Basics on Conformance Checking

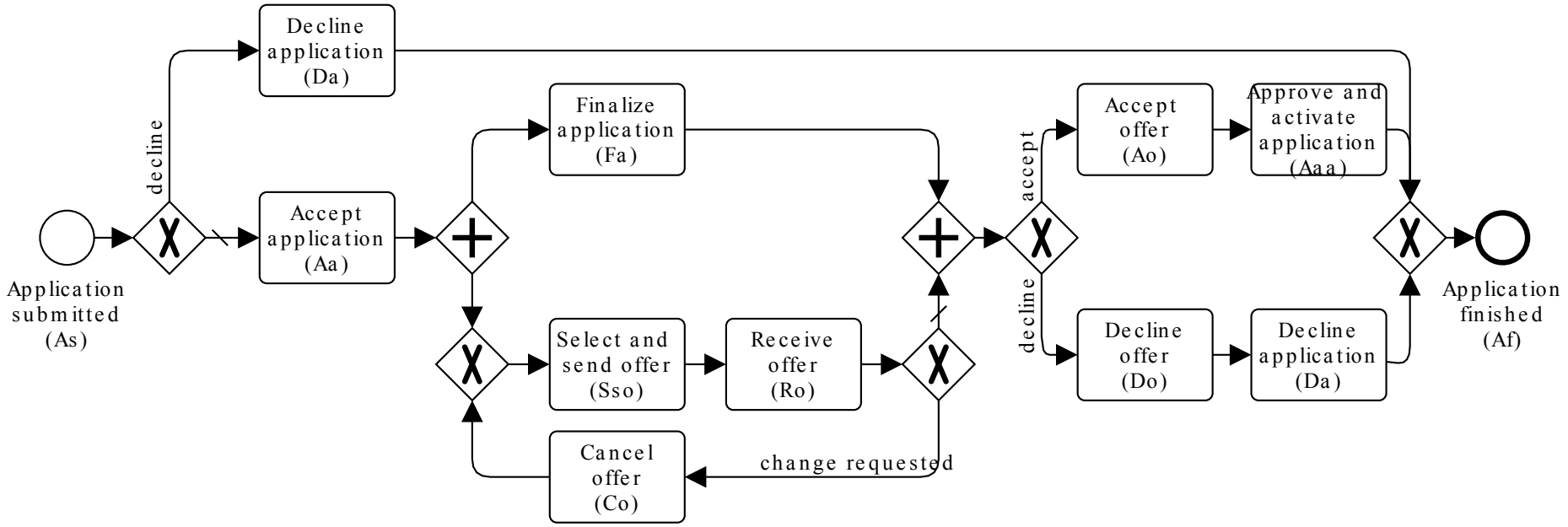
Josep Carmona

The Loan Application Process

A process defines how a loan application is handled once it has been submitted, with the outcome being its positive or negative assessment. A first activity is a check for eligibility of the respective applicant. Subsequently, a decision is taken, which determines whether an application is declined or accepted. If accepted, the application is finalised and an order is selected for further processing.

…

Process Models: BPMN



Control-flow Data Organizational Other: Flexibility, …

Covered Partially Covered Not Covered

Process Models (in Organizations)

- Process Models encompass knowledge in Organizations
- Unfortunately, they are not always:
  1. Known
  2. Understood
  3. Executed correctly
  4. Flexible enough when necessary, …
- Organizations need to have a systematic way to be aware of these problems so that remedies can be put in place: **Conformance Checking**

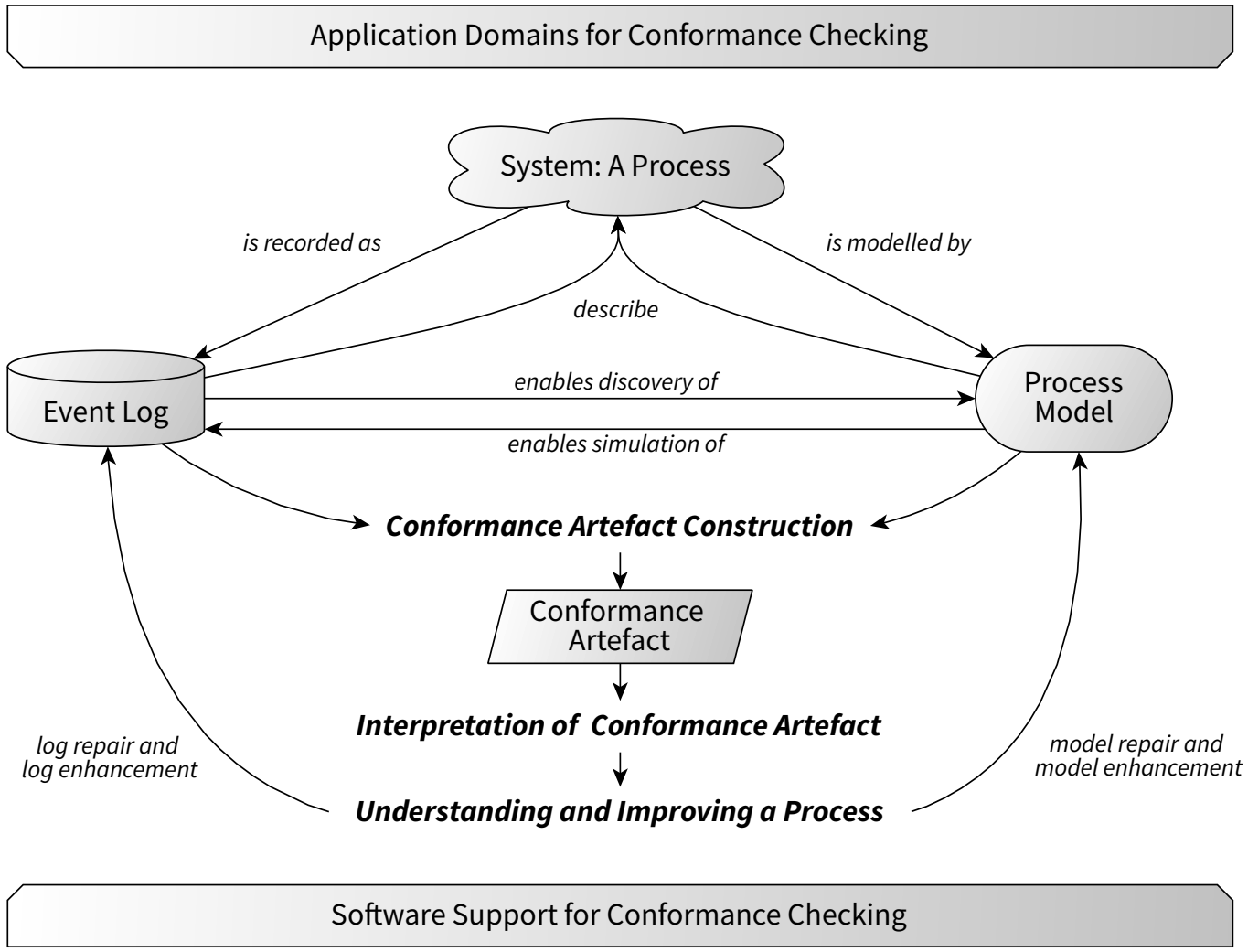
Event Logs

- Event log: recorded executions of a particular process (the reality!!!)
- Trace: Ordered collection of events corresponding to the same execution

Case A5634: < As, Aa, Fa, Sso, Ro, D, Da, Af >

Event	Application	Offer	Activity	Amount	Signed	Timestamp
…	…	…	…	…	…	…
e_{13}	*A*5634		Application submitted	€2,000		Jan 01, 12:31
e_{14}	*A*5634		Accept application	€2,000		Jan 01, 12:32
e_{15}	*A*5635		Application submitted	€5,000		Jan 02, 04:31
e_{16}	*A*5635		Accept application	€5,000		Jan 02, 04:32
e_{17}	*A*5636		Application submitted	€200		Jan 03, 06:59
e_{18}	*A*5636		Accept application	€200		Jan 03, 07:00
…	…	…	…	…	…	…
e_{22}	*A*5634		Finalise application			Jan 03, 09:00
e_{23}	*A*5636		Finalise application			Jan 03, 09:01
e_{24}	*A*5635		Decline application			Jan 03, 09:02
e_{25}	*A*5635		Decline application			Jan 03, 09:03
…	…	…	…	…	…	…
e_{30}	*A*5636	*O*3521	Select and send offer	€500		Jan 04, 16:32
…	…	…	…	…	…	…
e_{37}	*A*5634	*O*3541	Select and send offer	€1,500		Jan 05, 12:32
e_{38}	*A*5636	*O*3521	Receive offer		NO	Jan 05, 12:33
e_{38}	*A*5636	*O*3521	Cancel offer			Jan 05, 12:34
e_{39}	*A*5636	*O*3542	Select and send offer	€500		Jan 05, 13:29
e_{40}	*A*5636	*O*3542	Receive offer		YES	Jan 08, 08:33
e_{41}	*A*5636	*O*3542	Accept offer			Jan 08, 16:34
e_{42}	*A*5634	*O*3541	Receive offer		NO	Jan 10, 10:00
…	…	…	…	…	…	…
e_{54}	*A*5634	*O*3541	Decline offer			Jan 10, 10:04
…	…	…	…	…	…	…
e_{64}	*A*5634		Decline application			Jan 10, 10:05
e_{65}	*A*5634		Application finished			Jan 10, 10:06
e_{66}	*A*5636		Approve and activate application			Jan 10, 10:07
e_{67}	*A*5636		Application finished			Jan 10, 10:08
…	…	…	…	…	…	…

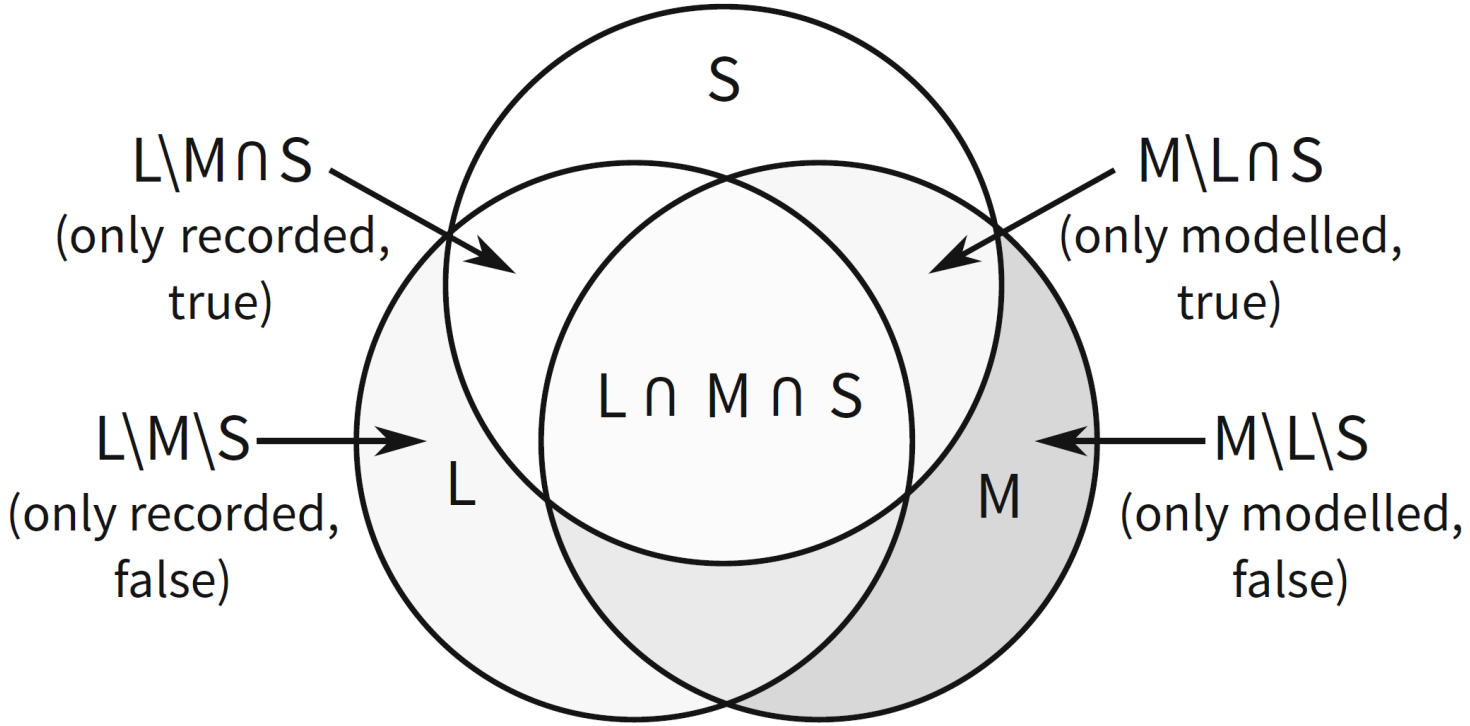
The Spectrum of Conformance Checking



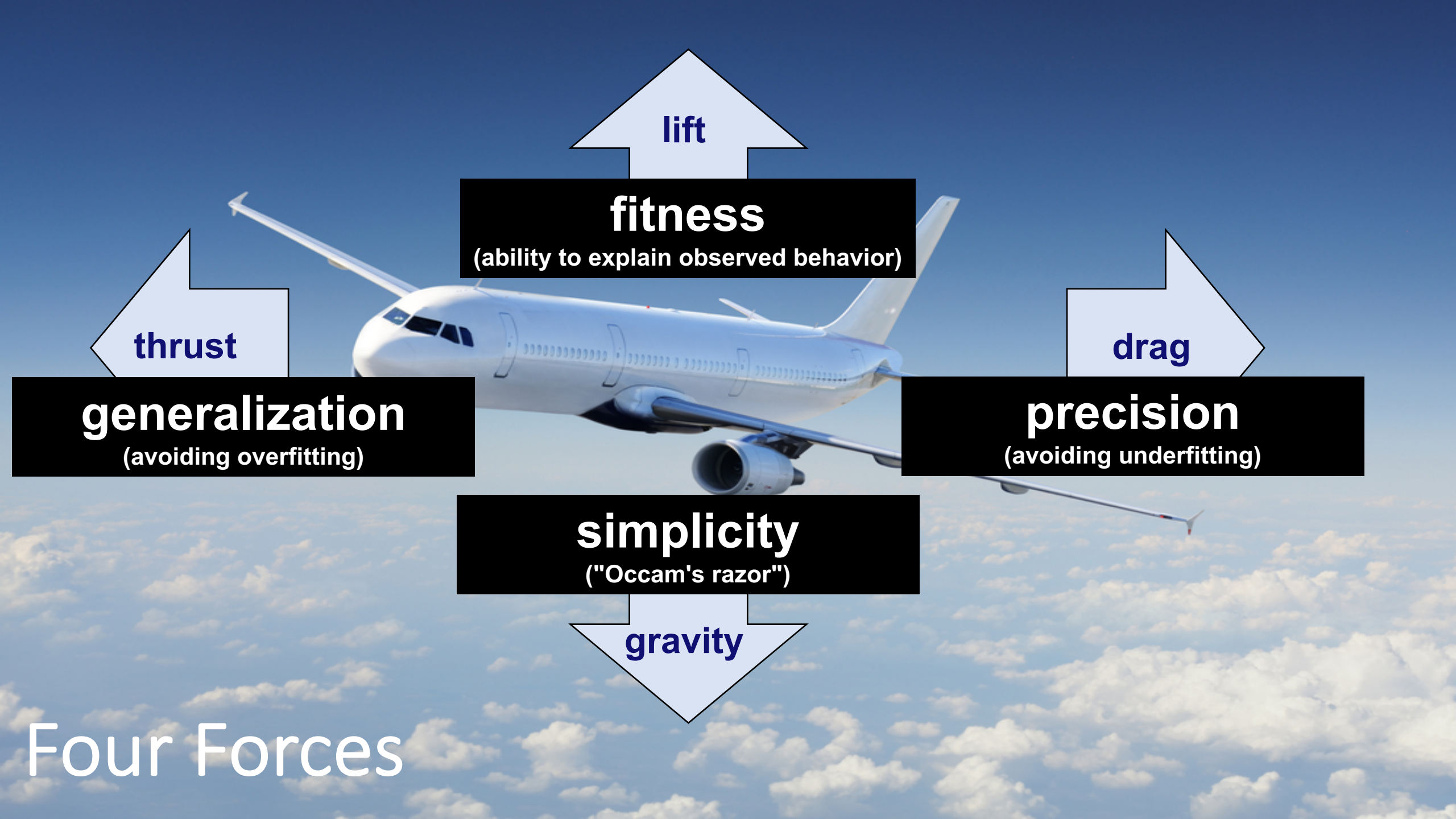
The Spectrum of Conformance Checking

- Conformance checking is about detecting and analyzing deviations between observed/recorded and modelled behavior.
- Modelled behaviour: Process models
- Observed behaviour: Event data (e.g., event logs)
- Conformance Artefact Construction:
  - Replaying event data on process models allows to discover deviations.
  - Two replay methods:
    - Token-Replay (intuitive, heuristic)
    - Alignments (sound, optimal).

Overview of Behaviours



Behaviour as recorded in an event log L, as captured in a process model M, and as shown by a system/process S

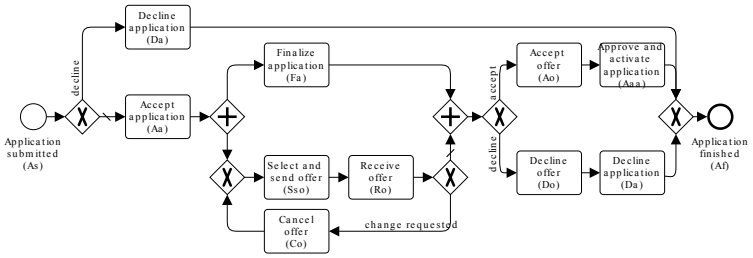

lift
fitness
(ability to explain observed behavior)
thrust
drag
generalization
(avoiding overfitting)
precision
(avoiding underfitting)
simplicity
("Occam's razor")
gravity

Four Forces

Principal Dimensions: fitness and precision

Fitness: Has the recorded behaviour been modelled?

Event	Application	Offer	Activity	Amount	Signed	Timestamp
...	...	...	...	...	...	...
e_{13}	A5634		Application submitted	€2,000		Jan 01, 12:31
e_{14}	A5634		Accept application	€2,000		Jan 01, 12:32
e_{15}	A5635		Application submitted	€5,000		Jan 02, 04:31
e_{16}	A5635		Accept application	€5,000		Jan 02, 04:32
e_{17}	A5636		Application submitted	€200		Jan 03, 06:59
e_{18}	A5636		Accept application	€200		Jan 03, 07:00
...	...	...	...	...	...	...
e_{22}	A5634		Finalise application			Jan 03, 09:00
e_{23}	A5636		Finalise application			Jan 03, 09:01
e_{24}	A5635		Decline application			Jan 03, 09:02
e_{25}	A5635		Decline application			Jan 03, 09:03
...	...	...	...	...	...	...
e_{30}	A5636	O3521	Select and send offer	€500		Jan 04, 16:32
...	...	...	...	...	...	...
e_{37}	A5634	O3541	Select and send offer	€1.500		Jan 05, 12:32
e_{38}	A5636	O3521	Receive offer		NO	Jan 05, 12:33
e_{38}	A5636	O3521	Cancel offer			Jan 05, 12:34
e_{39}	A5636	O3542	Select and send offer	€500		Jan 05, 13:29
e_{40}	A5636	O3542	Receive offer		YES	Jan 08, 08:33
e_{41}	A5636	O3542	Accept offer			Jan 08, 16:34
e_{42}	A5634	O3541	Receive offer		NO	Jan 10, 10:00
...	...	...	...	...	...	...
e_{54}	A5634	O3541	Decline offer			Jan 10, 10:04
...	...	...	...	...	...	...
e_{64}	A5634		Decline application			Jan 10, 10:05
e_{65}	A5634		Application finished			Jan 10, 10:06
e_{66}	A5636		Approve and activate application			Jan 10, 10:07
e_{67}	A5636		Application finished			Jan 10, 10:08
...	...	...	...	...	...	...



Precision: Has the modelled behaviour been recorded ?

Evaluation of Mined Process Models

Motivation

- Different process mining algorithms create different models
- How to judge quality of the created models?

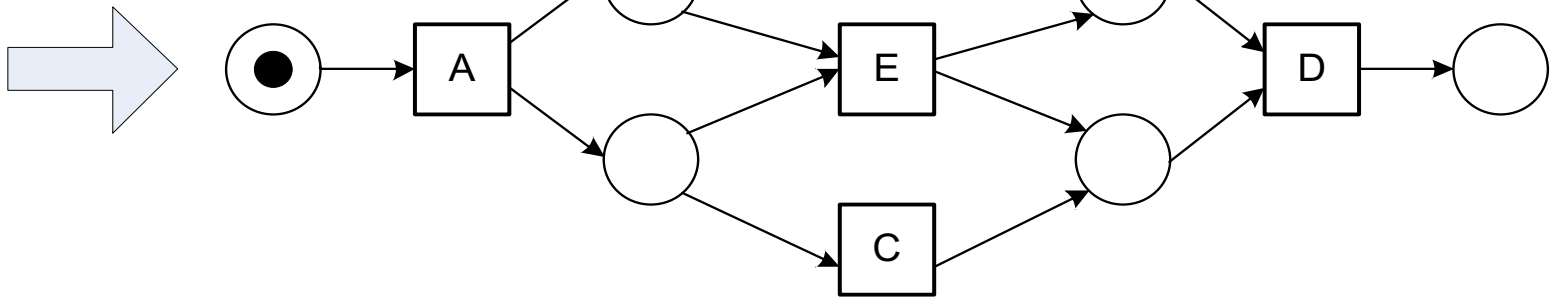
Evaluation of mined process models

- To compare process mining algorithms
- To make pros and cons of different approaches explicit

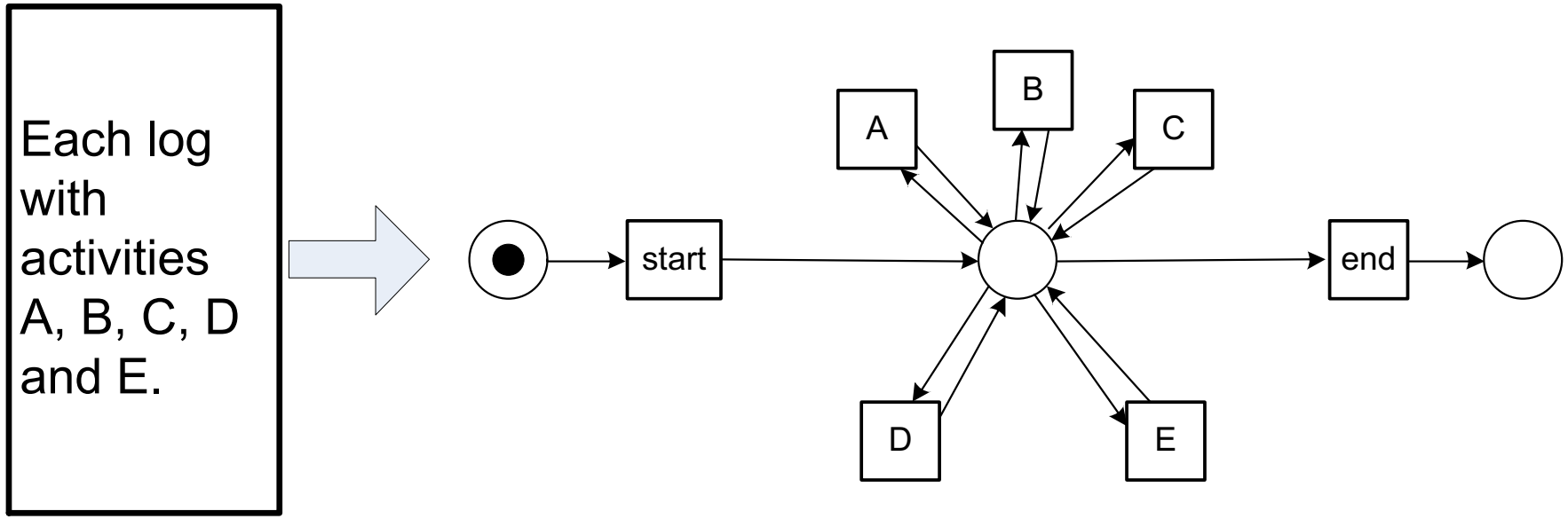
Evaluation in terms of overfitting/underfitting

Overfitting/Underfitting

ABCD
ACBD
AED
ABCD
ABCD
AED
ACBD
...



Overfitting/Underfitting cont.



Evaluation Dimensions

Fitness

- Is the observed behaviour covered by the model?

The Precision – Generalisation Trade-off

- Recall: in general, it is extremely unlikely that a data sample such as a log contains all (!) possible behaviour
  - Does the model allow for only the observed behaviour?
  - Does the model allow for more than the observed behaviour?

Simplicity

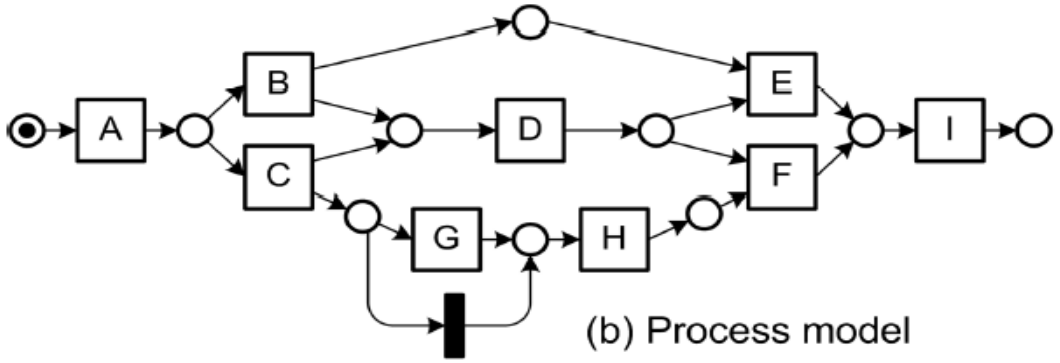
- Does the model have the minimal structure to represent the behaviour?

Example Process Model

- High fitness
- High precision

No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

(a) Event Log



(b) Process model

“Sequence“

- Low fitness
- High precision

No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

(a) Event Log

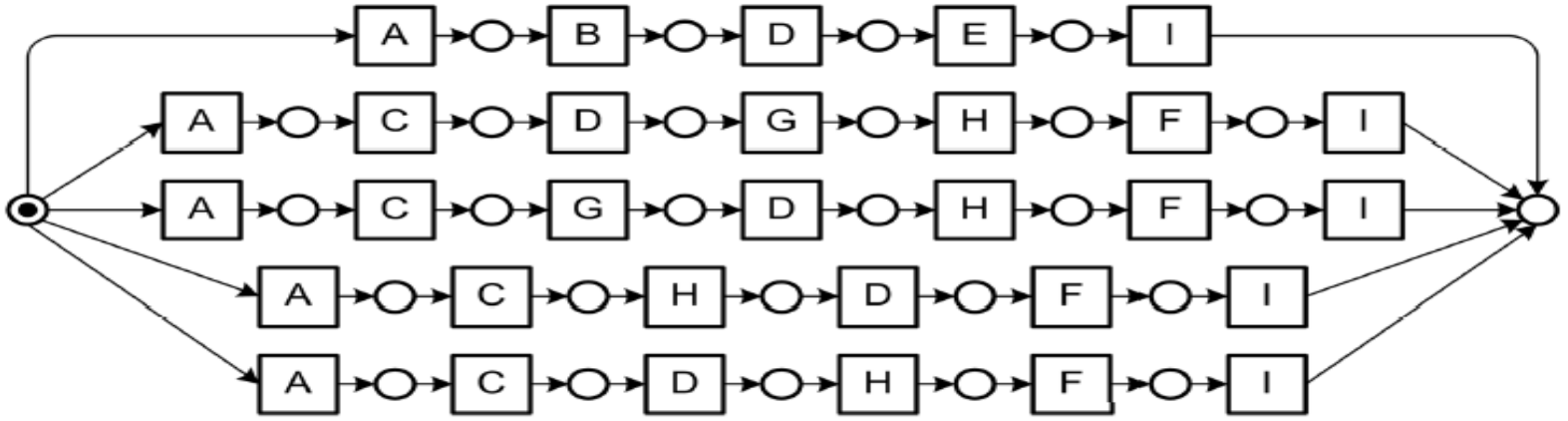
(c) Process model

“Log“

- High fitness
- High precision
- No generalisation
- Bad structure

No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

(a) Event Log



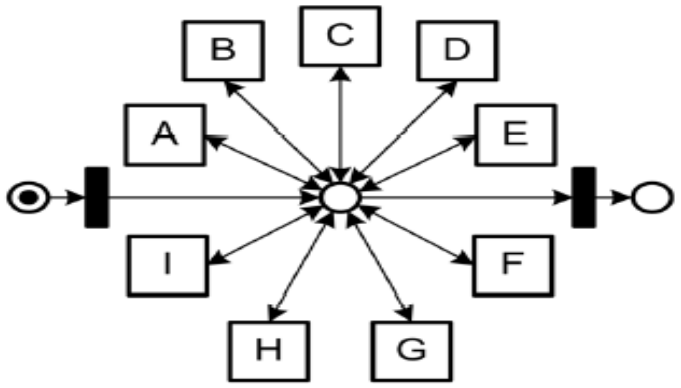
(e) Process model

“Flower“

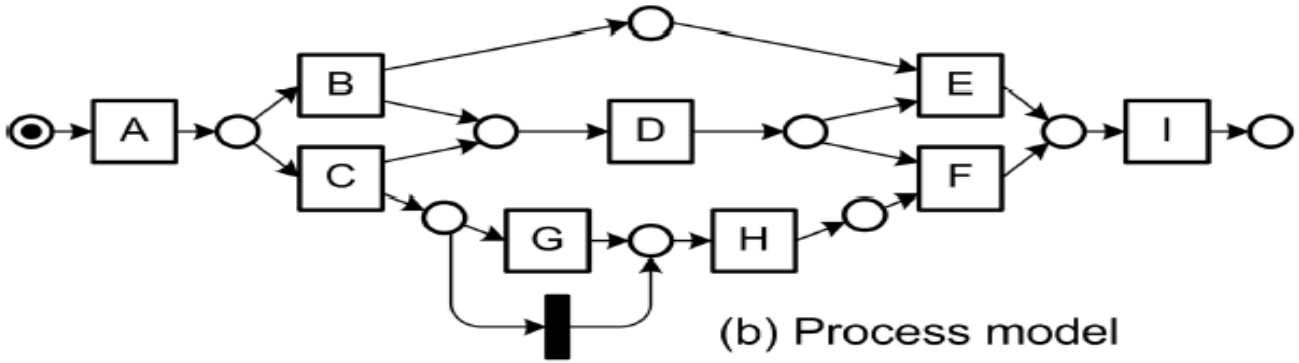
- High fitness
- Low precision

No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

(a) Event Log



(d) Process model



(b) Process model

(c) Process model

fitness +
precision +
generalization +
structure +

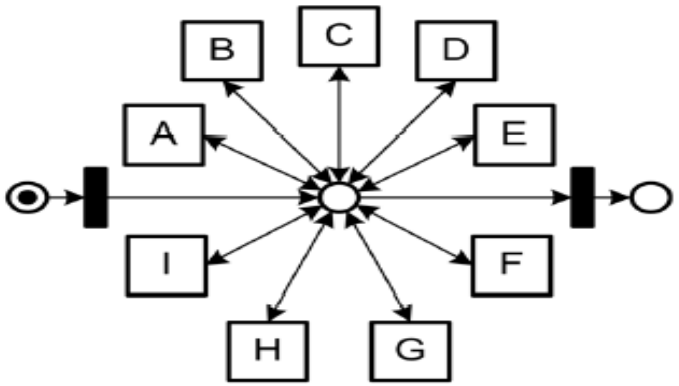
fitness +
precision -
generalization +
structure +

No. of Instances	Log Traces
1207	ABDEI
145	ACDGHFI
56	ACGDHFI
23	ACHDFI
28	ACDHFI

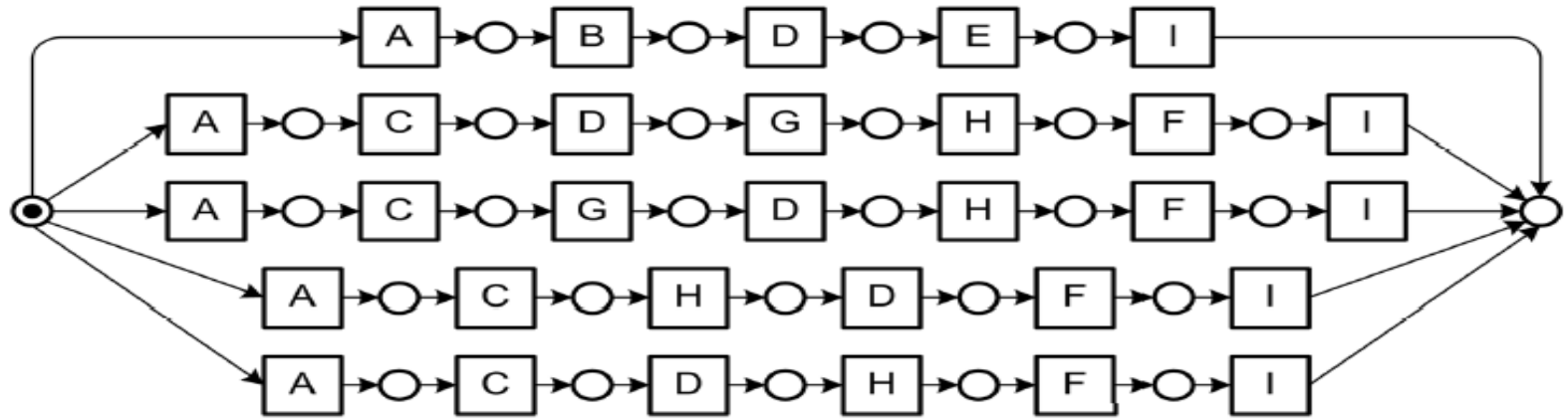
(a) Event Log

fitness -
precision +
generalization -
structure +

fitness +
precision +
generalization -
structure -



(d) Process model



(e) Process model

Fitness Measure

Given a process model and a process log

- How to quantify the behaviour represented by the traces w.r.t. the model?

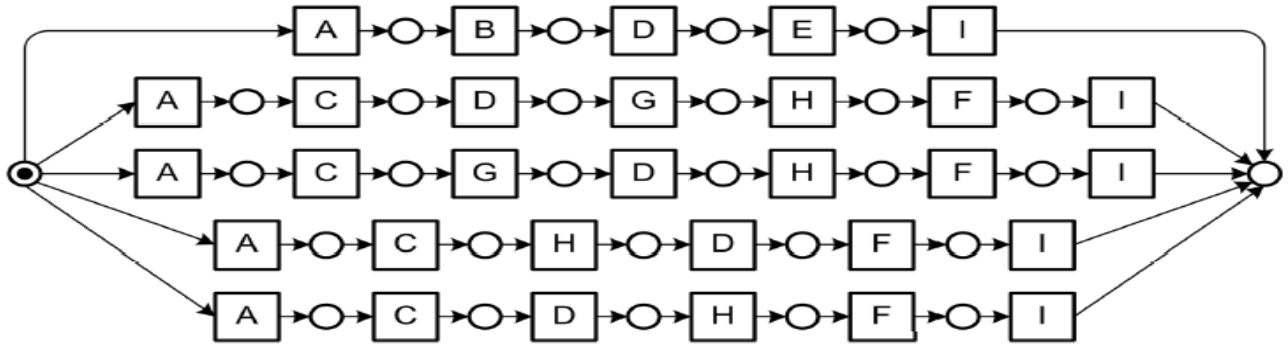
Idea

- Try to “replay” each trace in the model
- Try to fire transitions according to the trace
- Analyse the relation between log and model semantics
- Not ok:
  - Transition fires although it is not enabled
  - Tokens remain in the net and are not consumed

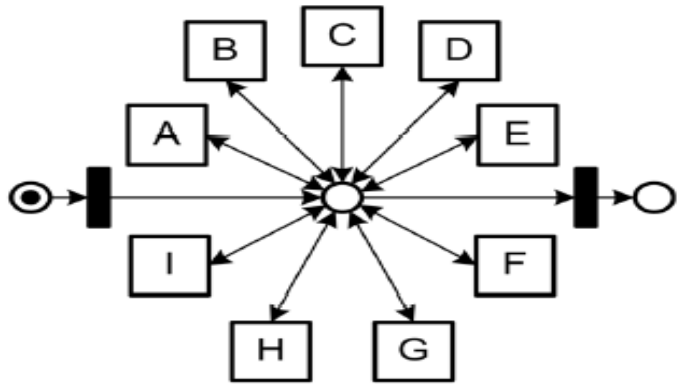
Precision Measures

Given a process model and a process log

- How to quantify the relation between *“observed behaviour / possible behaviour“?*
- How much of the behaviour of the model is observed in the log?
- Or, in terms of generalisation: how much behaviour of the model is not observed in the log?



(e) Process model



(d) Process model

Simplicity

Goal: find a *simple* model

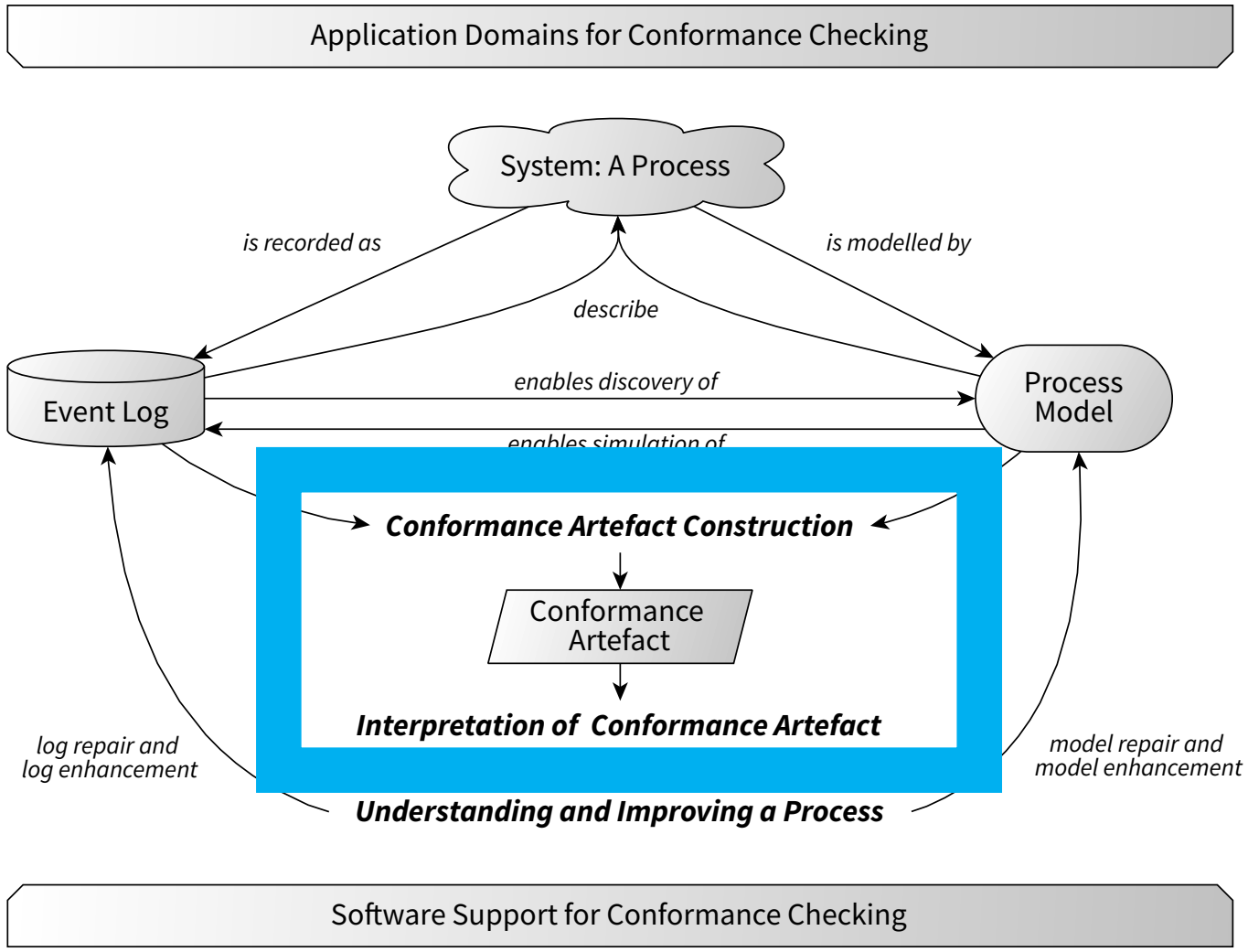
- … that meets the requirements in terms of fitness and precision/generalisation
- … to chose among a set of candidate models

Yet, different notions of simple are meaningful:

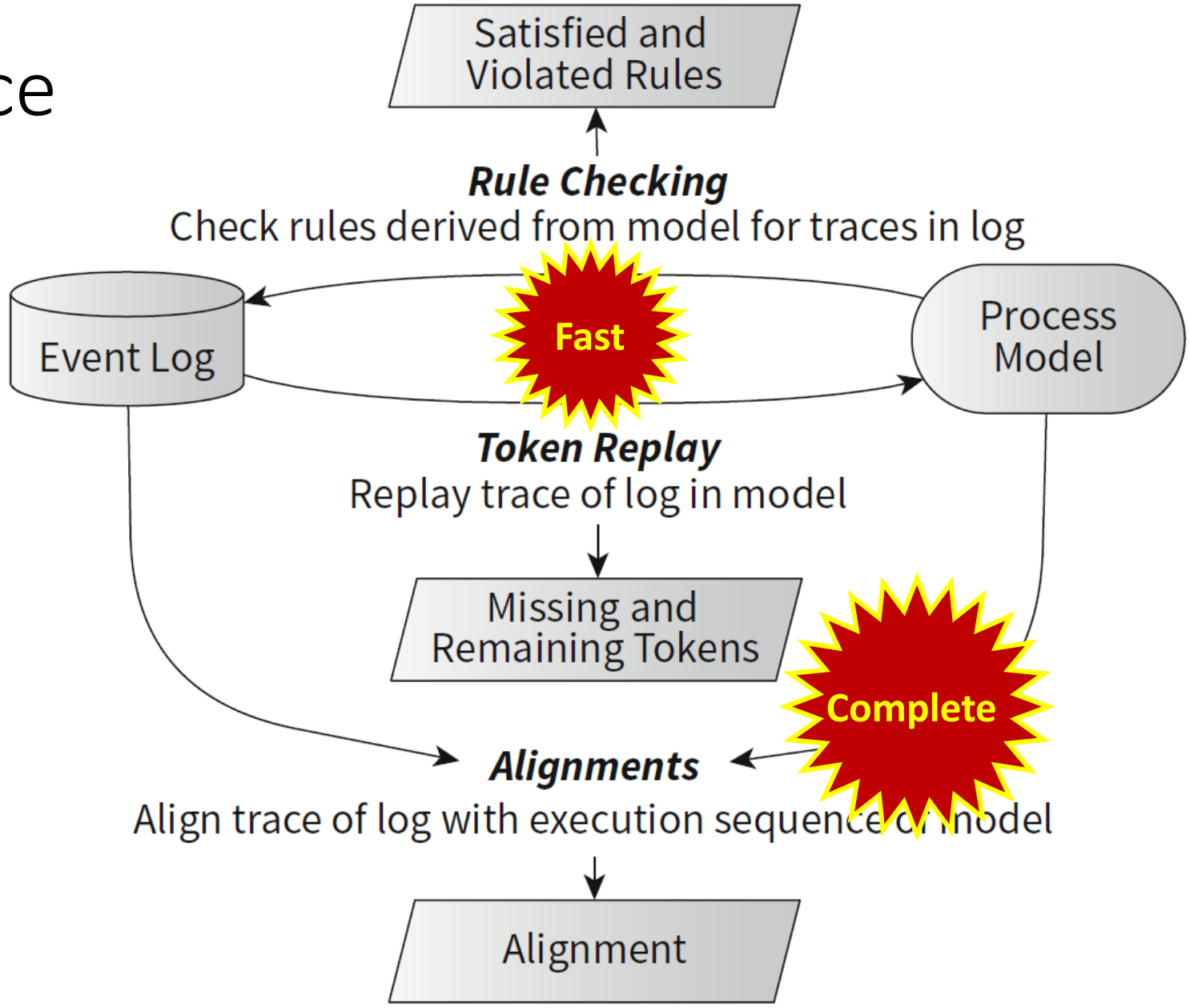
- Depends on modelling language
- Depends on understandability considerations
- Mostly related to model structure, yet may also concern semantics

Baseline measure: Simplicity as the number of model elements

The Spectrum of Conformance Checking



Computing Conformance Artefacts



Computing Conformance Checking Artefacts

Rule Checking

Idea of Rule Checking

General idea:

- Process model constrains the possible behaviour of a process
- Formalise constraints by rules that execution sequences have to obey to
- Then, verify these rules for the traces of a log

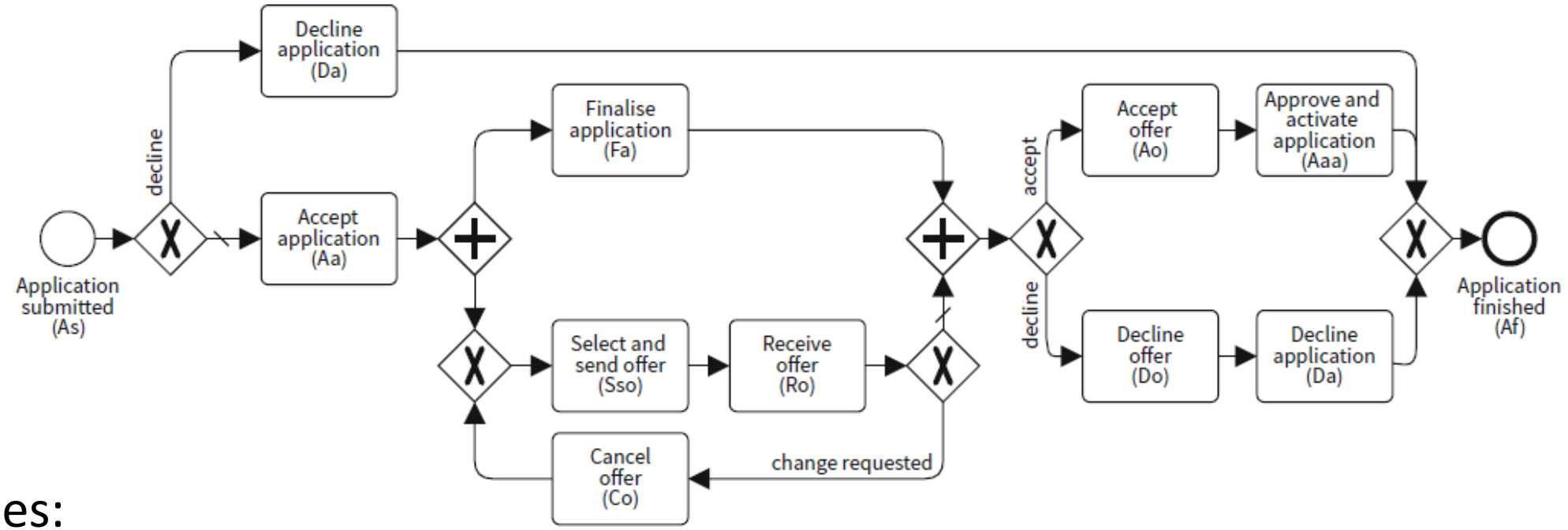
Rule checking focuses on the fitness dimension

- Traces are fitting, if they satisfy the rules
- Traces are non-fitting, if rules are violated

The dimension of precision is not targeted

- Specificity of rules is neglected
- Many more traces, not contained in the log, could also satisfy the rules

Illustration of Rules



Rules:

- R1: An application can be accepted (Aa) at most once
- R2: An accepted application (Aa), that must have been submitted (As) earlier, and eventually an offer needs to be selected and sent (Sso) for it
- R3: An application must never be finalised (Fa), if the respective offer has been declined (Do) already
- R4: An offer is either accepted (Ao) or declined (Do), but cannot be both accepted and declined

What rules?

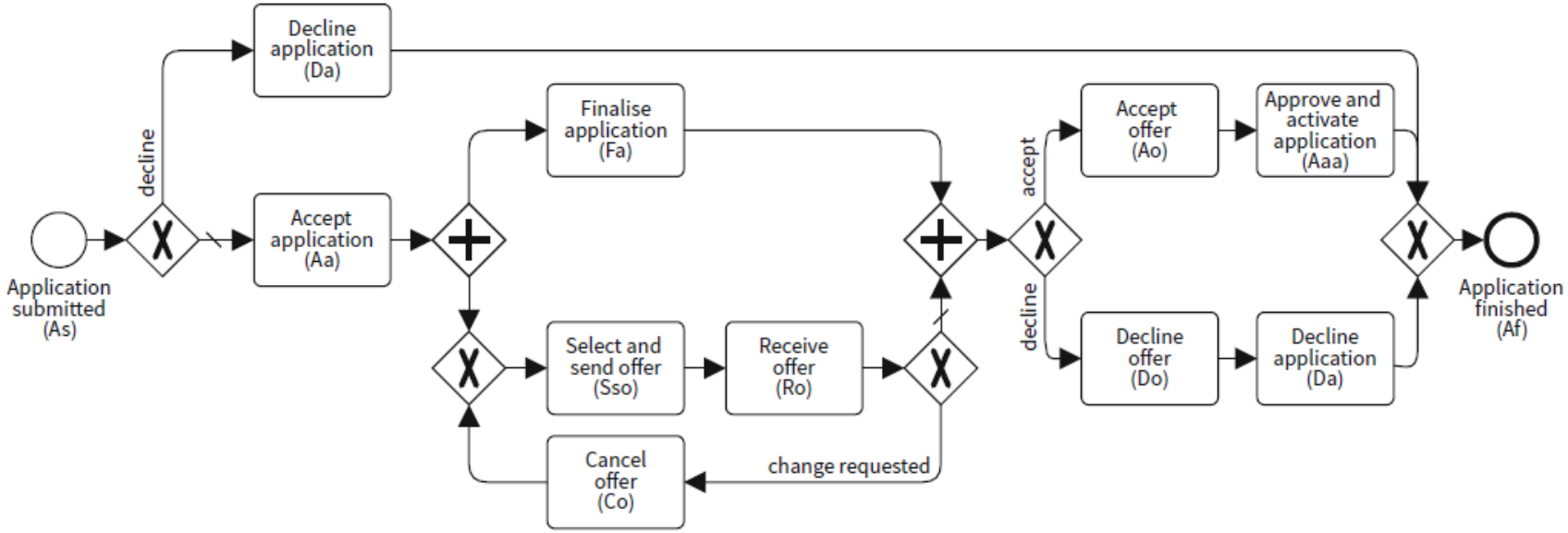
Often: unary and binary rules over activities

- Specify how a single activity is executed or how the executions of a pair of activities relate to each other
- N-ary rules would lead to combinatorial explosion

Common types of rules:

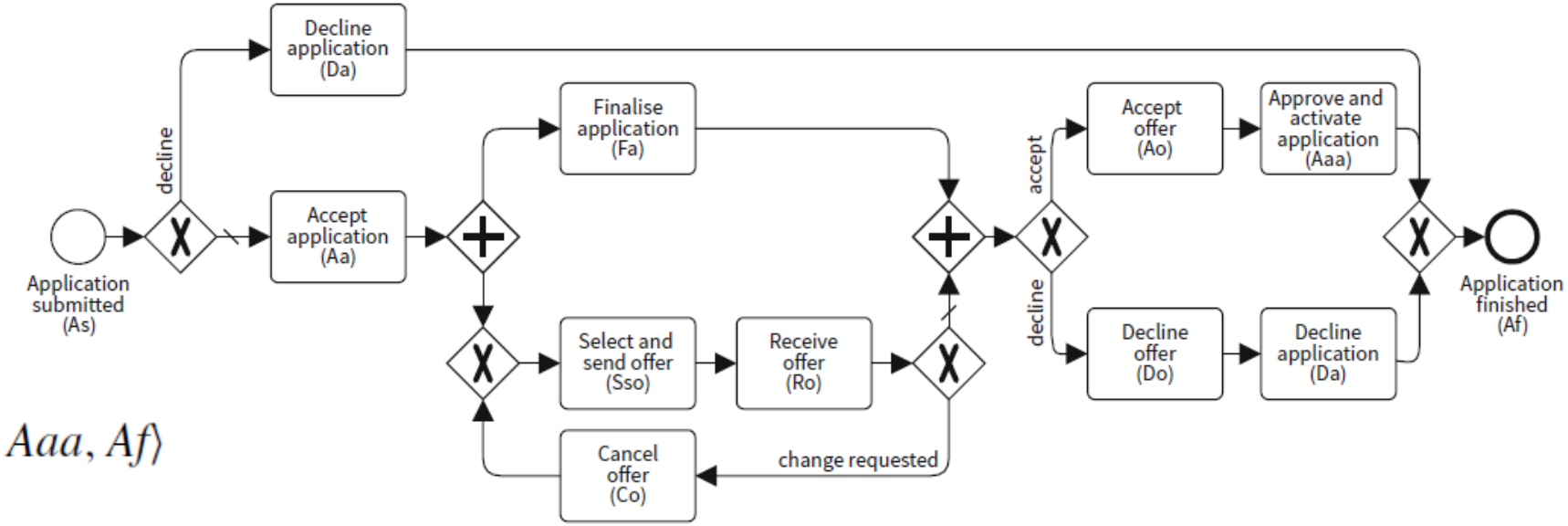
- Cardinality rules: Upper and/or lower bound for the number of executions of an activity
- Precedence and response rules: An activity is always preceded/succeeded by another one
- Ordering rules: Activities are independent, but if they are both executed, they are ordered
- Exclusiveness rules: Activities must not be executed in the same case

Cardinality Rules



Activity	*As*	*Da*	*Aa*	*Fa*	*Sso*	*Ro*	*Co*	*Ao*	*Aaa*	*Do*	*Af*
Cardinality:	**[1, 1]**	**[0, 1]**	**[0, 1]**	**[0, 1]**	**[0, *n*]**	**[0, *n*]**	**[0, *n*]**	**[0, 1]**	**[0, 1]**	**[0, 1]**	**[1, 1]**
$T_1 = \langle As,Aa,Sso,Ro,Ao,Aaa,Aaa \rangle$											
	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗
$T_2 = \langle As,Sso,Fa,Ro,Co,Ro,Aaa,Af \rangle$											
	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$T_3 = \langle As,Aa,Sso,Ro,Fa,Ao,Do,Da,Af \rangle$											
	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

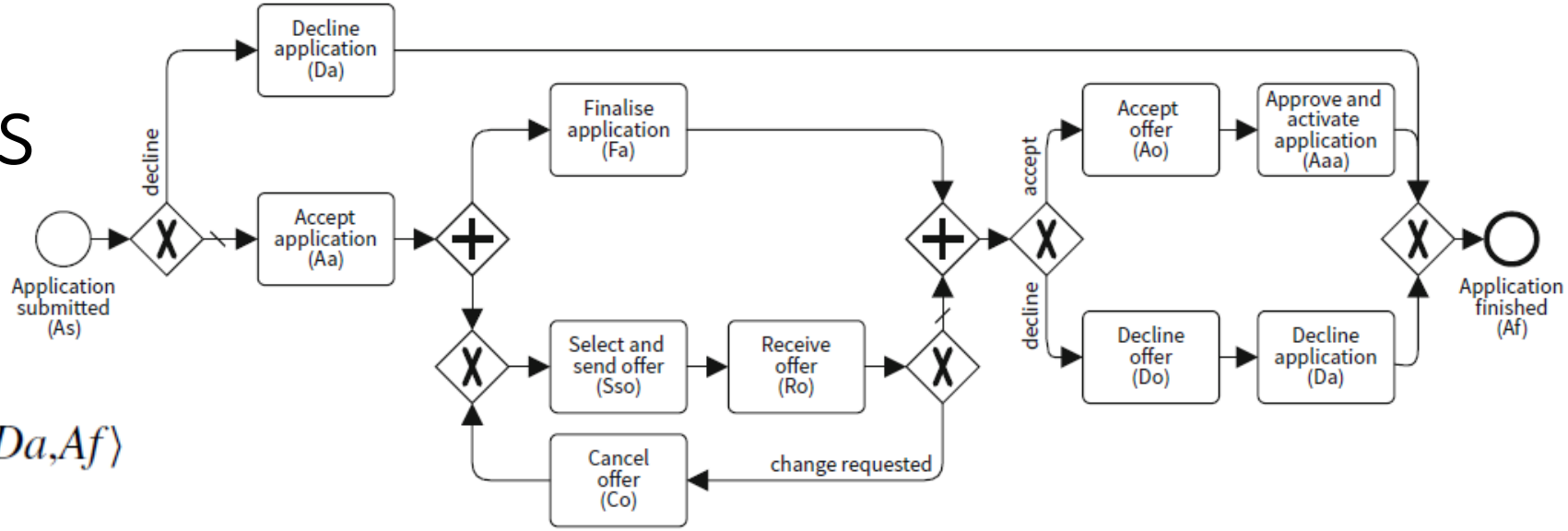
Precedence Rules



$T_2 = \langle As, Sso, Fa, Ro, Co, Ro, Aaa, Af \rangle$

	As	*Da*	*Aa*	*Fa*	*Sso*	*Ro*	*Co*	*Ao*	*Aaa*	*Do*	*Af*
As											
Da	✓										
Aa	✓										
Fa	✓		✗								
Sso	✓		✗								
Ro	✓		✗		✓						
Co	✓		✗		✓	✓					
Ao	✓		✓	✓	✓	✓					
Aaa	✓		✗	✓	✓	✓		✗			
Do	✓		✓	✓	✓	✓					
Af	✓										

Exclusiveness Rules



$T_3 = \langle As, Aa, Sso, Ro, Fa, Ao, Do, Da, Af \rangle$

	As	*Da*	*Aa*	*Fa*	*Sso*	*Ro*	*Co*	*Ao*	*Aaa*	*Do*	*Af*
As	✓										
Da		✓						✗	✓		
Aa			✓								
Fa				✓							
Sso											
Ro											
Co											
Ao		✗						✓		✗	
Aaa		✓							✓	✓	
Do								✗	✓	✓	
Af											✓

Feedback on Non-Conformance

Rule-checking gives information on fitness of trace

- Is the model well-suited to explain the recorded trace?
- Yes, if rules are satisfied; no, if rules are violated
- Fine-granular feedback on the level of activities

Aggregated feedback on the level of a log

- Rules that are frequently violated point to general conformance issues
- Indicator for hot-spots
- Association rule mining to highlight dependencies between two violations v and v’: Ratio of number of traces showing both violations and number of traces showing only violation v

Fitness Measure

Counting of satisfied rules enables quantification of fitness:

$$\text{fitness}(\sigma, M) = \frac{|\{r \in R_M \mid r \text{ is satisfied by } \sigma\}|}{|R_M|}$$

$$\text{fitness}(L, M) = \frac{|\{r \in R_M \mid r \text{ is satisfied by all } \sigma \in L\}|}{|R_M|}$$

Obviously, the above measures depend on the chosen set of rules

- Values under different rule sets cannot be compared
- Measures are influence by how many rules are affected by a single conformance issue (e.g., missing execution of an activity)

Computing Conformance Checking Artefacts

Token Replay

Idea of Token Replay

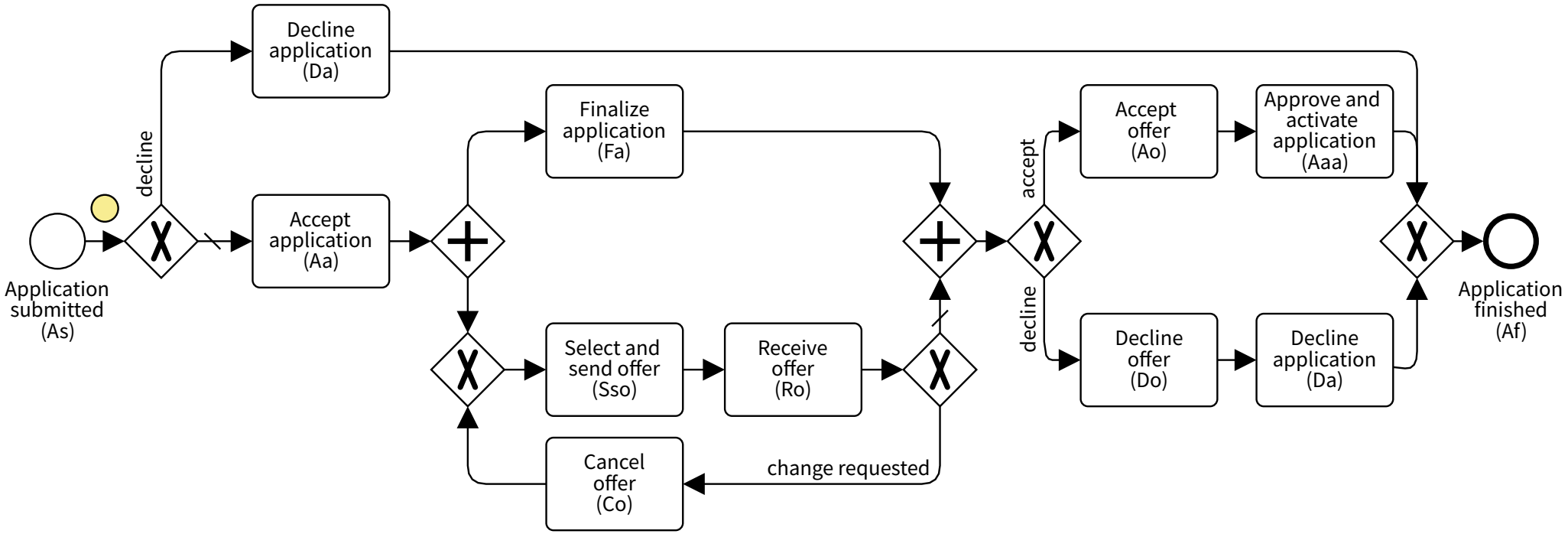
Idea

- Try to “replay” each trace in the model by executing activities according to the trace
- Not ok:
  - Activity is executed although it is not enabled
  - Tokens remain in the model and are not consumed

Procedure:

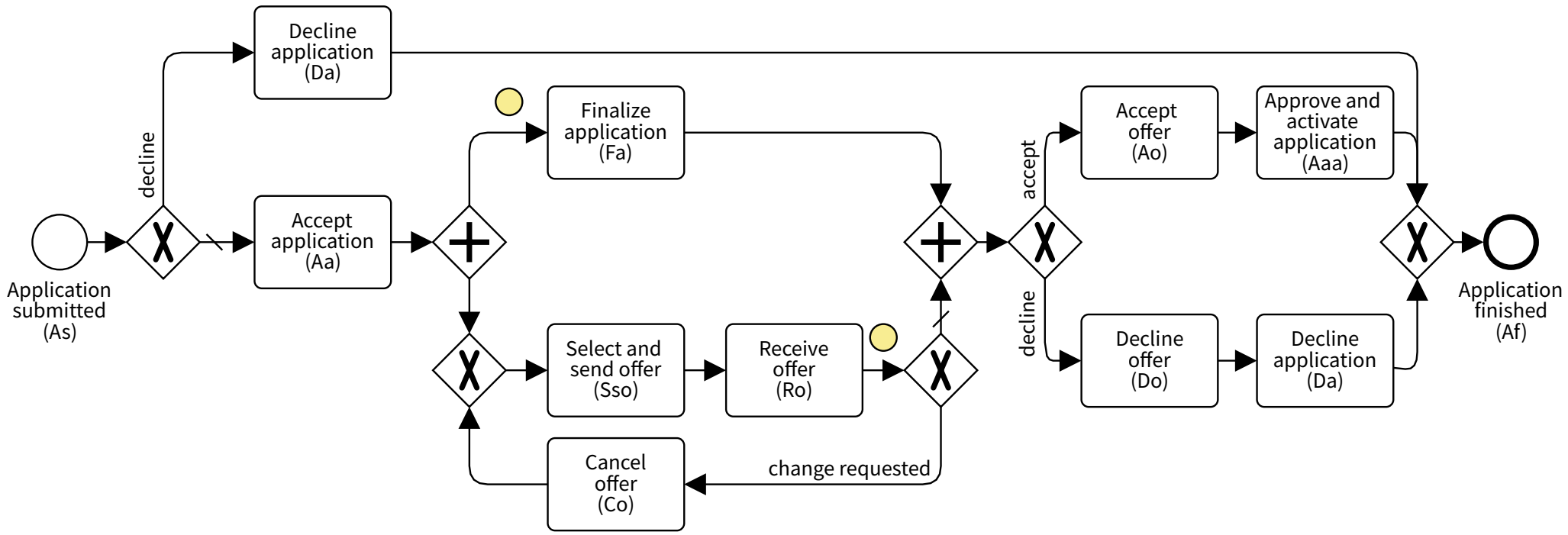
1) Initialise the replay procedure by setting the *first event* of the trace as the *current event* of the replay
2) Locate the activity for which execution is signalled by the *current event* of the trace in the model
3) Replay the *current event* of the trace by executing the identified activity in the *current state* of the model by:
   - Consuming a token on the incoming arc of the activity, even if there is none (if so, record as missing)
   - Producing a token on the outgoing arc of the respective task (if there is any)
4) Set the state reached as the *current state* of the process model
5) If the current event is not the last in the trace, consider the next event in the trace as the current event and continue the replay from step 2 onwards

Token-Based Replay



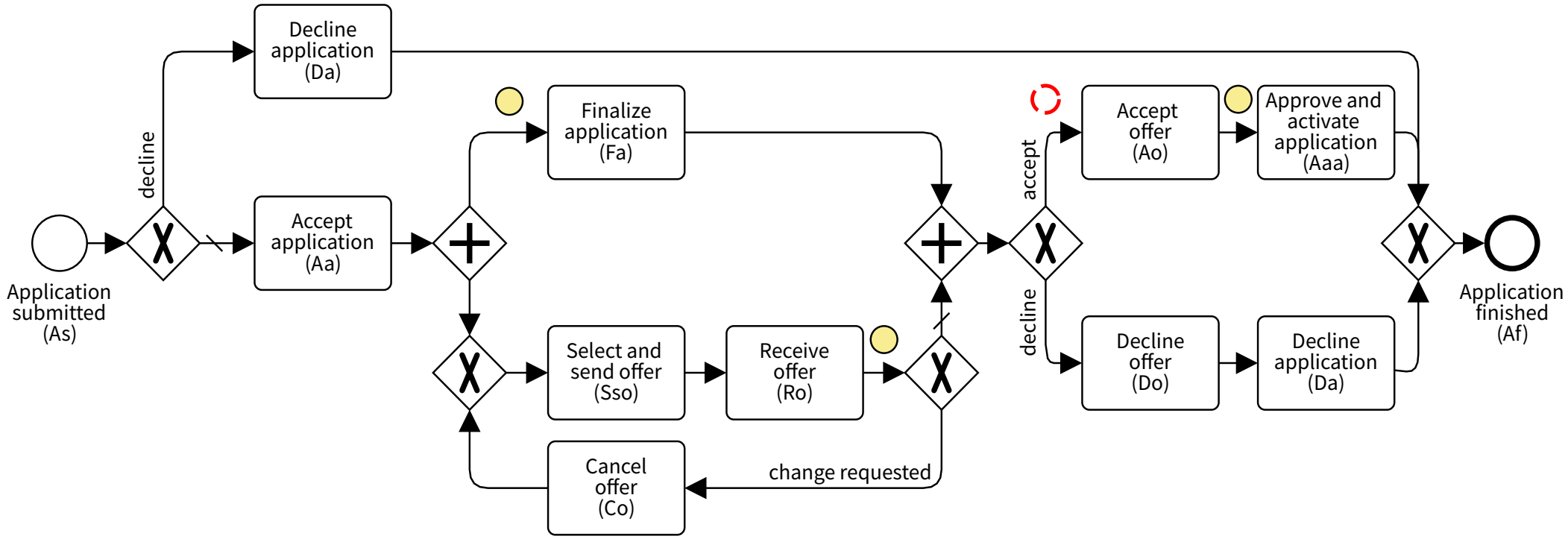
Observed Trace: < As, Aa, Sso, Ro, Ao, Aaa, Aaa >

Token-Based Replay



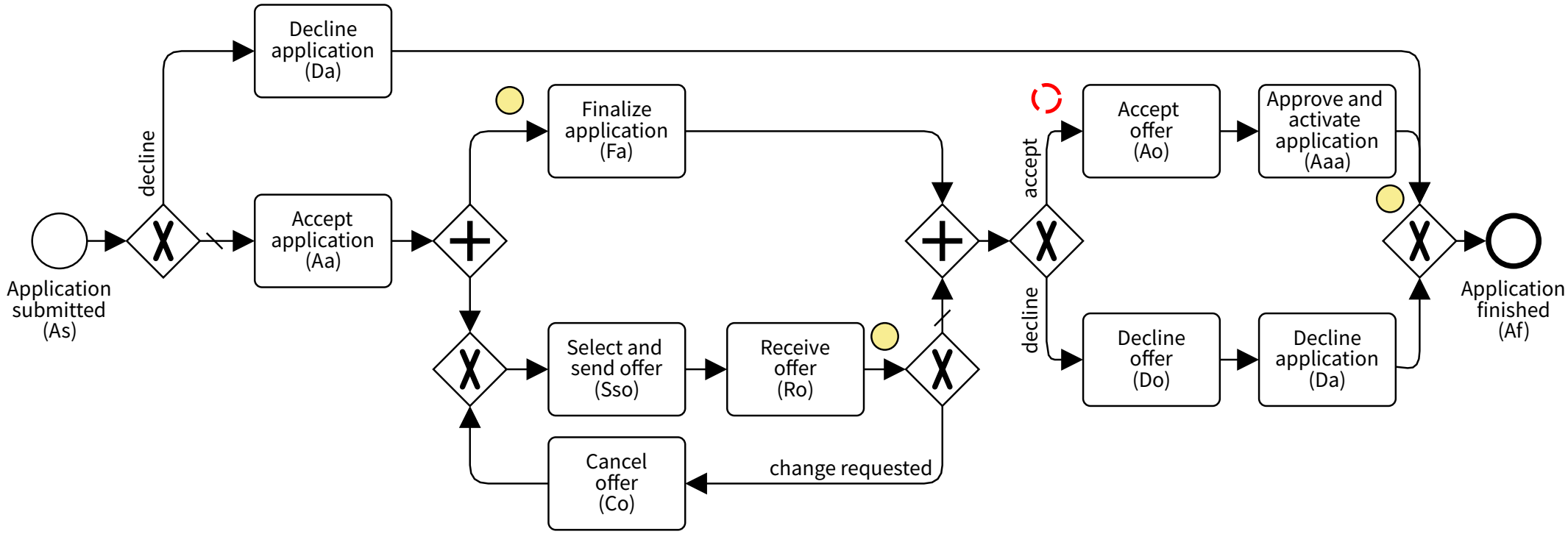
Observed Trace: < As, Aa, Sso, Ro, Ao, Aaa, Aaa >

Token-Based Replay



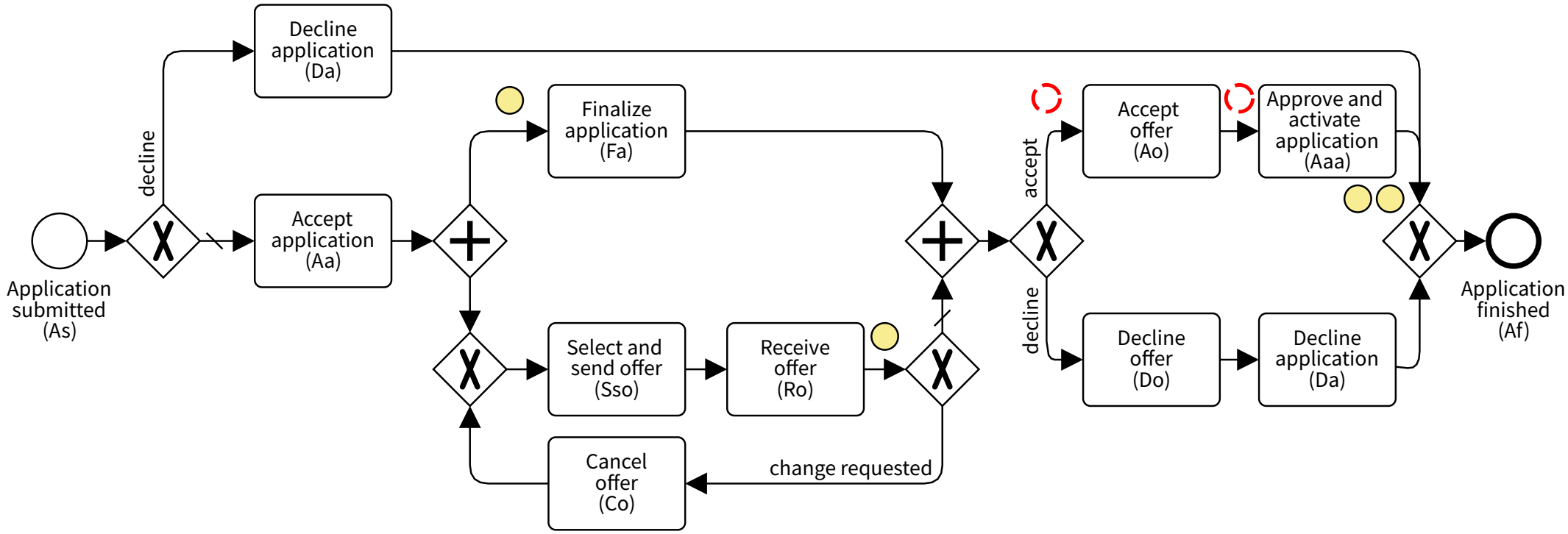
Observed Trace: < As, Aa, Sso, Ro, Ao, Aaa, Aaa >

Token-Based Replay



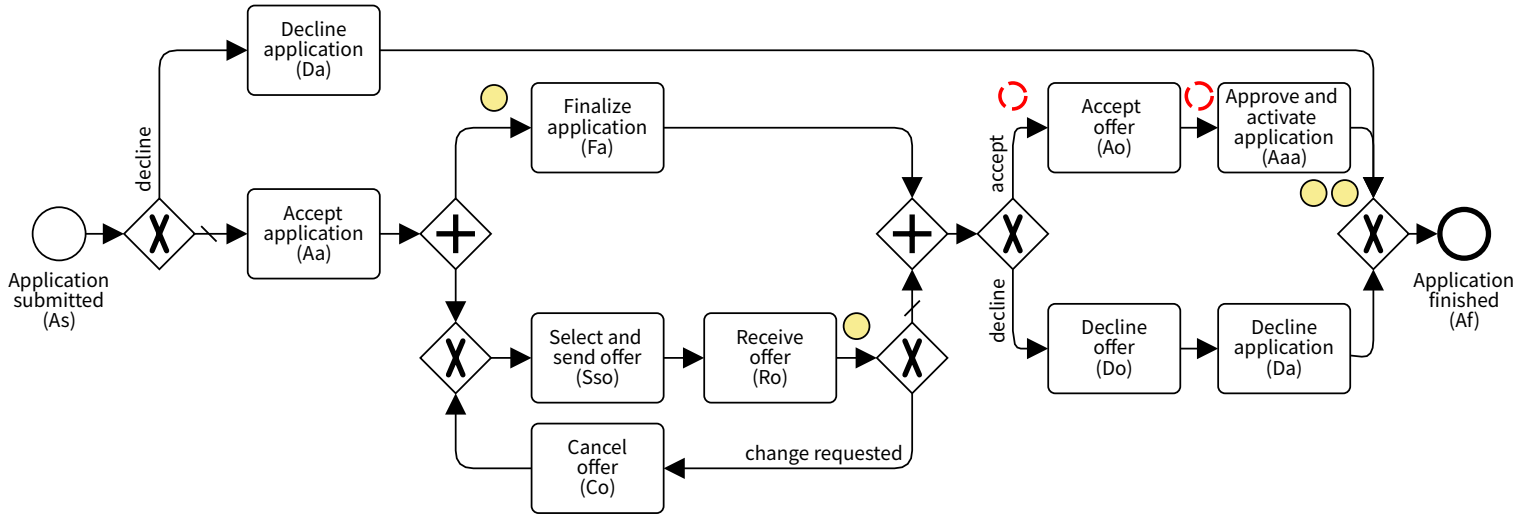
Observed Trace: < As, Aa, Sso, Ro, Ao, Aaa, Aaa >

Token-Based Replay



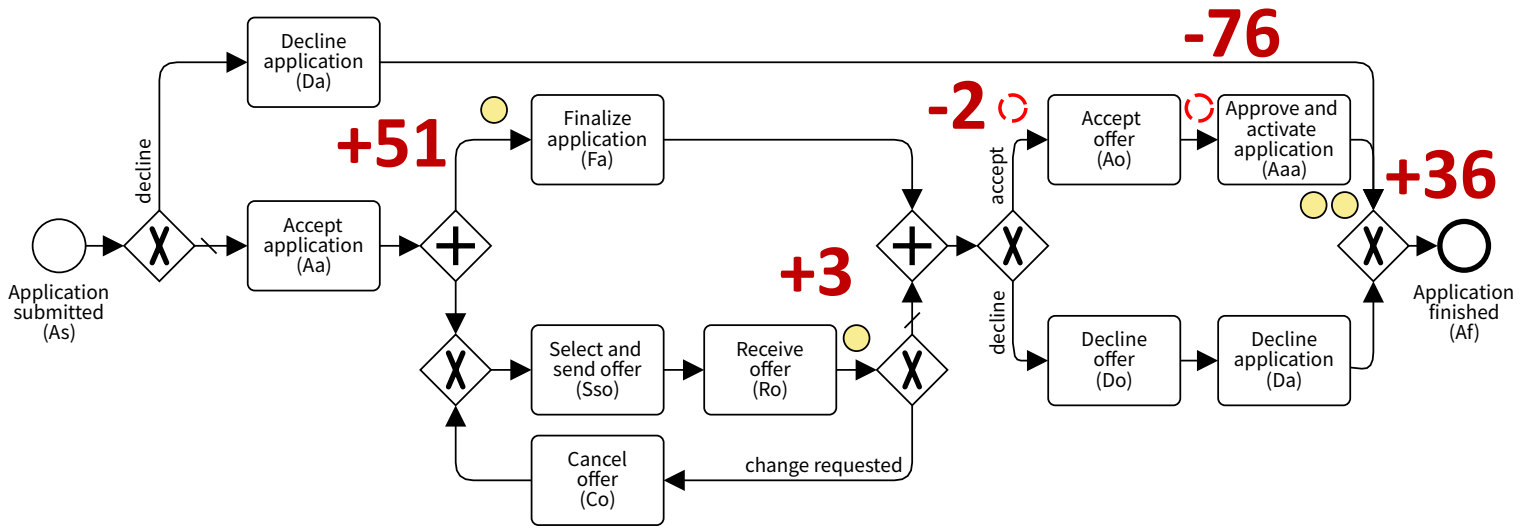
Observed Trace: < As, Aa, Sso, Ro, Ao, Aaa, Aaa >

Local Feedback on Non-Conformance



Deviations ≈ Tokens Missing or Tokens Remaining after replaying a trace

Aggregated Feedback on Non-Conformance



- Aggregated information about missing and remaining tokens
- Identification of “hotspots” of non-conformance
- Highlights major issues when replaying log and separates some from noise

Fitness Measure

Determine ratios based on missing and remaining tokens:

$$\text{fitness}(\sigma, M) = \frac{1}{2}\left(1 - \frac{missing(\sigma, M)}{consumed(\sigma, M)}\right) + \frac{1}{2}\left(1 - \frac{remaining(\sigma, M)}{produced(\sigma, M)}\right)$$

$$\text{fitness}(L, M) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} missing(\sigma, M)}{\sum_{\sigma \in L} consumed(\sigma, M)}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} remaining(\sigma, M)}{\sum_{\sigma \in L} produced(\sigma, M)}\right)$$

Overall conformance assessment based on aggregated fitness measure

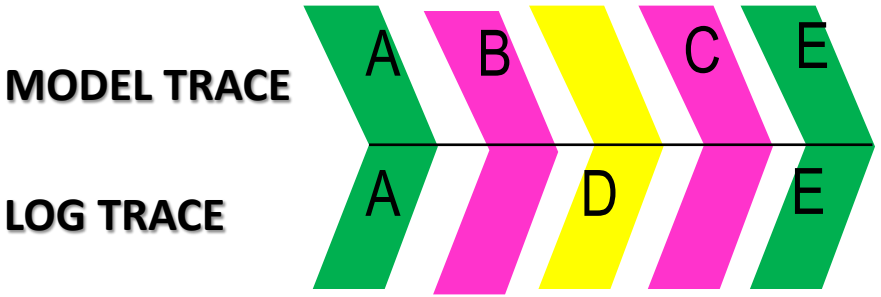
Condenses (non-)conformance into a single value

Yet, hard to interpret in terms of the absolute value

Computing Conformance Checking Artefacts

Alignments

Conformance Artefact: Alignments



SYNCHRONOUS MOVE: LOG AND MODEL AGREE

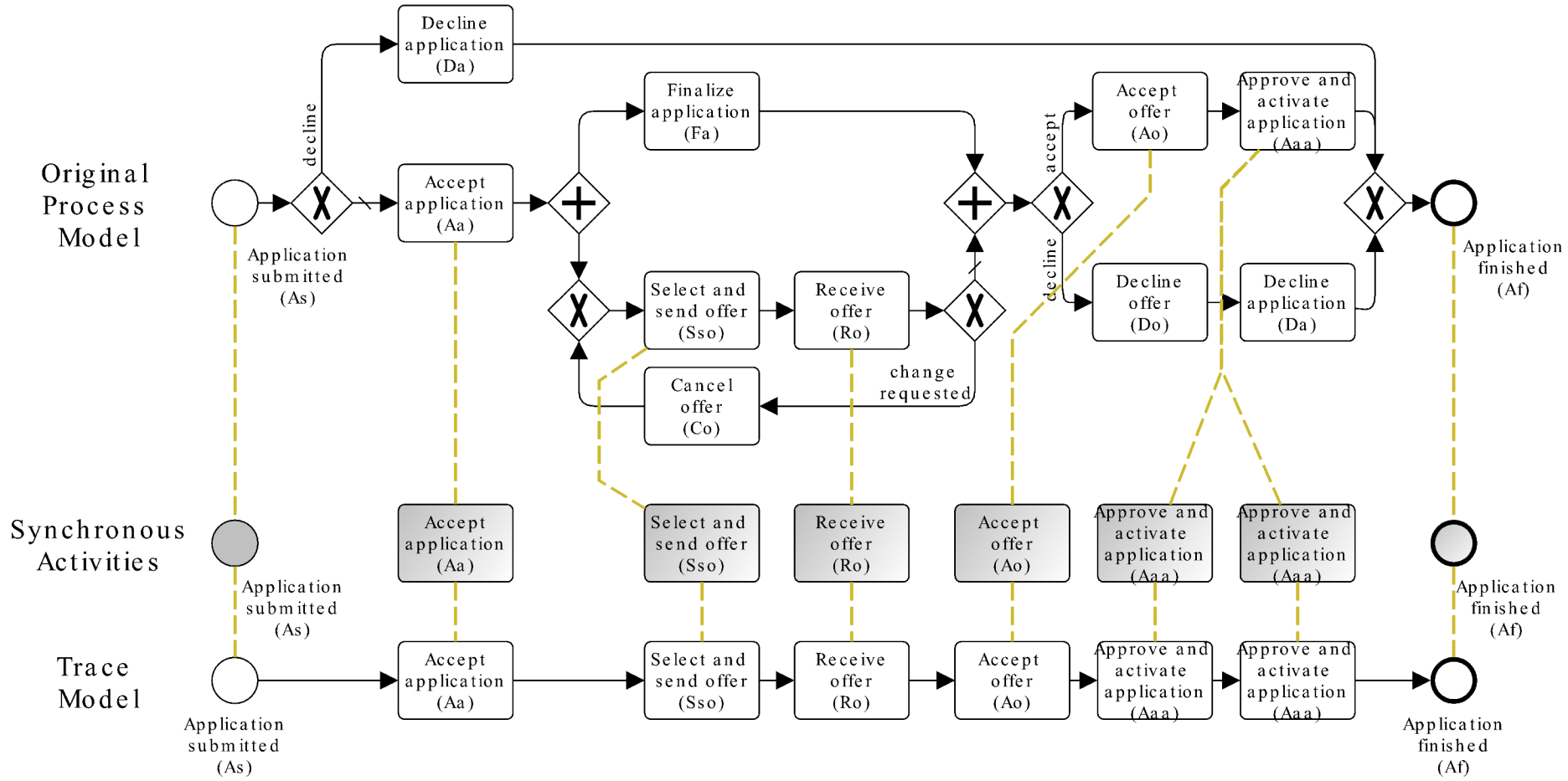
MODEL MOVE: TASKS NOT RECORDED THAT SHOULD BE EXECUTED

LOG MOVE: EVENTS RECORDED THAT CANNOT BE EXECUTED

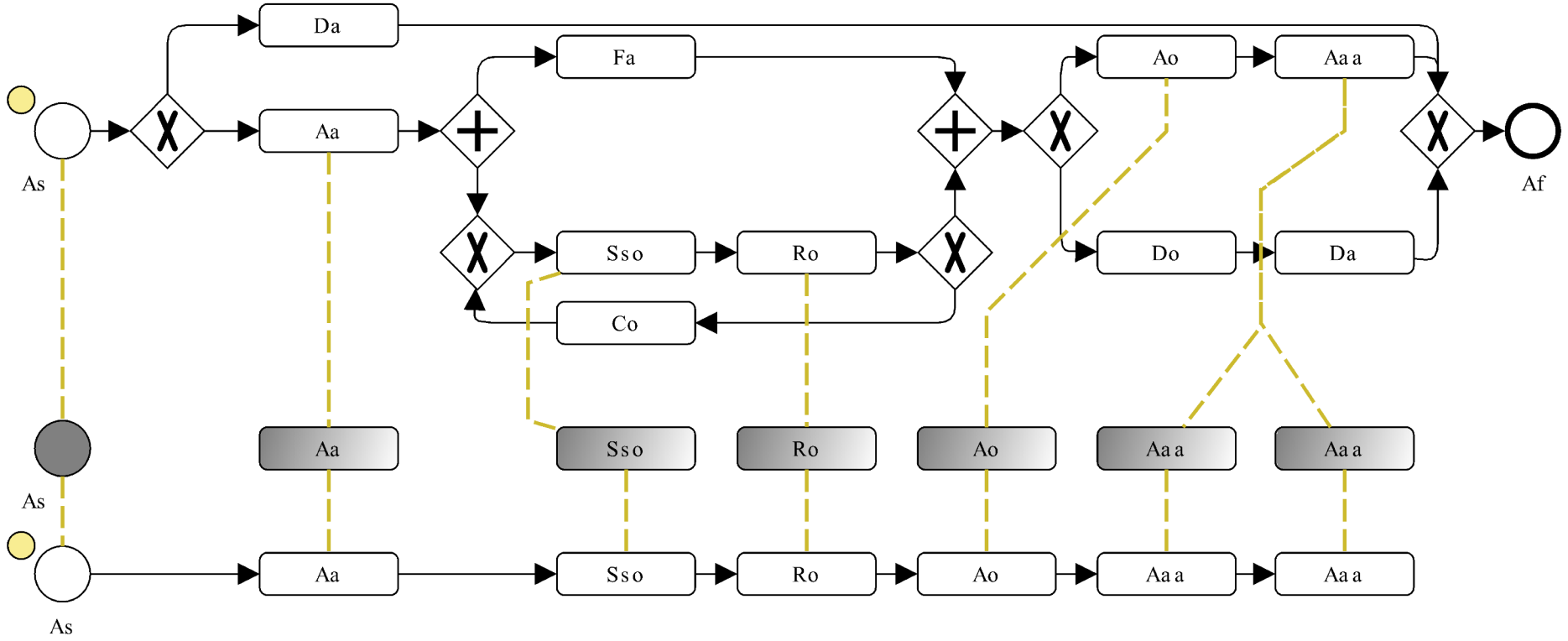
Computing Alignments through A*

- Reference technique for computing alignments
- Can guarantee optimality for an arbitrary cost function on deviations
- Can be adjusted to find not just one but all optimal alignments
- Implemented in ProM, with many optimizations.
- Based on the notion of *Synchronous Product Net*

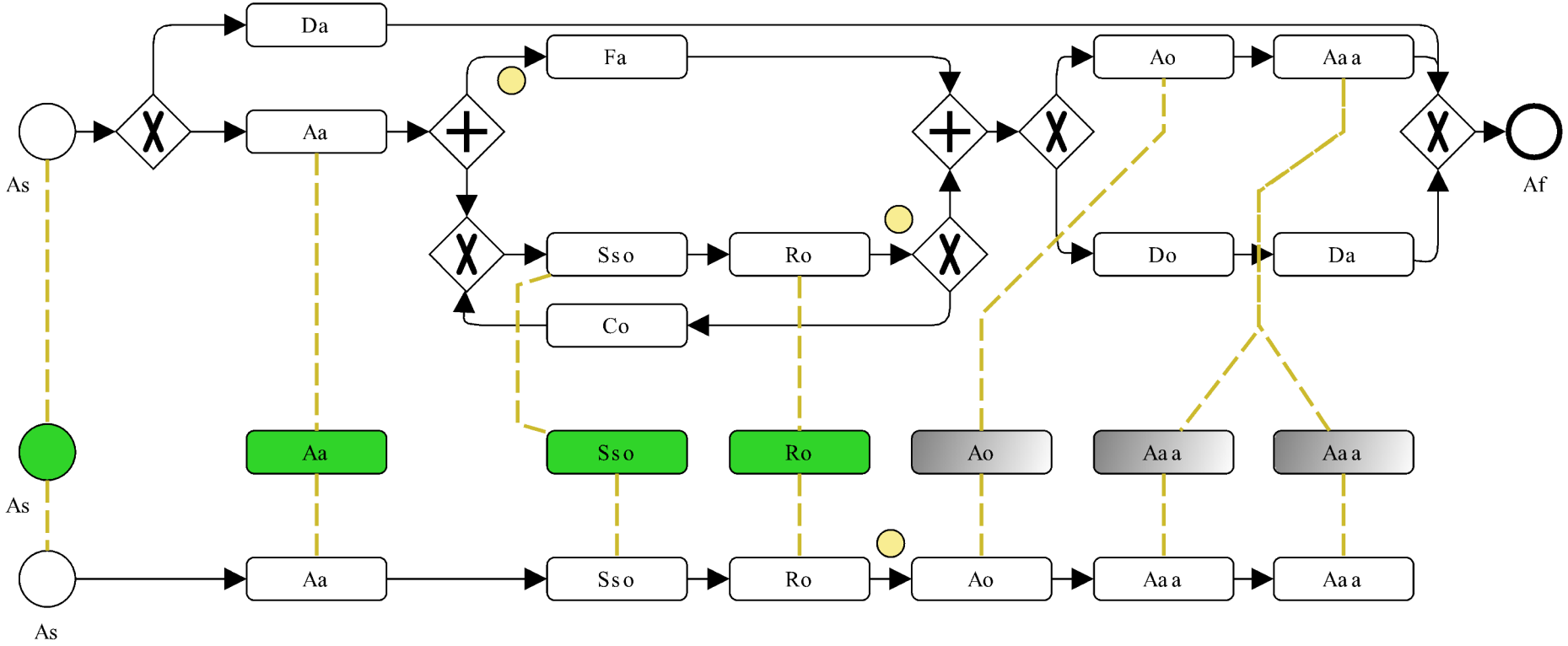
Synchronous Product Net (SPN)



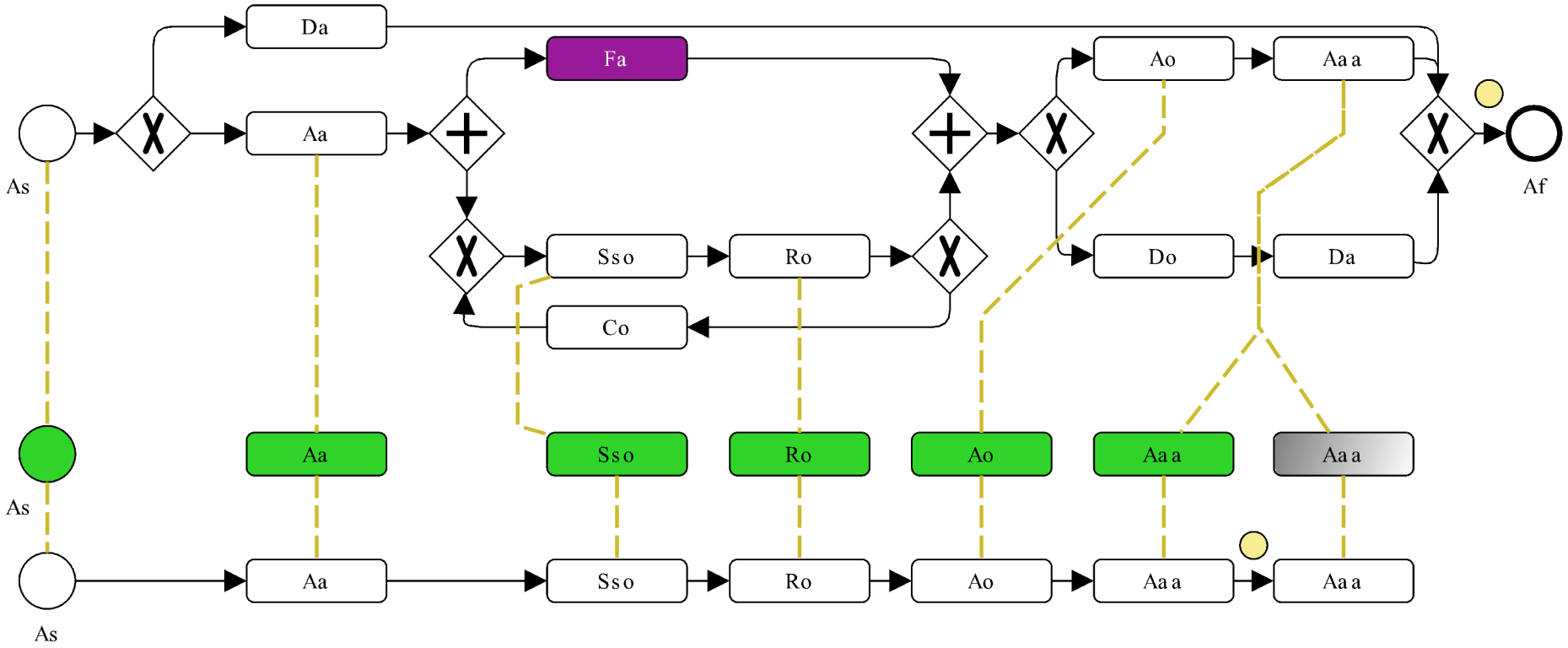
A run of the SPN



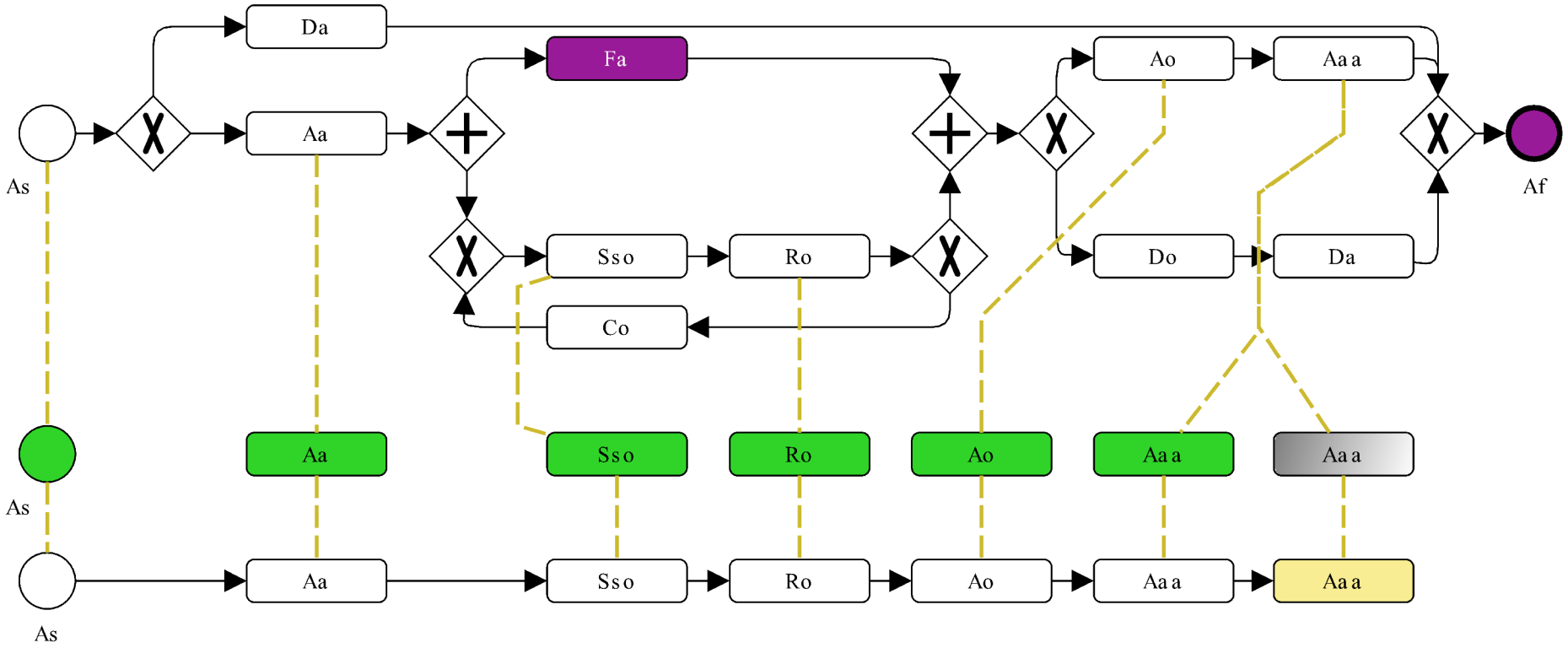
A run of the SPN



A run of the SPN



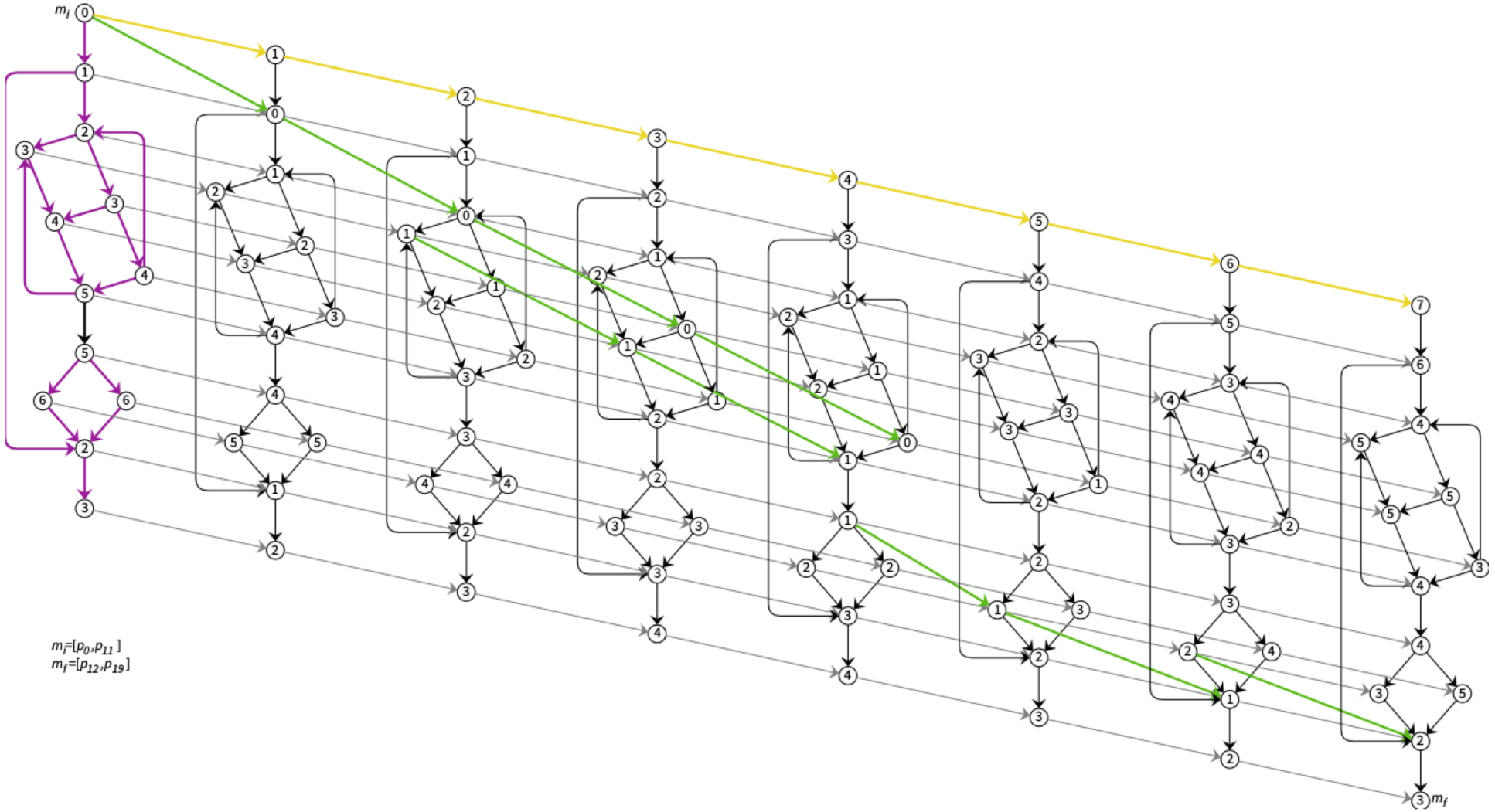
A run of the SPN



Algorithmics Behind the A*

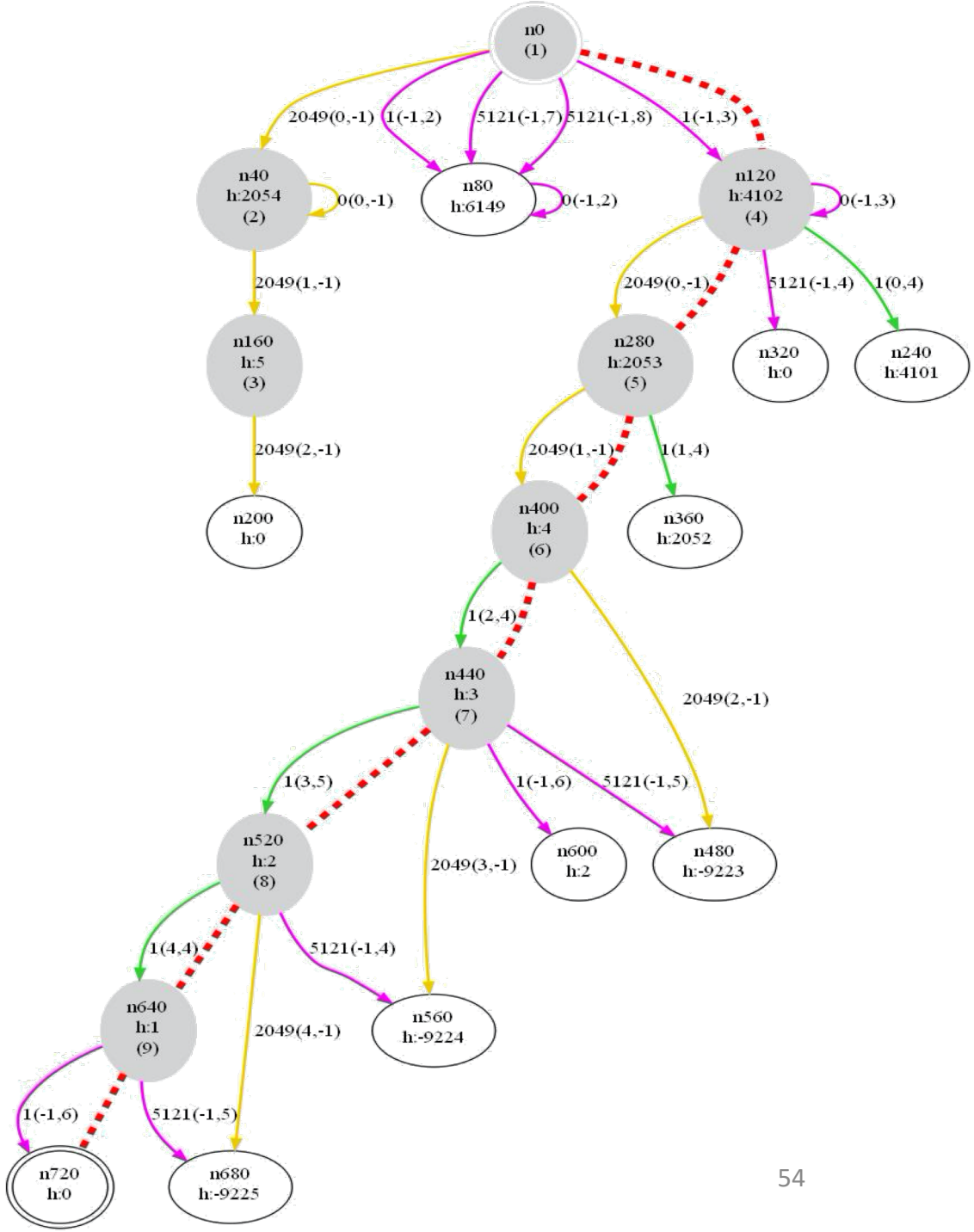
- In general, the SPN has an infinite set of runs !
- Instead: Traverse the SPN **state space** in a efficient way so that a cost-minimal path from the initial to the final state (representing the optimal alignment) is found.
- Using an estimation of the distance to the final state, prune parts of the state space.

SPN State Space



Computing Alignments

- The search space is the statespace of the synchronous product model
- 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* with fast lowerbound)

```
Initialize HashBackedPriorityQueue 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)

?

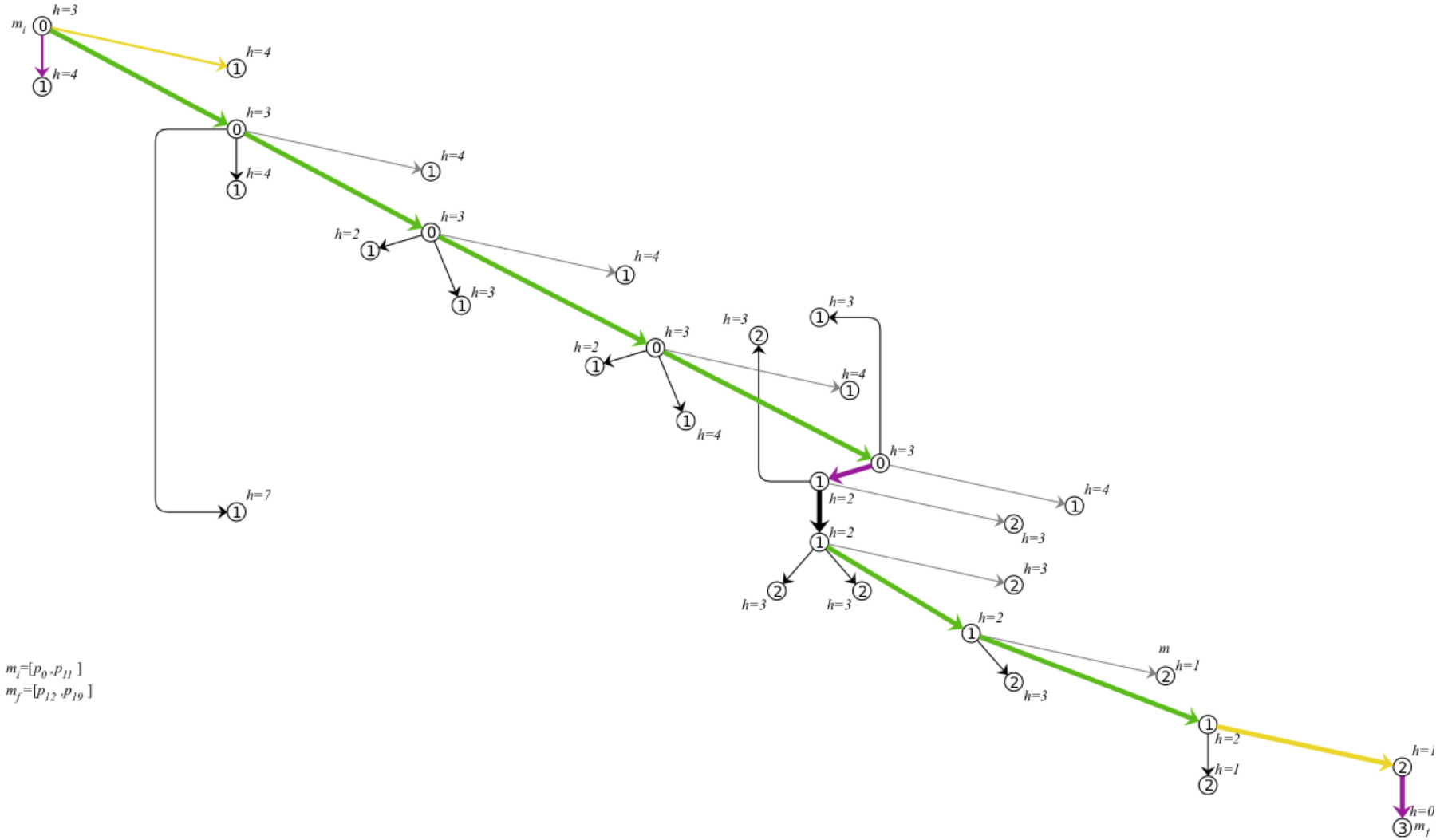
O(log(size q)+ α)

O(log(size q)+ α)

O(1)

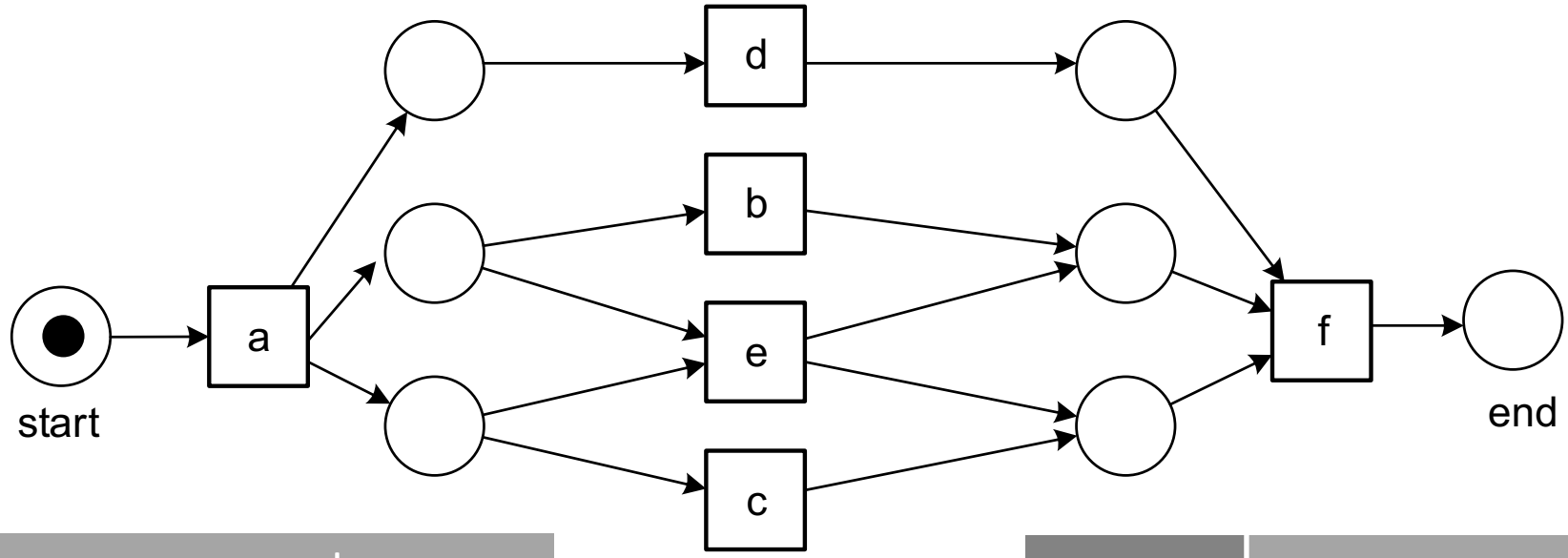
O(log(size q)+ α)

SPN State Space after A* Prunning



Alignments

- Find the optimal alignment(s).



trace	a	e	c	d	>>
model	a	e	>>	d	f

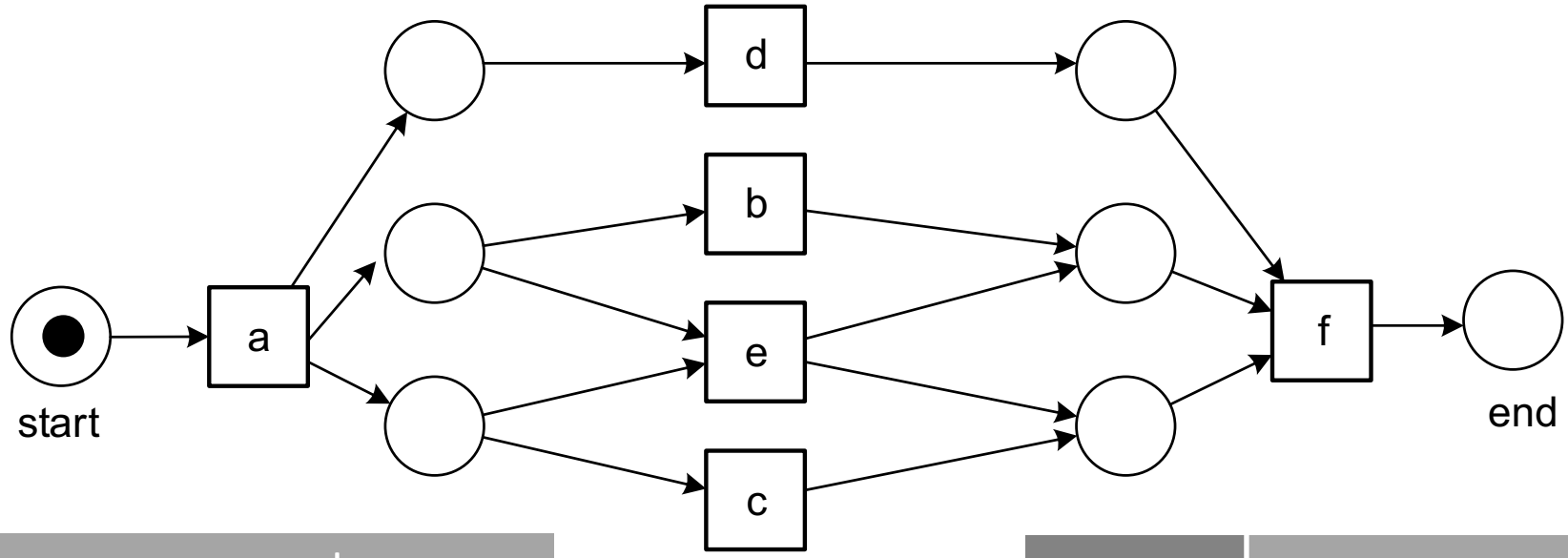
trace	a	e	c	d	>>	>>
model	a	>>	c	d	b	f

trace	a	e	c	d
model	a	>>	c	d

trace	a	e	c	d	>>	>>
model	a	e	>>	>>	d	f

Alignments - solution

- Find the optimal alignment(s).



Incomplete!

Cost 6

trace	a	e	c	d	>>
model	a	e	>>	d	f

Cost 1

trace	a	e	c	d
model	a	>>	c	d

Cost 3

trace	a	e	c	d	>>	>>
model	a	>>	c	d	b	f

Cost 12

trace	a	e	c	d	>>	>>
model	a	e	>>	>>	d	f

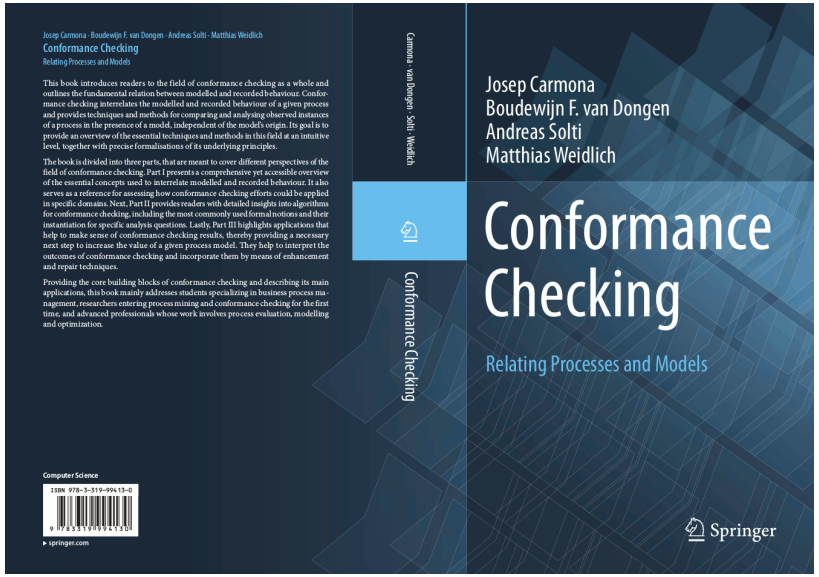
"Conformance Checking: Relating Processes and Models"

Josep Carmona

Boudewijn van Dongen

Andreas Solti

Matthias Weidlich



https://www.springer.com/gb/book/9783319994130

Springer