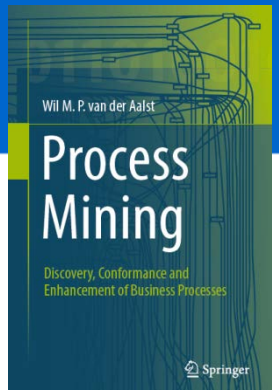
*Process Mining: Data Science in Action*

# Using Regions to Discover Concurrency

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

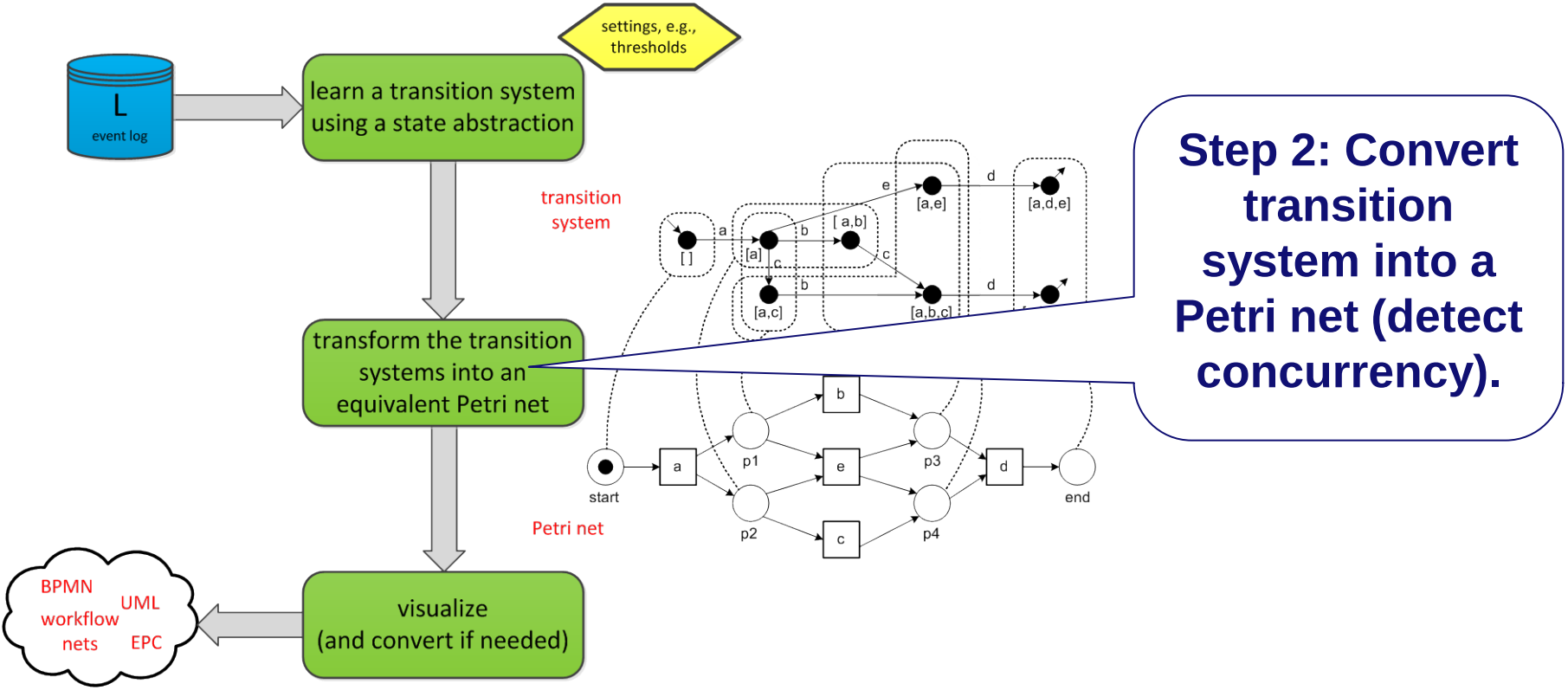



Technische Universiteit Eindhoven University of Technology

**Where innovation starts**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

# State-based regions

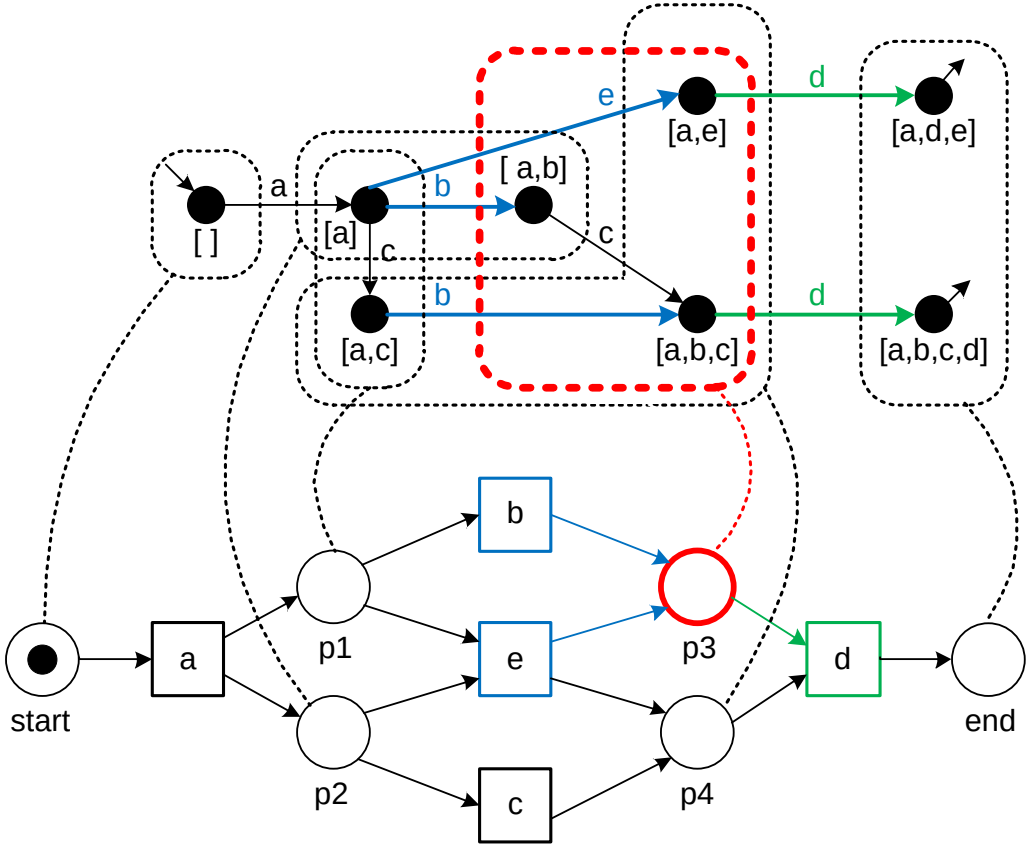



**Step 2: Convert transition system into a Petri net (detect concurrency).**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

# State-based regions correspond to places

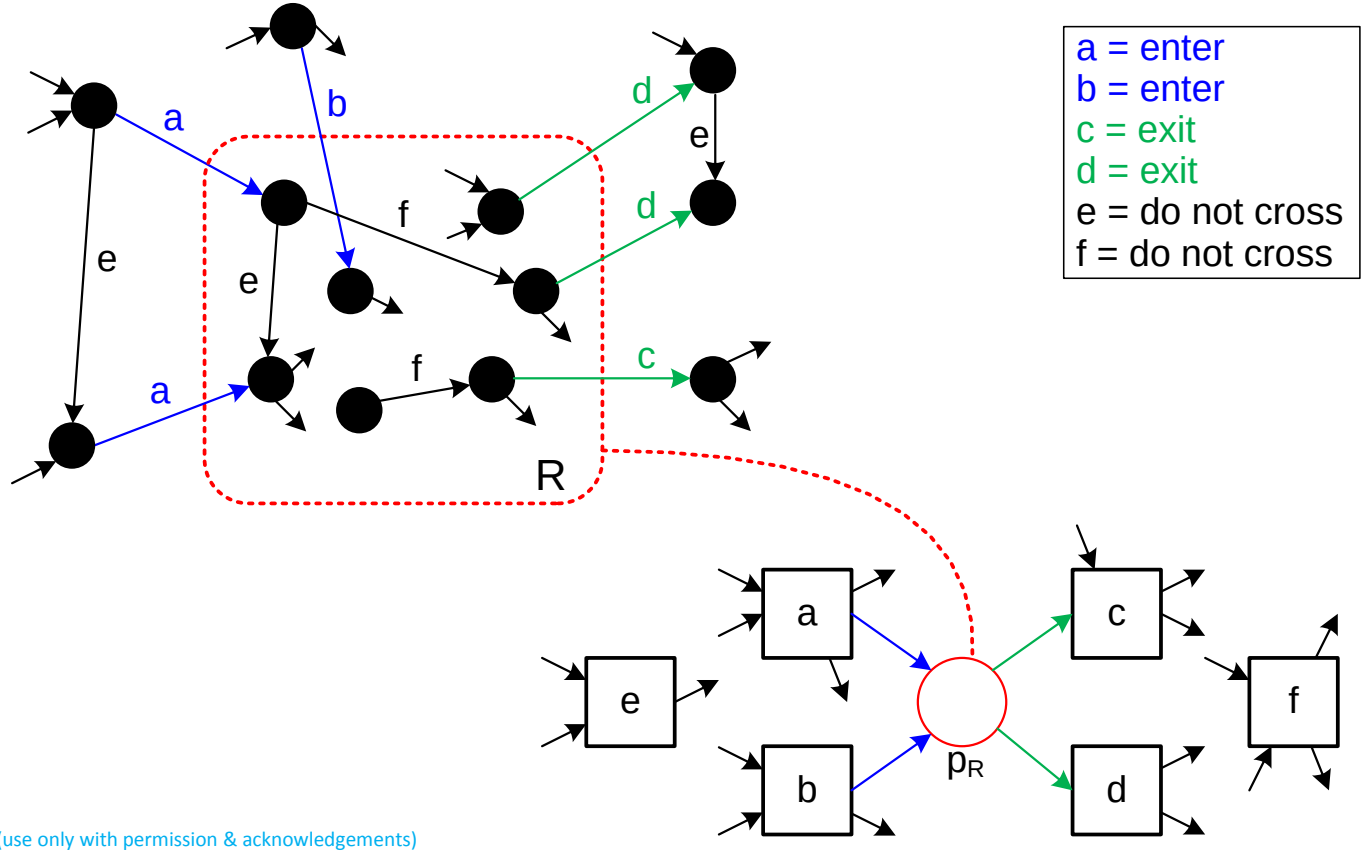
**It is all about discovering concurrency …**




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

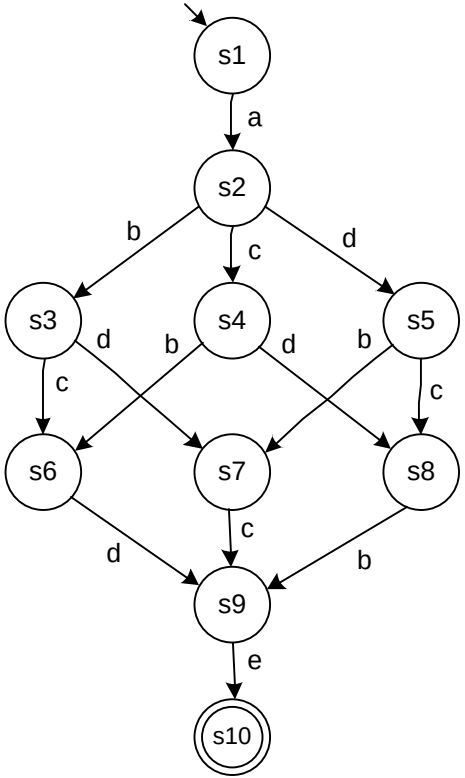
TU/e

# What is a (state-based) region?




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

# Starting point: A transition system



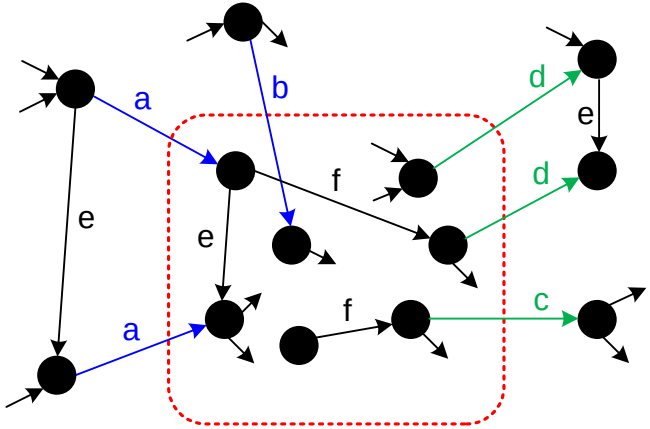

- We assume that there is only one initial state (otherwise preprocessing needed).
- It is convenient to also have just one final state that can always be reached (not strictly necessary)
- All states need to be reachable!

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Definition

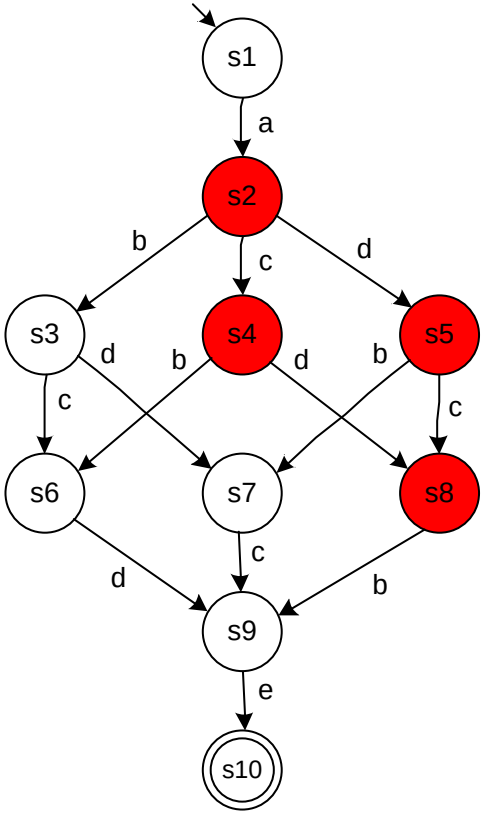
**A region is a set of states, such that, if a transition *exits* the region, then all equally labeled transitions *exit* the region, and if a transition *enters* the region, then all equally labeled transitions *enter* the region. All events not entering or exiting the region *do not cross* the region.**



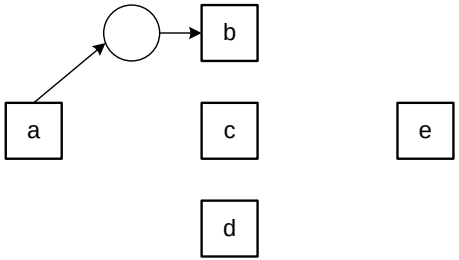

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Example of a region



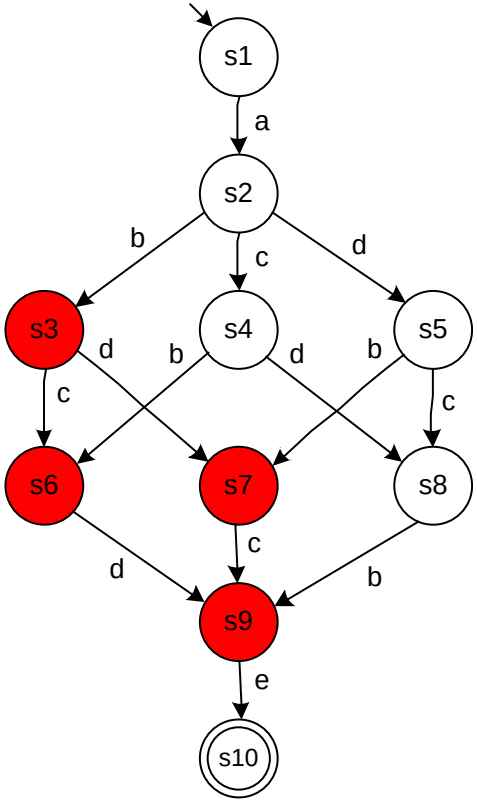

- a enters
- b exits
- c does not cross
- d does not cross
- e does not cross



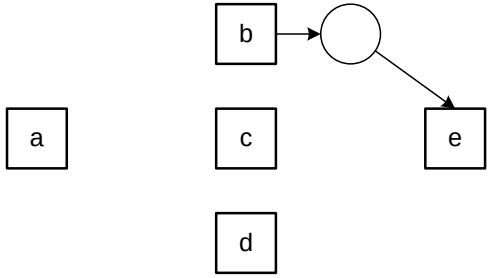

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Example of a region



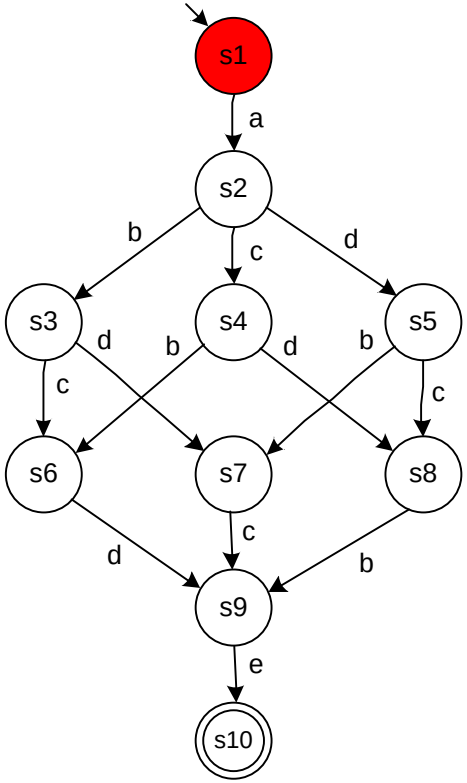

- a does not cross
- b enters
- c does not cross
- d does not cross
- e exits




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

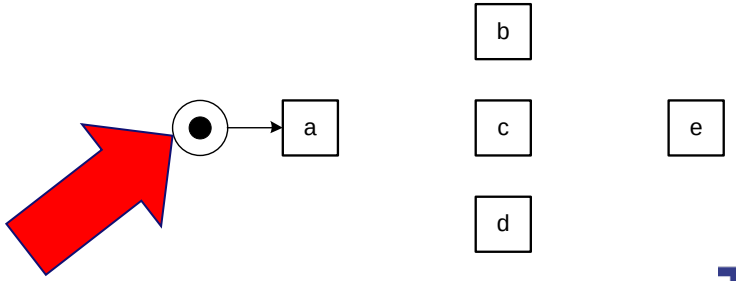
TU/e

# Example of a region




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

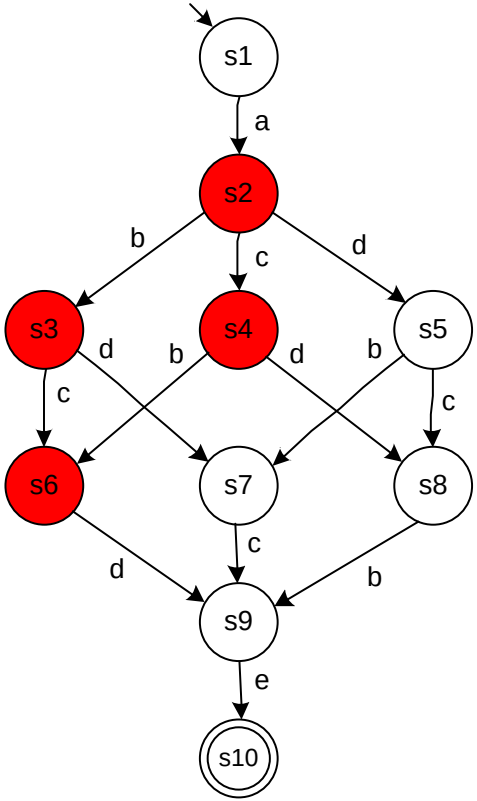
**As will be shown later: places corresponding to regions containing the initial state are initially marked.**



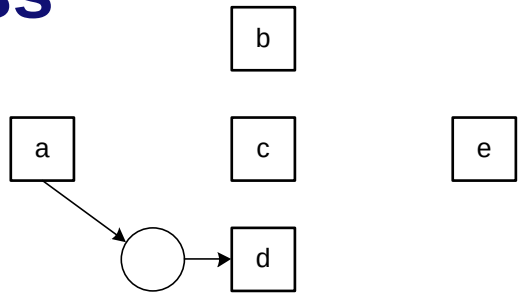

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Example of a region



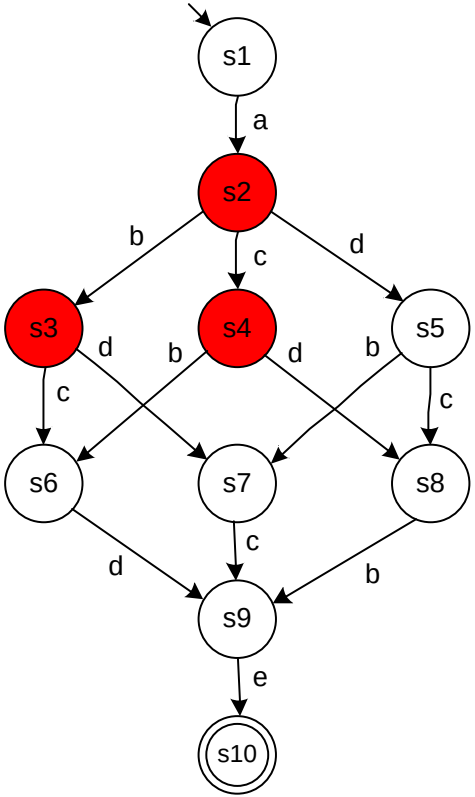

- a enters
- b does not cross
- c does not cross
- d exits
- e does not cross




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Not a region

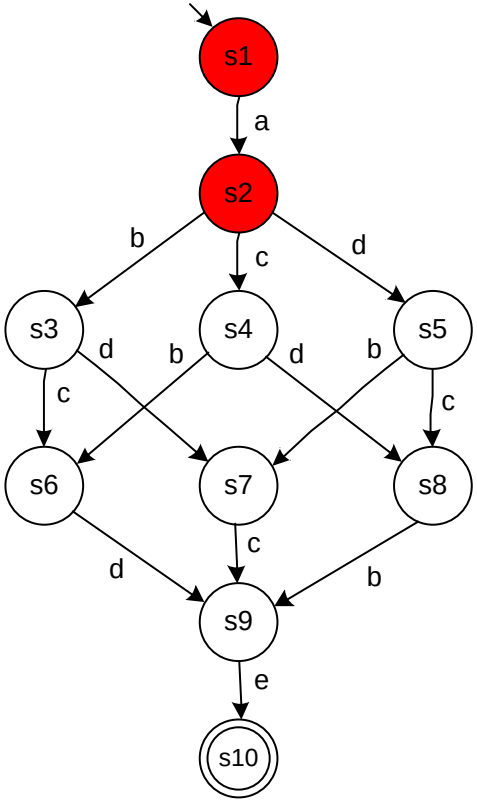



- a enters
- b does not cross and exits
- c does not cross and exits
- d does not cross and exits
- e does not cross

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Not a region

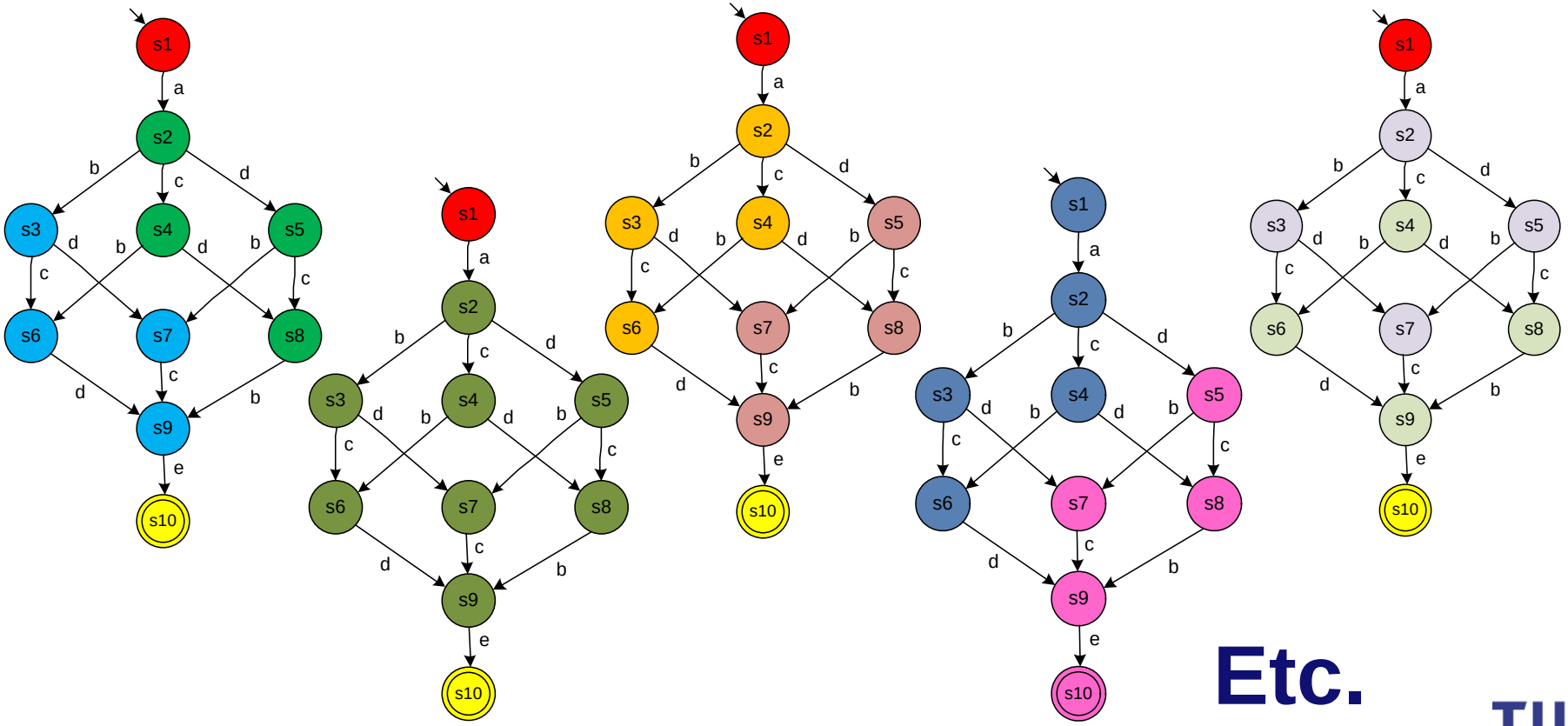



- a does not cross
- b does not cross and exits
- c does not cross and exits
- d does not cross and exits
- e does not cross

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

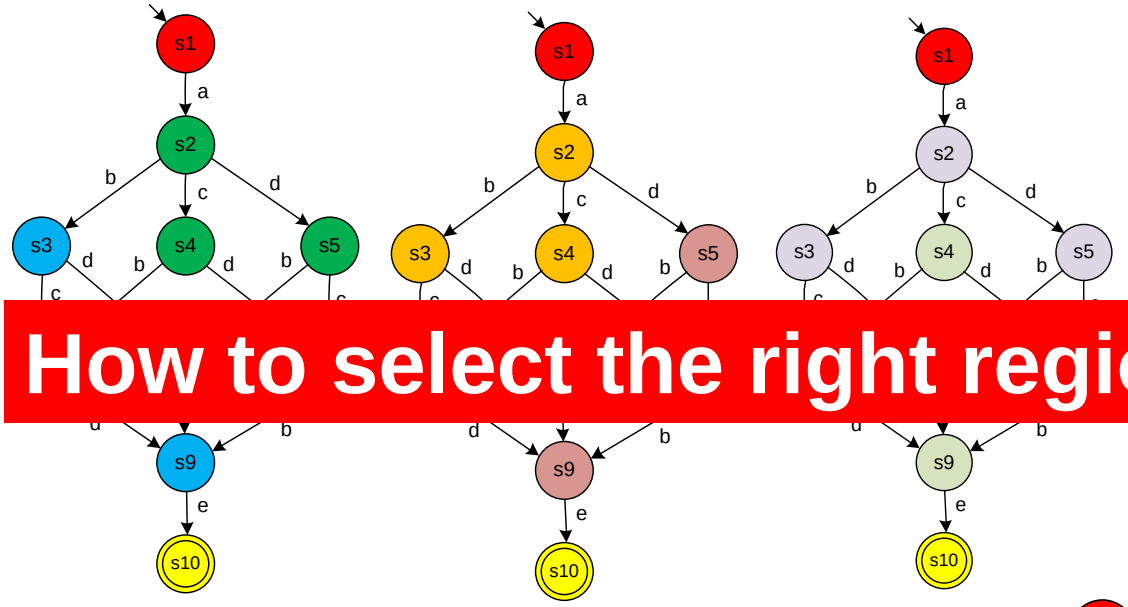
# Multiple regions



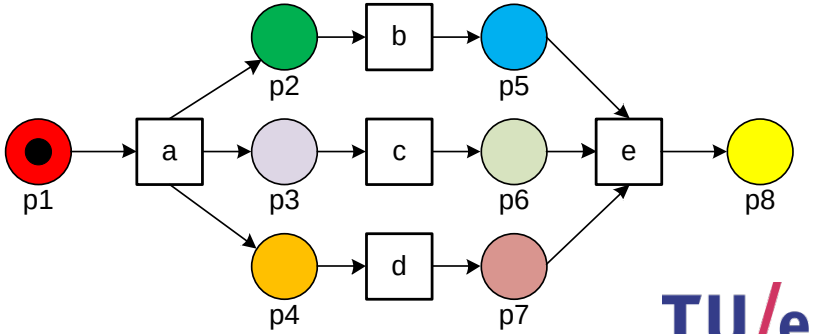

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Selectively chosen regions



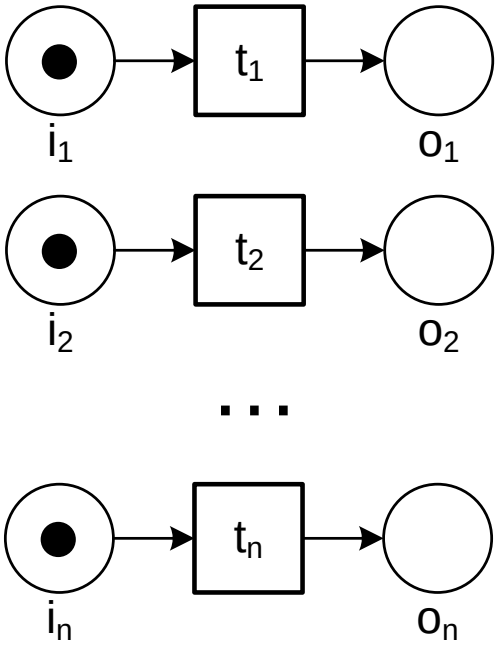

**How to select the right regions?**




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Question

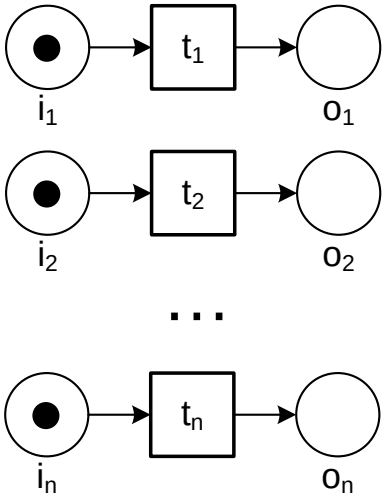



**How many regions does the reachability graph of this Petri net have?**

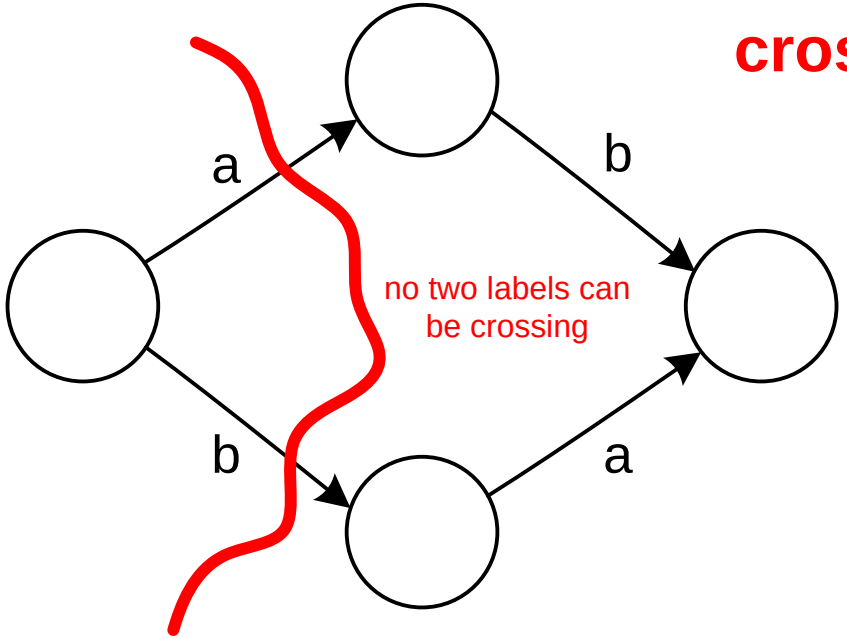
©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Answer: 2(n+1) regions




**$2^n$ states**

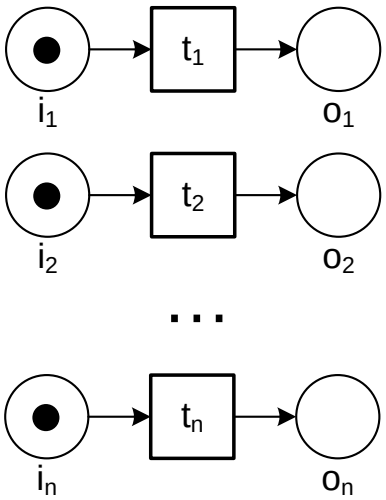


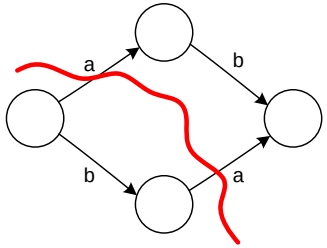

**At most 1 transition label can be crossing (or none).**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Answer: 2(n+1) regions



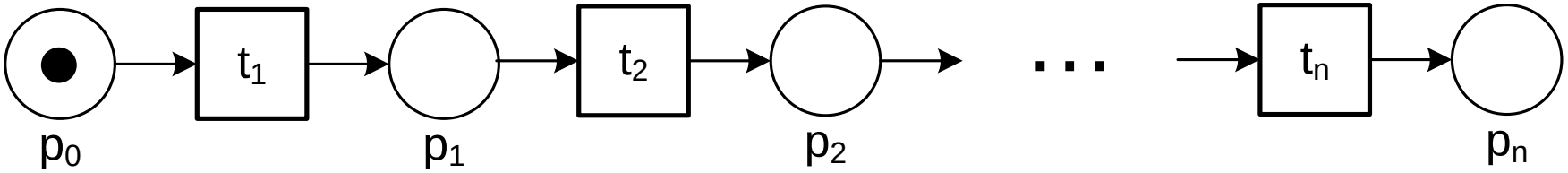

- There are *n* ways to split the states in two groups based on the label selected to cross.
- Each situation splits the set of states in two regions.
- Also the empty set and all the set of all states are regions.

**Answer: 2(n+1) regions.**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Question

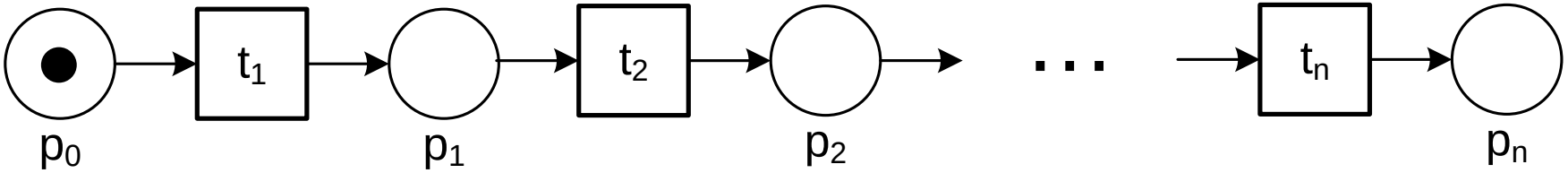



**How many regions does the reachability graph of this Petri net have?**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Answer

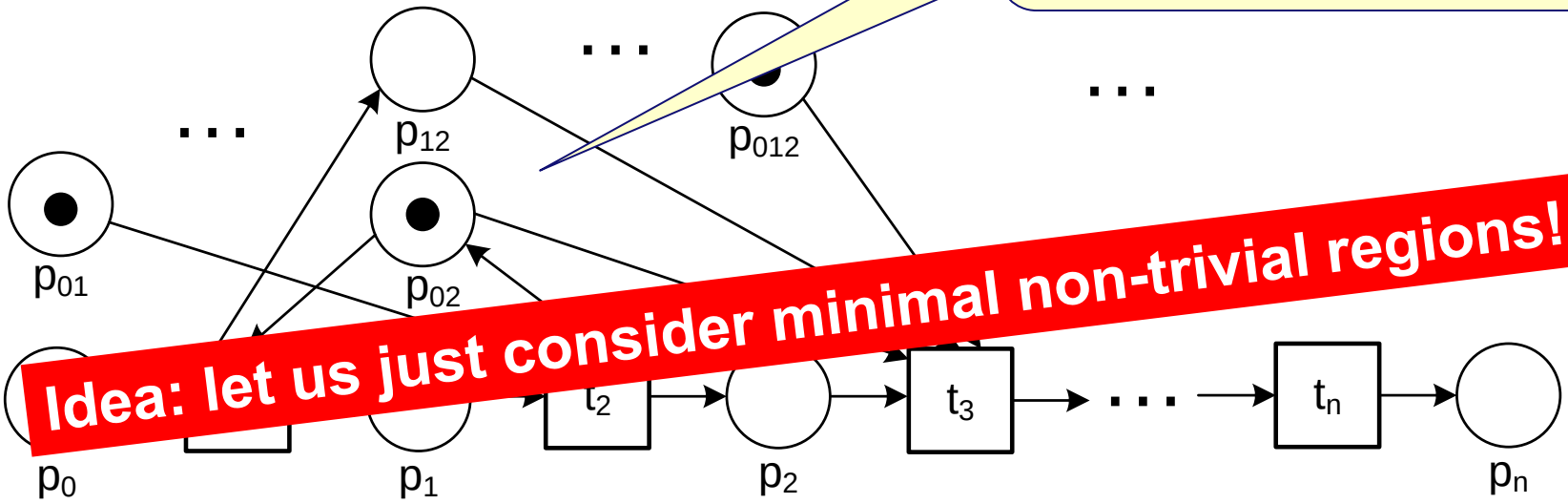



**Every subset of states forms a region.**
**Hence, there are $2^{(n+1)}$ regions.**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Illustration




**For a sequence of 10 activities we have $2^{(10+1)} = 2048$ regions.**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

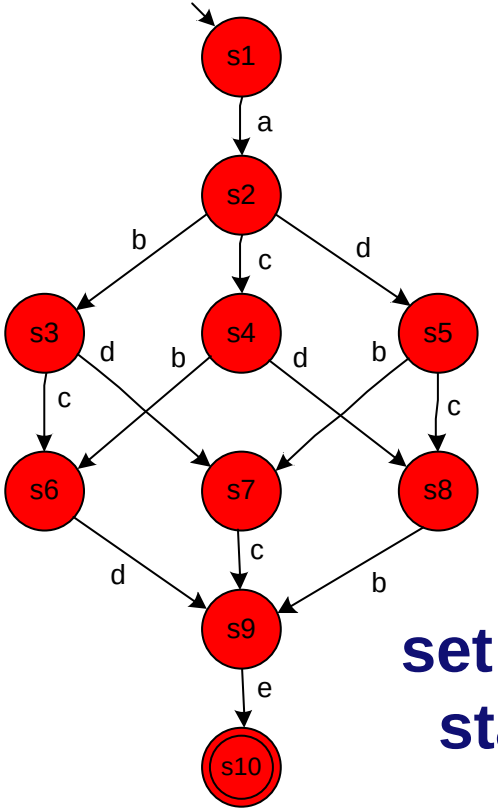
# Selecting the "interesting regions" only …

- **We only include the non-trivial minimal regions!**
- **Non-trivial**
  - **The empty set and the set of all states are regions by definition.**
  - **These trivial regions carry no information and should not be included.**
- **Minimal**
  - **A region is minimal if it cannot be decomposed into smaller (non-trivial) ones.**
  - **Non-minimal regions are implied by smaller ones and should not be included.**

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

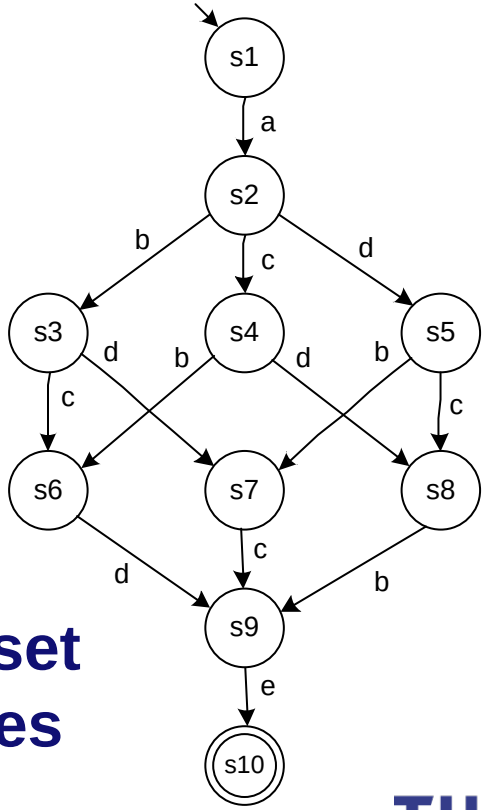
# Trivial regions: All or nothing




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

**set of all states**

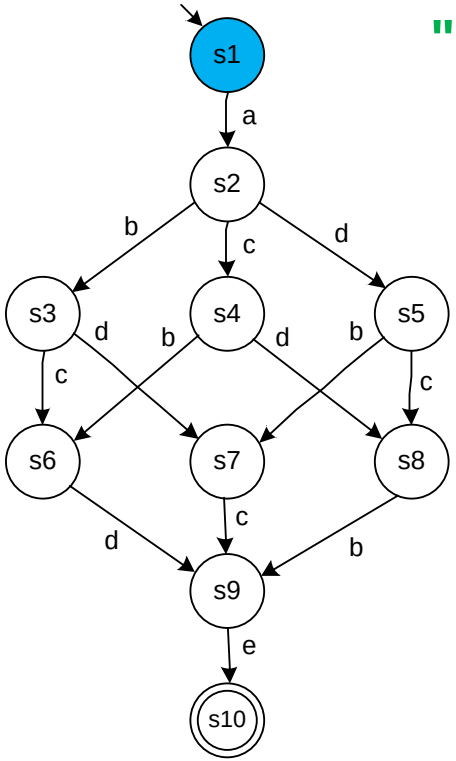
**empty set of states**



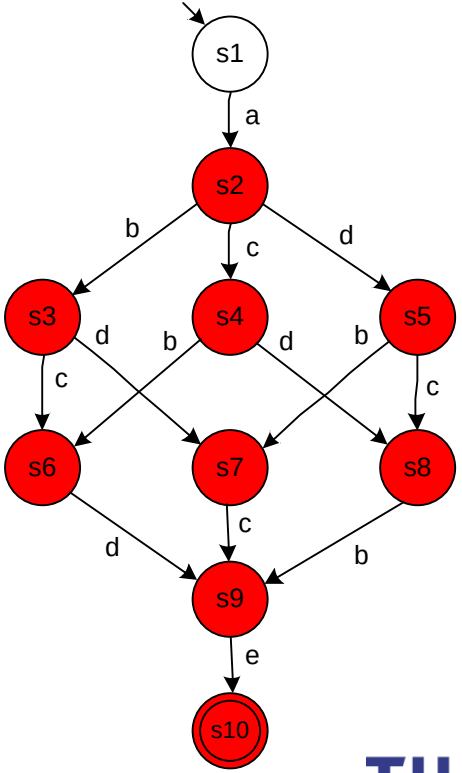

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

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



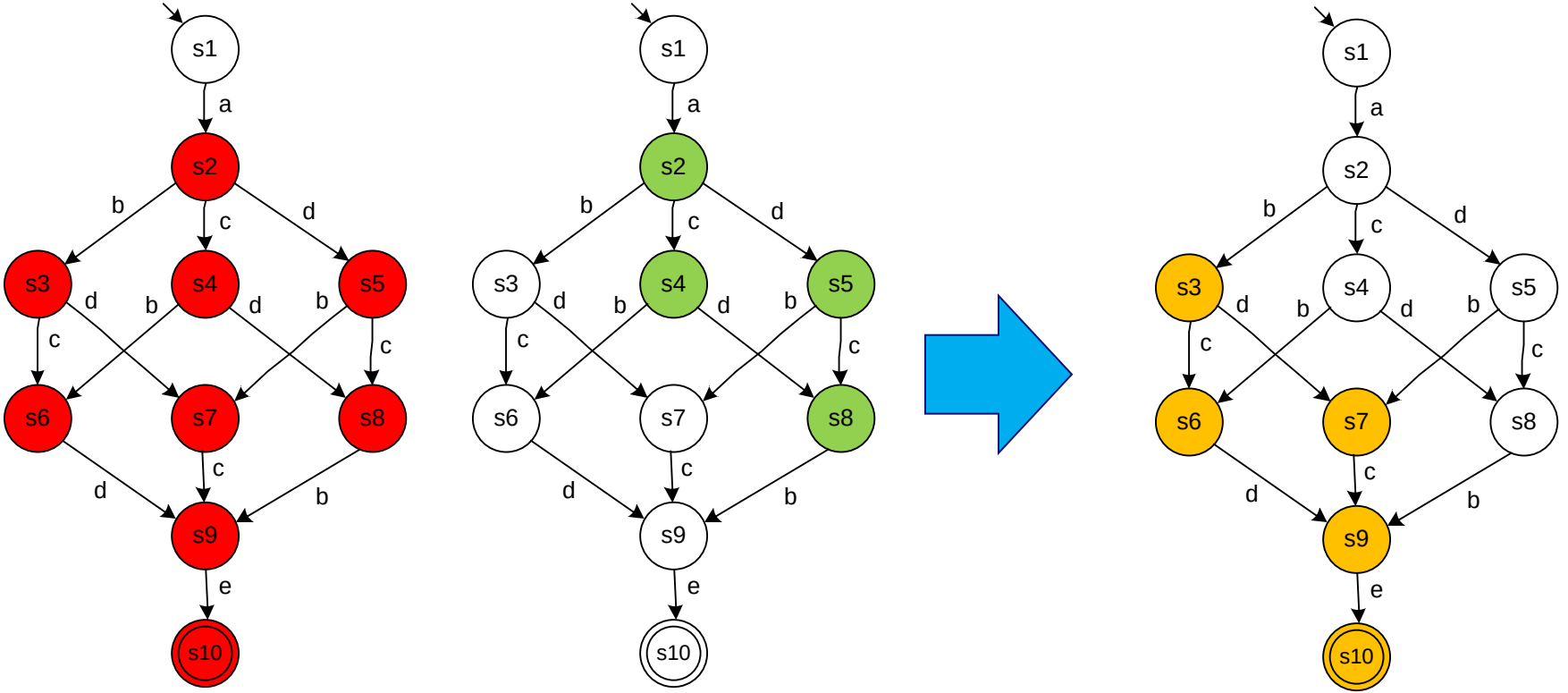

"exits" and "enters" are swapped




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

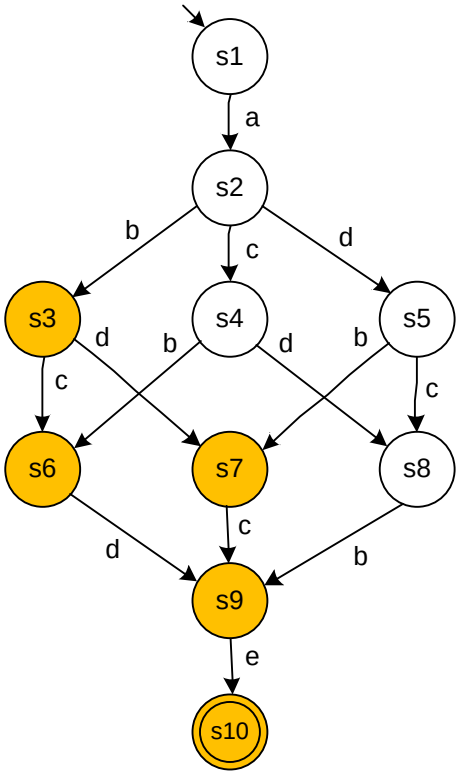
# If $r_1$ and $r_2$ are regions, and $r_1$ is a subset of $r_2$, then $r_2 \setminus r_1$ is a region.

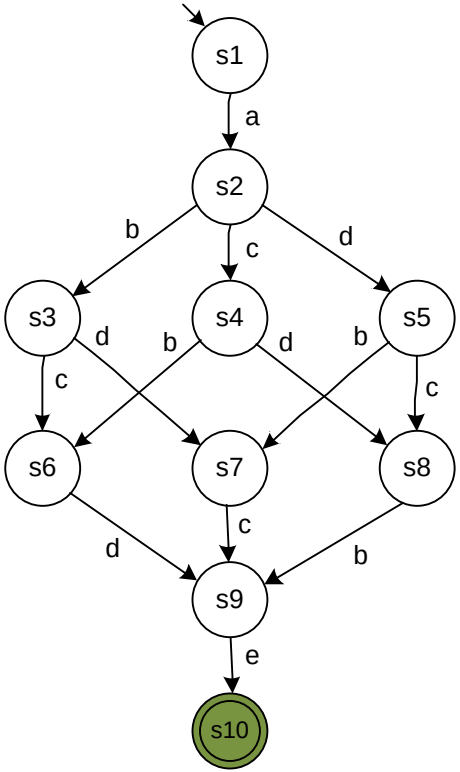


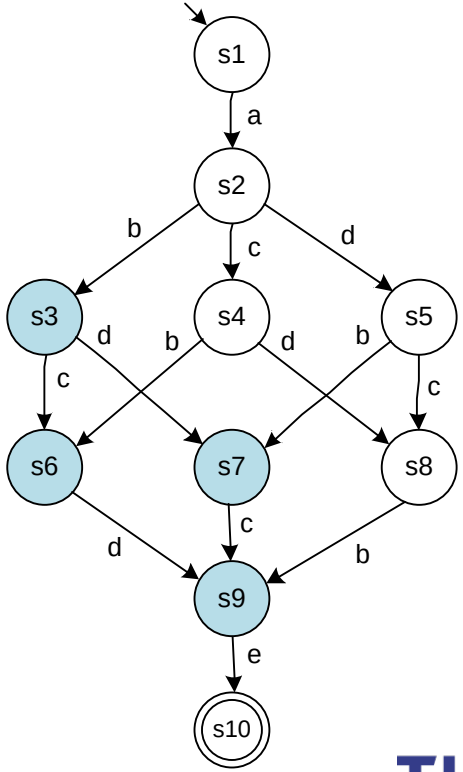

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Not minimal yet …

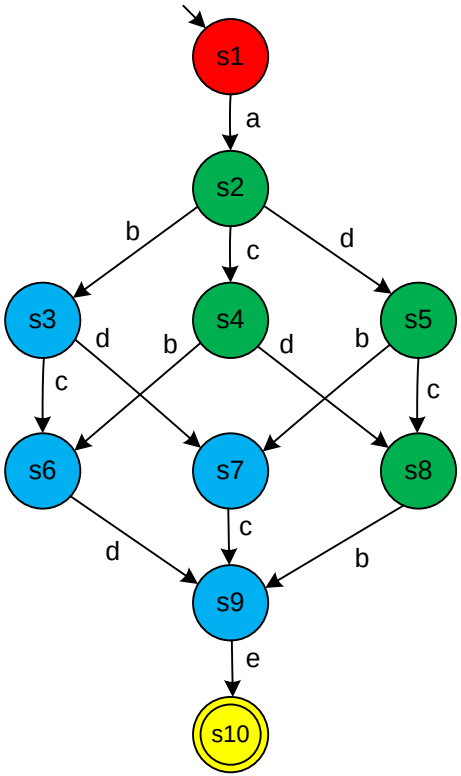


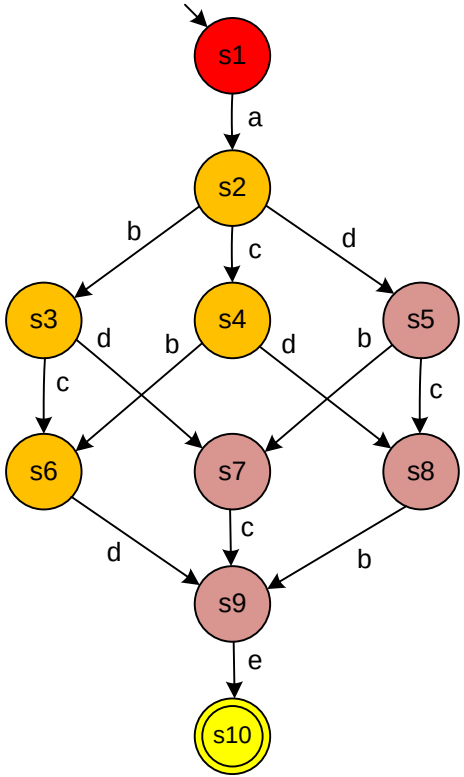



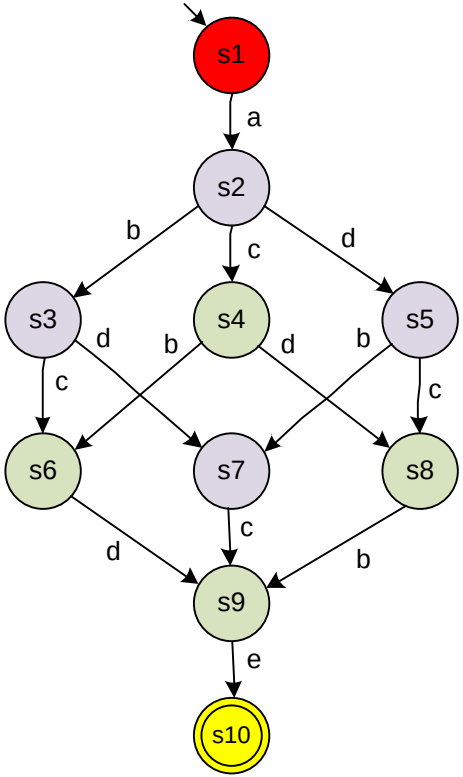




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

# Running example:
# 8 minimal non-trivial regions




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Basic algorithm to construct a Petri net

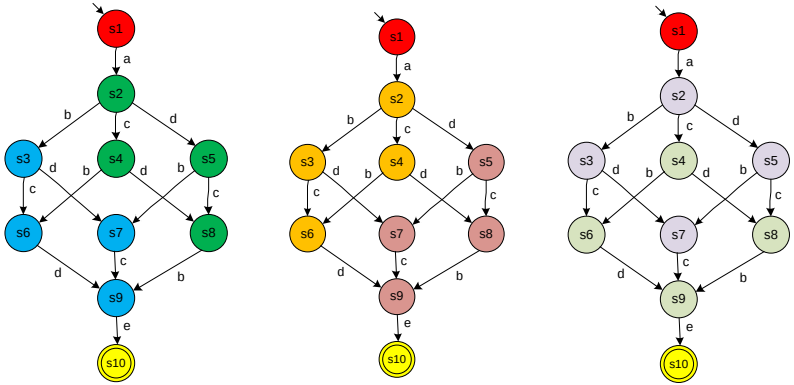
1. For each transition label in the transition system, a transition is added to the Petri net.
2. The minimal non-trivial regions are computed.
3. For each minimal non-trivial region in the transition system, a place is added to the Petri net.
4. The corresponding arcs are generated.
5. A token is added to each place that corresponds to a region containing the initial state.

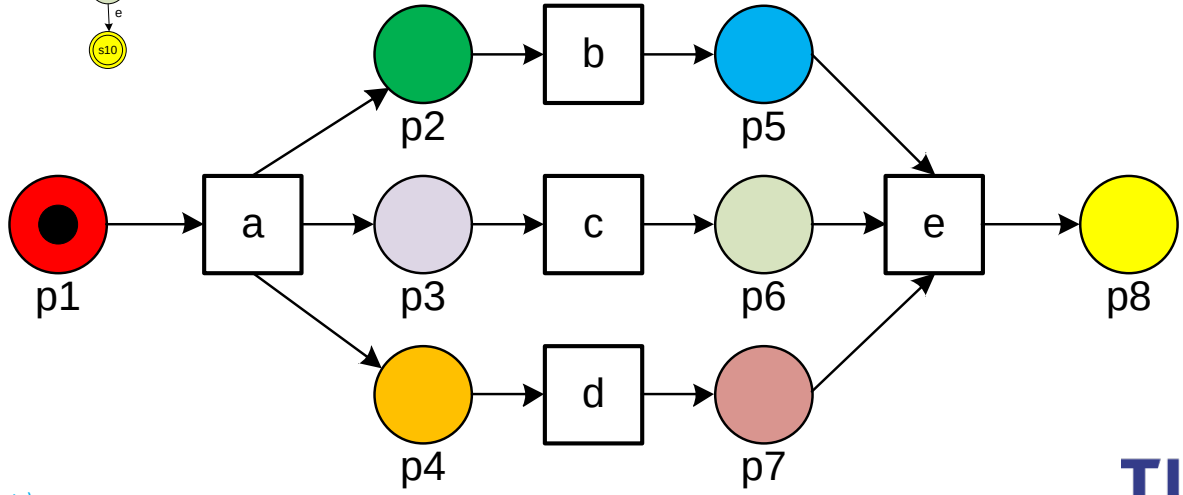
The resulting Petri net is called the minimal saturated net.

©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Applying the algorithm yields the expected result

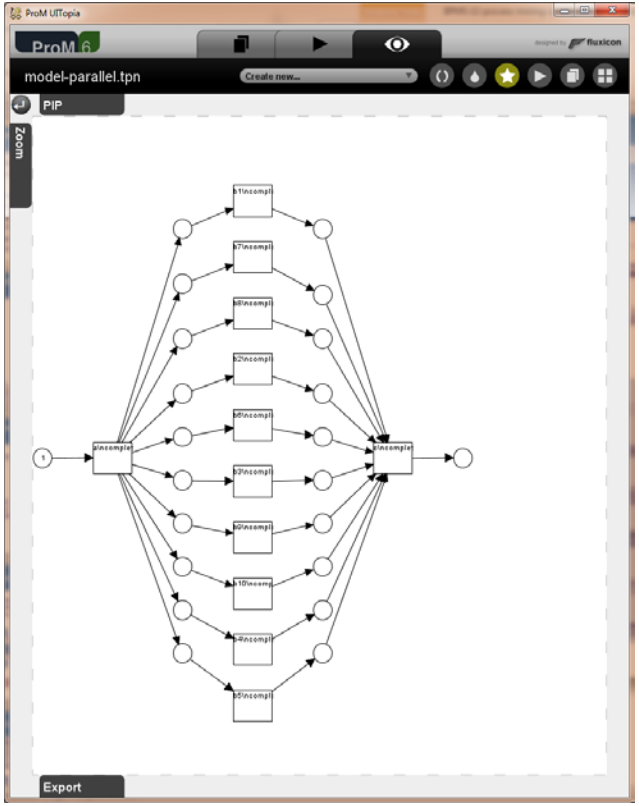


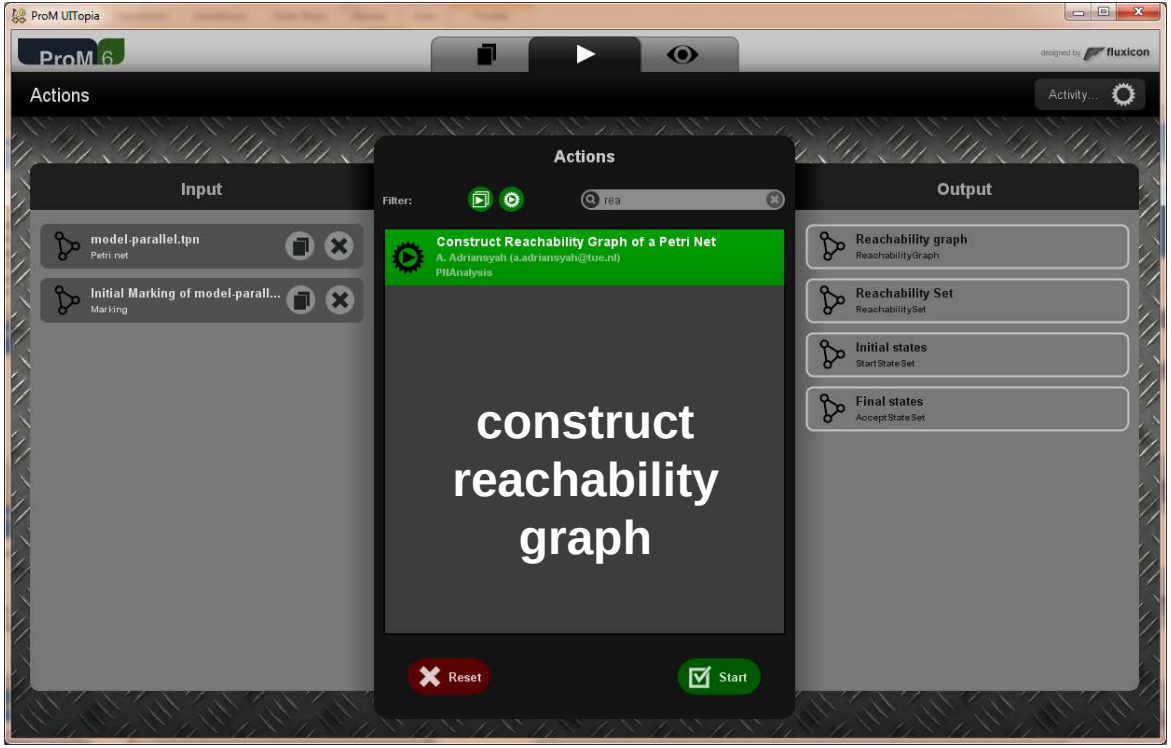




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

# Rediscovery experiment with 10 parallel activities



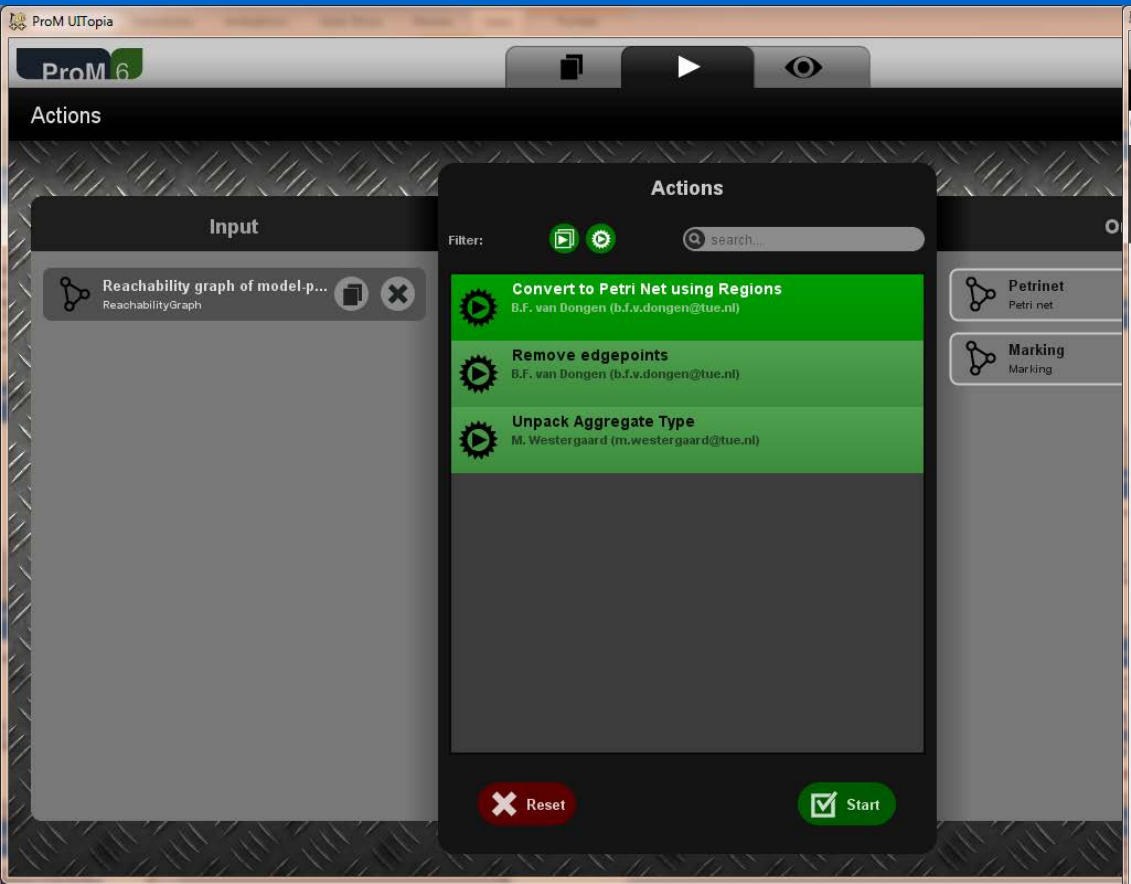

**past with multiset abstraction**

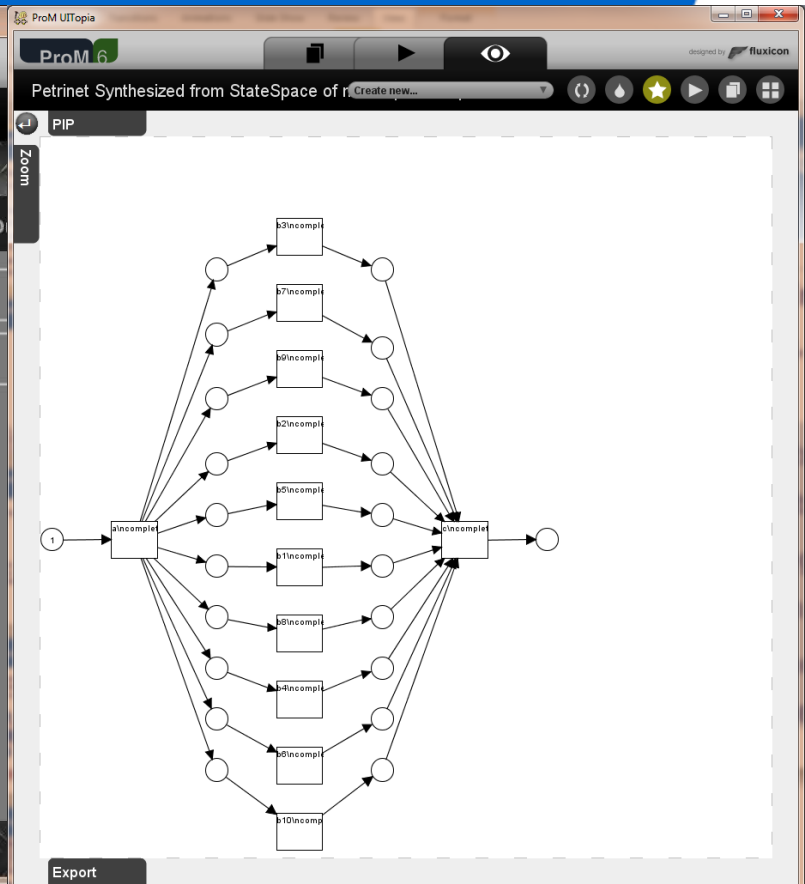
©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e

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

# Apply "Covert to Petri net using regions"

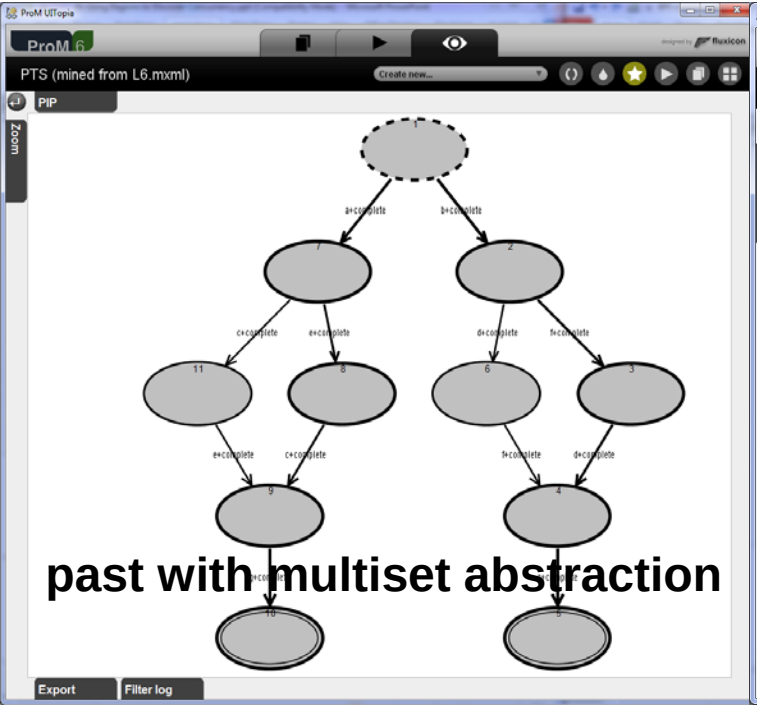


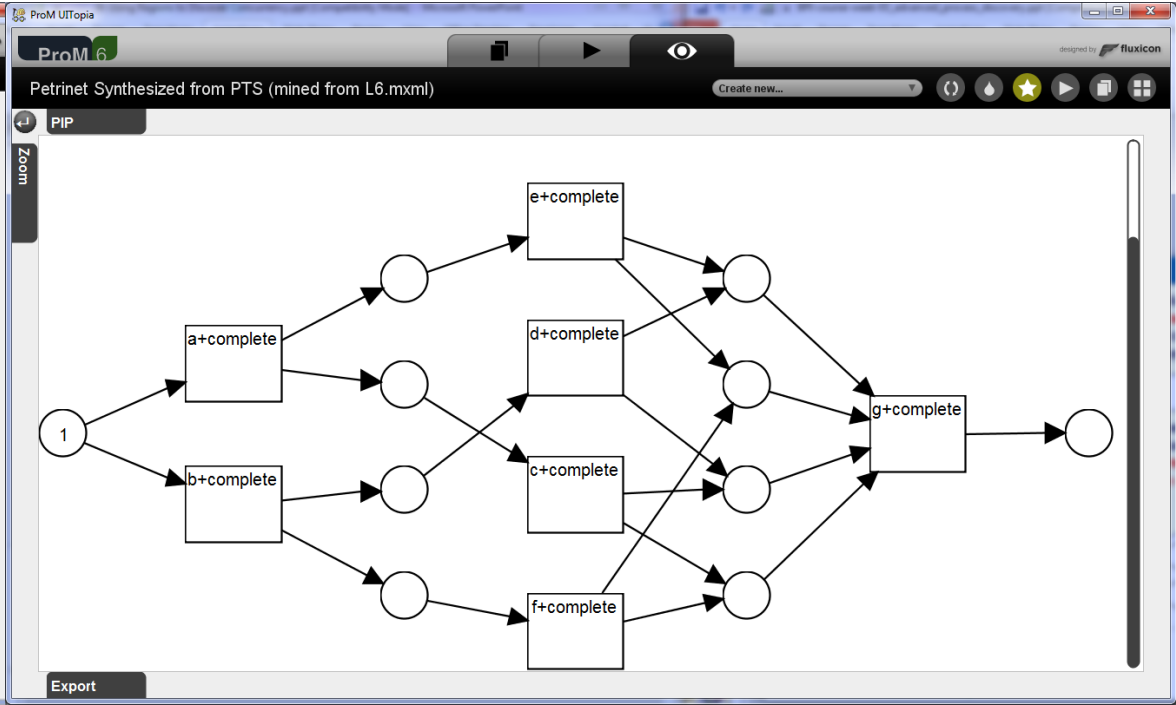
# Example

## (same result as Alpha algorithm)

$$L_6 = [\langle a,c,e,g\rangle^2, \langle a,e,c,g\rangle^3, \langle b,d,f,g\rangle^2, \langle b,f,d,g\rangle^4]$$

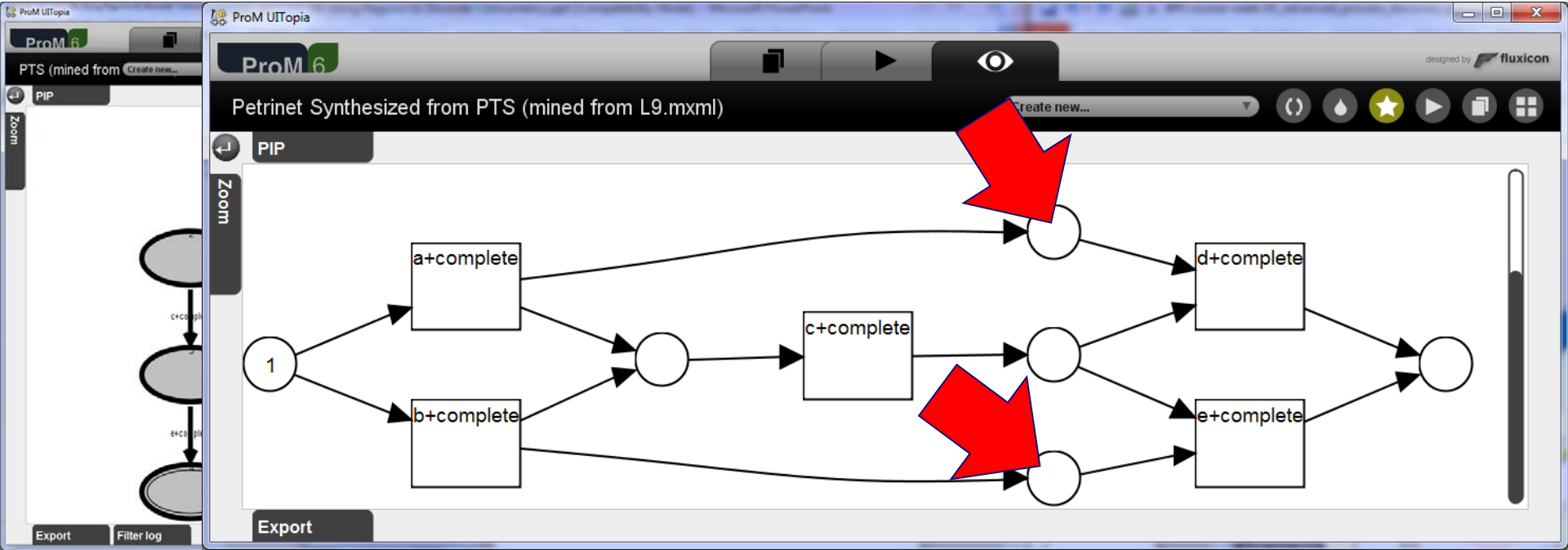



past with multiset abstraction

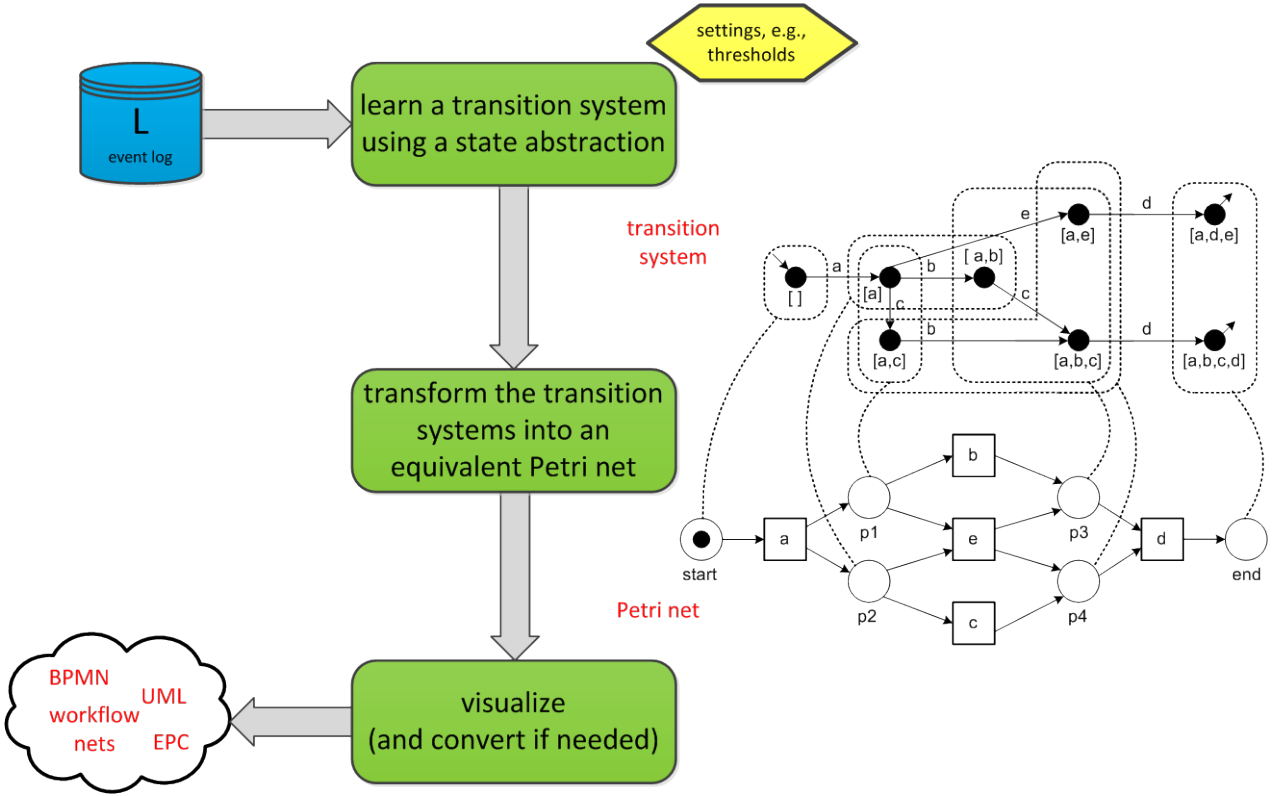


# Example
## (improved result compared to Alpha algorithm)

$$L_9 = [\langle a, c, d \rangle^{45}, \langle b, c, e \rangle^{42}]$$



# Next: limitations of the approach




©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

## Part I: Preliminaries

- Chapter 1: Introduction
- Chapter 2: Process Modeling and Analysis
- Chapter 3: Data Mining

## Part II: From Event Logs to Process Models

- Chapter 4: Getting the Data
- Chapter 5: Process Discovery: An Introduction
- Chapter 6: Advanced Process Discovery Techniques

## Part III: Beyond Process Discovery

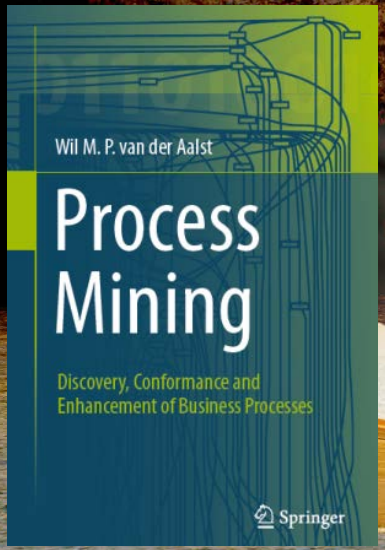
- Chapter 7: Conformance Checking
- Chapter 8: Mining Additional Perspectives
- Chapter 9: Operational Support

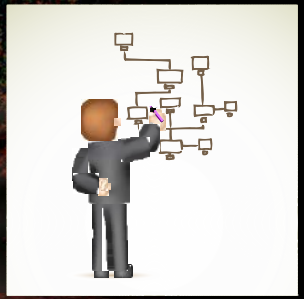
## Part IV: Putting Process Mining to Work

- Chapter 10: Tool Support
- Chapter 11: Analyzing "Lasagna Processes"
- Chapter 12: Analyzing "Spaghetti Processes"

## Part V: Reflection

- Chapter 13: Cartography and Navigation
- Chapter 14: Epilogue



©Wil van der Aalst & TU/e (use only with permission & acknowledgements)

TU/e