Enhancing workflow-nets with data for trace completion

R. De Masellis, C. Di Francescomarino, C. Ghidini, S. Tessaris

FBK-IRST and Free University of Bolzano-Bozen

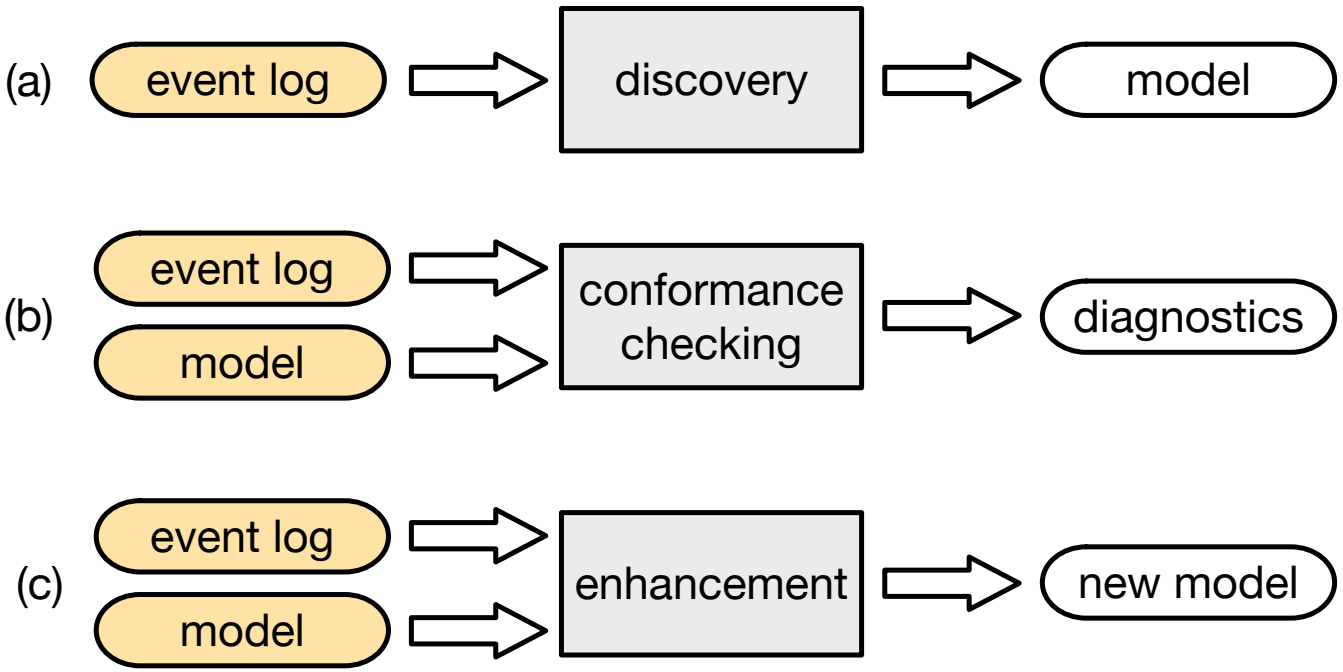
KAOS project meeting
May 26, 2017

Talk outline

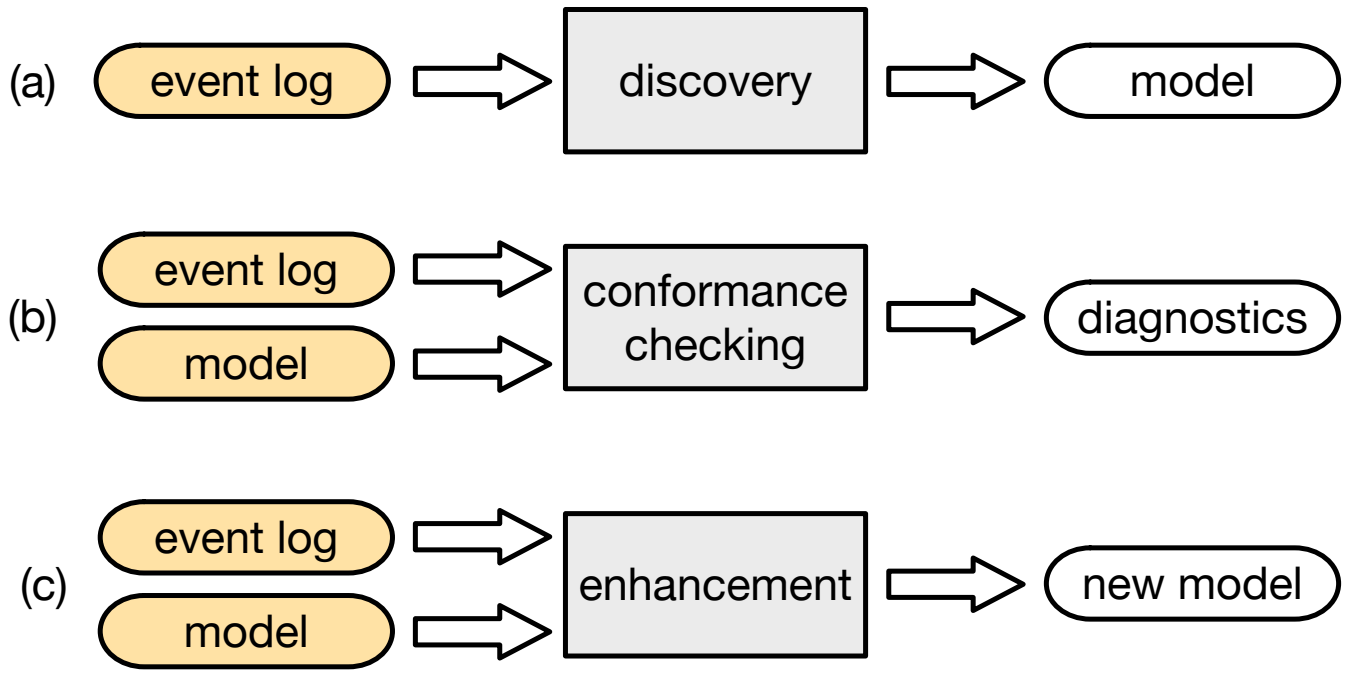
1. Problem
2. Framework
3. Technique

Talk outline

Process mining

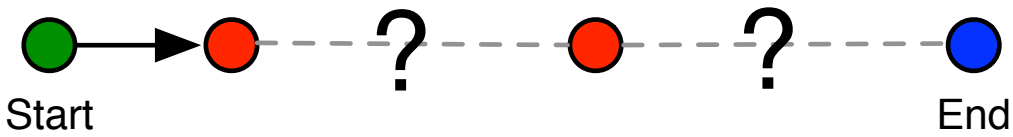


Process mining



Logs play a fundamental role!

What if logs are *incomplete*?



Causes:

- noise on data;
- human tasks;
- bad/no fully-automated logging policies;
- ...

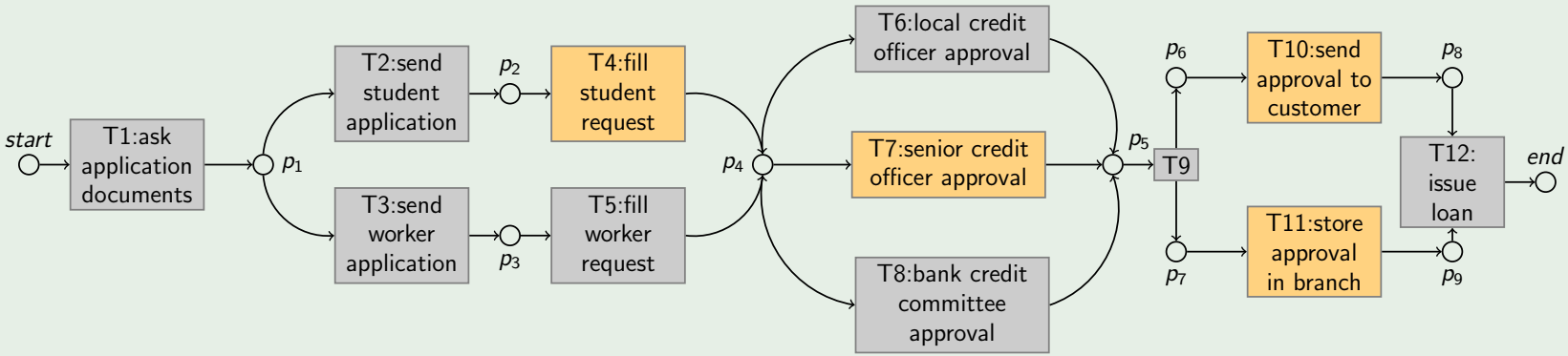
We “repair” it!

How?

We assume to have a prescriptive process model and find its possible complete executions *compliant* with the trace.

Example (easy)

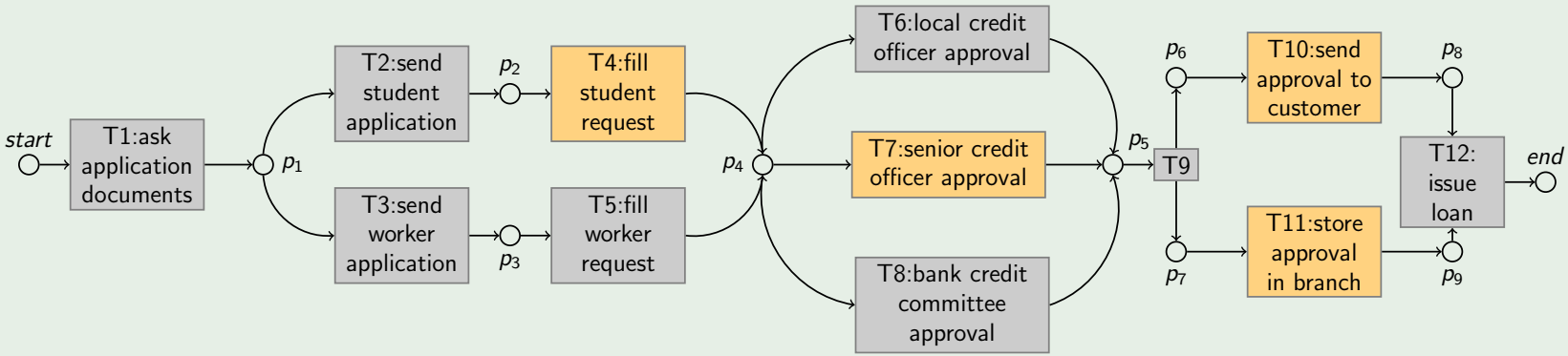
Example



$$\ldots \rightarrow T4 \rightarrow T7 \rightarrow T10 \rightarrow T11 \rightarrow \ldots$$

Example (easy)

Example



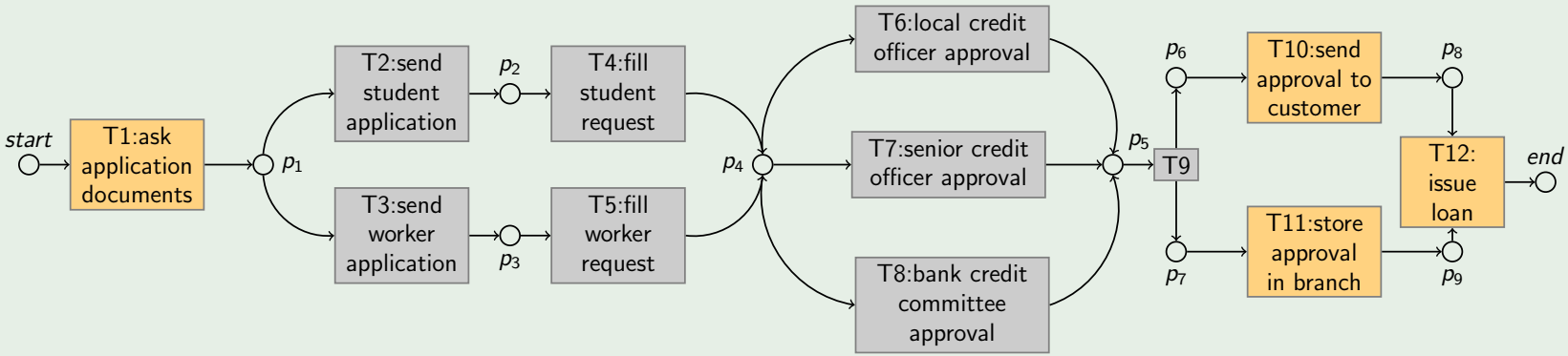
$$\ldots \rightarrow T4 \rightarrow T7 \rightarrow T10 \rightarrow T11 \rightarrow \ldots$$

$$\Downarrow$$

$$start \rightarrow T2 \rightarrow T4 \rightarrow T7 \rightarrow T10 \rightarrow T11 \rightarrow T12 \rightarrow end$$

Example (hard)

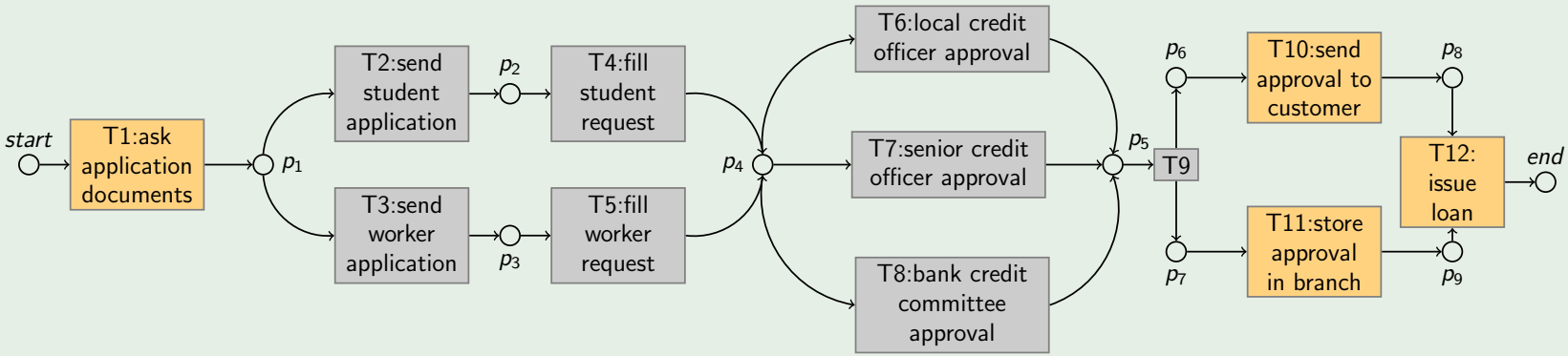
Example



$$start \rightarrow T1 \rightarrow \ldots \rightarrow T10 \rightarrow T11 \rightarrow T12 \rightarrow end$$

Example (hard)

Example



$$start \rightarrow T1 \rightarrow \ldots \rightarrow T10 \rightarrow T11 \rightarrow T12 \rightarrow end$$

$$\Downarrow$$

$$?$$

Contribution

Equip the process model with data information:

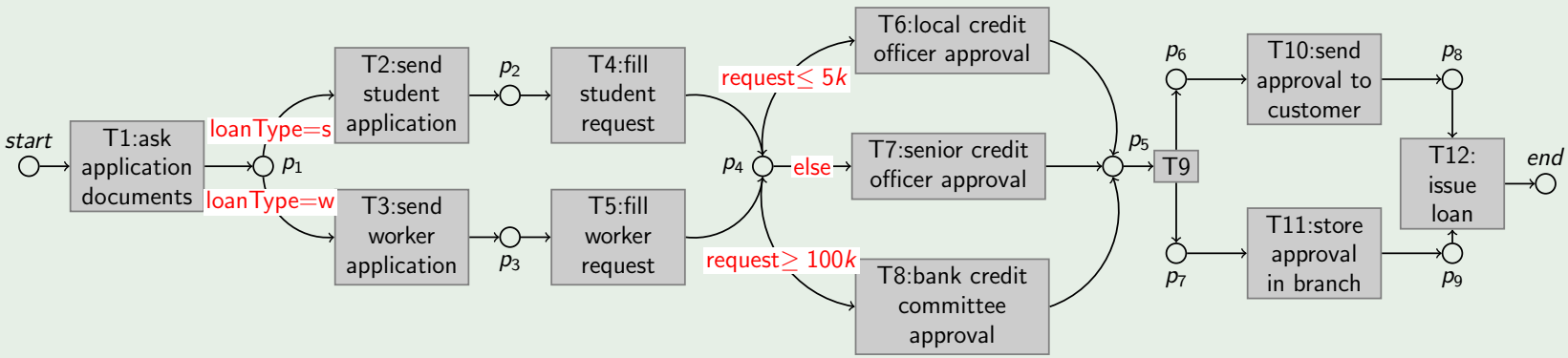
- conditions on exclusive branches;
- how tasks modify data.

Contribution

Equip the process model with data information:

- conditions on exclusive branches;
- how tasks modify data.

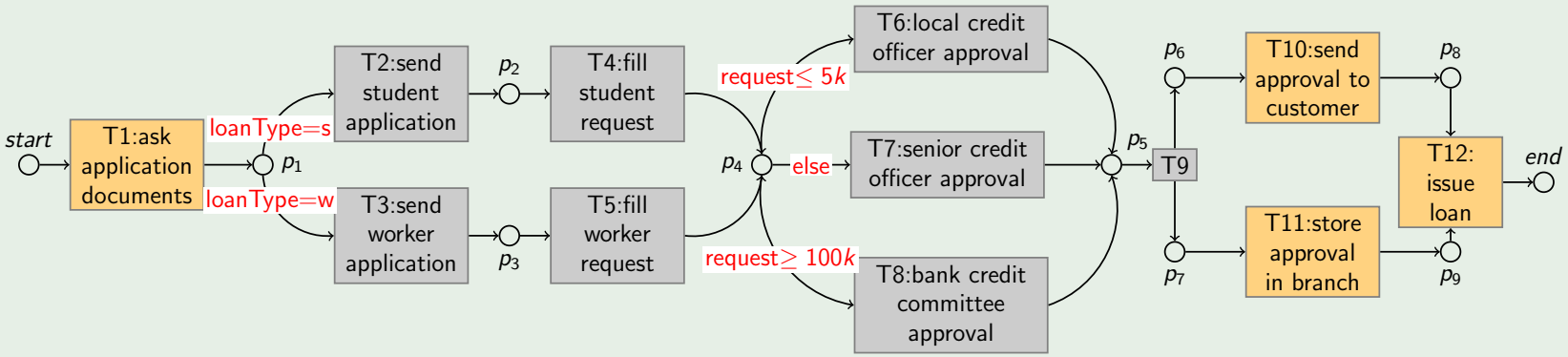
Example



- $T1$ writes $loanType$;
- $T4$ writes $3k \leq request \leq 5k$;
- $T5$ writes $request \geq 1k$.

Example (hard): solved!

Example

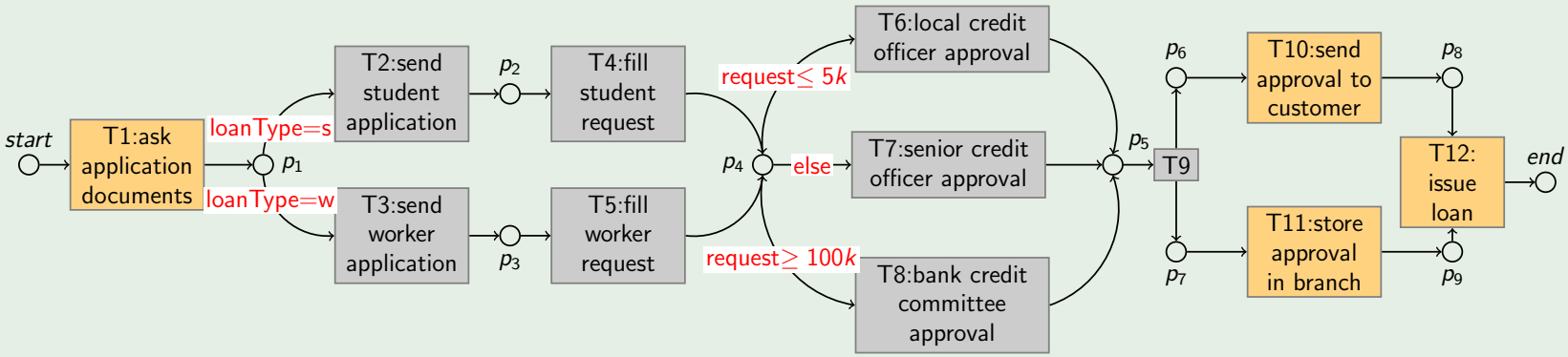


- $T1$ writes $loanType$;
- $T4$ writes $3\mathtt{k} \leq request \leq 5\mathtt{k}$;
- $T5$ writes $1\mathtt{k} \leq request \leq 500\mathtt{k}$.

$$start \rightarrow T1[\cdot] \rightarrow \ldots \rightarrow T10[\cdot] \rightarrow T11[*] \rightarrow T12[\cdot] \rightarrow end$$
$$* : loanType = s, request = 4k$$
$$\Downarrow$$

Example (hard): solved!

Example



- $T1$ writes $loanType$;
- $T4$ writes $3\mathtt{k} \leq request \leq 5\mathtt{k}$;
- $T5$ writes $1\mathtt{k} \leq request \leq 500\mathtt{k}$.

$$start \rightarrow T1[\cdot] \rightarrow \ldots \rightarrow T10[\cdot] \rightarrow T11[*] \rightarrow T12[\cdot] \rightarrow end$$
$$* : loanType = s, request = 4k$$
$$\Downarrow$$
$$start \rightarrow T1 \rightarrow T2 \rightarrow T4 \rightarrow T6 \rightarrow T10 \rightarrow T11 \rightarrow T12 \rightarrow end$$

Talk outline

Data model

Definition (Data model)

A *data model* is a tuple $\mathcal{D} = (\mathcal{V}, \Delta, \mathrm{dm}, \mathrm{ord})$ where:

- $\mathcal{V}$ is a possibly infinite set of variables;
- $\Delta = \{\Delta_1, \Delta_2, \ldots\}$ is a possibly infinite set of domains (not necessarily disjoint);
- $\mathrm{dm} : \mathcal{V} \to \Delta$ is a total and surjective function which associates to each variable v its domain Δ_i;
- ord is a partial function that, given a domain Δ_i, if $\mathrm{ord}(\Delta_i)$ is defined, then it returns a partial order (reflexive, antisymmetric and transitive) $\leq_{\Delta_i} \subseteq \Delta_i \times \Delta_i$.

Data model

Definition (Data model)

A *data model* is a tuple $\mathcal{D} = (\mathcal{V}, \Delta, \mathrm{dm}, \mathrm{ord})$ where:

- $\mathcal{V}$ is a possibly infinite set of variables;
- $\Delta = \{\Delta_1, \Delta_2, \ldots\}$ is a possibly infinite set of domains (not necessarily disjoint);
- $\mathrm{dm} : \mathcal{V} \to \Delta$ is a total and surjective function which associates to each variable v its domain Δ_i;
- ord is a partial function that, given a domain Δ_i, if $\mathrm{ord}(\Delta_i)$ is defined, then it returns a partial order (reflexive, antisymmetric and transitive) $\leq_{\Delta_i} \subseteq \Delta_i \times \Delta_i$.

Example

- $\mathcal{V} = \{\mathit{loanType}, \mathit{request}\}$;
- $\mathrm{dm}(\mathit{loanType}) = \{\mathtt{w}, \mathtt{s}\}$, $\mathrm{dm}(\mathit{request}) = \mathbb{N}$;
- $\mathrm{dm}(\mathit{loan})$ is total ordered by the natural ordering $\leq$ in $\mathbb{N}$.

DAW-net

Definition (DAW-net)

A *DAW-net* is a tuple $\langle \mathcal{D}, \mathcal{N}, \text{wr}, \text{gd} \rangle$ where:

- $\mathcal{N} = \{P, T, F\}$ is a WF-net;
- $\mathcal{D} = \{\mathcal{V}, \Delta, \text{dm}, \text{ord}\}$ is a data model;
- $\text{wr} : T \mapsto (\mathcal{V}' \mapsto 2^{\text{dm}(\mathcal{V})})$, where $\mathcal{V}' \subseteq \mathcal{V}$, $\text{dm}(\mathcal{V}) = \bigcup_{v \in V} \text{dm}(v)$ and $\text{wr}(t)(v) \subseteq \text{dm}(v)$ for each $v \in \mathcal{V}'$, is a function that associates each transition to a (*partial*) function mapping variables to a finite subset of their domain.
- $\text{gd} : T \mapsto \mathcal{L}(\mathcal{D})$ is a function that associates a guard to each transition.

DAW-net

Definition (DAW-net)

A *DAW-net* is a tuple $\langle \mathcal{D}, \mathcal{N}, \text{wr}, \text{gd} \rangle$ where:

- $\mathcal{N} = \{P, T, F\}$ is a WF-net;
- $\mathcal{D} = \{\mathcal{V}, \Delta, \text{dm}, \text{ord}\}$ is a data model;
- $\text{wr} : T \mapsto (\mathcal{V}' \mapsto 2^{\text{dm}(\mathcal{V})})$, where $\mathcal{V}' \subseteq \mathcal{V}$, $\text{dm}(\mathcal{V}) = \bigcup_{v \in V} \text{dm}(v)$ and $\text{wr}(t)(v) \subseteq \text{dm}(v)$ for each $v \in \mathcal{V}'$, is a function that associates each transition to a (*partial*) function mapping variables to a finite subset of their domain.
- $\text{gd} : T \mapsto \mathcal{L}(\mathcal{D})$ is a function that associates a guard to each transition.

Example

- $\text{wr}(T4) : \{request\} \mapsto \{\texttt{3k} \ldots \texttt{5k}\}$
- $\text{wr}(T5) : \{request\} \mapsto \{\texttt{1k} \ldots \texttt{500k}\}$

Talk outline

Trace repair as reachability

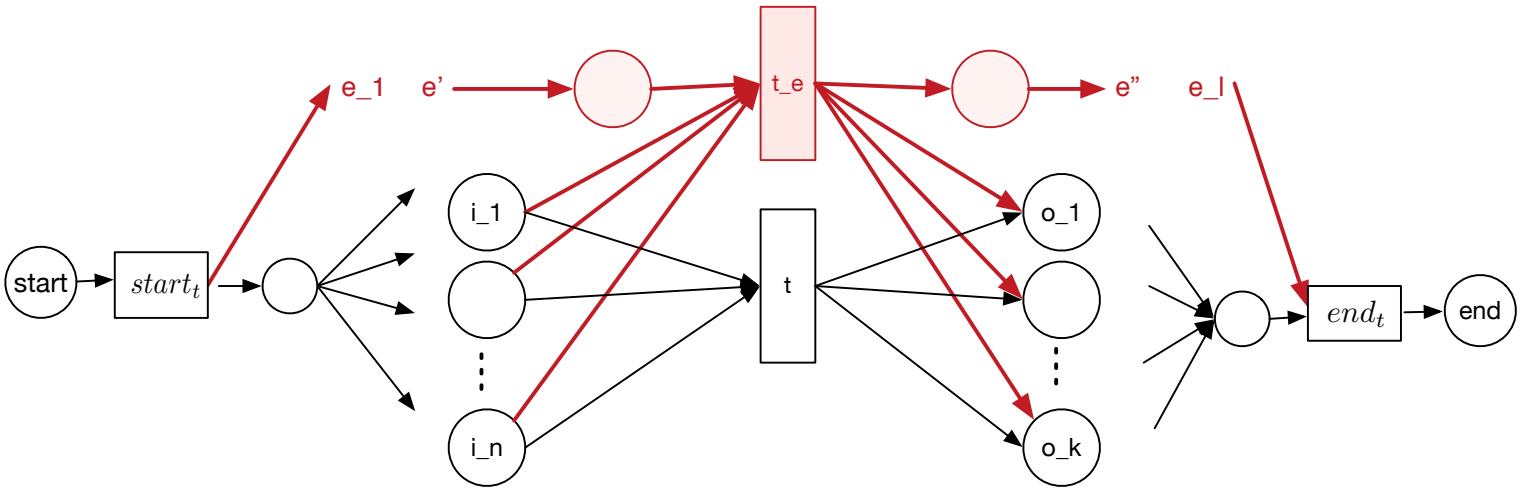
Problem:

- Input : 1) trace τ (events & payloads) and 2) DAW-net model W;
- output : yes if compliant, no otherwise.

Compliance: W contains all the events observed in τ (with the corresponding variable updates) in the right order and those can be extended so as to reach the final marking.

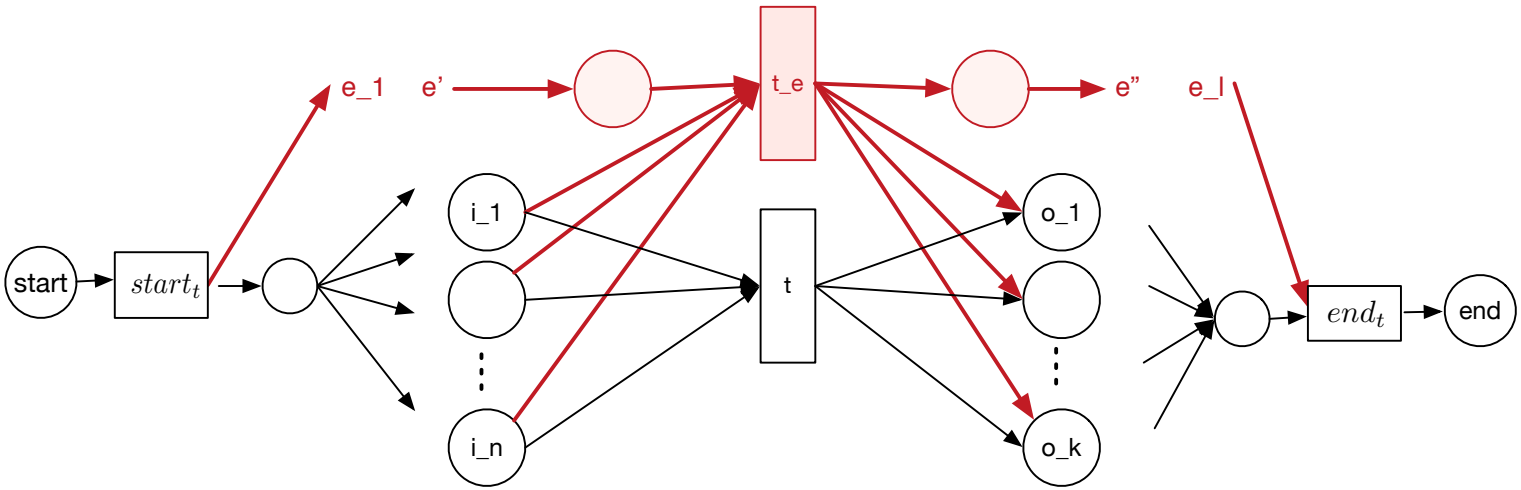
Trace repair as reachability

We "inject" τ in W thus obtaining W^{τ} as follows:



Trace repair as reachability

We "inject" τ in W thus obtaining W^τ as follows:



Sound and complete encoding:

- all executions of W^τ are compliant with τ;
- each execution of W^τ is also a case of W and
- if there is an execution of W compliant with τ, then that is also an execution for W^τ.

Trace repair as reachability

Reachability: it is possible to reach the final marking in the above net?

If so, then τ is compliant and the plan is a possible repair.

Reachability as a planning problem

Workflow net encoding:

- a fluent declaration p for each place;
- an action declaration t for each task;
- rules for generating and consuming tokens.

Reachability as a planning problem

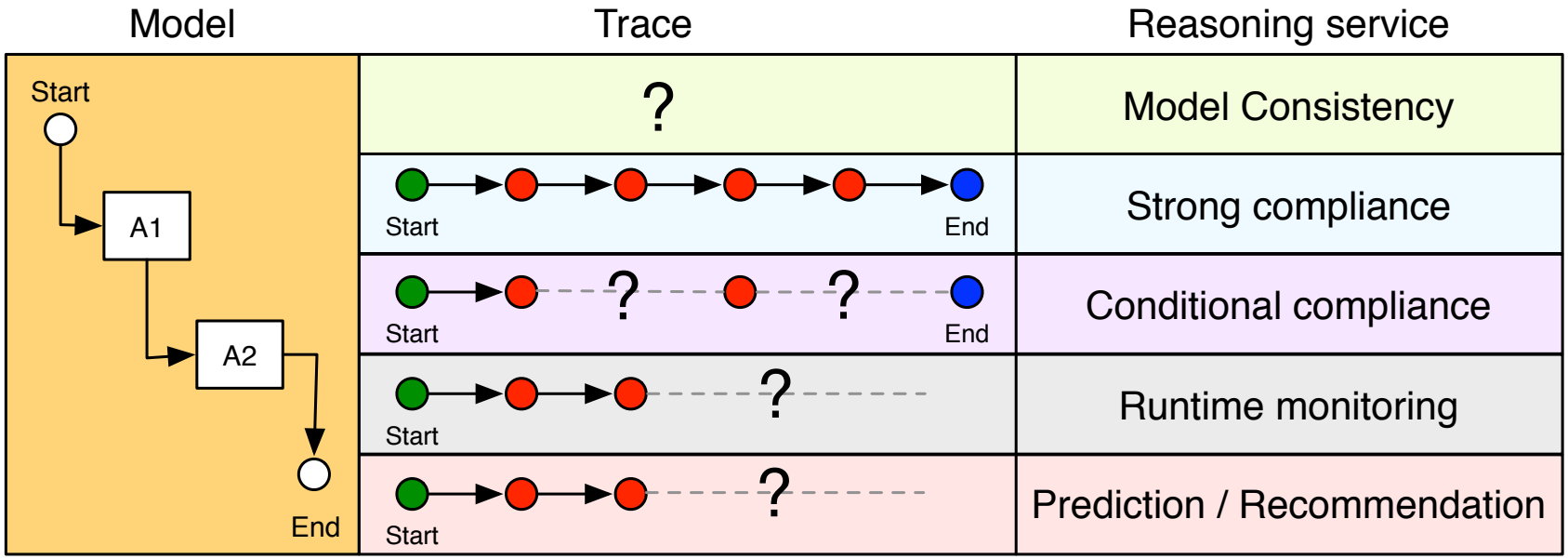
Workflow net encoding:

- a fluent declaration p for each place;
- an action declaration t for each task;
- rules for generating and consuming tokens.

Data encoding:

- a fluent unary predicate var_v for each variable $v \in \mathcal{V}$ holding its value;
- we impose *functionality*;
- nondeterministic value update only if corresponding task is executed.

(Other) reasoning tasks we get for free



Related work

Data-centric dynamic systems
(infinite domains, tasks modify relational data)

$\succeq$

DAW-nets
(finite domains, tasks modify global variables)

$\succeq$

workflow-nets with data
(Sidorova et al., reasoning on variables *defined* or *undefined*)

Future work

EXPERIMENTS!

With DLV$^{\mathcal{K}}$, an ASP-based planner, performances are :-(