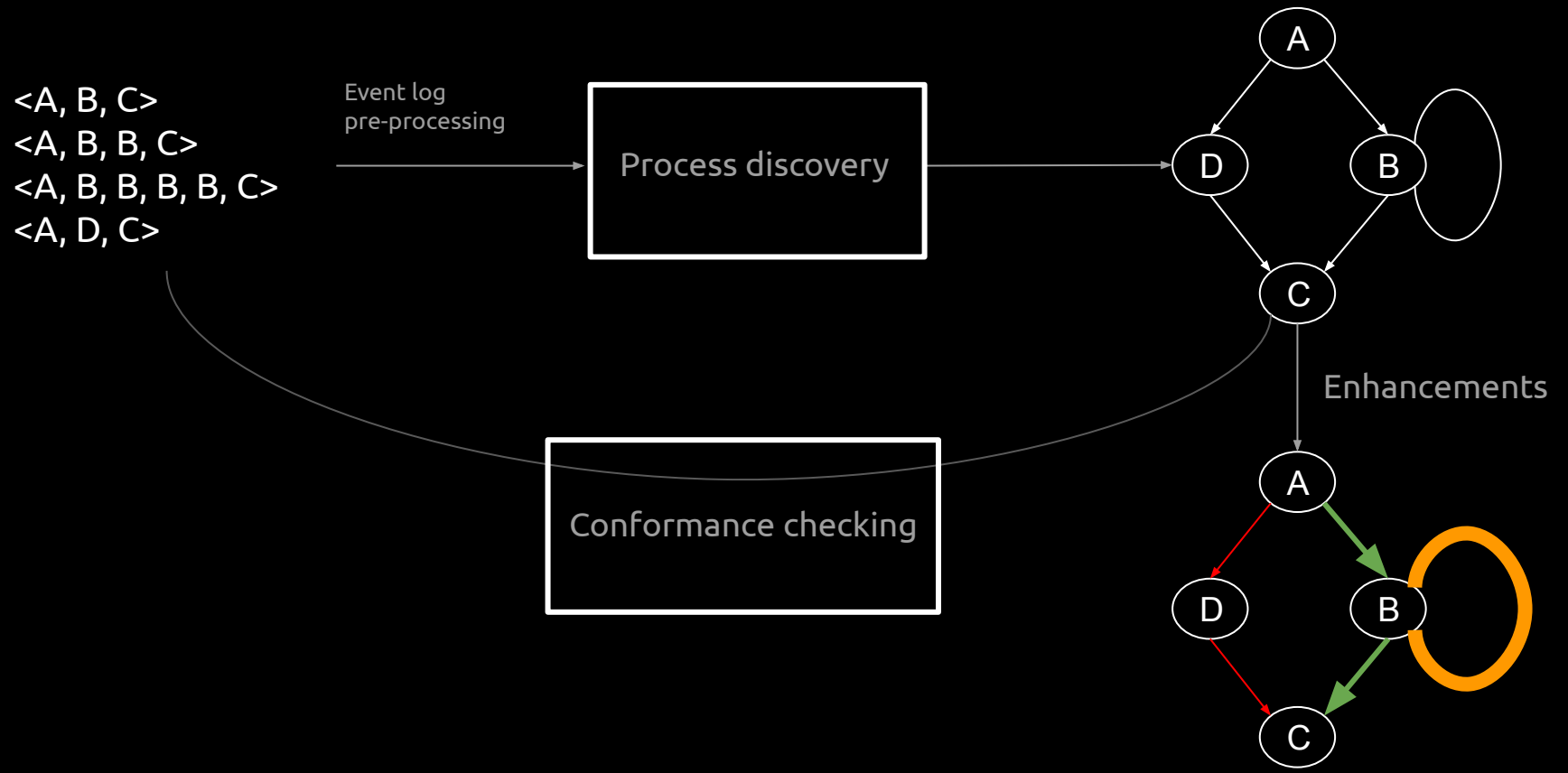
Streaming Process Mining

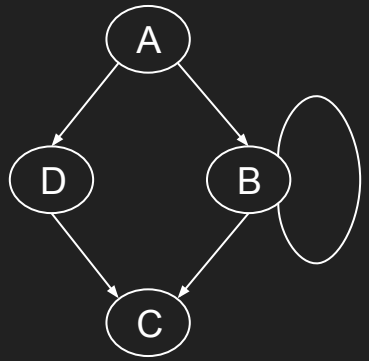
January 2025


<A, B, C>
<A, B, B, C>
<A, B, B, B, B, C>
<A, D, C>
Event log
pre-processing
Process discovery
A
D
B
C
Enhancements
Conformance checking
A
D
B
C

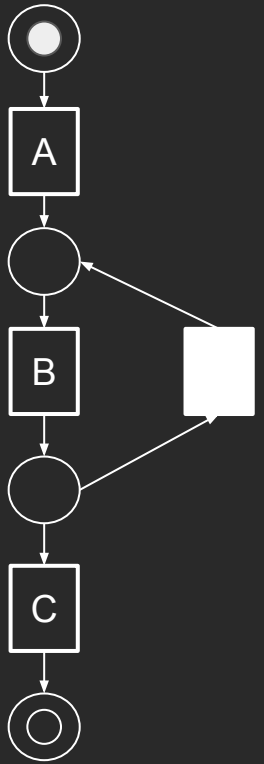
Initial event log

<A, B, C>
<A, B, B, C>
<A, B, B, B, B, C>
<A, D, C>

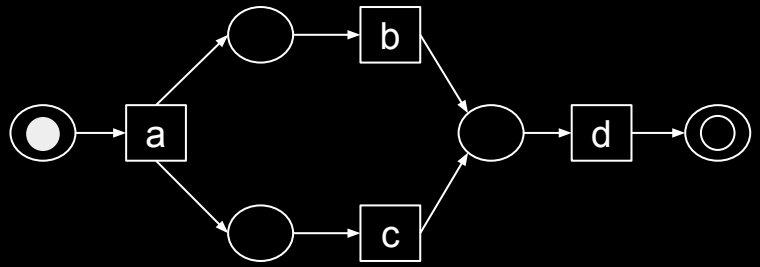
Discovered abstraction (i.e., directly-follows graph)



Discovered process model (i.e., Petri net, BPMN)



Petri nets



Definition 2.2 (Petri net, unlabelled Petri net)**.** *A* Petri net *over a given alphabet Σ is a tuple (P, T, A, M_0, F, l) consisting of:*

- *a set of places P;*
- *a set of transitions T, such that $P \cap T = \varnothing$;*
- *a multiset arc relation $A \subseteq \mathbb{M}((P \times T) \cup (T \times P))$;*
- *an initial marking $m_0 \subseteq \mathbb{M}(P)$;*
- *a set of final markings F, being a set of multisets over P;*
- *a partial labelling function $l \colon T \to \Sigma$.*

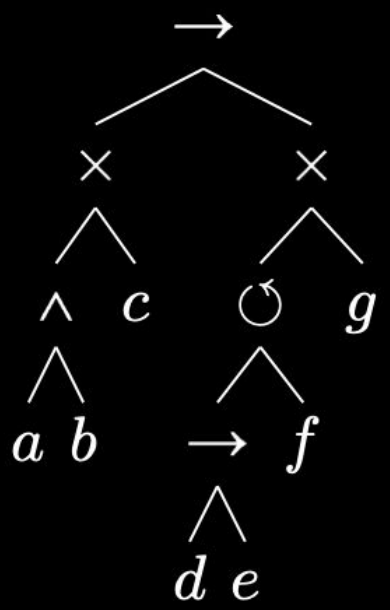
Definition 2.3 (workflow net)**.** *A* workflow net *is a Petri net (P, T, A, m_0, F, l) such that*

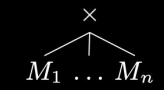
- *there is a single $i \in P$ such that ${}^\bullet i = \varnothing$;*
- *there is a single $o \in P$ such that $o^\bullet = \varnothing$;*
- *all places (P) and transitions (T) are on a path from i to o;*
- *the initial marking consists of i: $m_0 = [i]$;*
- *the only final marking consists of o: $F = \{[o]\}$.*

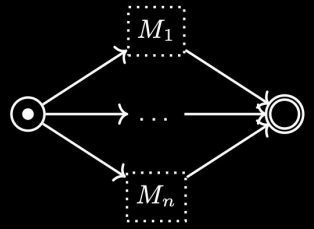
Definition 2.4 (soundness)**.** *Let $W = (P, T, A, [i], [o], l)$ be a workflow net, in which i is the source place and o is the sink place. W is* sound *if and only if:*

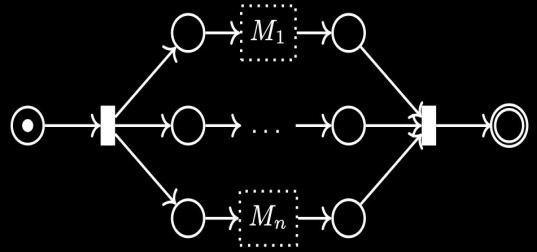
- *every transition can be fired, i.e. $\forall_{t \in T}\ \exists_{[o] \rightsquigarrow m}\ {}^\bullet t \subseteq m$;*
- *from every marking, reachable from $[i]$, it is possible to reach $[o]$, i.e. $\forall_{[i] \rightsquigarrow m}\ m \rightsquigarrow [o]$;*
- *every marking, reachable from $[i]$, that puts a token in o has no other tokens, i.e. $\forall_{[i] \rightsquigarrow m}\ o \in m \Rightarrow [o] = m$.*

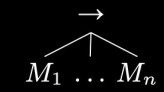
Process trees

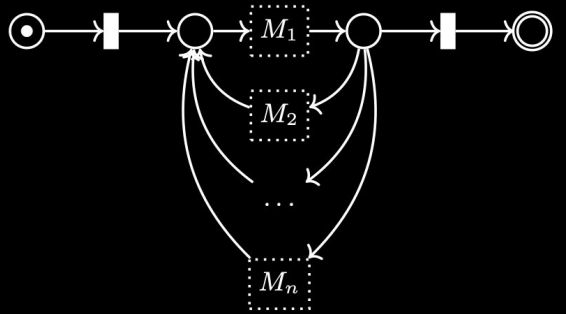


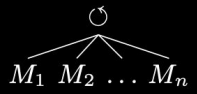












Alpha algorithm example

<A, B, C>
<A, B, B, C>
<A, B, B, B, B, C>
<A, D, C>

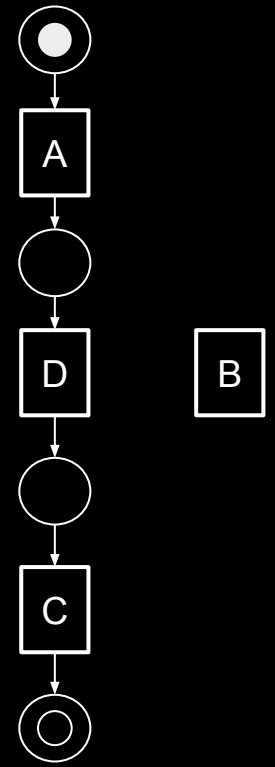
	A	B	C	D
A		3		1
B		3	3	
C				
D			1	

Directly follows relation
a -> b

Parallel relation
a -> b && b -> a

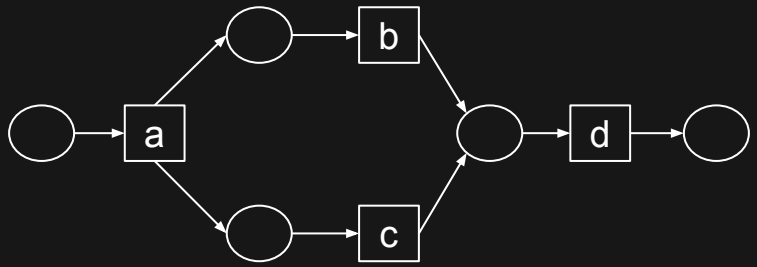
Causal relation
a -> b && !(b -> a)

Unrelated relation
!(a -> b) && !(b -> a)



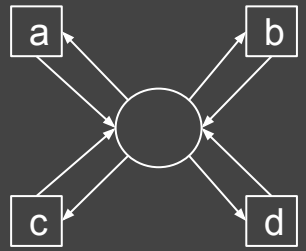
Fitness

fitness, precision, general, simple



Precision

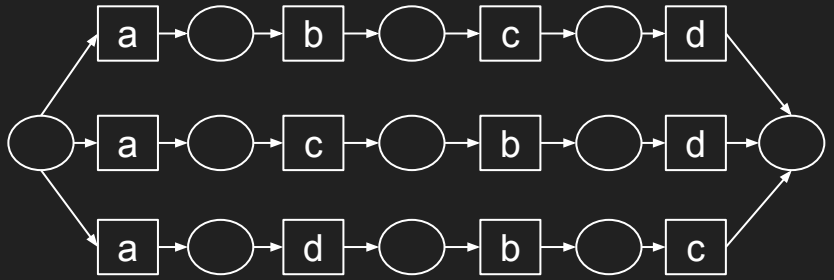
fitness, precision, general, simple



[<a, b, c, d>^{2}, <a, c, b, d>^{2}, <a, d, b, c>]

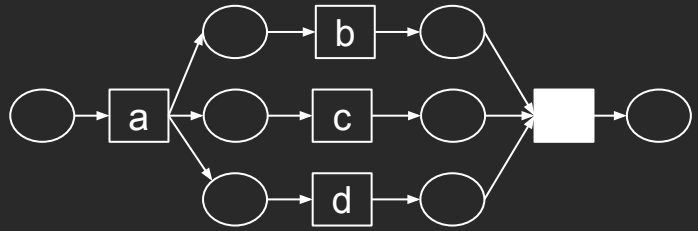
Simplicity

fitness, precision, general, simple



Generalization

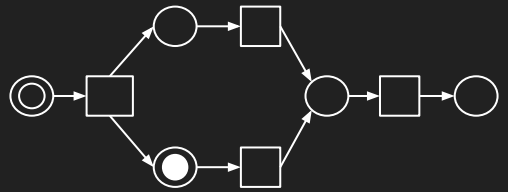
fitness, precision, general, simple



Naive approaches

Count the percentage of traces which fully fit the discovered model

Token replay game



P - produced tokens
C - consumed tokens
M - missing tokens
R - remaining tokens

Fitness = 0.5 * (1 - M/C) + 0.5 * (1 - R/P)

Alignments

Synchronous move

Log	a	b	>>
Model	>>	b	c

Log move

Model move

Fitness = 1 - alignment/worst_alignemnt

Stream of events

$(e_{n-1}, t_{n-1}, c_{n-1}), (e_n, t_n, c_n), (e_{n+1}, t_{n+1}, c_{n+1})$

e - an event class
t - a timestamp of the event
c - case identifier

Stream of traces

A complete trace

T1 algorithms

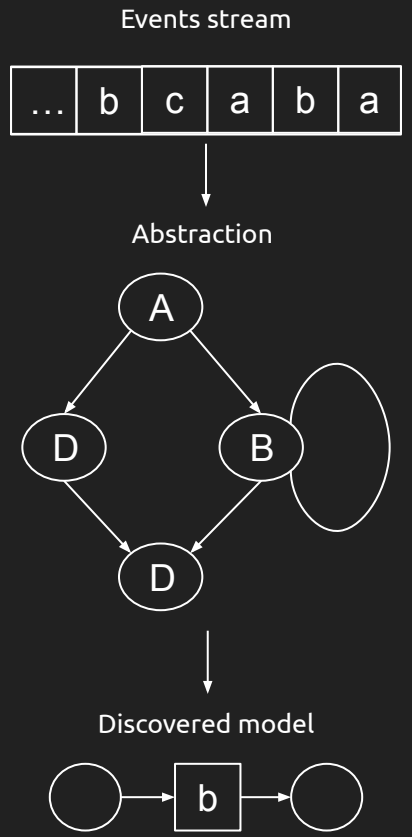
(caching whole log + rediscover model each time)

<A, B, C>
<A, B, B, C>
<A, B, B, B, B, C>
<A, D, C>

Different caching strategies, i.e., strategy based on time, last updated trace, etc.

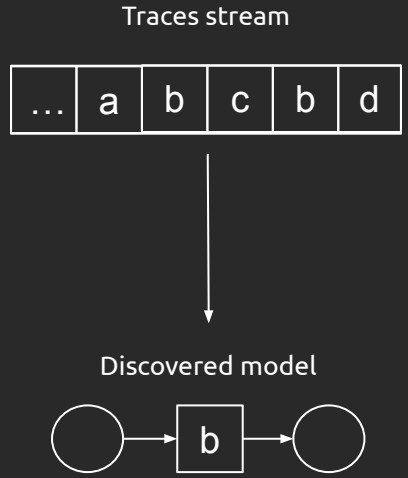
T2 algorithms

(online update of an abstraction + rediscovery each time)



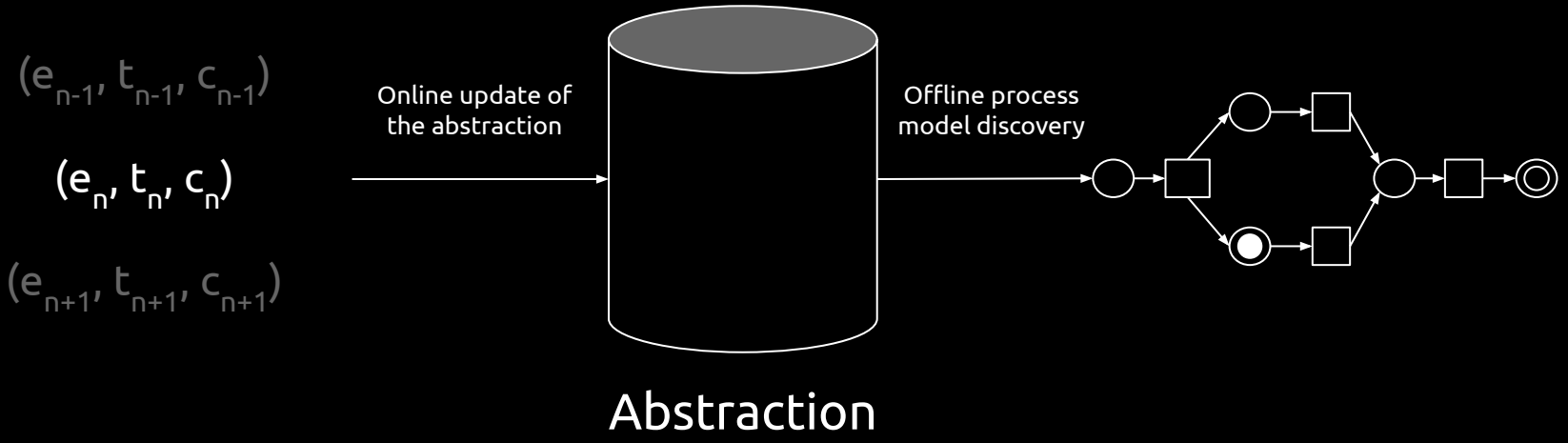
T3 algorithms

(incremental update of the discovered model)



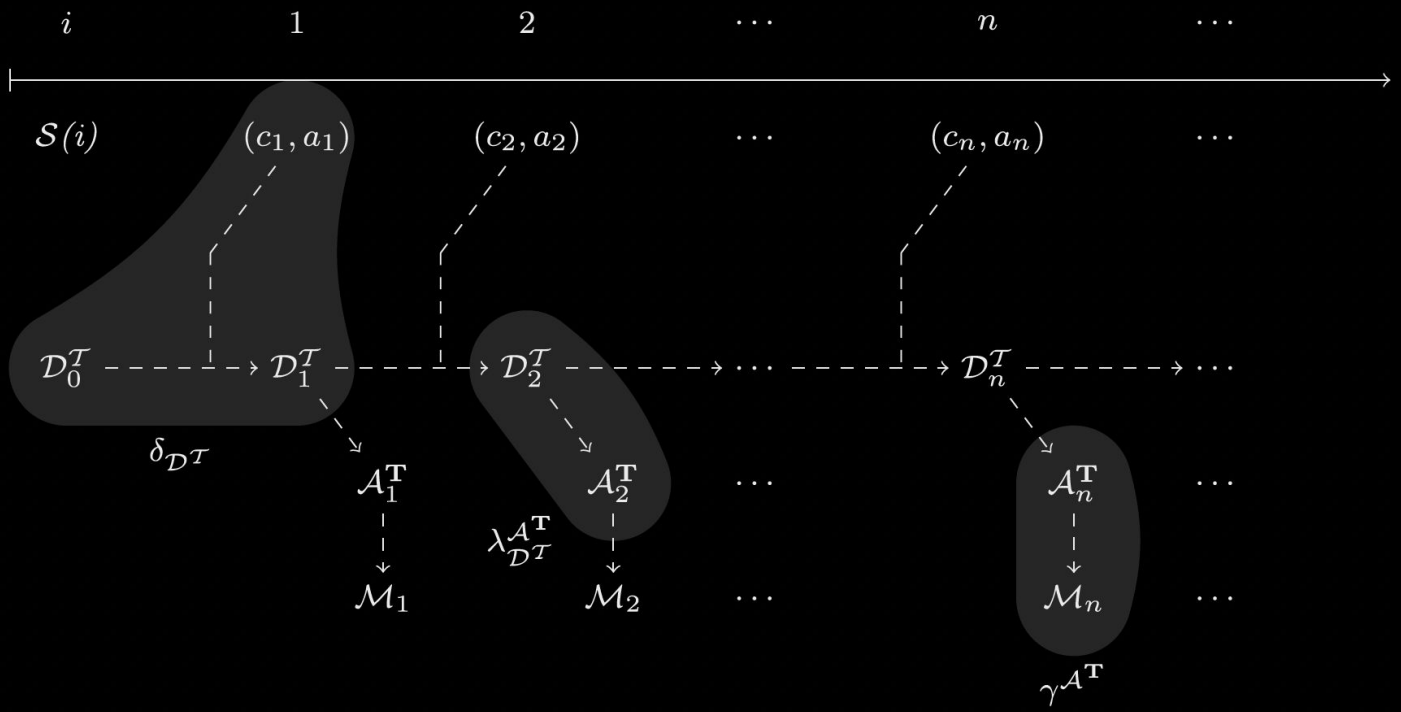
T2 Algorithms

S-BAR Architecture: concept



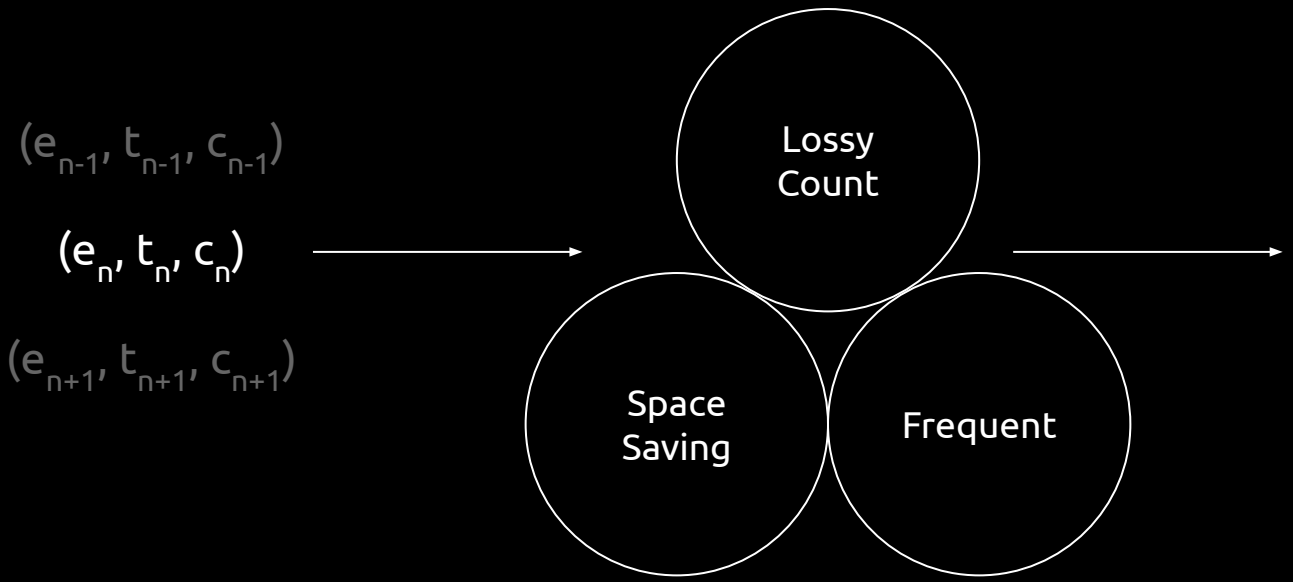
T2 Algorithms

S-BAR Architecture: formal description



T2 Algorithms

S-BAR Architecture: storing the abstraction



T2 Algorithms

S-BAR Architecture: Lossy Count and Frequent

Algorithm 2: $\mathcal{D}^{\mathcal{C}}$ (Lossy)

```
input  : k ∈ ℕ, S ∈ (C × A)*, D^A
begin
1   i, Δ ← 0; X ← ∅;
2   while true do
3       i ← i + 1;
4       (c, a) ← S(i);
5       if ∃_{(c',a')∈X}(c' = c) then
6           v_c ← v_c + 1;
7           D^A ⊎ {(a', a)};
8           X ← (X ∪ {(c, a)}) \ {(c, a')};
9       else
10          X ← X ∪ {(c, a)};
11          v_c ← Δ;
12      if ⌊i/k⌋ ≠ Δ then
13          foreach (c', a') ∈ X do
14              if v_{c'} ≤ Δ then
15                  X ← X \ (c', a');
16          Δ ← ⌊i/k⌋;
```

HashMap<String, String>

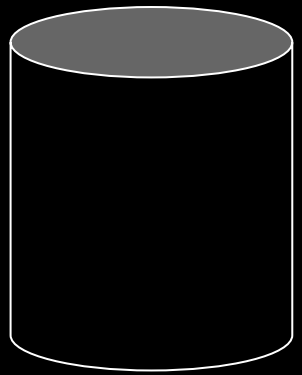
Algorithm 3: $\mathcal{D}^{\mathcal{A}}$ (Frequent)

```
input  : k ∈ ℕ, S_A ∈ (A × A)*
begin
1   X ← ∅, i ← 0;
2   while true do
3       i ← i + 1;
4       (a, a') ← S_A(i);
5       if (a, a') ∈ X then
6           v_{(a,a')} ← v_{(a,a')} + 1;
7       else if |X| < k then
8           X ← X ∪ {(a, a')};
9           v_{(a,a')} ← 1;
10      else
11          foreach (x, y) ∈ X do
12              v_{(x,y)} ← v_{(x,y)} − 1;
13              if v_{(x,y)} = 0 then
14                  X ← X \ {(x, y)};
```

HashMap<(String, String), u64>

T2 Algorithms

S-BAR Architecture: Instantiations



Streaming
directly-follows graph

Enables streaming
implementation for

- Alpha-miner
- Inductive Miner
- Heuristics Miner

T3 Algorithms

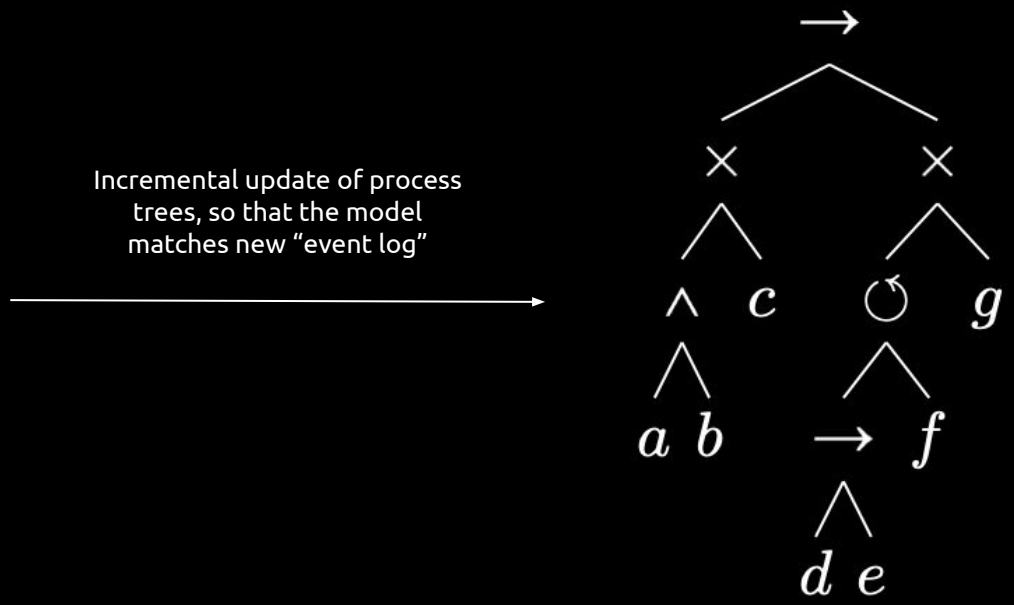
Incremental update of existing model

<a, b, c, d, a,…>

<a, b, c, d, a, b, d, c,…>

<a, b, c, d, a, b,…>

Incremental update of process trees, so that the model matches new “event log”



T3 Algorithms

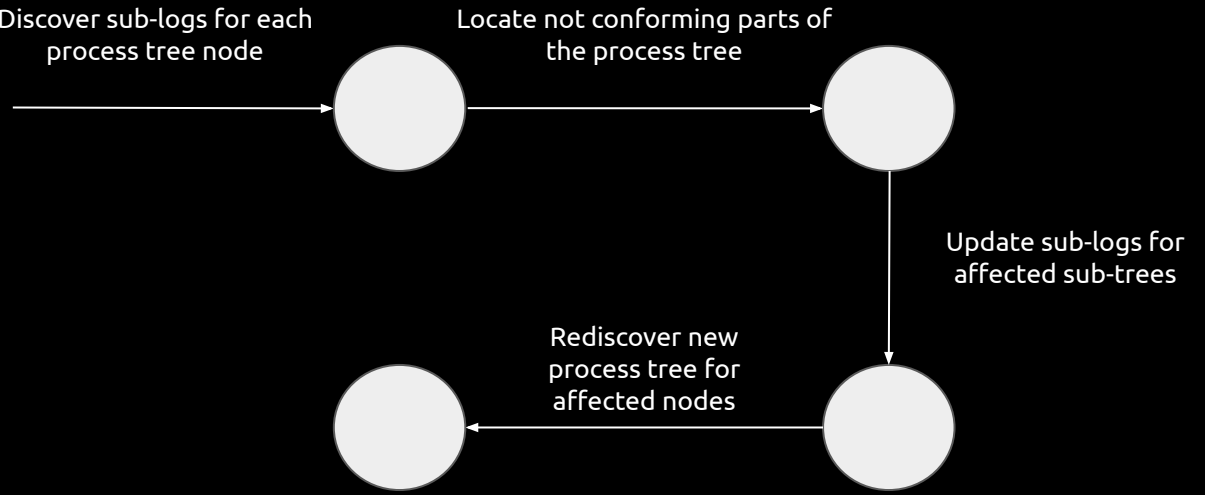
Example approach overview

Already discovered process tree

Existing event log

New trace

Offline discovery algorithms (IM)



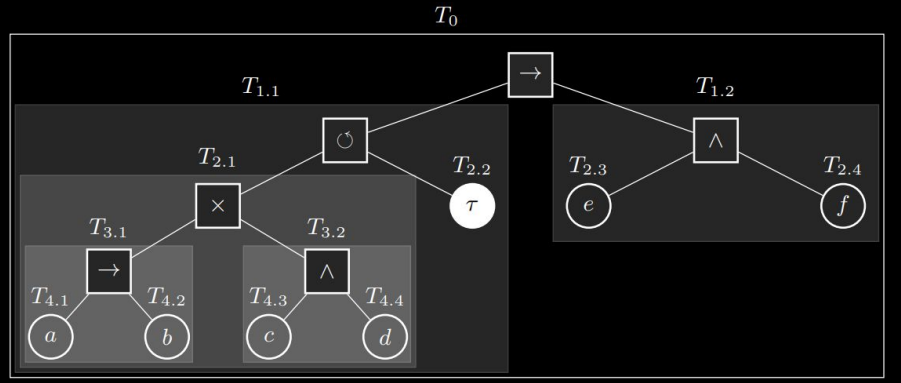
T3 Algorithms

Example approach overview: creating sub-logs

Algorithm 1: Calculation of sub event logs for process trees

Input: $L \in \mathcal{B}(\mathcal{A}^*), T \in \mathcal{T}_\mathcal{A}$ (Assumption: $\{\sigma \in L\} \subseteq \mathcal{L}(T)$))
Output: sub event log for each subtree of T, i.e., $s : \mathcal{T}_\mathcal{A} \to \mathcal{B}(\mathcal{A}^*)$
begin

1 **forall** *subtrees* T' *of* T **do**
2 $s(T') \leftarrow []$ // initialize sub event logs
3 **forall** $\sigma \in L$ **do**
4 let $\gamma \in \bar{\Gamma}(\sigma, T)$ // calculate optimal alignment for σ and T
5 **forall** *subtrees* T' *of* T **do**
6 $t(T') \leftarrow \langle\rangle$ // initialize trace for each subtree
7 **for** $i \in \{1, \ldots, |\gamma|\}$ **do**
8 $m \leftarrow \gamma(i)$ // extract i-th alignment move
9 $T_l \leftarrow \pi_2(m)$ // extract executed process leaf node
10 **forall** *subtrees* T' *of* T **do**
11 **if** T_l *is subtree of* $T' \vee T_l = T'$ **then**
12 $t(T') \leftarrow t(T') \cdot \langle \pi_2(m) \rangle$ // add executed activity to T''s trace
13 **else if** $t(T') \neq \langle\rangle$ **then**
14 $s(T') \leftarrow s(T') \uplus [t(T')]$ // add trace to T''s sub event log
15 $t(T') \leftarrow \langle\rangle$ // reset trace
16 **return** s



a	b	$\gg$	c	d	$\gg$	a	b	e	f
a	b	τ	c	d	τ	a	b	e	f
$(T_{4.1})$	$(T_{4.2})$	$(T_{2.2})$	$(T_{4.3})$	$(T_{4.4})$	$(T_{2.2})$	$(T_{4.1})$	$(T_{4.2})$	$(T_{2.3})$	$(T_{2.4})$
T_0	T_0		T_0	T_0		T_0	T_0	T_0	T_0
$T_{1.1}$	$T_{1.1}$		$T_{1.1}$	$T_{1.1}$		$T_{1.1}$	$T_{1.1}$		
$T_{2.1}$	$T_{2.1}$		$T_{2.1}$	$T_{2.1}$		$T_{2.1}$	$T_{2.1}$		
$T_{3.1}$	$T_{3.1}$					$T_{3.1}$	$T_{3.1}$		
			$T_{3.2}$	$T_{3.2}$				$T_{1.2}$	$T_{1.2}$

(a) Alignment and listing of subtrees containing the executed process tree leave nodes

T_0	$[\langle a, b, c, d, a, b, e, f \rangle]$
$T_{1.1}$	$[\langle a, b, c, d, a, b \rangle]$
$T_{2.1}$	$[\langle a, b \rangle^2, \langle c, d \rangle]$
$T_{3.1}$	$[\langle a, b \rangle^2]$
$T_{3.2}$	$[\langle c, d \rangle]$
$T_{1.2}$	$[\langle e, f \rangle]$

(b) Sub event logs

T3 Algorithms

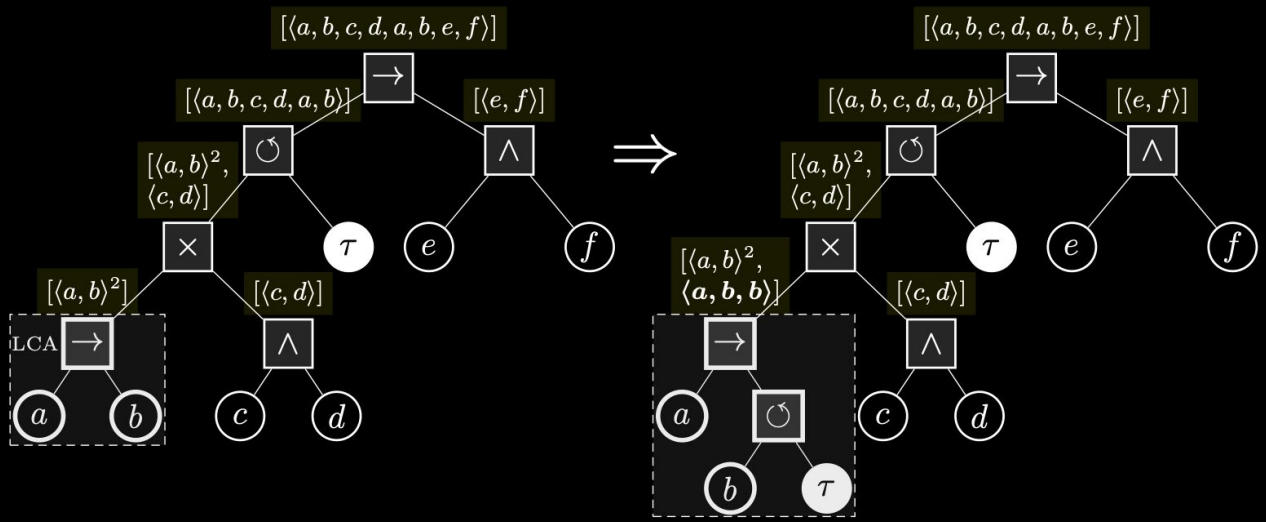
Example approach overview: locating deviations

$\cdots$	x'_i	$\cdots$	x_i	$\cdots$ $deviation(s)$ $\cdots$		x_j	$\cdots$	x'_j	$\cdots$
$\cdots$	T'_i	$\cdots$	T_i	$\cdots$ $deviation(s)$ $\cdots$		T_j	$\cdots$	T'_j	$\cdots$

T3 Algorithms

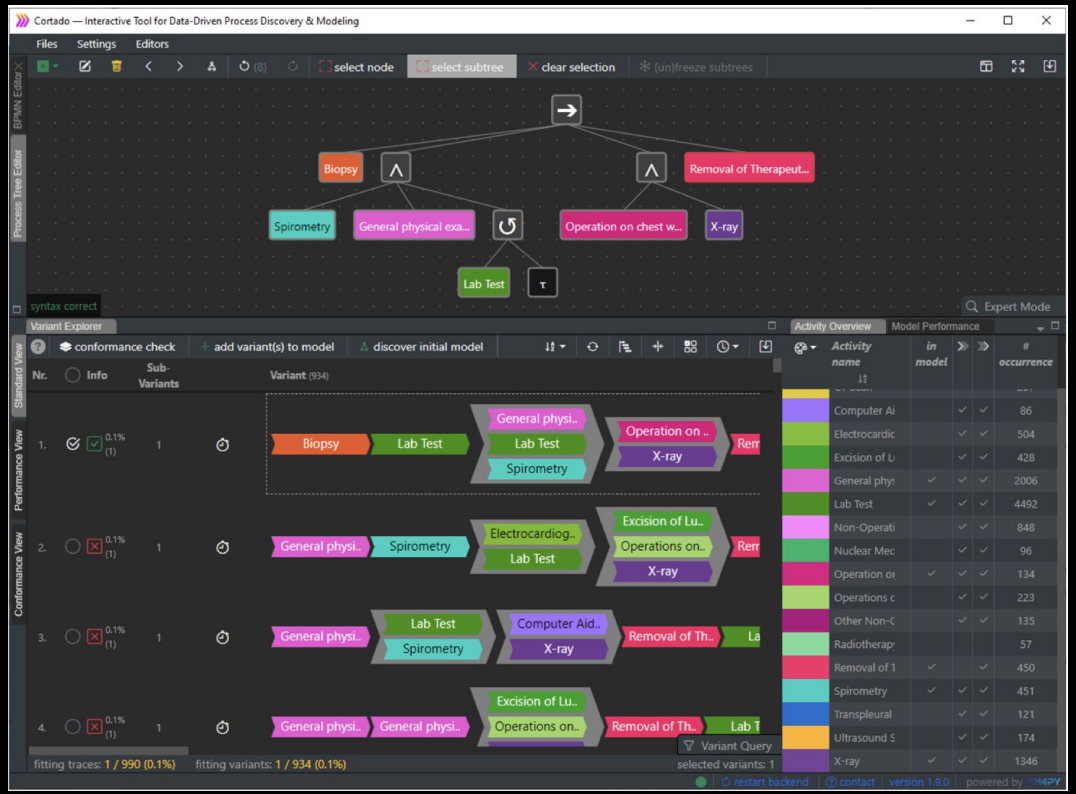
Example approach overview: repairing existing tree

New trace: <a, b, b, e, f>



T3 Algorithms

Practical application: Cortado



Online Conformance Checking

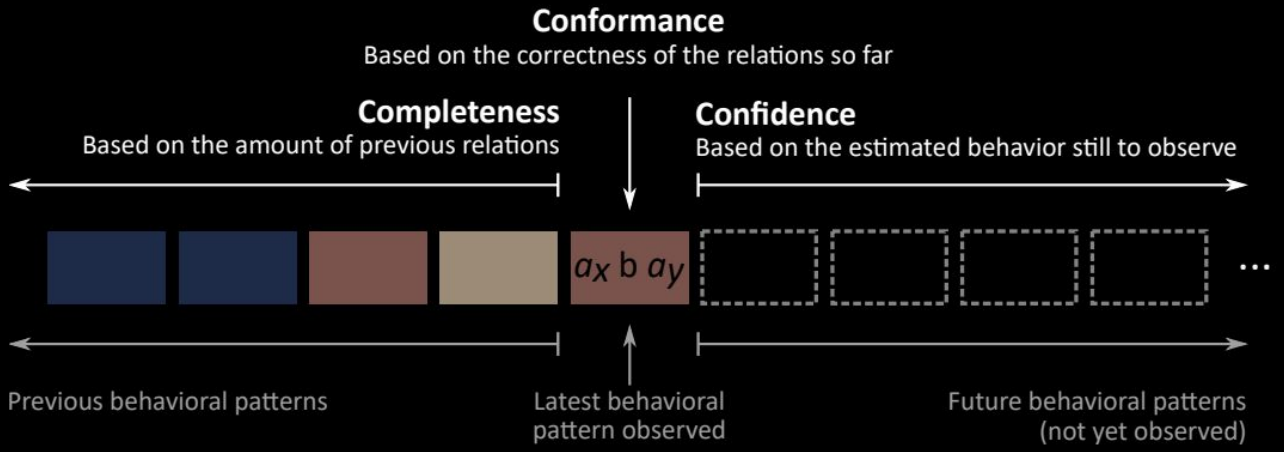
Incremental computing of prefix-alignments

Conformance checking based on behavioral patterns

Online Conformance Checking

Behavioral patterns

Definition 1 (Behavioural Pattern). *Given a set of activities $\mathcal{A}$ and a set of possible control-flow relations $\mathcal{R}$, a* behavioural pattern *is defined as $b(a_1, a_2)$ where $a_1, a_2 \in \mathcal{A}$ are activities and $b \in \mathcal{R}$ represents a control-flow relation. An alternative writing of $b(a_1, a_2)$ is $a_1\ b\ a_2$.*



Online Conformance Checking

Behavioral patterns

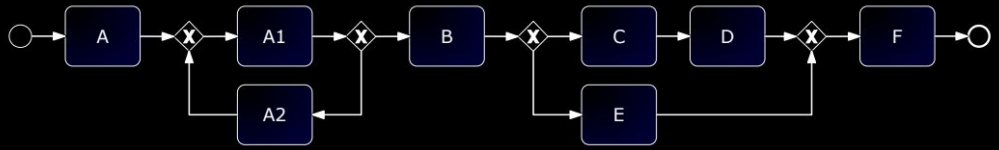


Fig. 1: Running example considered throughout this paper.

Table 1: Comparison of offline ([2]) and online conformance values (as proposed in this paper) based on the process model in Fig. 1.

Trace	Offline Conformance	Online Conformance	Online Completeness	Online Confidence
$t_1 = \langle A, A1, B, E, F \rangle$	1.00	1.00	1.00	1.00
$t_2 = \langle B, C, D, F \rangle$	0.78	1.00	0.60	1.00
$t_3 = \langle A, A1, A2, A1, B \rangle$	0.80	1.00	1.00	0.50
$t_4 = \langle B, C, D \rangle$	0.62	1.00	0.50	0.75

Online Conformance Checking

Behavioral patterns

Algorithm 1: Online conformance computation

Input: S: stream of behavioural patterns
$M = (B, P, F)$: process model for online conformance

```
Initialize map obs          // Maps case ids to (finite) set of observed prescribed patterns
Initialize map inc                                          // Maps case ids to integers
forever do
    (c, b, t) ← observe(S)                          // New observable unit from the stream
    // Step 1: update internal data structures
    if b ∈ B then
        obs(c) ← obs(c) ∪ {b}                   // If b already in obs(c), then no effect
    else
        inc(c) ← inc(c) + 1
    // Step 2: compute online conformance values
    conformance(c) ← |obs(c)| / (|obs(c)| + inc(c))
    Notify new value of conformance(c)
    if b ∈ B then
        if P_min(b) ≤ |obs(c)| ≤ P_max(b) then
            completeness(c) ← 1
        else
            completeness(c) ← min{1, |obs(c)| / (P_min(b) + 1)}
        confidence(c) ← 1 − F(b) / max_{b'∈B} F(b')
        Notify new values of completeness(c) and confidence(c)
    // Step 3: cleanup
    if size of obs and inc is close to max capacity then
        Remove oldest entries from obs and inc
```

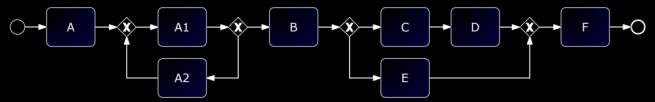


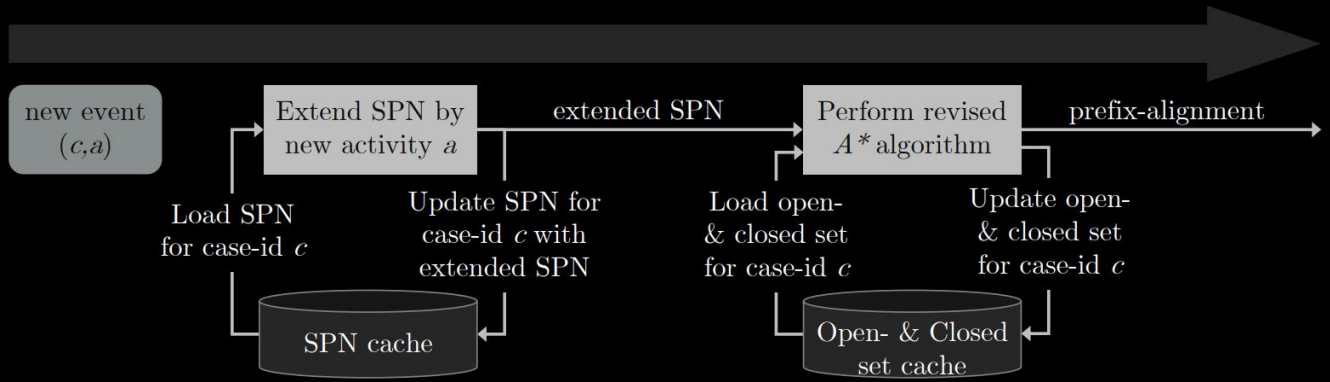
Fig. 1: Running example considered throughout this paper.

Table 1: Comparison of offline ([2]) and online conformance values (as proposed in this paper) based on the process model in Fig. 1.

Trace	Offline	Online		
	Conformance	Conformance	Completeness	Confidence
$t_1 = \langle A, A1, B, E, F \rangle$	1.00	1.00	1.00	1.00
$t_2 = \langle B, C, D, F \rangle$	0.78	1.00	0.60	1.00
$t_3 = \langle A, A1, A2, A1, B \rangle$	0.80	1.00	1.00	0.50
$t_4 = \langle B, C, D \rangle$	0.62	1.00	0.50	0.75

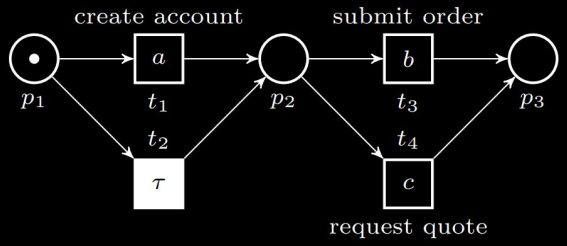
Online Conformance Checking

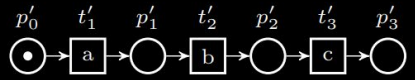
Prefix-alignments



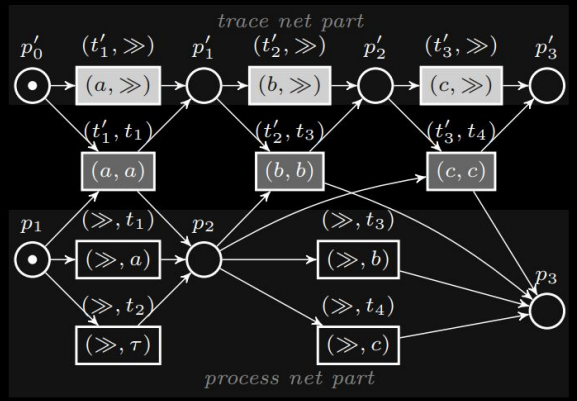
Online Conformance Checking

Prefix-alignments



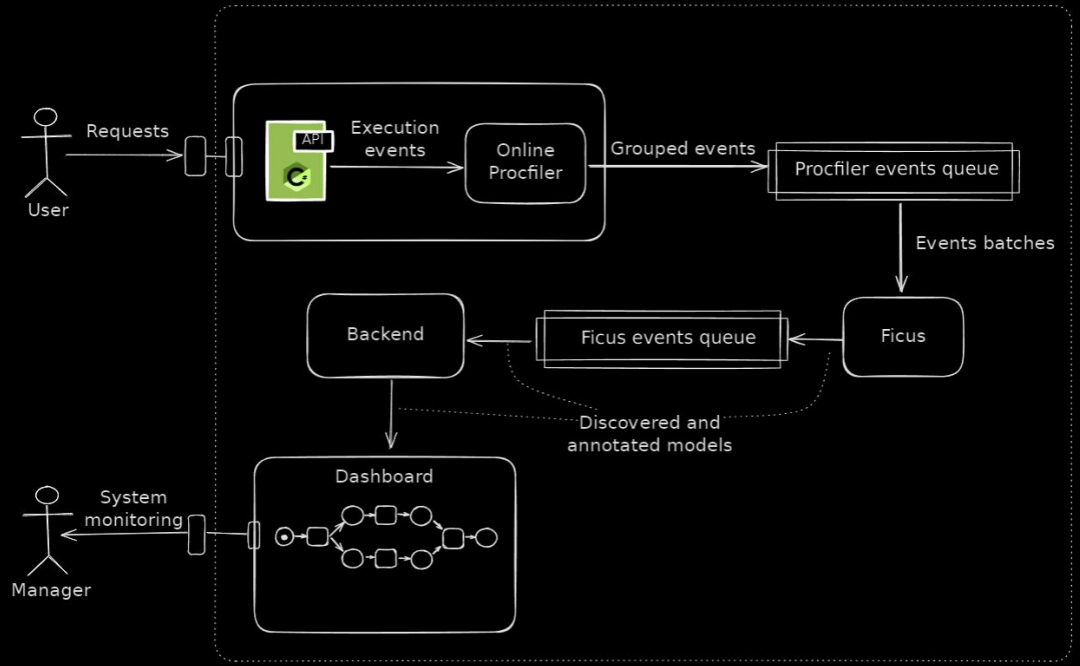


(a) Trace net of the trace $\langle a, b, c \rangle$



(b) SPN N_1^S of $\langle a, b, c \rangle$ and the WF-net N_1

The Need for Streaming Process Mining



Thanks for the attention

References

Streaming Process Discovery

Sebastiaan van Zelst, Boudewijn van Dongen, and Wil van der Aalst. 2018. Event Stream-Based Process Discovery using Abstract Representations. Knowledge and Information Systems 54 (02 2018). https://doi.org/10.1007/s10115-017-1060-2

G. Manku and R. Motwani. 2002. Approximate frequency counts over data streams. Proceedings of 28th International Conference on Very Large Data Bases 5 (01 2002).

Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. 2002. Frequency Estimation of Internet Packet Streams with Limited Space. In Algorithms - ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17-21, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2461), Rolf H. Möhring and Rajeev Raman (Eds.). Springer, 348–360. https://doi.org/10.1007/3-540-45749-6_33

Daniel Schuster, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. 2020. Incremental Discovery of Hierarchical Process Models. In RCIS (Lecture Notes in Business Information Processing, Vol. 385). Springer, Berlin, 417–433. https://doi.org/10.1007/978-3-030-50316-1_25

Daniel Schuster, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. 2021. Freezing Sub-models During Incremental Process Discovery. In Conceptual Modeling - 40th International Conference, ER 2021, Virtual Event, October 18-21, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13011), Aditya K. Ghose, Jennifer Horkoff, Vítor E. Silva Souza, Jeffrey Parsons, and Joerg Evermann (Eds.). Springer, Berlin, 14–24. https://doi.org/10.1007/978-3-030-89022-3_2

References

Online Conformance Checking

Daniel Schuster and Gero Joss Kolhof. 2020. Scalable Online Conformance Checking Using Incremental Prefix-Alignment Computation. In Service-Oriented Computing - ICSOC 2020 Workshops and Satellite Events, Dubai, United Arab Emirates, December 14-17, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12632), Hakim Hacid, Fatma Outay, Hye-young Paik, Amira Alloum, Marinella Petrocchi, Mohamed Reda Bouadjenek, Amin Beheshti, Xumin Liu, and Abderrahmane Maaradji (Eds.). Springer, 379–394. https://doi.org/10.1007/978-3-030-76352-7_36

Sebastiaan van Zelst, Alfredo Bolt, Marwan Hassani, Boudewijn van Dongen, and Wil van der Aalst. 2019. Online conformance checking: relating event streams to process models using prefix-alignments. International Journal of Data Science and Analytics 8 (10 2019). https: //doi.org/10.1007/s41060-017-0078-6

Andrea Burattin, Sebastiaan J. van Zelst, Abel Armas-Cervantes, Boudewijn F. van Dongen, and Josep Carmona. 2018. Online Conformance Checking Using Behavioural Patterns. In Business Process Management - 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11080), Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke (Eds.). Springer, 250–267. https://doi.org/10.1007/978-3-319-98648-7_15

References

Other Used Sources

Robust Process Mining with Guarantees. Sander J.J. Leemans. Eindhoven University of Technology, PhD thesis 2017.

Wil M. P. van der Aalst and Josep Carmona (Eds.). 2022. Process Mining Handbook. Lecture Notes in Business Information Processing, Vol. 448. Springer, Berlin. https://doi.org/10.1007/978-3-031-08848-3