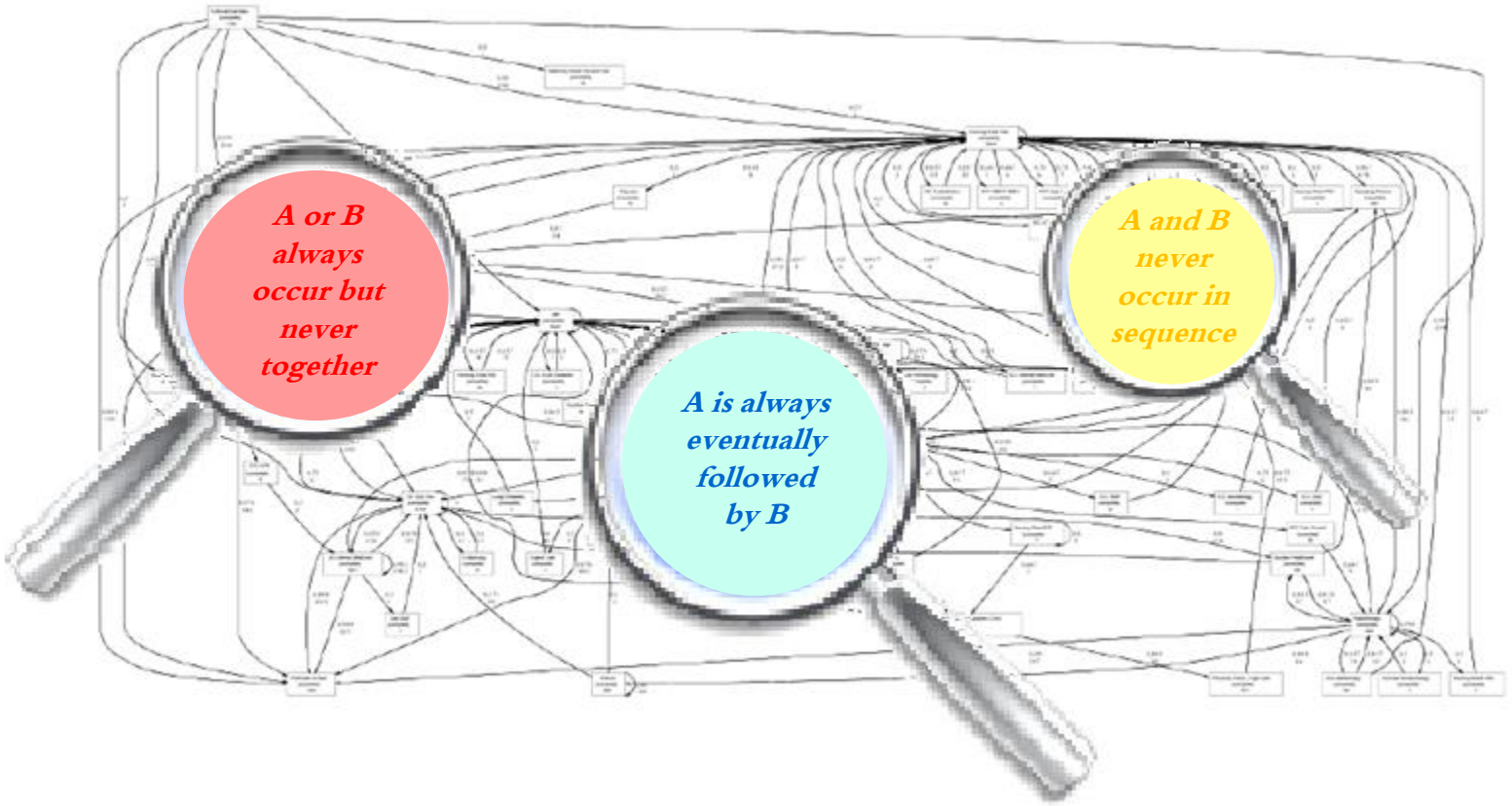
Declarative Process Discovery



Declarative Process Discovery

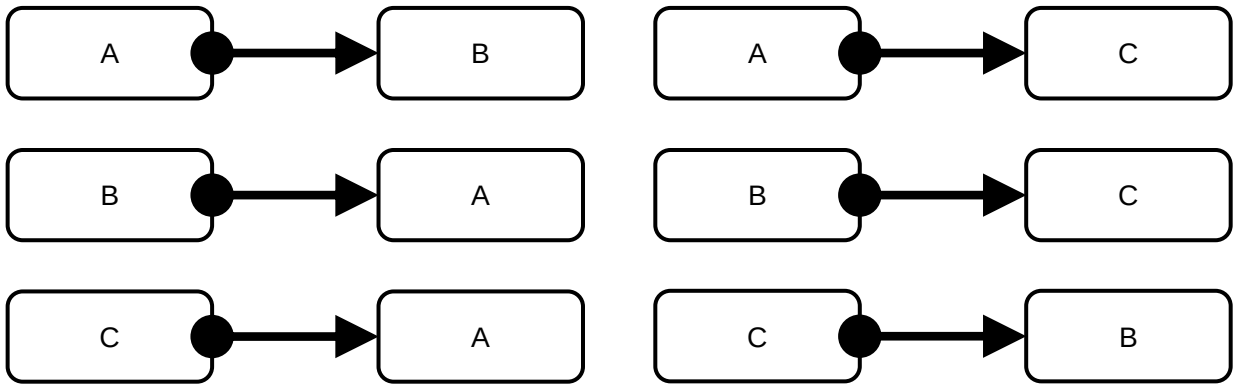


Declarative Process Discovery



Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]

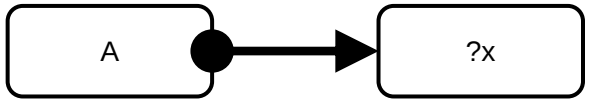


- finite number of constraint types
- finite set of activities

Fabrizio M. Maggi, R. P. Jagadeesh Chandra Bose, Wil M. P. van der Aalst: ***Efficient Discovery of Understandable Declarative Process Models from Event Logs***. CAiSE 2012: 270-285

A Parenthesis on Query Checking

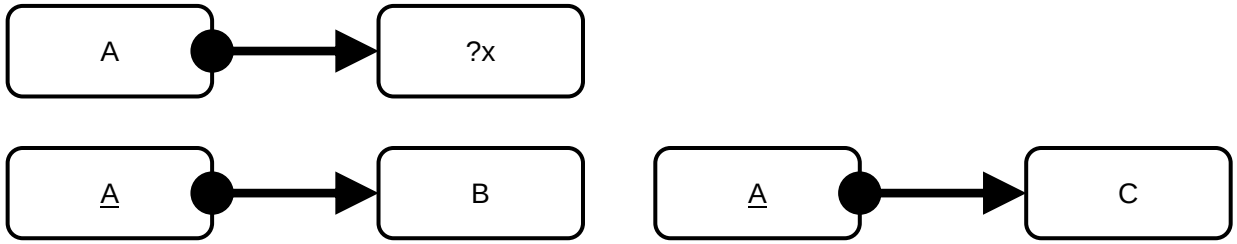
W = [<A C B C>, <C B A C>, <A C A C A C B>]



- finite number of constraint types
- finite set of activities

A Parenthesis on Query Checking

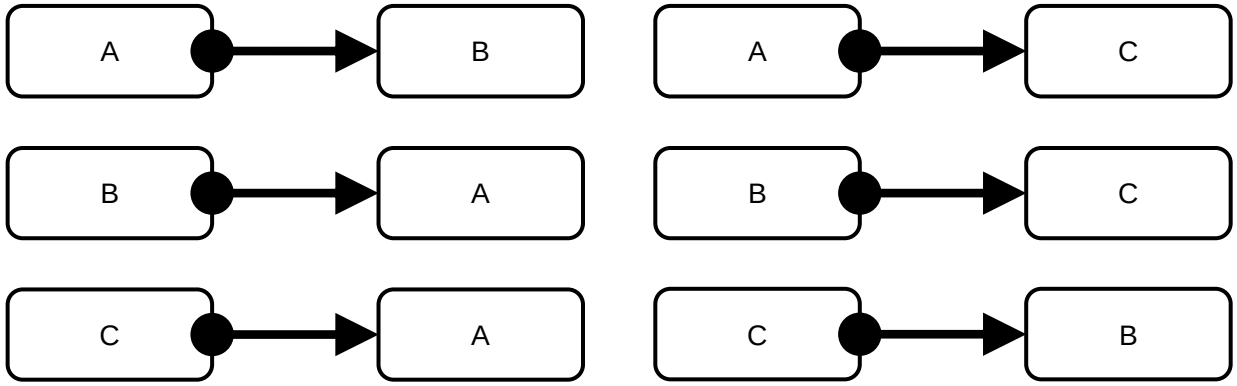
W = [<A C B C>, <C B A C>, <A C A C A C B>]



- finite number of constraint types
- finite set of activities

Query Checking

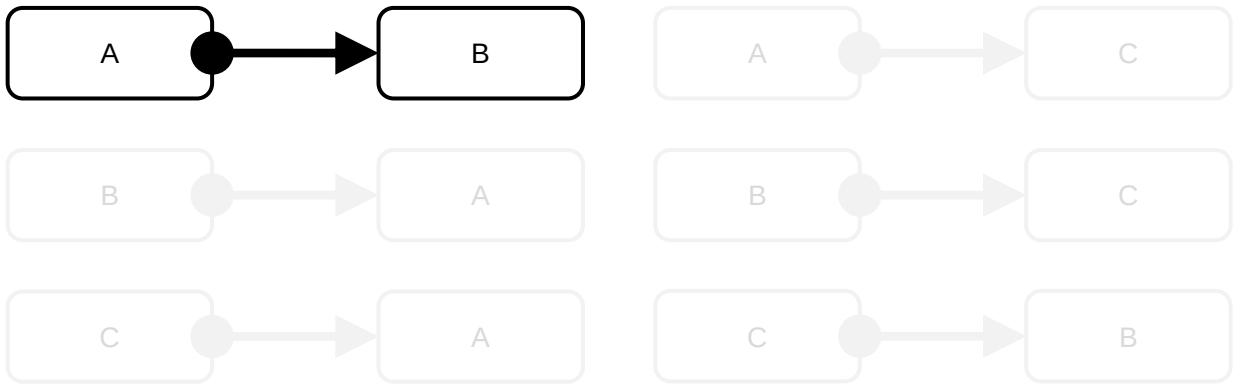
W = [<A C B C>, <C B A C>, <A C A C A C B>]



- finite number of constraint types
- finite set of activities

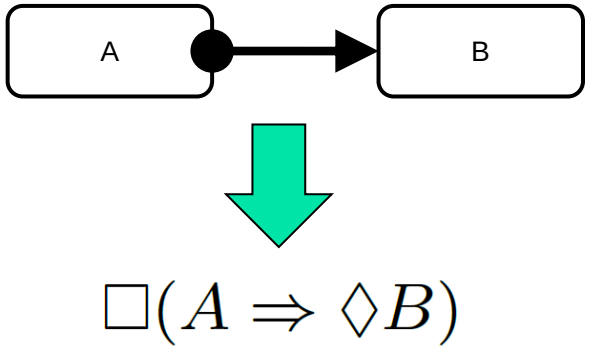
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



Discovery algorithm

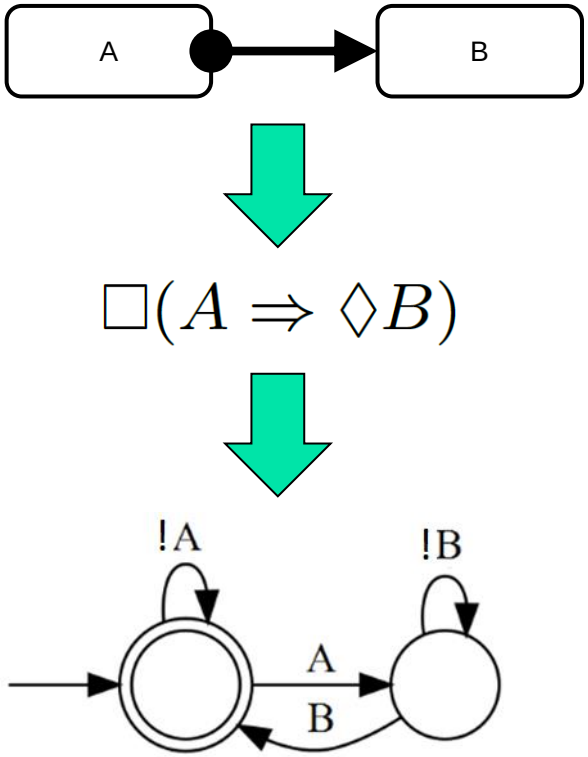
W = [<A C B C>, <C B A C>, <A C A C A C B>]



$$\Box(A \Rightarrow \Diamond B)$$

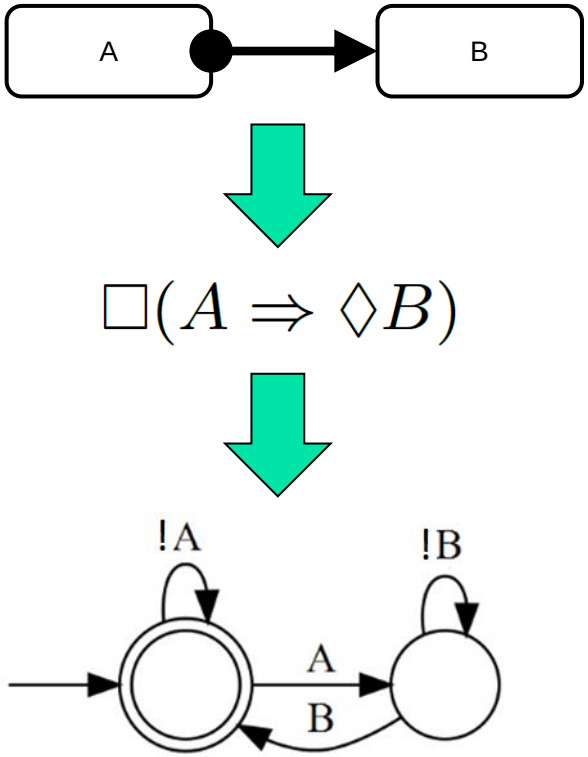
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



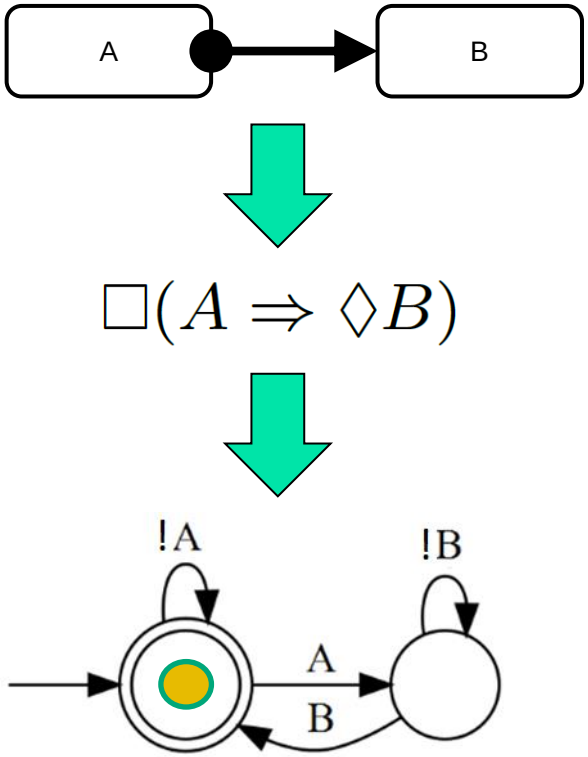
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



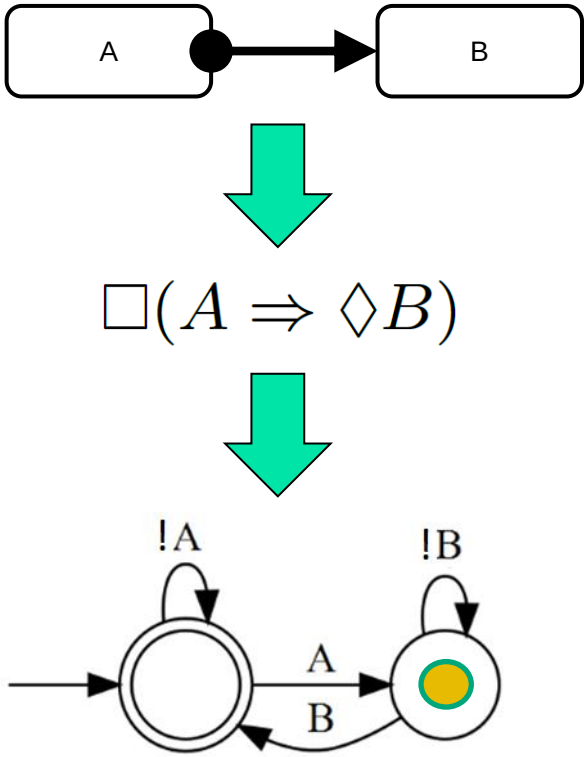
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



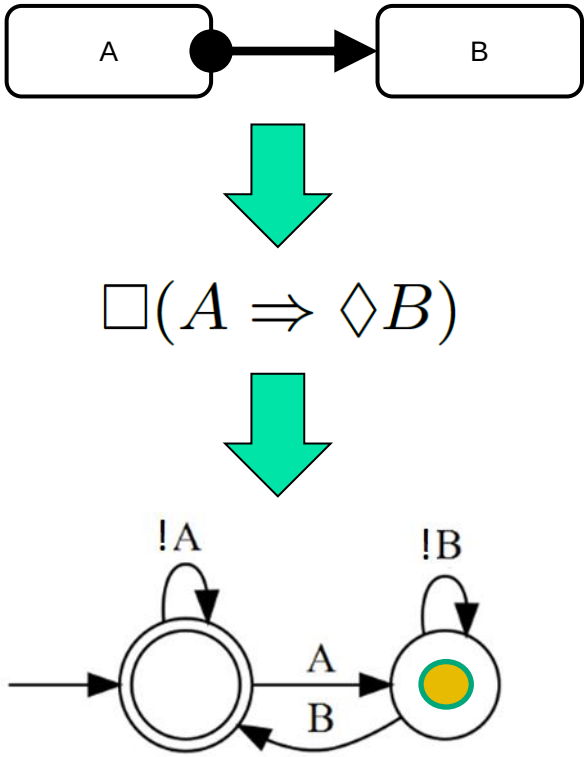
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



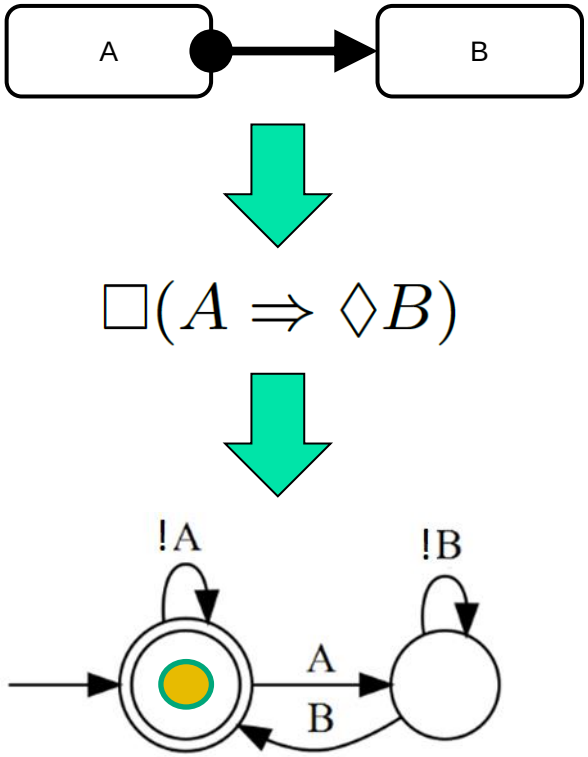
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



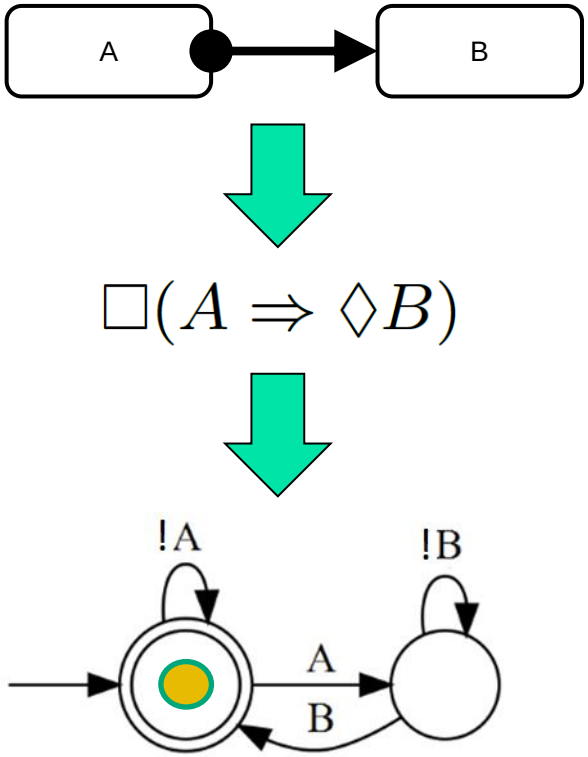
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



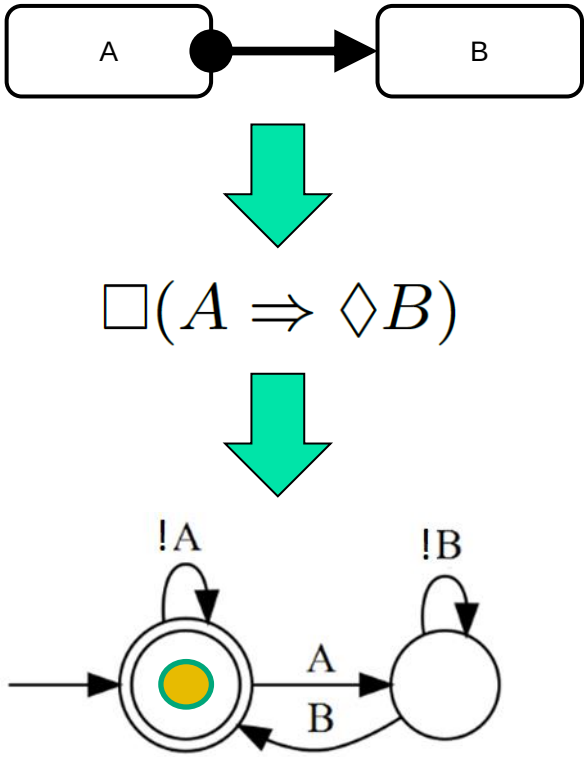
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



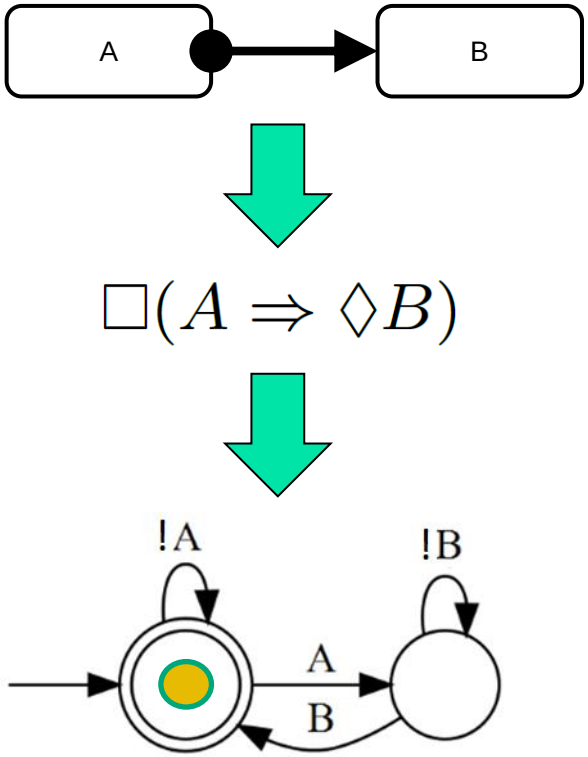
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



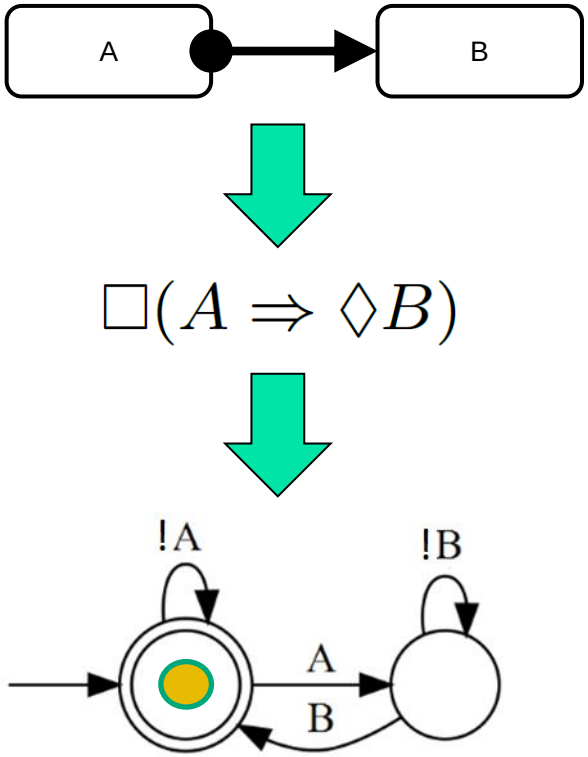
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



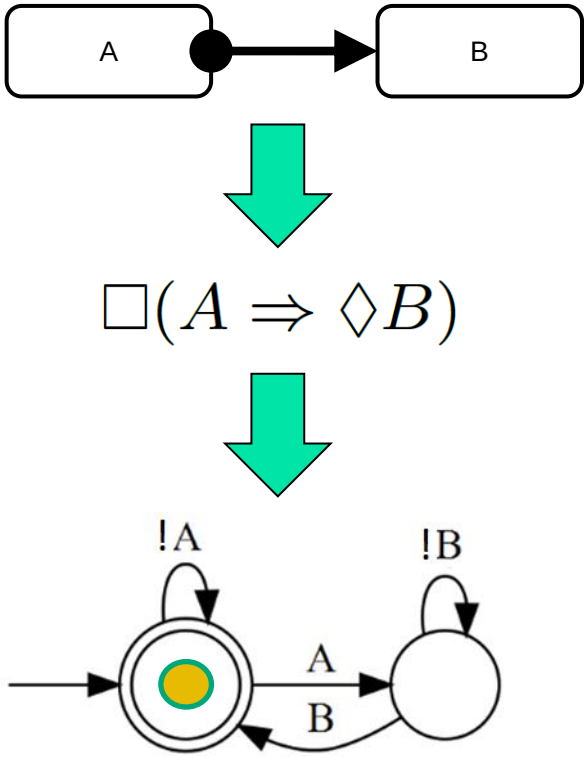
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



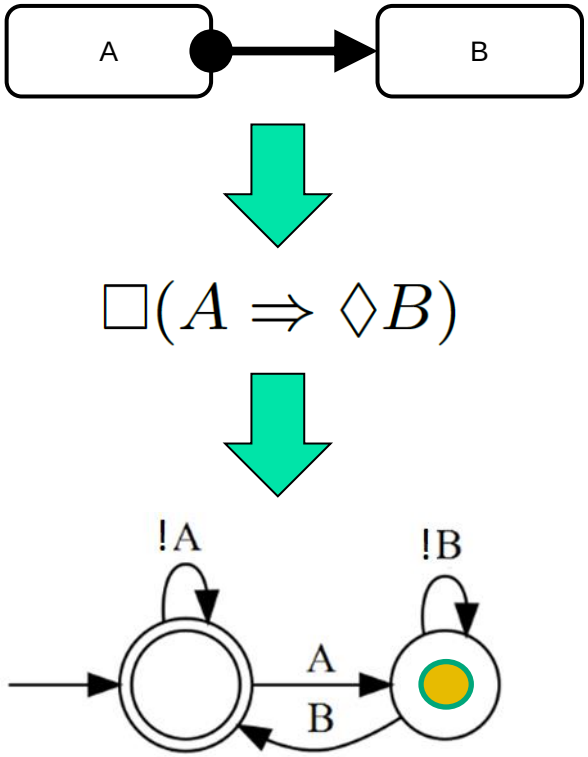
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



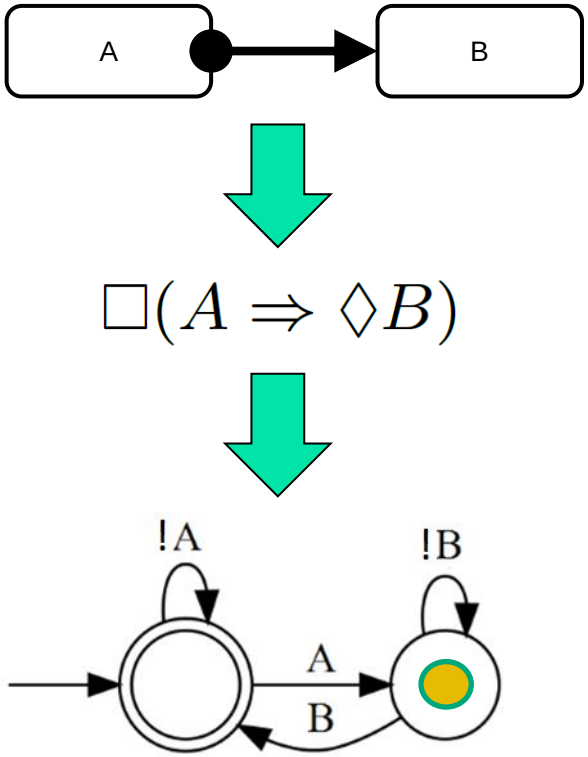
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



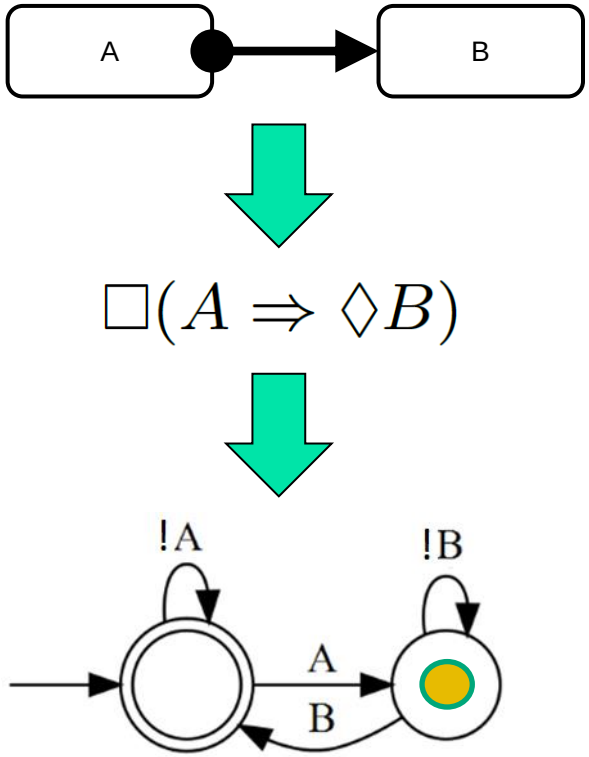
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



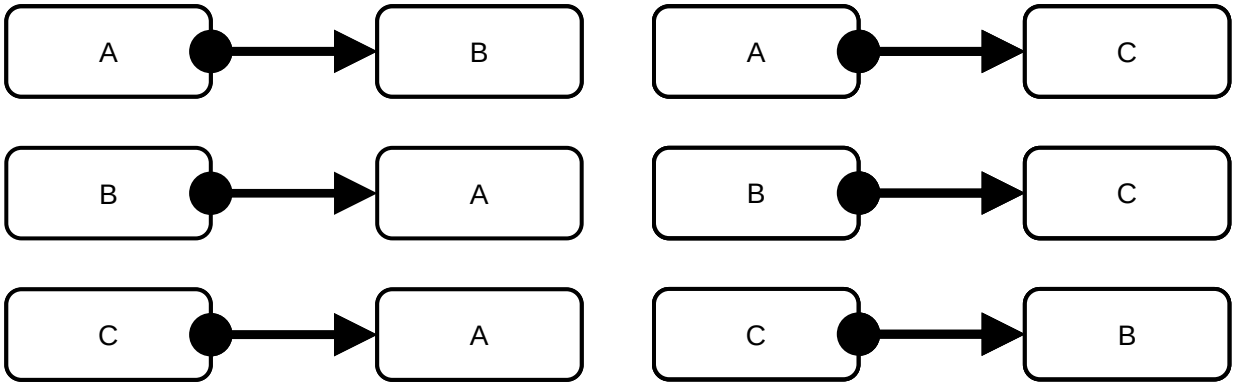
Discovery algorithm

W = [<A C B C>, <C B A C>, <A C A C A C B>]



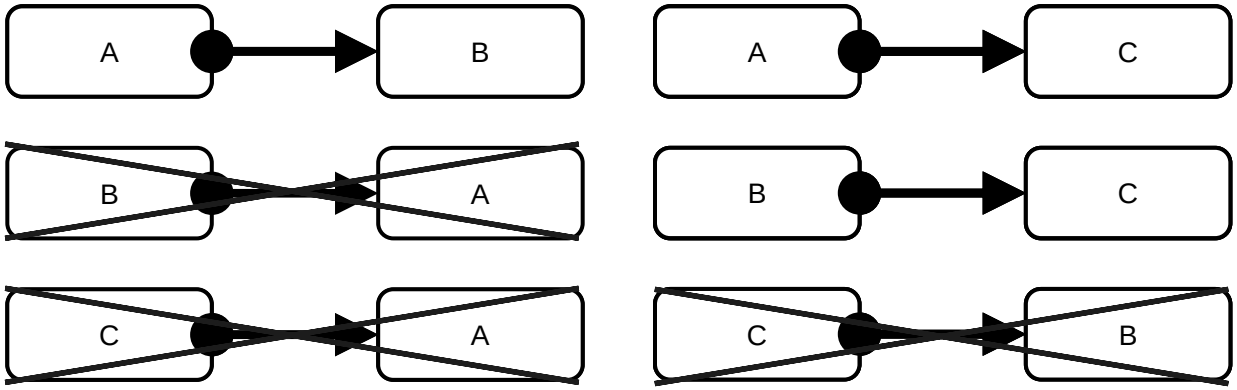
Discovery algorithm: Pruning

W = [<A C B C>, <C B A C>, <A C A C A C B>]



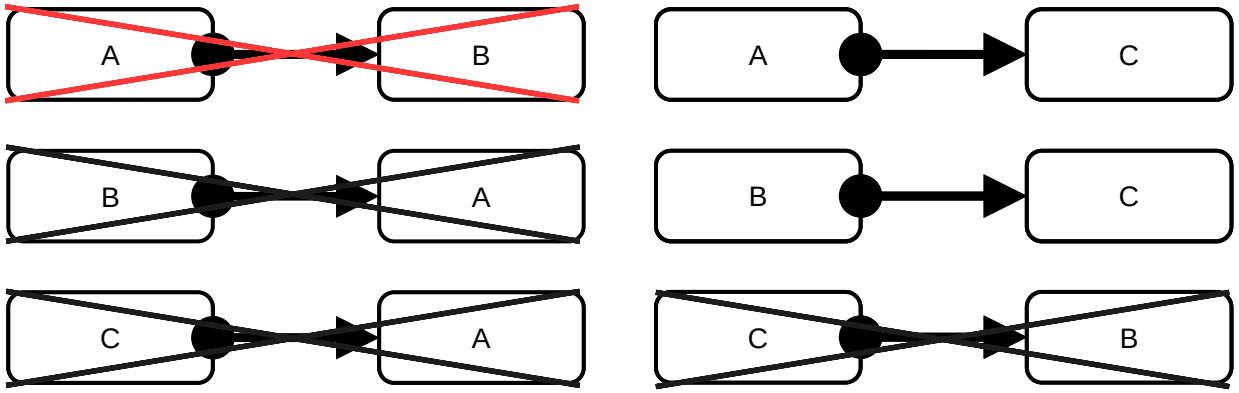
Discovery algorithm: Pruning

W = [<A C B C>, <C B A C>, <A C A C A C B>]



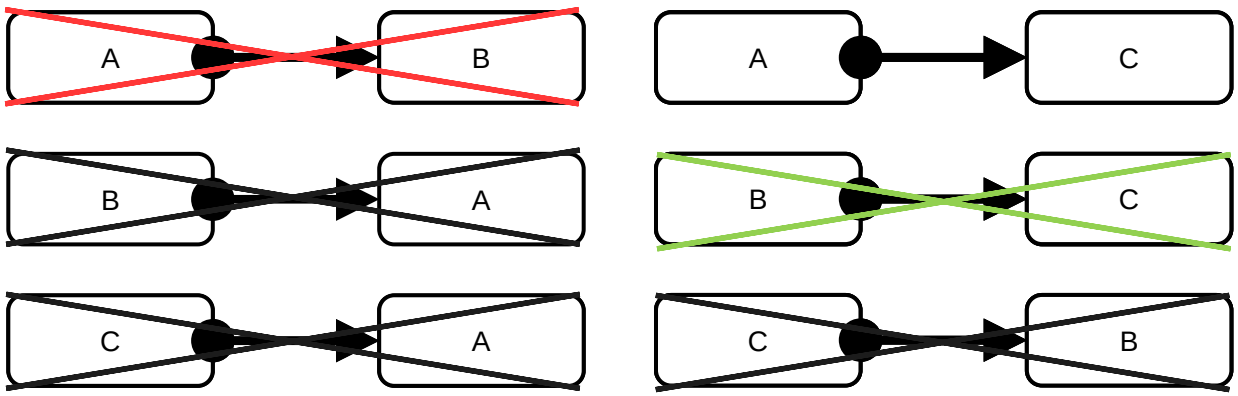
Discovery algorithm: Pruning

W = [<A C B C>, <C B A C>, <A C A C A C B>]



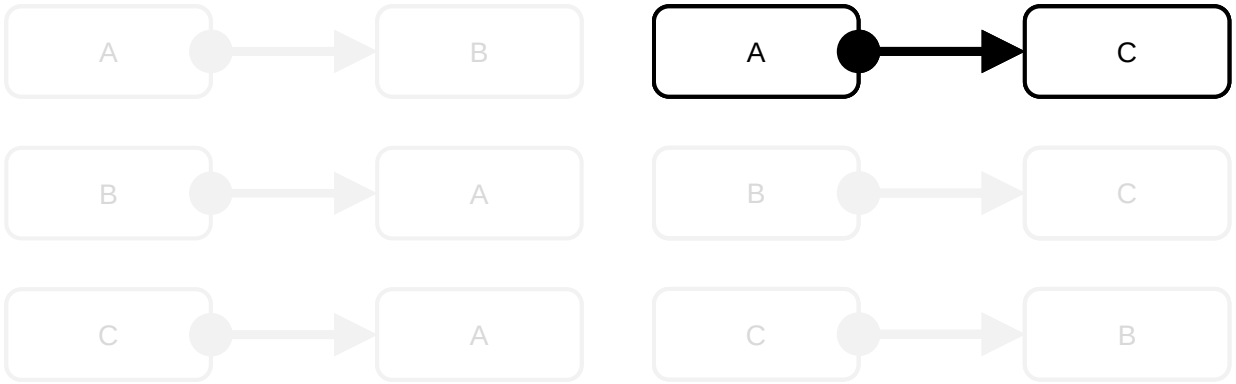
Discovery algorithm: Pruning

W = [<A C B C>, <C B A C>, <A C A C A C B>]



Discovery algorithm: Pruning

W = [<A C B C>, <C B A C>, <A C A C A C B>]



Process Discovery

rulemining.org

Roadmap

Automated Process Discovery

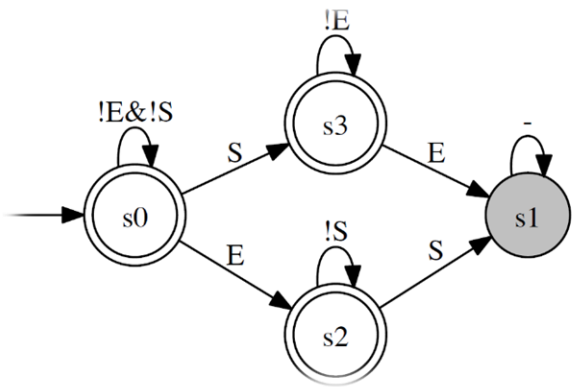
Compliance Monitoring

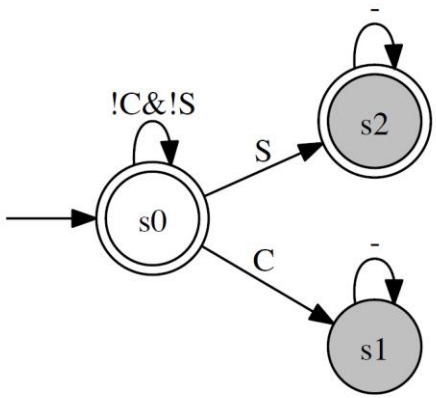
Conformance Checking

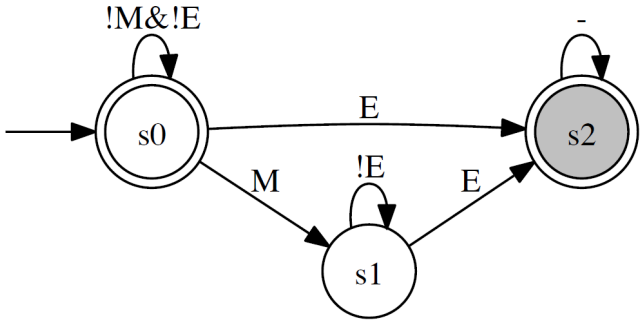
Log Generation

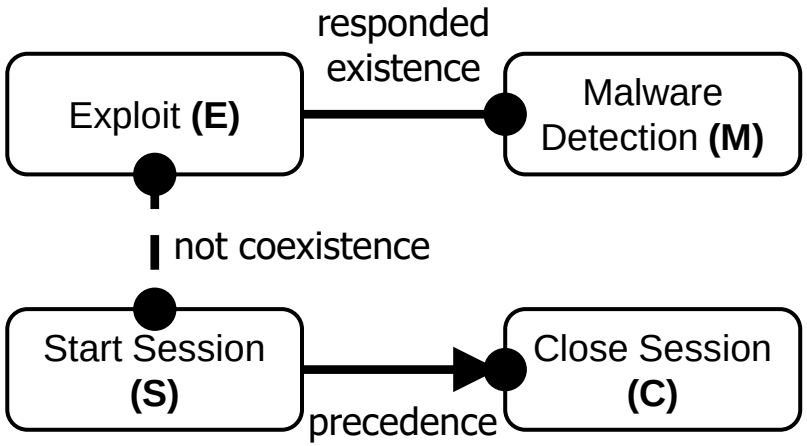
Compliance

Compliance Models

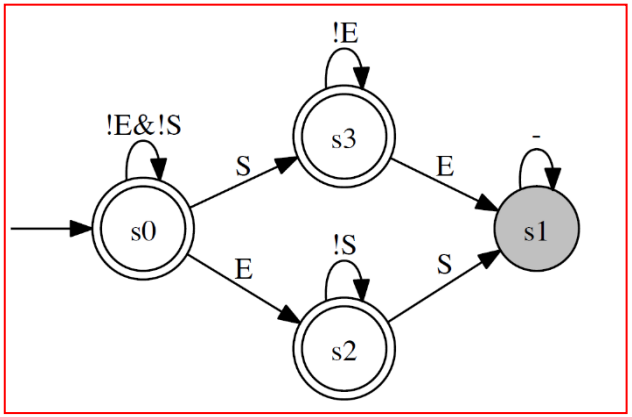


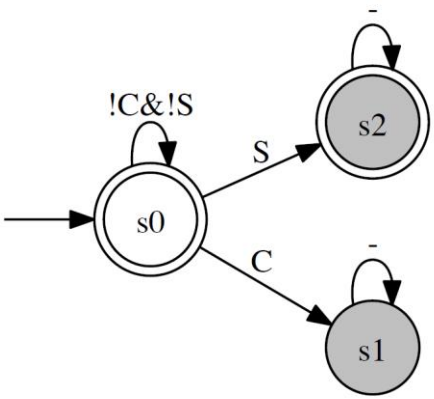


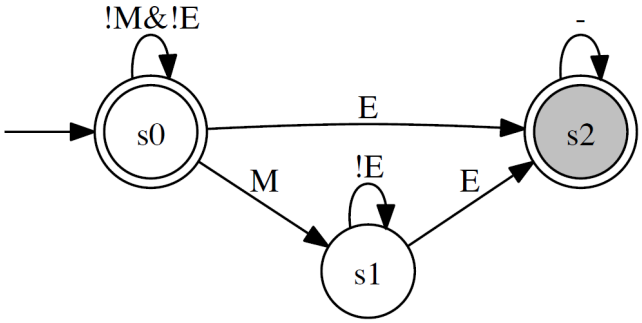


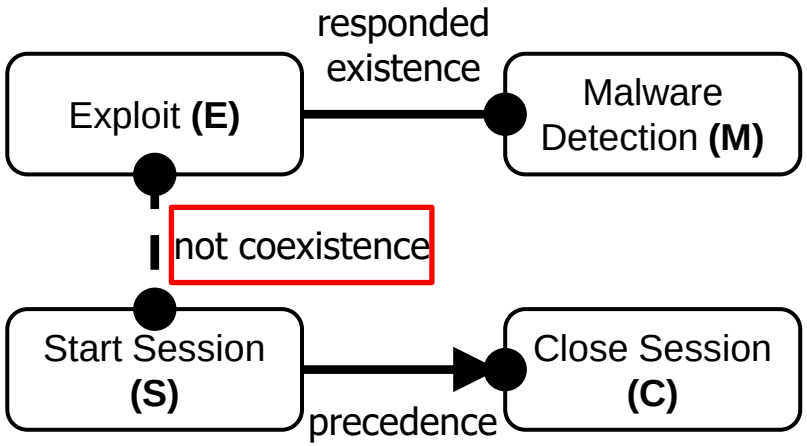


Compliance Models

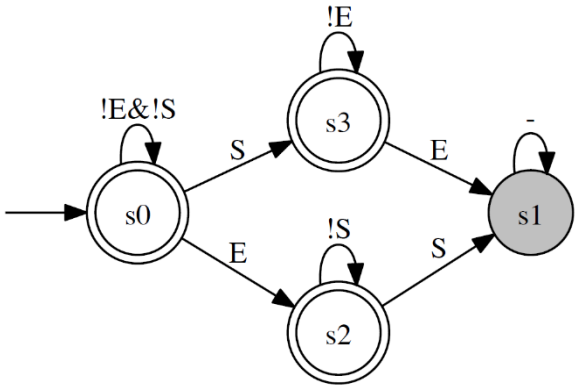


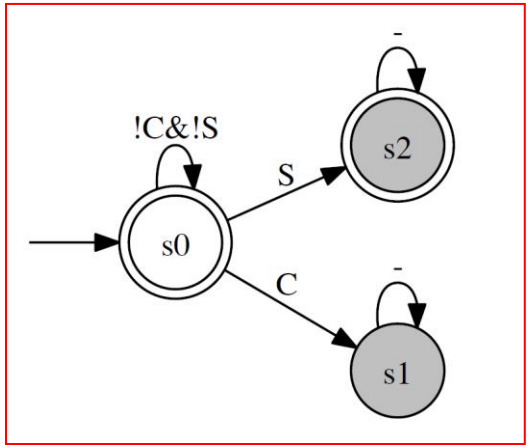


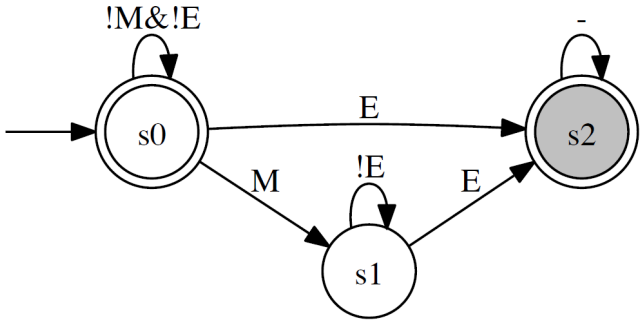


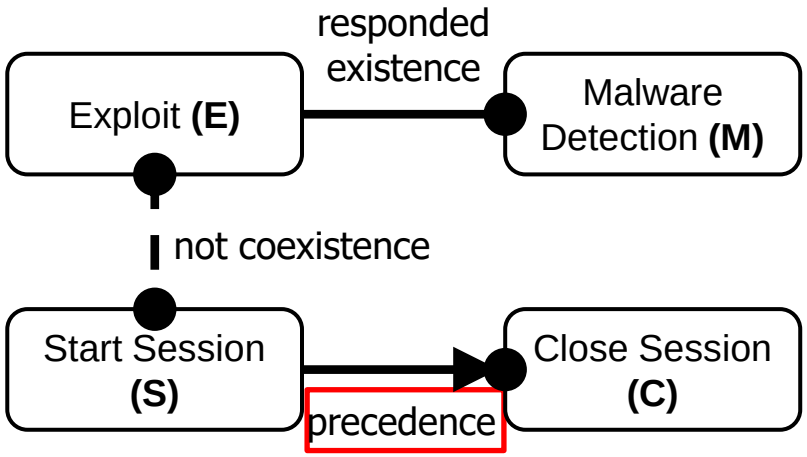


Compliance Models

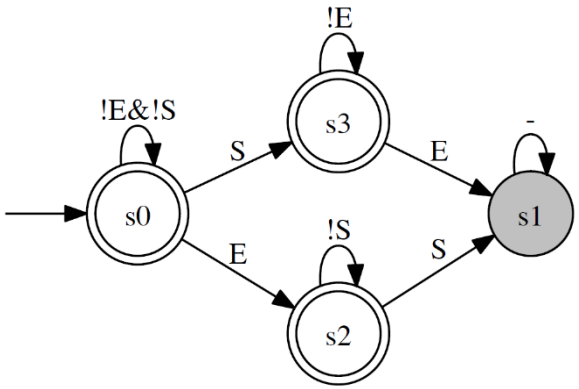


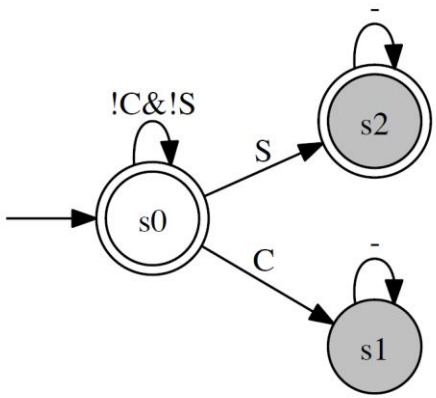


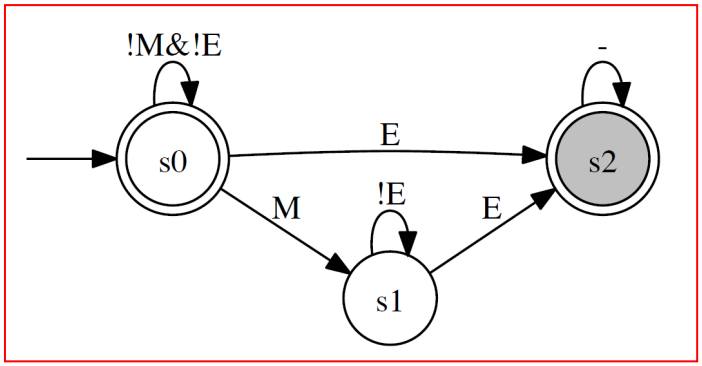


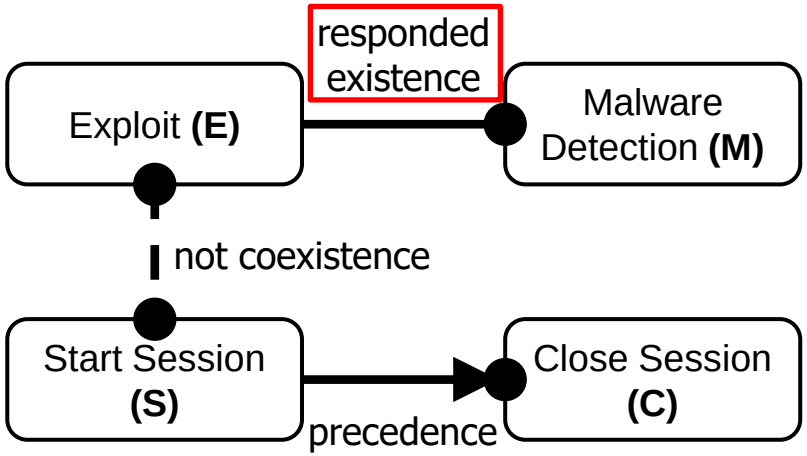


Compliance Models





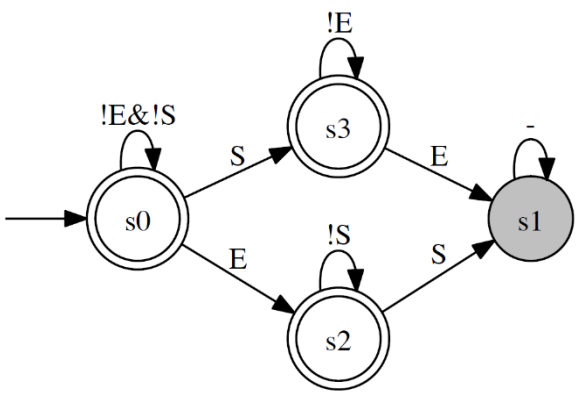


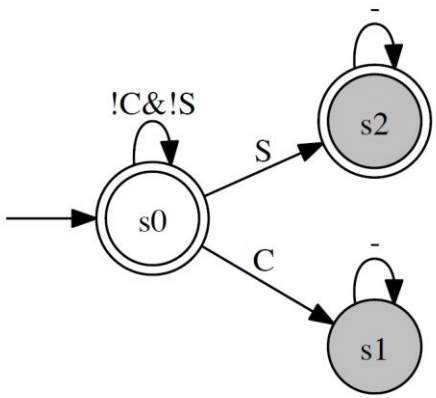


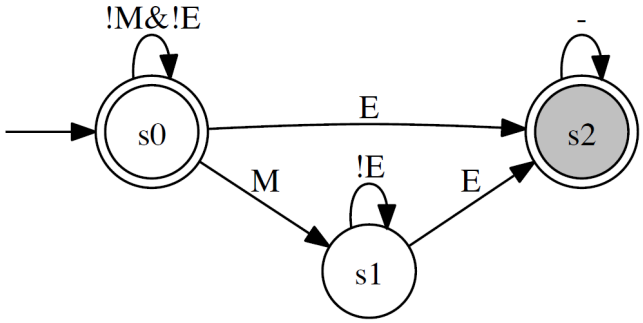
RV-LTL Semantics

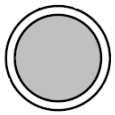
Andreas Bauer, Martin Leucker, Christian Schallhart: ***The Good, the Bad, and the Ugly, But How Ugly Is Ugly?*** RV 2007: 126-138

RV-LTL Semantics



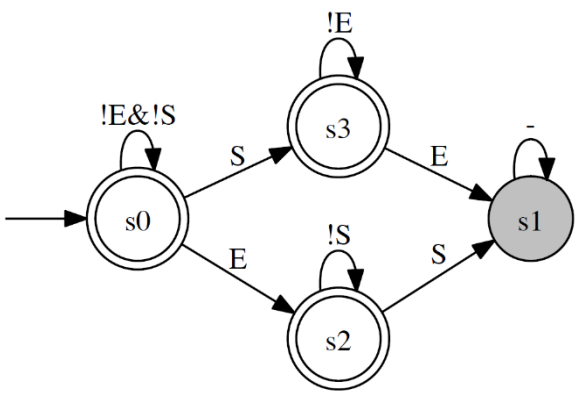


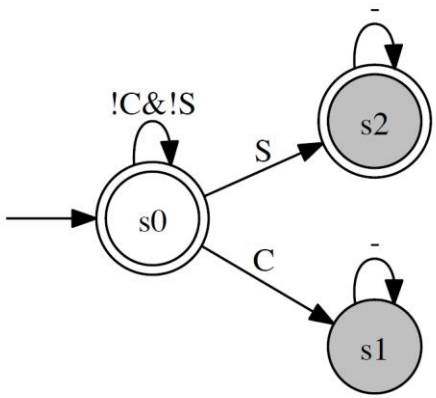


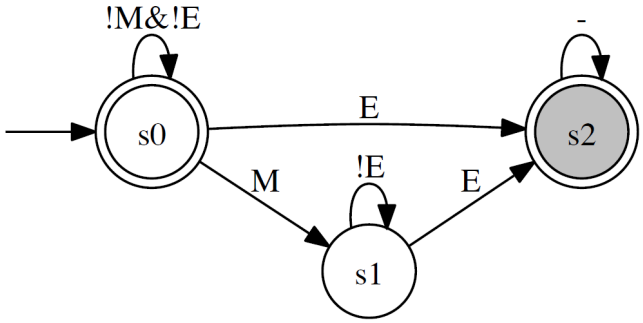


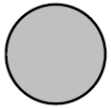
"the good" (permanently sat)

RV-LTL Semantics



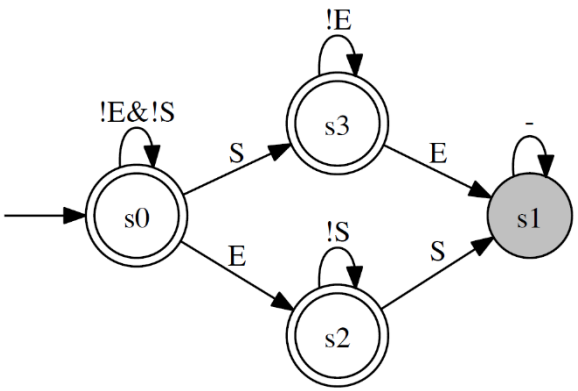


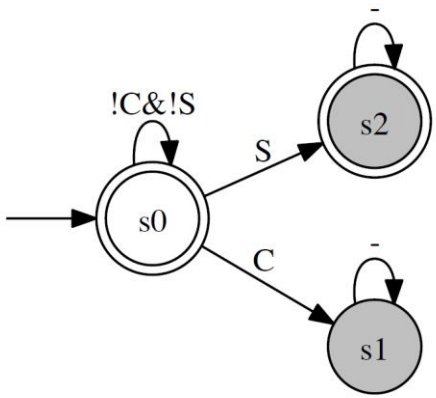


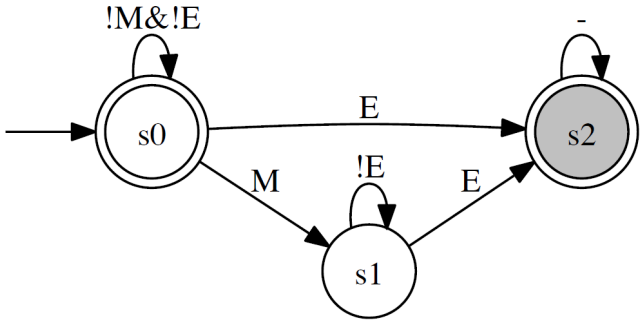


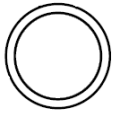
“the bad” (permanently viol)

RV-LTL Semantics

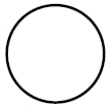








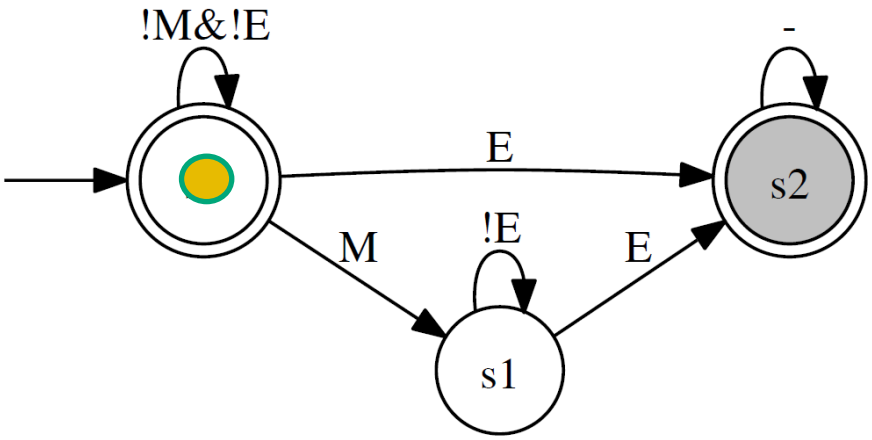
"inconclusive" (possibly sat)

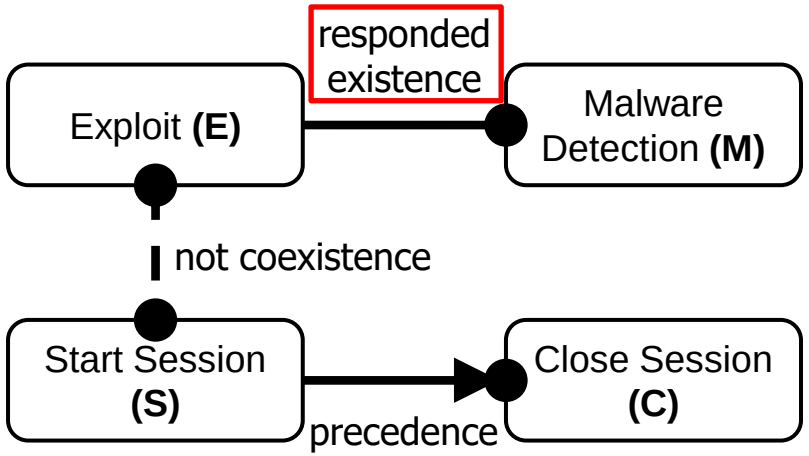


"inconclusive" (possibly viol)

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

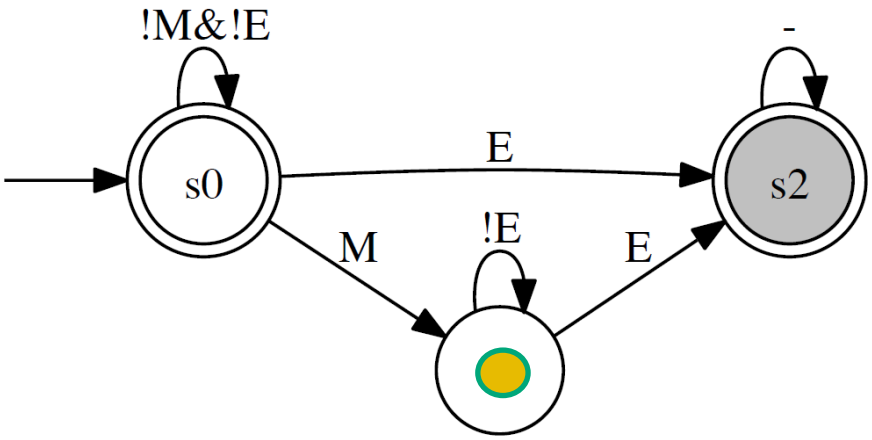


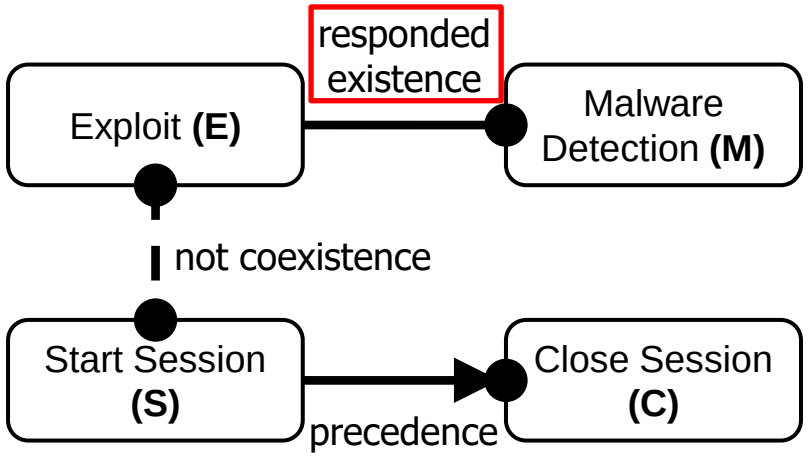


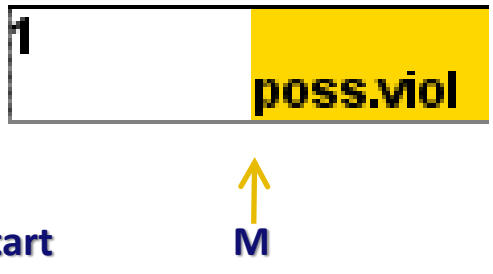
start M S E

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

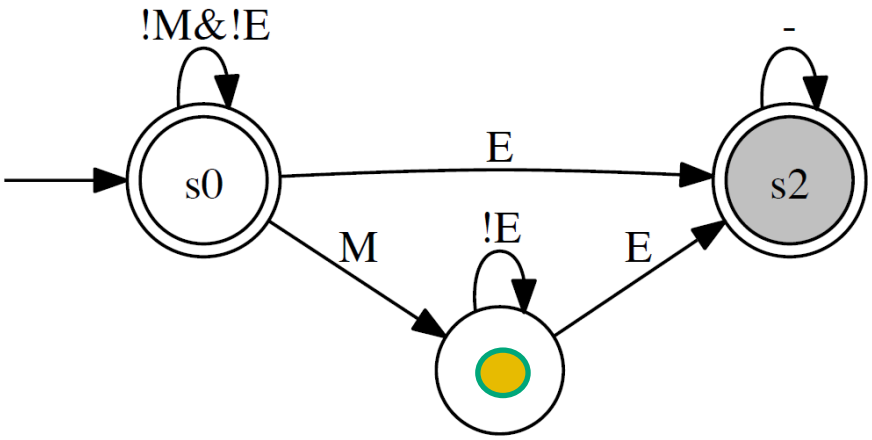


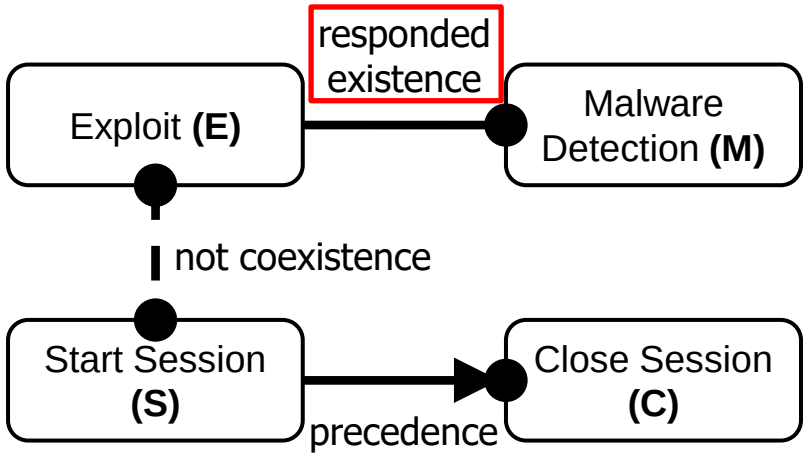




Monitoring Approach

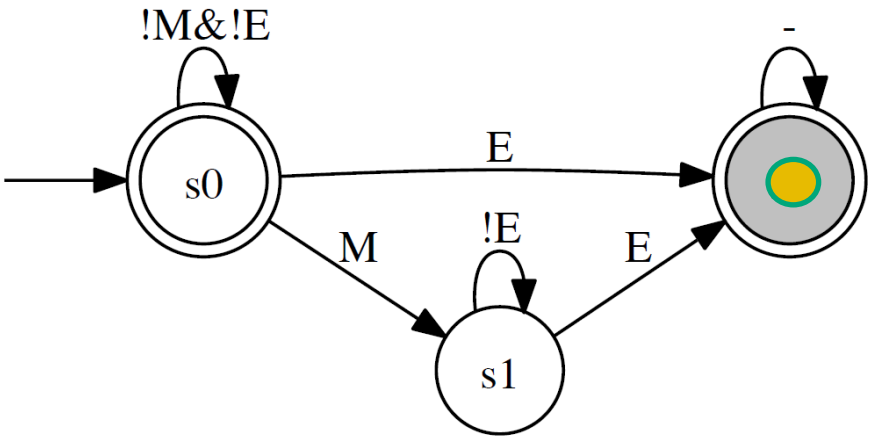
Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

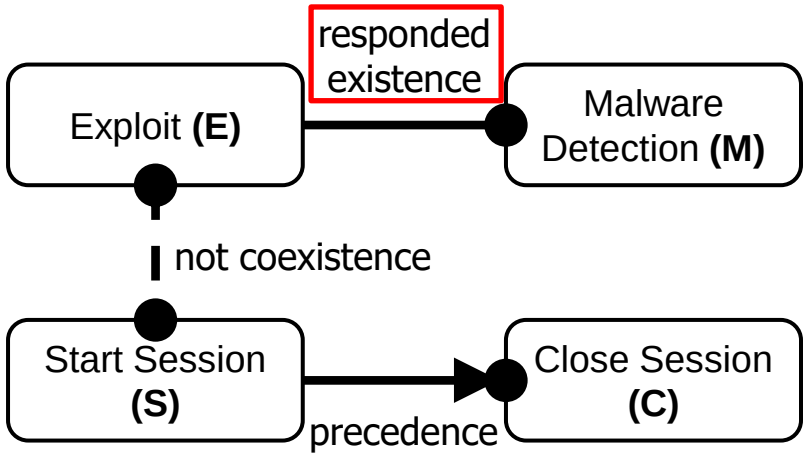




Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

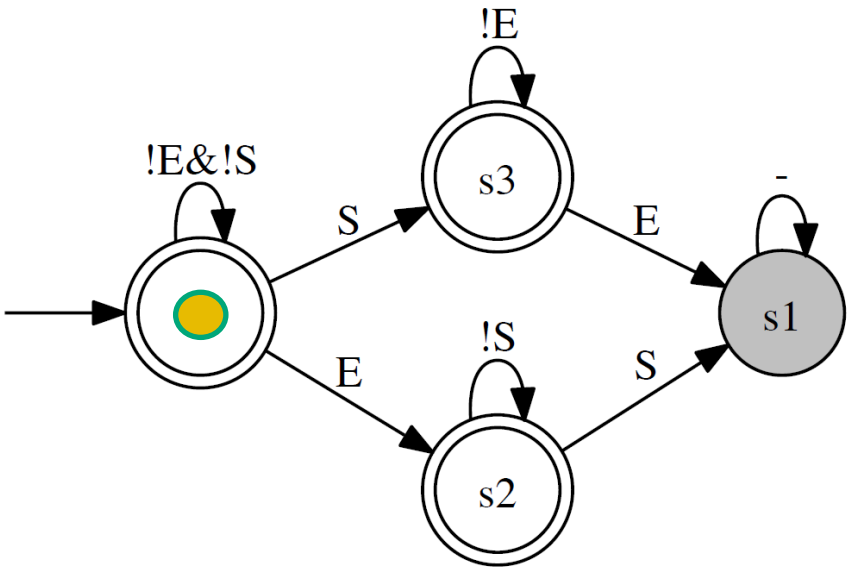


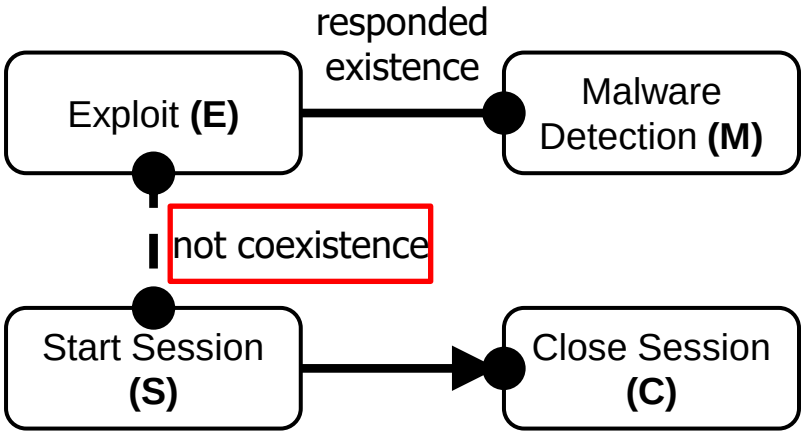


start M S E

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

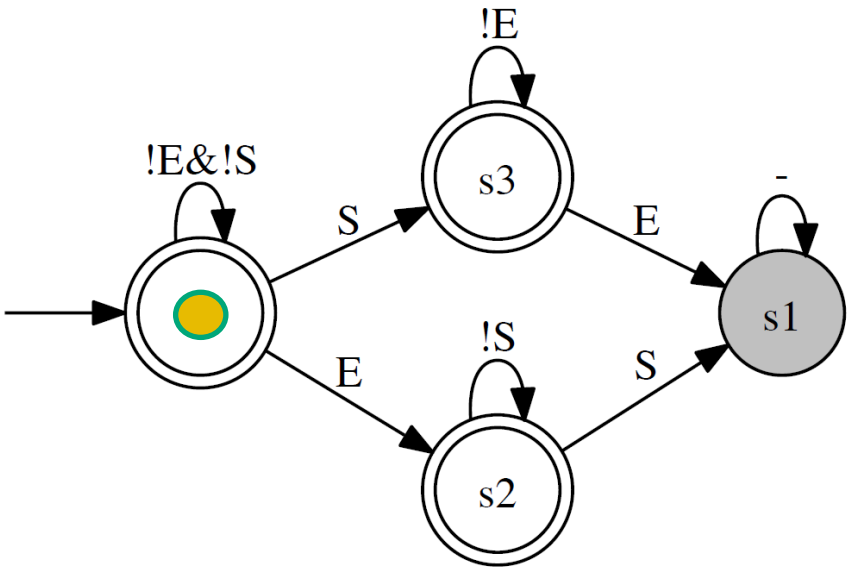


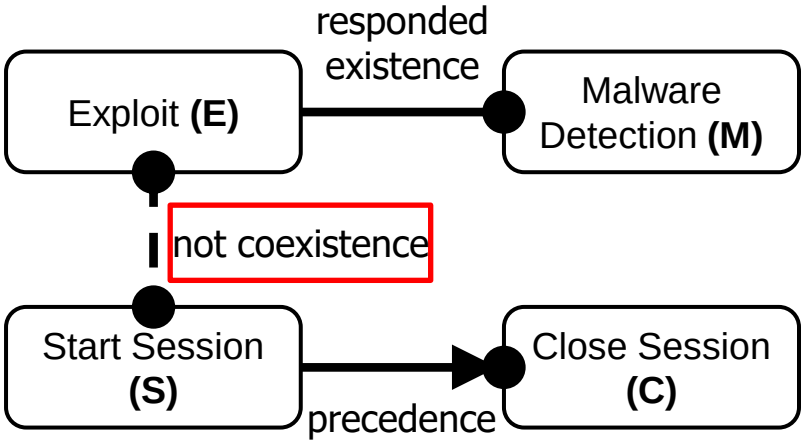


start M S E

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

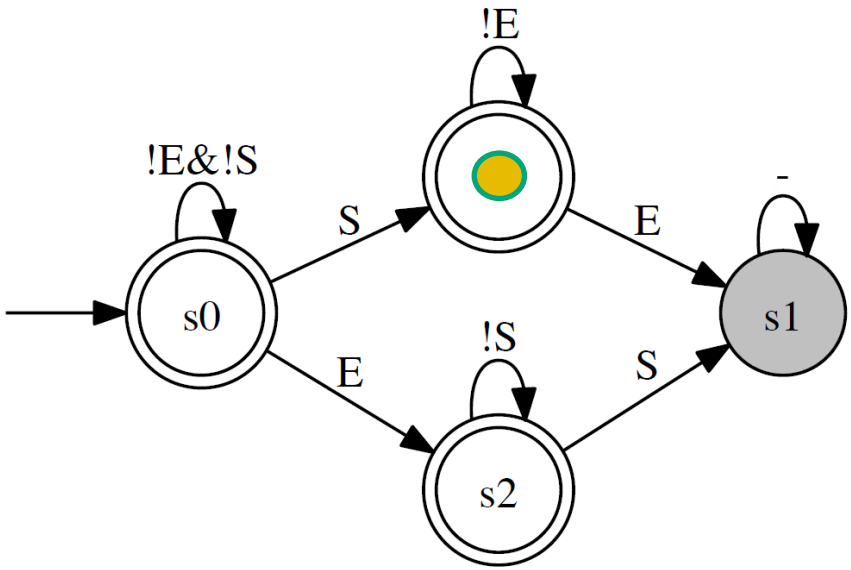


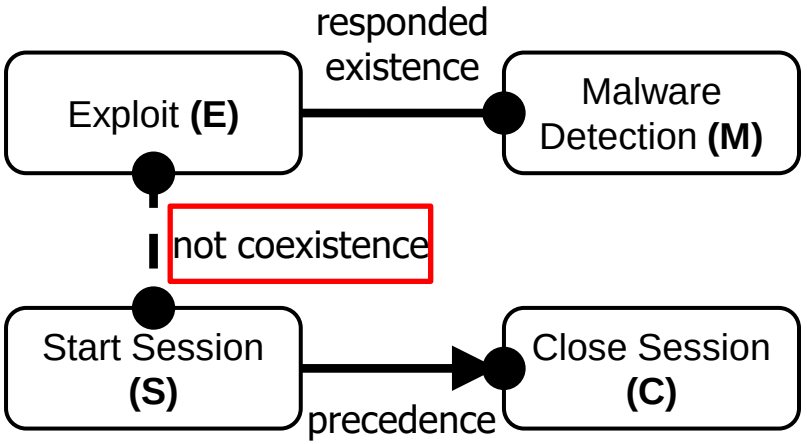


start M S E

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147

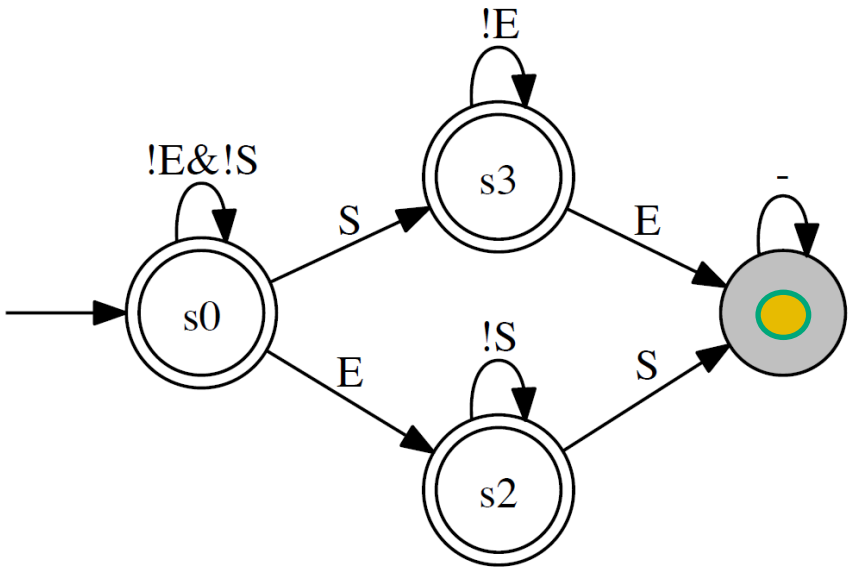


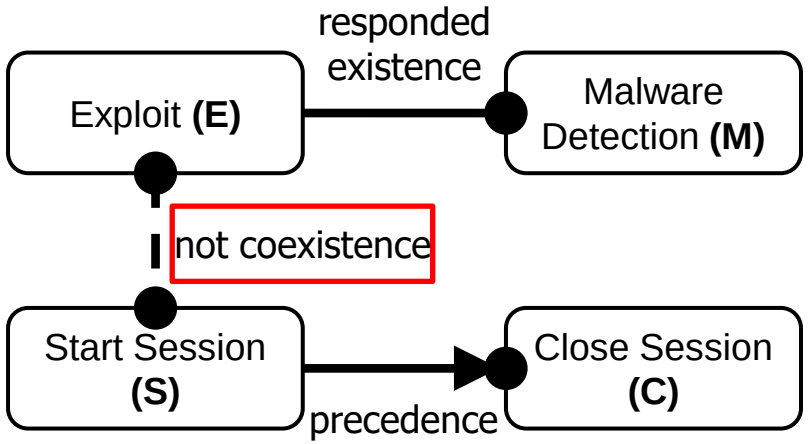


start M S E

Monitoring Approach

Fabrizio M. Maggi, Marco Montali, Michael Westergaard, Wil M. P. van der Aalst: ***Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata***. BPM 2011: 132-147





start M S E

Compliance Monitoring

rulemining.org

Roadmap

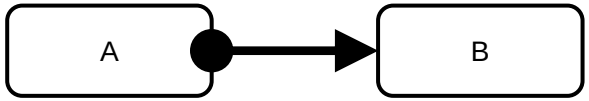
Automated Process Discovery

Compliance Monitoring

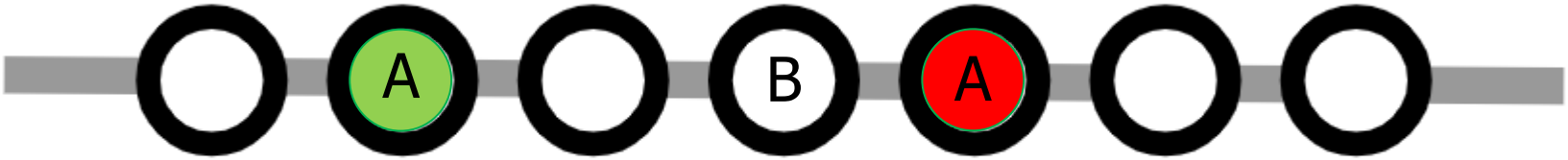
Conformance Checking

Log Generation

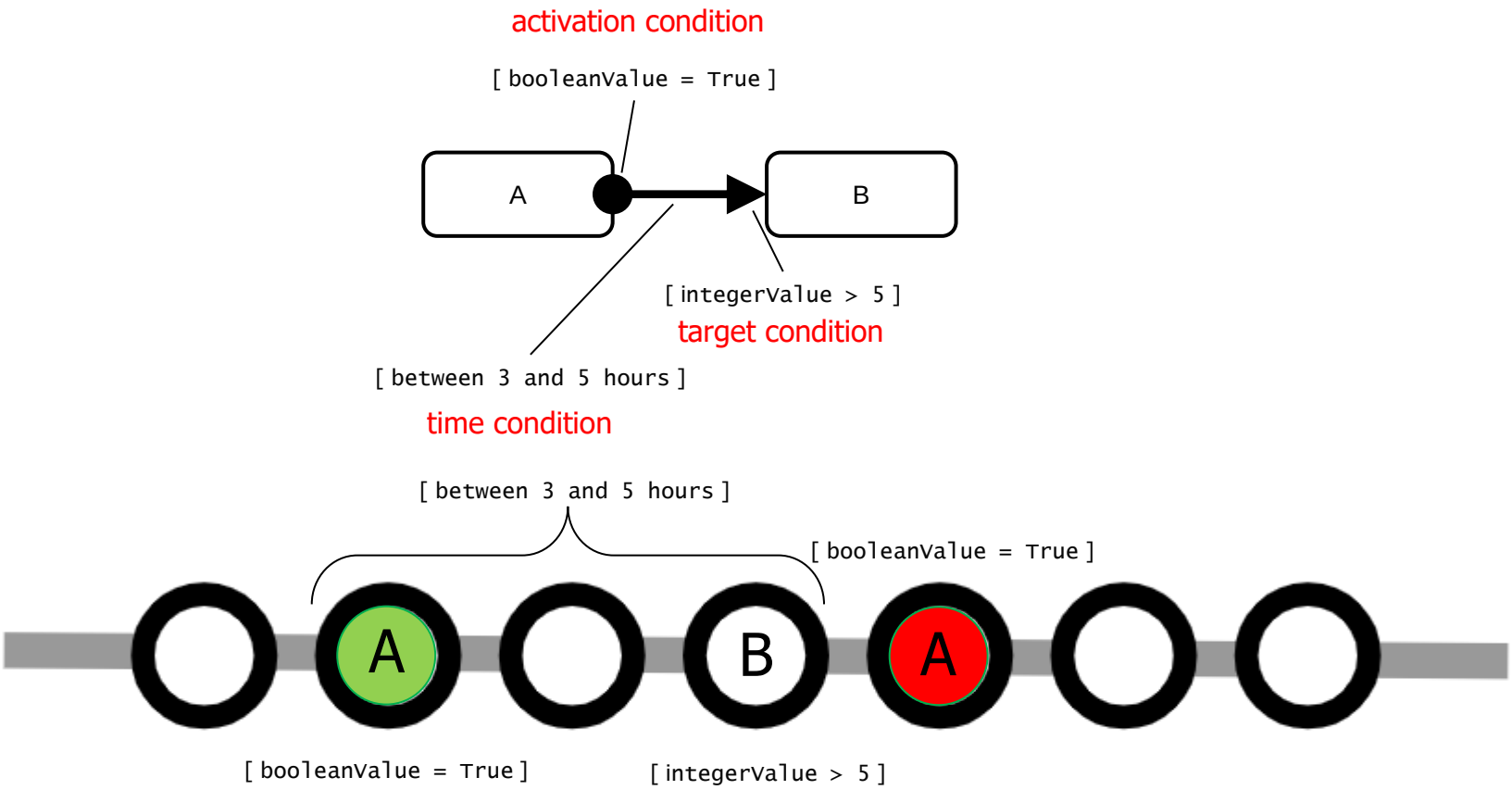
Conformance Checking with Declare Analyzer



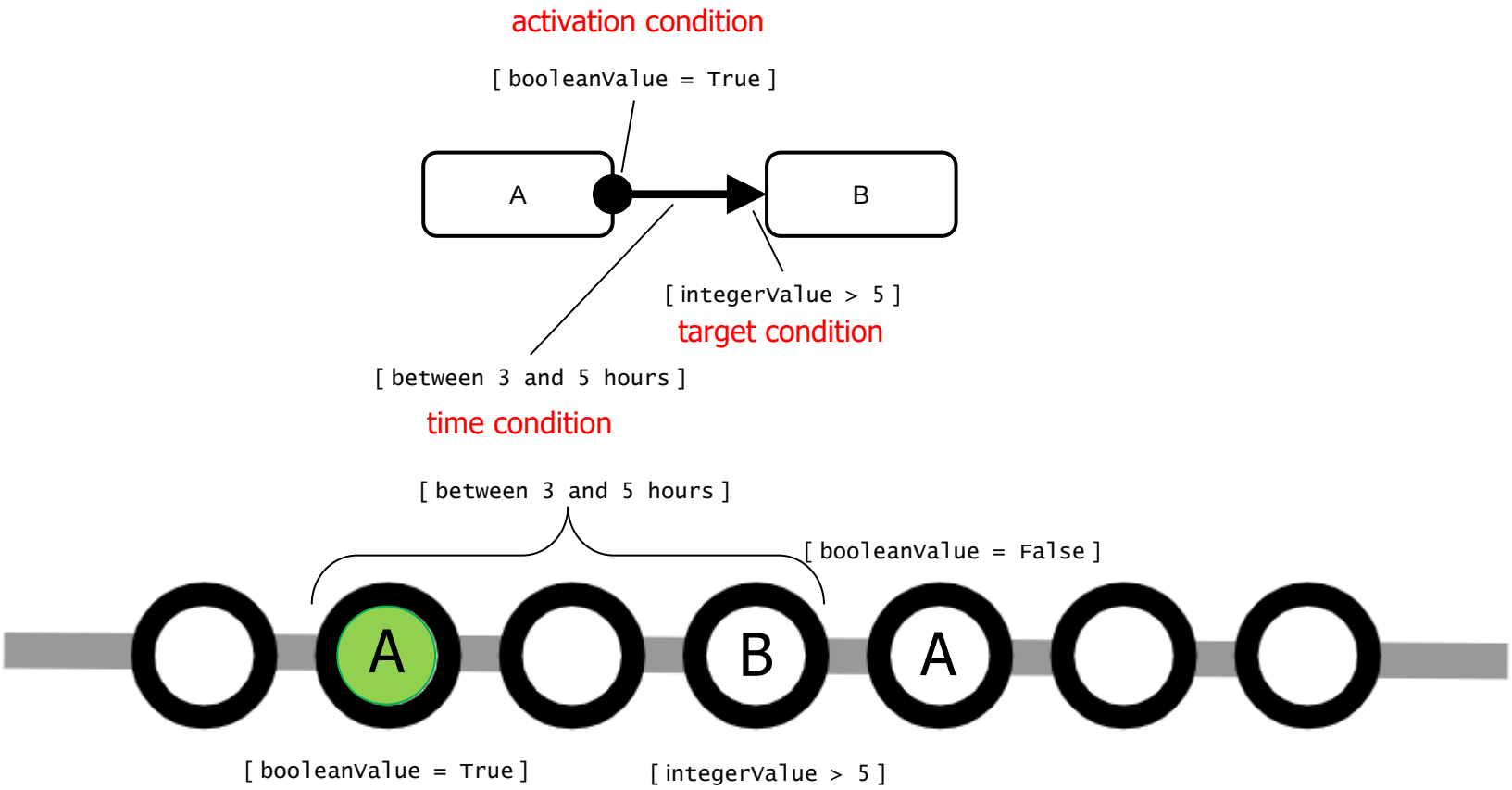
Andrea Burattin, Fabrizio M. Maggi, Alessandro Sperduti Wil M. P. van der Aalst: ***Conformance checking based on multi-perspective declarative process models.*** Expert Syst. Appl. 65: 194-211 (2016)



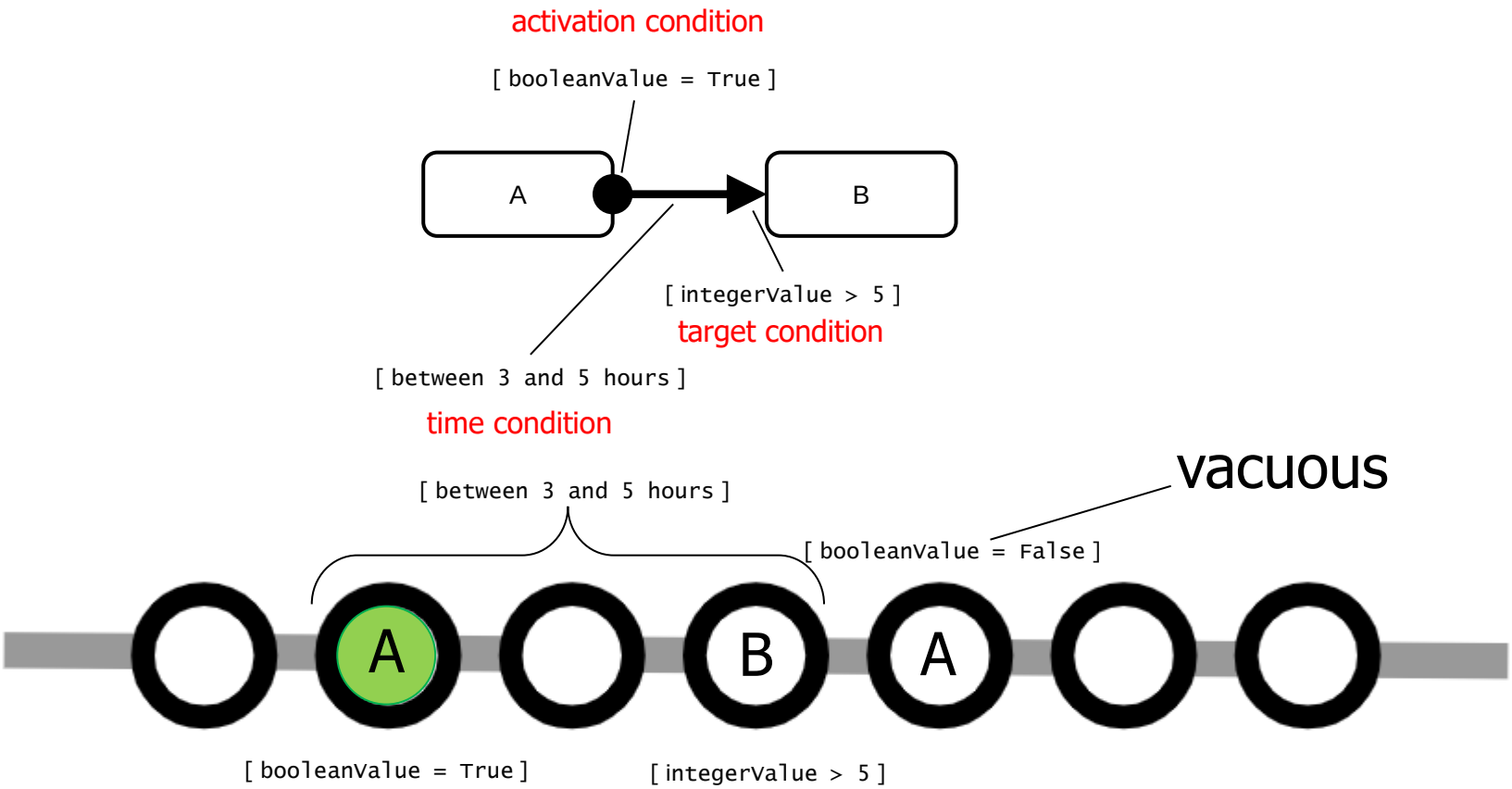
Conformance Checking with Declare Analyzer



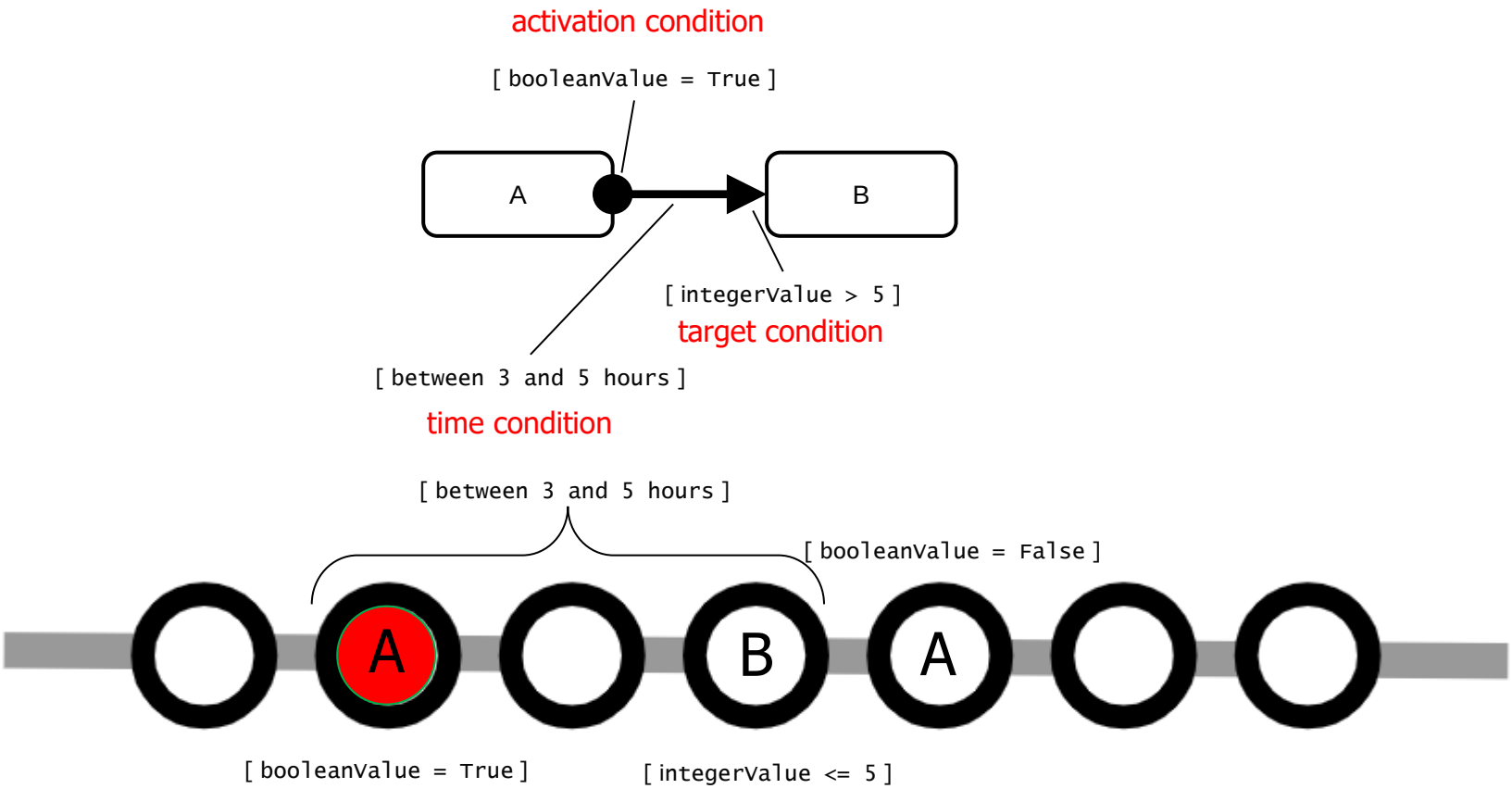
Conformance Checking with Declare Analyzer



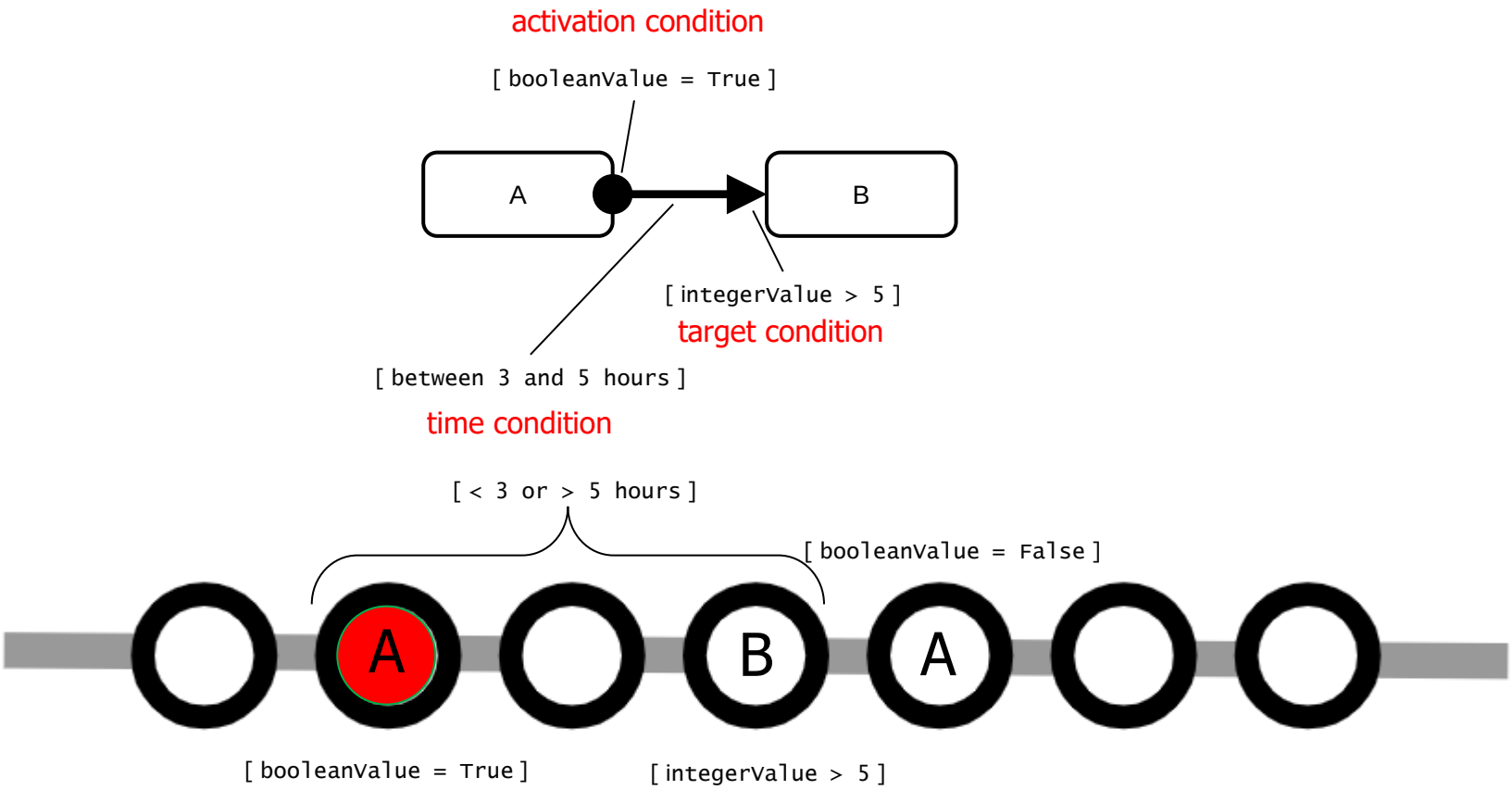
Conformance Checking with Declare Analyzer



Conformance Checking with Trace Alignment



Conformance Checking with Trace Alignment



Conformance Checking with Trace Alignment

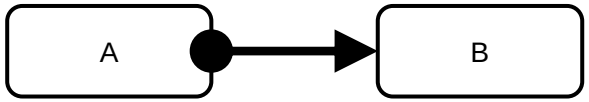
out of Alignment

in Alignment

Trace Alignment

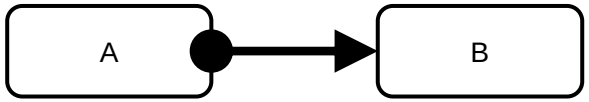
Massimiliano de Leoni, Fabrizio M. Maggi, Wil M. P. van der Aalst: ***Aligning Event Logs and Declarative Process Models for Conformance Checking***. BPM 2012: 82-97

<A A D F C D E>



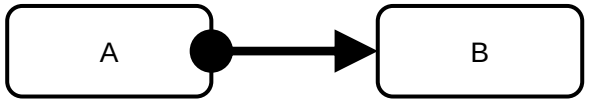
Trace Alignment

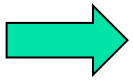
<A A D F C D E>



Trace Alignment

<A A D F C D E>

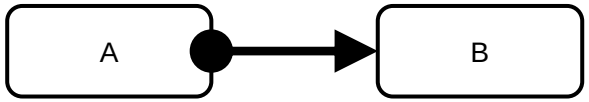




<A A D F C D E B>

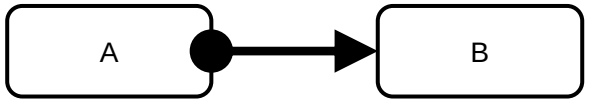
Trace Alignment

<A A D F C D E>



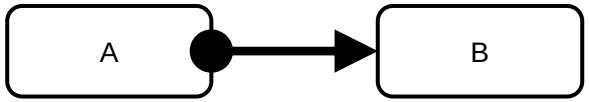
Trace Alignment

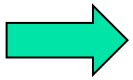
<A A D F C D E>



Trace Alignment

<A A D F C D E>





<A A D F C D E>

in Alignment

Optimal Alignments

Modifications have a cost!

Optimal Alignments

Giuseppe De Giacomo, Fabrizio Maria Maggi, Andrea Marrella, Fabio Patrizi: ***On the Disruptive Effectiveness of Automated Planning for LTLf-Based Trace Alignment***. AAAI 2017: 3555-3561

Modifications have a cost!

Conformance Checking

rulemining.org

Roadmap

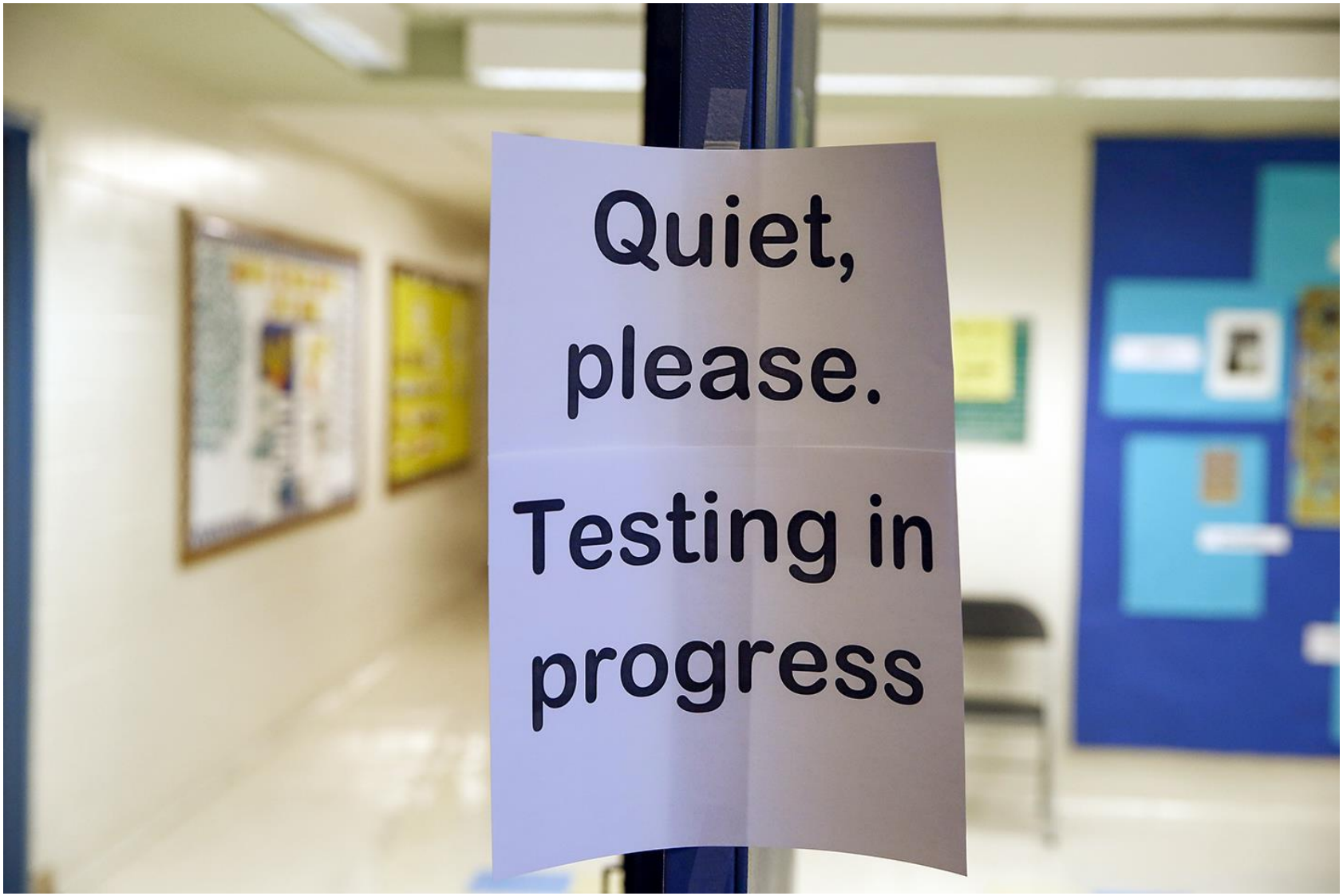
Automated Process Discovery

Compliance Monitoring

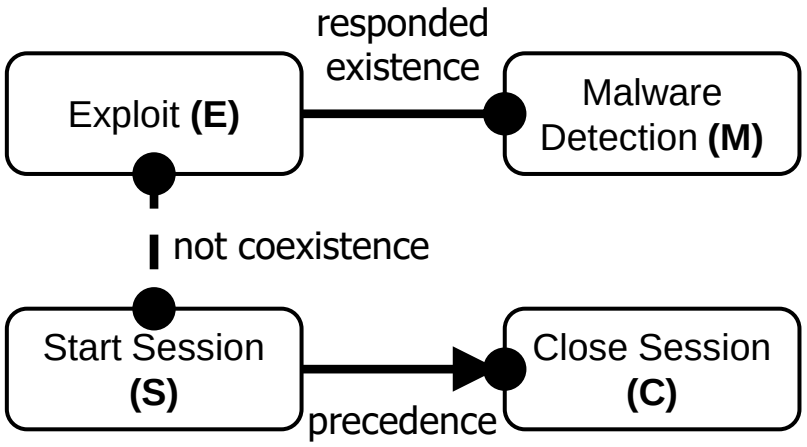
Conformance Checking

Log Generation

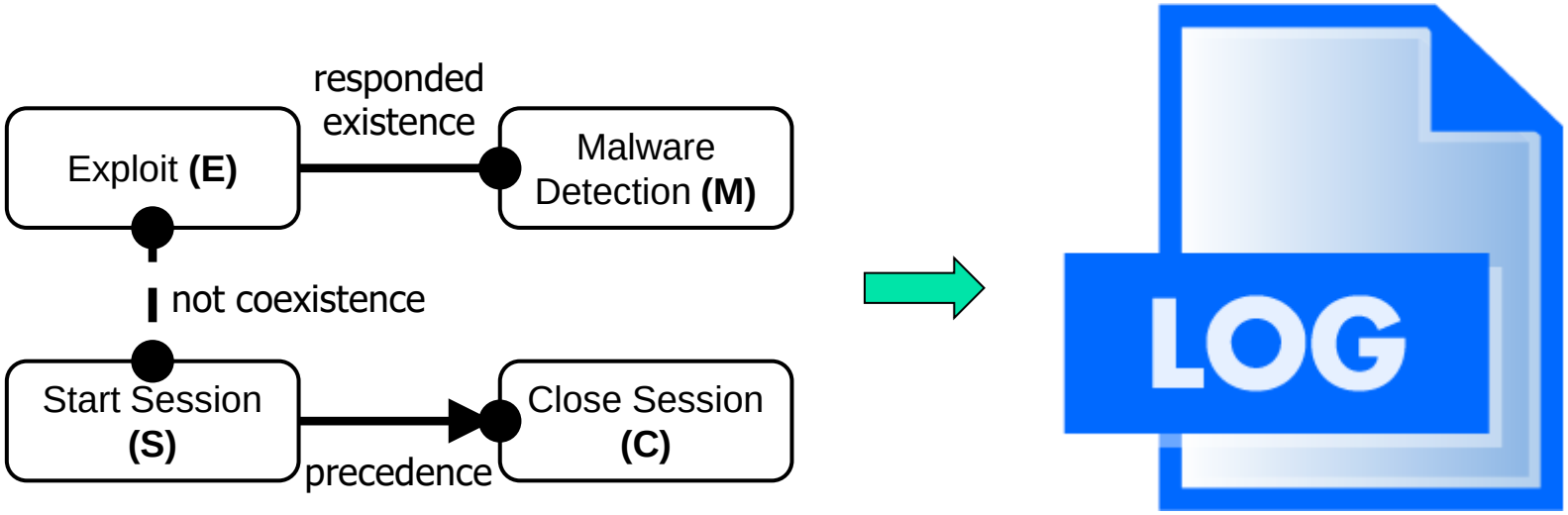
Log Generation



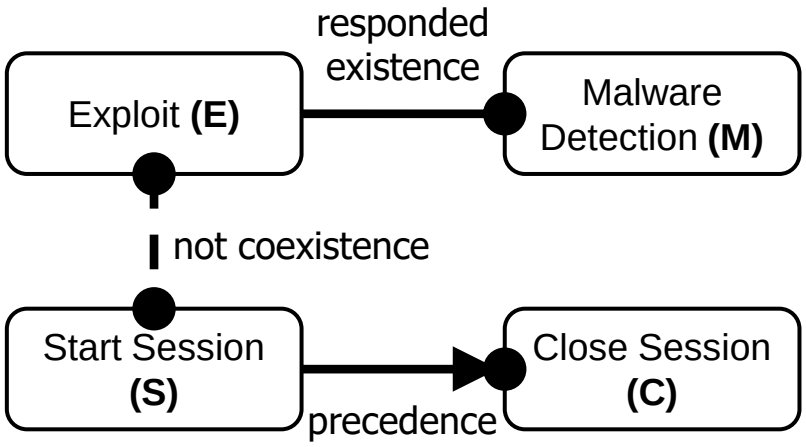
Log Generation



Log Generation



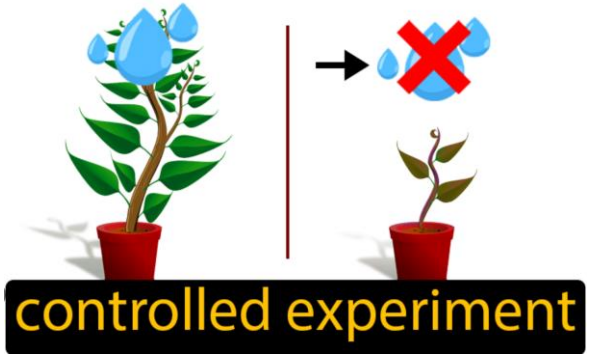
Log Generation



#traces in the log

#events in a trace

...



Log Generation with ASP

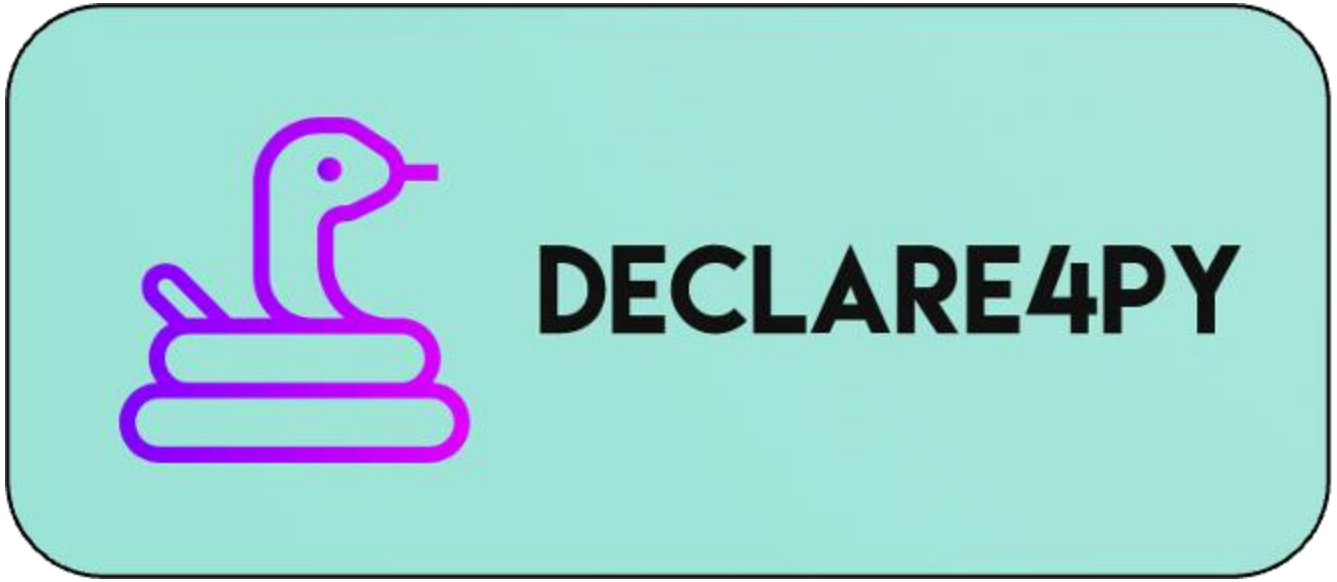
- Log generation can be encoded in Answer Set Programming (ASP) using predicates and rules to define:
  - Each constraint automaton of the input model
  - The length of a trace to be generated
  - The requirement that every automaton ends up in a final state in the last time point of the generated trace
- Predicate ***trace*** is the guessed predicate and contains a sequence of activities satisfying all the input constraints
- To generate a solution for the guessed predicate, an ASP solver can be used
  - e.g., **Clingo**

Francesco Chiariello, Fabrizio M. Maggi, Fabio Patrizi: ***ASP-Based Declarative Process Mining***. AAAI 2022: 5539-5547

Log Generation

rulemining.org

For Python Users…



https://github.com/ivanDonadello/Declare4Py

Thank you
for your attention!