Technische Universiteit Eindhoven University of Technology

Aligning Observed and Modeled Behavior

An overview of alignments, current research and future challenges

by

Boudewijn van Dongen

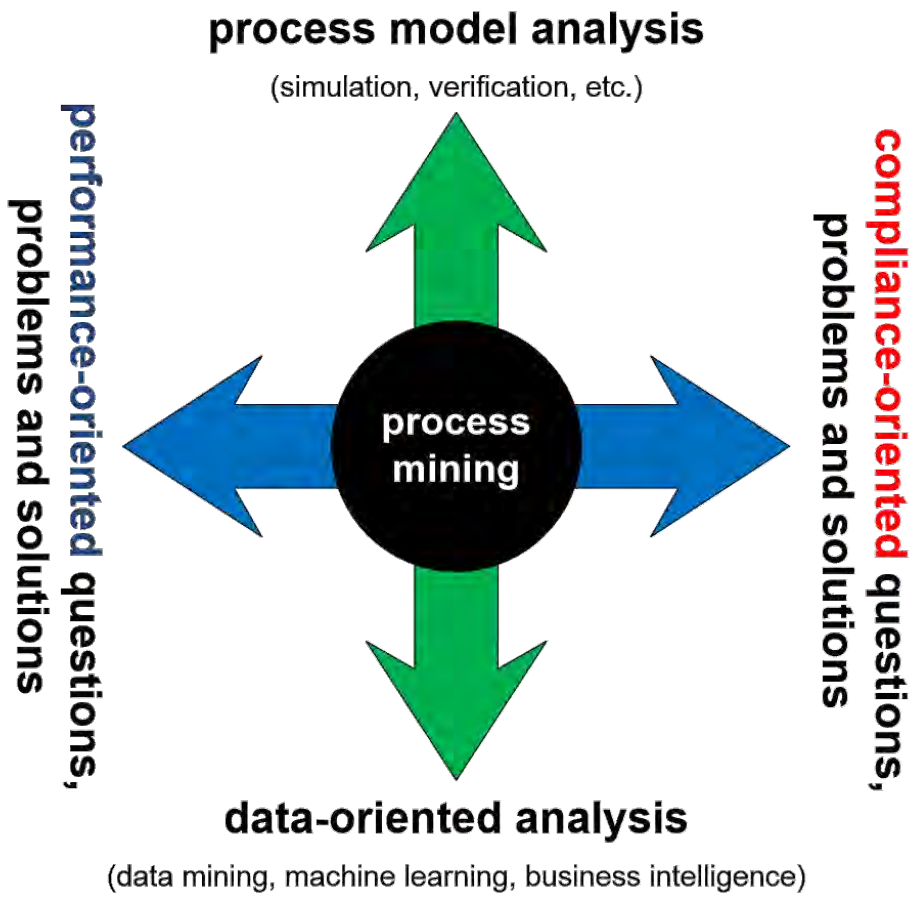
Eindhoven University of Technology

Where innovation starts

Process Mining

The goal of process mining is to extract gain insights into business processes by analyzing operational behavior captured in event logs, through:

- Discovering processes
  - How do people behave?
- Compliance oriented
  - Where and why do people deviate?
- Performance oriented
  - Where are bottlenecks in my processes?



Deviations in Business Processes

Het Nieuwe Werken
@ikzie

Follow

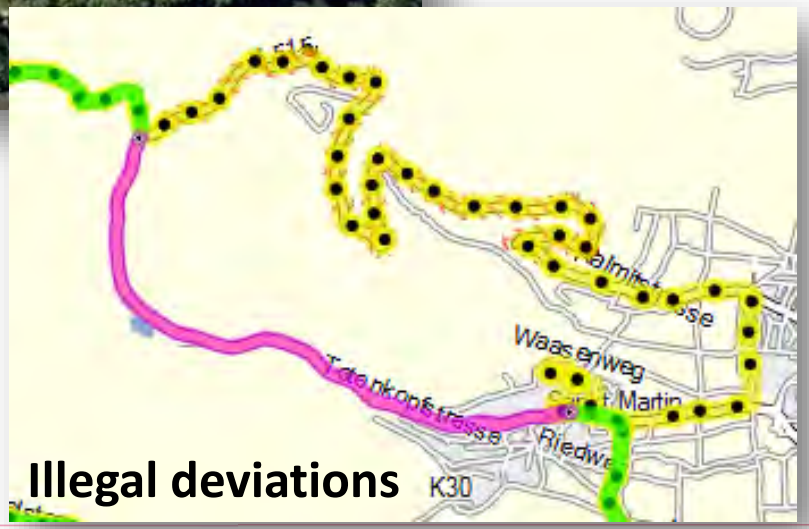
The commonality between a process manual and a concrete barrier: it's been there for years and everybody walks around it.
#olifantenpaadjes

Reply Retweet Favorite

Outline

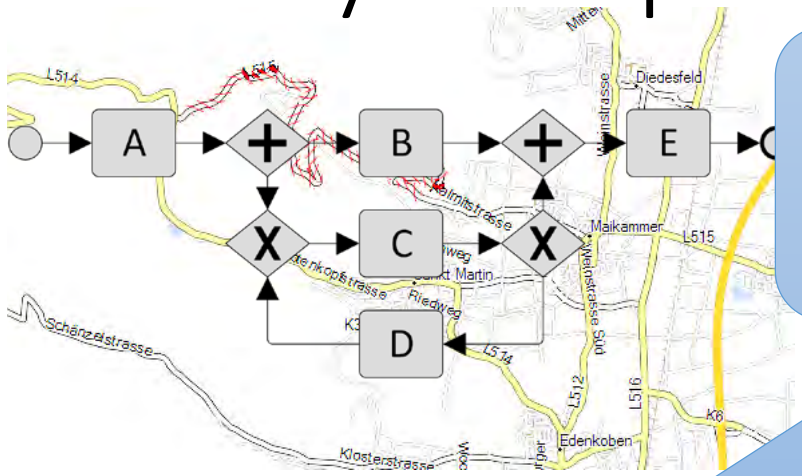
- Introduction to alignments
- Computing alignments
- Active research
- Online alignments
- Conclusions
- Future challenges

Typical Deviations



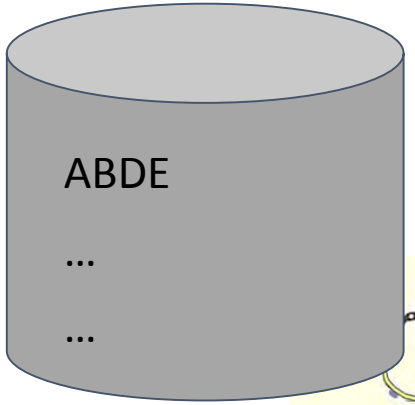
Alignments: A way to explain deviations

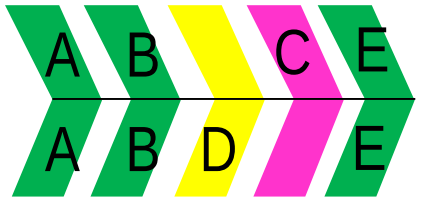
Model:

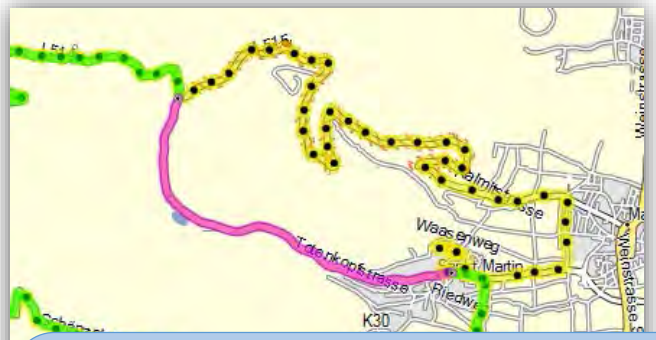


An alignment contains the most likely execution of the model, corresponding to an observed execution

Event log



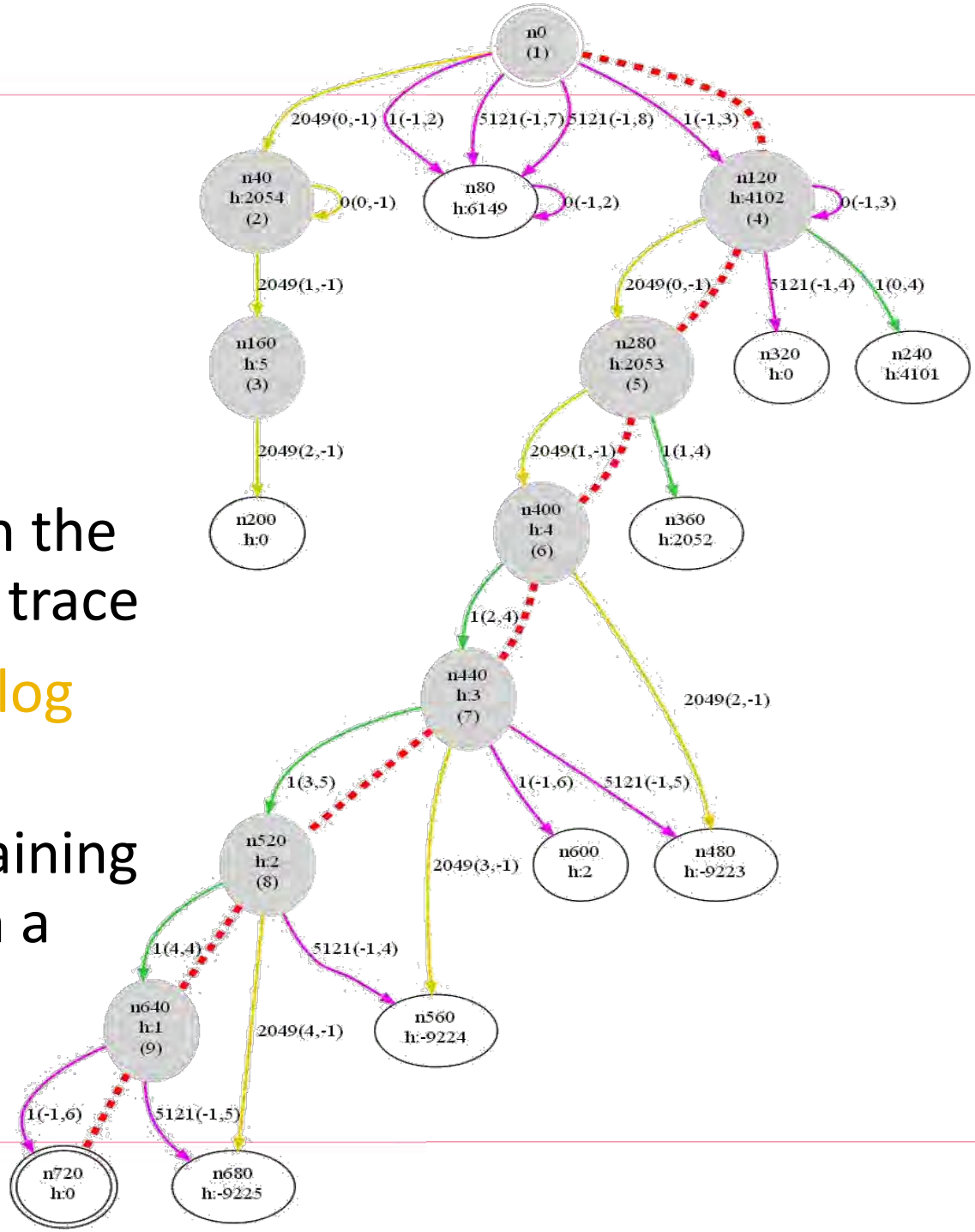




An alignment shows where deviations occurred and why these deviation are considered as such

Computing Alignments

- The search space is a “product” of the statespace of the model and the trace
- Each node is a combination of a state in the model and the remaining events in the trace
- Each arc is a move on model, move on log or a synchronous move
- A heuristic function estimates the remaining distance to the final node (i.e. model in a final state and all events executed)
- We find a shortest path!



Computing Alignments (A*)

```
Initialize PriorityQueue q
While (peek(q) is not target t)
  VisitedNode n = head(q)
  add n to the considered nodes
  For each edge in the graph from node(n) to m
    If m was considered before, continue
    If m is in the queue with lower cost, continue
    If m is in the queue with higher cost, update and reposition it
    If m is new,
      compute an estimate for the remaining distance to t
      V = new VisitedNode(m)
      set n a predecessor for v
      add v to the priority queue
Return head(q)
```

O(log(size q)+α)

O(log(size q)+ α)

?

O(log(size q)+ α)

Computing Alignments (A* with fast lowerbound)

```
Initialize PriorityQueue q
While (peek(q) is not target t)
  VisitedNode n = head(q)
  If the estimate for node(n) is not exact
    Compute the exact estimate for the remaining distance to t and
    add n to the priority queue
    continue
  add n to the considered nodes
  For each edge in the graph from node(n) to m
    If m was considered before, continue
    If m is in the queue with lower cost, continue
    If m is in the queue with higher cost, update and reposition it
    If m is new,
      compute a fast lowerbound for the remaining distance to t
      v = new VisitedNode(m)
      set n a predecessor for v
      add v to the priority queue
Return head(q)
```

O(log(size q)+α)

?

O(log(size q)+ α)

O(log(size q)+ α)

O(1)

O(log(size q)+ α)

Estimating remaining distance

- Naïve estimation: 0 Trivial
- Smarter estimation: trace Trivial
- LP-based estimation: "Polynomial"
  - Minimize $c.x$
    Where $A.x = r$
    $c.x \geq c.x'$
- ILP-based estimation: Exponential
- Hybrid ILP (1 sec for "I" part) Exponential
- Caching LP basis solutions Exponential

Little to no difference in practice

Implementations: Estimator Versions

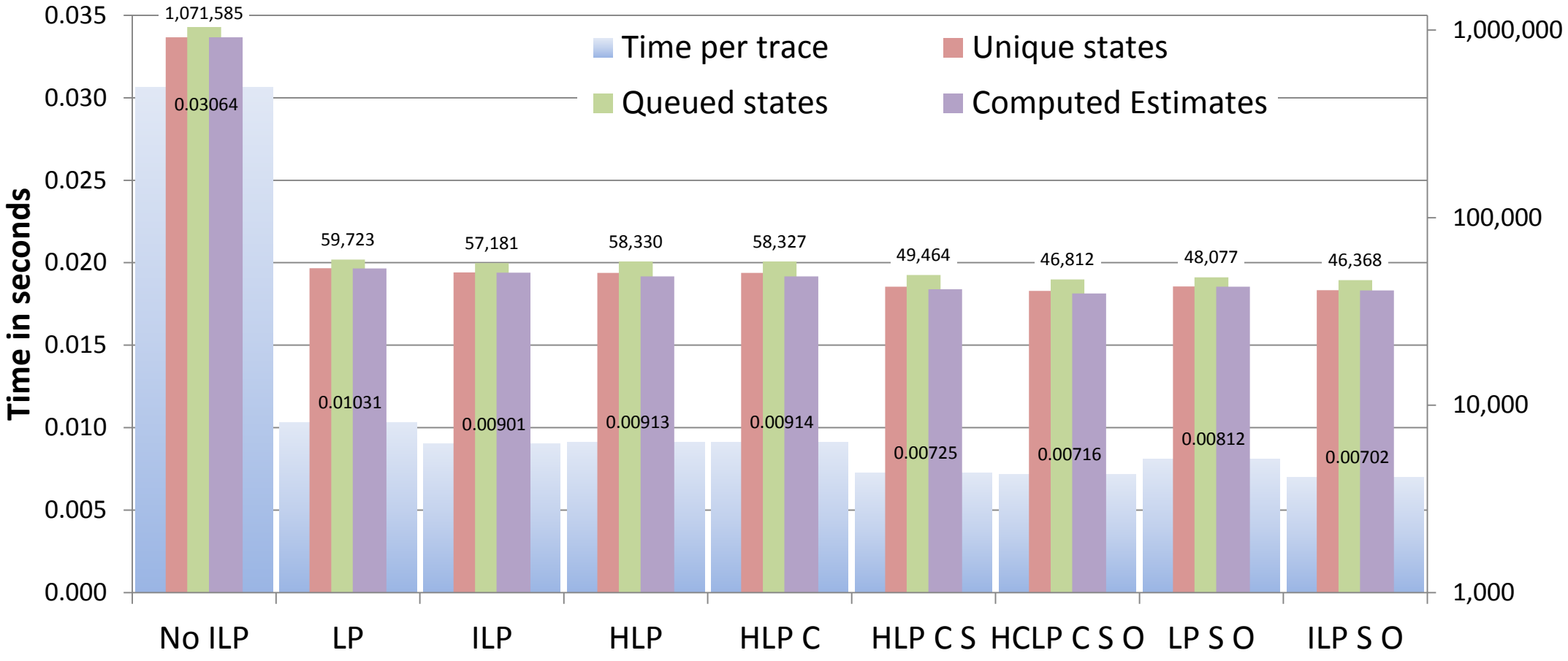
Petri nets:

- Dijkstra (estimator 0)
- Naive (estimator parikh)
- ILP (estimator using ILP)
- Basis caching
- Fast lowerbounds

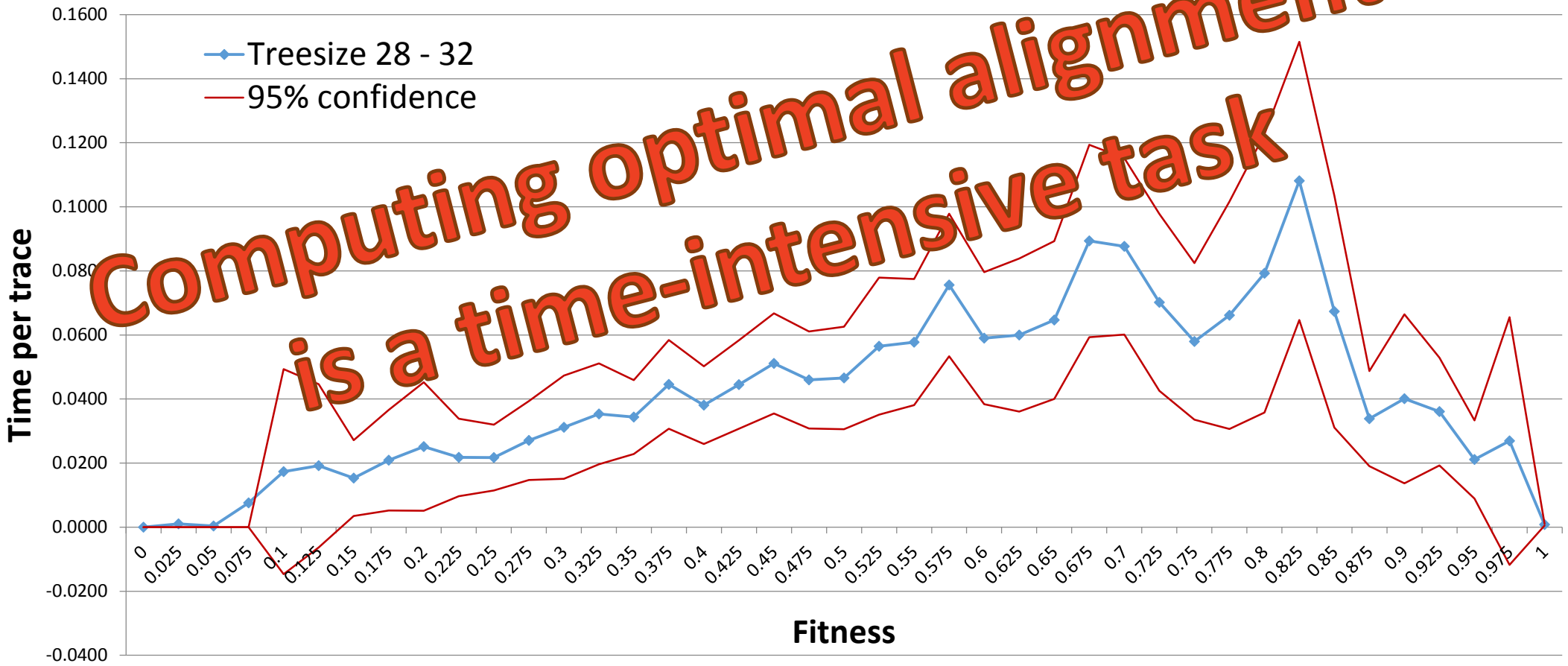
Process Trees:

- Naive (estimator parikh)
- LP (estimator using LP)
- Hybrid ILP (ILP & LP)
- Special constraints for OR
- Fast lowerbounds
- Basis caching within LP
- Statespace reduction (stubborn sets)

Time vs. Space

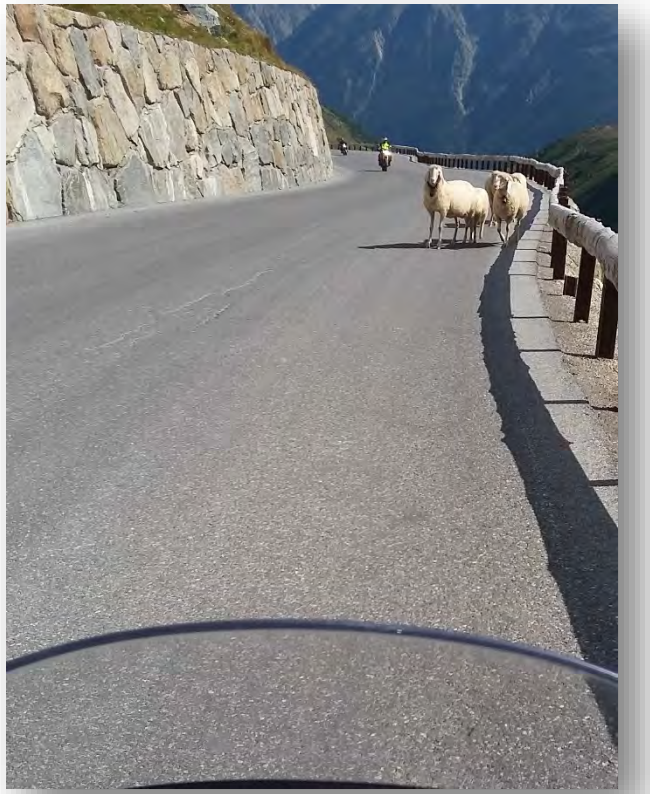


Time vs. fitness



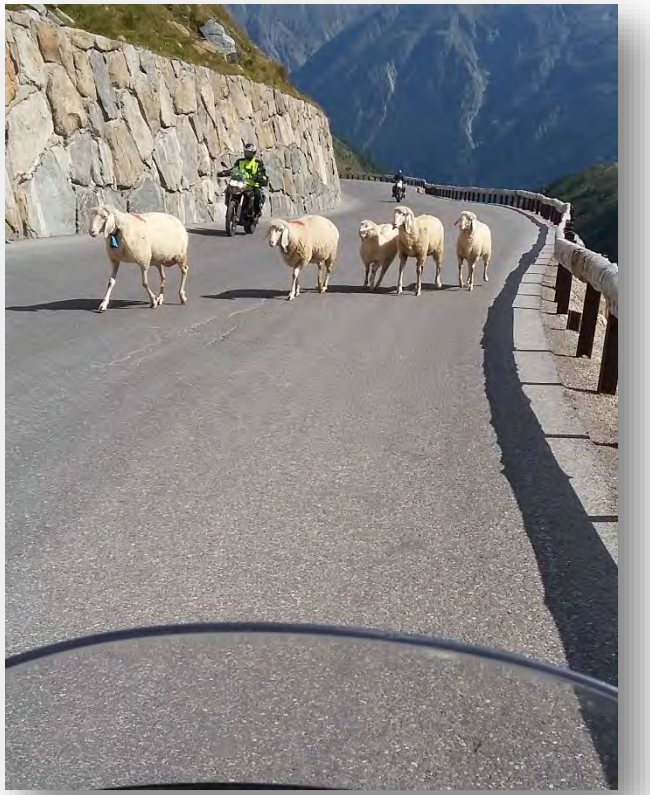
Computing optimal alignments is a time-intensive task

Active research on alignments I



- Conformance checking foundations
  - Fitness (Arya Adriansyah)
  - Precision (Jorge Munoz)
- Process discovery
  - Genetic algorithms (PhD work Joos Buijs)
  - Model Repair (Dirk Fahland, Artem Polyvyany)
  - Alignments without models (Xixi Lu)
- Automated compliance checking
  - Needs *n*-to-*m* mappings for activities / transitions (Elham Rhamezani)

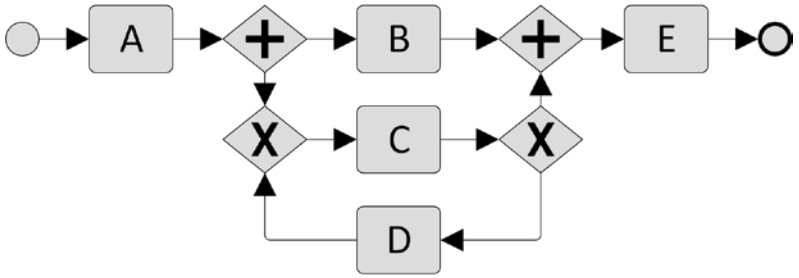
Active research on alignments II



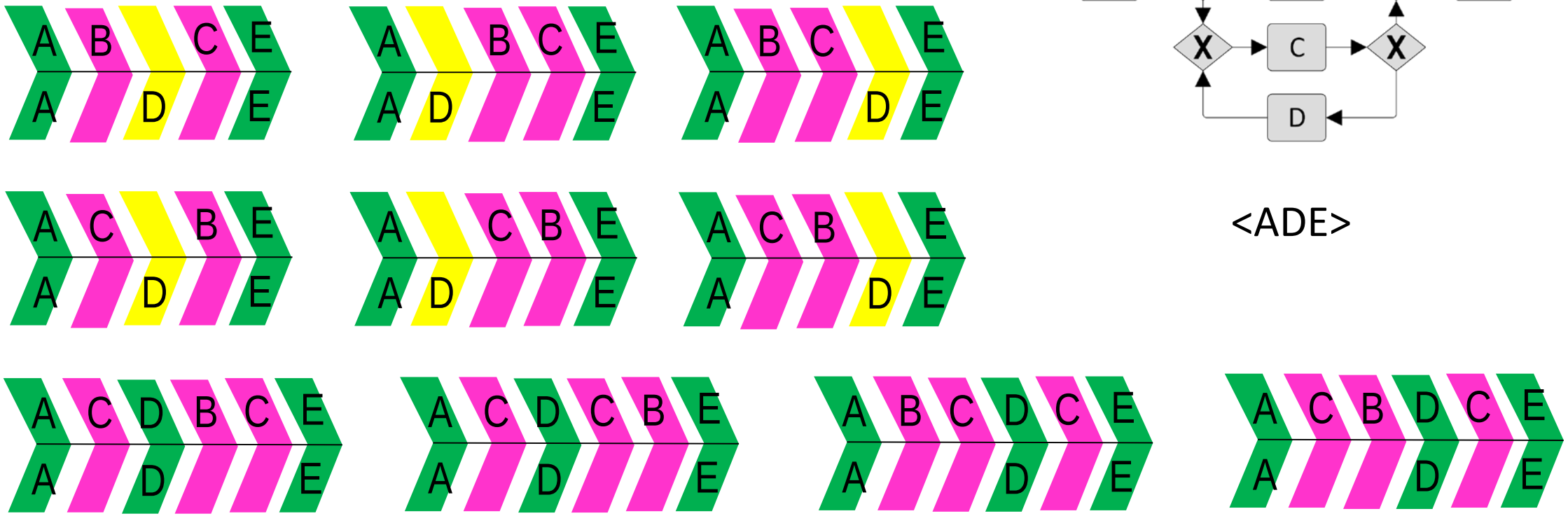
- Data/Resource aware alignments
  - Include data into the statespace (Felix Mannhardt)
  - First control-flow, then data (Massimiliano De Leoni)
  - Multi perspective
- Process model animation
  - Animate models based on alignments (Sander Leemans)
- Online Alignments
  - Requires “constant” time and memory (Andrea Burattin, me)

Alignments – Some Insights

Consider the model and trace on the right



<ADE>



Online conformance checking

Requirements:

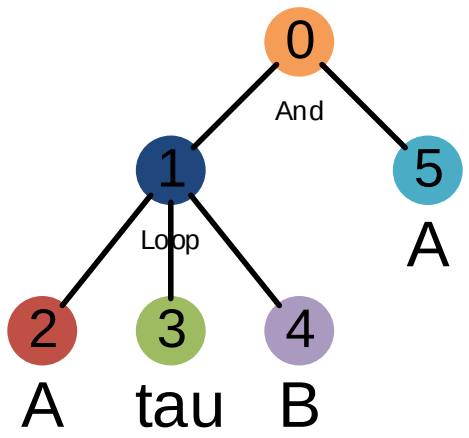
- Compute alignments without knowledge of the future
- Use constant time per event
- Use constant space

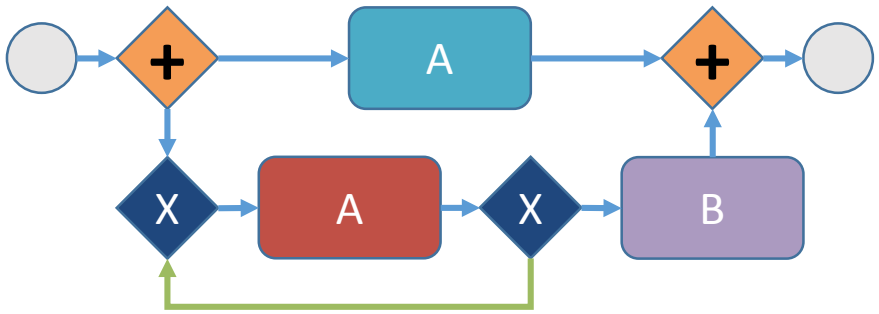
For this, we introduce a representation of alignments, such that between each pair of synchronous moves:

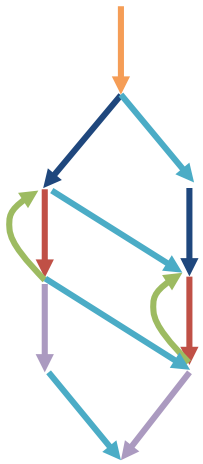
- Log moves are done first,
- Model moves are postponed until strictly necessary

Aligning process models – search space

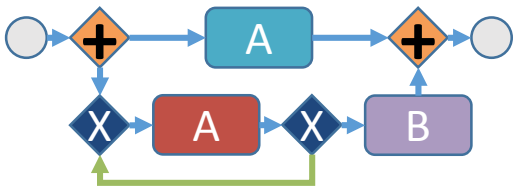
- Aligning process models requires traversing a state space which is the synchronous product of the model's state space and the trace



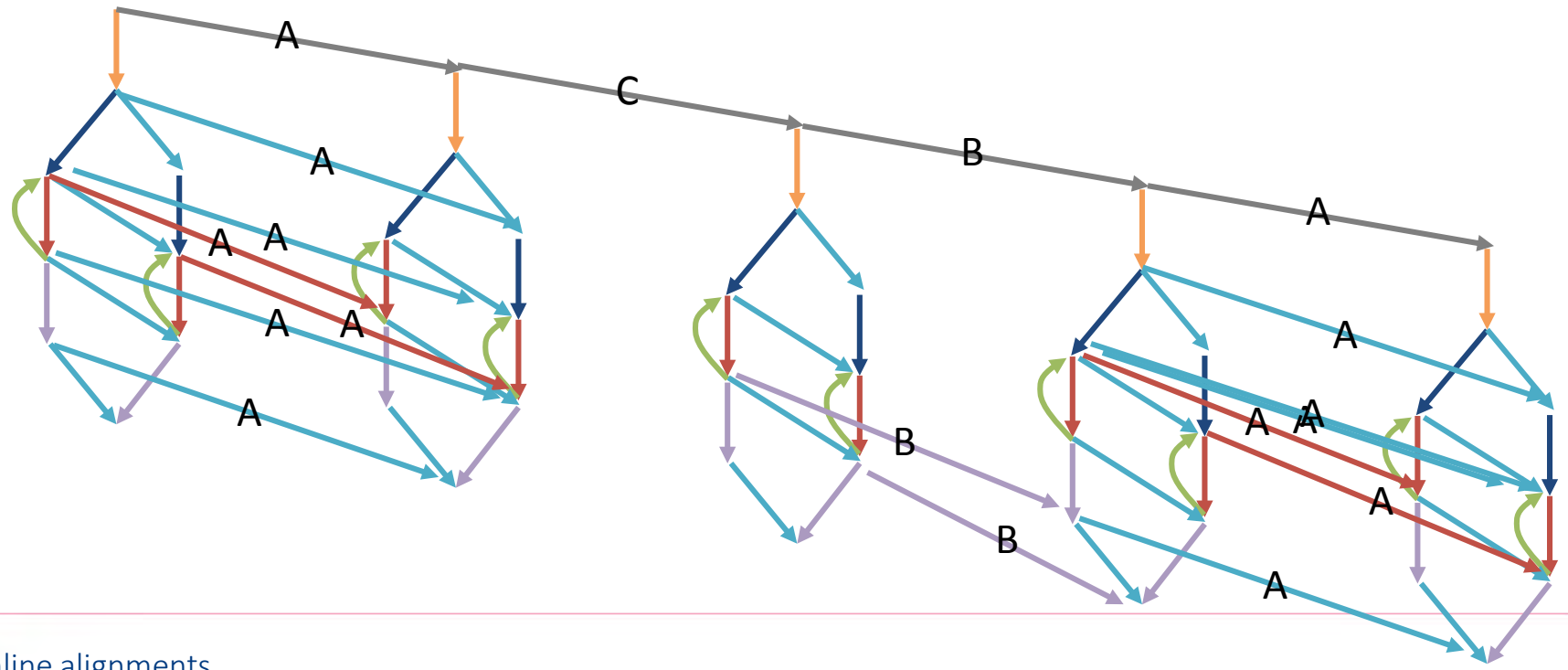




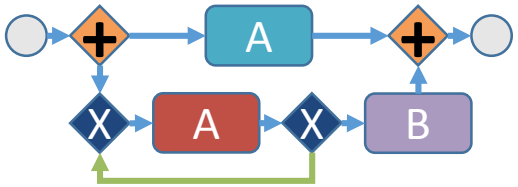
Aligning process models – search space



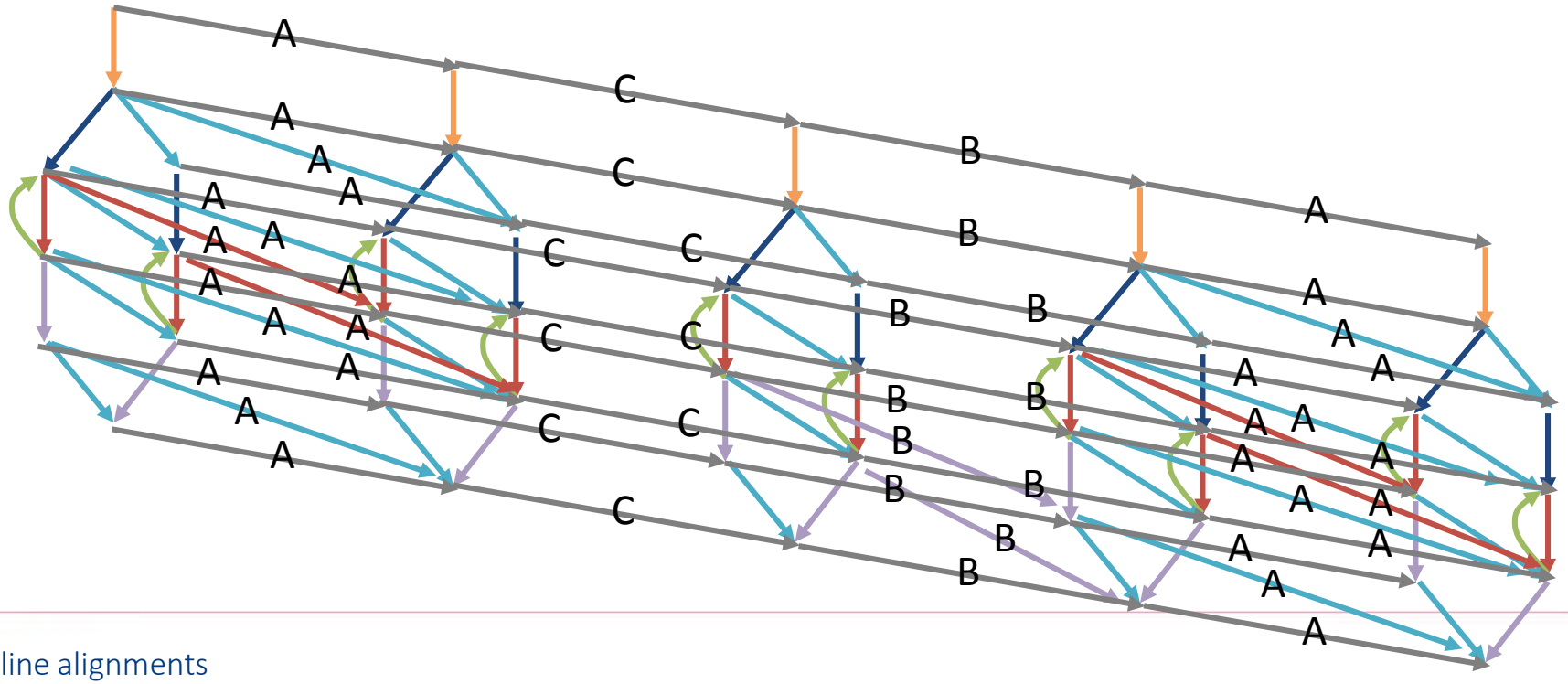
<A C B A>



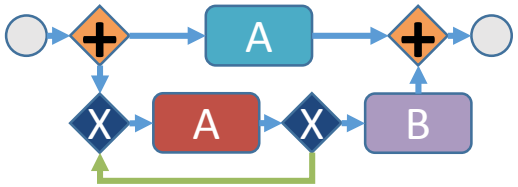
Aligning process models – search space



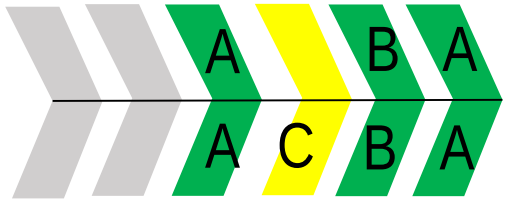
<A C B A>

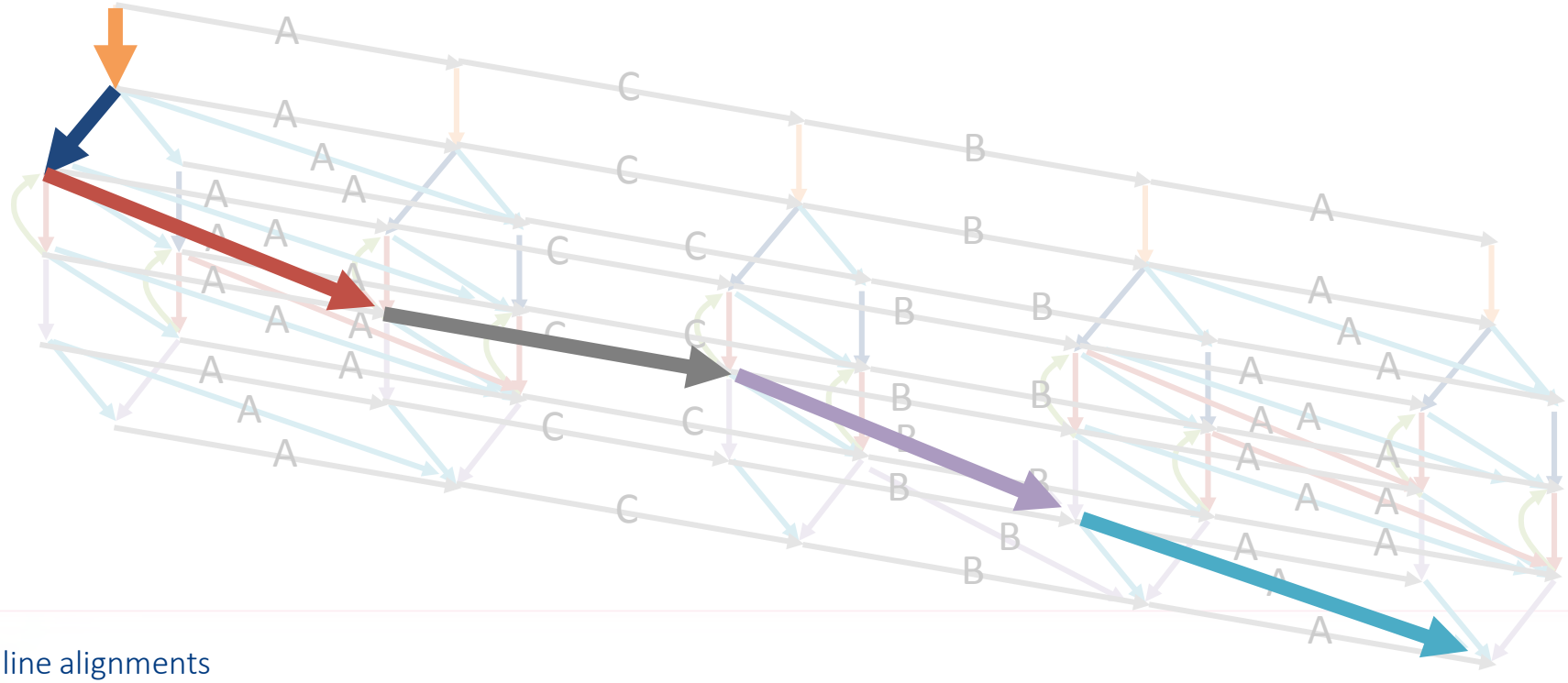


Aligning process models – search space



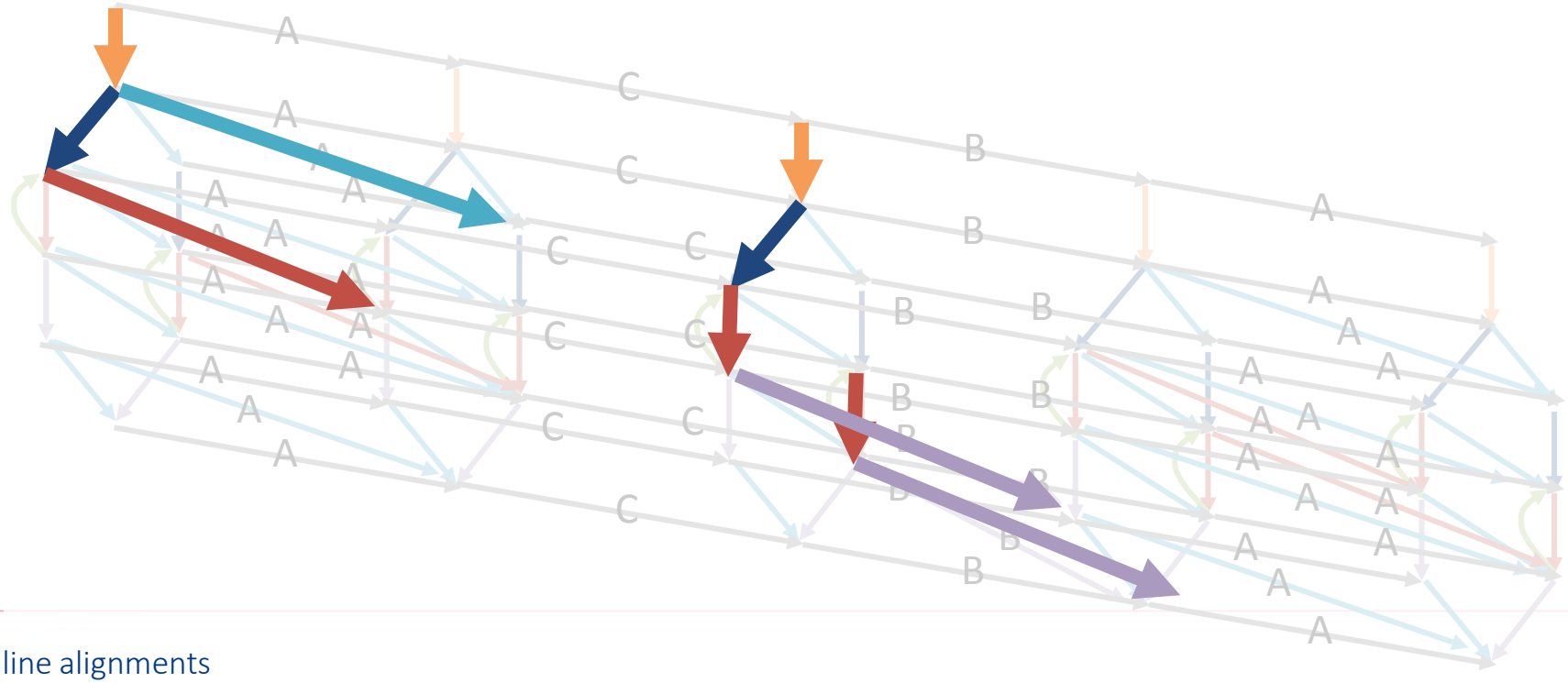
<A C B A>





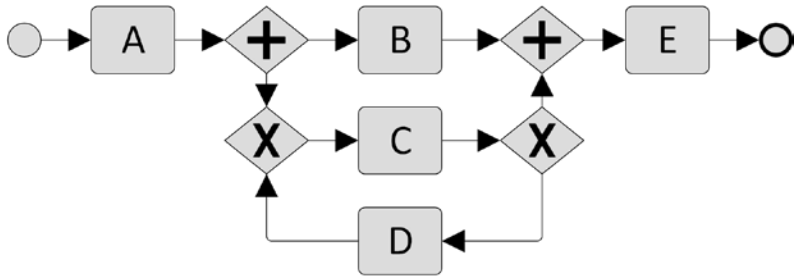
Properties of the shortest path

For a given label A, there is a limited number of shortest paths that end in a synchronous move on some activity A between a state s and another state s’



Redefined search space

- For each state s and label A compute the shortest paths ending in a synchronous move on some activity labelled A.
- The maximum number of modelmoves allowed in this path is limited by a given parameter “depth”



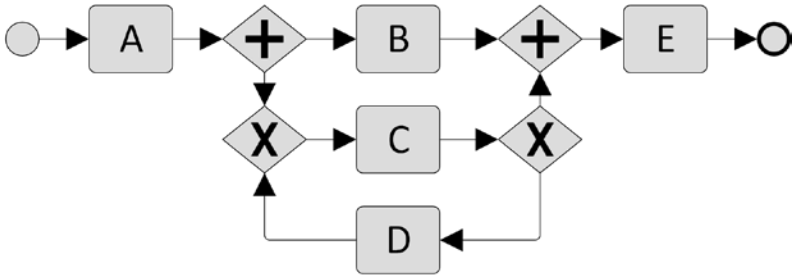
To execute an activity D in the state after A was executed, we need a sequence of model moves consisting of <C> with length 1.

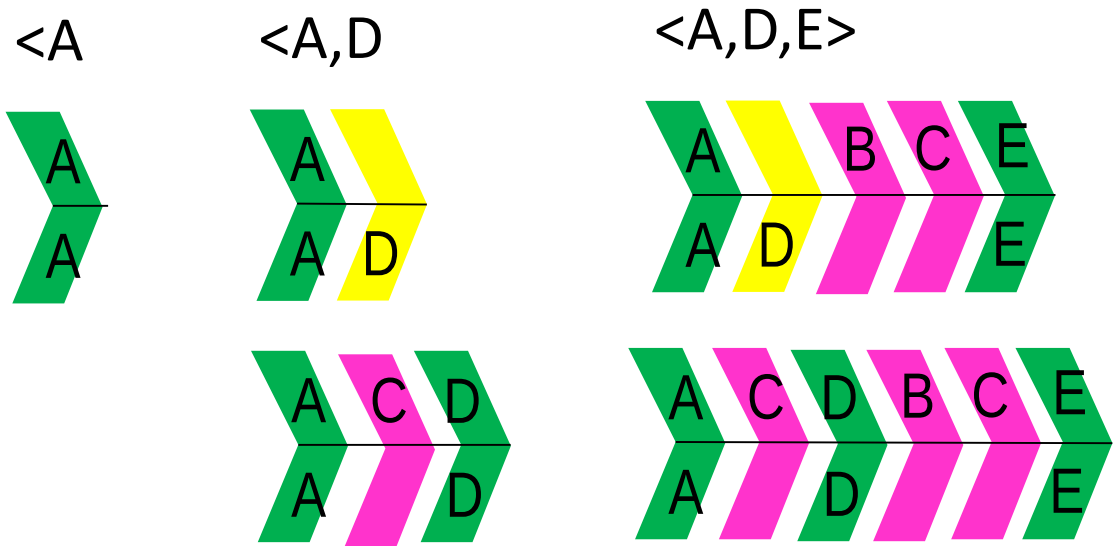
Another possibility is a path <C,D,C> with length 3, hence this is not the shortest sequence.

Online conformance checking

- Suppose we have an oracle that produces, given a state s and an label A some reachable states s' trough a series of model moves followed by a synchronous move on A, and the cost of reaching s'
- For each trace, we keep (an ordered list of) partial alignment(s) for the events in the trace we have observed so far
- For each new event, we update all partial alignments using the oracle

Online conformance checking - example





- For each event, we build a partial alignment
- Between two synchronous moves, there's always first log moves, then model moves (if any)
- We may need to keep more than one partial alignment
- We may even need to keep sub-optimal partial alignments

Online conformance checking - parameters

- The oracle needs a *depth* parameter, indicating the maximum number of model moves allowed to get to a synchronous move
- The replayer needs to keep a number of possible partial alignments in a *queue* (sorted by cost)
- Per event, the oracle needs to look through the search space in a depth first manner, until maximum depth is reached. Hence the time per event is bounded by a constant (given the model)
- The number of partial alignments to keep per trace is bounded by the size of the statespace.

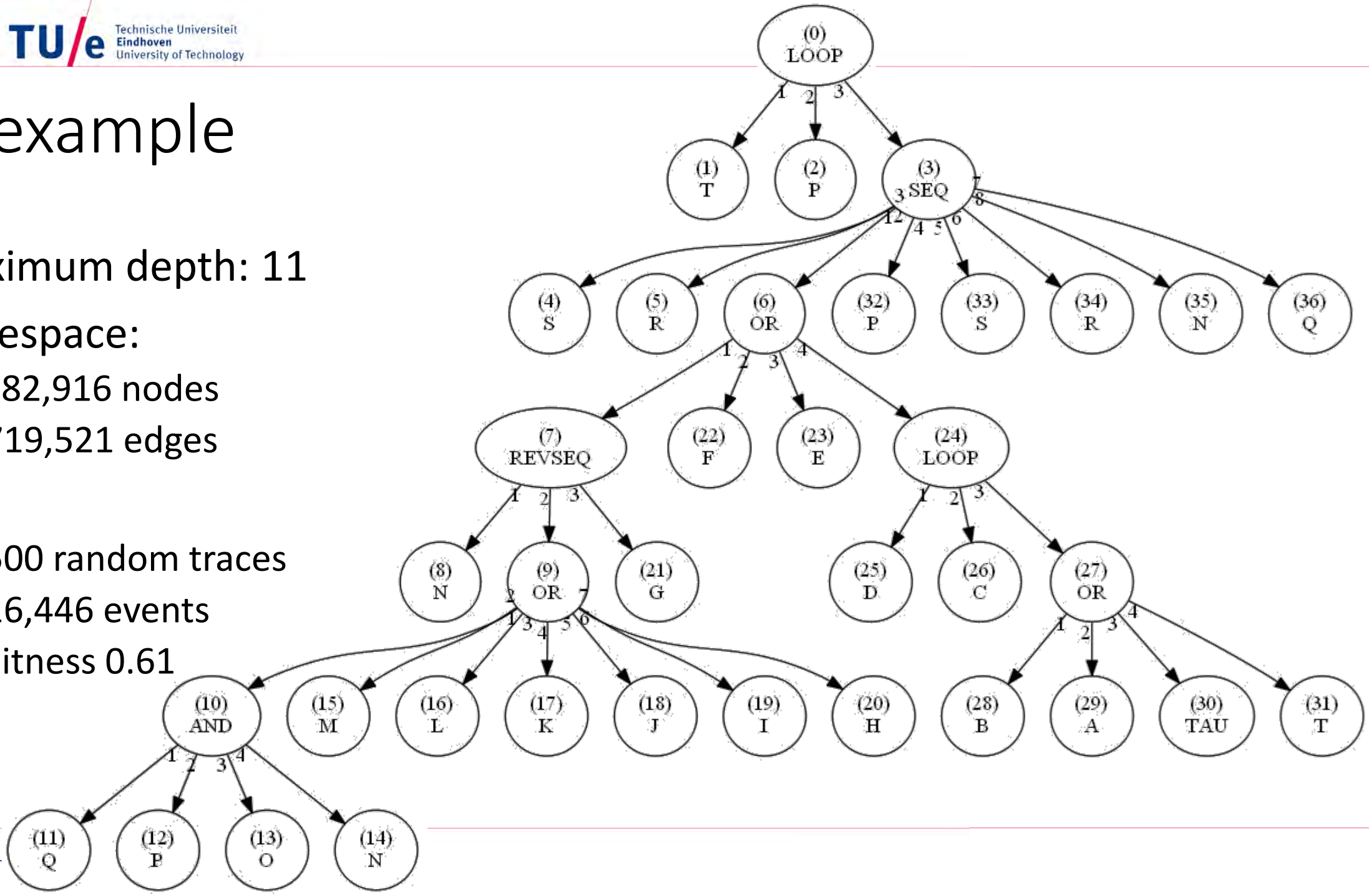
Online conformance checking - guarantees

Given sufficient depth and sufficient queue size:

- Each optimal partial alignment gives a lower bound on the total alignment cost.
- The final alignment (after ending the trace) is an optimal alignment.
- We need constant time and space, as long as the actual alignment is not required, but only the cost of deviating.
- If the actual alignment is needed, the space required scales linearly in the length of the trace.

An example

- Maximum depth: 11
- Statespace:
  - 82,916 nodes
  - 719,521 edges
- Log:
  - 500 random traces
  - 16,446 events
  - Fitness 0.61



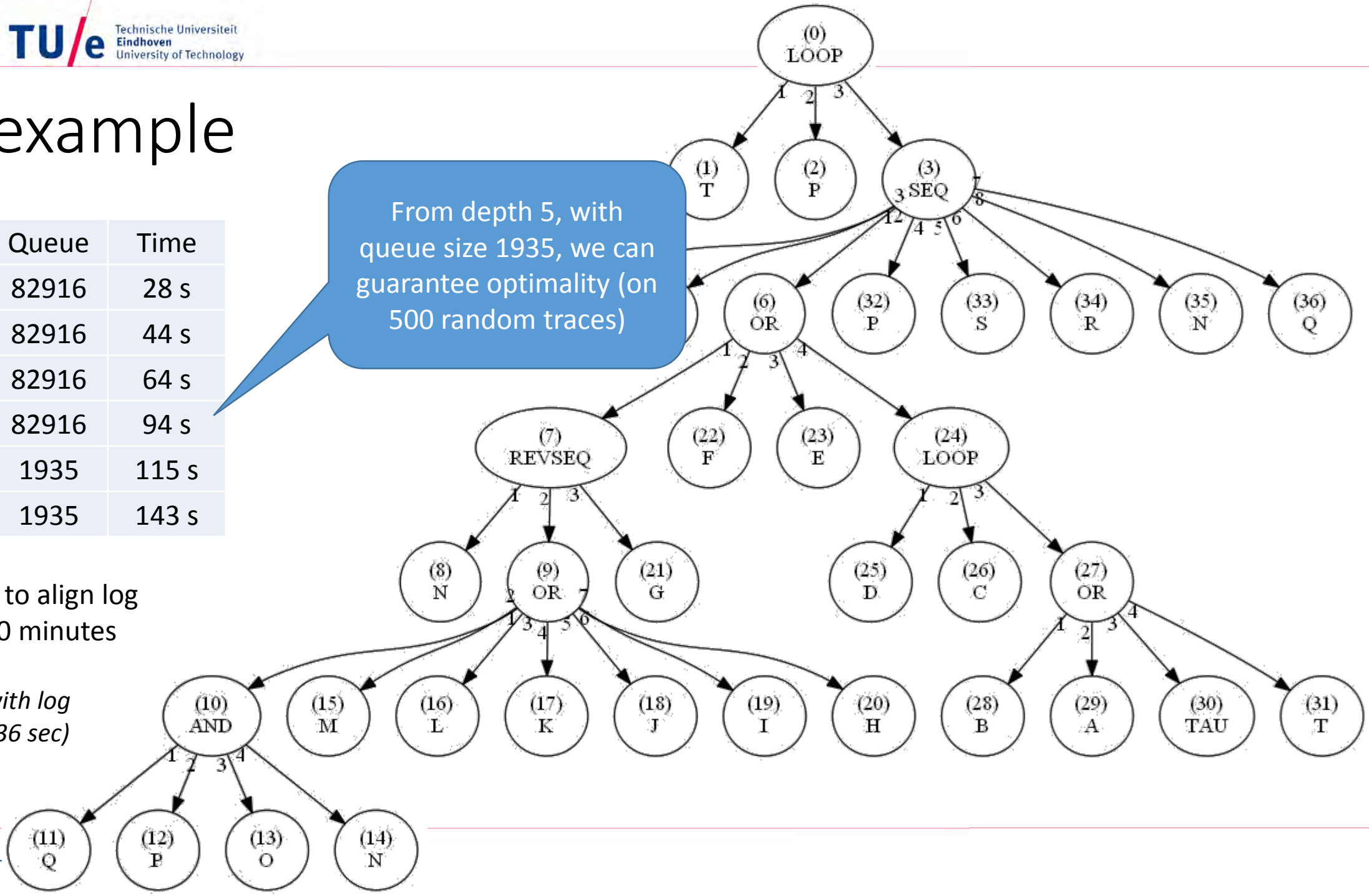
An example

Depth	Queue	Time
0	82916	28 s
1	82916	44 s
2	82916	64 s
3	82916	94 s
5	1935	115 s
8	1935	143 s

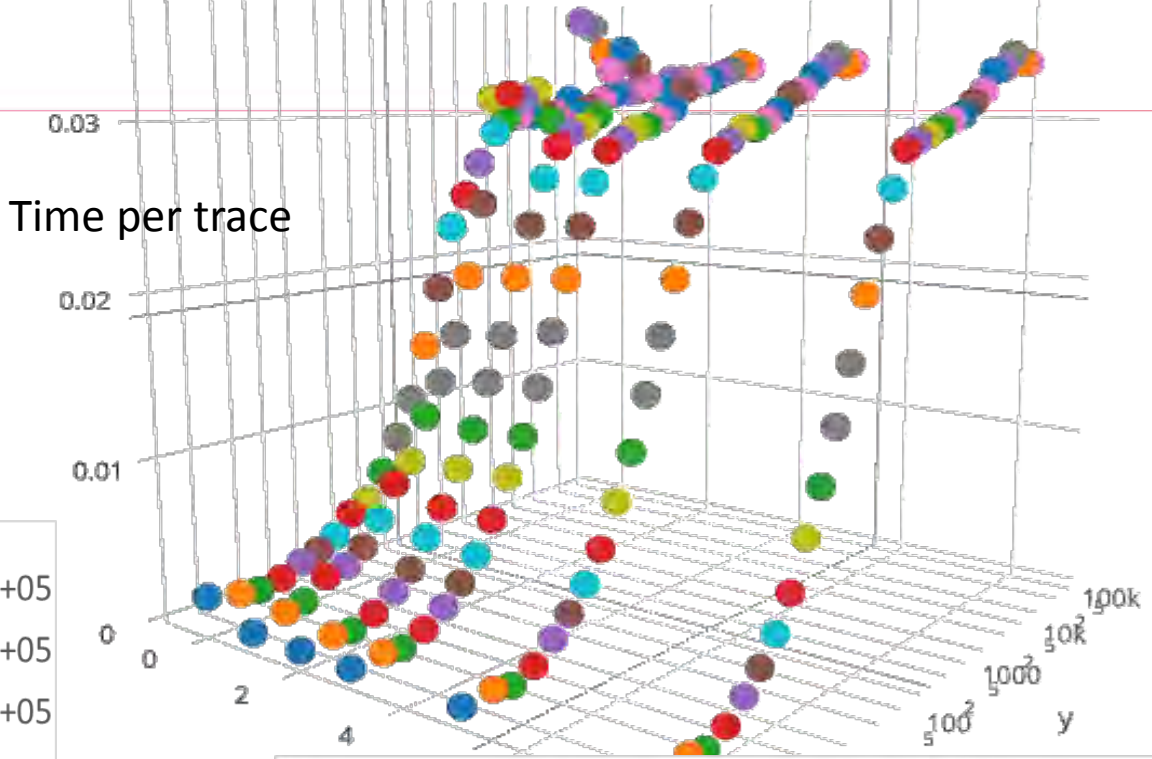
From depth 5, with queue size 1935, we can guarantee optimality (on 500 random traces)

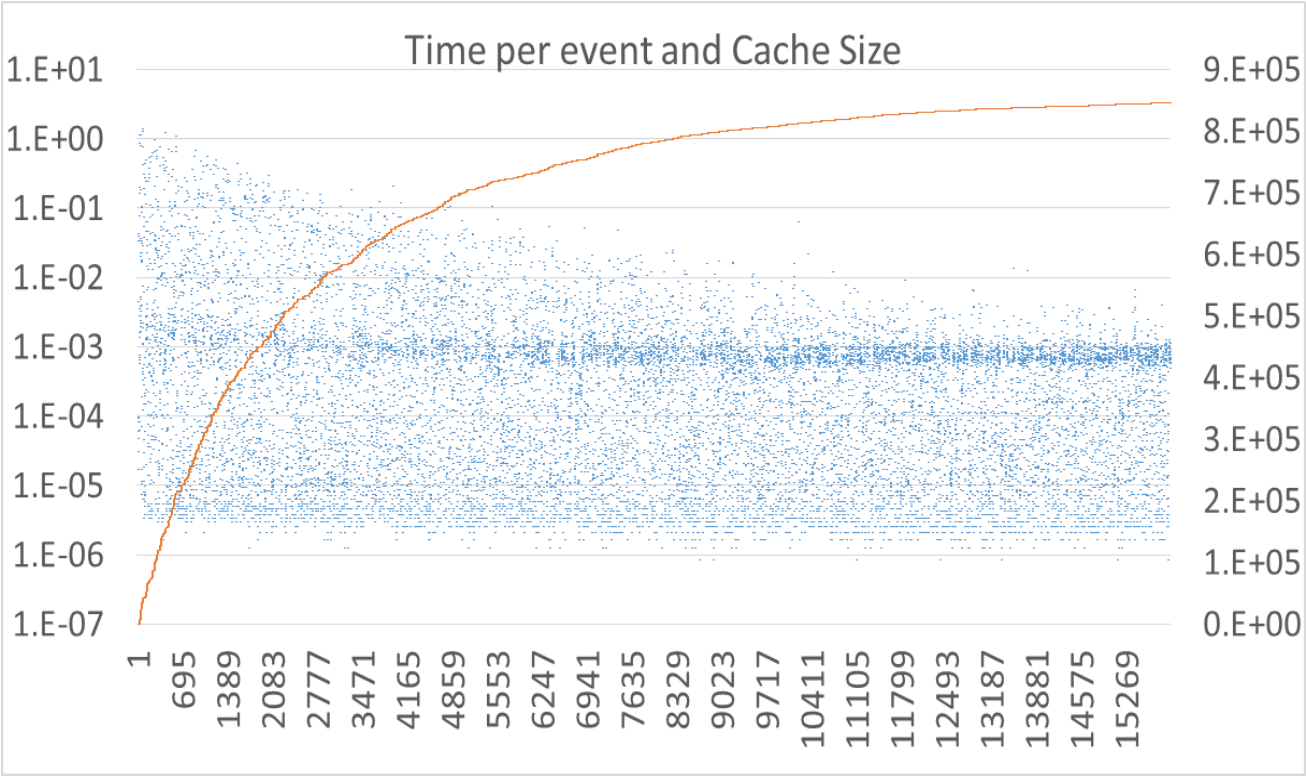
Time needed to align log offline: +/ - 50 minutes

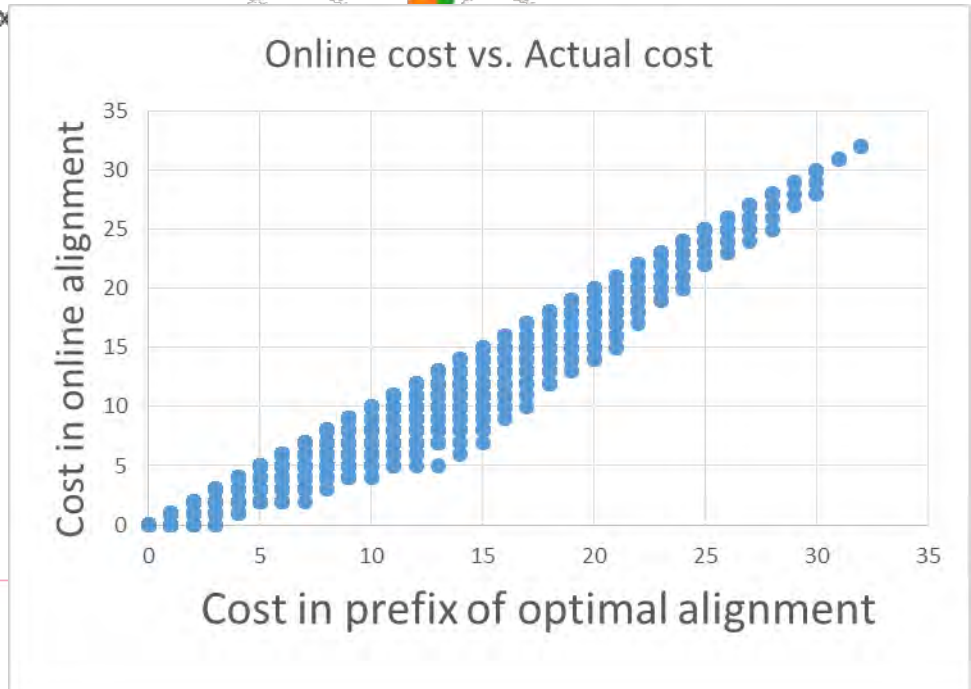
(same model with log fitness 0.81: 136 sec)



An example

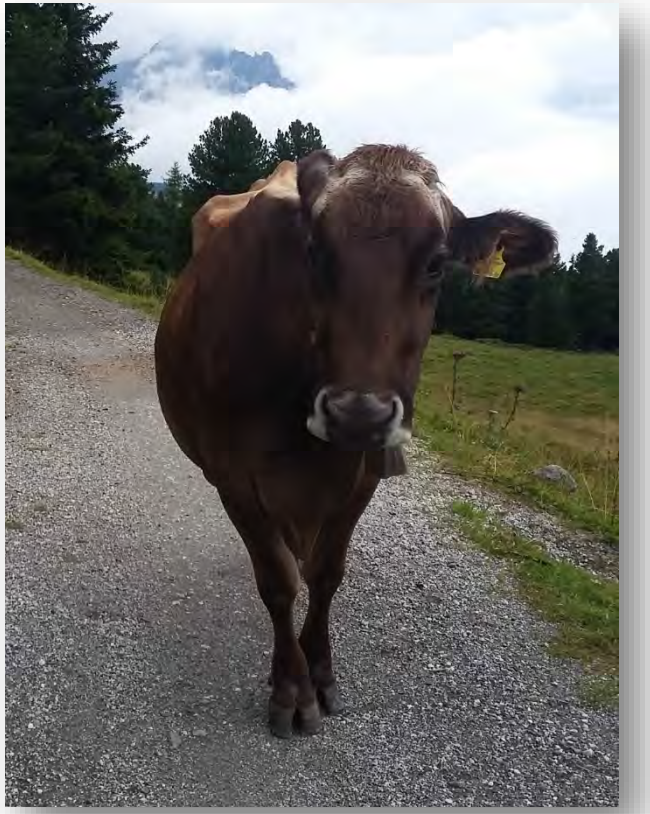

Time per trace
0.03
0.02
0.01
0
0
2
4
x
100
1000
10k
100k
y


Time per event and Cache Size
1.E+01
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
1.E-07
9.E+05
8.E+05
7.E+05
6.E+05
5.E+05
4.E+05
3.E+05
2.E+05
1.E+05
0.E+00
1
695
1389
2083
2777
3471
4165
4859
5553
6247
6941
7635
8329
9023
9717
10411
11105
11799
12493
13187
13881
14575
15269


Online cost vs. Actual cost
Cost in online alignment
Cost in prefix of optimal alignment
0
5
10
15
20
25
30
35

Conclusions

- Alignments explain relations between event logs and process models
- Alignments are foundational to process mining
- Computing alignments is computationally hard
- Many variants exist, but all guarantee:
  - The projection to the log provides the observed trace
  - The projection to the model provides a valid run thereof



Future challenges

- Computational complexity
  - Especially with data, resources
  - Find tight bounds for online alignments
  - Fast “approximate” alignments
- Determinism of alignments
  - Same choices across alignments
- Obtaining the right models,
  - Translate informal text to formal models
  - Alignments on BPMN
- Industry adaptation


Questions?
Stay clear of trees, align with roads instead!