THE UNIVERSITY OF MELBOURNE

Automated Repair of Process Models Using Non-Local Constraints

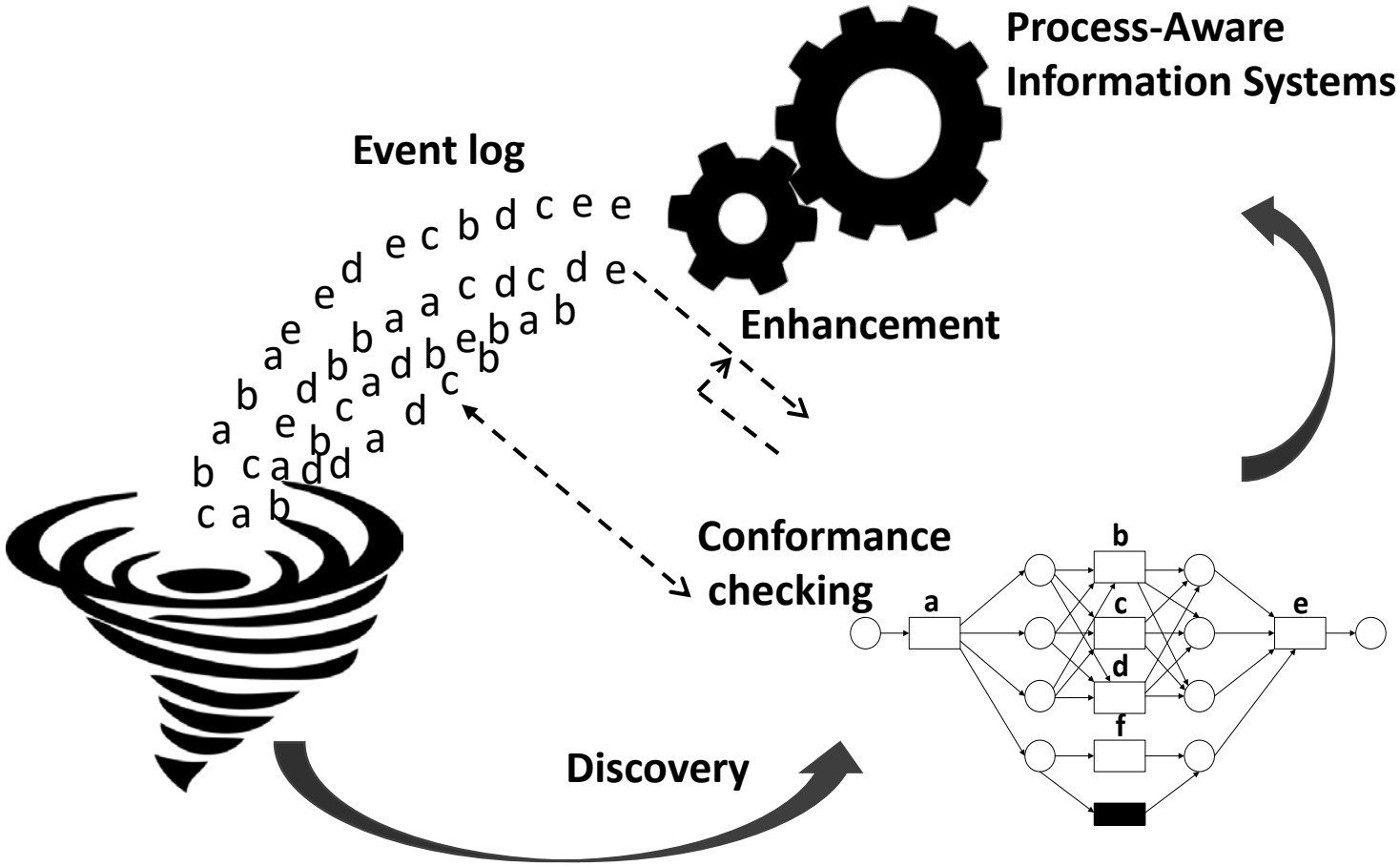
Anna Kalenkova*, Josep Carmona**, Artem Polyvyanyy*, Marcello La Rosa*

* **The University of Melbourne**

** **Polytechnic University of Catalonia**

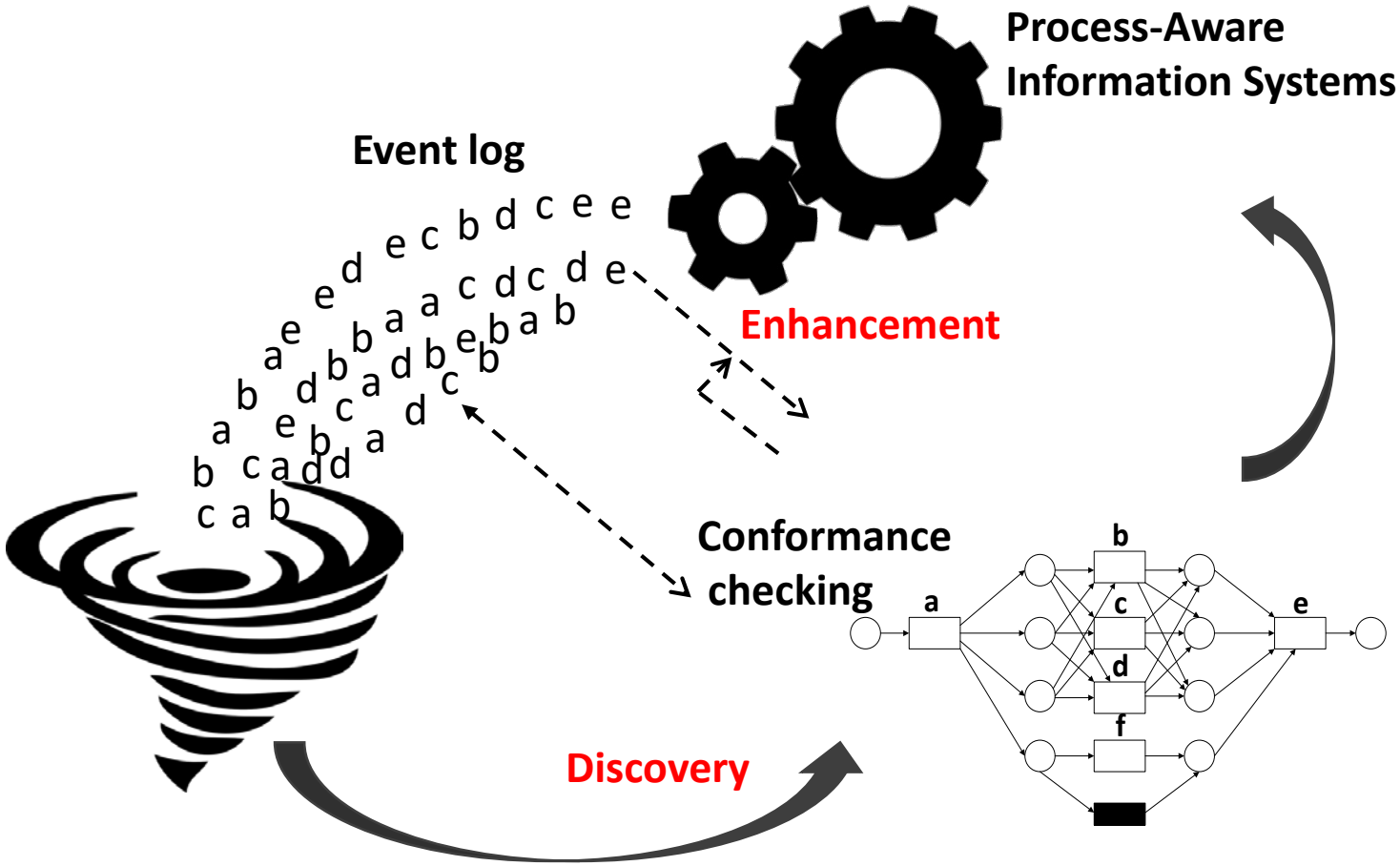
Process mining

- Discovery
- Conformance checking
- Enhancement



Process mining

- Discovery
- Conformance checking
- Enhancement



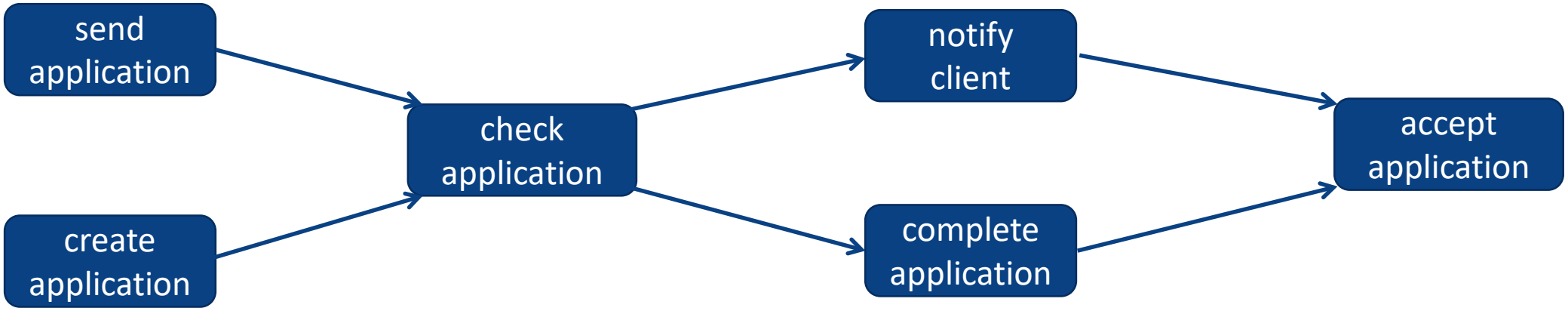
Discovery. Event log

Case ID	Activity name	Timestamp
1	*send application*	*2019-10-28T10:02:30*
2	*create application*	*2019-10-28T10:03:17*
1	*check application*	*2019-10-28T10:03:58*
1	*notify client*	*2019-10-28T10:04:20*
1	*accept application*	*2019-10-28T10:04:25*
2	*check application*	*2019-10-28T14:32:51*
2	*complete application*	*2019-10-29T09:45:13*
2	*accept application*	*2019-10-29T09:50:45*

$$L = \{\langle send\ application, check\ application, notify\ client, accept\ application\rangle, \langle create\ application, check\ application, complete\ application, accept\ application\rangle\}.$$

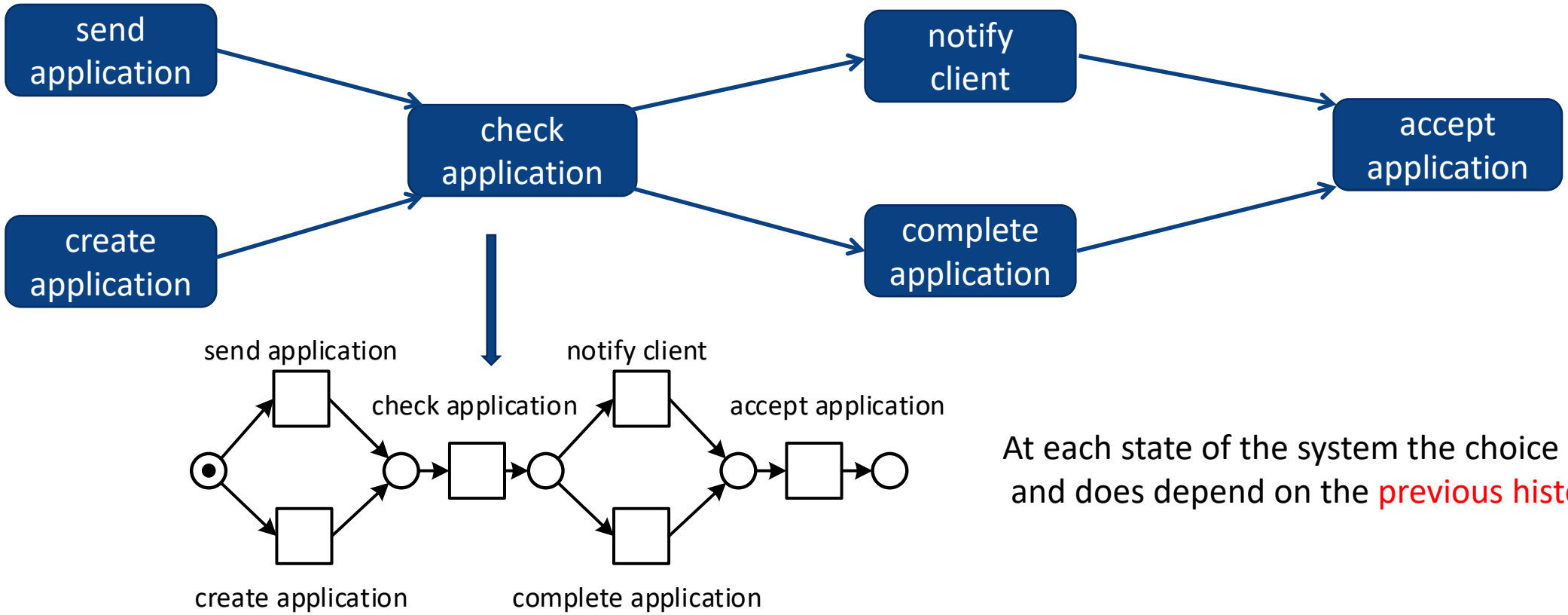
Discovery. Directly-Follows Graphs (DFGs)

$$L = \{\langle send\ application, check\ application, notify\ client, accept\ application\rangle, \langle create\ application, check\ application, complete\ application, accept\ application\rangle\}.$$



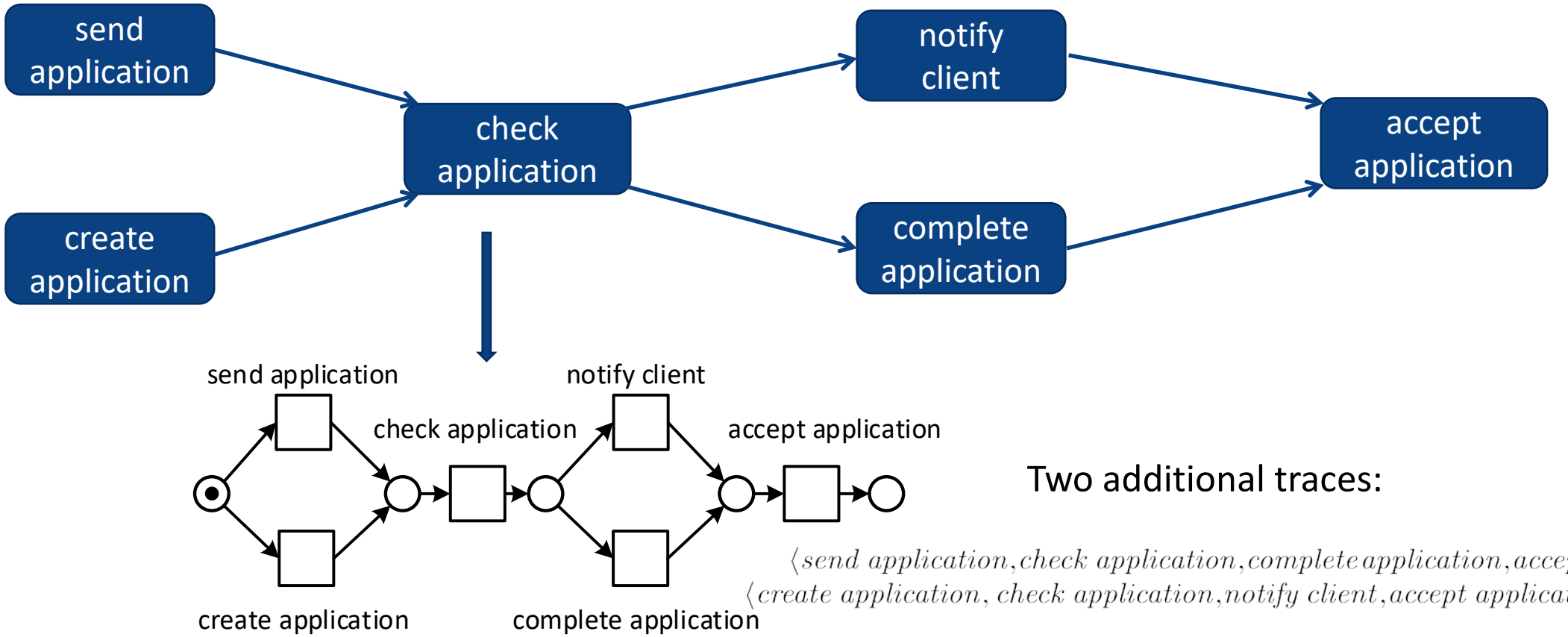
DFGs are used as a final process representation in commercial process mining tools. As well as an intermediate process representation within scalable process mining algorithms, e.g., Inductive miner and Split miner that discover free-choice nets such as BPMN models.

Discovery. Directly-Follows Graphs (DFGs)



At each state of the system the choice is free and does depend on the previous history.

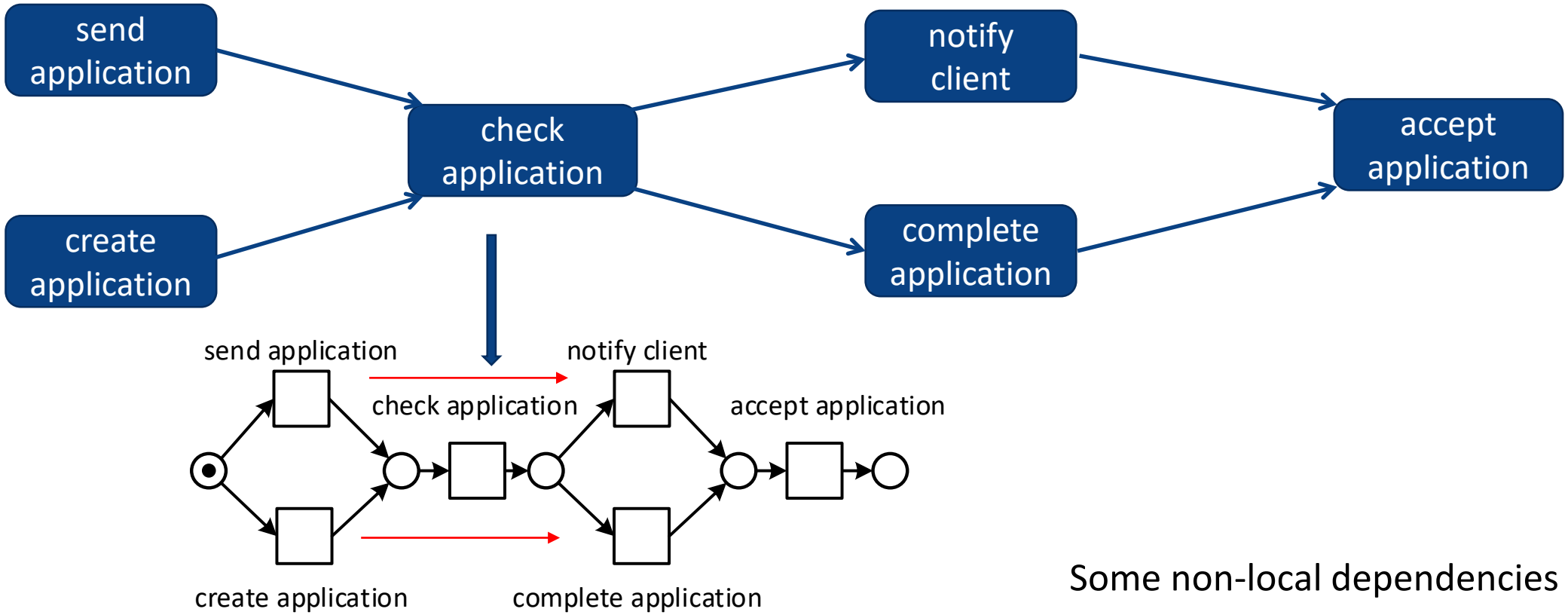
Discovery. Directly-Follows Graphs (DFGs)



Two additional traces:

$\langle send\ application, check\ application, complete\ application, accept\ application \rangle$,
$\langle create\ application, check\ application, notify\ client, accept\ application \rangle$.

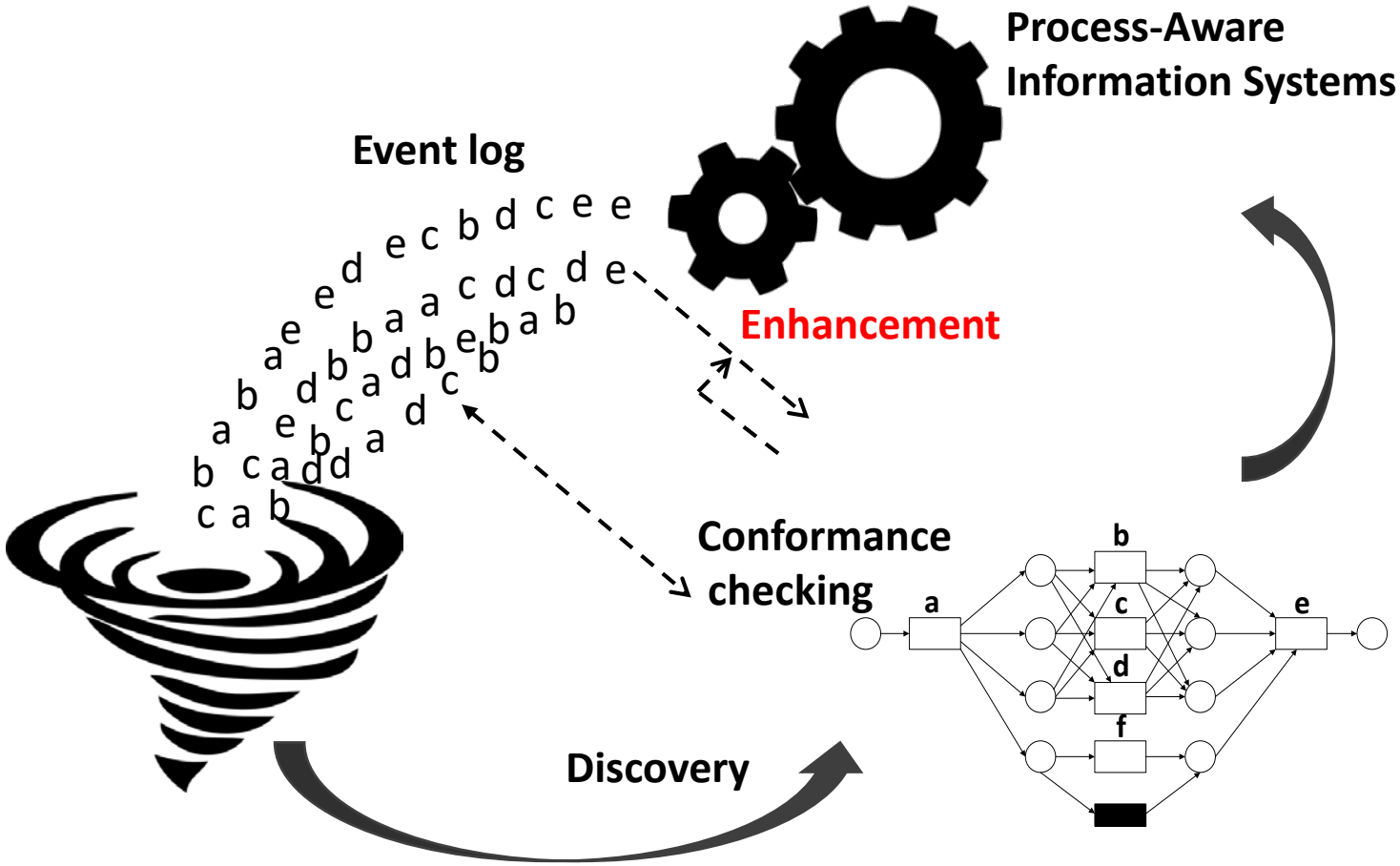
Discovery. Directly-Follows Graphs (DFGs)



Some non-local dependencies may be lost.

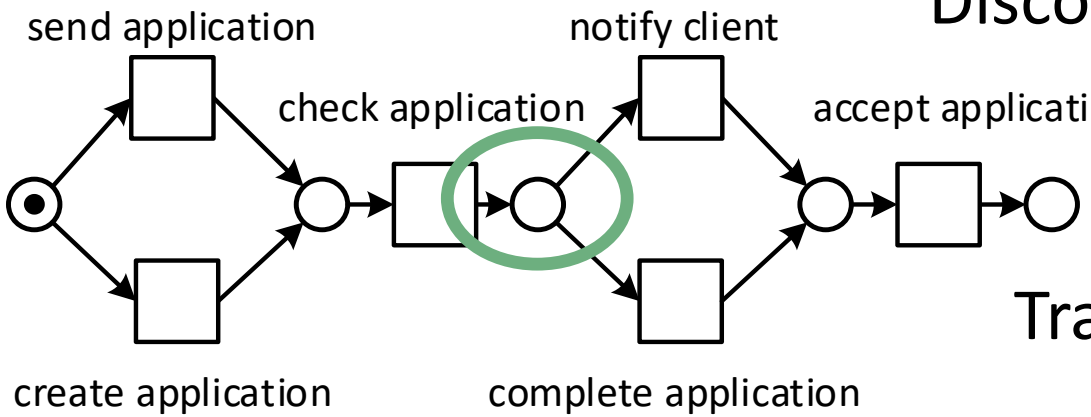
Process mining

- Discovery
- Conformance checking
- Enhancement
  - Repair

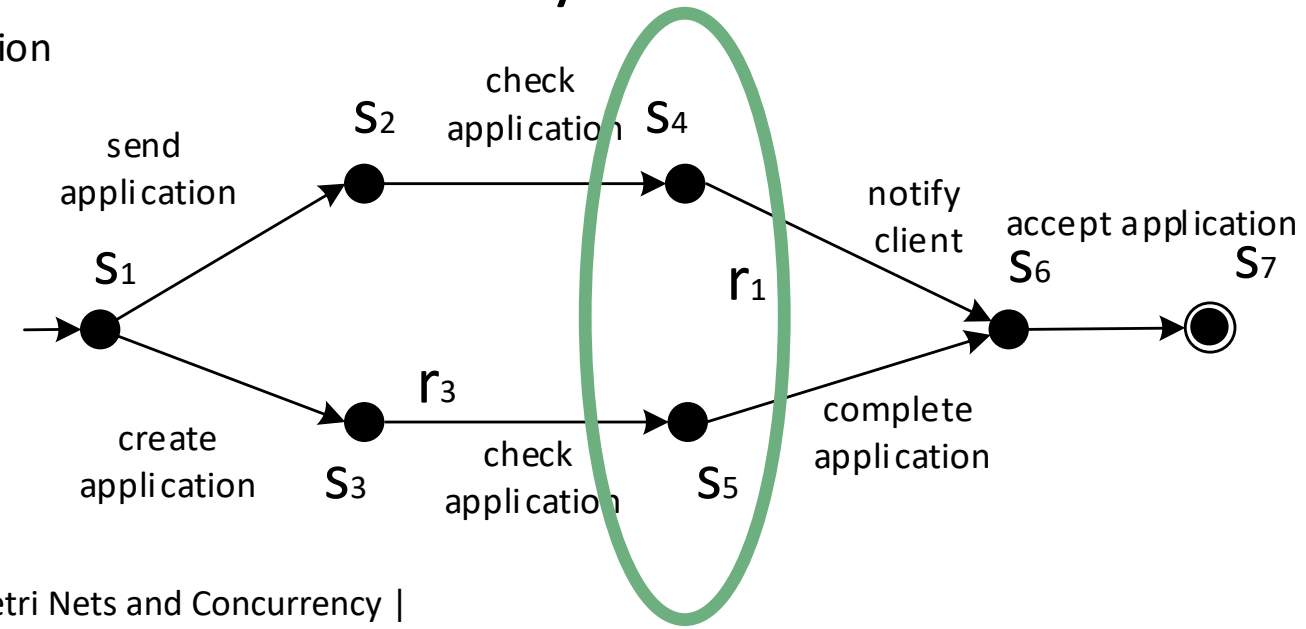


False Free-Choice Relation

Discovered process model



Transition system that encodes event log L



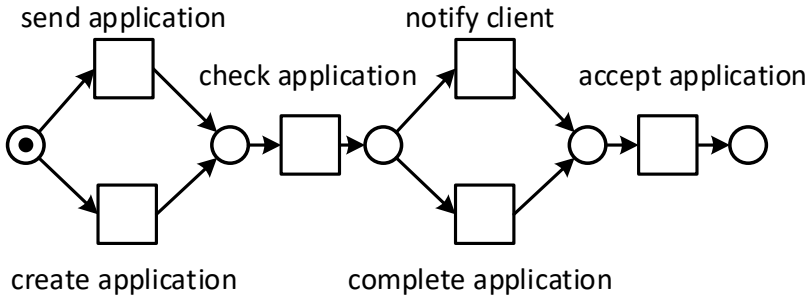
The local choice may depend on the process history

Repair Using Non-Local Constraints

Event log

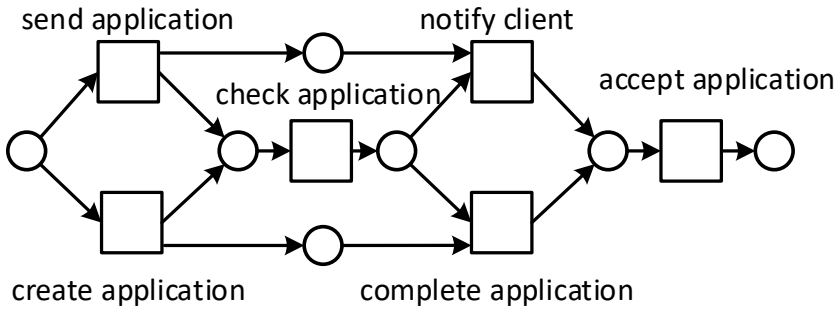
$$L = \{\langle send\ application, check\ application, notify\ client, accept\ application\rangle, \langle create\ application, check\ application, complete\ application, accept\ application\rangle\}.$$

Free-choice process model



Non-local constraints

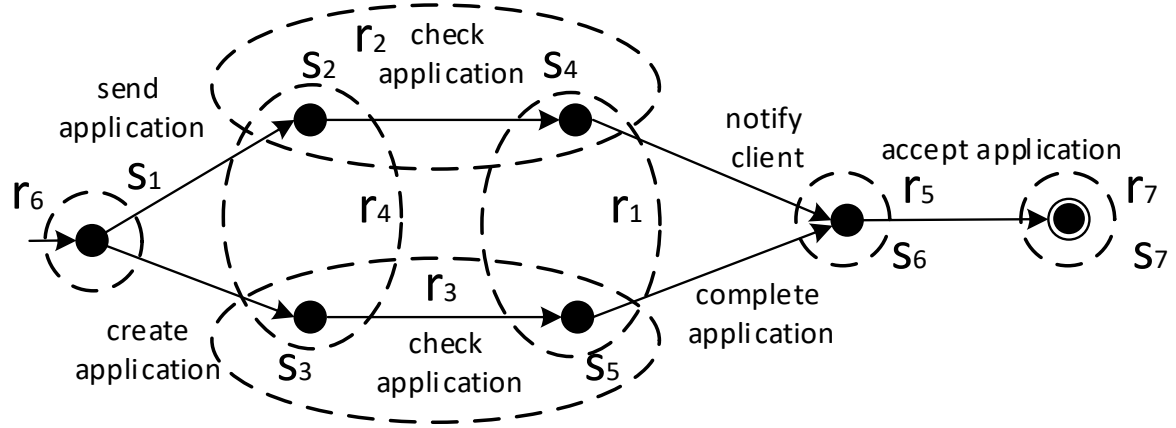
Process model with non-local constraints



State-based Region Synthesis

Let $TS = (S, E, T, s_i, S_f)$ be a transition system and $r \subseteq S$ be a subset of states. Subset r is a **region** iff for each event $e \in E$ one of the following conditions holds:

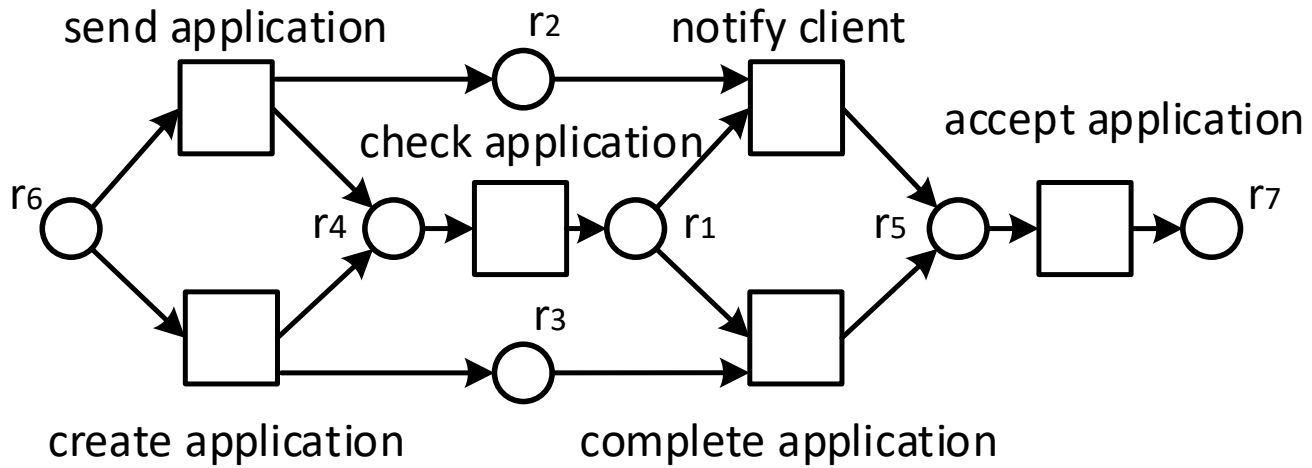
- all the transitions $s_1 \xrightarrow{e} s_2$ **enter** r, i.e., $s_1 \notin r$ and $s_2 \in r$,
- all the transitions $s_1 \xrightarrow{e} s_2$ **exit** r, i.e., $s_1 \in r$ and $s_2 \notin r$,
- all the transitions $s_1 \xrightarrow{e} s_2$ **do not cross** r, i.e., $s_1, s_2 \in r$ or $s_1, s_2 \notin r$.



State-based Region Synthesis

Let $TS = (S, E, T, s_i, S_f)$ be a transition system and $r \subseteq S$ be a subset of states. Subset r is a **region** iff for each event $e \in E$ one of the following conditions holds:

- all the transitions $s_1 \xrightarrow{e} s_2$ **enter** r, i.e., $s_1 \notin r$ and $s_2 \in r$,
- all the transitions $s_1 \xrightarrow{e} s_2$ **exit** r, i.e., $s_1 \in r$ and $s_2 \notin r$,
- all the transitions $s_1 \xrightarrow{e} s_2$ **do not cross** r, i.e., $s_1, s_2 \in r$ or $s_1, s_2 \notin r$.



State-based Region Synthesis

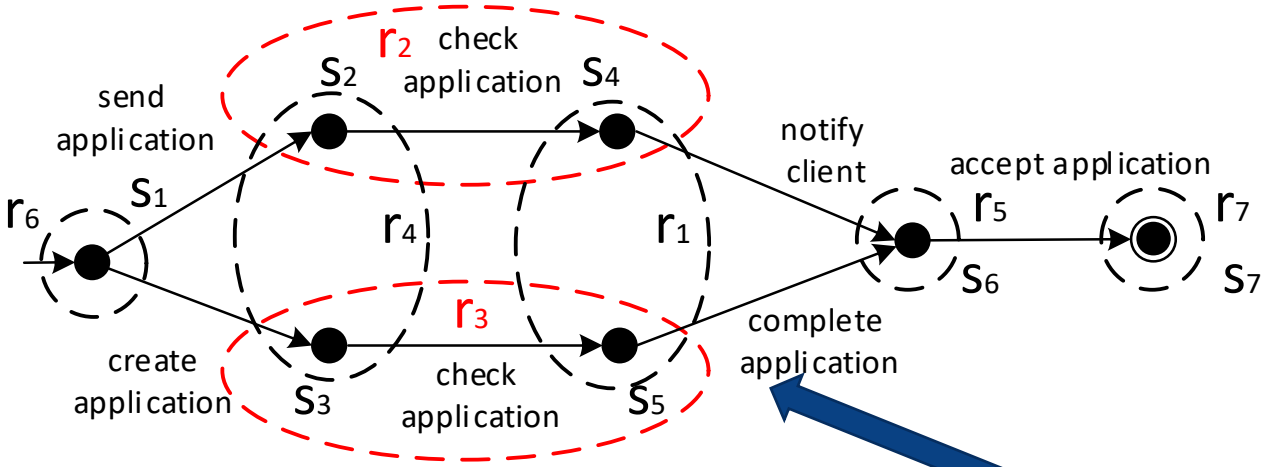
Region r *separates* two different states $s, s' \in S$, $s \neq s'$, iff $s \in r$ and $s' \notin r$. Finding such a region is the *state separation problem* between s and s' and is denoted by $SSP(s, s')$.

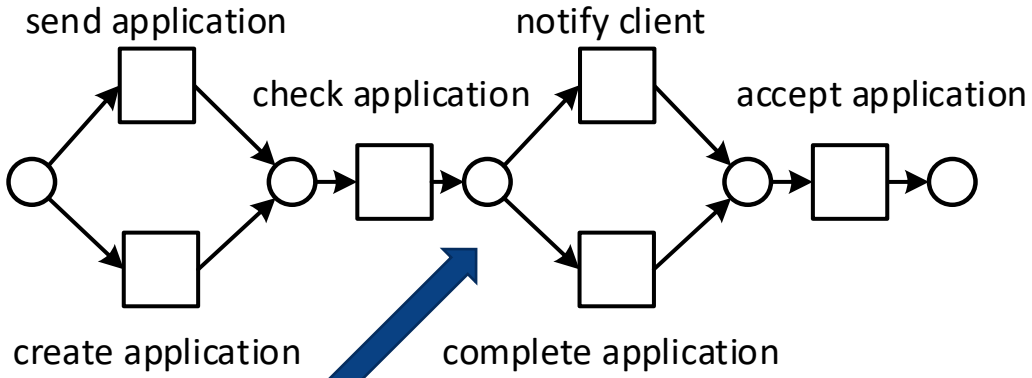
When an event e is not enabled in a state s, i.e., $s \not\rightarrow e$, a region r, containing s may be found, such that e does not *exit* r. Finding such a region is known as the *event/state separation problem* between s and e and is denoted by $ESSP(s, e)$.

Theorem. A TS can be synthesized into a safe Petri net N such that the reachability graph of N is isomorphic to TS if all SSP and $ESSP$ problems are solvable.

State-based Region Synthesis

When an event e is not enabled in a state s, i.e., $s \not\rightarrow e$, a region r, containing s may be found, such that e does not *exit* r. Finding such a region is known as the *event/state separation problem* between s and e and is denoted by $ESSP(s, e)$.





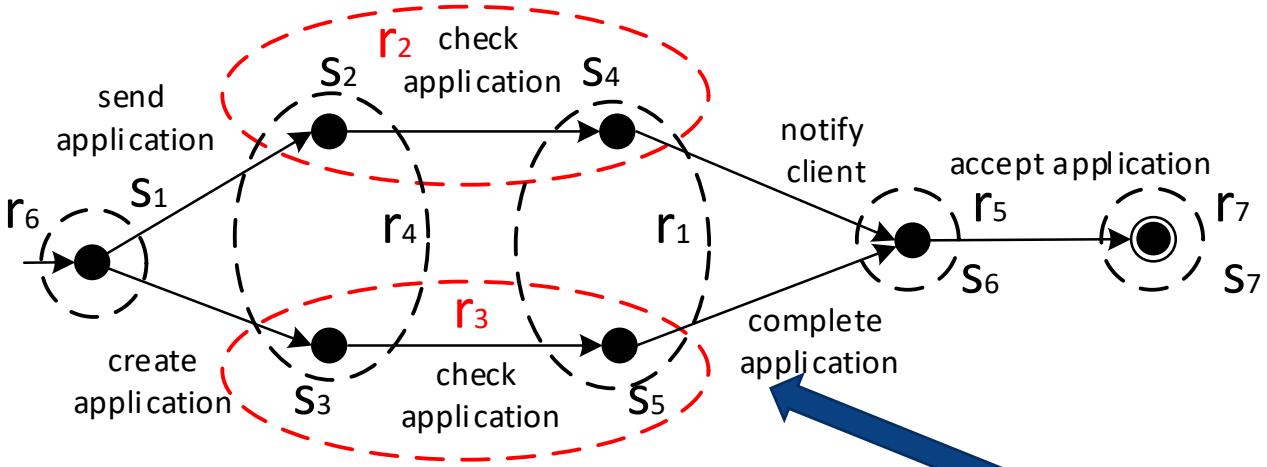
ESSP(s4, “complete application”)?

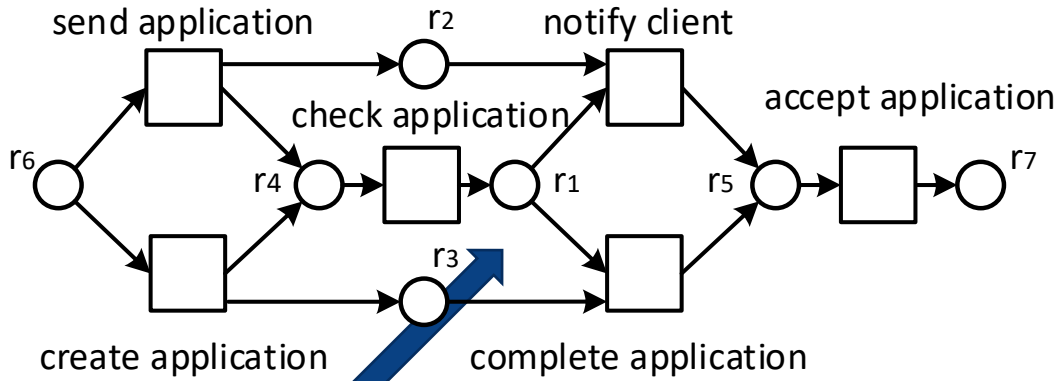
ESSP(s5, “notify client”)?

False free-choice relation

State-based Region Synthesis

When an event e is not enabled in a state s, i.e., $s \not\rightarrow e$, a region r, containing s may be found, such that e does not *exit* r. Finding such a region is known as the *event/state separation problem* between s and e and is denoted by $ESSP(s, e)$.





ESSP(s4, “complete application”)?

ESSP(s5, “notify client”)?

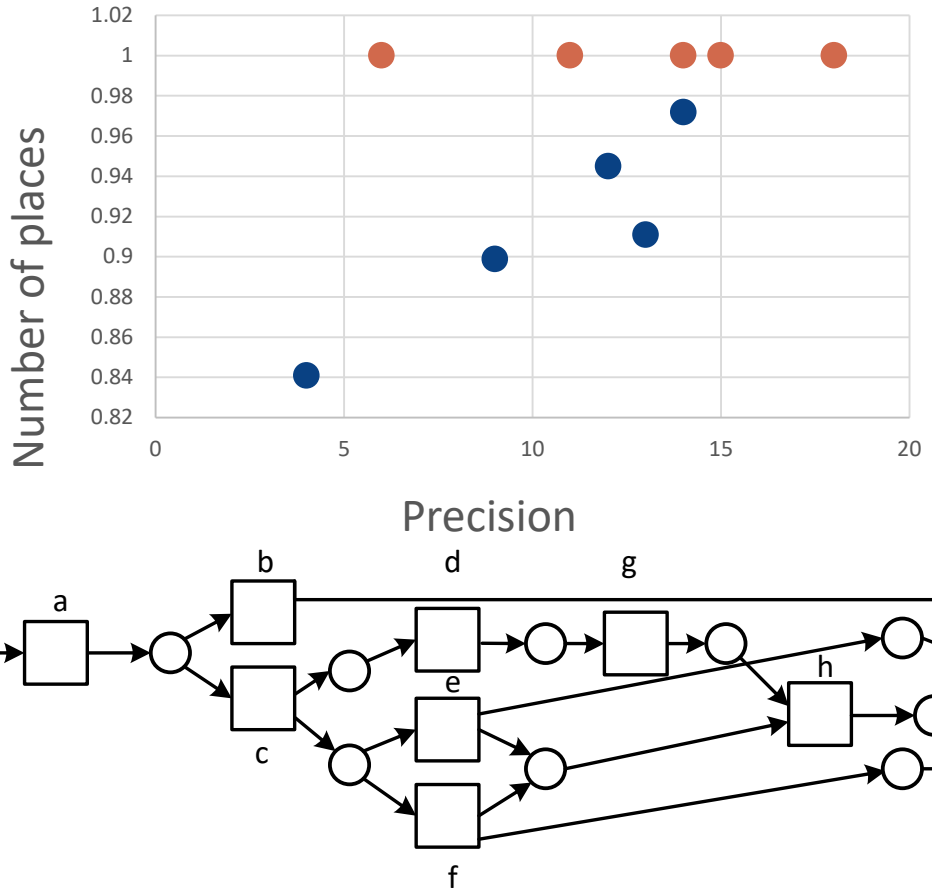
False free-choice relation

Formal Properties of Repair Algorithm

Threorem (Fitness). Let $\sigma \in L$ be a trace of an event log $L \in E^*$, and $N = (P, T, F, l)$, $l : T \to E$ be a free-choice workflow net, such that its language contains σ, i.e., $\sigma \in \mathcal{L}(N)$. Workflow net $N' = (P \cup P', T, F', l)$, $l : T \to E$, is obtained from N and L using Repair algorithm. Then the language of N' contains σ, i.e., $\sigma \in \mathcal{L}(N')$.

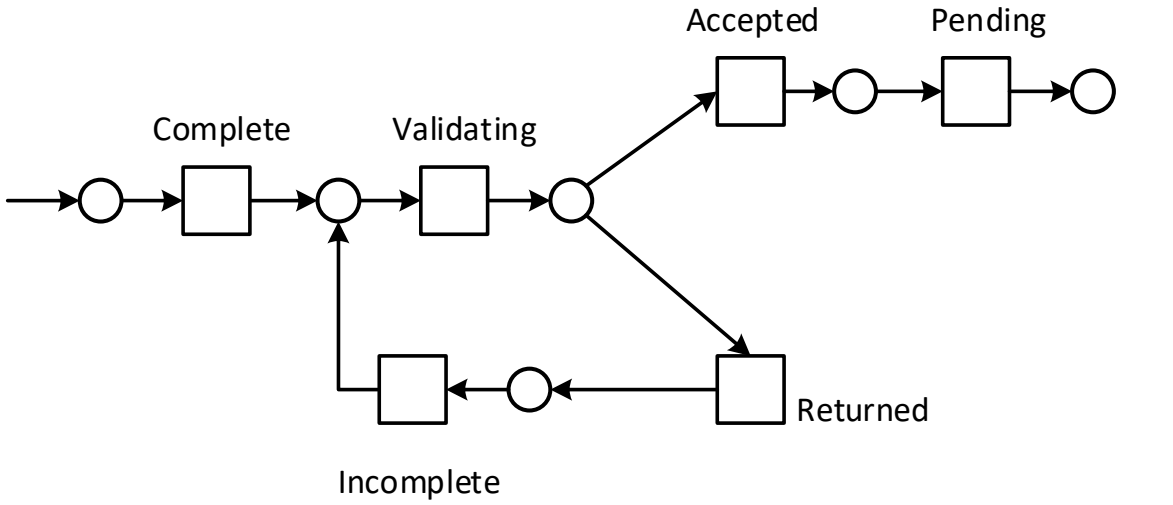
Theorem (Precision). Let $N = (P, T, F, l)$, $l : T \to E$, be a free-choice workflow net and let L be an event log over set of events E. If workflow net N' is obtained from N and L by Repair algorithm, then the language of N contains the language of N', i.e., $\mathcal{L}(N') \subseteq \mathcal{L}(N)$.

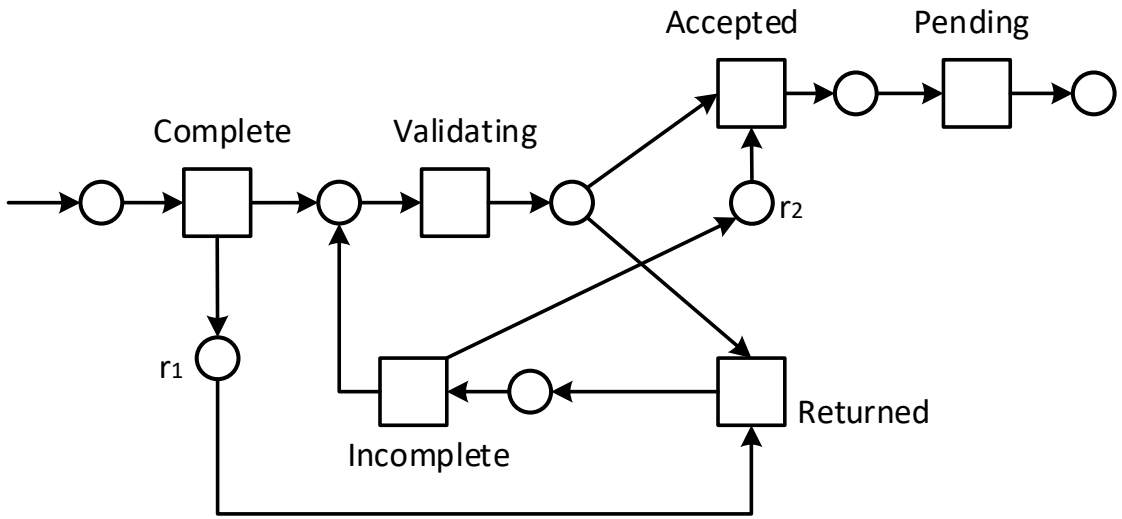
Evaluation



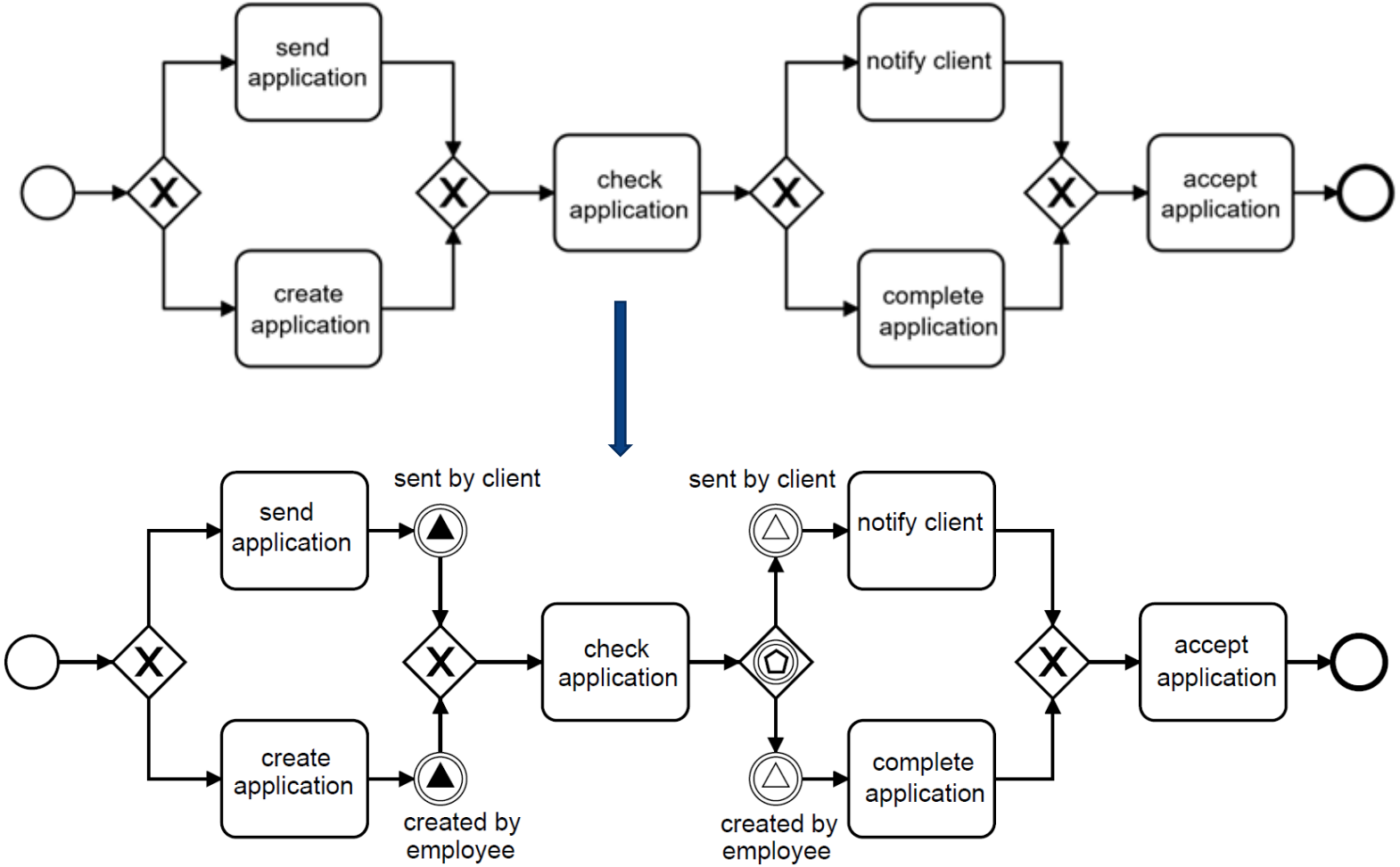
The plugin is open-source and was implemented in Apromore Community Edition

Evaluation


Accepted
Pending
Complete
Validating
Returned
Incomplete


Accepted
Pending
Complete
Validating
r2
r1
Returned
Incomplete

BPMN (Business Process Model and Notation)



Conclusion and Future work

Although the Petri net synthesis problem is reduced (only some constraints are to be found), it still remains NP-complete.

Thus, an extension of the proposed algorithm is to be advised to improve its performance characteristics.

THE UNIVERSITY OF MELBOURNE

Thank you