Using Discriminative Rule Mining to Discover Declarative Process Models with Non-Atomic Activities

Mario Luca Bernardi[1] Marta Cimitile[2]
Chiara Di Francescomarino[3] Fabrizio Maria Maggi[4]

[1]University of Sannio, Benevento, Italy [2]Te.L.Ma. University, Rome, Italy
[3]FBK-IRST, Trento, Italy [4]University of Tartu, Estonia

RuleML 2014, August 18-20 2014,
Prague, Czech Republic

Outline

Motivations

- Process discovery techniques are **widely considered as critical** for successful business process management and monitoring;
- The discovery of **declarative models can be used in complex environments** where process executions involve multiple alternatives and **high flexibility is needed**;
- Existing process discovery techniques for generating declarative specifications, **do not take activity lifecycles** and their characteristics into consideration.

Motivations

- Process discovery techniques are **widely considered as critical** for successful business process management and monitoring;
- The discovery of **declarative models can be used in complex environments** where process executions involve multiple alternatives and **high flexibility is needed**;
- Existing process discovery techniques for generating declarative specifications, **do not take activity lifecycles** and their characteristics into consideration.

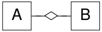
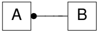
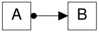
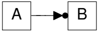
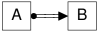
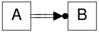
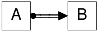
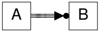
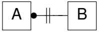
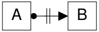
Motivations

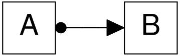
- Process discovery techniques are **widely considered as critical** for successful business process management and monitoring;
- The discovery of **declarative models can be used in complex environments** where process executions involve multiple alternatives and **high flexibility is needed**;
- Existing process discovery techniques for generating declarative specifications, **do not take activity lifecycles** and their characteristics into consideration.

- Declare is a language for **describing declarative process models** consisting of a set of **constraints** applied to (atomic) activities.
- Constraints, in turn, are based on **templates** that are **abstract parameterized patterns**: constraints are their **concrete instantiations** on real activities.
- Template semantics can be formalized using **different logics**, the main one being **Linear Temporal Logic over finite traces**, making them verifiable and executable

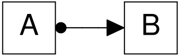
- Declare is a language for **describing declarative process models** consisting of a set of **constraints** applied to (atomic) activities.
- Constraints, in turn, are based on **templates** that are **abstract parameterized patterns**: constraints are their **concrete instantiations** on real activities.
- Template semantics can be formalized using **different logics**, the main one being **Linear Temporal Logic over finite traces**, making them verifiable and executable

- Declare is a language for **describing declarative process models** consisting of a set of **constraints** applied to (atomic) activities.
- Constraints, in turn, are based on **templates** that are **abstract parameterized patterns**: constraints are their **concrete instantiations** on real activities.
- Template semantics can be formalized using **different logics**, the main one being **Linear Temporal Logic over finite traces**, making them verifiable and executable

TEMPLATE	FORMALIZATION	NOTATION
existence(A)	$\Diamond A$	
absence(A)	$\neg\Diamond A$	
choice(A,B)	$\Diamond A \vee \Diamond B$	
exclusive choice(A,B)	$(\Diamond A \vee \Diamond B) \wedge \neg(\Diamond A \wedge \Diamond B)$	
responded existence(A,B)	$\Diamond A \rightarrow \Diamond B$	
response(A,B)	$\Box(A \rightarrow \Diamond B)$	
precedence(A,B)	$\neg B\, \mathcal{W}\, A$	
alternate response(A,B)	$\Box(A \rightarrow \bigcirc(\neg A\, \mathcal{U}\, B))$	
alternate precedence(A,B)	$(\neg B\, \mathcal{W}\, A) \wedge \Box(B \rightarrow \bigcirc(\neg B\, \mathcal{W}\, A))$	
chain response(A,B)	$\Box(A \rightarrow \bigcirc B)$	
chain precedence(A,B)	$\Box(\bigcirc B \rightarrow A)$	
not resp. existence(A,B)	$\Diamond A \rightarrow \neg\Diamond B$	
not response(A,B)	$\Box(A \rightarrow \neg\Diamond B)$	
not precedence(A,B)	$\Box(A \rightarrow \neg\Diamond B)$	

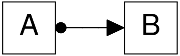
TEMPLATE	FORMALIZATION	NOTATION
response(A,B)	$\Box(A \rightarrow \Diamond B)$	

$\langle a, a, b, c \rangle$ Satisfied
$\langle b, b, c, d \rangle$ Satisfied (vacuously)
$\langle a, b, c, b \rangle$ Satisfied
$\langle a, b, a, c \rangle$ Not Satisfied

TEMPLATE	FORMALIZATION	NOTATION
response(A,B)	$\Box(A \rightarrow \Diamond B)$	

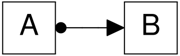
$\langle a, a, b, c \rangle$ one violation

Violated

TEMPLATE	FORMALIZATION	NOTATION
response(A,B)	$\Box(A \rightarrow \Diamond B)$	

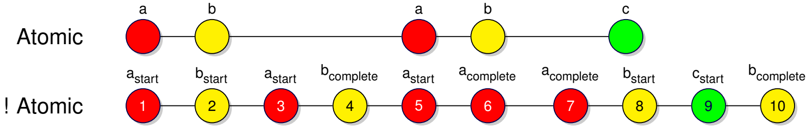
$\downarrow \downarrow$ two violations

$\langle a, a, b, c \rangle$ Violated

TEMPLATE	FORMALIZATION	NOTATION
response(A,B)	$\Box(A \rightarrow \Diamond B)$	

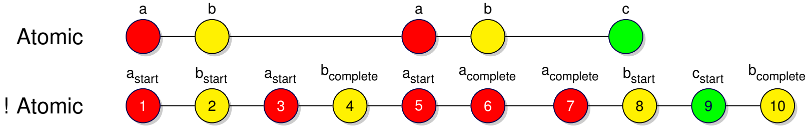
$\downarrow \; \downarrow \; \downarrow$
$\langle a, a, b, c \rangle$

2 fullfillments
Satisfied



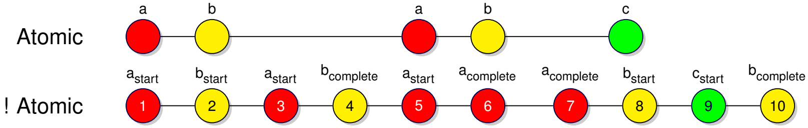
Activity Lifecycle

- In the reality, activities have a **duration spanning over time intervals** in which transactional states (a.k.a. event types) of the activity can occur;
- The **sequences of event types** that occur when an activity is executed, **characterize the lifecycle** of that activity;
- If available, **event type data is very important** and should be considered: it allows to understand the constraints **between event types inside activities**.



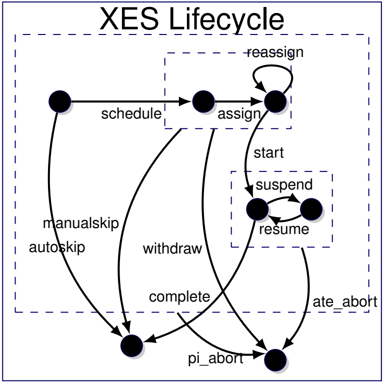
Activity Lifecycle

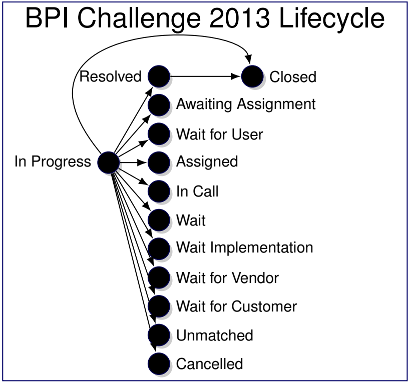
- In the reality, activities have a **duration spanning over time intervals** in which transactional states (a.k.a. event types) of the activity can occur;
- The **sequences of event types** that occur when an activity is executed, **characterize the lifecycle** of that activity;
- If available, **event type data is very important** and should be considered: it allows to understand the constraints **between event types inside activities**.



Activity Lifecycle

- In the reality, activities have a **duration spanning over time intervals** in which transactional states (a.k.a. event types) of the activity can occur;
- The **sequences of event types** that occur when an activity is executed, **characterize the lifecycle** of that activity;
- If available, **event type data is very important** and should be considered: it allows to understand the constraints **between event types inside activities**.


XES Lifecycle
reassign
schedule
assign
start
suspend
resume
manualskip
autoskip
withdraw
complete
ate_abort
pi_abort


BPI Challenge 2013 Lifecycle
Resolved
Closed
Awaiting Assignment
Wait for User
In Progress
Assigned
In Call
Wait
Wait Implementation
Wait for Vendor
Wait for Customer
Unmatched
Cancelled

Discriminative Mining

- Discriminative mining was exploited **to highlight patterns that are discriminative** with respect to a **given criterion** from the existing set of constraints activations over an event logs;
- In order to mine the **set of declarative rules that discriminate between fullfillmens and violations**, decision tree supervised learning
- **Decision tree learning** uses a **decision tree as a model to predict** the value of a target variable based on input variables (features) and are built from a set of training dataset;
- Internal nodes of the tree are **labeled with input features** and edges with **possible value ranges** of the feature;
- Each leaf specifies **support** (the number of examples in the training dataset correctly classified on the path) and **class probability** (the percentage of examples correctly classified).

Discriminative Mining

- Discriminative mining was exploited **to highlight patterns that are discriminative** with respect to a **given criterion** from the existing set of constraints activations over an event logs;
- In order to mine the **set of declarative rules that discriminate between fullfillmens and violations**, decision tree supervised learning
- **Decision tree learning** uses a **decision tree as a model to predict** the value of a target variable based on input variables (features) and are built from a set of training dataset;
- Internal nodes of the tree are **labeled with input features** and edges with **possible value ranges** of the feature;
- Each leaf specifies **support** (the number of examples in the training dataset correctly classified on the path) and **class probability** (the percentage of examples correctly classified).

Discriminative Mining

- Discriminative mining was exploited **to highlight patterns that are discriminative** with respect to a **given criterion** from the existing set of constraints activations over an event logs;
- In order to mine the **set of declarative rules that discriminate between fullfillmens and violations**, decision tree supervised learning
- **Decision tree learning** uses a **decision tree as a model to predict** the value of a target variable based on input variables (features) and are built from a set of training dataset;
- Internal nodes of the tree are **labeled with input features** and edges with **possible value ranges** of the feature;
- Each leaf specifies **support** (the number of examples in the training dataset correctly classified on the path) and **class probability** (the percentage of examples correctly classified).

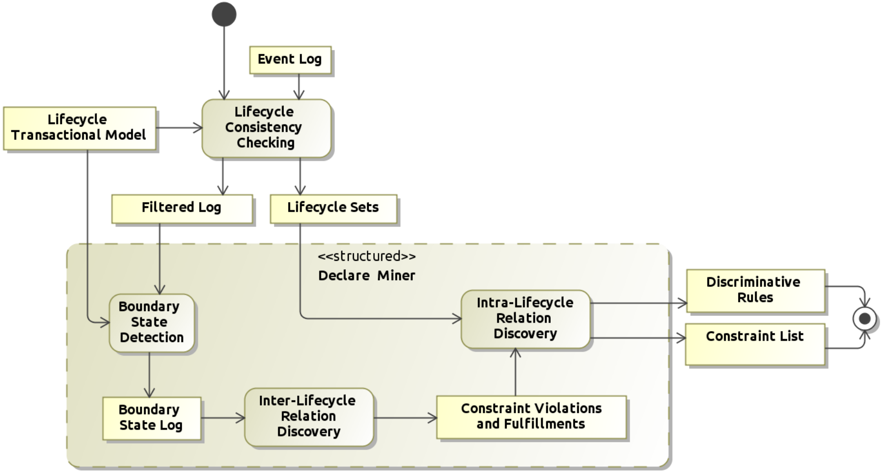
Discriminative Mining

- Discriminative mining was exploited **to highlight patterns that are discriminative** with respect to a **given criterion** from the existing set of constraints activations over an event logs;
- In order to mine the **set of declarative rules that discriminate between fullfillmens and violations**, decision tree supervised learning
- **Decision tree learning** uses a **decision tree as a model to predict** the value of a target variable based on input variables (features) and are built from a set of training dataset;
- Internal nodes of the tree are **labeled with input features** and edges with **possible value ranges** of the feature;
- Each leaf specifies **support** (the number of examples in the training dataset correctly classified on the path) and **class probability** (the percentage of examples correctly classified).

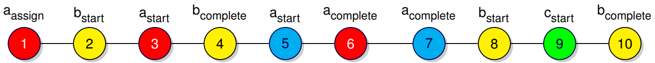
Discriminative Mining

- Discriminative mining was exploited **to highlight patterns that are discriminative** with respect to a **given criterion** from the existing set of constraints activations over an event logs;
- In order to mine the **set of declarative rules that discriminate between fullfillmens and violations**, decision tree supervised learning
- **Decision tree learning** uses a **decision tree as a model to predict** the value of a target variable based on input variables (features) and are built from a set of training dataset;
- Internal nodes of the tree are **labeled with input features** and edges with **possible value ranges** of the feature;
- Each leaf specifies **support** (the number of examples in the training dataset correctly classified on the path) and **class probability** (the percentage of examples correctly classified).

The Process


Event Log
Lifecycle Transactional Model
Lifecycle Consistency Checking
Filtered Log
Lifecycle Sets
<<structured>>
Declare Miner
Boundary State Detection
Intra-Lifecycle Relation Discovery
Discriminative Rules
Constraint List
Boundary State Log
Inter-Lifecycle Relation Discovery
Constraint Violations and Fulfillments

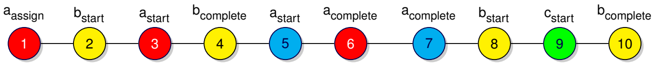
Lifetime Consistency Checking



The proposed technique aims at processing the input log to **connect together** activity states **belonging to the same lifecycle**; this can be done with 2 approaches:

- **FIFO** - applying our FIFO-based algorithm, we can disambiguate the correlation using a conservative approach : $a_{complete}$ at pos. 6 can be connected to a_{start} at different positions but we connect it to the occurrence that occurred first (hence position 3) generating lifecycle 1-3-6.
- **Data correlation** - if the event log contains (event) data attributes, it is possible to connect activity states that share some data values, e.g., an event id

Lifetime Consistency Checking



The proposed technique aims at processing the input log to **connect together** activity states **belonging to the same lifecycle**; this can be done with 2 approaches:

- **FIFO** - applying our FIFO-based algorithm, we can disambiguate the correlation using a conservative approach : $a_{complete}$ at pos. 6 can be connected to a_{start} at different positions but we connect it to the occurrence that occurred first (hence position 3) generating lifecycle 1-3-6.
- **Data correlation** - if the event log contains (event) data attributes, it is possible to connect activity states that share some data values, e.g., an event id

Boundary State Detection

- This step is **to replace activity lifecycles with placeholders** marking the **start and the end of each lifecycle** in the log.
- The formulas directly follow by the corresponding formulas in standard Declare.
- The idea is that, for verifying constraints involving non-atomic activities, **it is sufficient to take into account the boundary states of lifecycles**, abstracting away from the lifecycle detail;
- From the Filtered Log and the Lifecycle specification this step generates a **simplified log** in which **only boundary states** are included.

Boundary State Detection

- This step is **to replace activity lifecycles with placeholders** marking the **start and the end of each lifecycle** in the log.
- The formulas directly follow by the corresponding formulas in standard Declare.
- The idea is that, for verifying constraints involving non-atomic activities, **it is sufficient to take into account the boundary states of lifecycles**, abstracting away from the lifecycle detail;
- From the Filtered Log and the Lifecycle specification this step generates a **simplified log** in which **only boundary states** are included.

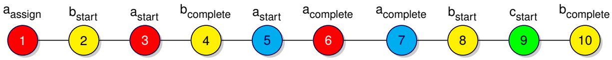
Boundary State Detection

- This step is **to replace activity lifecycles with placeholders** marking the **start and the end of each lifecycle** in the log.
- The formulas directly follow by the corresponding formulas in standard Declare.
- The idea is that, for verifying constraints involving non-atomic activities, **it is sufficient to take into account the boundary states of lifecycles**, abstracting away from the lifecycle detail;
- From the Filtered Log and the Lifecycle specification this step generates a **simplified log** in which **only boundary states** are included.

Boundary State Detection

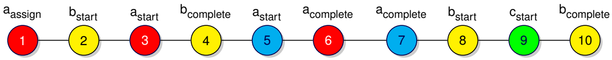
- This step is **to replace activity lifecycles with placeholders** marking the **start and the end of each lifecycle** in the log.
- The formulas directly follow by the corresponding formulas in standard Declare.
- The idea is that, for verifying constraints involving non-atomic activities, **it is sufficient to take into account the boundary states of lifecycles**, abstracting away from the lifecycle detail;
- From the Filtered Log and the Lifecycle specification this step generates a **simplified log** in which **only boundary states** are included.

Discovering inter-lifecycle relations



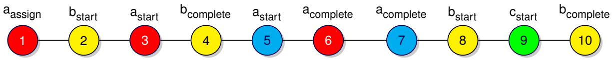
- Boundary state log derived from the previous step is mined using the existing **standard Declare mining approach** presented in a previous work;
- Declare template semantics has been **slightly modified for non-atomic activities** (in the template the initial state is indicated with i and the final one with f);
- The outcome is **a set of candidate Declare constraints connecting elements of different life-cycles** (inter-lifecycle relations). For each candidate, **we extract the set of fulfillments and the set of violations** in the log; ad es:
  - $b_{complete}(4)$ is a fulfillment for constraint $\Box(b_f \rightarrow \Diamond a_i)$
  - $b_{complete}(10)$ is a violation for the same constraint.

Discovering inter-lifecycle relations



- Boundary state log derived from the previous step is mined using the existing **standard Declare mining approach** presented in a previous work;
- Declare template semantics has been **slightly modified for non-atomic activities** (in the template the initial state is indicated with i and the final one with f);
- The outcome is **a set of candidate Declare constraints connecting elements of different life-cycles** (inter-lifecycle relations). For each candidate, **we extract the set of fulfillments and the set of violations** in the log; ad es:
  - $b_{complete}(4)$ is a fulfillment for constraint $\Box(b_f \rightarrow \Diamond a_i)$
  - $b_{complete}(10)$ is a violation for the same constraint.

Discovering inter-lifecycle relations



- Boundary state log derived from the previous step is mined using the existing **standard Declare mining approach** presented in a previous work;
- Declare template semantics has been **slightly modified for non-atomic activities** (in the template the initial state is indicated with i and the final one with f);
- The outcome is **a set of candidate Declare constraints connecting elements of different life-cycles** (inter-lifecycle relations). For each candidate, **we extract the set of fulfillments and the set of violations** in the log; ad es:
  - $b_{complete}(4)$ is a fulfillment for constraint $\Box(b_f \rightarrow \Diamond a_i)$
  - $b_{complete}(10)$ is a violation for the same constraint.

Discovering intra-lifecycle relations

- The problem is then **riformulated as a supervised learning problem**: the features of the lifecycle of activation ***a*** discriminating with respect to fullfillments/violations of constraint ***constr***, are learned **from a set L of sequences** representing all the lifecycles of activation *a* in the log;
- Sequences **are classified in two sets** L_{ful} and L_{viol} according to whether the lifecycle corresponds to a fulfillment or to a violation of *constr*;
- **Each lifecycle is encoded in terms of a set of Declare constraints** as a vector of (boolean) values representing whether all these (intra-lifecycle) constraints are satisfied or not on that lifecycle. The decision tree is trained using the **intra-lifecycle conditions as features** and the **classification of the lifecycle as part of** L_{ful} **or** L_{viol}.

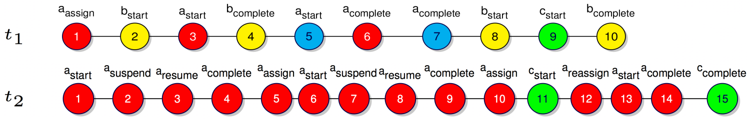
Discovering intra-lifecycle relations

- The problem is then **riformulated as a supervised learning problem**: the features of the lifecycle of activation ***a*** discriminating with respect to fullfillments/violations of constraint ***constr***, are learned **from a set L of sequences** representing all the lifecycles of activation *a* in the log;
- Sequences **are classified in two sets** L_{ful} and L_{viol} according to whether the lifecycle corresponds to a fulfillment or to a violation of *constr*;
- **Each lifecycle is encoded in terms of a set of Declare constraints** as a vector of (boolean) values representing whether all these (intra-lifecycle) constraints are satisfied or not on that lifecycle. The decision tree is trained using the **intra-lifecycle conditions as features** and the **classification of the lifecycle as part of** L_{ful} **or** L_{viol}.

Discovering intra-lifecycle relations

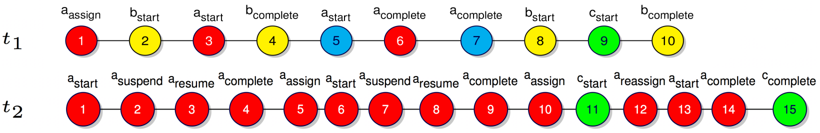
- The problem is then **riformulated as a supervised learning problem**: the features of the lifecycle of activation ***a*** discriminating with respect to fullfillments/violations of constraint ***constr***, are learned **from a set L of sequences** representing all the lifecycles of activation *a* in the log;
- Sequences **are classified in two sets** L_{ful} and L_{viol} according to whether the lifecycle corresponds to a fulfillment or to a violation of *constr*;
- **Each lifecycle is encoded in terms of a set of Declare constraints** as a vector of (boolean) values representing whether all these (intra-lifecycle) constraints are satisfied or not on that lifecycle. The decision tree is trained using the **intra-lifecycle conditions as features** and the **classification of the lifecycle as part of** L_{ful} **or** L_{viol}.

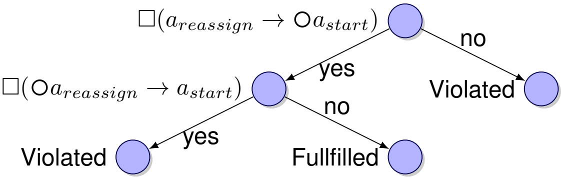
Discovering intra-lifecycle relations



For example, consider constraint $\Box(a_f \rightarrow \Diamond b_i)$ over t_1 and t_2:
$a_1 = \langle a_{start}, a_{complete} \rangle$ $a_2 = \langle a_{assign}, a_{start}, a_{complete} \rangle$ in trace $\mathbf{t}_1$ correspond to **fulfillments** for the constraint (followed by b_i).
$a_3 = \{\langle a_{start}, a_{suspend}, a_{resume}, a_{complete} \rangle$, $a_4 = \langle a_{assign}, a_{start}, a_{suspend}, a_{resume}, a_{complete} \rangle$, and $a_5 = \langle a_{assign}, a_{reassign}, a_{start}, a_{complete} \rangle$ in trace $\mathbf{t}_2$ correspond to **violations**.
Therefore $L_{ful} = \{a_1, a_2\}$ and $L_{viol} = \{a_3, a_4, a_5\}$ and hence it is likely that **the constraint is verified when** a_{start} **is immediately followed by** $a_{complete}$ **but is not immediately preceded by** $a_{reassign}$ in the lifecycle of a.

Discovering intra-lifecycle relations





The Case Study Setup

- We have implemented the approach as a plug-in of the process mining tool ProM.
- The implemented plug-in has been applied to a set of execution logs synthetically generated (to verify its capability to capture known discriminating behaviors);
- To assess the approach it was applied to a real-life log (logs provided by the BPI Challenge 2013).

The Case Study Setup

- We have implemented the approach as a plug-in of the process mining tool ProM.
- The implemented plug-in has been applied to a set of execution logs synthetically generated (to verify its capability to capture known discriminating behaviors);
- To assess the approach it was applied to a real-life log (logs provided by the BPI Challenge 2013).

The Case Study Setup

- We have implemented the approach as a plug-in of the process mining tool ProM.
- The implemented plug-in has been applied to a set of execution logs synthetically generated (to verify its capability to capture known discriminating behaviors);
- To assess the approach it was applied to a real-life log (logs provided by the BPI Challenge 2013).

Synthetic Log 1

Synthetic Log 1 Synthetic Log 1 contains 1000 traces in which *Register* occurs with one of the following possible lifecycles:

- $\langle\ Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{abort}\ \rangle$
- $\langle\ Register_{assign}, Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{start}, Register_{abort}\ \rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{abort}\ \rangle$

- Whenever Register is aborted (see second, fourth and fifth lifecycles in the list), the claimer is notified via phone;
- If the registration completes normally (see first and third lifecycle in the list), the e-mail notification is required;
- Hence whenever Register is aborted, the response constraint $\square(Register \rightarrow \lozenge Notify\_by\_phone)$ is verified, otherwise this constraint does not hold; the intra-lifecycle analysis confirms this giving the constraint **exactly(1,**$Register_{abort}$**)** as discriminative constraints within lifecycle of Register.

Synthetic Log 1

Synthetic Log 1 Synthetic Log 1 contains 1000 traces in which *Register* occurs with one of the following possible lifecycles:

- $\langle\ Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{abort}\ \rangle$
- $\langle\ Register_{assign}, Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{start}, Register_{abort}\ \rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{abort}\ \rangle$

- Whenever Register is aborted (see second, fourth and fifth lifecycles in the list), the claimer is notified via phone;
- If the registration completes normally (see first and third lifecycle in the list), the e-mail notification is required;
- Hence whenever Register is aborted, the response constraint $\Box(Register \rightarrow \Diamond Notify\_by\_phone)$ is verified, otherwise this constraint does not hold; the intra-lifecycle analysis confirms this giving the constraint **exactly(1,**$Register_{abort}$**)** as discriminative constraints within lifecycle of Register.

Synthetic Log 1

Synthetic Log 1 Synthetic Log 1 contains 1000 traces in which $Register$ occurs with one of the following possible lifecycles:

- $\langle\ Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{abort}\ \rangle$
- $\langle\ Register_{assign}, Register_{start}, Register_{complete}\ \rangle$
- $\langle\ Register_{start}, Register_{abort}\ \rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{abort}\ \rangle$

- Whenever Register is aborted (see second, fourth and fifth lifecycles in the list), the claimer is notified via phone;
- If the registration completes normally (see first and third lifecycle in the list), the e-mail notification is required;
- Hence whenever Register is aborted, the response constraint $\Box(Register \rightarrow \Diamond Notify\_by\_phone)$ is verified, otherwise this constraint does not hold; the intra-lifecycle analysis confirms this giving the constraint **exactly(1,**$Register_{abort}$**)** as discriminative constraints within lifecycle of Register.

Synthetic Log 2

Synthetic Log 2 Synthetic Log 2 contains 1000 traces in which the non-atomic activity *Send_questionnaire* occurs with one of the following possible lifecycles:

- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{complete} \rangle$
- $\langle Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{assign}, Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{suspend}, Send\_questionnaire_{abort} \rangle$
- $\langle Send\_questionnaire_{complete} \rangle$

- When *Send_questionnaire* is **withdrawn** or **aborted** (second, third and fourth), *Skip_response* for skipping the response is executed;
- Whenever *Send_questionnaire* **completes normally** (first and fifth), *Skip_response* **is not executed**.
- Therefore, in the log, the response constraint $\Box(Send\_questionnaire \rightarrow \Diamond(Skip\_response))$ is **verified only if** *Send_questionnaire* **does not complete normally** and in this case intra-lifecycle gives: **exclusive-choice(**$Send - questionnaire_{abort}, Send - questionnaire_{withdraw}$**)**.

Synthetic Log 2

Synthetic Log 2 Synthetic Log 2 contains 1000 traces in which the non-atomic activity *Send_questionnaire* occurs with one of the following possible lifecycles:

- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{complete}\rangle$
- $\langle Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{assign}, Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{suspend}, Send\_questionnaire_{abort} \rangle$
- $\langle Send\_questionnaire_{complete} \rangle$

- When *Send_questionnaire* is **withdrawn** or **aborted** (second, third and fourth), *Skip_response* for skipping the response is executed;
- Whenever *Send_questionnaire* **completes normally** (first and fifth), *Skip_response* **is not executed**.
- Therefore, in the log, the response constraint $\Box(Send\_questionnaire \rightarrow \Diamond(Skip\_response))$ is **verified only if** *Send_questionnaire* **does not complete normally** and in this case intra-lifecycle gives: **exclusive-choice(**$Send - questionnaire_{abort}$, $Send - questionnaire_{withdraw}$**)**.

Synthetic Log 2

Synthetic Log 2 Synthetic Log 2 contains 1000 traces in which the non-atomic activity *Send_questionnaire* occurs with one of the following possible lifecycles:

- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{complete} \rangle$
- $\langle Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{assign}, Send\_questionnaire_{withdraw} \rangle$
- $\langle Send\_questionnaire_{start}, Send\_questionnaire_{suspend}, Send\_questionnaire_{abort} \rangle$
- $\langle Send\_questionnaire_{complete} \rangle$

- When *Send_questionnaire* is **withdrawn** or **aborted** (second, third and fourth), *Skip_response* for skipping the response is executed;
- Whenever *Send_questionnaire* **completes normally** (first and fifth), *Skip_response* **is not executed**.
- Therefore, in the log, the response constraint $\Box(Send\_questionnaire \rightarrow \Diamond(Skip\_response))$ is **verified only if** *Send_questionnaire* **does not complete normally** and in this case intra-lifecycle gives: **exclusive-choice(**$Send - questionnaire_{abort}$**,** $Send - questionnaire_{withdraw}$**)**.

Synthetic Log 3

Synthetic Log 3 Synthetic Log 3 contains 1000 traces in which the non-atomic activity *High_medical_history_check* occurs with one of the following possible lifecycles:

- $\langle High\_medical\_history\_check_{withdraw} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{abort} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{autoskip} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete} \rangle$

- When *High_medical_history_check* **does not complete normally** (third and fourth), *Contact_hospital* **is executed eventually**.
- When verification procedure **completes normally** (second and fifth), **there is no need to contact** the hospital.
- Therefore, in the log, the response constraint $\square(High\_medical\_history\_check \rightarrow \Diamond Contact\_hospital)$ **holds if and only if** *High_medical_history_check* **fails**.

Synthetic Log 3

Synthetic Log 3 Synthetic Log 3 contains 1000 traces in which the non-atomic activity *High_medical_history_check* occurs with one of the following possible lifecycles:

- $\langle High\_medical\_history\_check_{withdraw} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{abort} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{autoskip} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{start},$ $High\_medical\_history\_check_{complete} \rangle$

- When *High_medical_history_check* **does not complete normally** (third and fourth), *Contact_hospital* **is executed eventually**.
- When verification procedure **completes normally** (second and fifth), **there is no need to contact** the hospital.
- Therefore, in the log, the response constraint $\Box(High\_medical\_history\_check \rightarrow \Diamond Contact\_hospital)$ **holds if and only if** *High_medical_history_check* **fails**.

Synthetic Log 3

Synthetic Log 3 Synthetic Log 3 contains 1000 traces in which the non-atomic activity *High_medical_history_check* occurs with one of the following possible lifecycles:

- $\langle\, High\_medical\_history\_check_{withdraw}\, \rangle$
- $\langle\, High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete} \rangle$
- $\langle\, High\_medical\_history\_check_{start}, High\_medical\_history\_check_{abort} \rangle$
- $\langle\, High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{autoskip} \rangle$
- $\langle\, High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete}\, \rangle$

- When *High_medical_history_check* **does not complete normally** (third and fourth), *Contact_hospital* **is executed eventually**.
- When verification procedure **completes normally** (second and fifth), **there is no need to contact** the hospital.
- Therefore, in the log, the response constraint $\Box(High\_medical\_history\_check \rightarrow \Diamond Contact\_hospital)$ **holds if and only if** *High_medical_history_check* **fails**.

Synthetic Log 3

Synthetic Log 3 Synthetic Log 3 contains 1000 traces in which the non-atomic activity *High_medical_history_check* occurs with one of the following possible lifecycles:

- $\langle High\_medical\_history\_check_{withdraw} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{complete} \rangle$
- $\langle High\_medical\_history\_check_{start}, High\_medical\_history\_check_{abort} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{autoskip} \rangle$
- $\langle High\_medical\_history\_check_{assign}, High\_medical\_history\_check_{start},$ $High\_medical\_history\_check_{complete} \rangle$

- **For this to be true** the intra-lifecycle relation ¬ **alternate response** between *High_medical_history_check*$_{start}$ and *High_medical_history_check*$_{complete}$ **must hold**.
- The **result is in line with the log intent**: whenever, in the lifecycle of *High_medical_history_check*, there is a *High_medical_history_check*$_{start}$ **that is not followed by** *High_medical_history_check*$_{complete}$, the hospital has to be contacted.

Synthetic Log 3

Synthetic Log 3 Synthetic Log 3 contains 1000 traces in which the non-atomic activity *High_medical_history_check* occurs with one of the following possible lifecycles:

- $\langle$ $High\_medical\_history\_check_{withdraw}$ $\rangle$
- $\langle$ $High\_medical\_history\_check_{start}$, $High\_medical\_history\_check_{complete}$ $\rangle$
- $\langle$ $High\_medical\_history\_check_{start}$, $High\_medical\_history\_check_{abort}$ $\rangle$
- $\langle$ $High\_medical\_history\_check_{assign}$, $High\_medical\_history\_check_{autoskip}$ $\rangle$
- $\langle$ $High\_medical\_history\_check_{assign}$, $High\_medical\_history\_check_{start}$, $High\_medical\_history\_check_{complete}$ $\rangle$

- **For this to be true** the intra-lifecycle relation $\neg$ **alternate response** between $High\_medical\_history\_check_{start}$ and $High\_medical\_history\_check_{complete}$ **must hold**.
- The **result is in line with the log intent**: whenever, in the lifecycle of *High_medical_history_check*, there is a $High\_medical\_history\_check_{start}$ **that is not followed by** $High\_medical\_history\_check_{complete}$, the hospital has to be contacted.

Synthetic Log 4

Synthetic Log 4 Synthetic Log 4 contains 1000 traces in which the non-atomic activity *Register* occurs with one of the following possible lifecycles:

- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{complete} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{abort} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{resume}, Register_{complete} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{withdraw} \rangle$

- When *Register* **is suspended and ends with an abort or a withdraw** (second and fourth lifecycles), or **is suspended (and resumed) more than once** (third lifecycle in the list), the claimer has to be notified via phone;
- **if there is only one suspension correctly resumed** (first lifecycle in the list), the claimer can be notified via e-mail.
- In these cases the response constraint $\Box(Register \rightarrow \Diamond Notify\_by\_phone)$ is verified, otherwise not.
- The discriminative rule discovered for the verification of this constraint is the **alternate succession between** $Register_{resume}$ **and** $Register_{complete}$.

Synthetic Log 4

Synthetic Log 4 Synthetic Log 4 contains 1000 traces in which the non-atomic activity *Register* occurs with one of the following possible lifecycles:

- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{complete} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{abort} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{resume}, Register_{complete} \rangle$
- $\langle Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{withdraw} \rangle$

- When *Register* **is suspended and ends with an abort or a withdraw** (second and fourth lifecycles), or **is suspended (and resumed) more than once** (third lifecycle in the list), the claimer has to be notified via phone;
- **if there is only one suspension correctly resumed** (first lifecycle in the list), the claimer can be notified via e-mail.
- In these cases the response constraint $\Box(Register \rightarrow \Diamond Notify\_by\_phone)$ is verified, otherwise not.
- The discriminative rule discovered for the verification of this constraint is the **alternate succession between** $Register_{resume}$ **and** $Register_{complete}$.

Synthetic Log 4

Synthetic Log 4 Synthetic Log 4 contains 1000 traces in which the non-atomic activity *Register* occurs with one of the following possible lifecycles:

- $\langle\, Register_{start}, Register_{suspend}, Register_{resume}, Register_{complete}\, \rangle$
- $\langle\, Register_{start}, Register_{suspend}, Register_{abort}\, \rangle$
- $\langle\, Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{resume}, Register_{complete} \rangle$
- $\langle\, Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{withdraw}\, \rangle$

- When *Register* **is suspended and ends with an abort or a withdraw** (second and fourth lifecycles), or **is suspended (and resumed) more than once** (third lifecycle in the list), the claimer has to be notified via phone;
- **if there is only one suspension correctly resumed** (first lifecycle in the list), the claimer can be notified via e-mail.
- In these cases the response constraint $\square(Register \rightarrow \lozenge Notify\_by\_phone)$ is verified, otherwise not.
- The discriminative rule discovered for the verification of this constraint is the **alternate succession between** $Register_{resume}$ **and** $Register_{complete}$.

Synthetic Log 4

Synthetic Log 4 Synthetic Log 4 contains 1000 traces in which the non-atomic activity *Register* occurs with one of the following possible lifecycles:

- $\langle\ Register_{start}, Register_{suspend}, Register_{resume}, Register_{complete}\ \rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{abort}\ \rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{resume}, Register_{complete}\rangle$
- $\langle\ Register_{start}, Register_{suspend}, Register_{resume}, Register_{suspend}, Register_{withdraw}\ \rangle$

- When *Register* **is suspended and ends with an abort or a withdraw** (second and fourth lifecycles), or **is suspended (and resumed) more than once** (third lifecycle in the list), the claimer has to be notified via phone;
- **if there is only one suspension correctly resumed** (first lifecycle in the list), the claimer can be notified via e-mail.
- In these cases the response constraint $\square(Register \rightarrow \lozenge Notify\_by\_phone)$ is verified, otherwise not.
- The discriminative rule discovered for the verification of this constraint is the **alternate succession between** $Register_{resume}$ **and** $Register_{complete}$.

BPI Challenge 2013 Results

Table 4: BPI 2013 Results.

	INTER-LIFECYCLE RELATION	INTRA-LIFECYCLE RELATION	CLASS PROB.	SUPPORT
(1)	precedence ($Accepted$,$Completed$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.65	3711
(2)	precedence ($Queued$,$Accepted$)	init($Accepted_{Assigned}$) $\vee$ init ($Accepted_{Wait\_implementation}$)	0.75	92
(3)	precedence ($Queued$, $Completed$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4551
(4)	response ($Completed$, $Queued$)	$\neg$ responded existence ($Completed_{Resolved}$,$Completed_{Closed}$)	0.98	7570
(5)	responded existence ($Completed$, $Accepted$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.66	3771
(6)	responded existence ($Completed$, $Queued$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4595

- The first row of the table (1) suggests as discriminating behavior for *Completed* to be preceded by *Accepted* is when either $Completed_{Cancelled}$ or $Completed_{Closed}$ occurs in the lifecycle of *Completed*.

BPI Challenge 2013 Results

Table 4: BPI 2013 Results.

	INTER-LIFECYCLE RELATION	INTRA-LIFECYCLE RELATION	CLASS PROB.	SUPPORT
(1)	precedence ($Accepted$,$Completed$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.65	3711
(2)	precedence ($Queued$,$Accepted$)	init($Accepted_{Assigned}$) $\vee$ init ($Accepted_{Wait\_implementation}$)	0.75	92
(3)	precedence ($Queued$, $Completed$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4551
(4)	response ($Completed$, $Queued$)	$\neg$ responded existence ($Completed_{Resolved}$,$Completed_{Closed}$)	0.98	7570
(5)	responded existence ($Completed$, $Accepted$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.66	3771
(6)	responded existence ($Completed$, $Queued$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4595

- The same rule is also discriminating for responded existence between *Completed* and *Accepted* (expressing that whenever $Completed$ occurs, then, *Accepted* has to occur in the future or has already occurred before).

BPI Challenge 2013 Results

Table 4: BPI 2013 Results.

	INTER-LIFECYCLE RELATION	INTRA-LIFECYCLE RELATION	CLASS PROB.	SUPPORT
(1)	precedence ($Accepted$,$Completed$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.65	3711
(2)	precedence ($Queued$,$Accepted$)	init($Accepted_{Assigned}$) $\vee$ init ($Accepted_{Wait\_implementation}$)	0.75	92
(3)	precedence ($Queued$, $Completed$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4551
(4)	response ($Completed$, $Queued$)	$\neg$ responded existence ($Completed_{Resolved}$,$Completed_{Closed}$)	0.98	7570
(5)	responded existence ($Completed$, $Accepted$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.66	3771
(6)	responded existence ($Completed$, $Queued$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4595

- In the second row (2), whenever the lifecycle of *Accepted* **starts with** $Accepted_{Assigned}$ **or with** $Accepted_{Wait\_Implementation}$, *Accepted* **is preceded by** *Queued*. These results are the ones with the lowest class probability and support. (the support of (2) indicates that the cases in which the constraint and the discovered discriminative rule are both verified are 92).
- On the other hand, the remaining rules present both a reasonable class probability (> 0.8) and a good support (> 4500).

BPI Challenge 2013 Results

Table 4: BPI 2013 Results.

	INTER-LIFECYCLE RELATION	INTRA-LIFECYCLE RELATION	CLASS PROB.	SUPPORT
(1)	precedence ($Accepted$,$Completed$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.65	3711
(2)	precedence ($Queued$,$Accepted$)	init($Accepted_{Assigned}$) $\vee$ init ($Accepted_{Wait\_implementation}$)	0.75	92
(3)	precedence ($Queued$, $Completed$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4551
(4)	response ($Completed$, $Queued$)	$\neg$ responded existence ($Completed_{Resolved}$,$Completed_{Closed}$)	0.98	7570
(5)	responded existence ($Completed$, $Accepted$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.66	3771
(6)	responded existence ($Completed$, $Queued$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4595

- In particular, the **co-occurrence of *Completed*$_{In\_Call}$ and *Completed*$_{Cancelled}$ discriminates both on the precedence (3) and on the responded existence (6) between *Queued* and *Completed***, i.e., whenever both *In_Call* and *Cancelled* occur in the lifecycle of *Completed*, it means that *Completed* is preceded by *Queued* or, more in general (with a slightly higher support), that *Queued* occurs at least once before or after $Completed$.

BPI Challenge 2013 Results

Table 4: BPI 2013 Results.

	INTER-LIFECYCLE RELATION	INTRA-LIFECYCLE RELATION	CLASS PROB.	SUPPORT
(1)	precedence ($Accepted$,$Completed$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.65	3711
(2)	precedence ($Queued$,$Accepted$)	init($Accepted_{Assigned}$) $\vee$ init ($Accepted_{Wait\_implementation}$)	0.75	92
(3)	precedence ($Queued$, $Completed$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4551
(4)	response ($Completed$, $Queued$)	¬ responded existence ($Completed_{Resolved}$,$Completed_{Closed}$)	0.98	7570
(5)	responded existence ($Completed$, $Accepted$)	not responded existence ($Completed_{Cancelled}$,$Completed_{Closed}$)	0.66	3771
(6)	responded existence ($Completed$, $Queued$)	co-existence ($Completed_{In\_call}$,$Completed_{Cancelled}$)	0.8	4595

- Finally, the discovered discriminating behavior with the highest class probability (almost 1) and support (more than 7000) is the one related to the response constraint between *Completed* and *Queued* (6);
- in this case *Queued* eventually follows $Completed$ if and only if only one among *Completed*$_{Resolved}$ and *Completed*$_{Closed}$ occurs in the lifecycle of *Completed*.

Conclusions and Future Work

- This paper presents **an approach for the discovery of declarative process models from logs containing non-atomic activities**.
- **Discriminative rule mining is used** to characterize the lifecycle of each constraint activation and discriminate between fulfillments and violations of the constraint under examination.
- Our **experiments show the effectiveness** of the approach and its applicability in real-life scenarios.
- As future work, we will conduct a **wider experimentation of the proposed framework on several case studies in real-life scenarios** and **different transactional models** for activity lifecycles.

Thank you for listening!

Any questions?