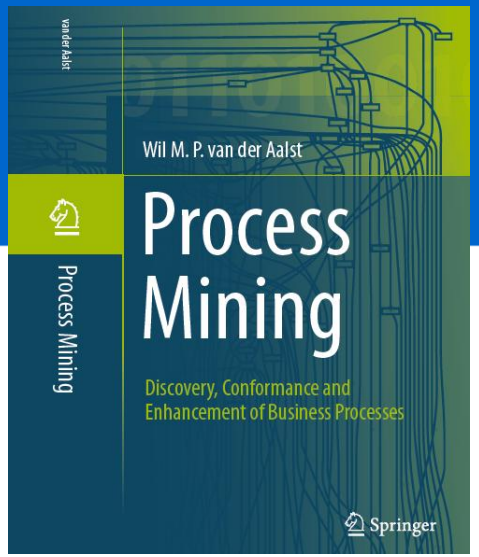
Process Discovery

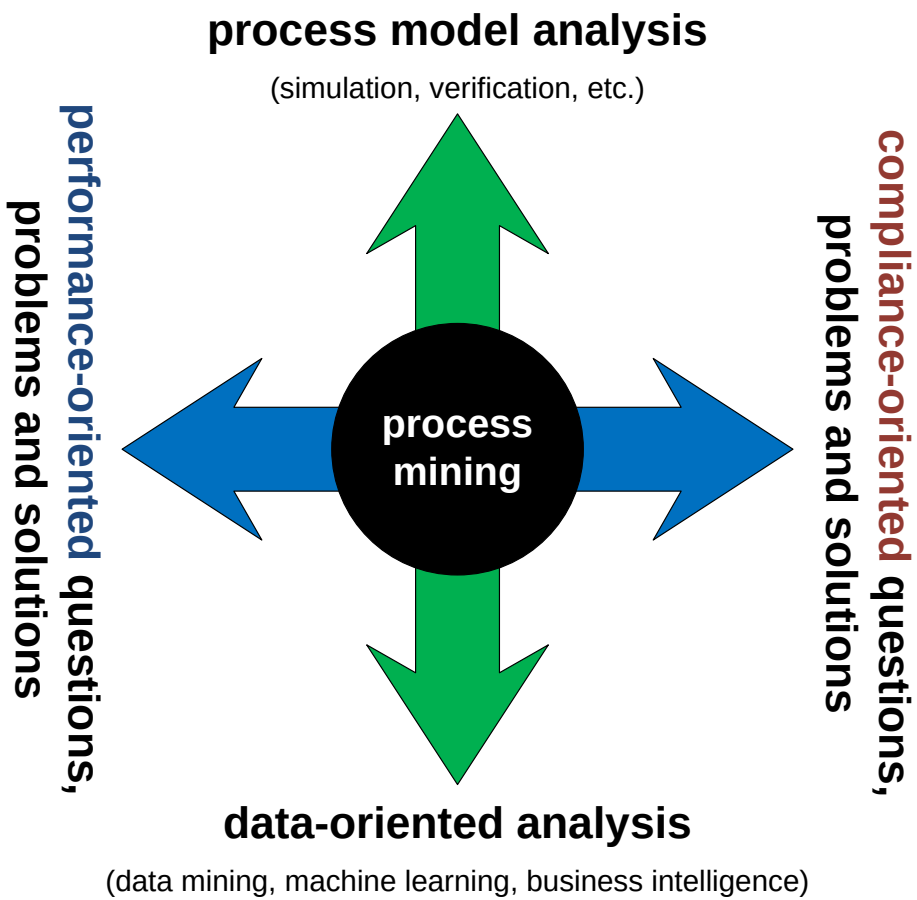
prof.dr.ir. Wil van der Aalst
www.processmining.org



Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

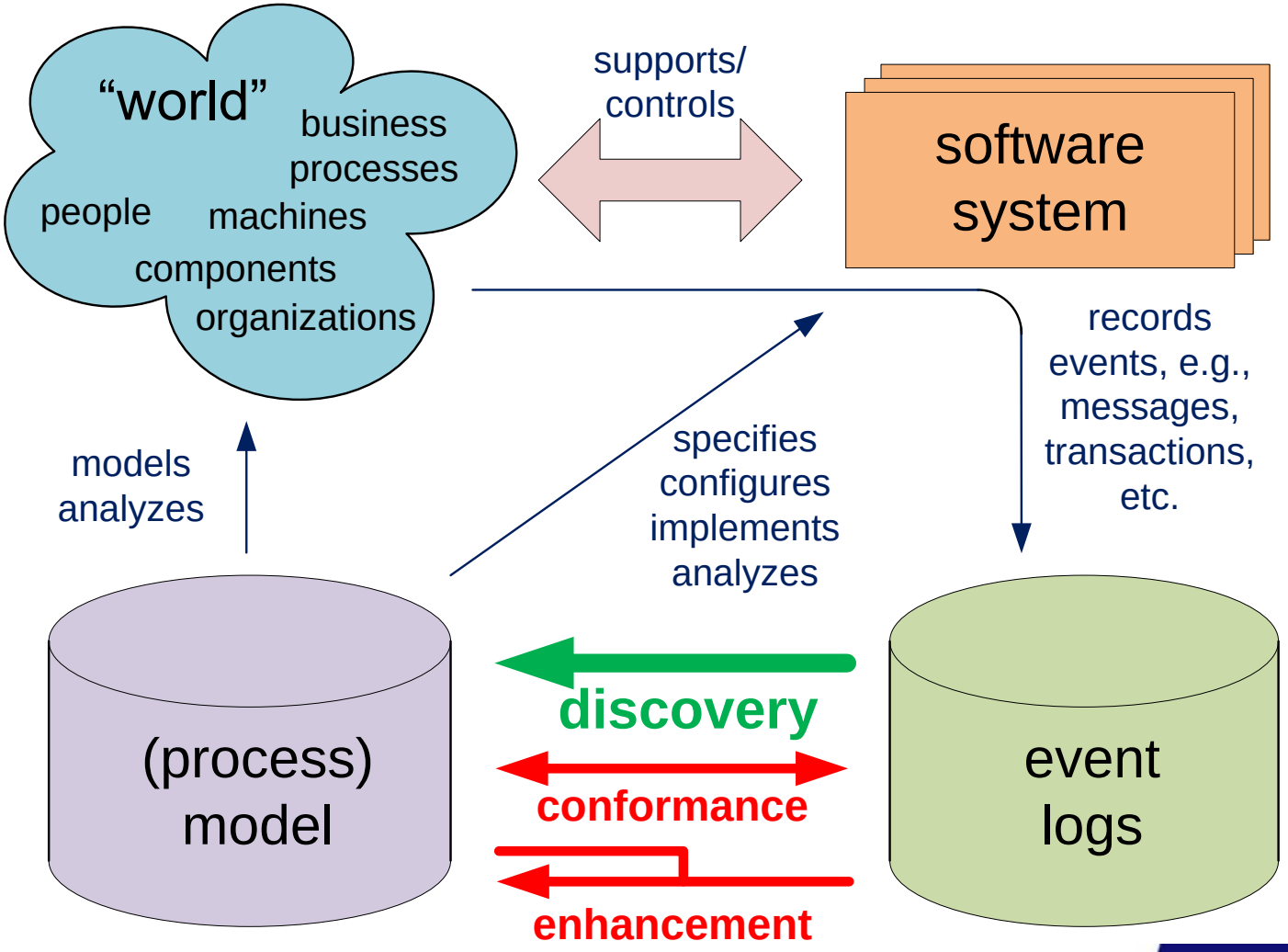
Positioning Process Mining




www.olifantenpaadjes.nl
60
0100110010101010
0100110101010


0100110101010101010

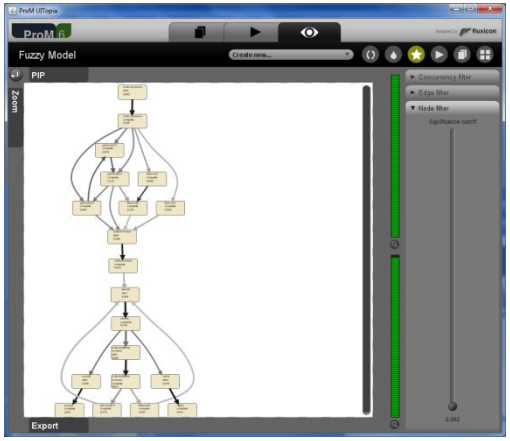
Overview

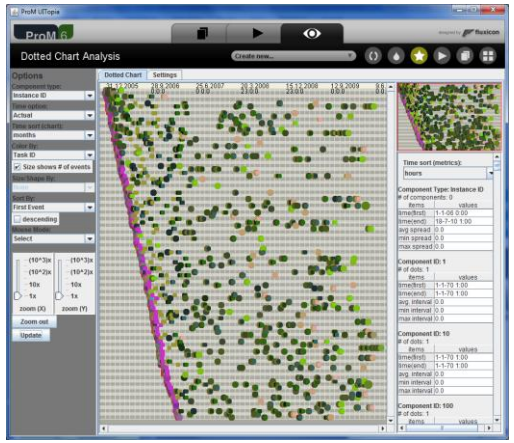

"world"
business processes
people
machines
components
organizations
supports/ controls
software system
records events, e.g., messages, transactions, etc.
models analyzes
specifies configures implements analyzes
(process) model
discovery
conformance
enhancement
event logs

Hundreds of plug-ins available covering the whole process mining spectrum

ProM
process mining workbench

open-source (L-GPL)

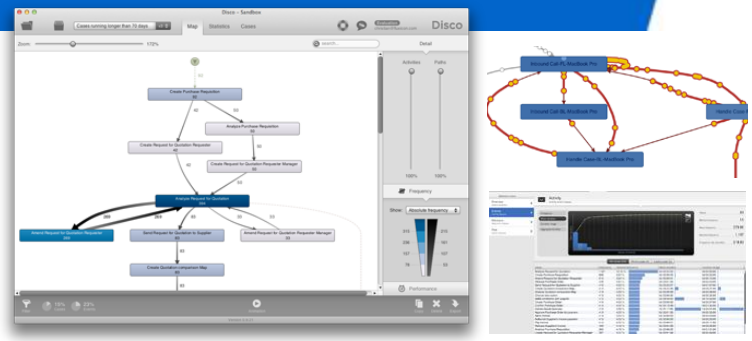


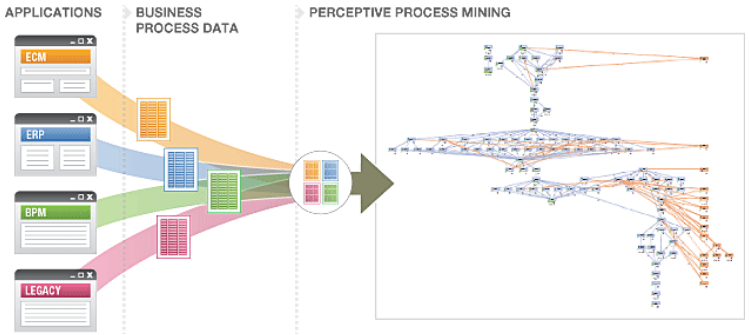


Download from: www.processmining.org

Commercial Alternatives

- **Disco (Fluxicon)**
- **Perceptive Process Mining**
  (before Futura Reflect and BPM|one)
- **ARIS Process Performance Manager**
- **QPR ProcessAnalyzer**
- **Interstage Process Discovery (Fujitsu)**
- **Discovery Analyst (StereoLOGIC)**
- **XMAnalyzer (XMPro)**
- **…**





Process Discovery

Process Discovery (small selection)

distributed genetic mining

automata-based learning

language-based regions

heuristic mining

genetic mining

state-based regions

stochastic task graphs

LTL mining

neural networks

fuzzy mining

hidden Markov models

mining block structures

α algorithm

conformal process graph

multi-phase mining

partial-order based mining

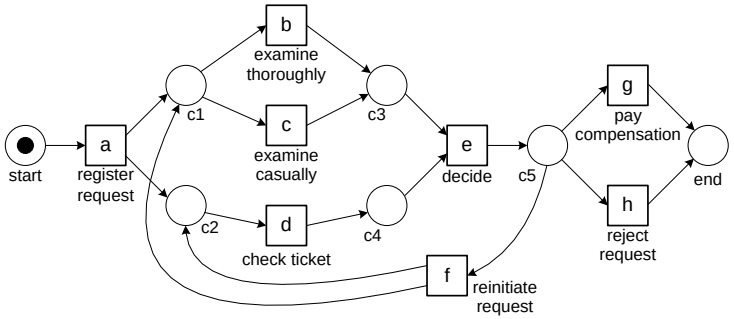
α# algorithm

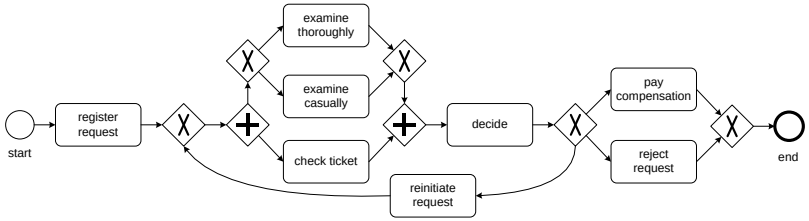
ILP mining

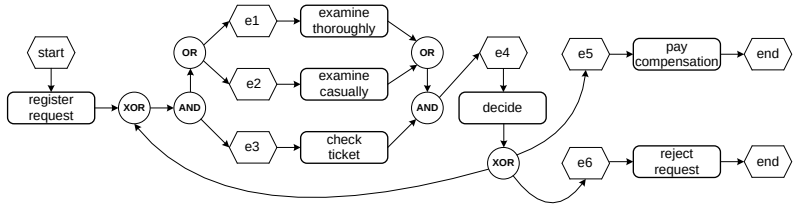
α++ algorithm

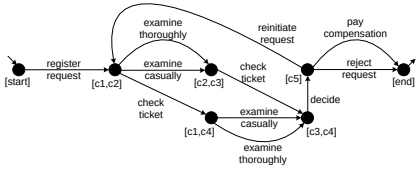
Typical Representational Bias

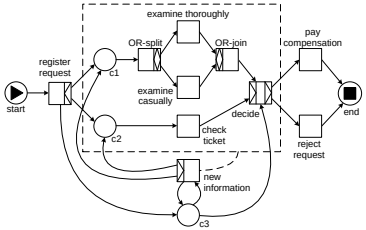
- **(Labeled) Petri Nets, WF-nets, etc.**
- **Subsets of**
  - **BPMN diagrams,**
  - **UML Activity Diagrams,**
  - **Event-Driven Process Chains (EPCs),**
  - **YAWL,**
  - **Statecharts?**
  - **etc.**
- **Transition Systems**
- **(Hidden) Markov Models**
- **…**





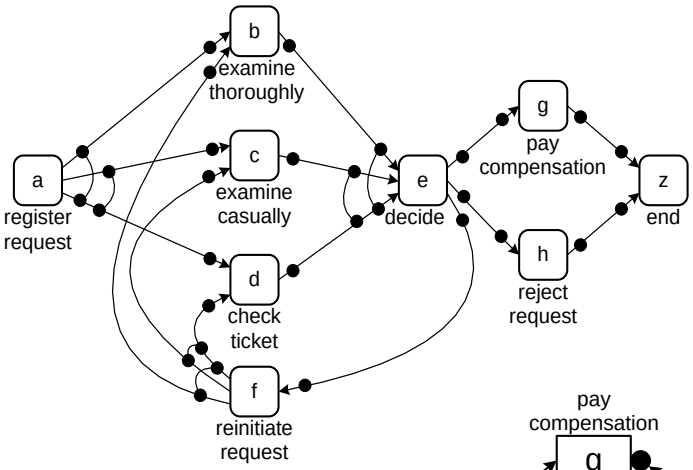


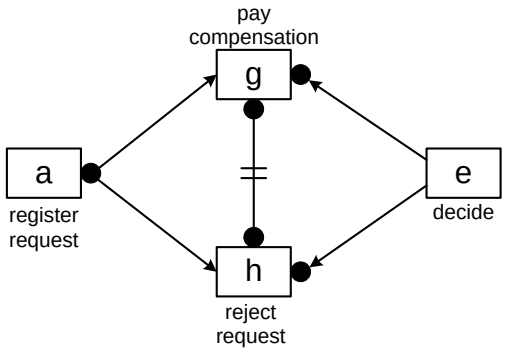


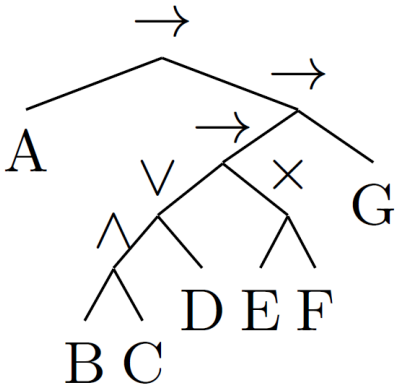


Alternative Representational Bias

1. **C-nets (XOR/AND/OR-split/join graphs; more likely to be sound due to declarative semantics).**
2. **Declare models (constraint based, grounded in LTL; anything is possible unless forbidden)**
3. **Process Trees (similar to subsets of various process algebras; sound by structure)**





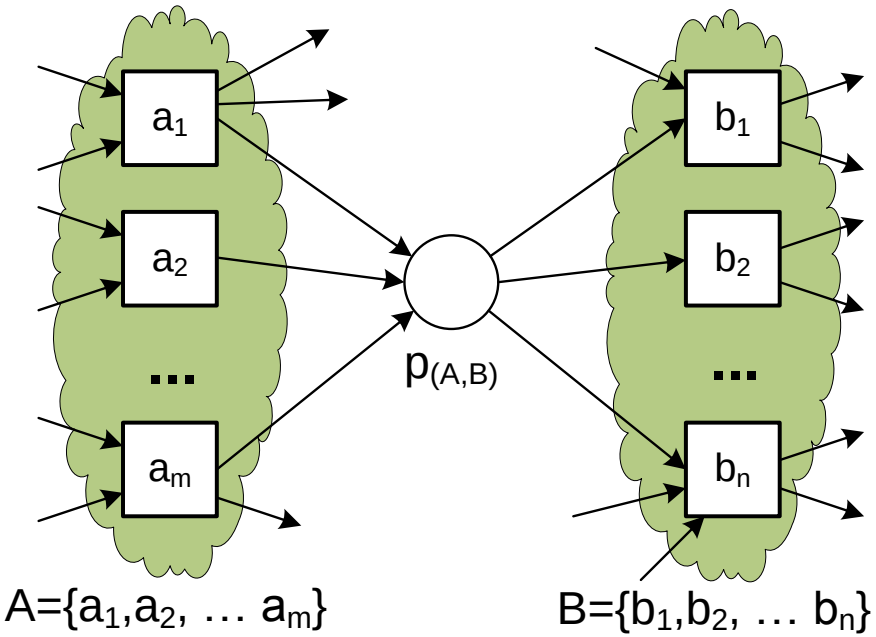


Petri net/Region-based View

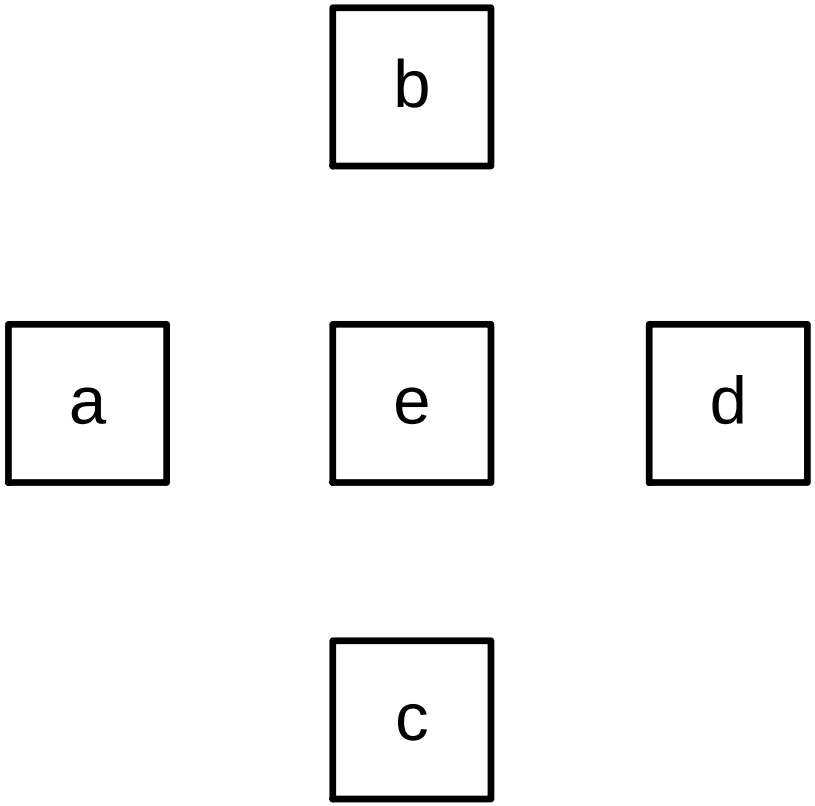
Petri net view:
Just discover the places …

Adding a place limits behavior:

- **overfitting ≈ adding too many places**
- **underfitting ≈ adding too few places**



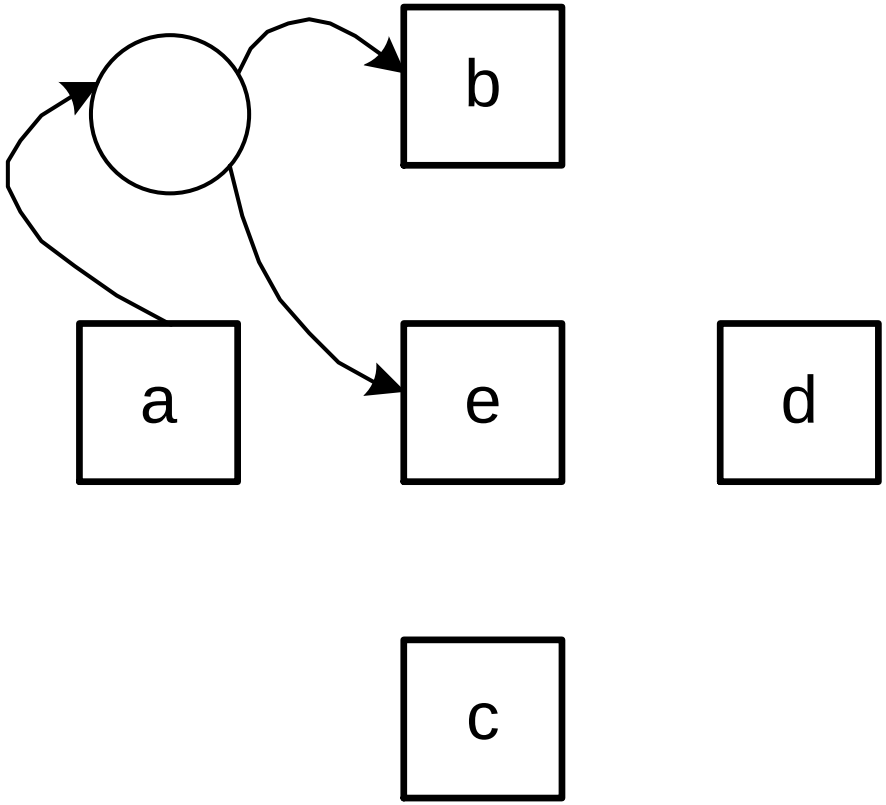
The Petri net below can replay any trace over {a,b,c,d,e}



1.	abcd	1.	aaa	1.	a
2.	aed	2.	aaa	2.	a
3.	acbd	3.	bbb	3.	a
4.	abcd	4.	bb	4.	a
5.	abcd	5.	aaa	5.	a
6.	acbd	6.	ddd	6.	a
		7.	ee	7.	a
		8.	...	8.	...

$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

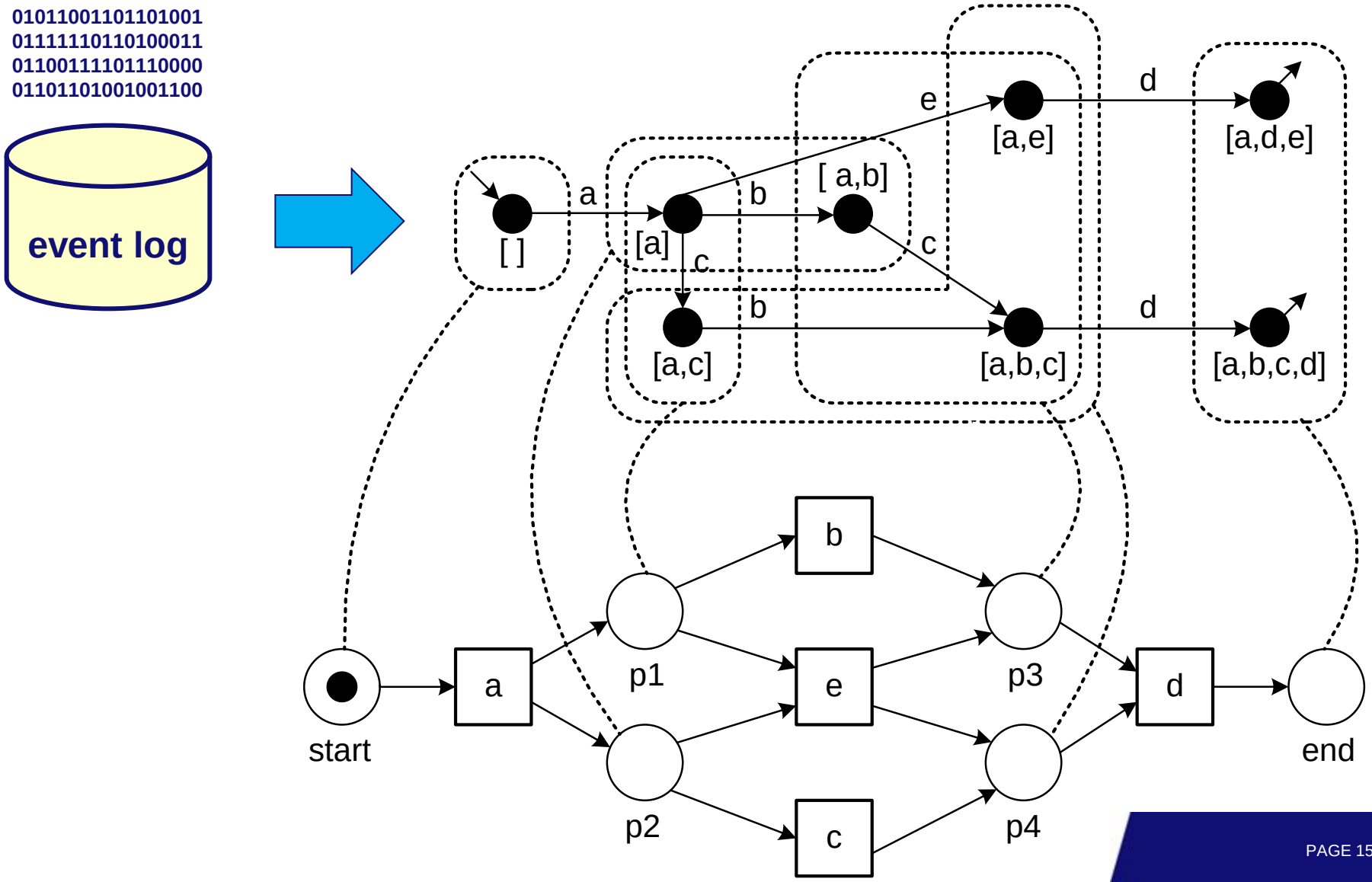
Place limits behavior



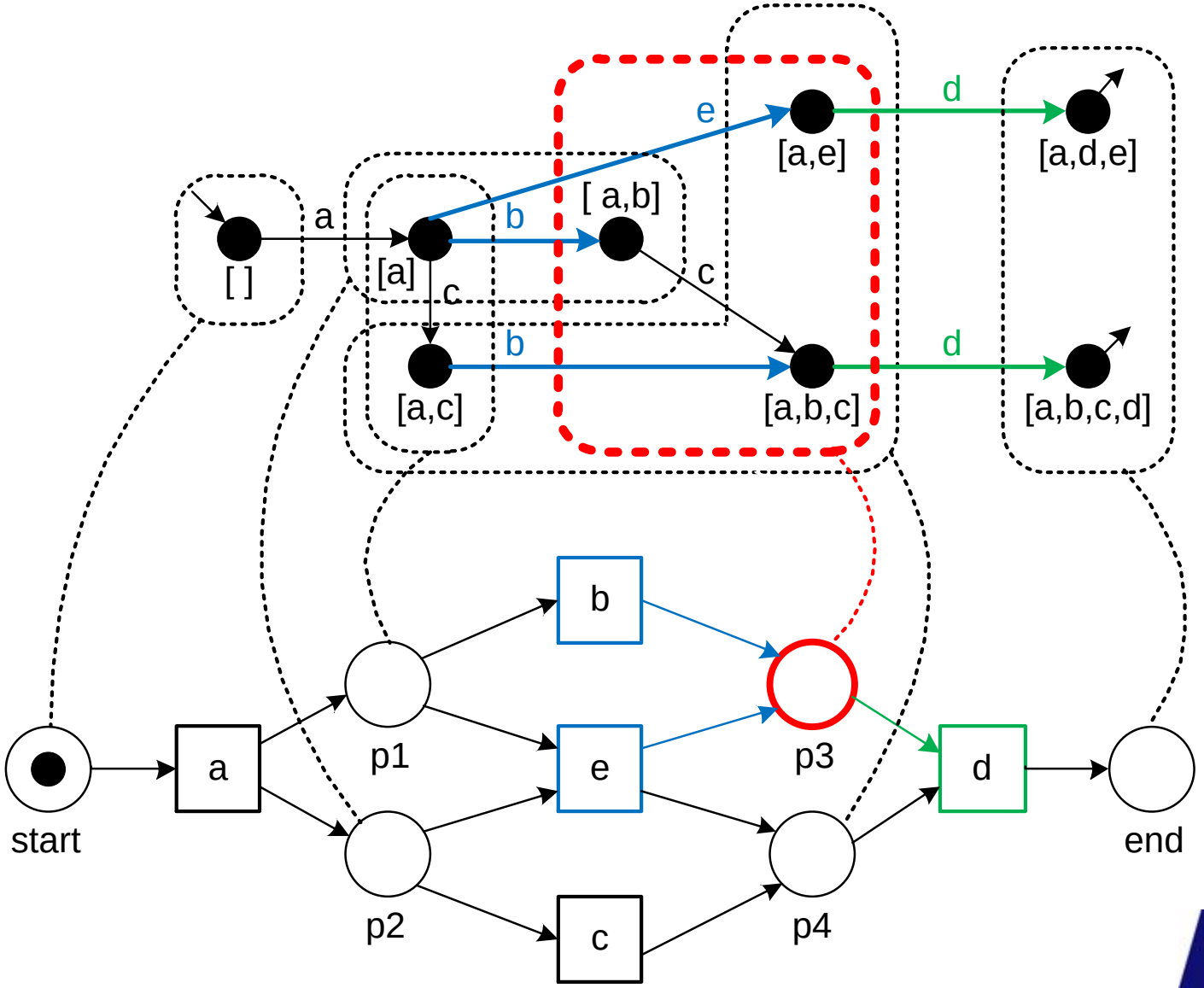
1. abcd
2. aed
3. acbd
4. abcd
5. abcd
6. acbd

1. abcd
2. bcd
3. aed
4. abed
5. cbd
6. acbd

Example: Process Discovery Using State-Based Regions

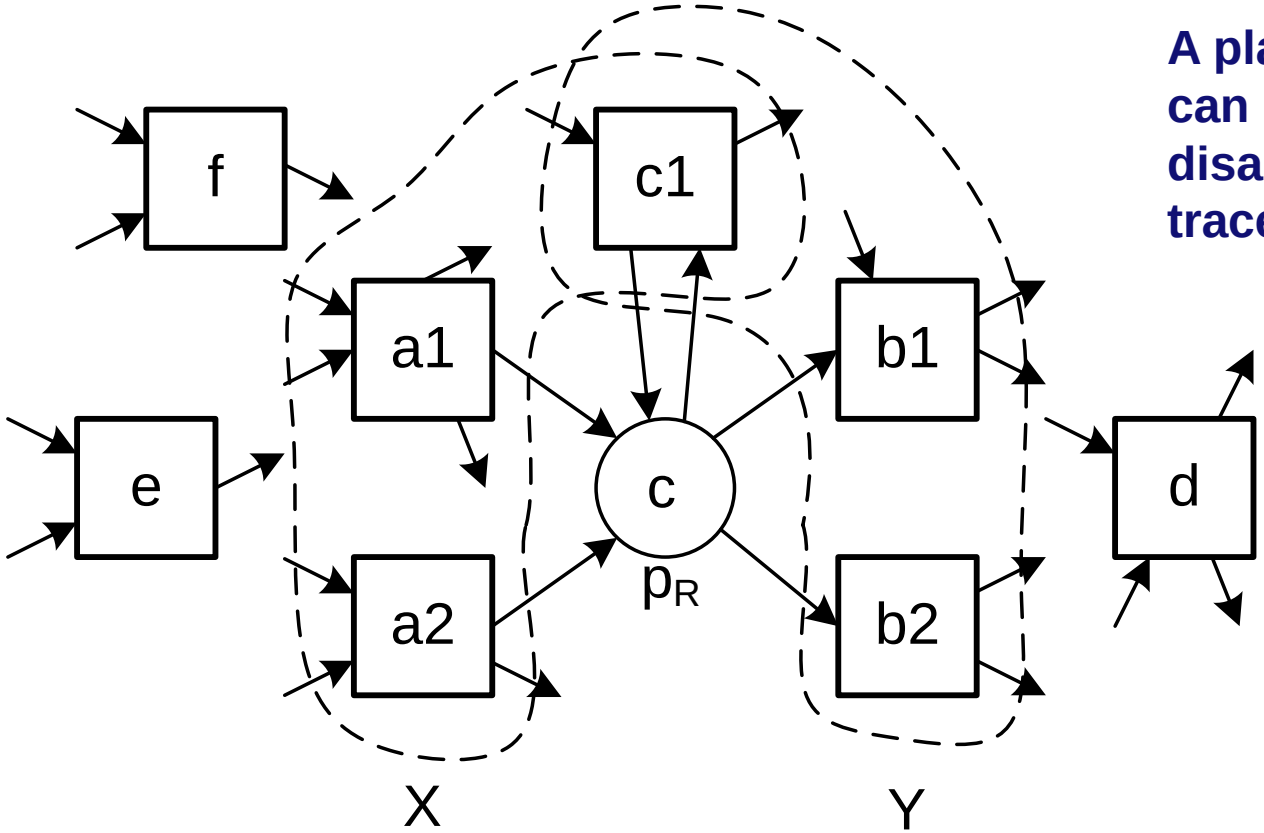


Example of State-Based Region



enter: b,e
leave: d
do-not-cross: a,c

Example: Process Discovery Using Language-Based Regions



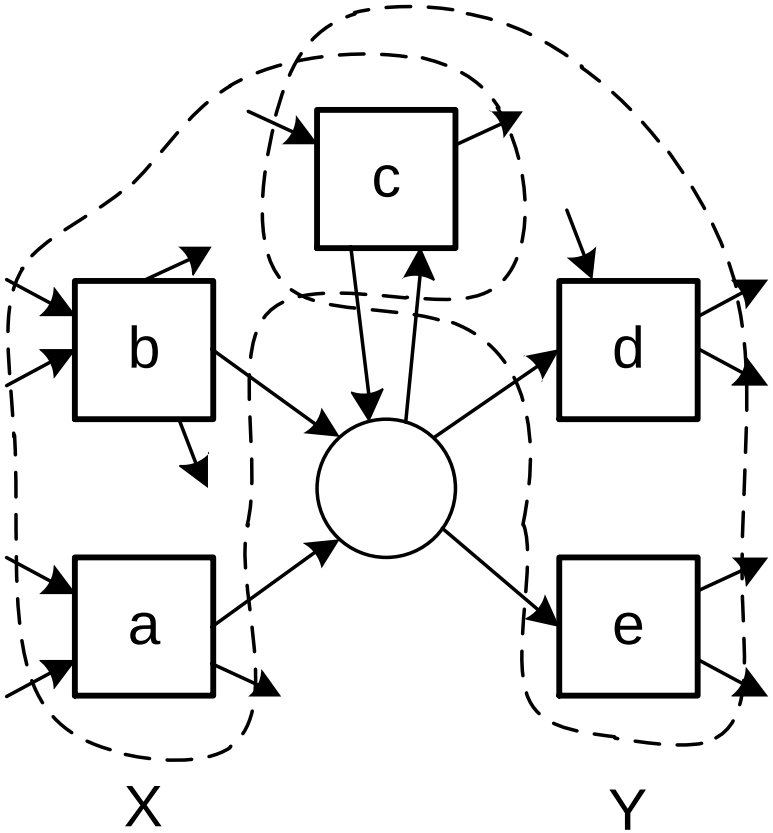
A place is feasible if it can be added without disabling any of the traces in the event log.

for any $\sigma \in L, k \in \{1, \ldots, |\sigma|\}, \sigma_1 = hd^{k-1}(\sigma), a = \sigma(k), \sigma_2 = hd^k(\sigma) = \sigma_1 \oplus a$:

$$c + \sum_{t \in X} \partial_{multiset}(\sigma_1)(t) - \sum_{t \in Y} \partial_{multiset}(\sigma_2)(t) \geq 0.$$

Example of Language-Based Regions

1. accd
2. bd
3. bce
4. ace
5. acd
6. bcce
7. ade



↓accd : 0 + 0 - 0 ≥ 0

a↓ccd : 0 + 1 - 1 ≥ 0

ac↓cd : 0 + 2 - 2 ≥ 0

acc↓d : 0 + 3 - 3 ≥ 0

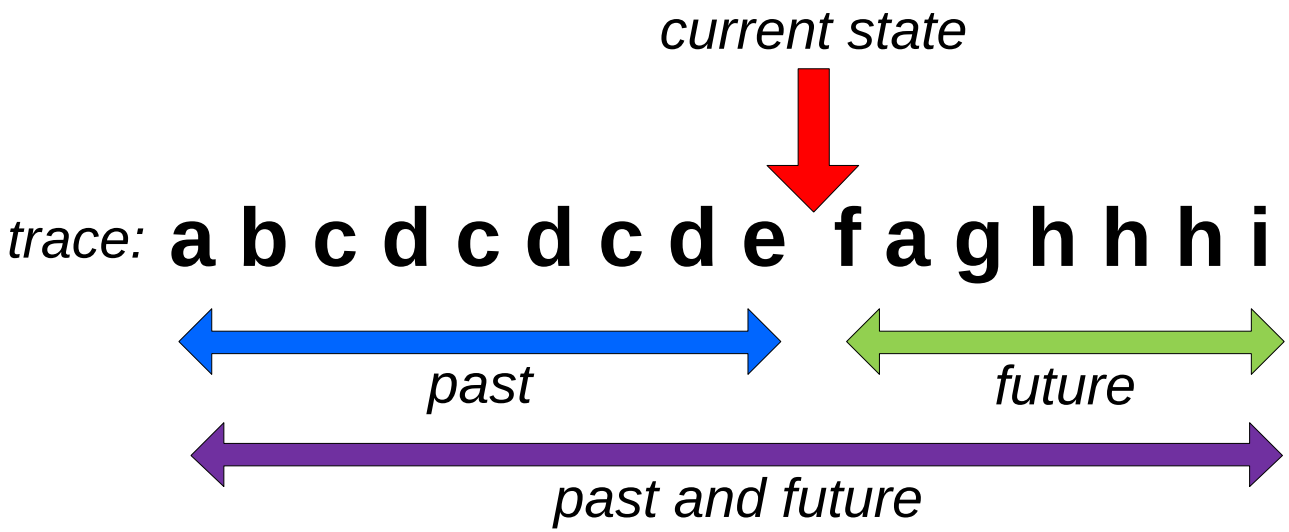
↓ade : 0 + 0 - 0 ≥ 0

a↓de : 0 + 1 - 1 ≥ 0

ad↓e : 0 + 1 - 2 < 0

Creating a Transition System

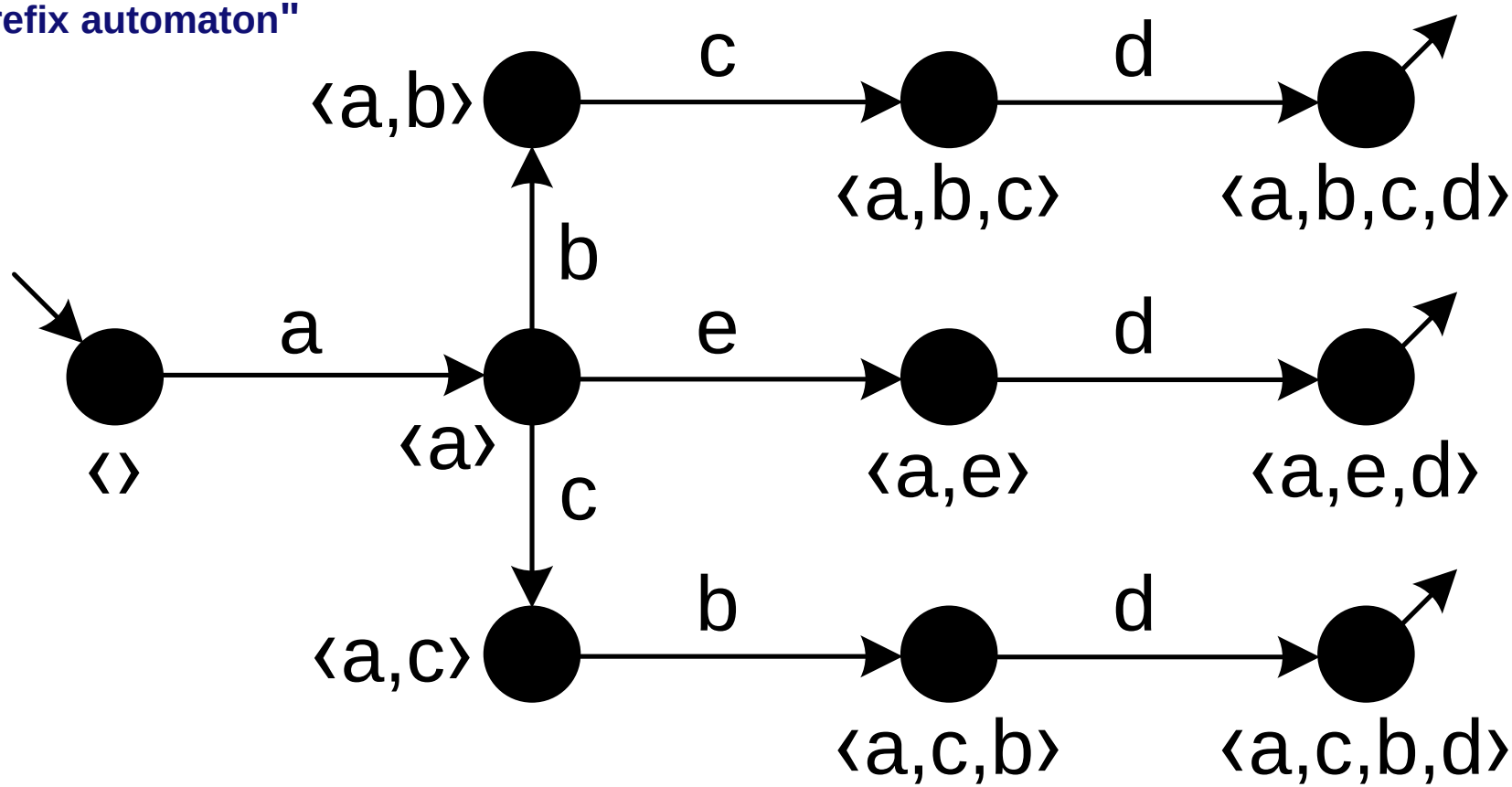
Learning a Transition System



- **past, future, past+future**
- **sequence, multiset, set abstraction**
- **limited horizon to abstract further**
- **filtering e.g. based on transaction type, names, etc.**
- **labels based on activity name or other features**

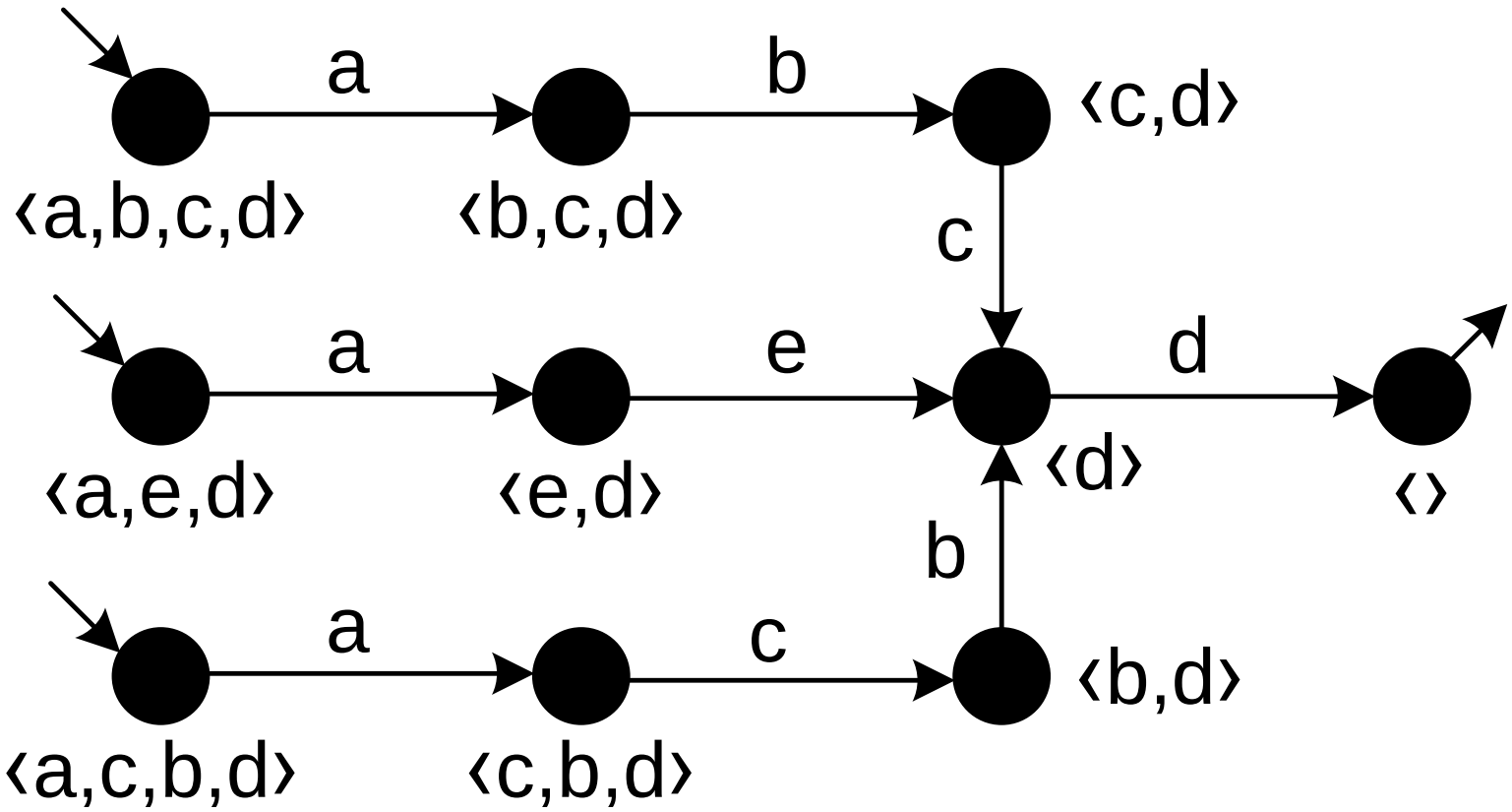
Past Without Abstraction (Full Sequence)

Sometimes called the "prefix automaton"



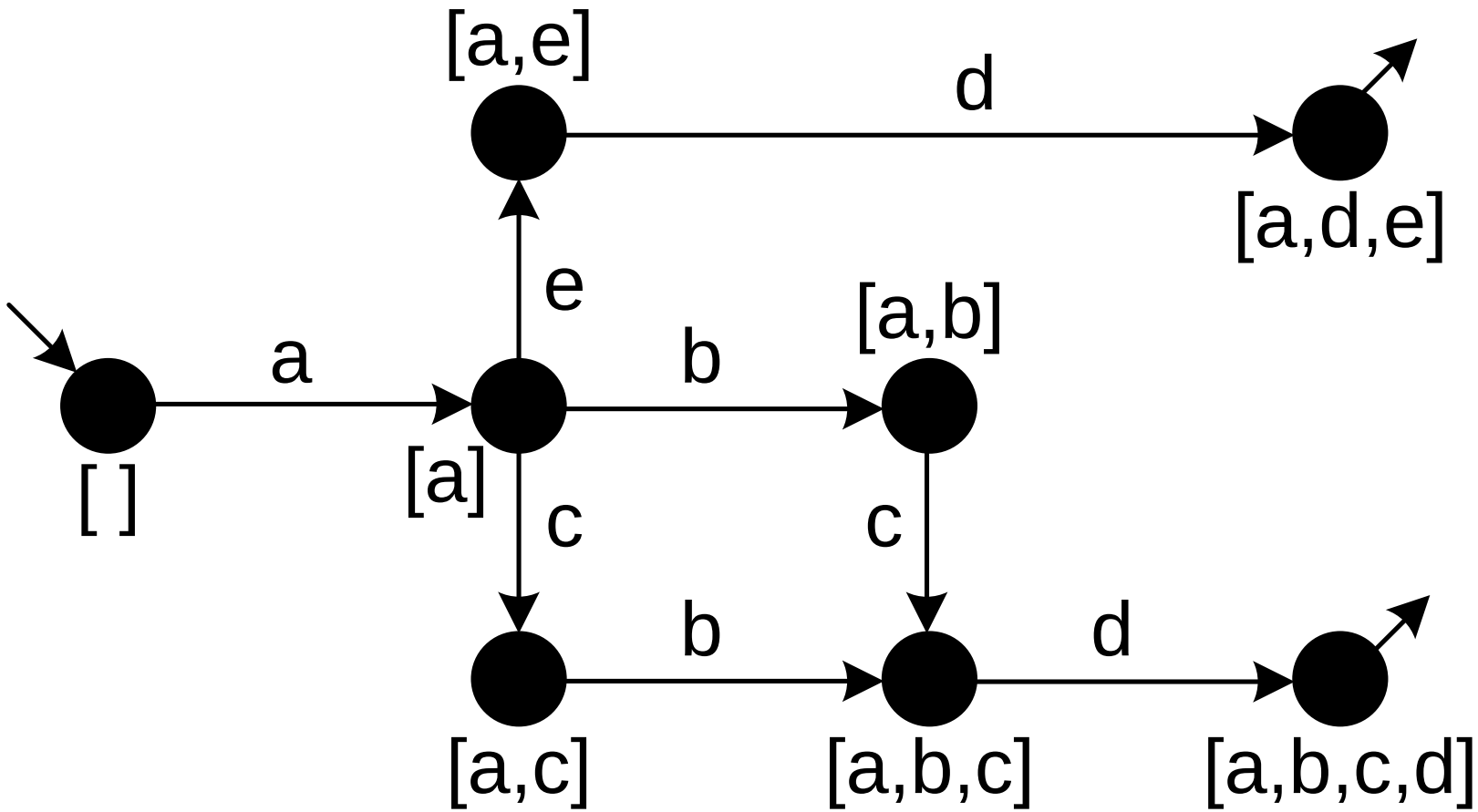
$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Future Without Abstraction



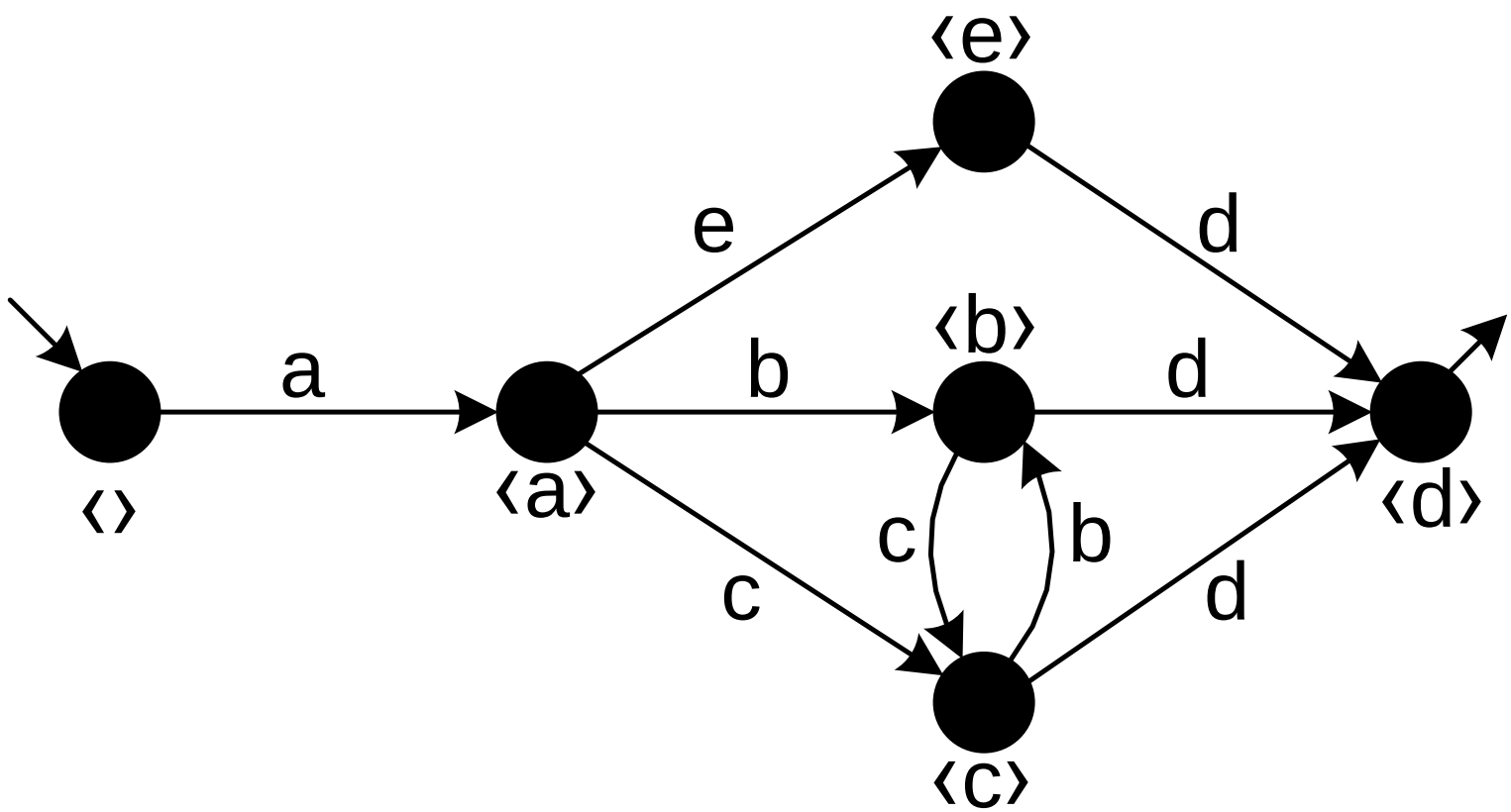
$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Past with Multiset Abstraction



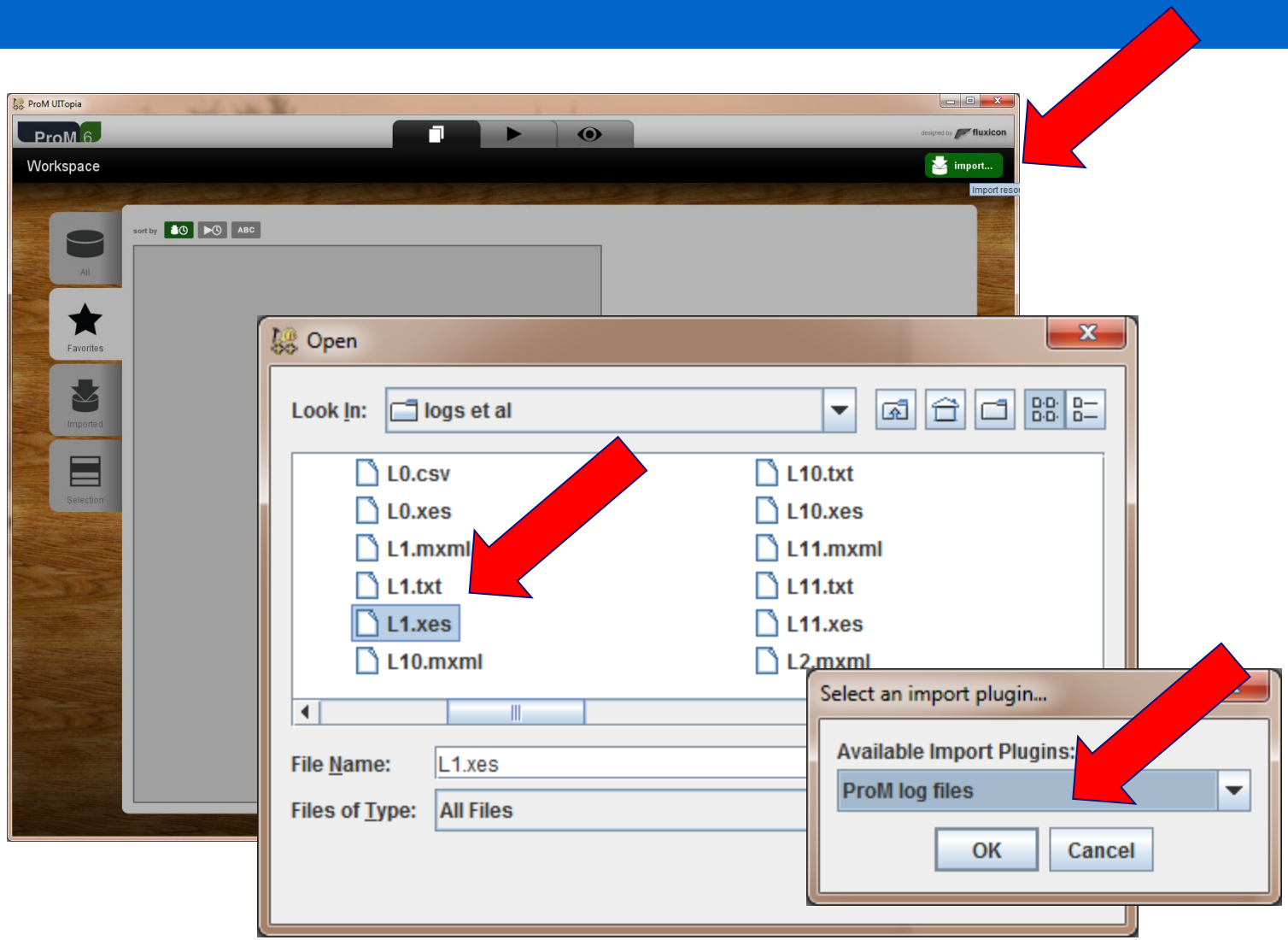
$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Only Last Event Matters For State

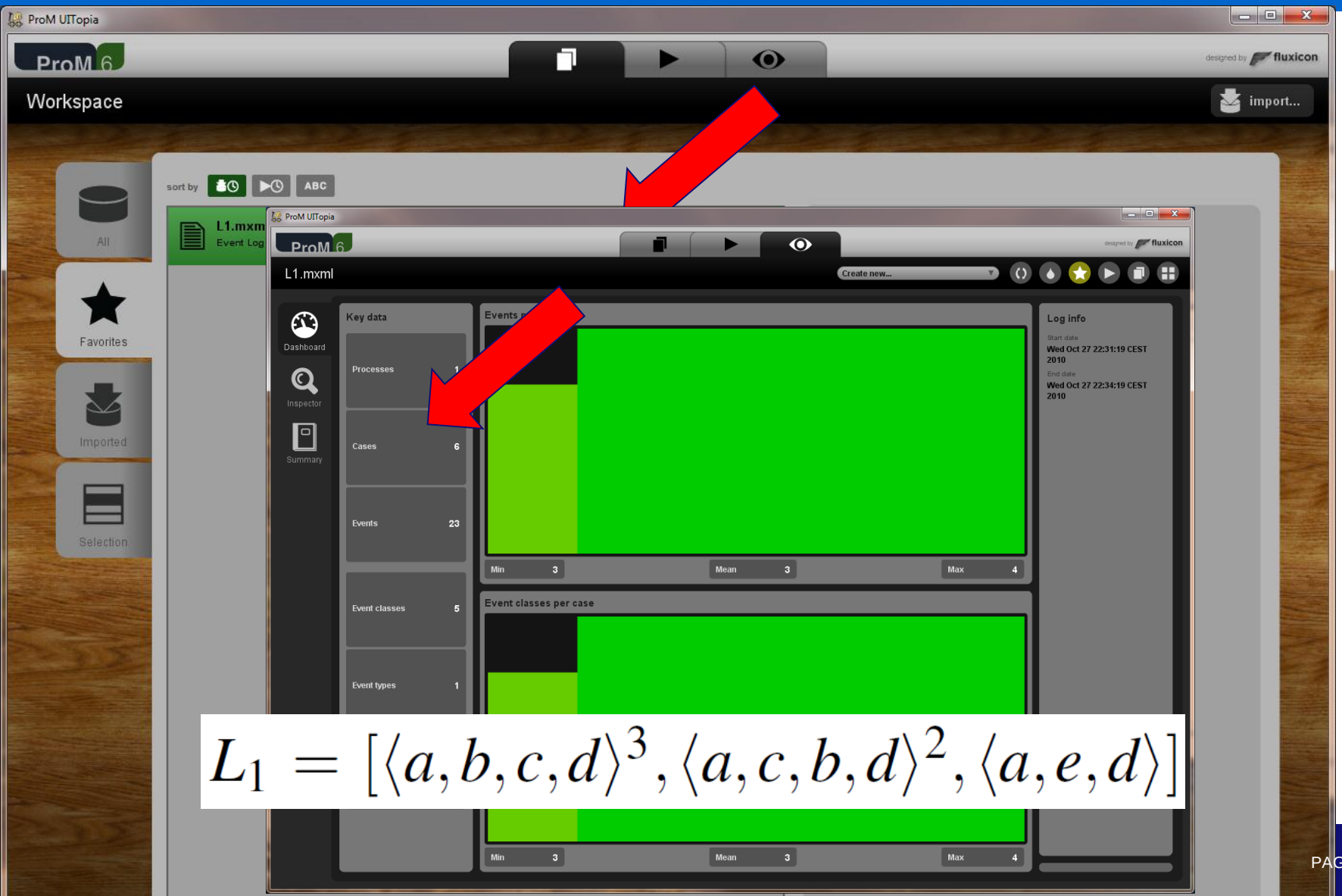


$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Using ProM

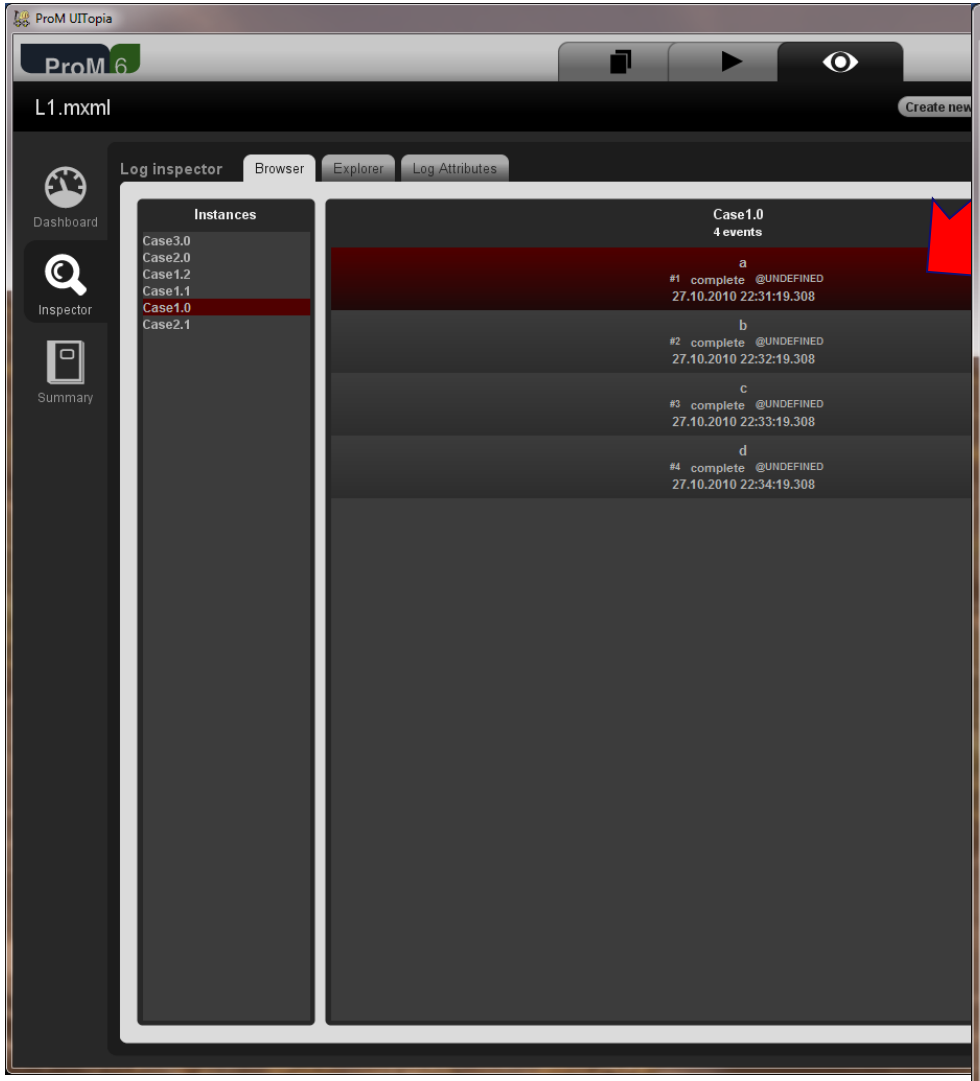

ProM UITopia
ProM 6
Workspace
import...
All
Favorites
Imported
Selection
Open
Look In:
logs et al
L0.csv
L0.xes
L1.mxml
L1.txt
L1.xes
L10.mxml
L10.txt
L10.xes
L11.mxml
L11.txt
L11.xes
L2.mxml
File Name:
L1.xes
Files of Type:
All Files
Select an import plugin...
Available Import Plugins:
ProM log files
OK
Cancel

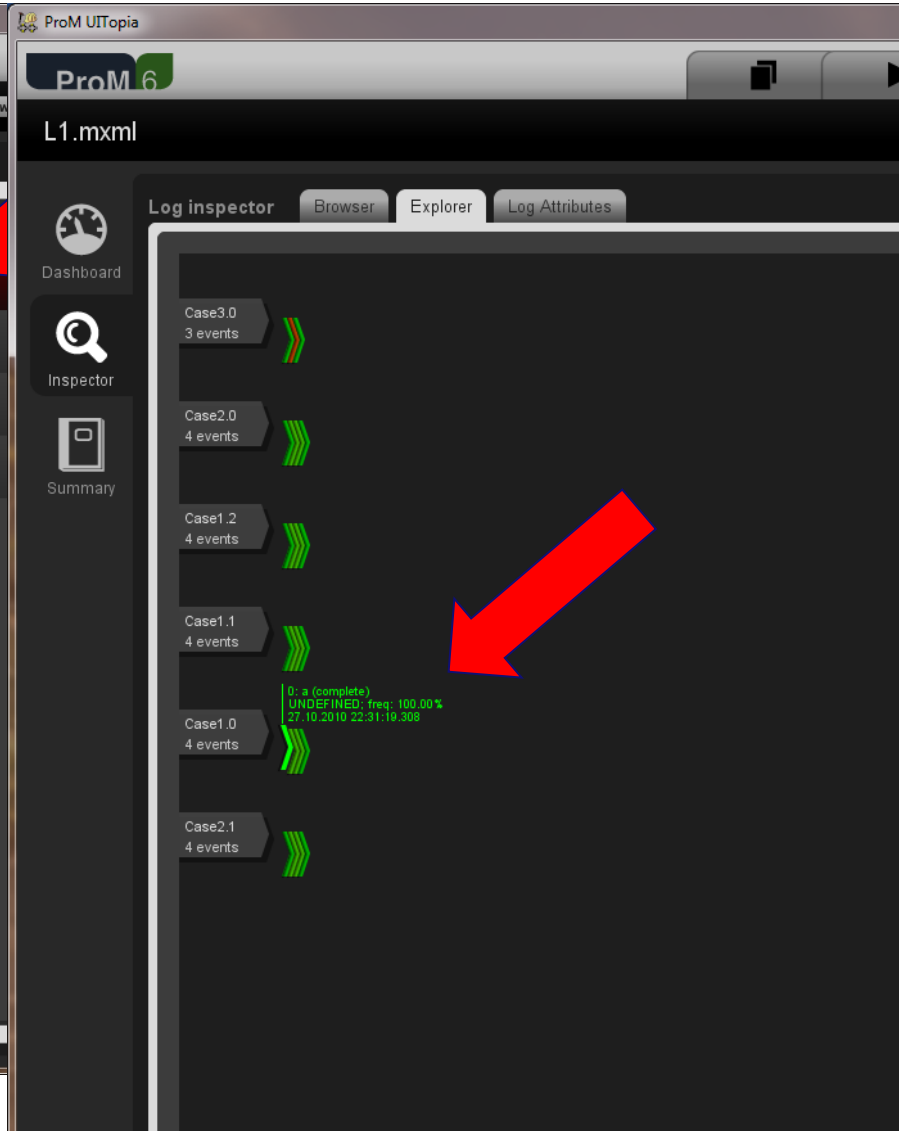
Inspect Event Log



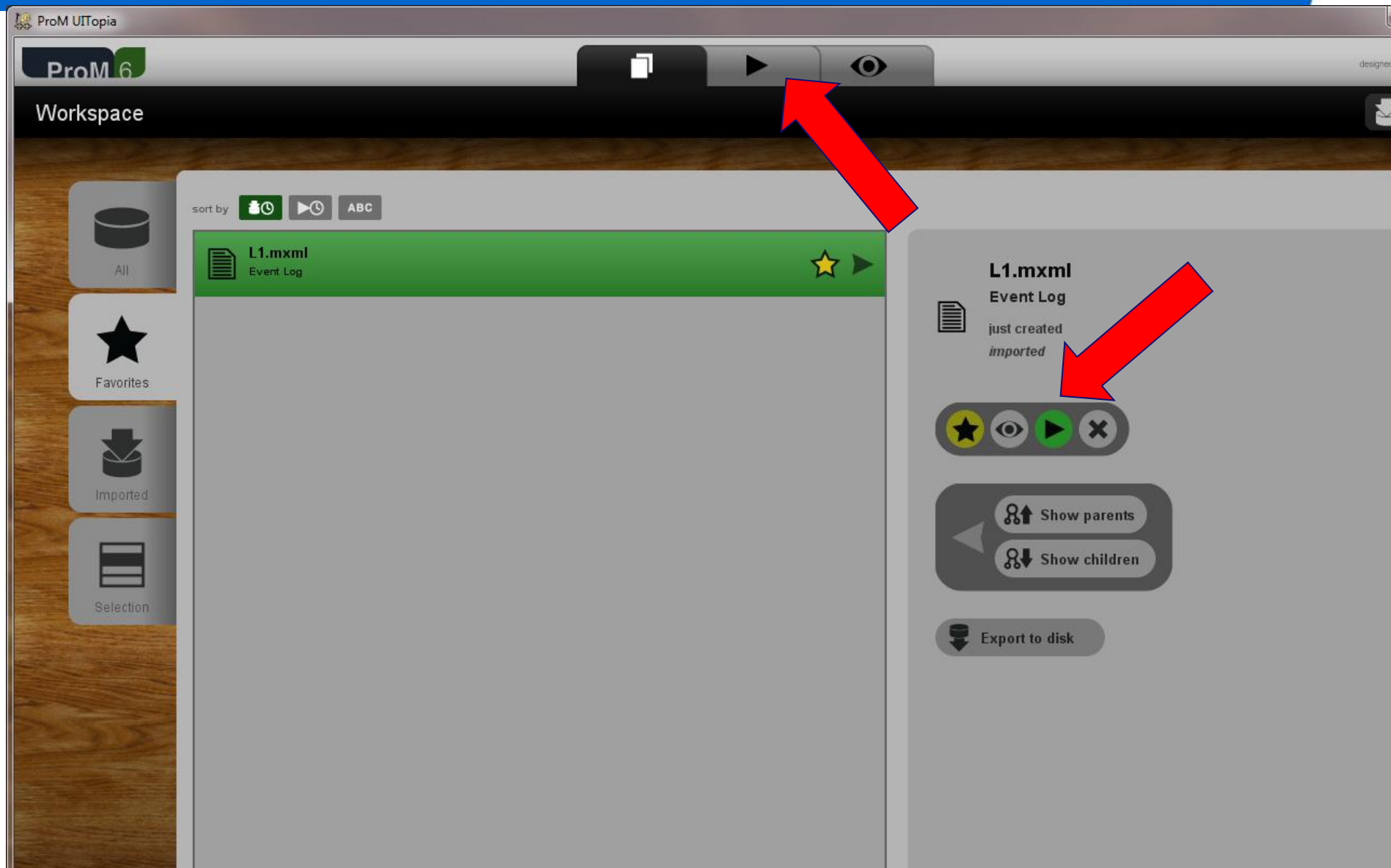
$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

Inspect Traces





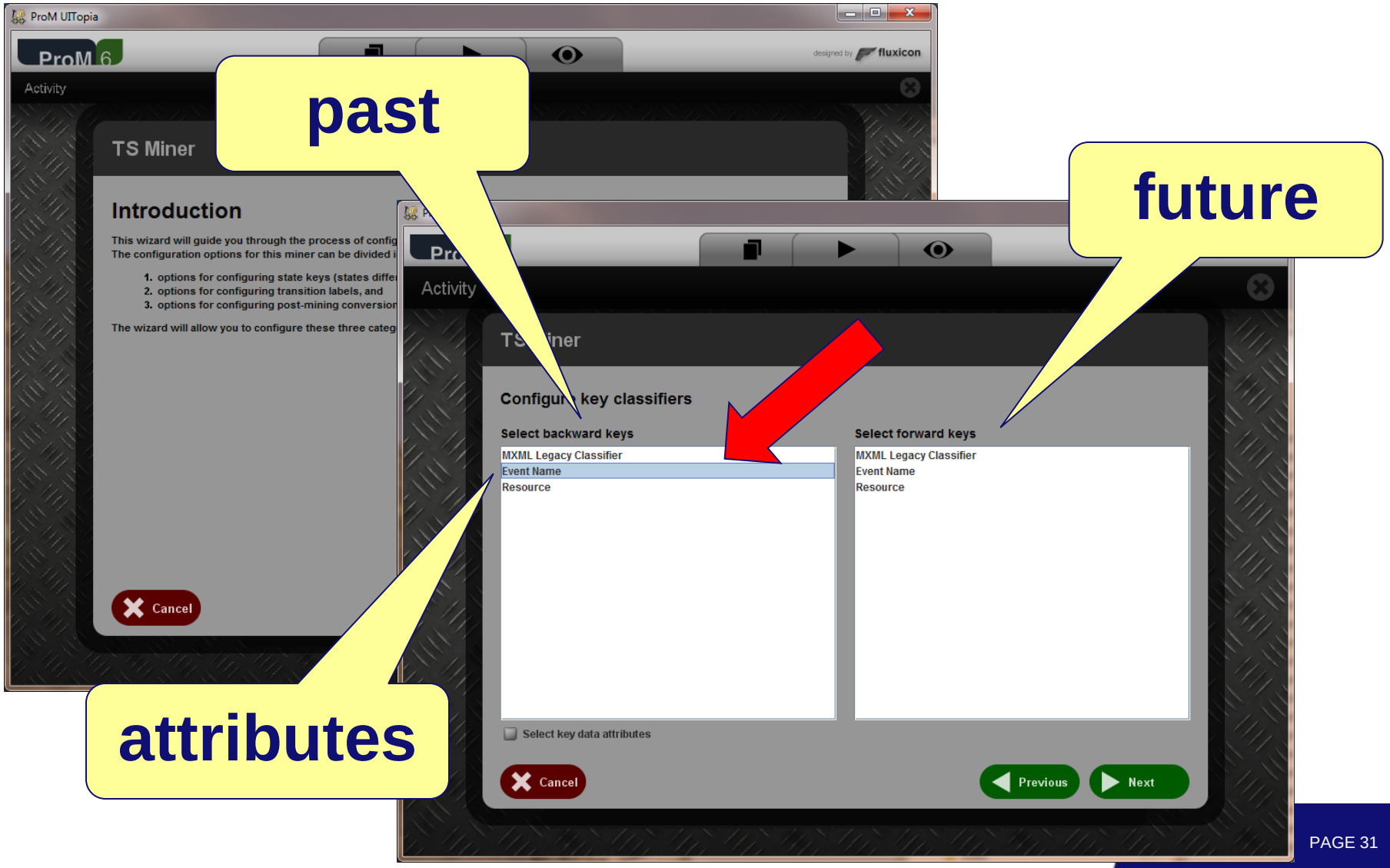
Run Plugin


ProM UITopia
ProM 6
Workspace
All
Favorites
Imported
Selection
sort by
ABC
L1.mxml
Event Log
L1.mxml
Event Log
just created
imported
Show parents
Show children
Export to disk

Select (scroll or by name)

Start Plugin "Mine Transition System"

Start Window

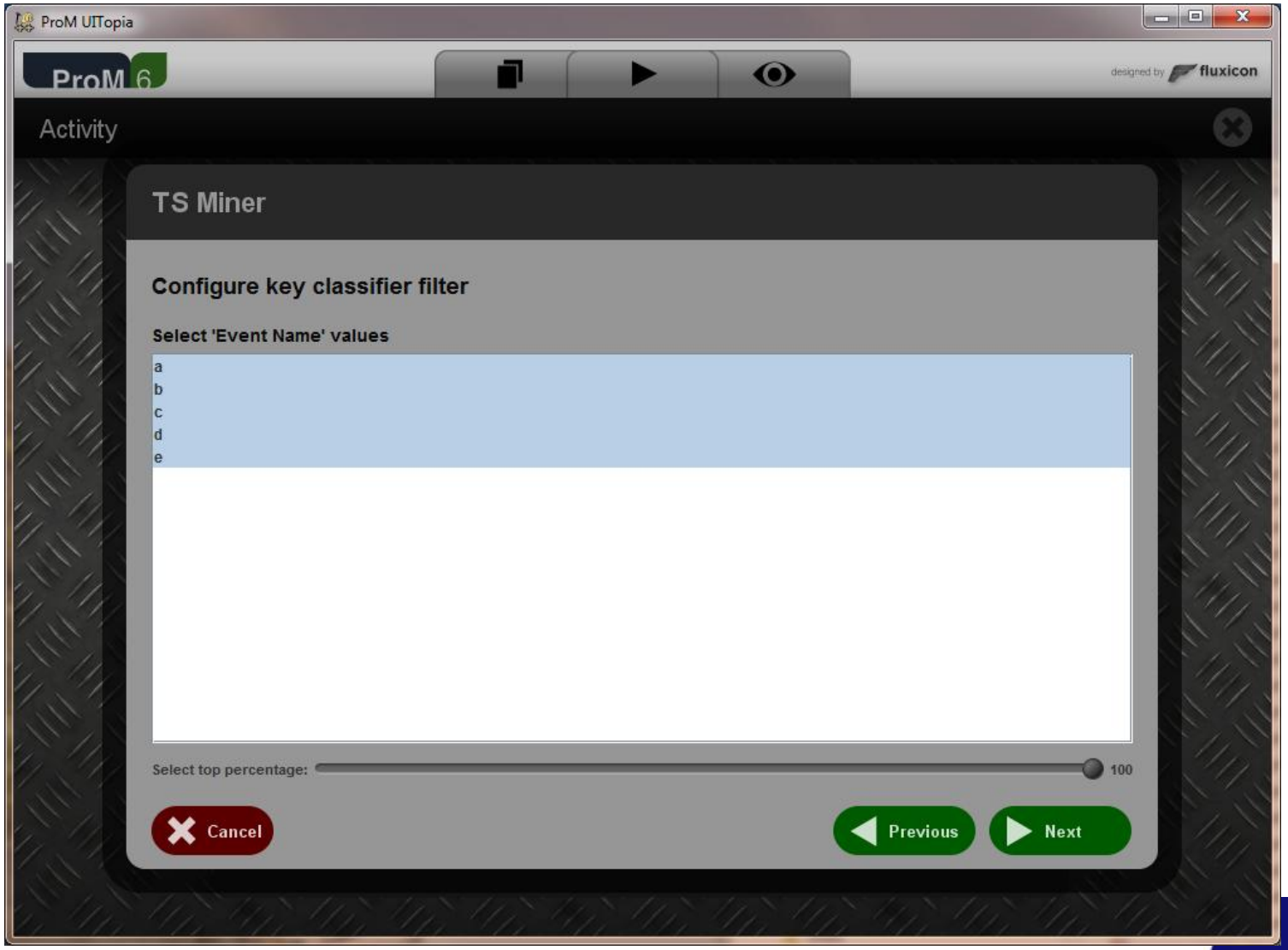


Abstraction

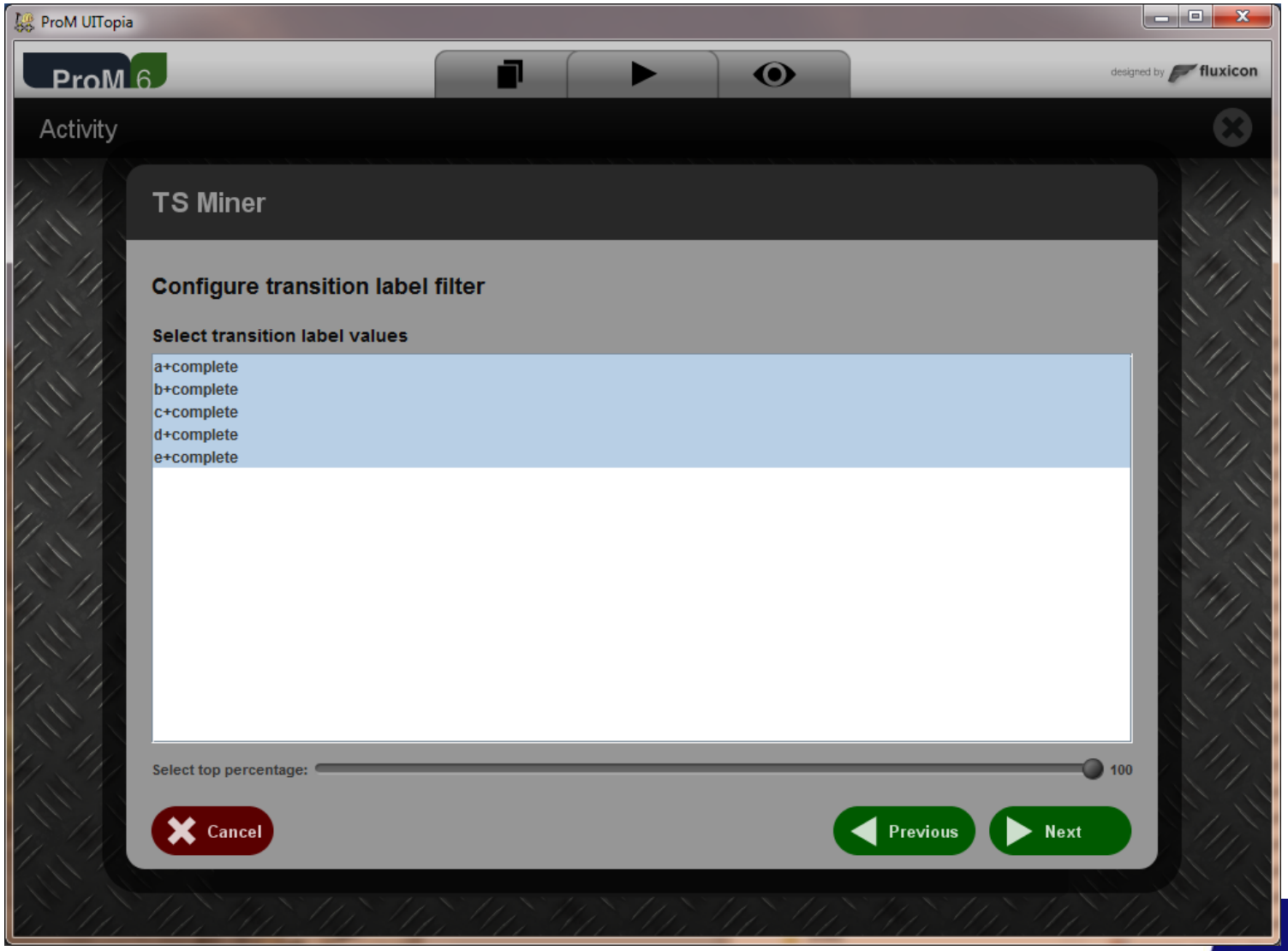
list, multiset, or set

all, or only last k events

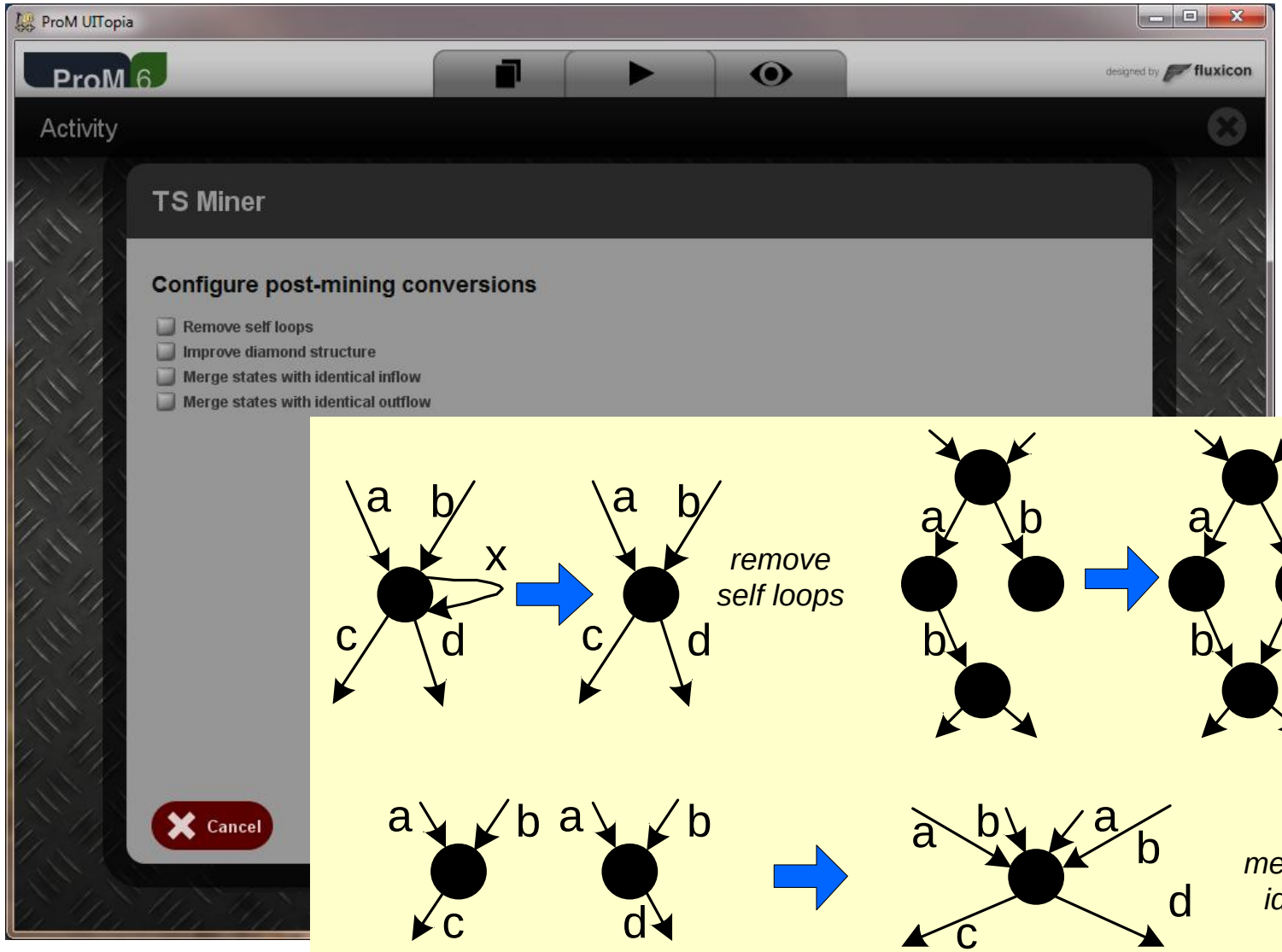
Which events to filter?

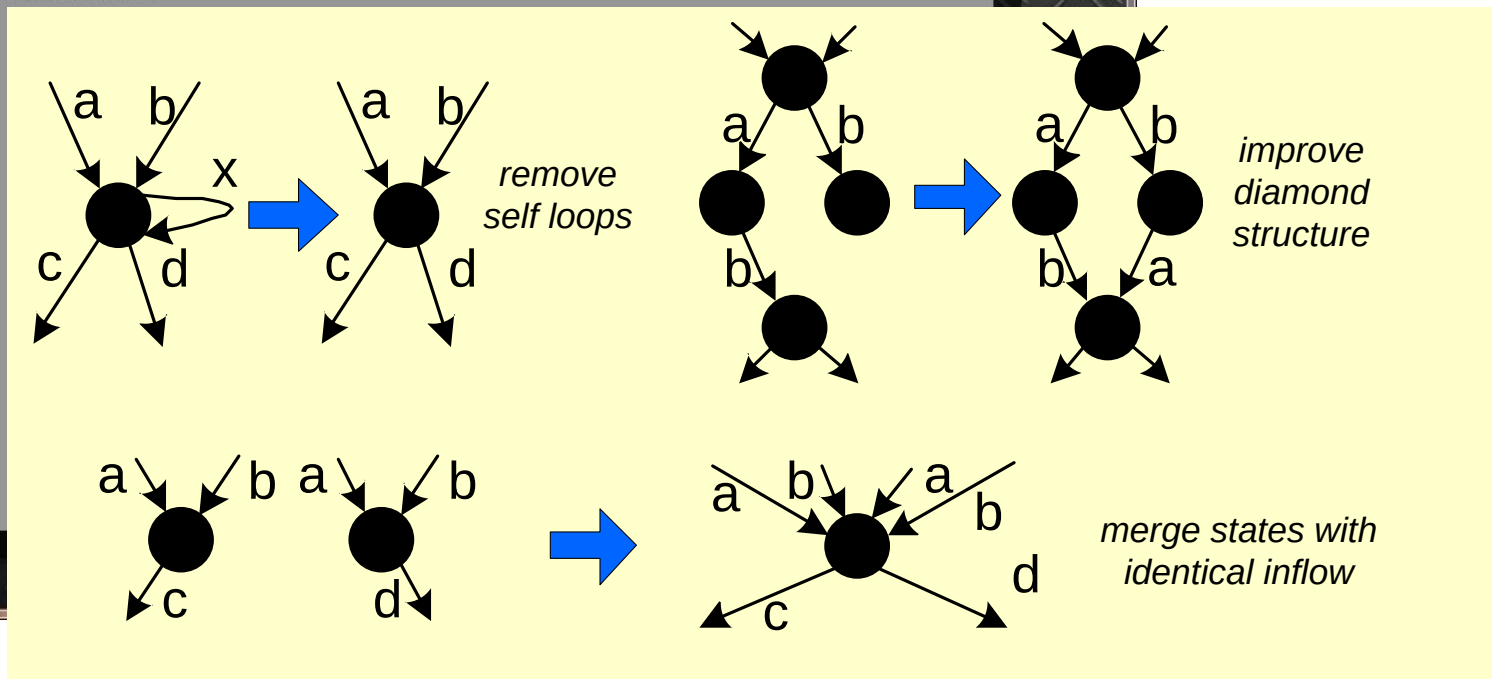


Which labels need to be visible?

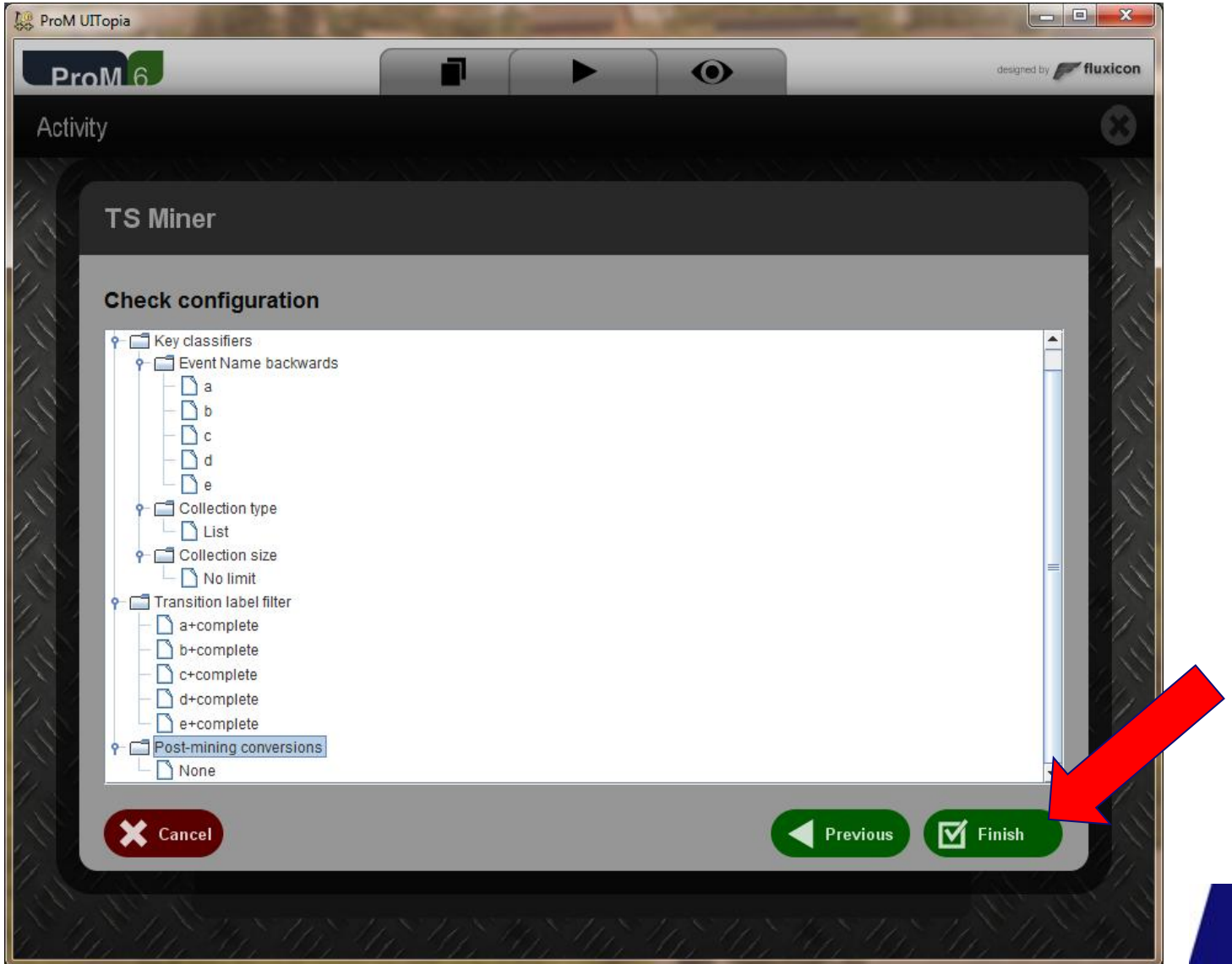


Any repair actions?

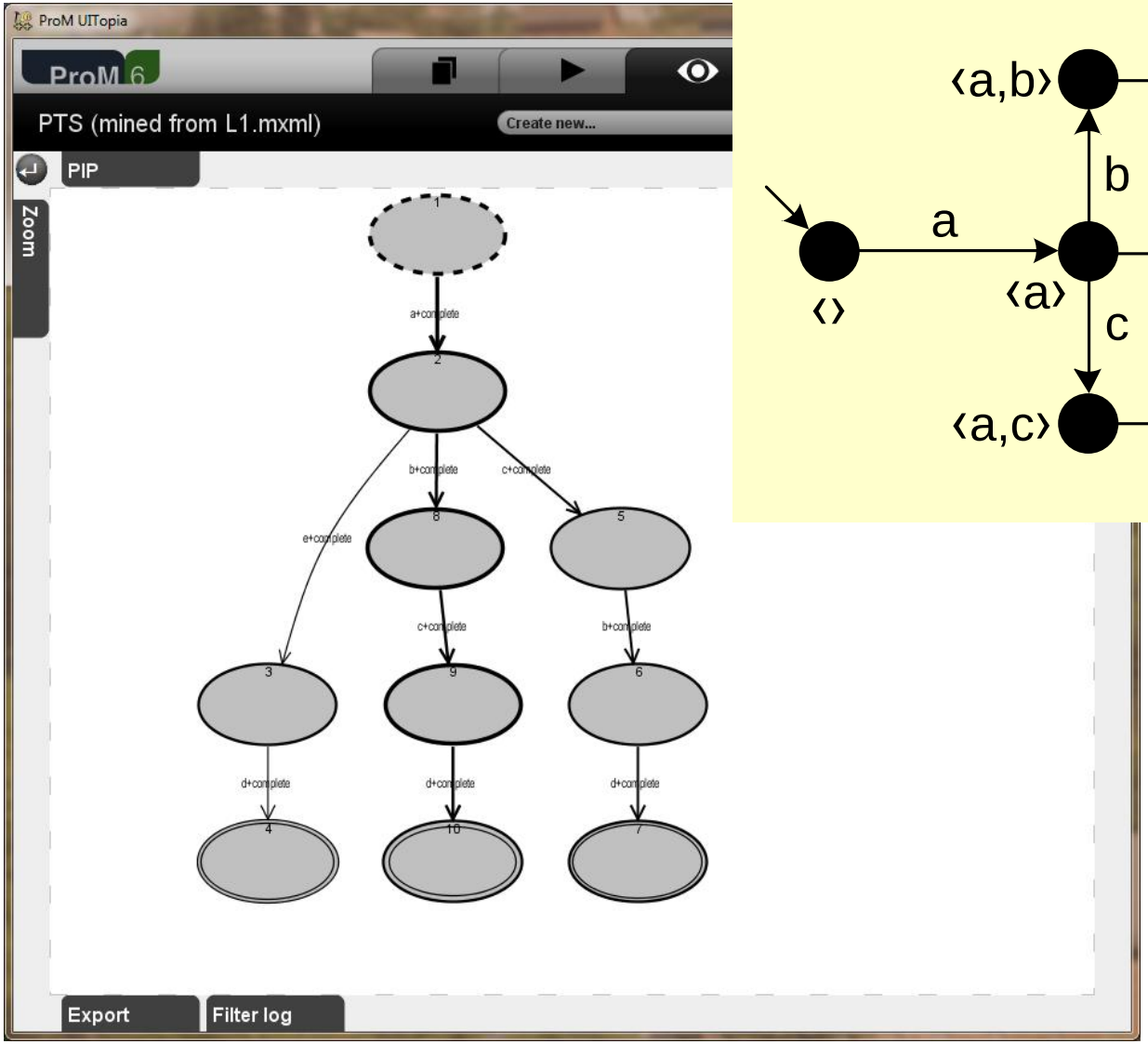


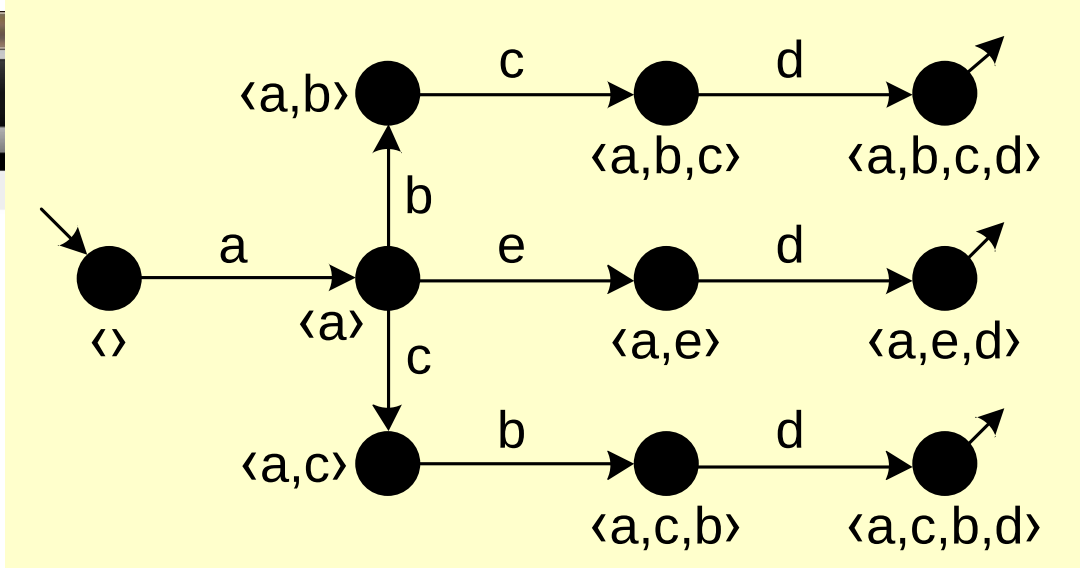


Check configuration

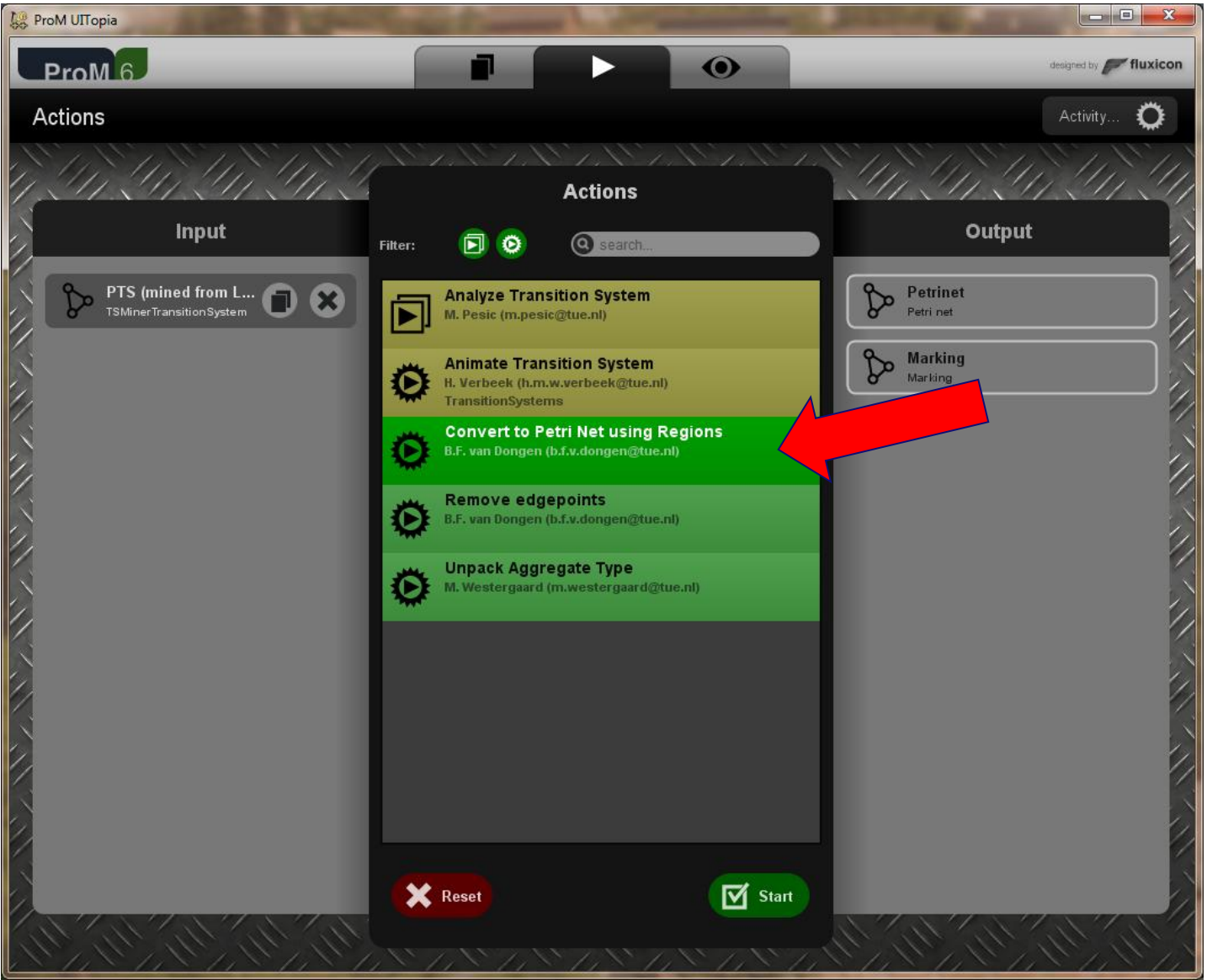


Resulting transition system

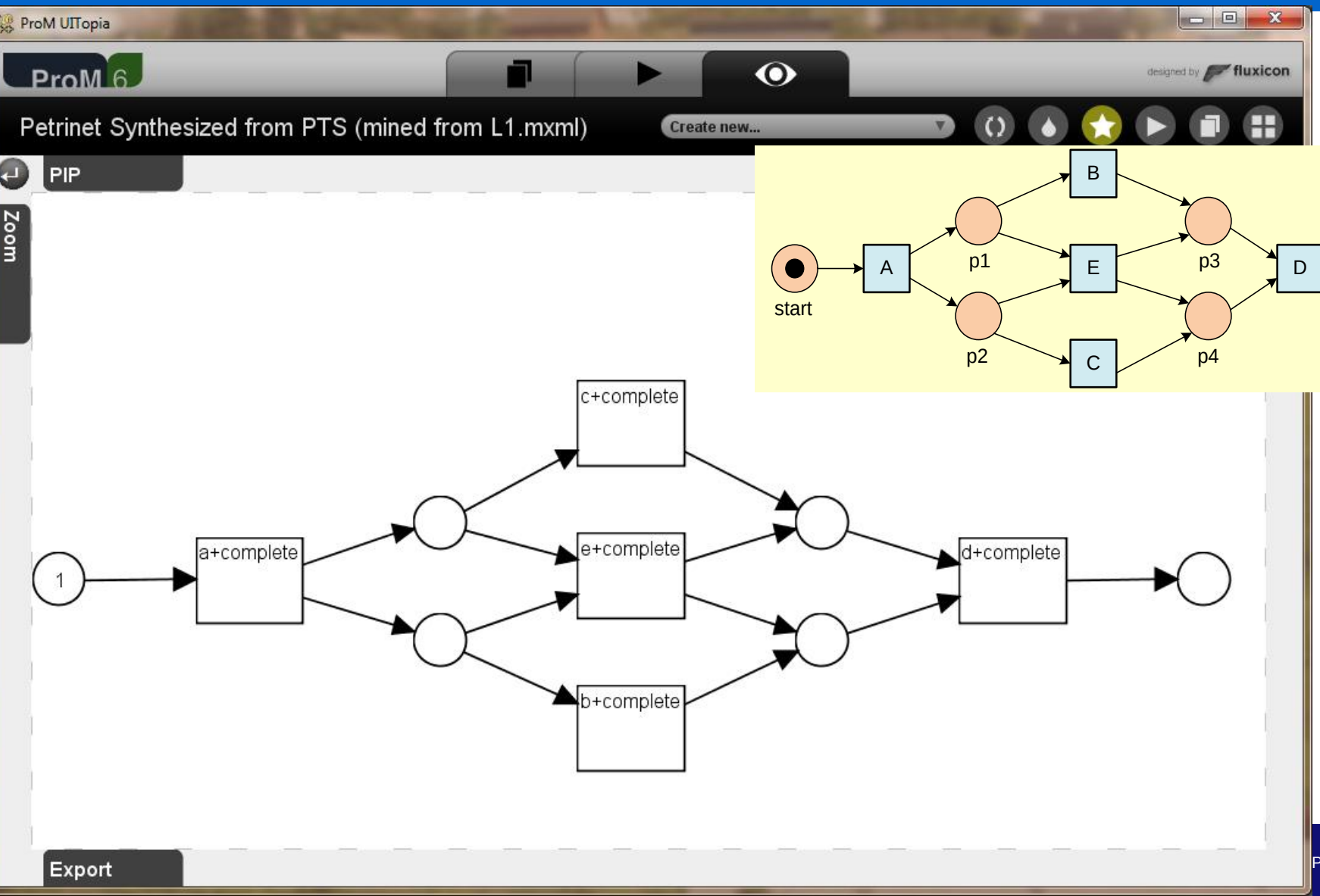


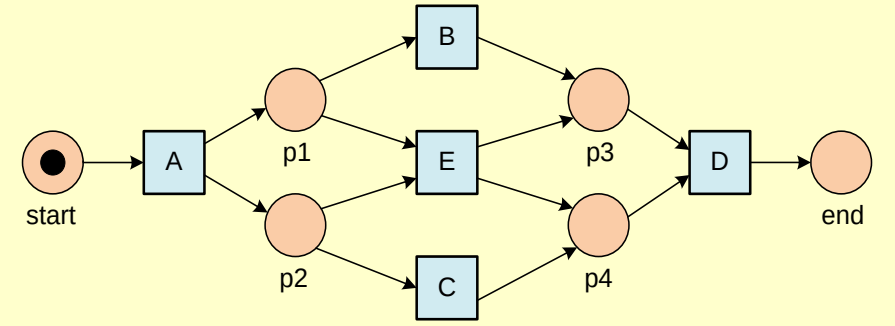


Convert transition system to Petri net



Resulting Petri net

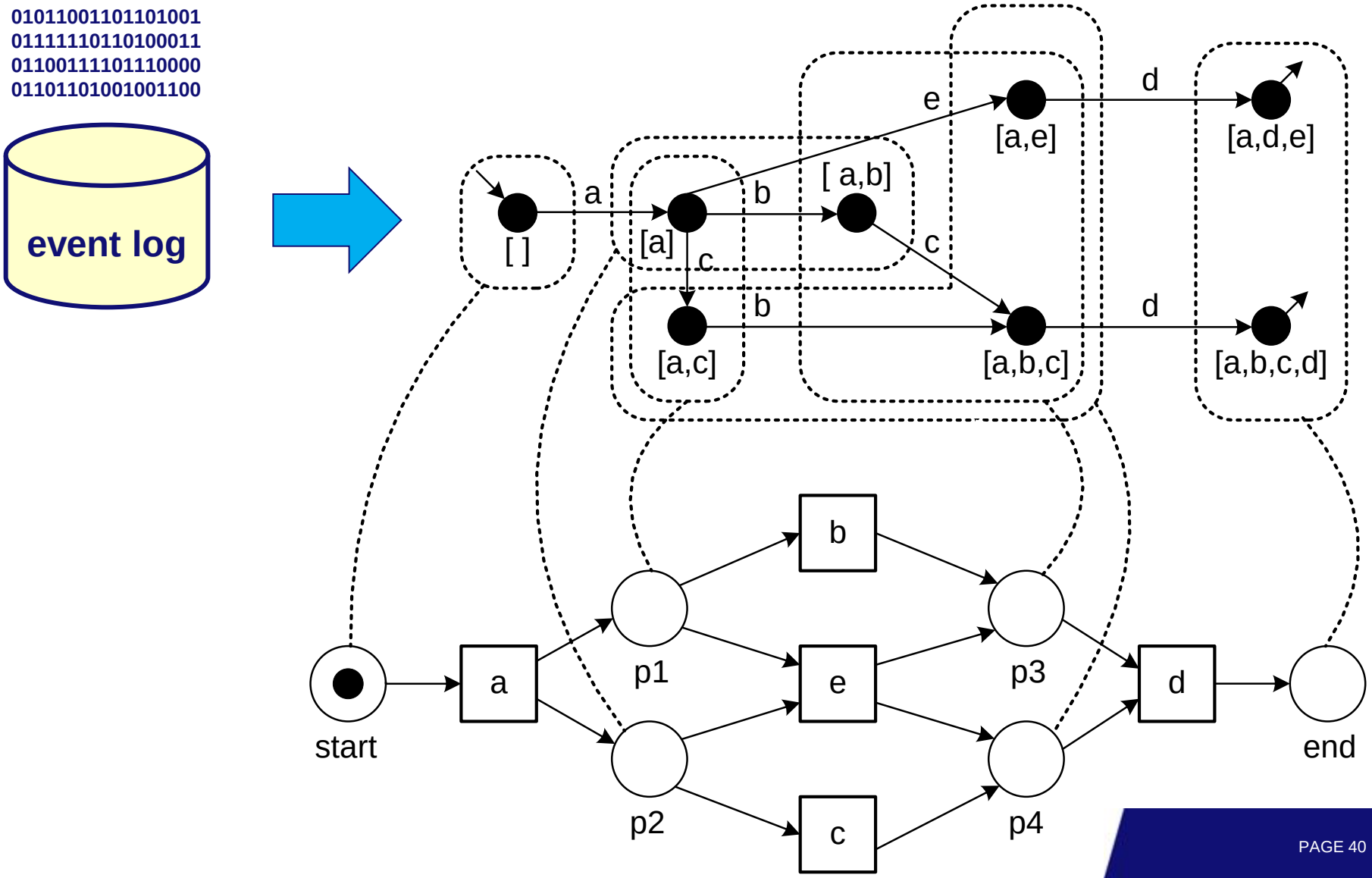




Summary

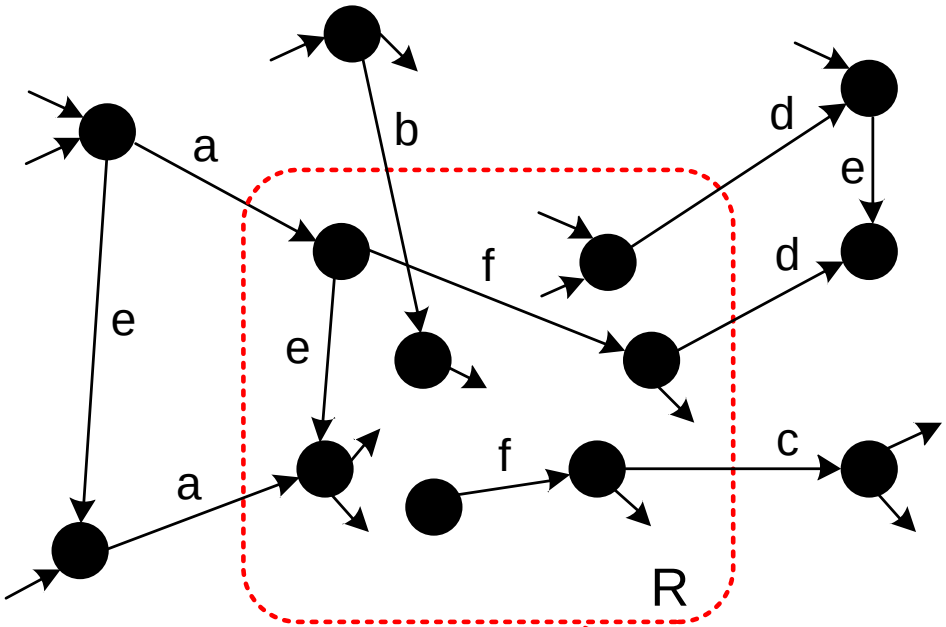
$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

01011001101101001
01111110110100011
01100111101110000
01101101001001100

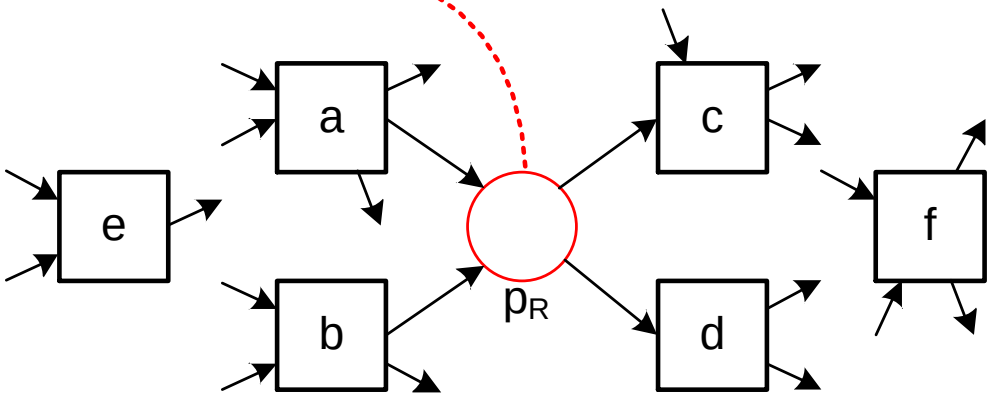


State-Based Regions

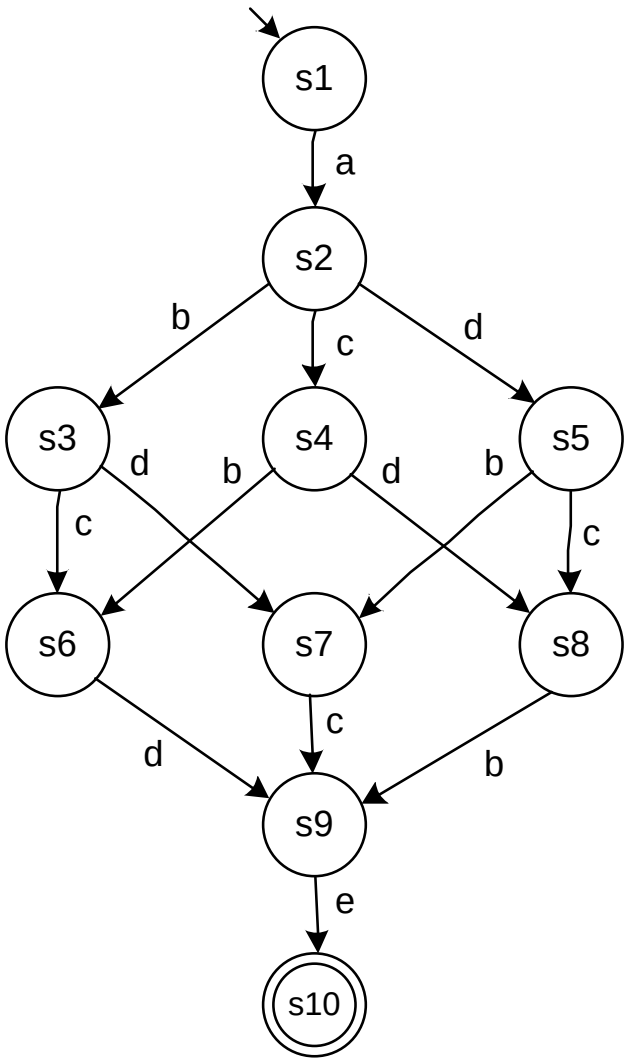
What is a (state-based) region?



a = enter
b = enter
c = exit
d = exit
e = do not cross
f = do not cross



Starting point: A Transition System

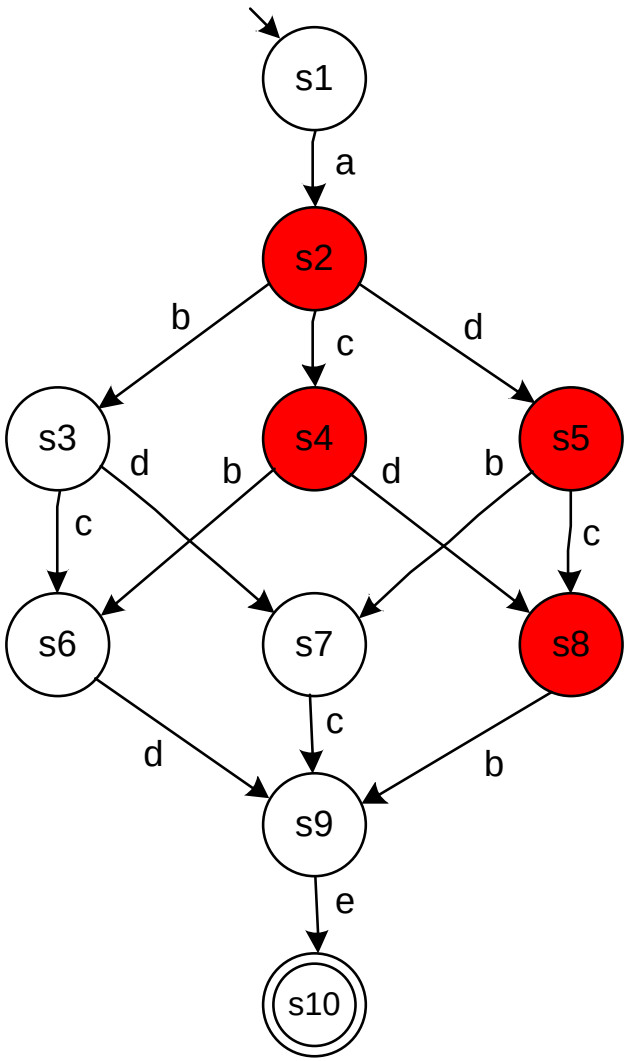


- **We assume that there is only one initial state (otherwise preprocessing needed).**
- **It is convenient to also have just one final state that can always be reached (not strictly necessary)**
- **All states need to be reachable!**

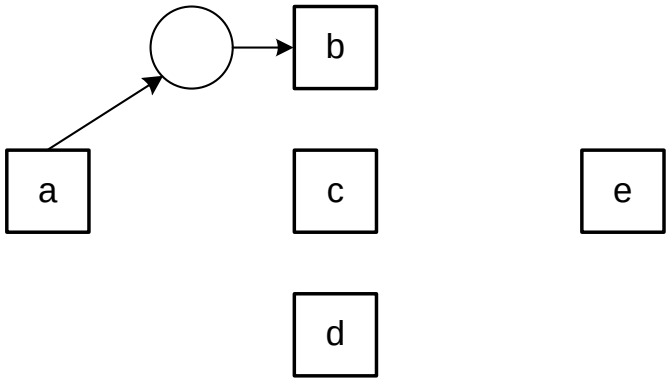
Definition

- **A region *r* is a set of states, such that for all transitions *(s_0, e, s_0')*, *(s_1, e, s_1')* *in the transition system* holds that:**
  - ***1)*** **$s_0 \in r$ and $s_0' \notin r$ implies that $s_1 \in r$ and $s_1' \notin r$**
  - ***2)*** **$s_0 \notin r$ and $s_0' \in r$ implies that $s_1 \notin r$ and $s_1' \in r$**
- **In words: A region is a set of states, such that, if a transition *exits* the region, then all equally labeled transitions *exit* the region, and if a transition *enters* the region, then all equally labeled transitions *enter* the region. All events not entering or exiting the region *do not cross* the region.**

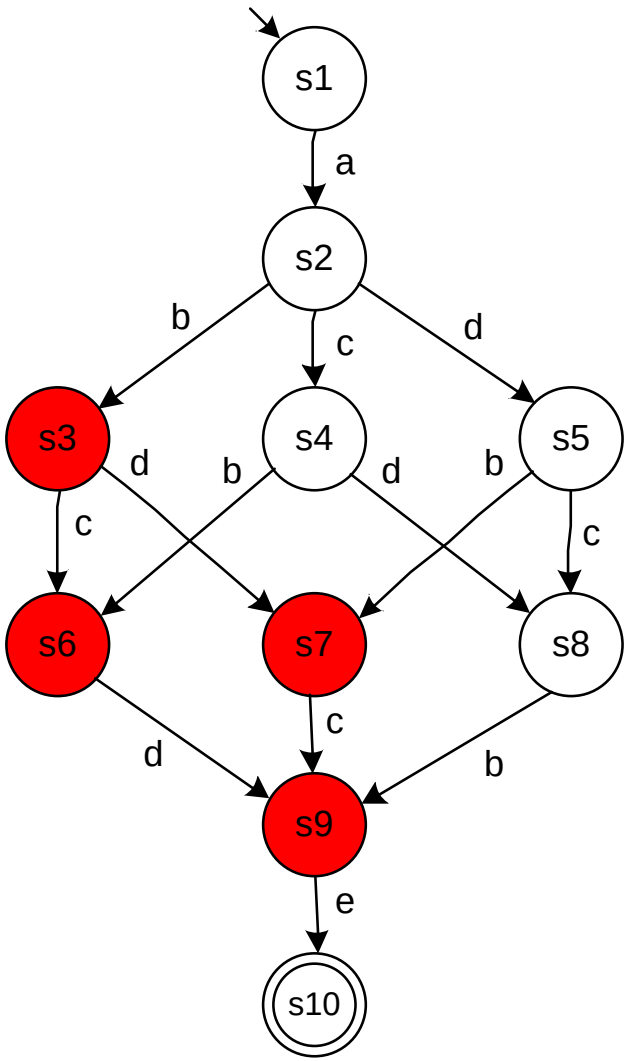
Example of a region



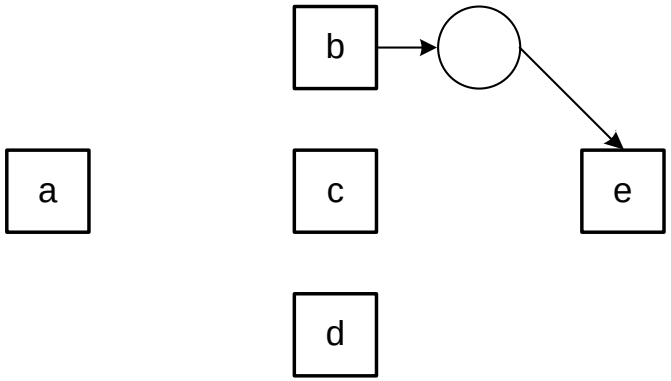
- a enters
- b exits
- c does not cross
- d does not cross
- e does not cross



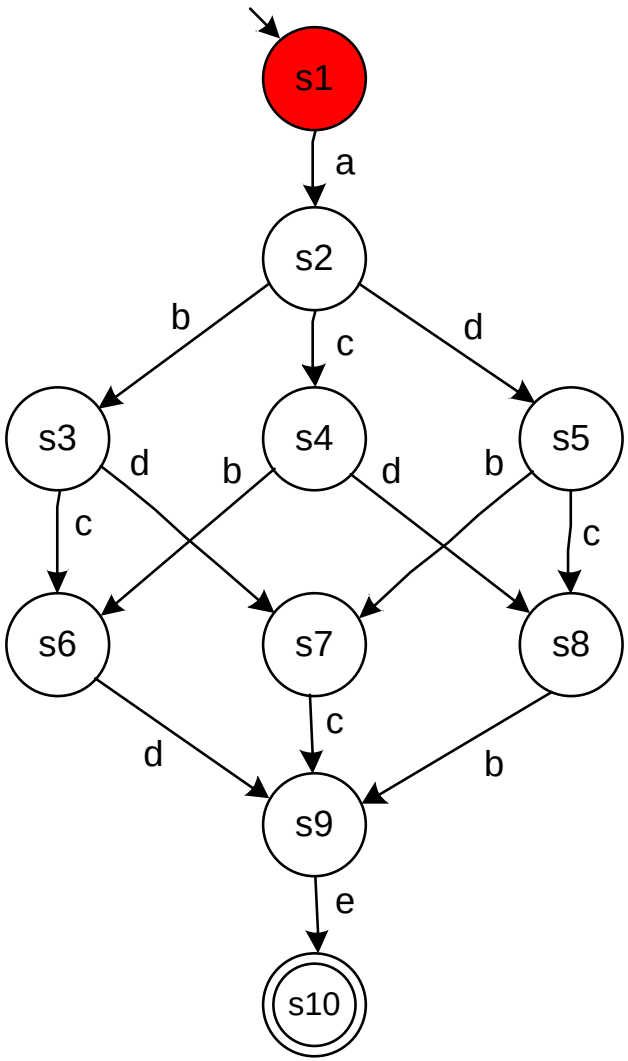
Example of a region



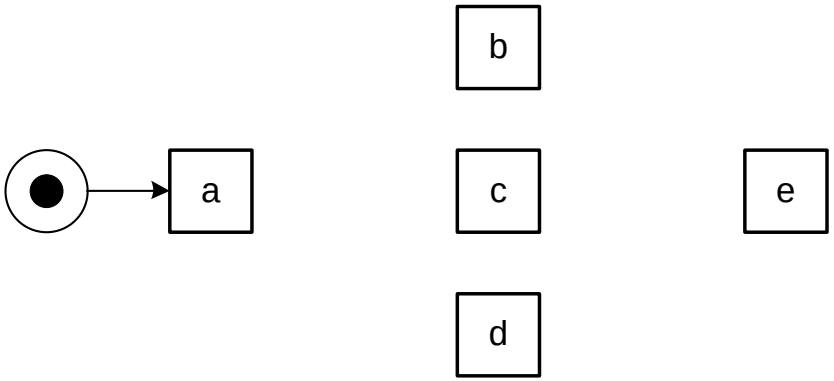
- a does not cross
- b enters
- c does not cross
- d does not cross
- e exits



Example of a region

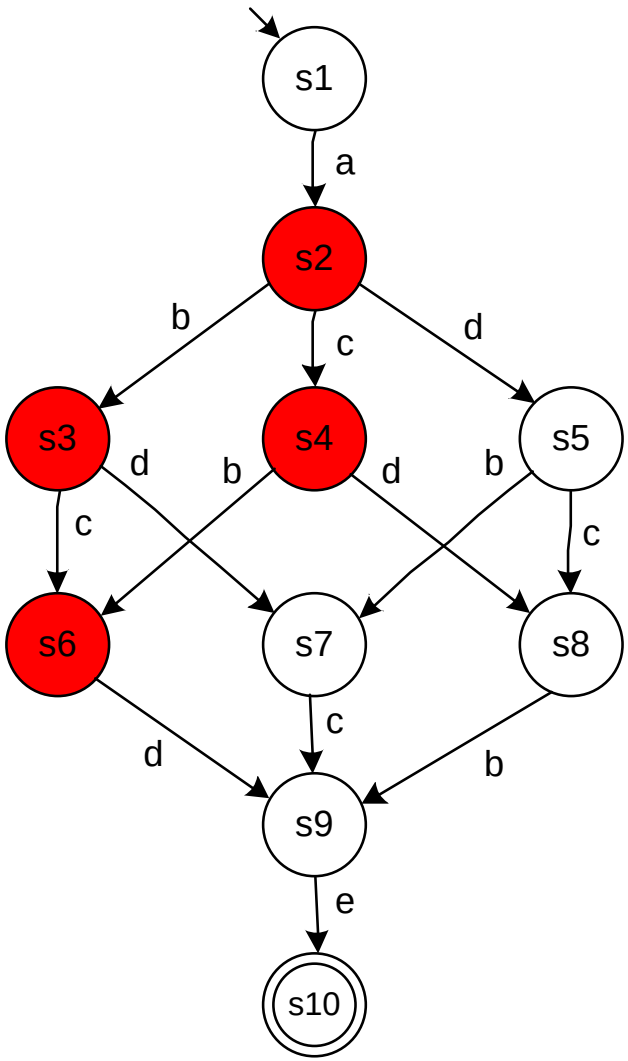


- a exits
- b does not cross
- c does not cross
- d does not cross
- e does not cross

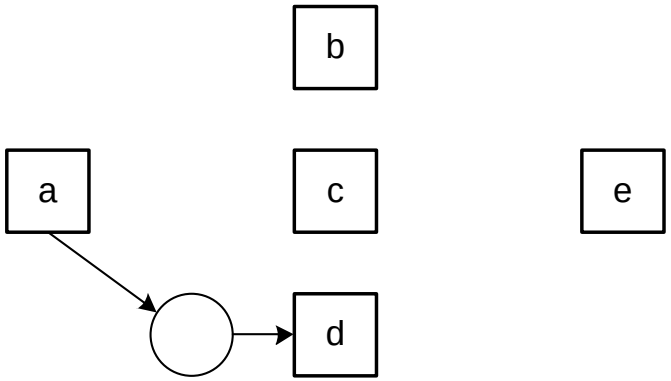


Places corresponding to regions containing the initial state are initially marked.

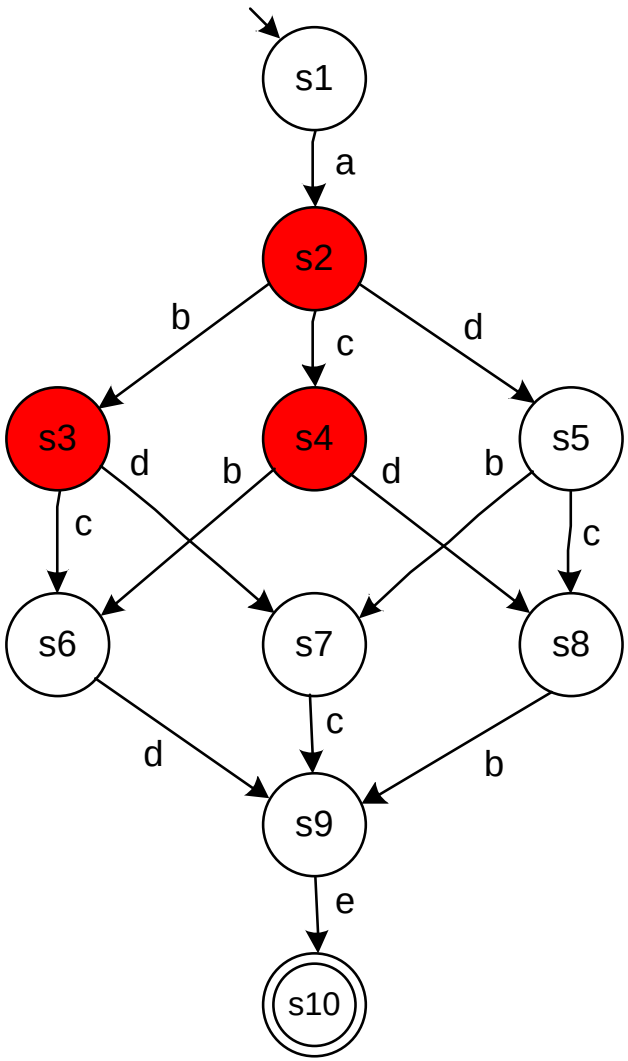
Example of a region



- a enters
- b does not cross
- c does not cross
- d exits
- e does not cross

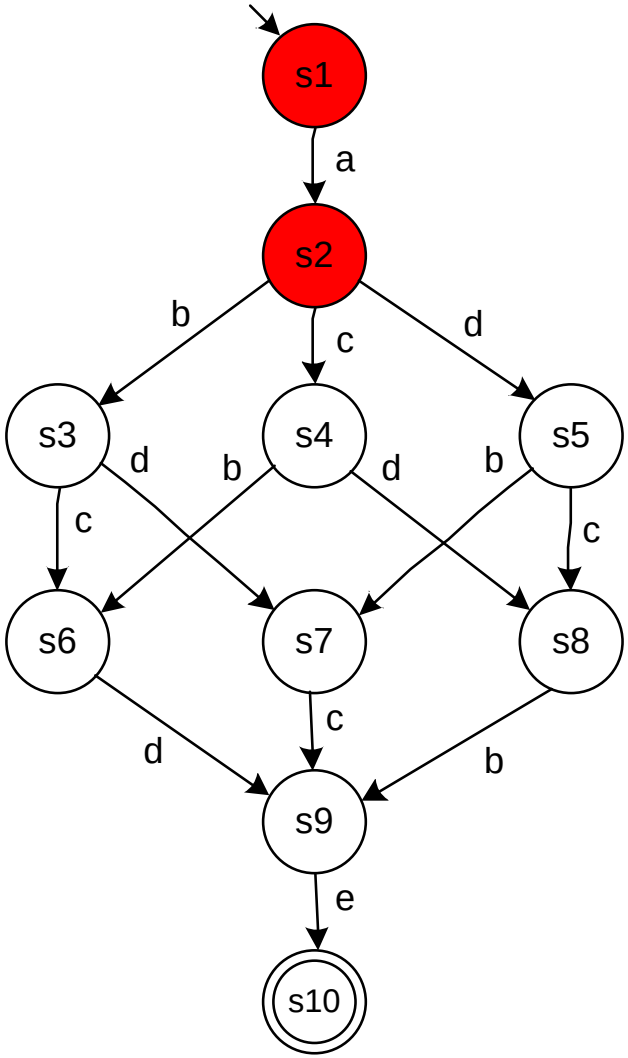


Not a region

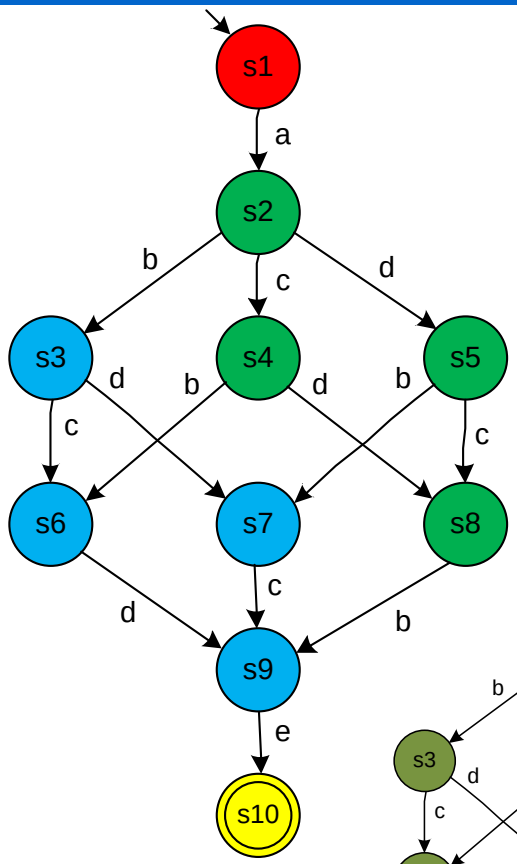


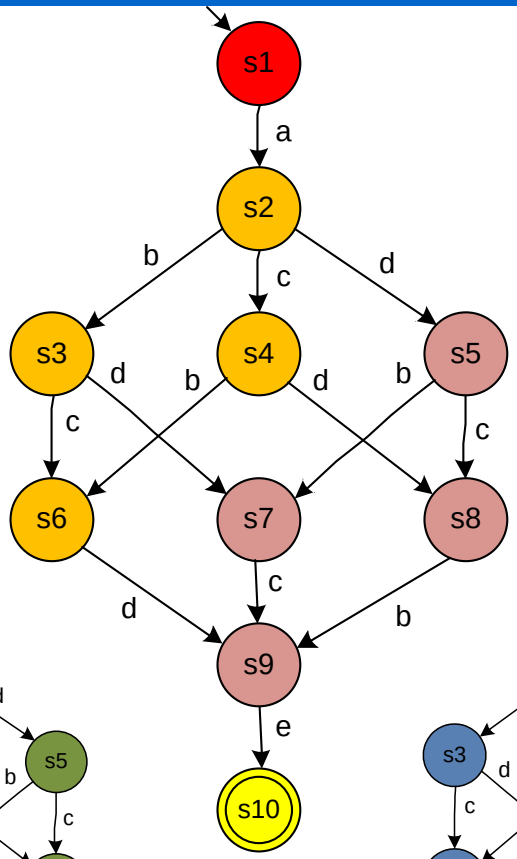
- a enters
- b does not cross and exits
- c does not cross and exits
- d does not cross and exits
- e does not cross

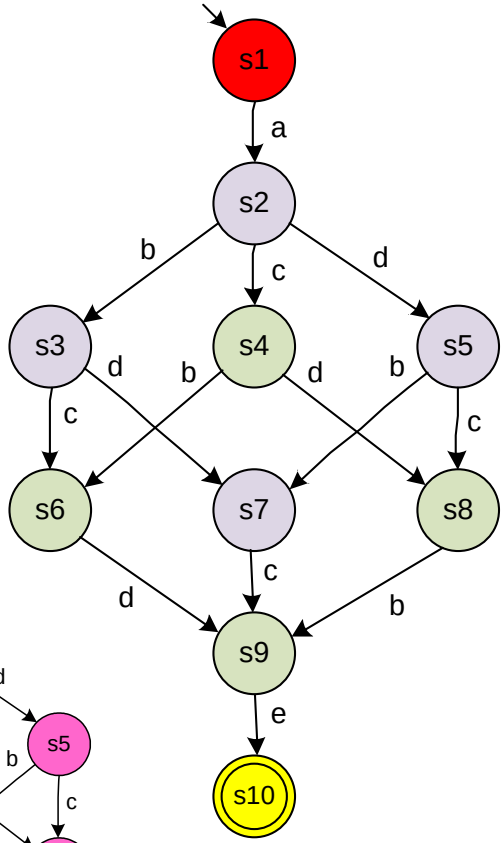
Not a region

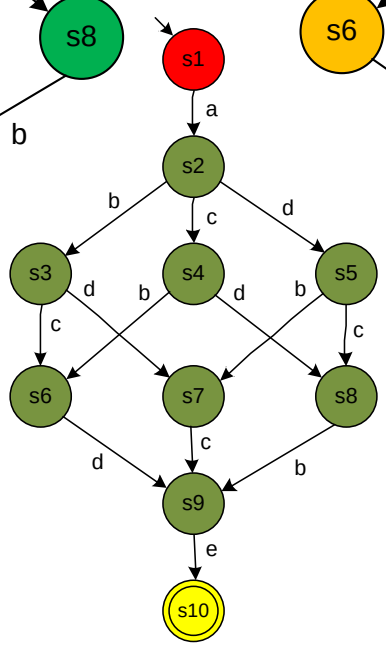


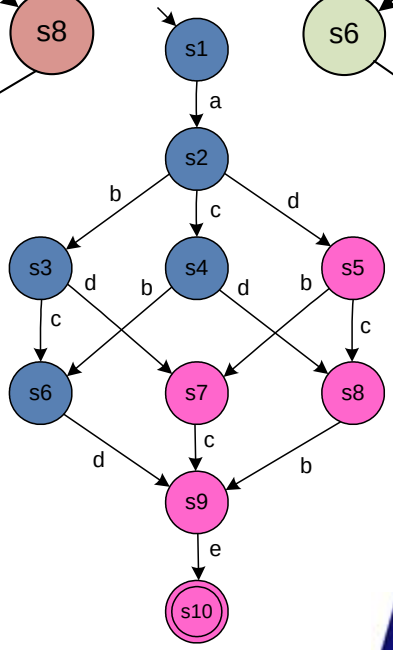
- a does not cross
- b does not cross and exits
- c does not cross and exits
- d does not cross and exits
- e does not cross





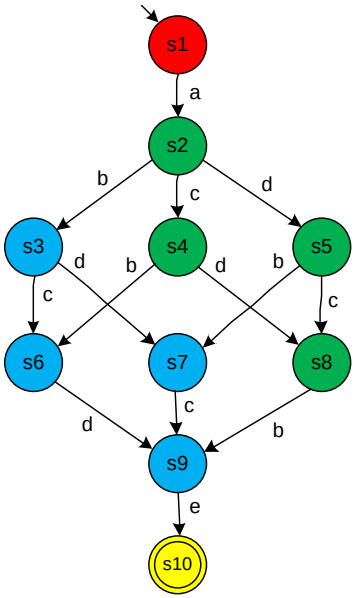


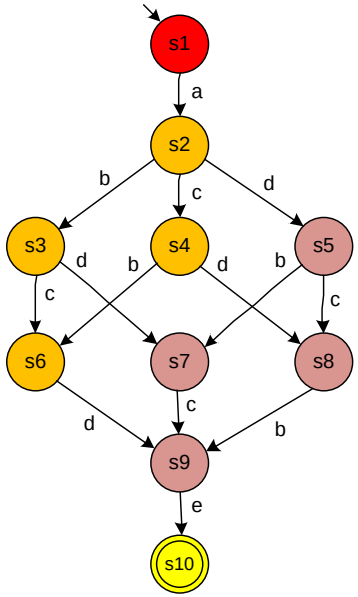


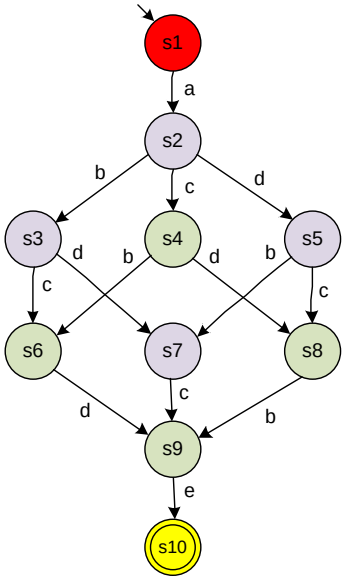


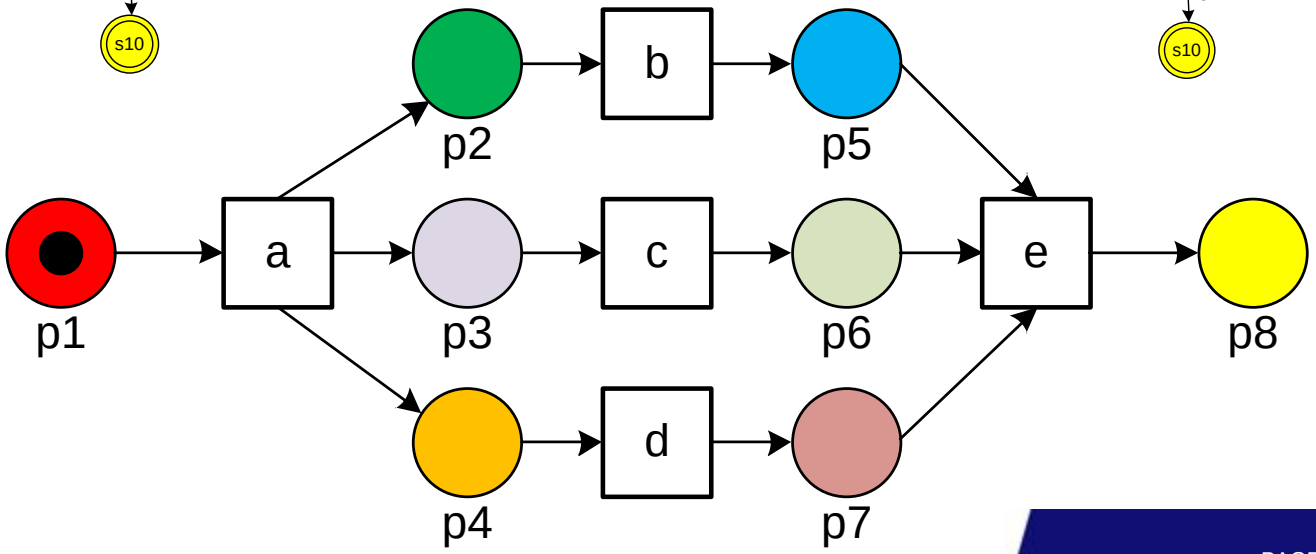
Multiple regions

Etc.







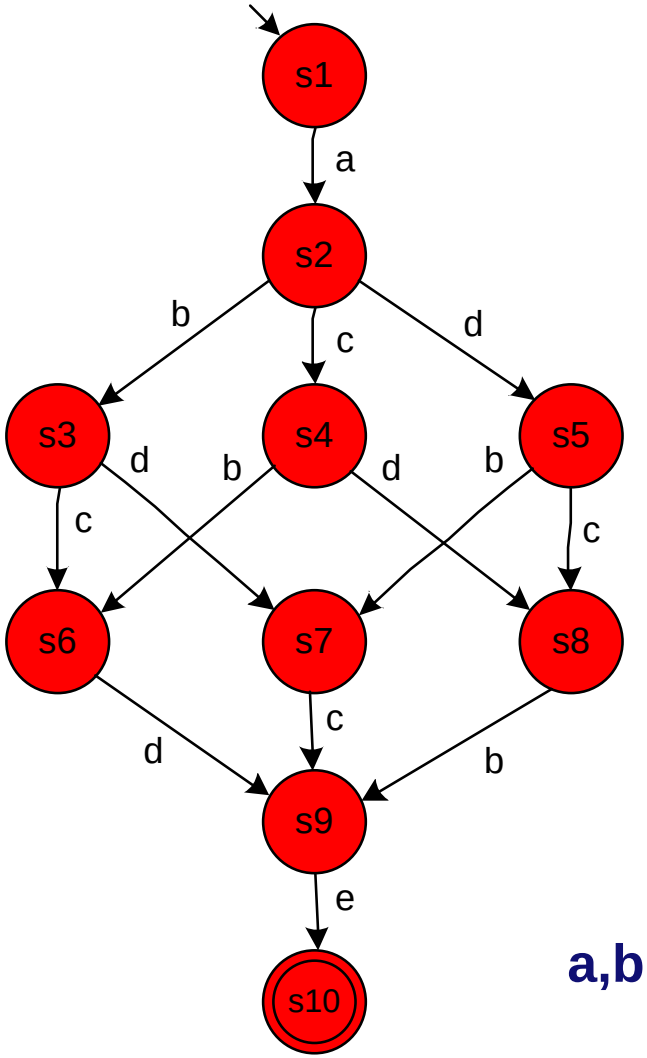


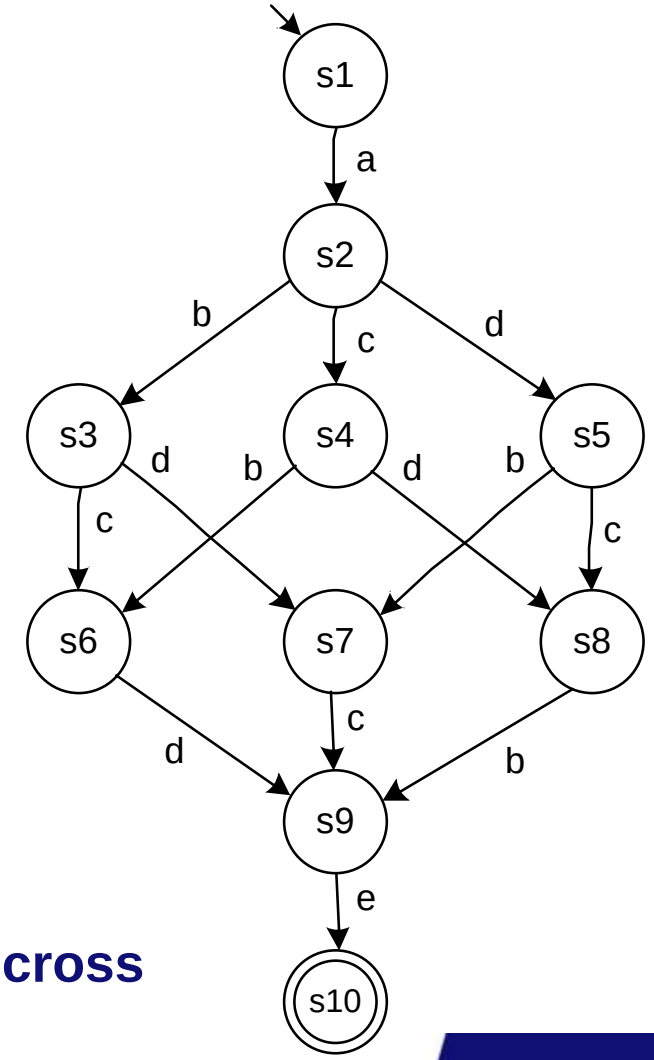
Selectively chosen regions …

Regions – Region Properties

- **Let S be the set of all states of a transition system.**
- **Trivial Regions:** **Both *S* and ∅ are called the trivial regions,**
- **Complements:** **If *r* is a region, then *S*\ *r* is a region,**
- **Pre-/Post-regions:** **If event *e* exits (enters) a region *r*, then *r* is a pre- (post-)region of *e*,**
- **Minimal regions:** **If r_0 and r_1 are regions, and r_0 is a subset of r_1, then $r_1 \setminus r_0$ is a region.**
- **The latter implies the existence of (non-trivial) *minimal regions*.**

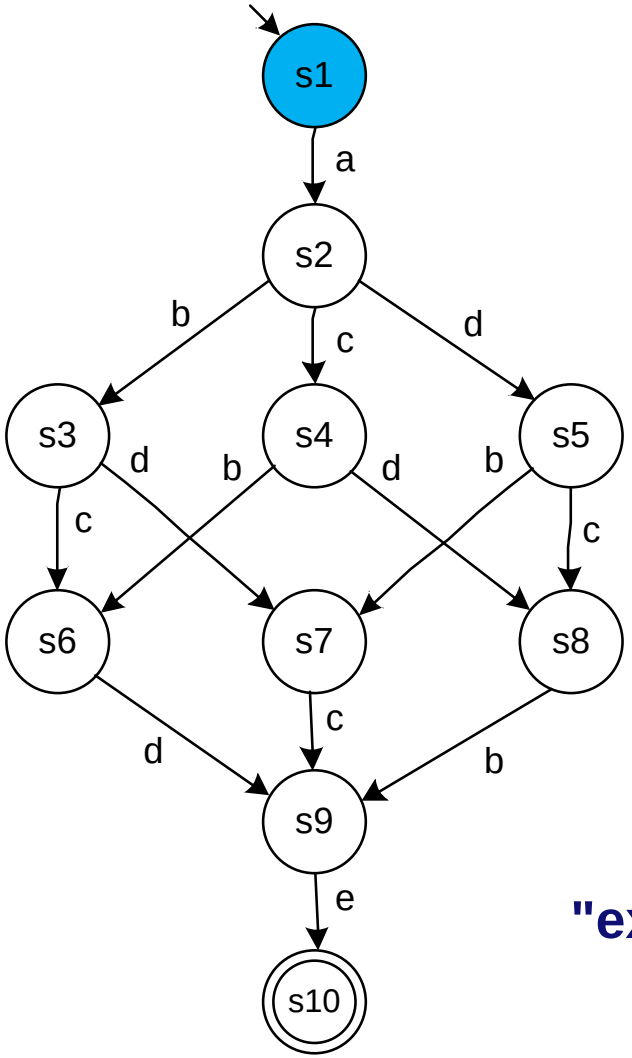
Trivial regions



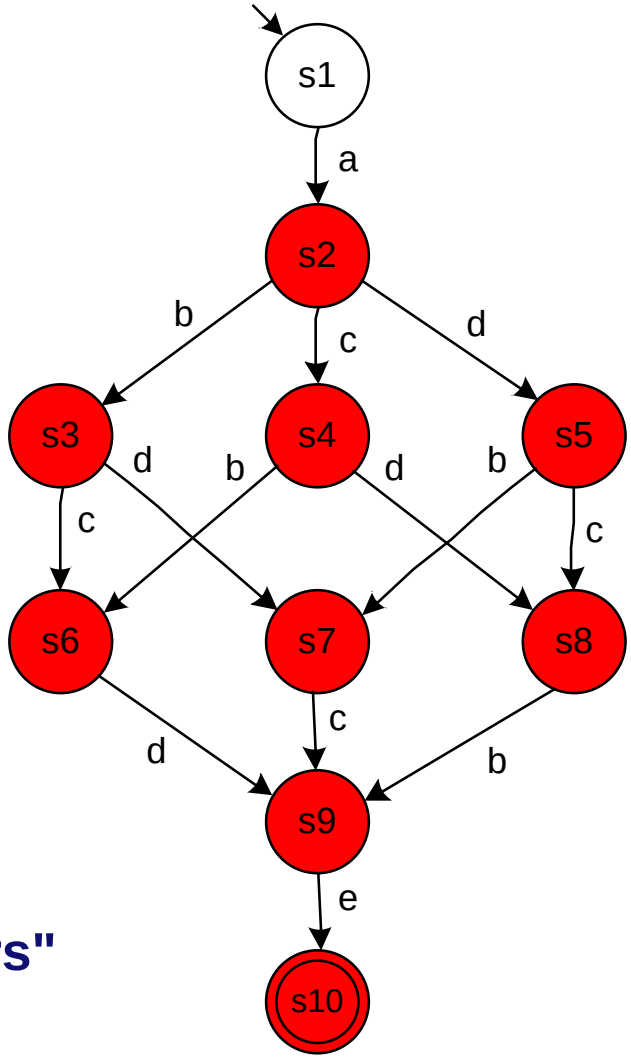


a,b,c,d,e do not cross

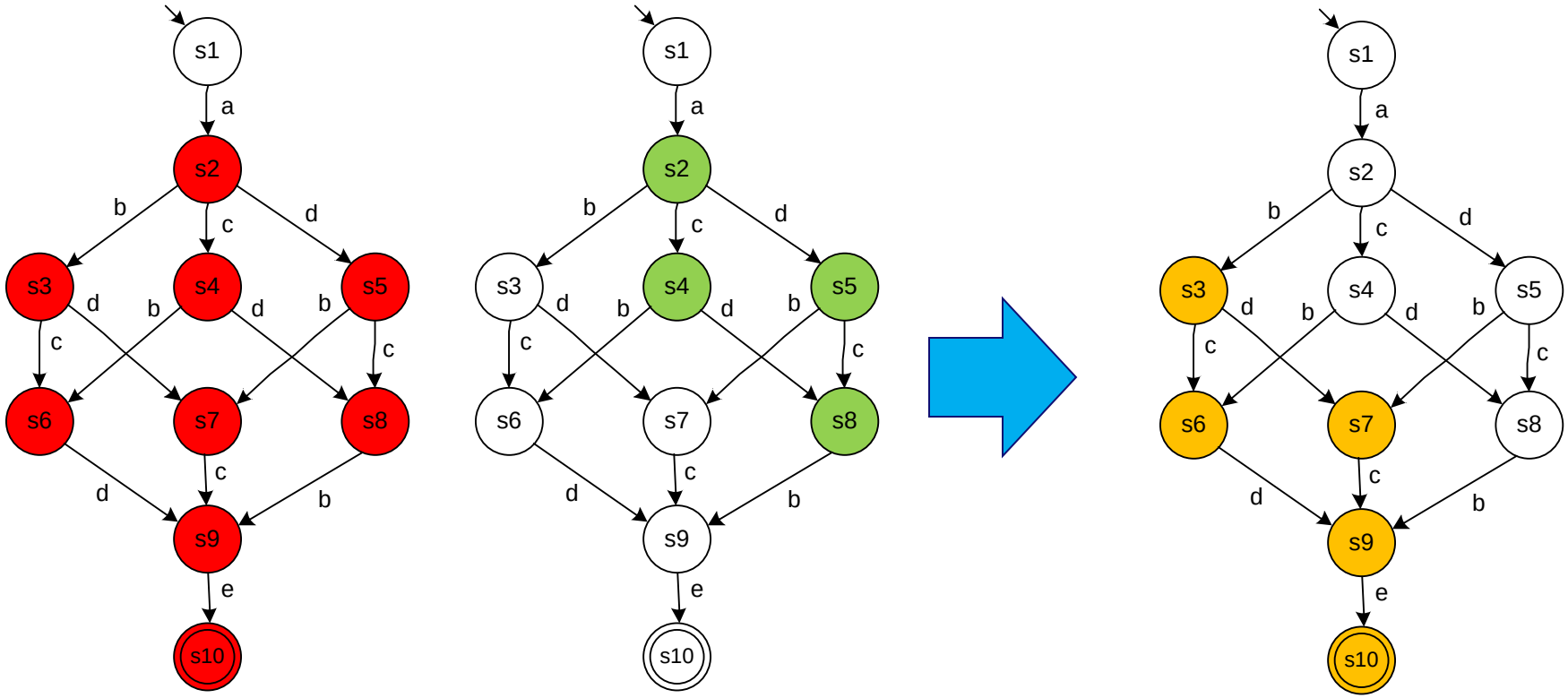
Complement:
If *r* is a region, then *S*\ *r* is a region



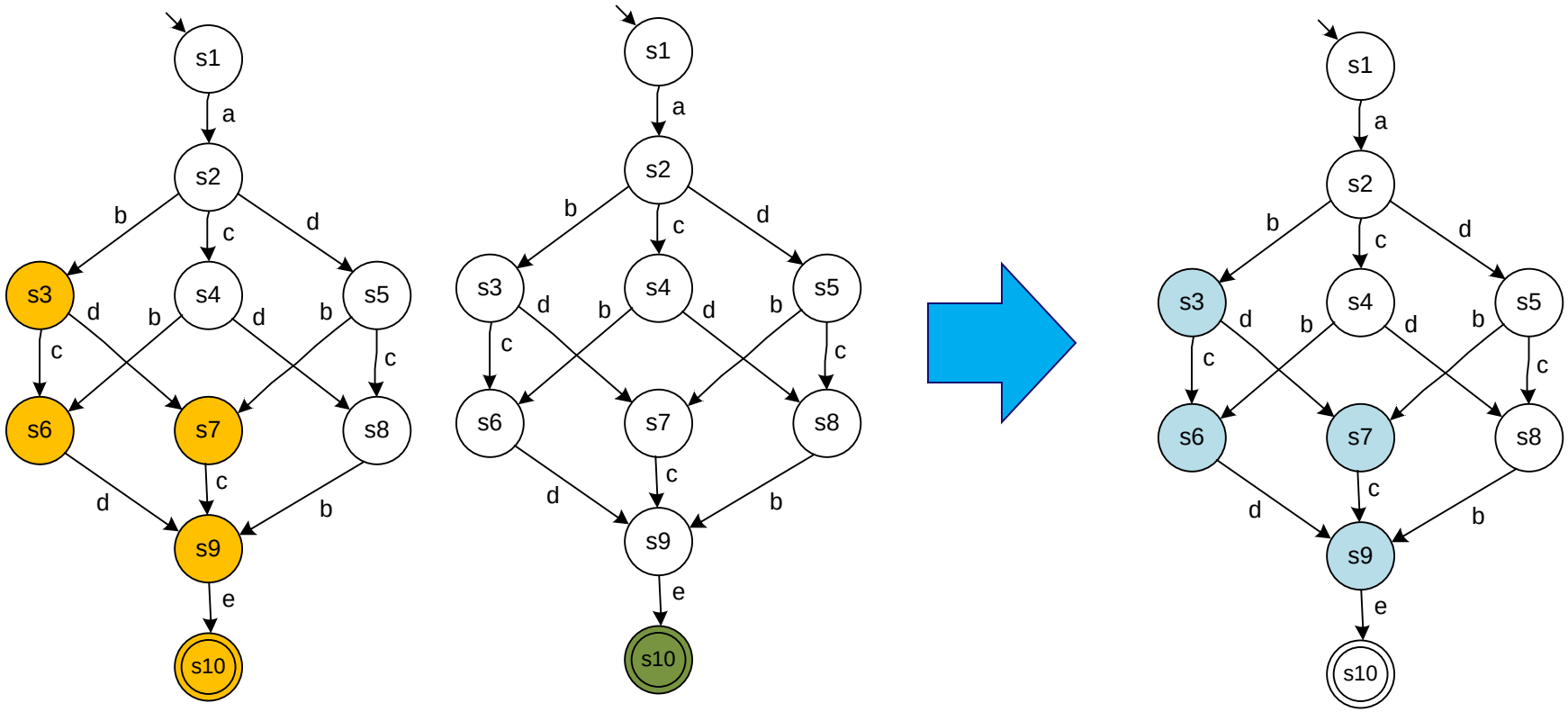
"exits" and "enters" are swapped



If r_0 and r_1 are regions, and r_0 is a subset of r_1, then $r_1 \setminus r_0$ is a region.

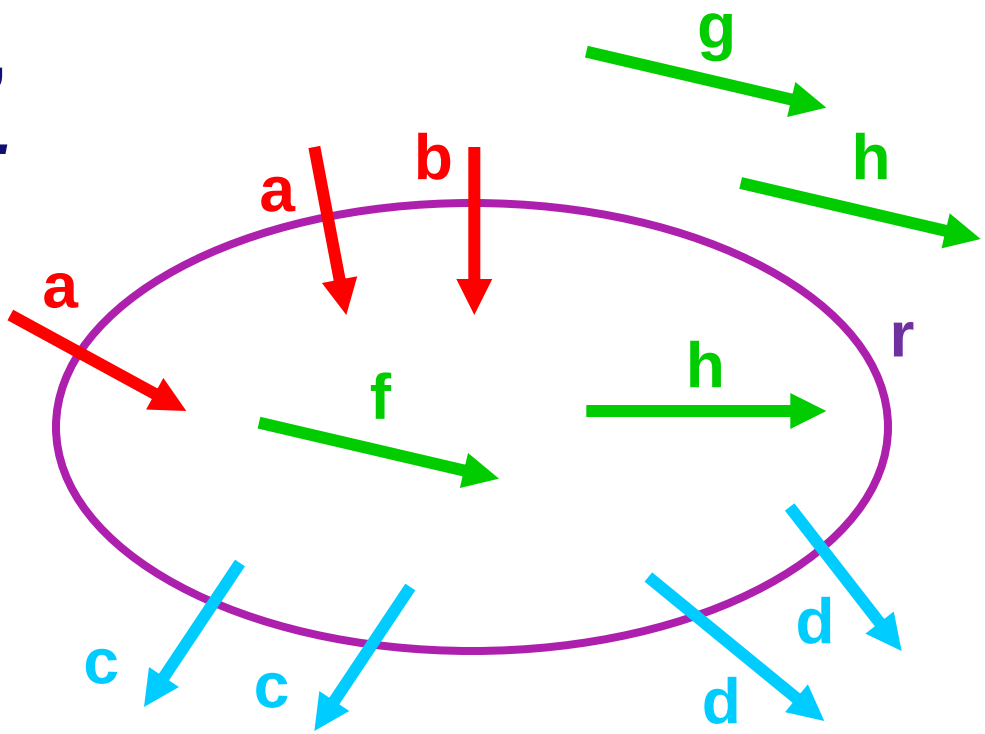


Not minimal yet …



Pre and post regions

- **If event *e* enters a region *r*, then *r* is a post-region of *e*.**
  - **r is post-region of a**
  - **r is post-region of b**



- **If event *e* exits a region *r*, then *r* is a pre-region of *e*.**
  - **r is pre-region of c**
  - **r is pre-region of d**

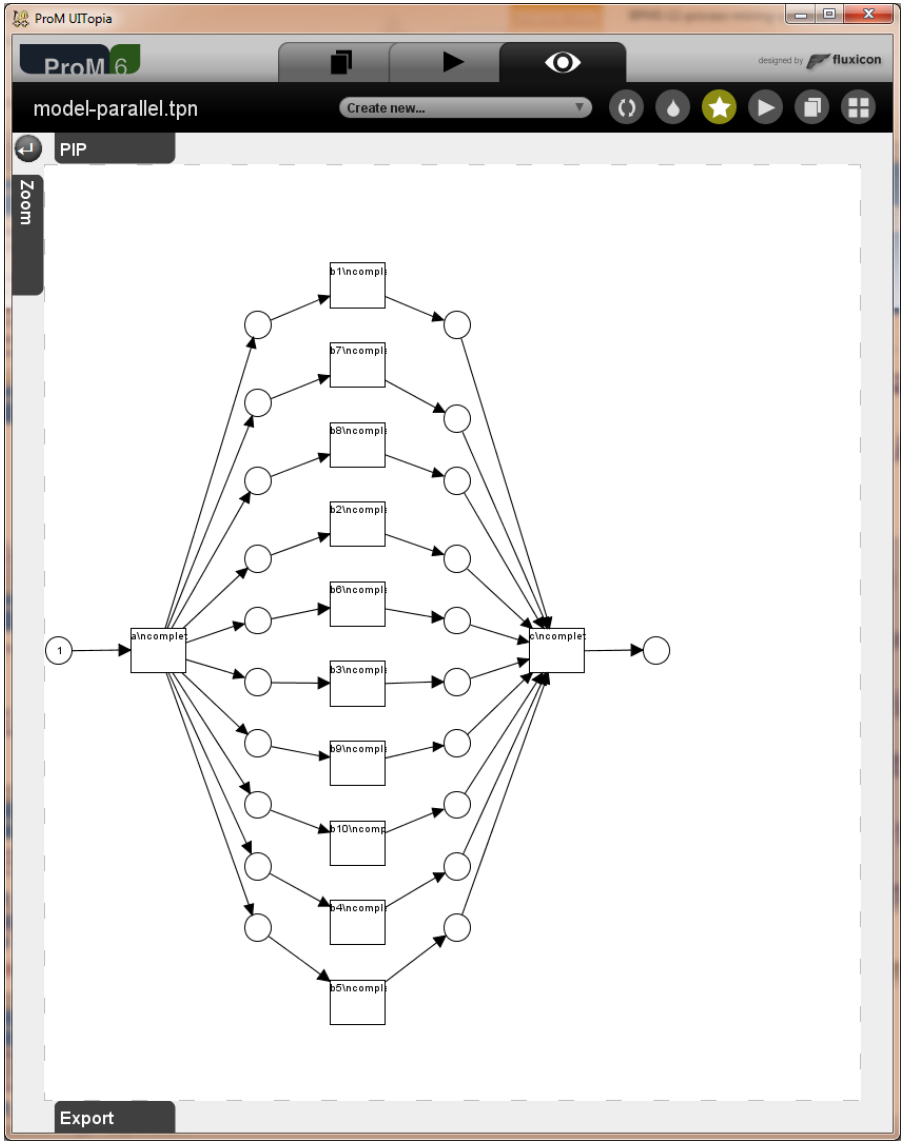
- pre(e) is the set of all (minimal) pre-regions or e.
- pre(e) is the set of all (minimal) pre-regions or e.
- Both are sets of sets!

Basic algorithm to construct a Petri net

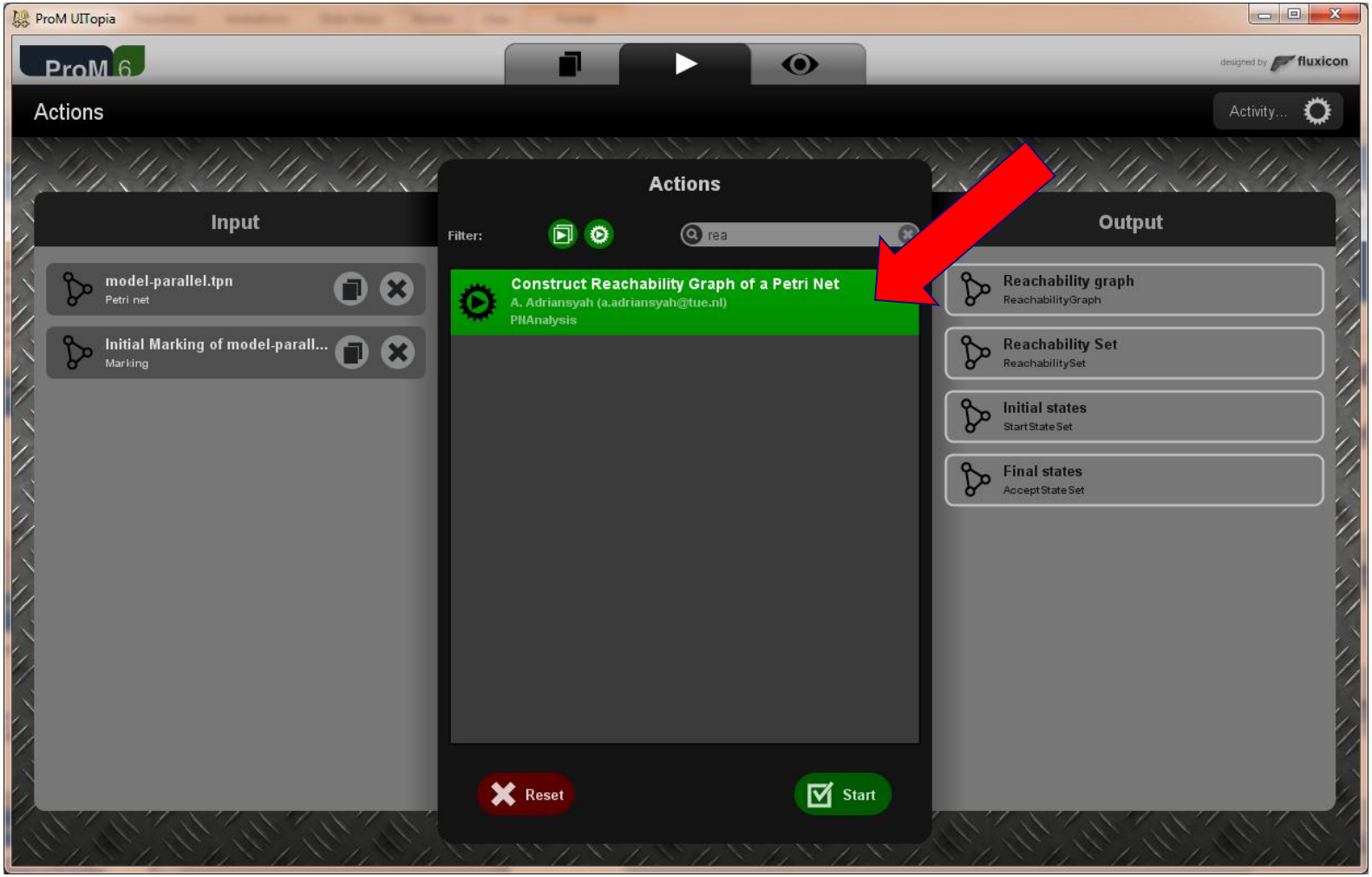
- For each event in the transition system, a transition is generated in the Petri net.
- Compute the minimal non-trivial regions.
- For each minimal non-trivial in the transition system, a place is generated in the Petri net.
- Add corresponding arcs (post-regions are output places and pre-regions are input places).
- A token is added to each place that corresponds to a region containing the initial state.

The resulting Petri net is called the minimal saturated net.

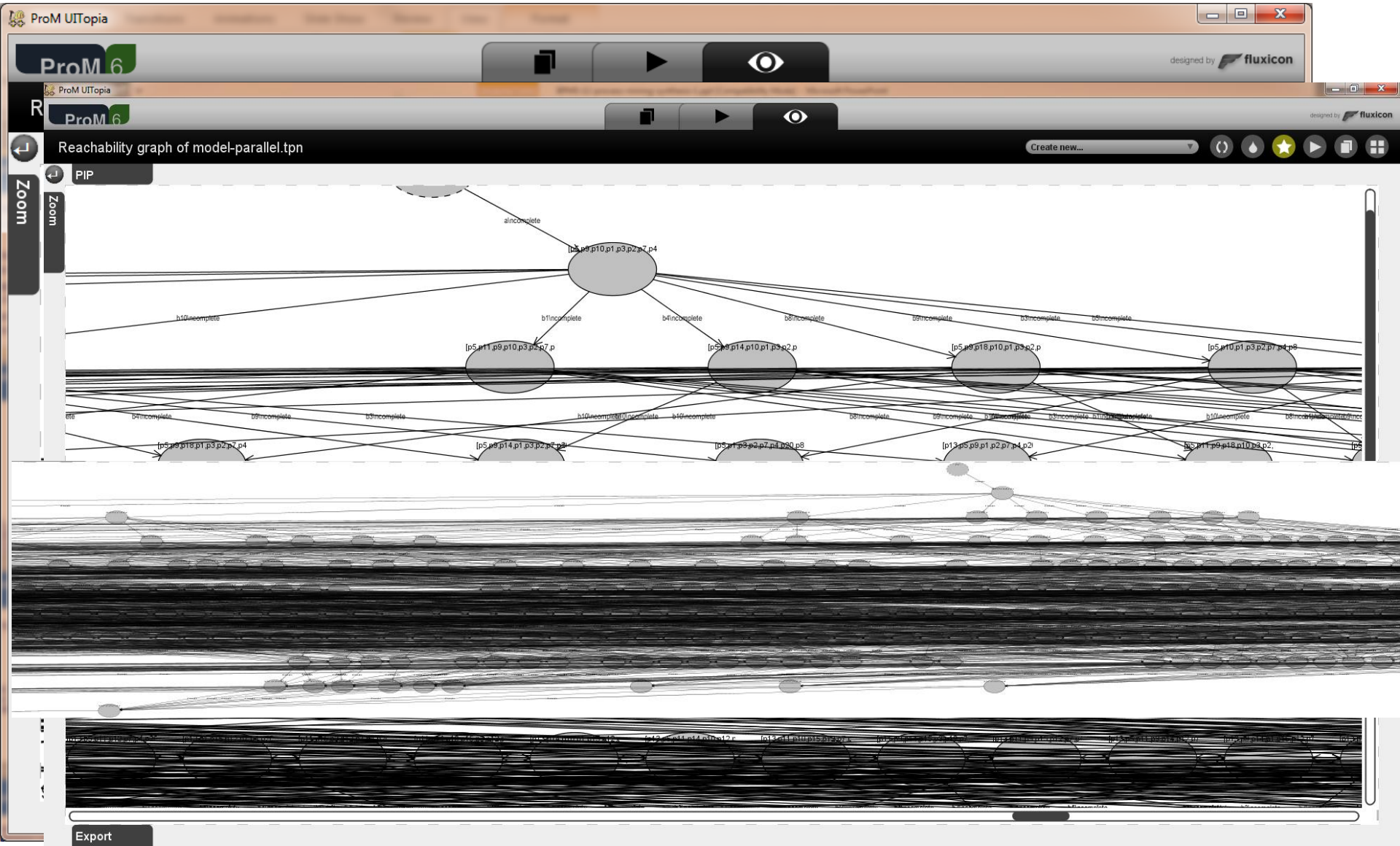
Load Petri net with 10 parallel activities



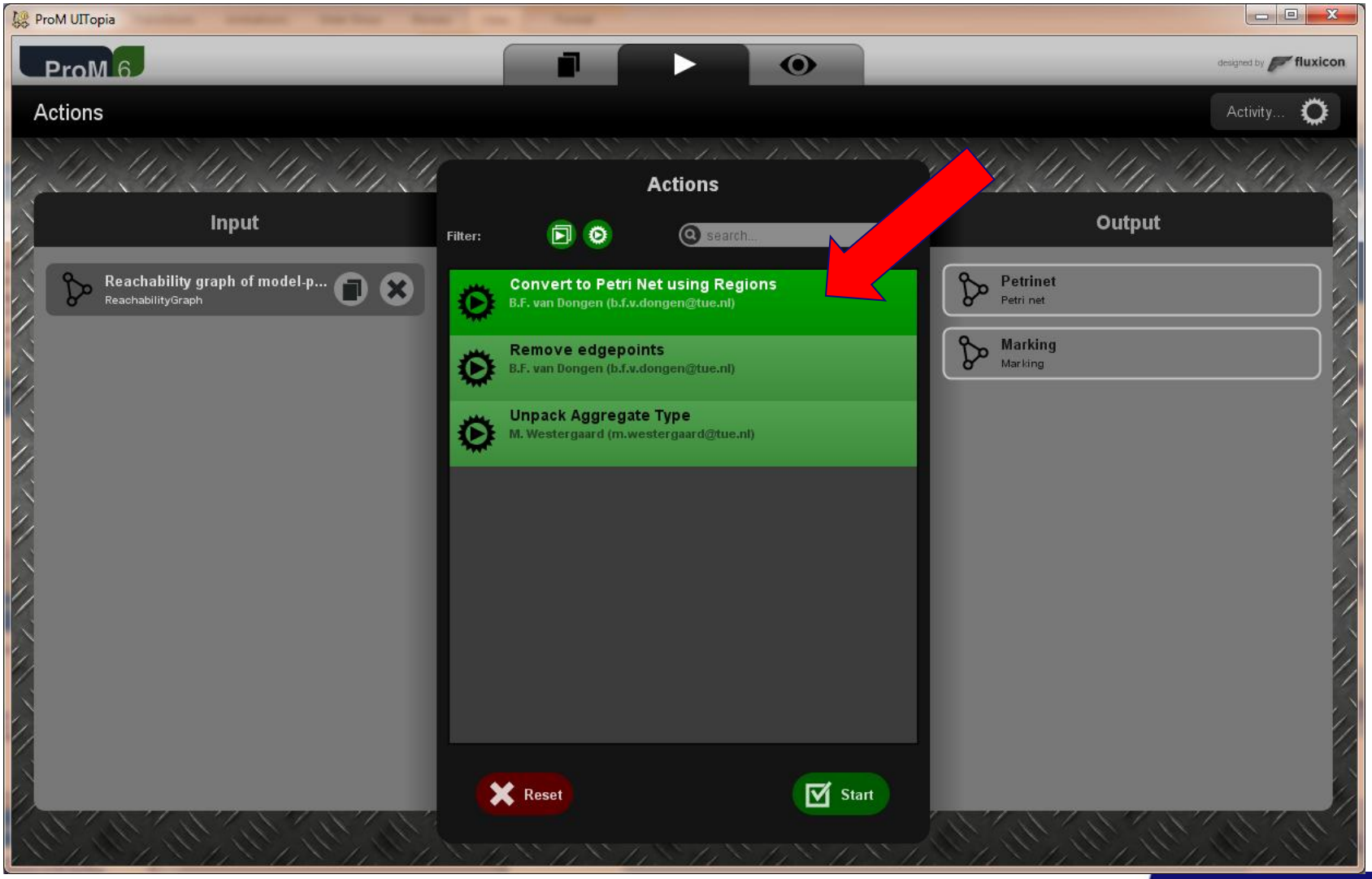
Construct reachability graph



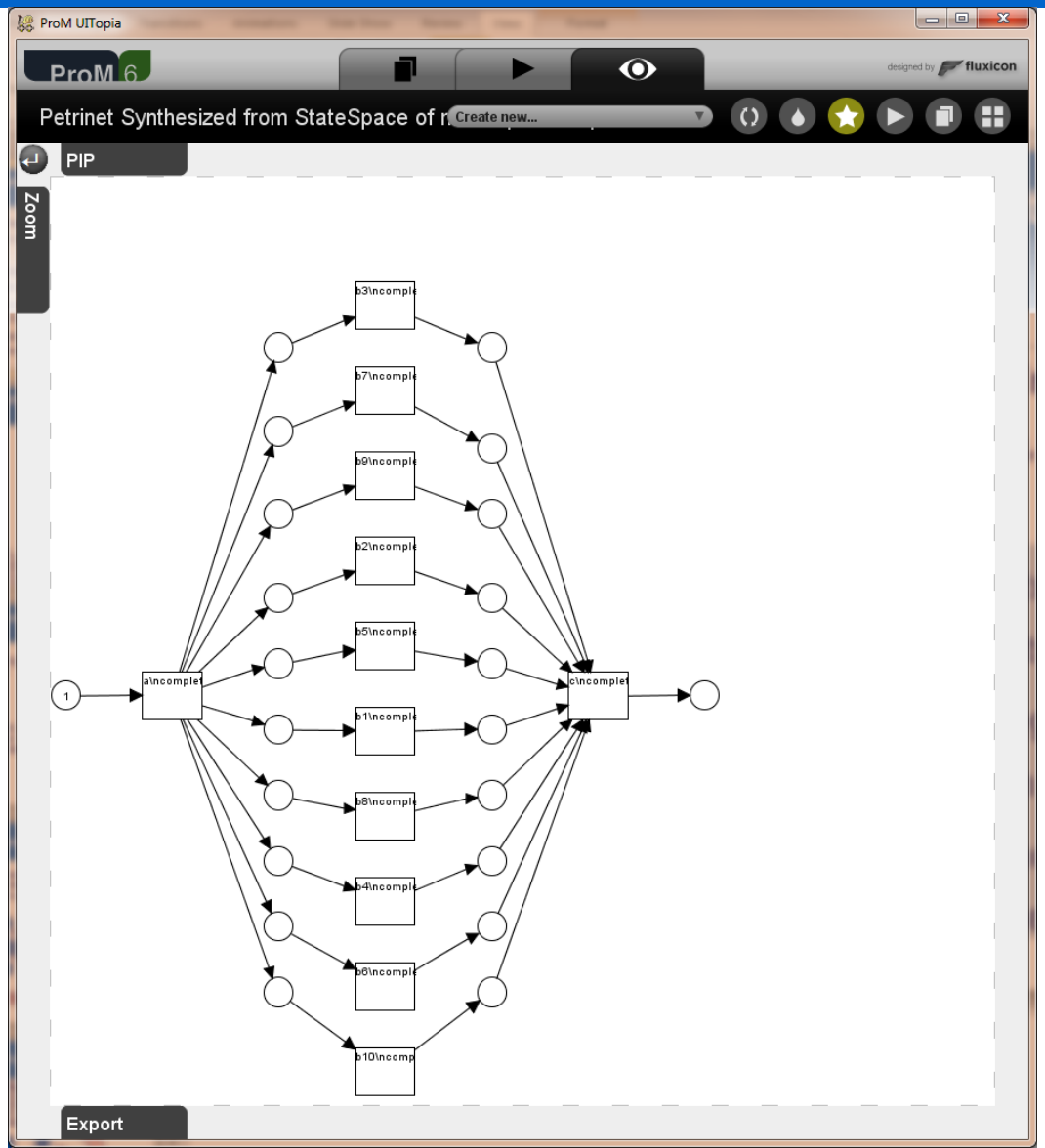
Reachability graph ($1+2^{10}+1$ =1026 states)



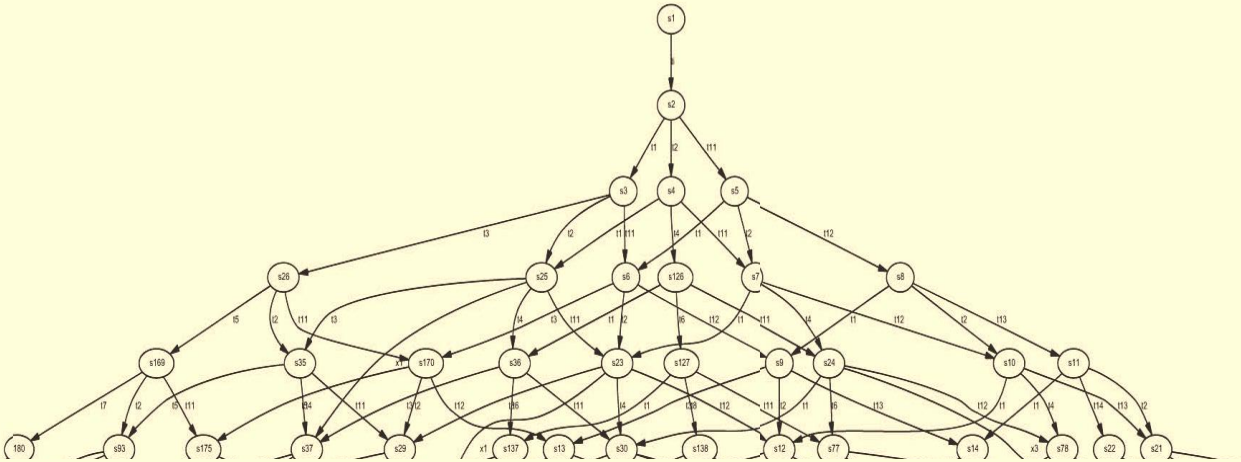
Apply state-based regions to fold state space

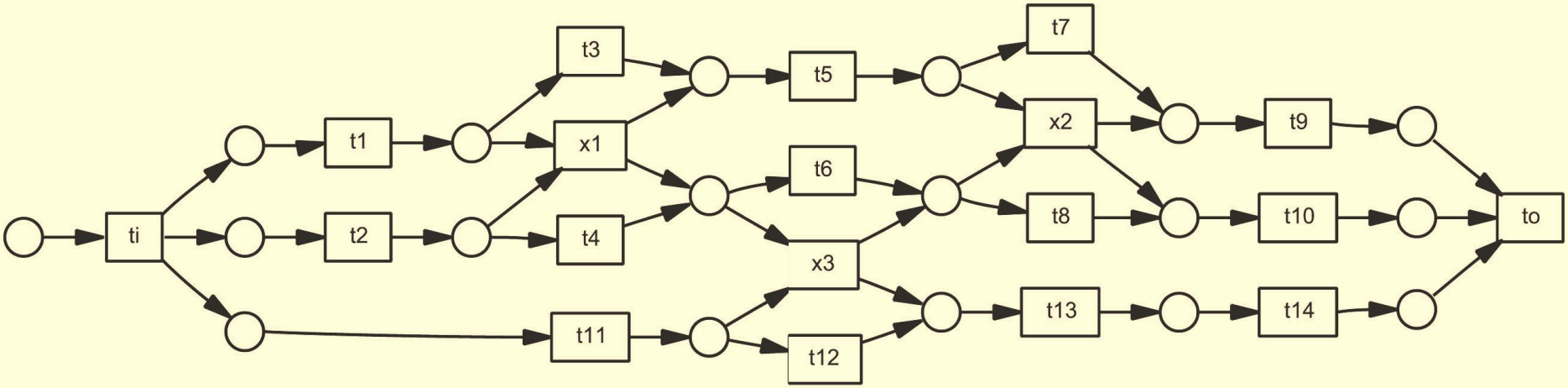


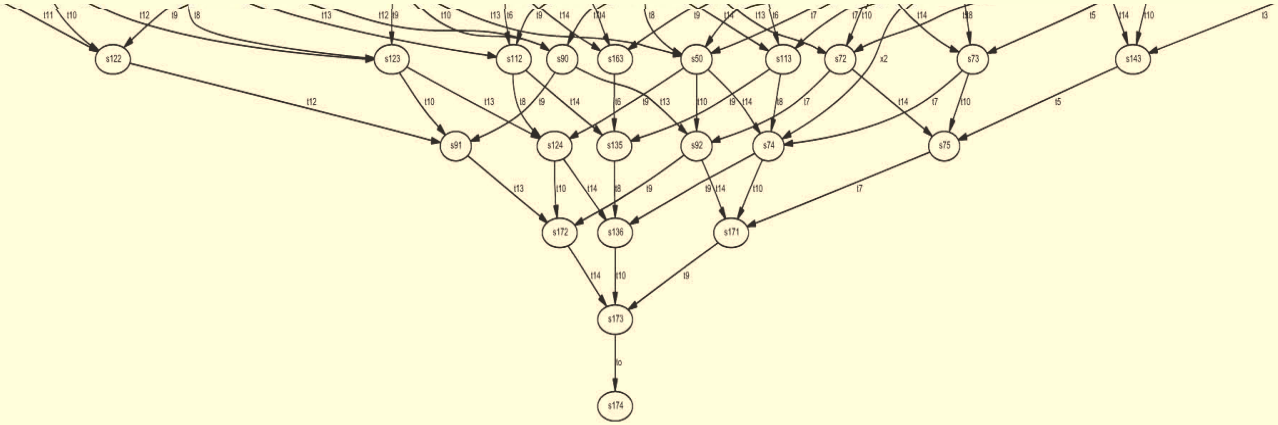
Discovered Petri net

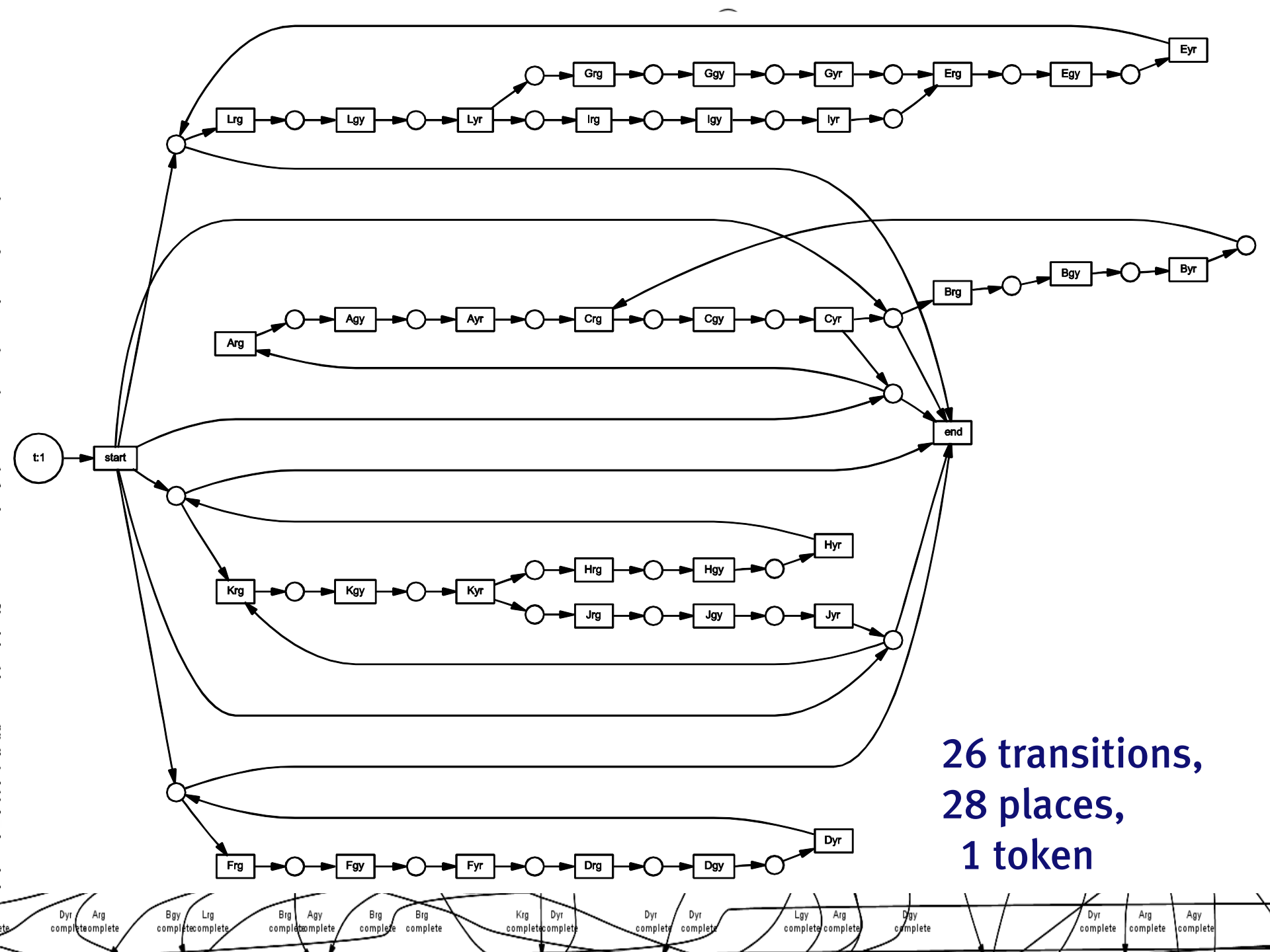


- Petri net is rediscovered!
- Odd example, normally the transition system is constructed from an event log.


s1
s2
s3
s4
s5
s26
s25
s6
s126
s7
s8
s169
s35
s170
s36
s23
s127
s9
s24
s10
s11
s180
s93
s175
s37
s29
s137
s13
s30
s138
s12
s77
s14
s78
s22
s21


ti
t1
t2
t3
x1
t4
t11
t5
t6
x3
t12
t7
x2
t8
t13
t9
t10
t14
to


s122
s123
s112
s90
s163
s50
s113
s72
s73
s143
s91
s124
s135
s92
s74
s75
s172
s136
s171
s173
s174

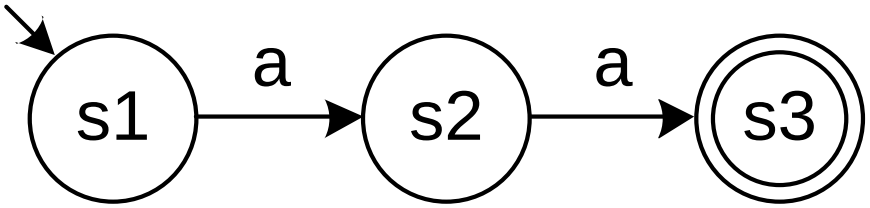

t:1
start
Lrg
Lgy
Lyr
Grg
Ggy
Gyr
Erg
Egy
Eyr
Irg
Igy
Iyr
Arg
Agy
Ayr
Crg
Cgy
Cyr
Brg
Bgy
Byr
end
Krg
Kgy
Kyr
Hrg
Hgy
Hyr
Jrg
Jgy
Jyr
Frg
Fgy
Fyr
Drg
Dgy
Dyr
26 transitions,
28 places,
1 token

It is not that simple…

(but all problems can be repaired)

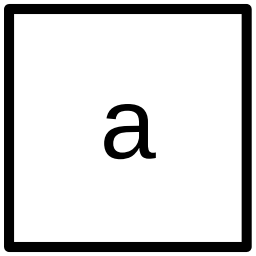
Consider an event log containing just <a,a> traces

prefix automaton



Only trivial regions: ∅ and {s1,s2,s3}

Petri net

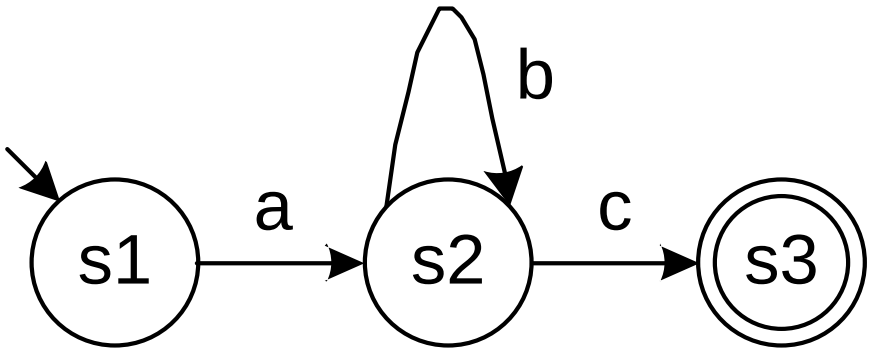


Also allows for:

- **a**
- **aaaa**
- **aaaaaaaa**

Consider an event log containing traces <a,c>, <a,b,c>, <a,b,b,c>,<a,b,b,b,c>, …

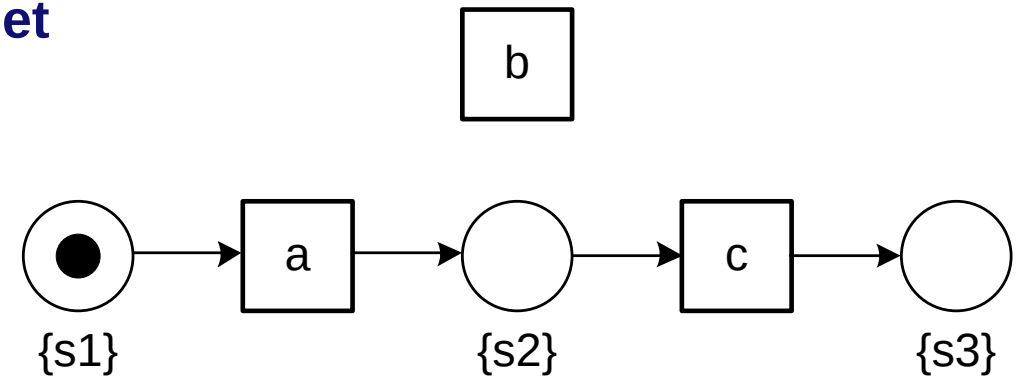
transition system able to generate log



Regions:

- **{s1} (a exits, b and c do not cross)**
- **{s2} (a enters, b does not cross, c exits)**
- **{s3} (a and b does not cross, c enters)**

Petri net

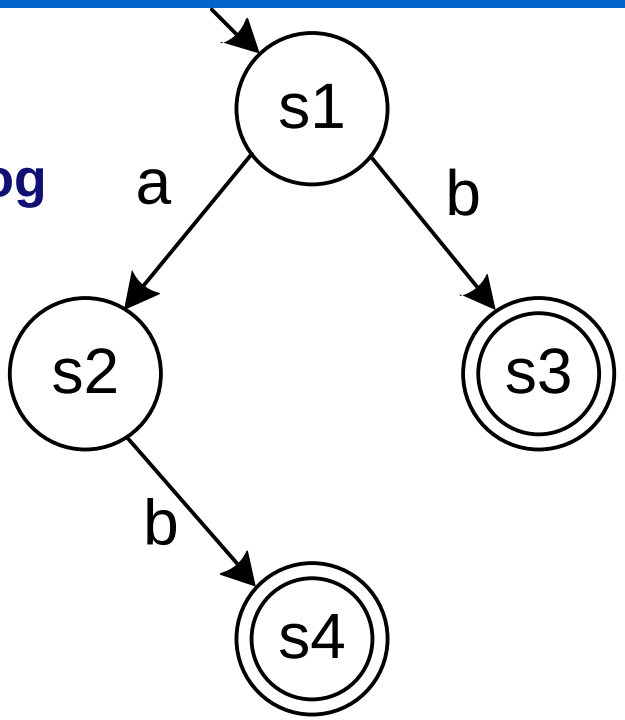


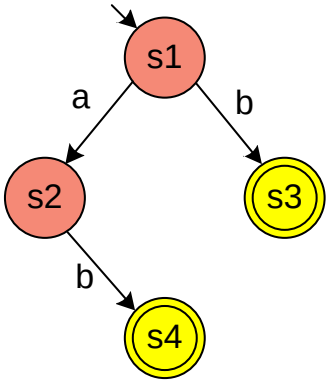
Also allows for:

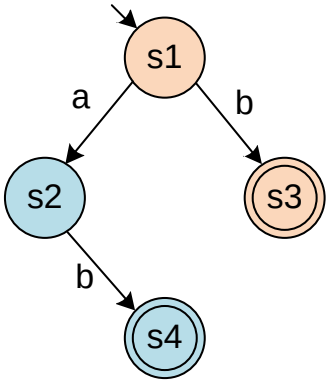
- **bbac**
- **acbbbb**
- **babcb**

Consider an event log containing traces <a,b>, <b>

transition system able to generate log



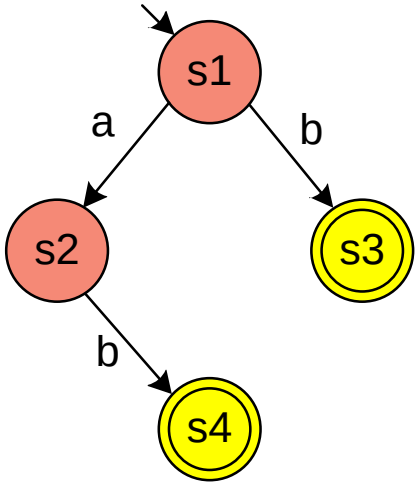


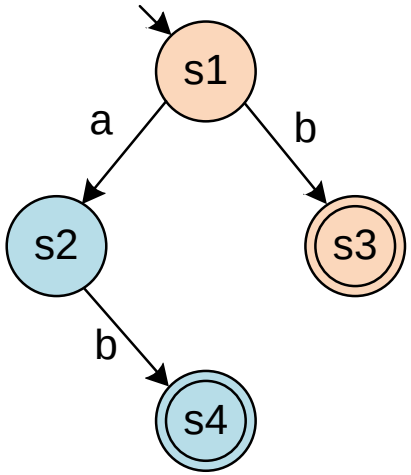


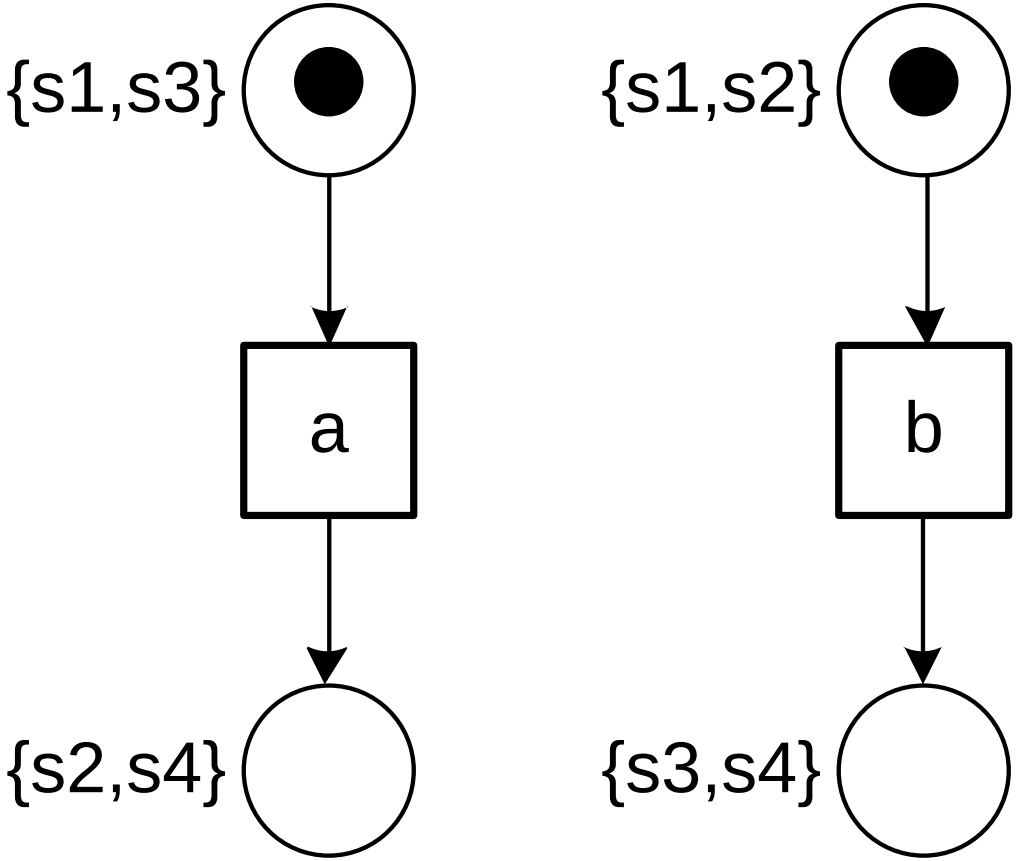
Regions:

- **{s1,s2} (a does not cross, b exits)**
- **{s3,s4} (a does not cross, b enters)**
- **{s1,s3} (a exits and b does not cross)**
- **{s2,s4} (a enters and b does not cross)**

Petri net

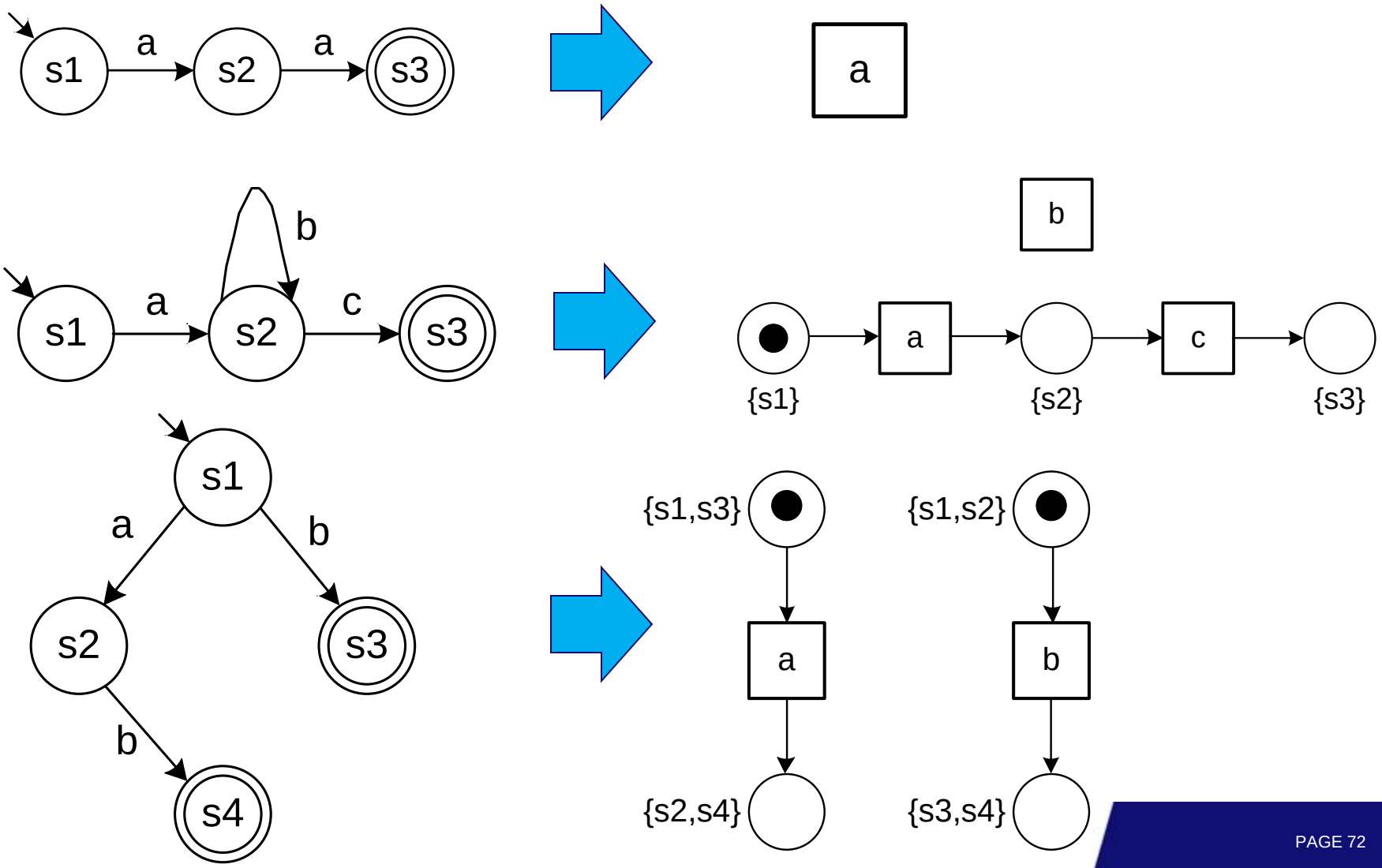




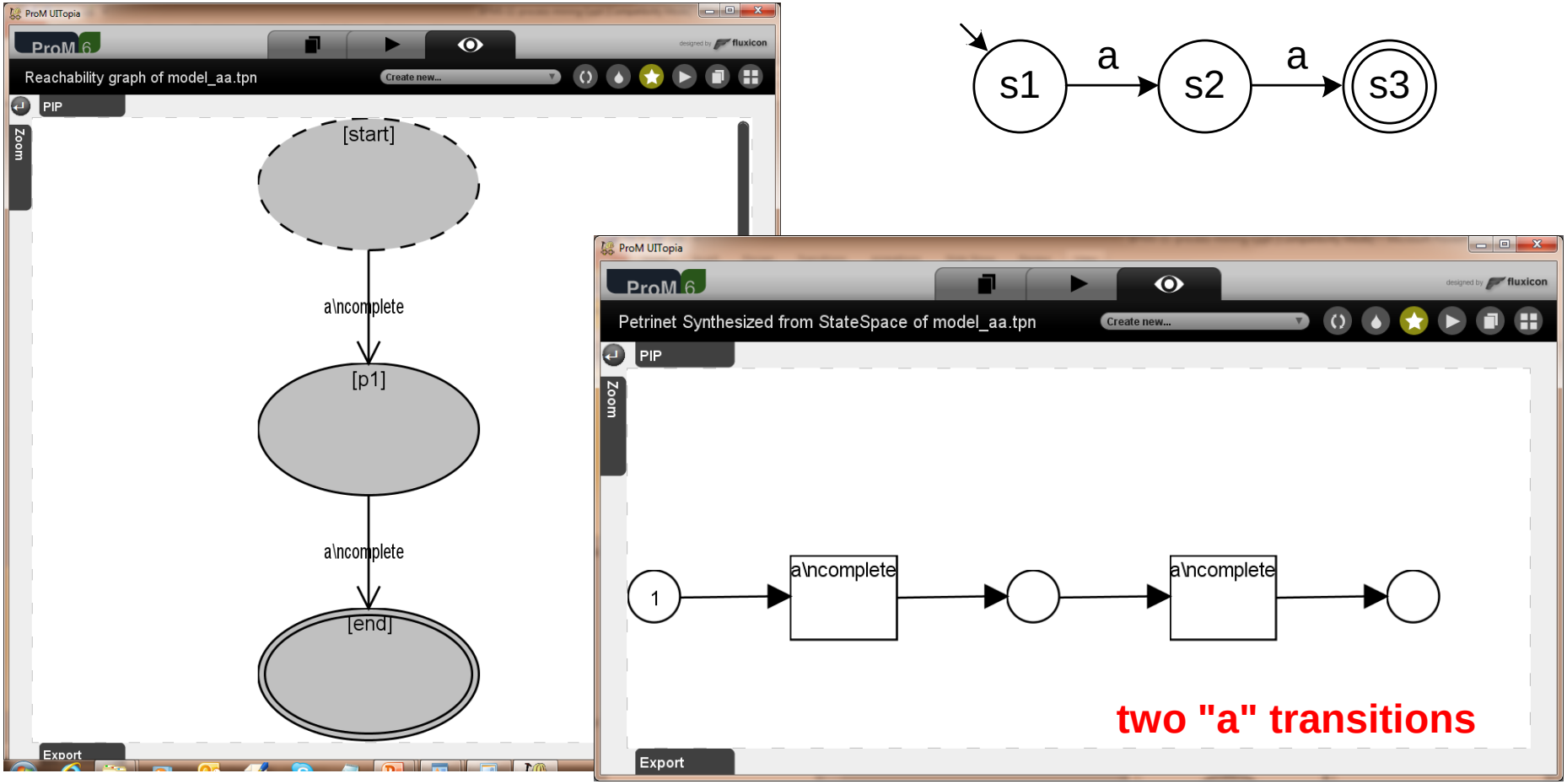


Also allows for trace <b,a>!

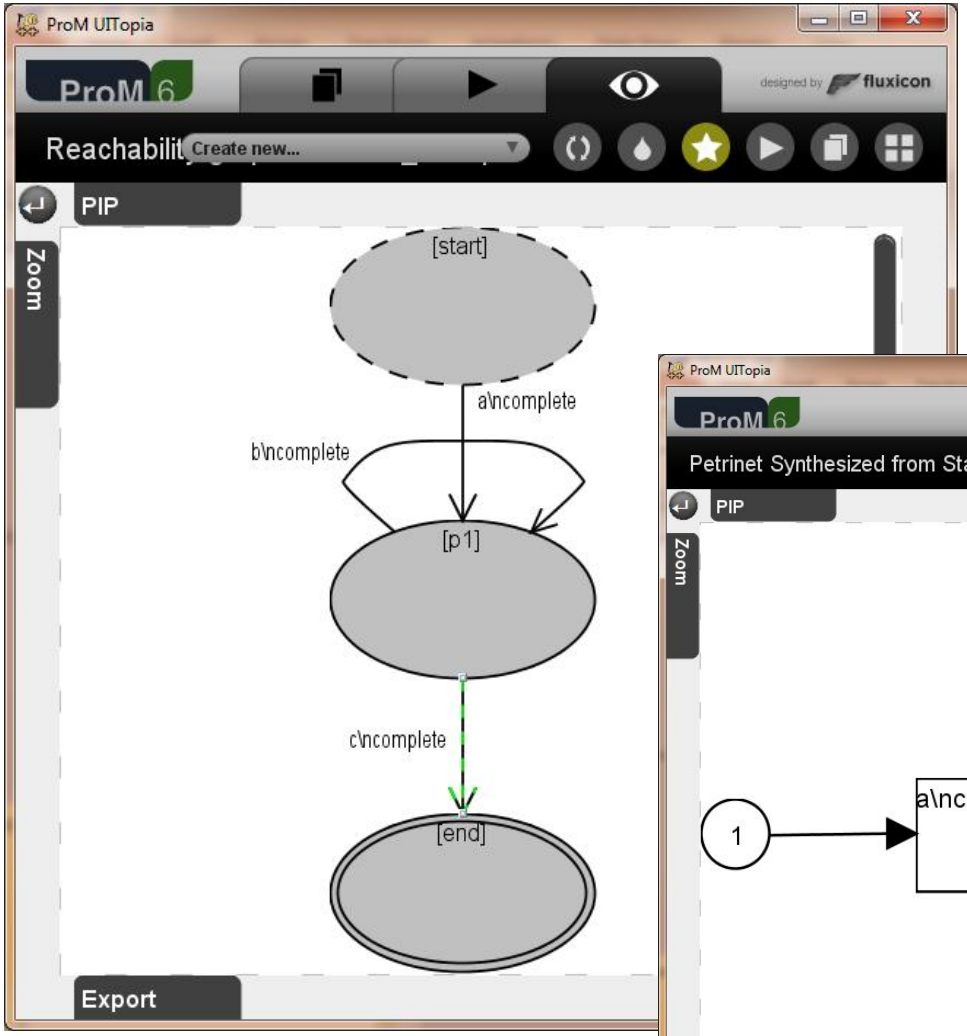
All underfitting, but feasible

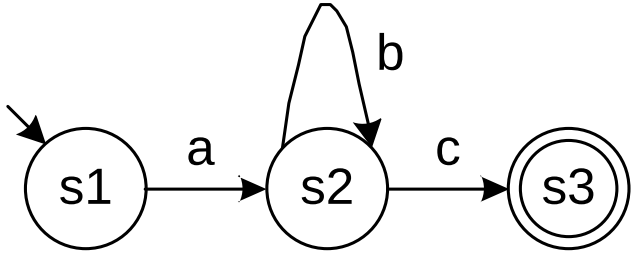


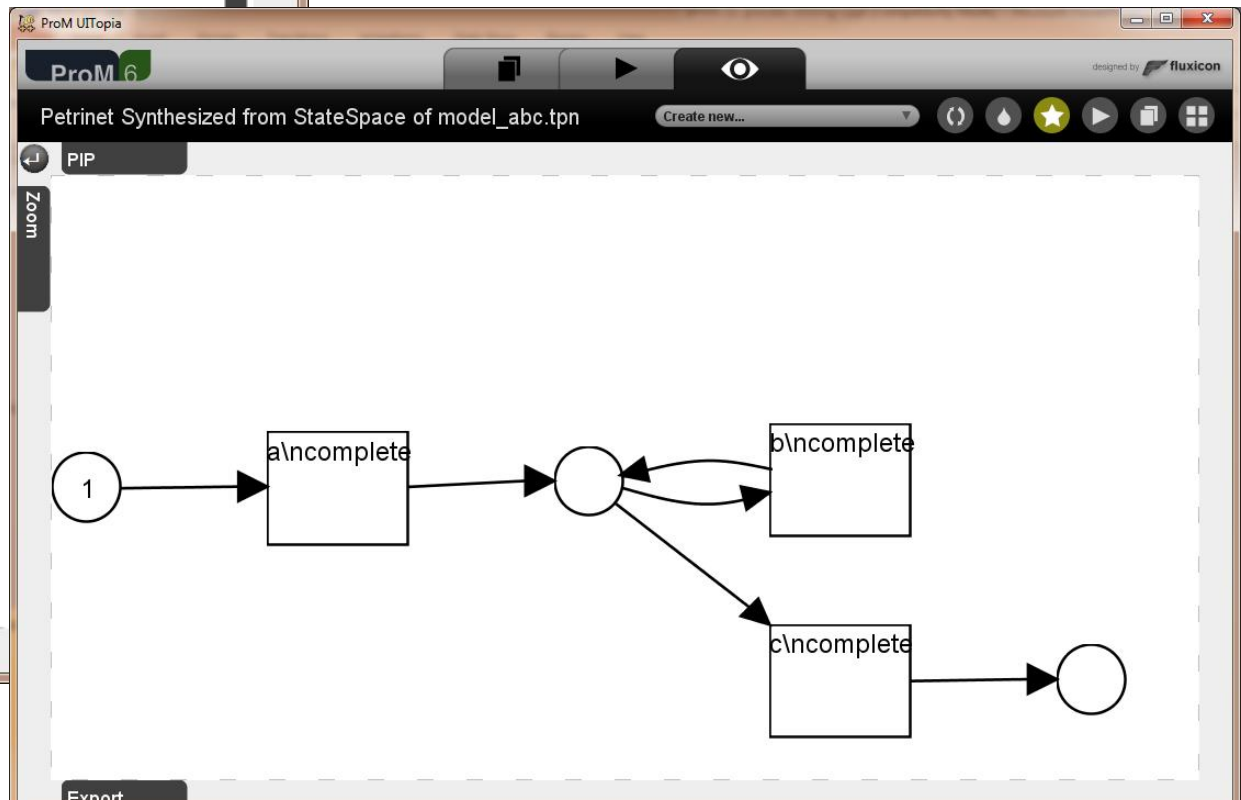
Using ProM
(uses label splitting to solve problem)



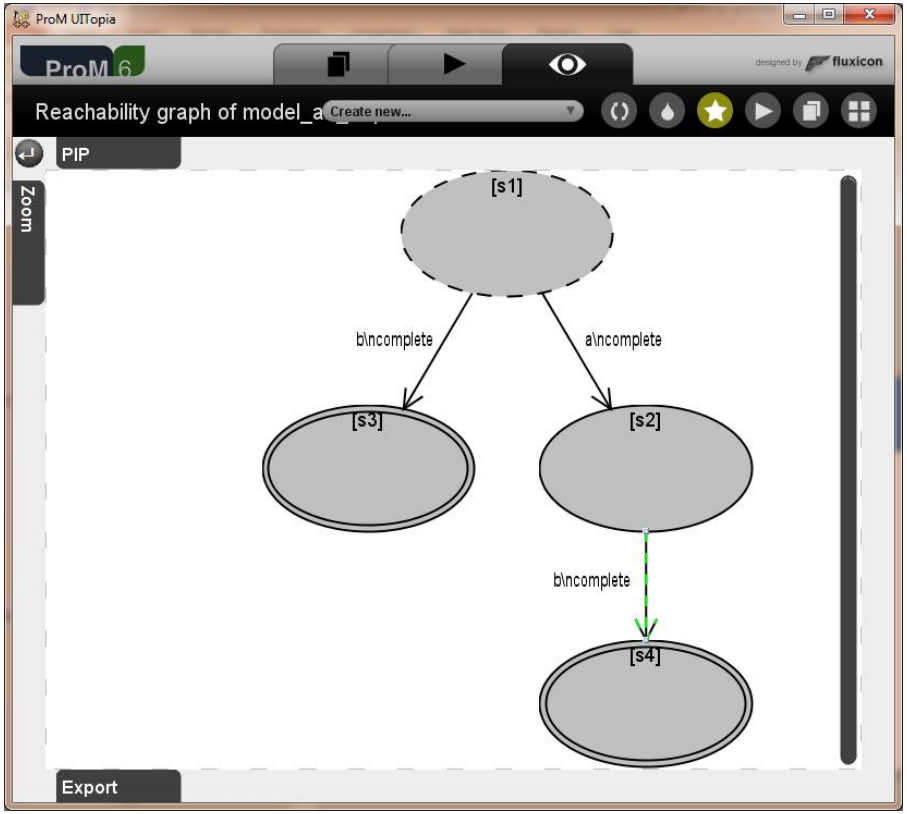
Using ProM
(addresses self-loop problem)

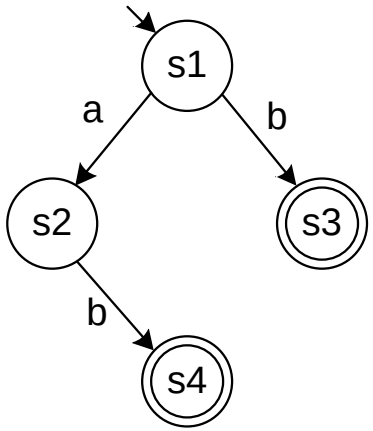


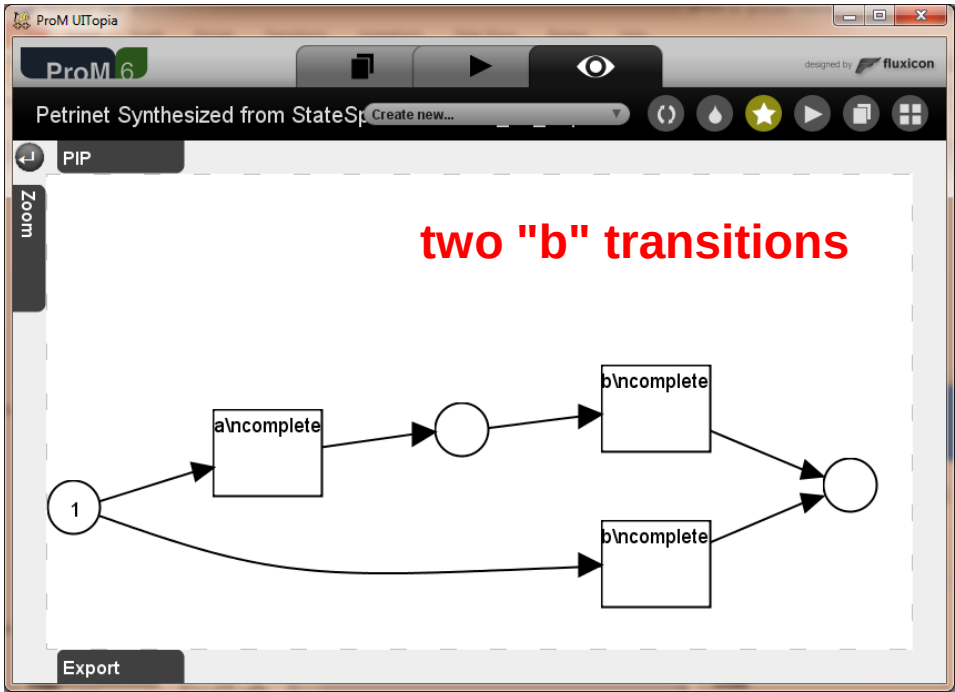




Using ProM
(uses label splitting to solve problem)



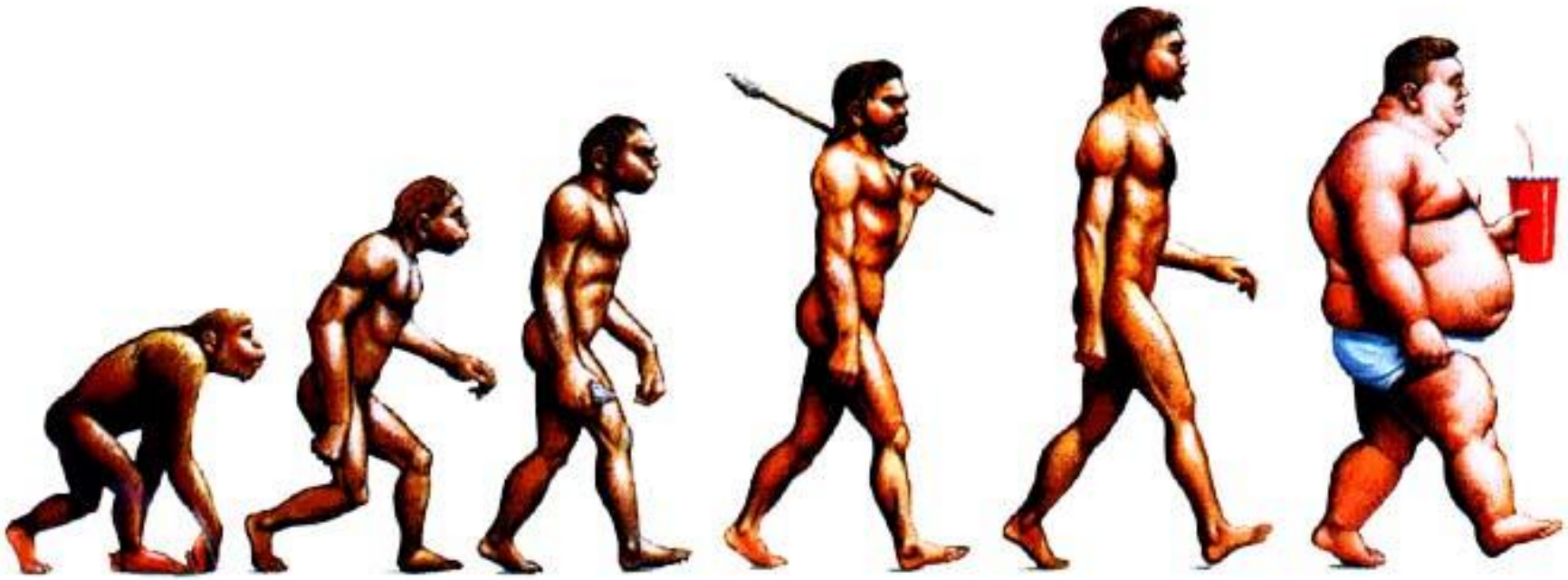




At the other end of the spectrum …

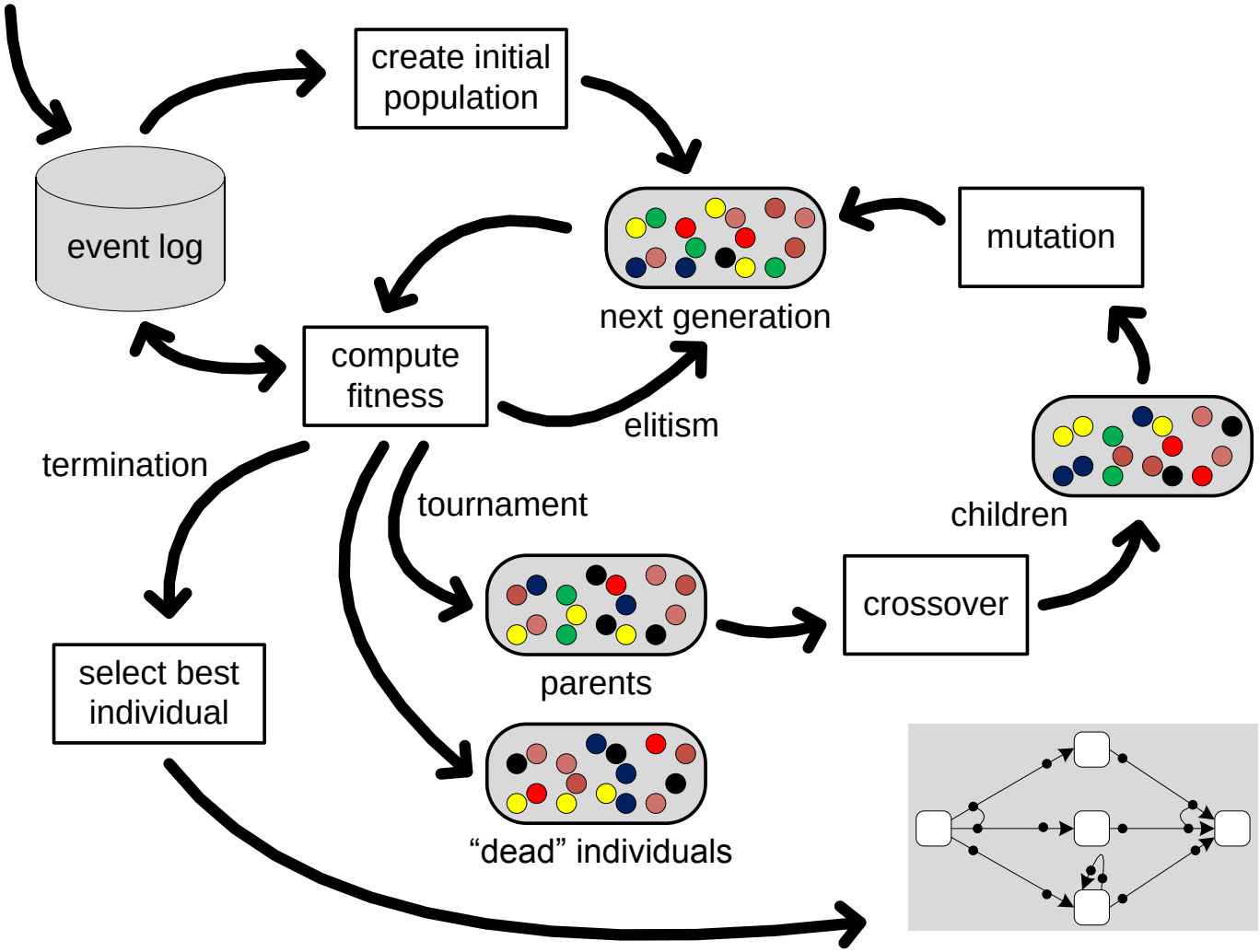
A completely different example of a process discovery technique:

Genetic Mining

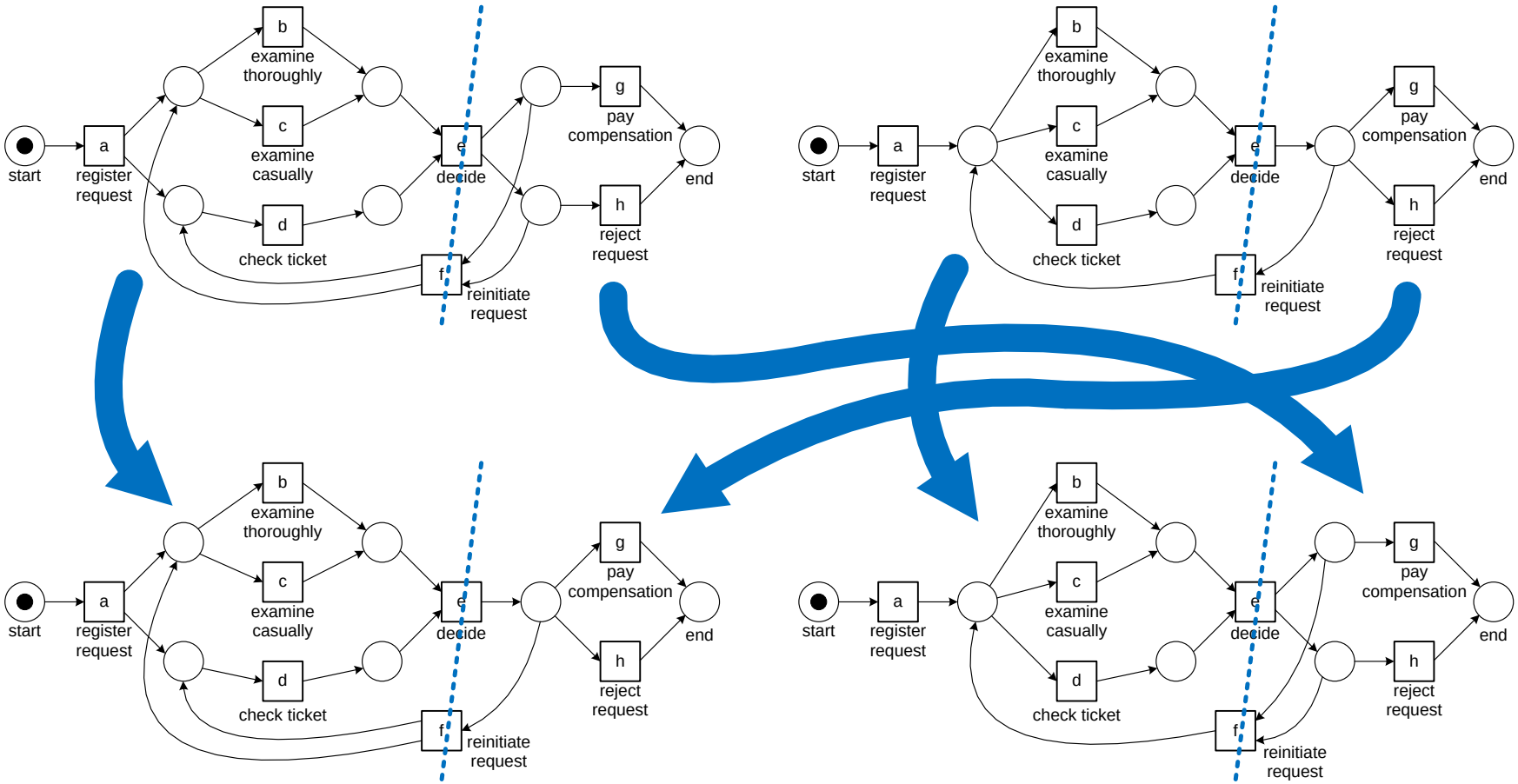


- requires a lot of computing power, but can be distributed easily,
- can deal with noise, infrequent behavior, duplicate tasks, invisible tasks,
- allows for incremental improvement and combinations with other approaches (heuristics post-optimization, etc.).

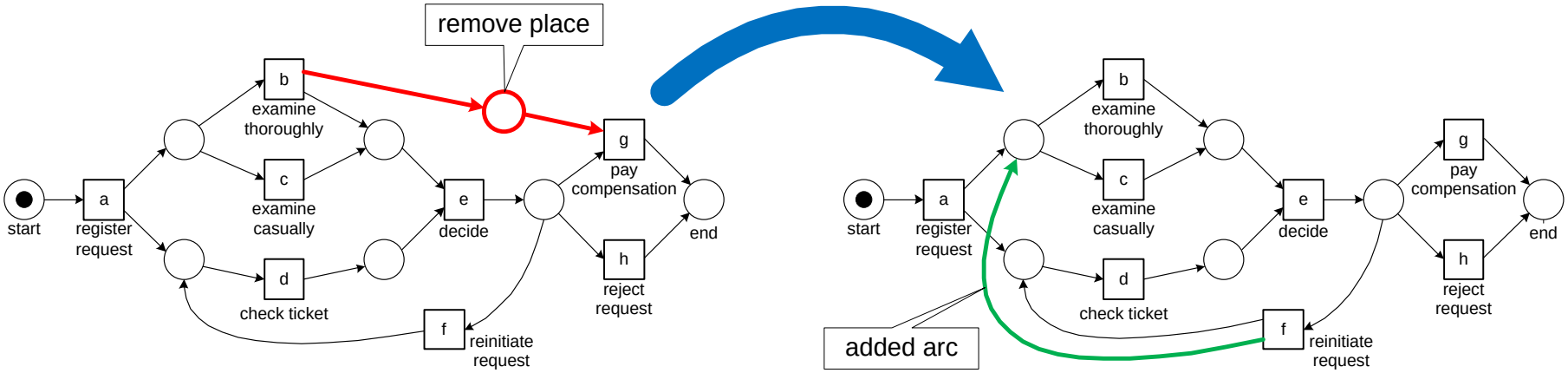
Genetic process mining: Overview



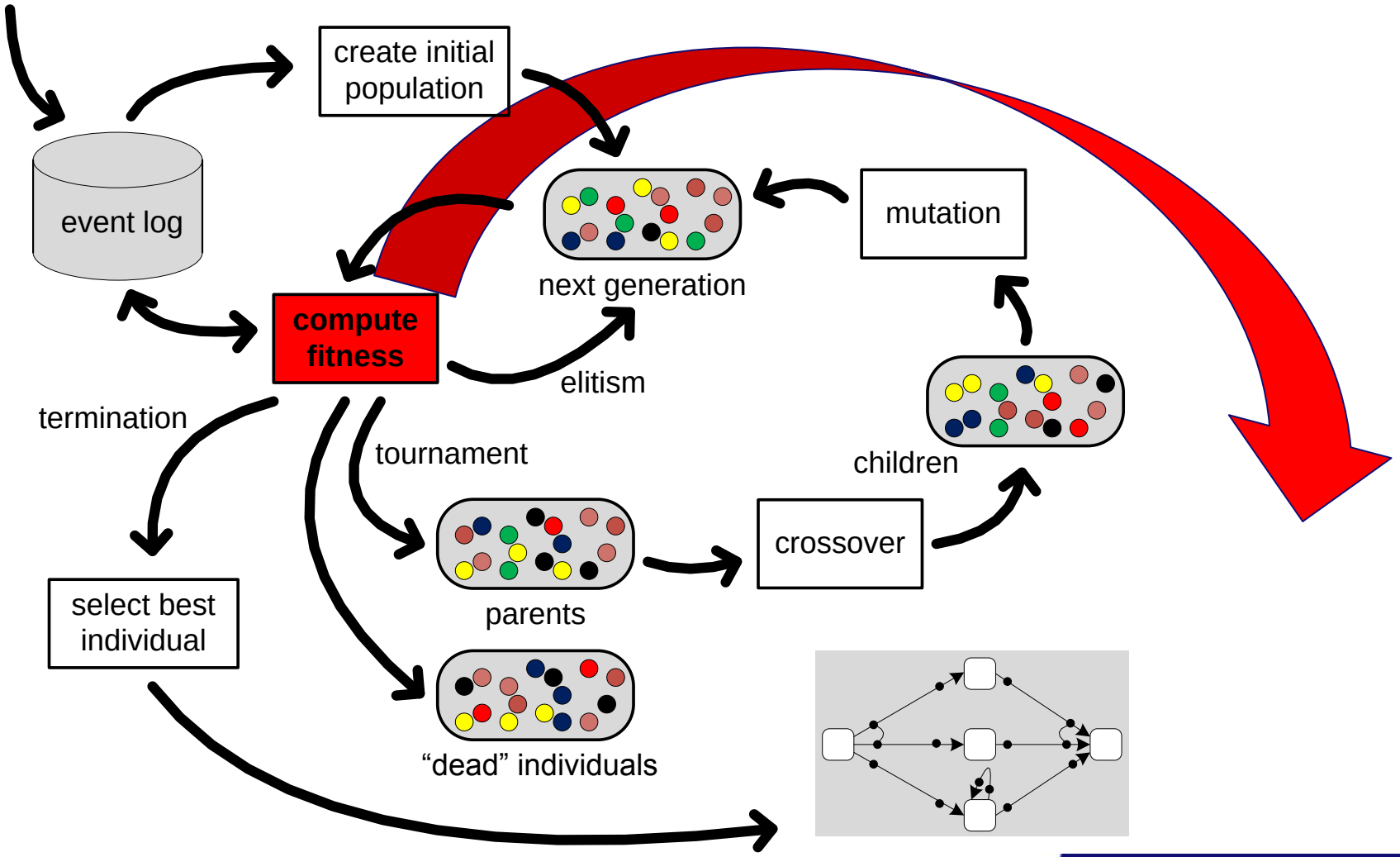
Example: crossover



Example: mutation



Link between process discovery and conformance checking



How good is my model?

Four Competing Quality Criteria

"able to replay event log"

fitness

"Occam's razor"

simplicity

process discovery

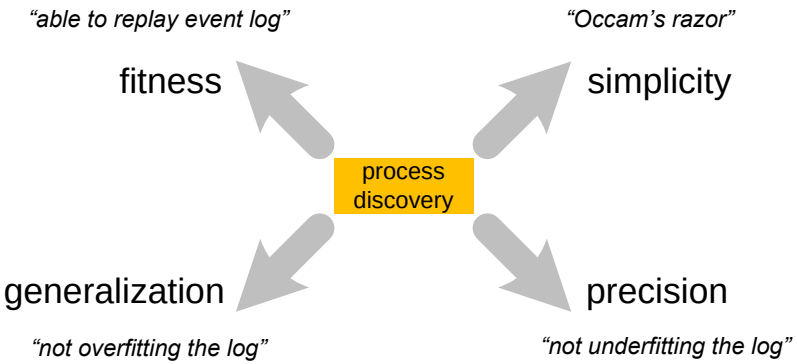
generalization

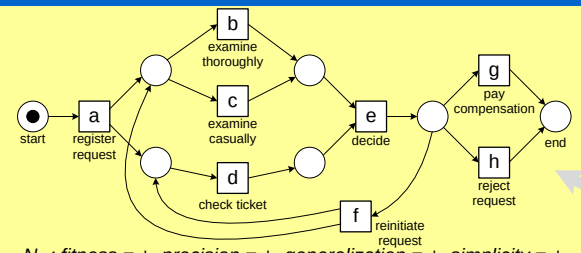
"not overfitting the log"

precision

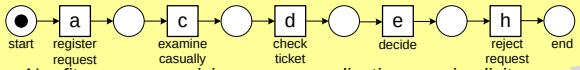
"not underfitting the log"

Example: one log four models

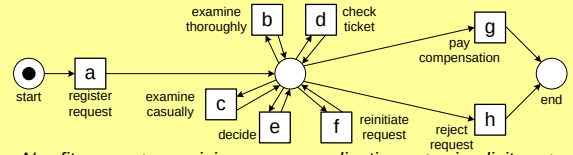




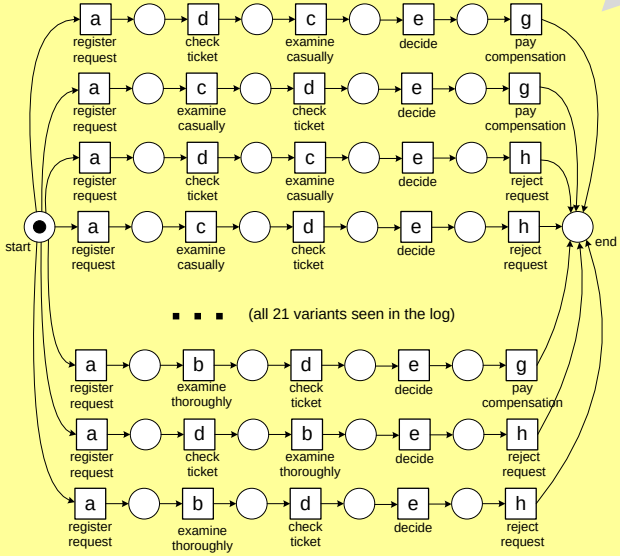
N_1 : *fitness = +, precision = +, generalization = +, simplicity = +*



N_2 : *fitness = -, precision = +, generalization = -, simplicity = +*



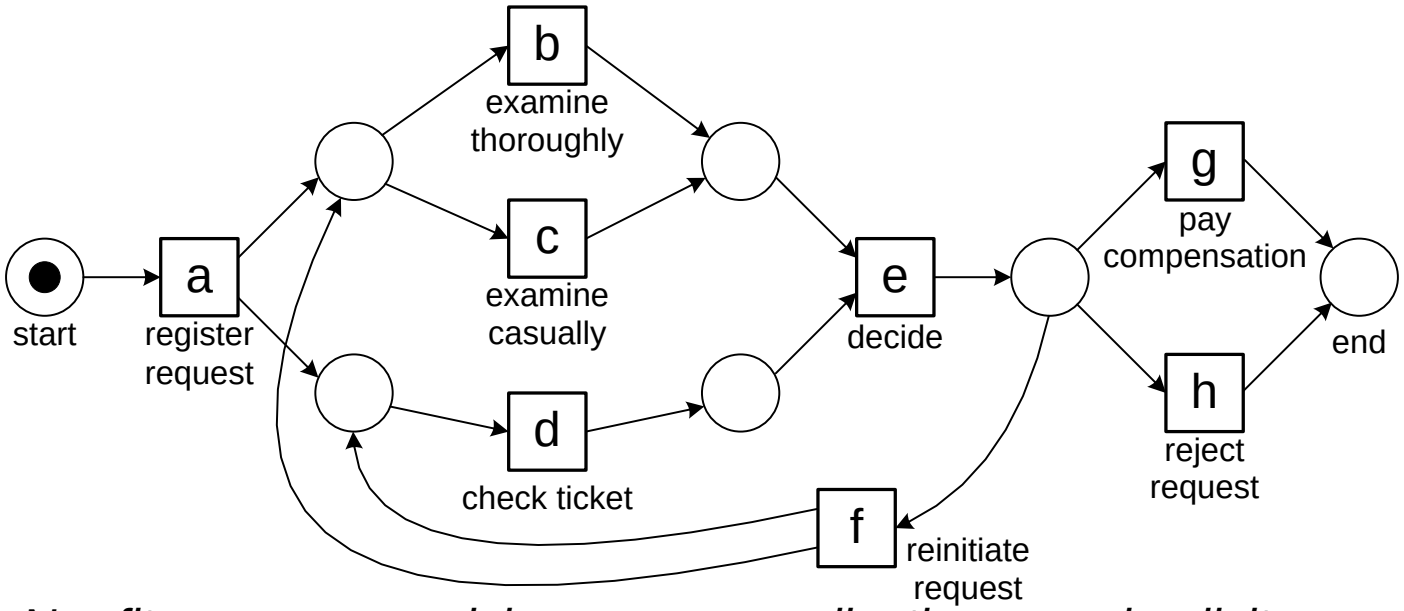
N_3 : *fitness = +, precision = -, generalization = +, simplicity = +*



N_4 : *fitness = +, precision = +, generalization = -, simplicity = -*

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

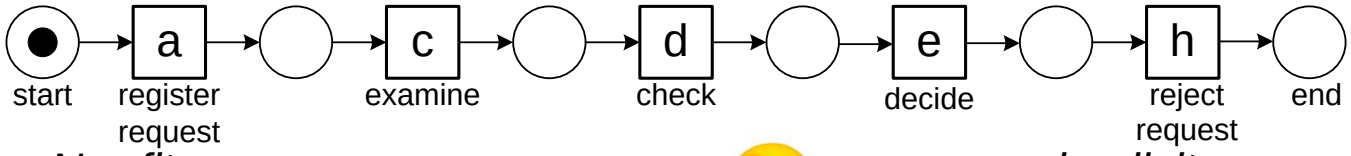
Model N_1



N_1 : fitness = +, precision = +, generalization = +, simplicity = +

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

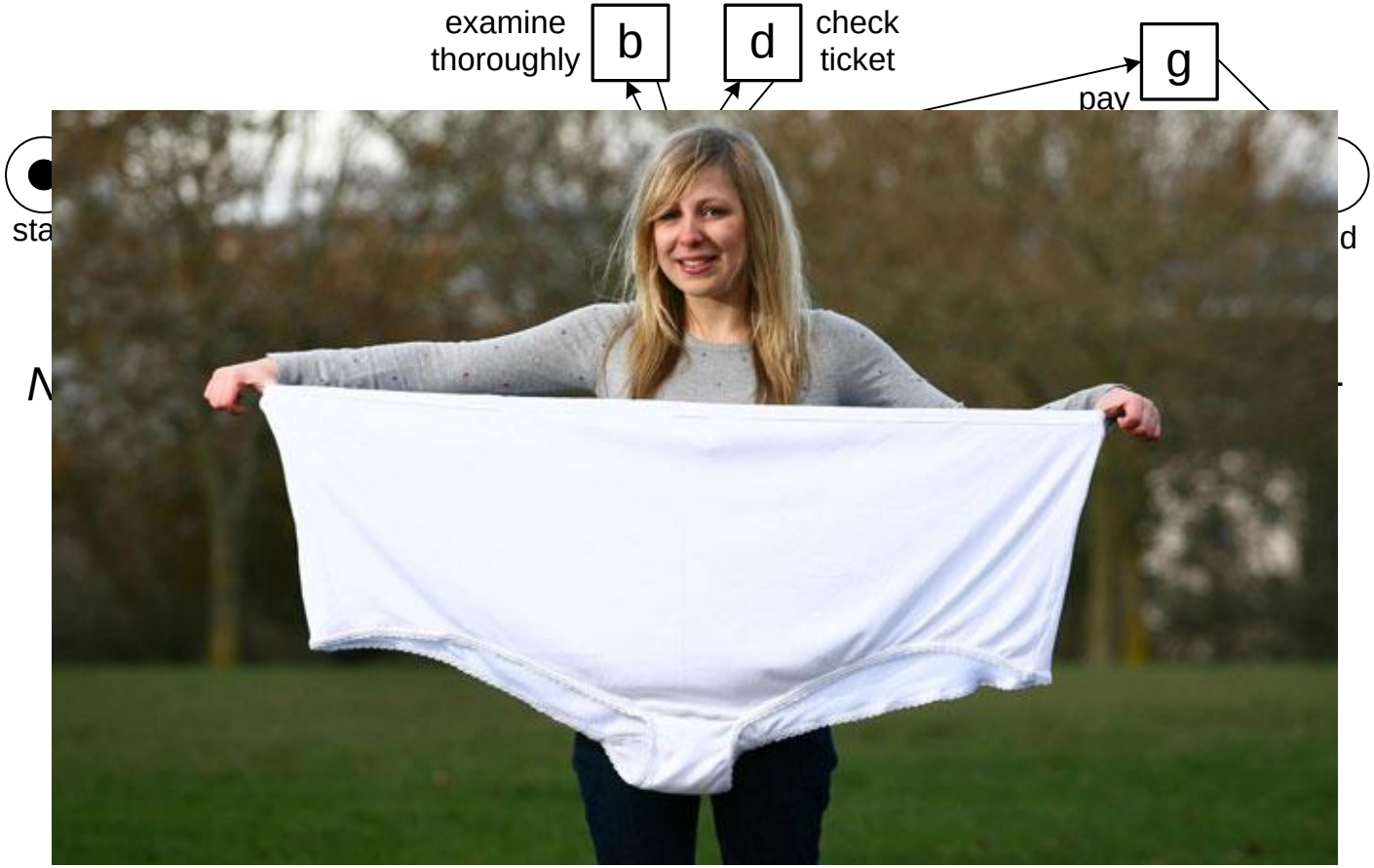
Model N_2



N_2 : fitness = *, simplicity = +*

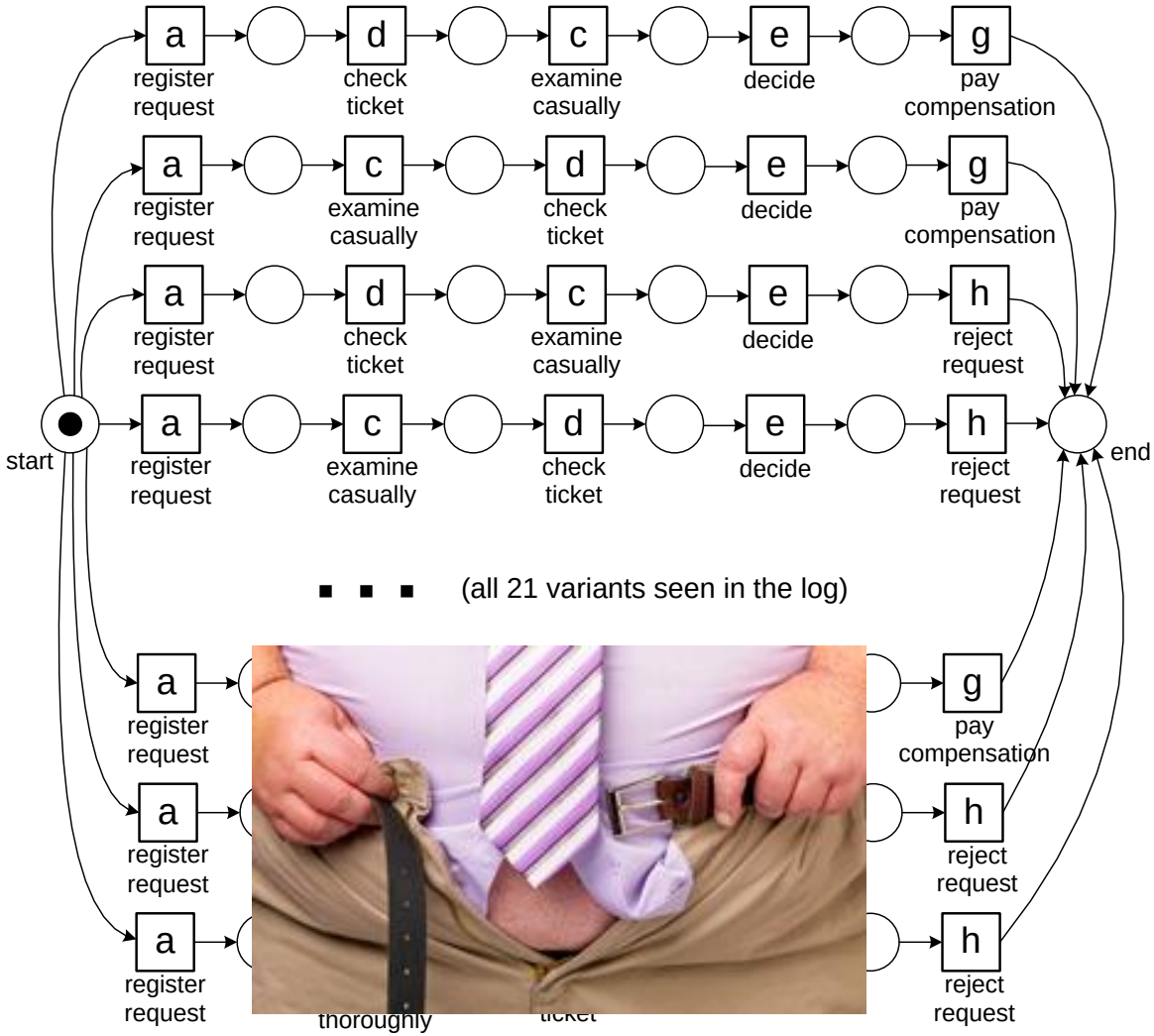
#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

Model N_3



#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

Model N_4

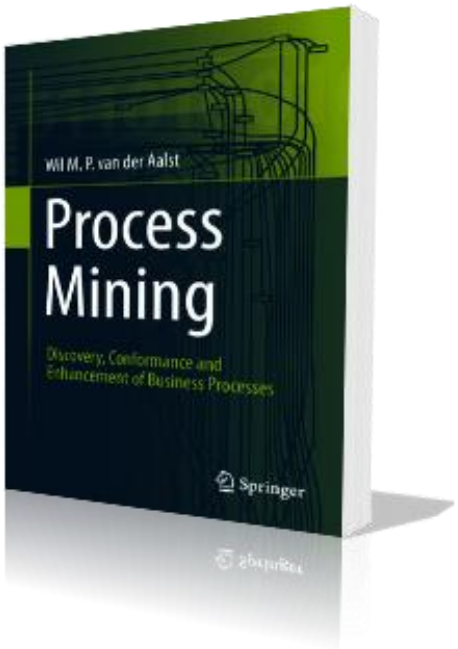


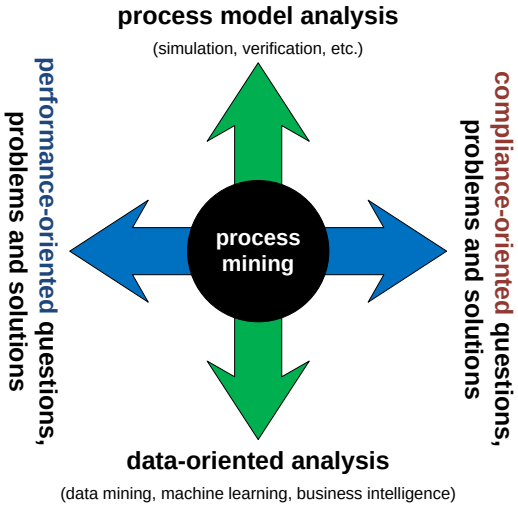
N_4 : fitness = +, precision = +, generalization = -, simplicity = -

#	trace
455	acdeh
191	abdeg
177	adceh
144	abdeh
111	acdeg
82	adceg
56	adbeh
47	acdefdbeh
38	adbeg
33	acdefbdeh
14	acdefbdeg
11	acdefdbeg
9	adcefcdeh
8	adcefdbeh
5	adcefbdeg
3	acdefbdefdbeg
2	adcefdbeg
2	adcefbdefbdeg
1	adcefdbefbdeh
1	adbefbdefdbeg
1	adcefdbefcdefdbeg
1391	

Conclusion

Still many challenging and highly relevant open problems in process discovery!





ProM
process mining workbench

processmining.org