Finding Complex Process-Structures by Exploiting the Token-Game

Lisa L. Mannel, Wil M. P. van der Aalst
June 27th, 2019

Chair of Process and Data Science
RWTH AACHEN UNIVERSITY

Process Discovery

Introducing the Problem

Process Discovery - Introduction

Input: Event Log

- Multiset of *traces* (sequences) of *activities*
- Interpretable as a finite *language*

Output: Process Model

- Petri Nets (subset)

Example:

Set of Activities: $A = \{a, b, c, d, e, \blacktriangleright, \blacksquare\}$

$L = \{\{ (\blacktriangleright, a, c, d, \blacksquare)^3, (\blacktriangleright, b, c, e, \blacksquare)^5 \}\}$

Process Discovery - Introduction

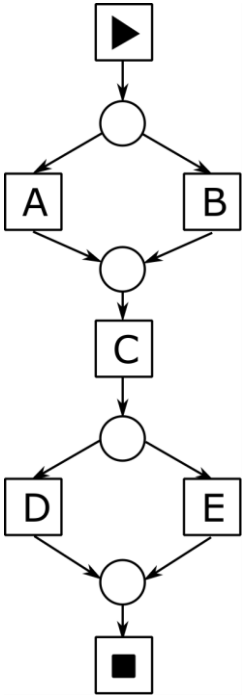
Input: Event Log

- Multiset of *traces* (sequences) of *activities*
- Interpretable as a finite *language*

Example:

Set of Activities: $A = \{a, b, c, d, e, ▶, ■\}$

$L = \{\{ (▶,a,c,d,■)^3, (▶,b,c,e,■)^5 \}\}$



Fitness

Precision

Simplicity

Noise Handling

Time Efficiency

Process Discovery - Introduction

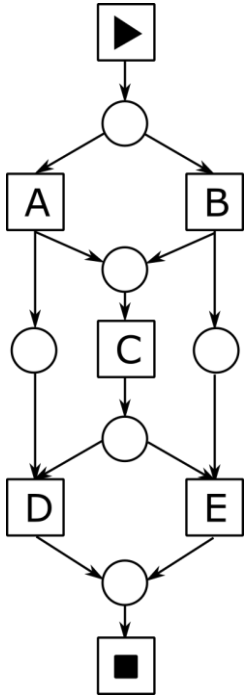
Input: Event Log

- Multiset of *traces* (sequences) of *activities*
- Interpretable as a finite *language*

Example:

Set of Activities: $A = \{a, b, c, d, e, ▶, ■\}$

$L = \{\{ (▶,a,c,d,■)^3, (▶,b,c,e,■)^5 \}\}$



Fitness

Precision

Simplicity

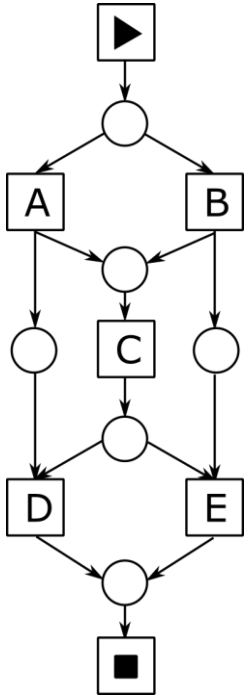
Noise Handling

Time Efficiency

Process Discovery – Related Work

Region-based approaches

- High fitness & precision
- Can find complex structures
- Low simplicity (‘Spaghetti’-Models)
- Cannot handle infrequent behavior



Fitness

Precision

Simplicity

Noise Handling

Time Efficiency

Process Discovery – Related Work

Region-based approaches

- High fitness & precision
- Can find complex structures
- Low simplicity (‘Spaghetti’-Models)
- Cannot handle infrequent behavior

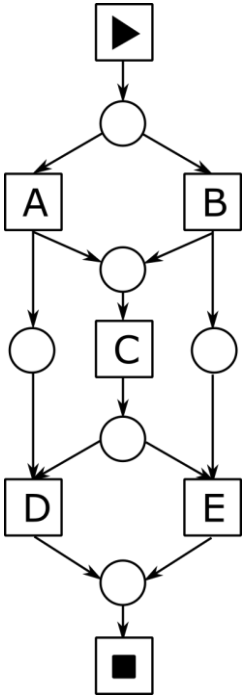
Our approach

(inspired by language-based regions)

- High fitness & precision
- Can find complex structures

→ **Improve simplicity**

→ **Handle infrequent behavior**



Fitness

Precision

Simplicity

Noise Handling

Time Efficiency

Our Approach

An Overview

Our Approach – General Idea

1. Input: Log & Threshold
   - Special start/end activities attached to each trace
   - Initialize Petri net without places
   - Each transition corresponds to one activity

2. Evaluate all possible places
   - find all fitting places
   - high fitness and precision

3. Post-processing
   - Remove implicit places

Our Approach – General Idea

1. Input: Log & Threshold
   - Special start/end activities attached to each trace
   - Initialize Petri net without places
   - Each transition corresponds to one activity

2. **Evaluate all possible places**
   - **find all fitting places**
   - **high fitness and precision**

3. Post-processing
   - Remove implicit places

Our Approach – General Idea

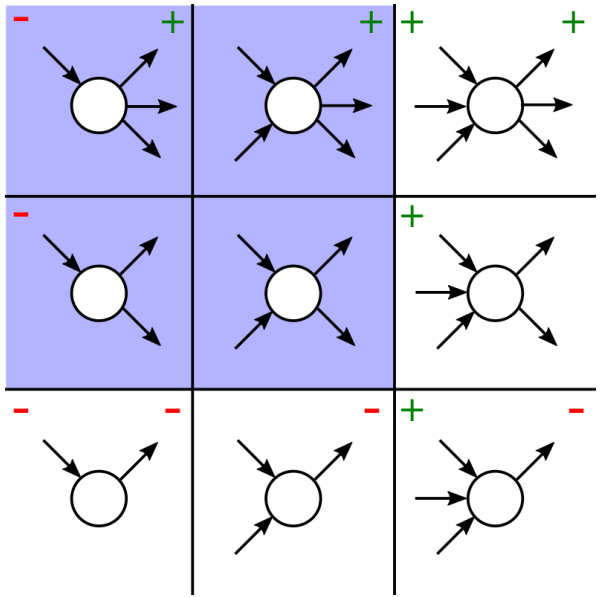
1. Input: Log & Threshold
   - Special start/end activities attached to each trace
   - Initialize Petri net without places
   - Each transition corresponds to one activity
2. **Evaluate all possible places**
   - **find all fitting places**
   - **high fitness and precision**
3. Post-processing
   - Remove implicit places

- Definition of Places:
  (I | O) – incoming & outgoing activities
- Number of candidate places:
  $|\mathbf{P}(A)| * |\mathbf{P}(A)| \rightarrow 2^{|A|} * 2^{|A|}$
  Exponential in the number of activities!
  For example: $|A|=10 \rightarrow 1{,}048{,}576$
- Brute Force: play the token game for each trace on each candidate place

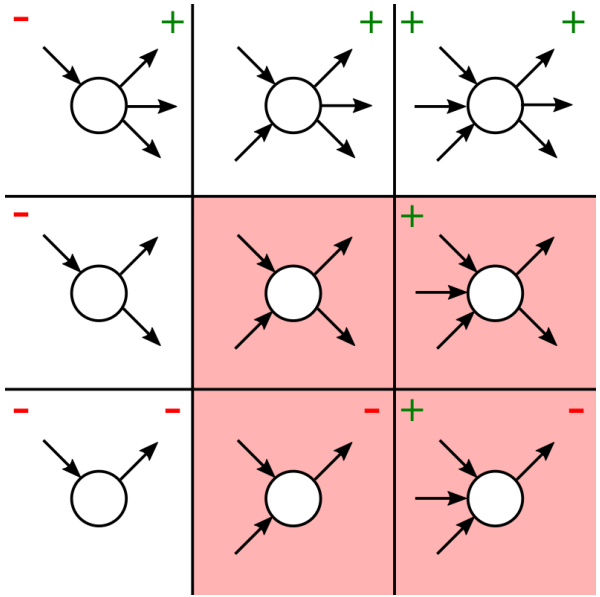
→ **Increase efficiency**
→ **Handle infrequent behavior**

Our Approach – Monotonicity Results

Underfed Places



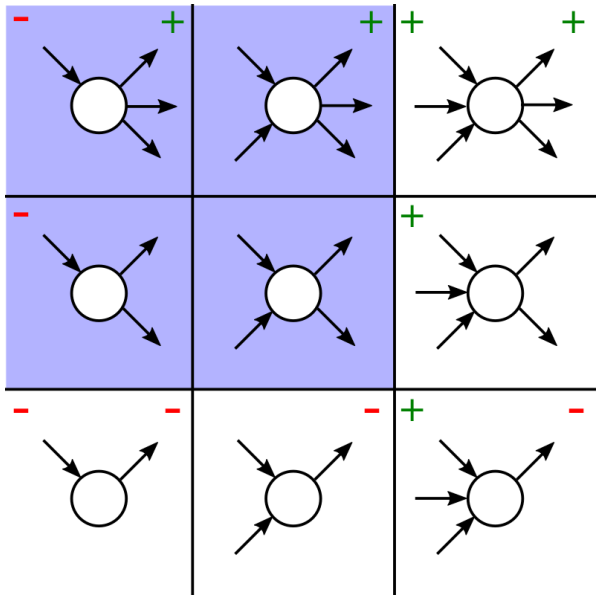
Overfed Places



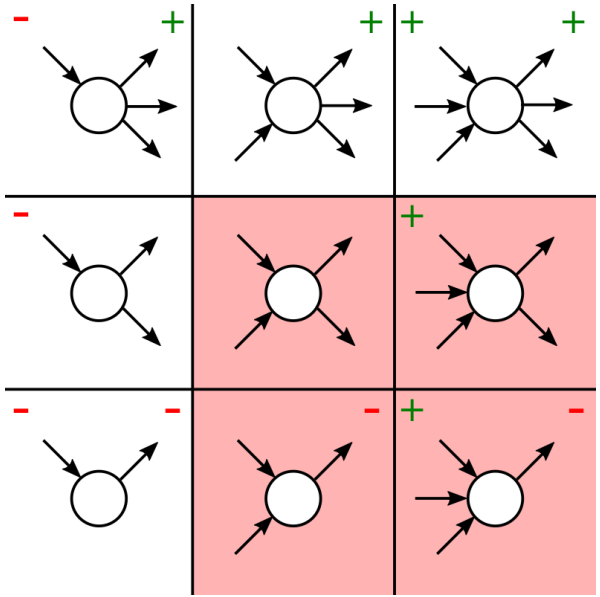
Our Approach – Monotonicity Results

Use threshold for infrequent behavior!

Underfed Places



Overfed Places



Our Approach – Running Example

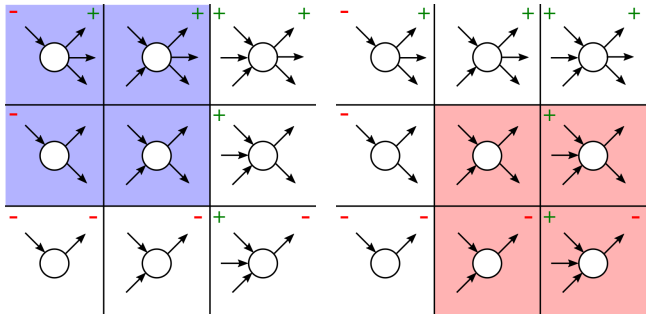
L = {{ (▶,*a*,*b*,■) }} → Transitions = {▶,*a*,*b*,■}

Our Approach – Running Example

L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}

Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$



(►|a) (►|b) (►|■) (a|a) (a|b) (a|■) (b|a) (b|b) (b|■)

(►|a,b) (►|a,■) (►|b,■) (a|a,b) (a|a,■) (a|b,■) (b|a,b) (b|a,■) (b|b,■)

(►,a|a) (►,b|a) (a,b|a) (►,a|b) (►,b|b) (a,b|b) (►,a|■) (►,b|■) (a,b|■)

(►|a,b,■) (a|a,b,■) (b|a,b,■) (►,a|a,b) (►,a|a,■) (►,a|b,■) (►,b|a,b) (►,b|a,■) (►,b|b,■) (a,b|a,b) (a,b|a,■) (a,b|b,■) (►,a,b|a) (►,a,b|b) (►,a,b|■)

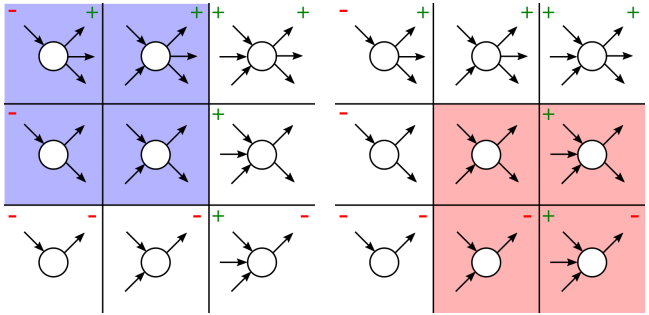
(►,a|a,b,■) (►,b|a,b,■) (a,b|a,b,■) (►,a,b|a,b) (►,a,b|a,■) (►,a,b|b,■)

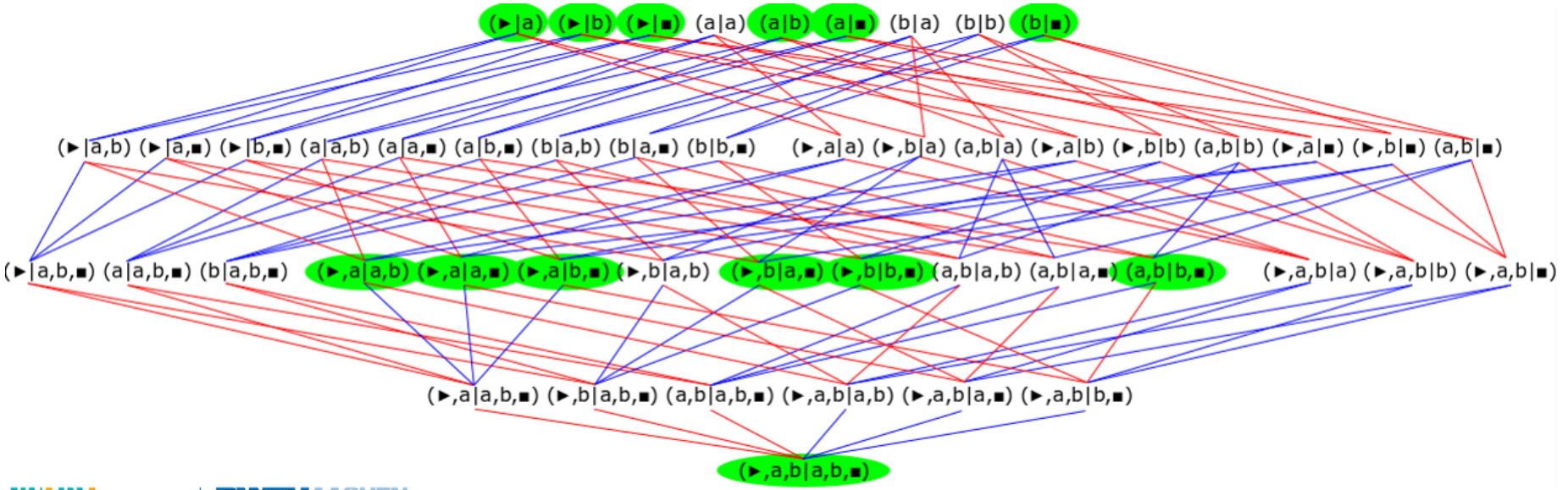
(►,a,b|a,b,■)

Chair of Process and Data Science | RWTH AACHEN UNIVERSITY

Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$



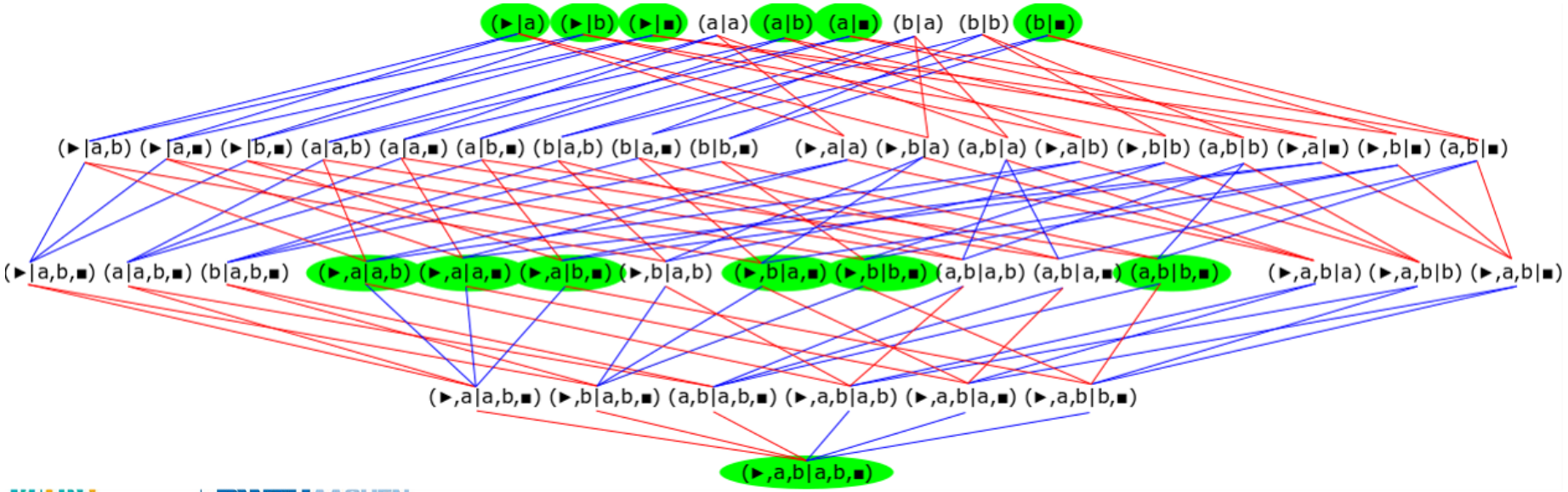


Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$

Traversal Strategy:

- Exploit monotonicity
- Find all fitting places
- Visit each place at most once
- Limited storage



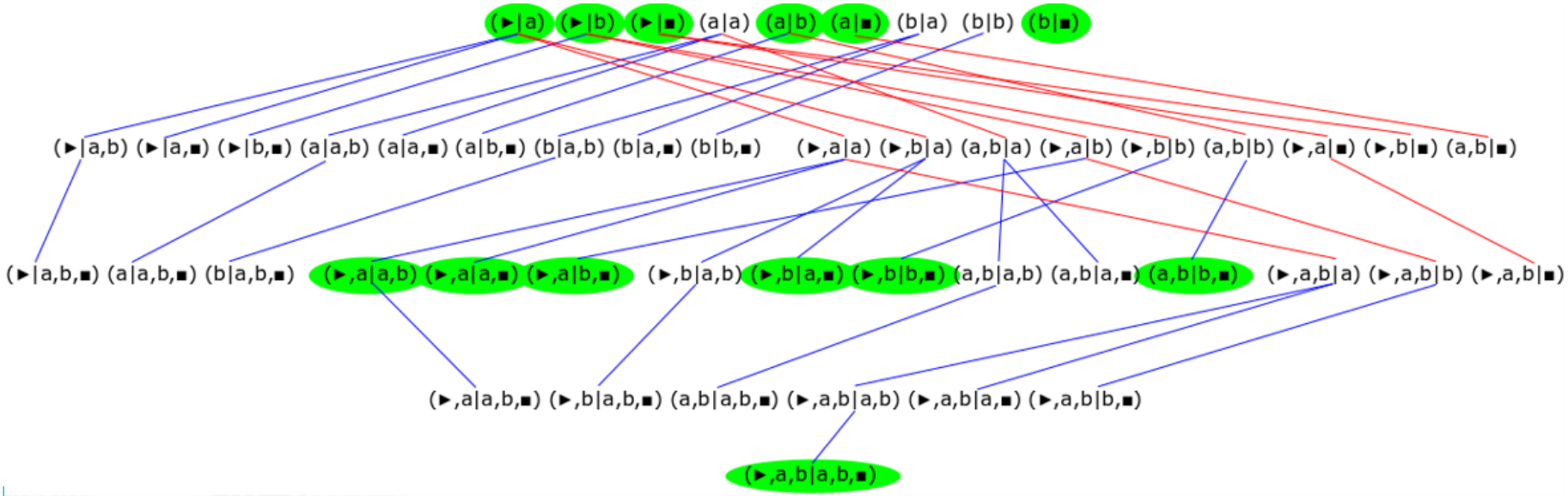
Chair of Process and Data Science | RWTH AACHEN UNIVERSITY

Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$

Traversal Strategy:

- Exploit monotonicity
- Find all fitting places
- Visit each place at most once
- Limited storage

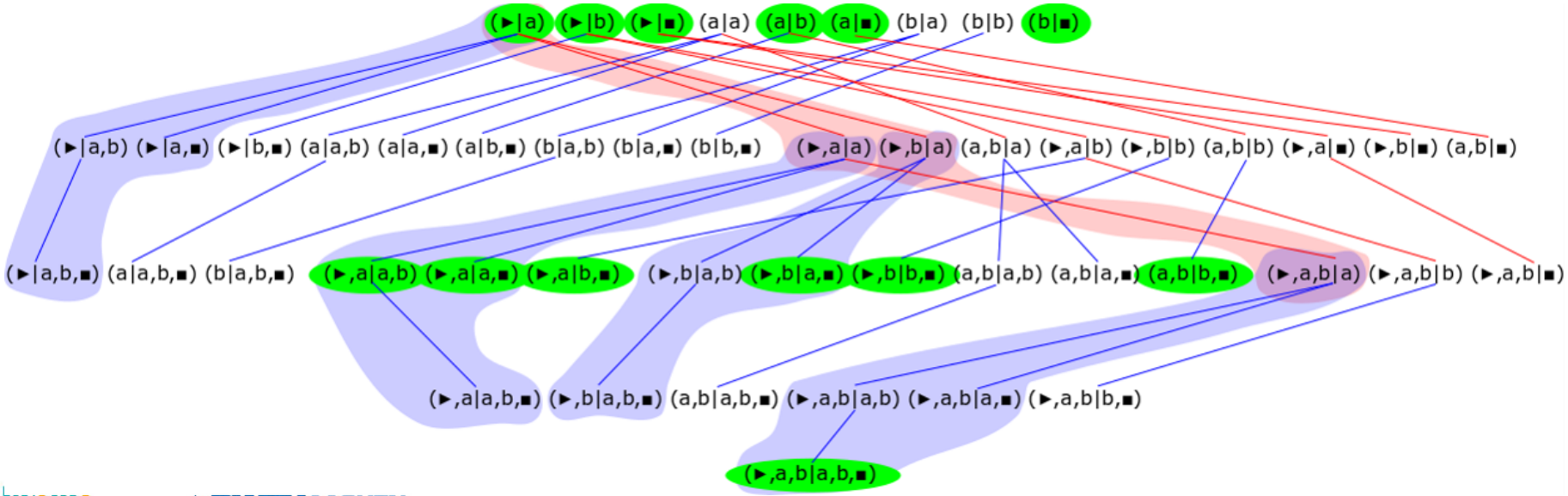


Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$

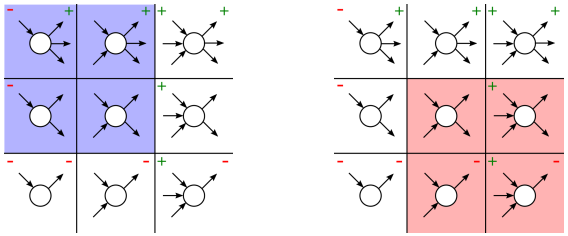
Traversal Strategy:

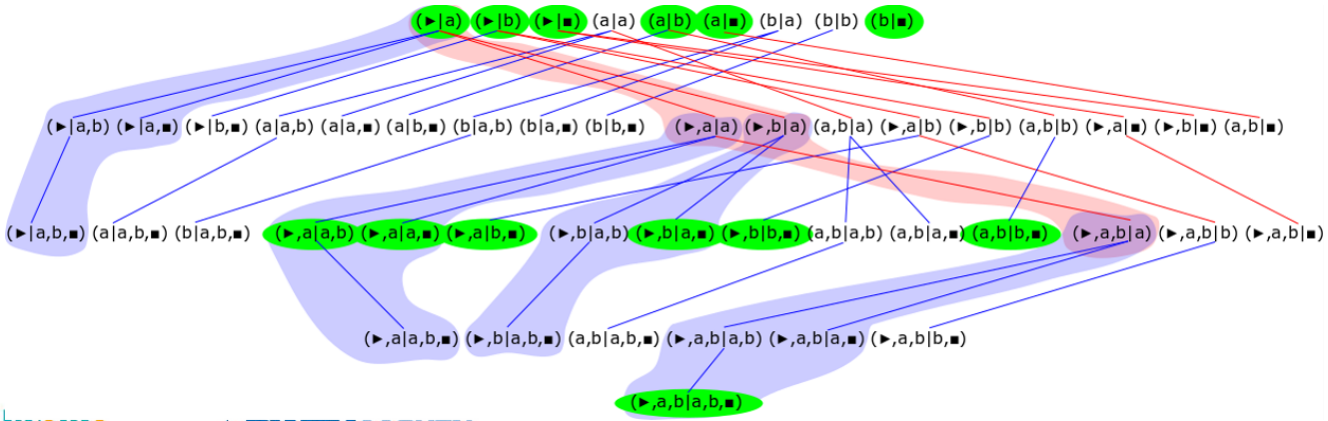
- Exploit monotonicity
- Find all fitting places
- Visit each place at most once
- Limited storage



Our Approach – Running Example

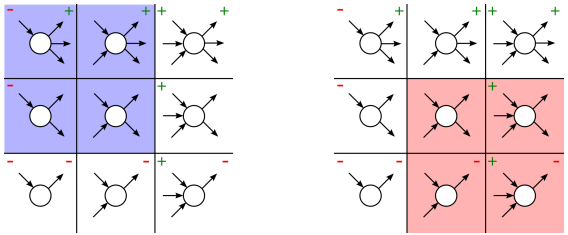
L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}

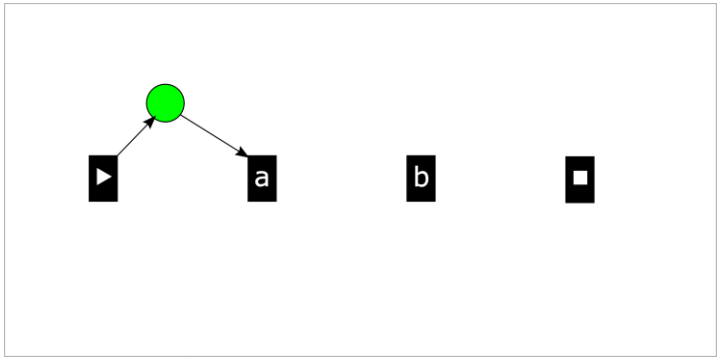


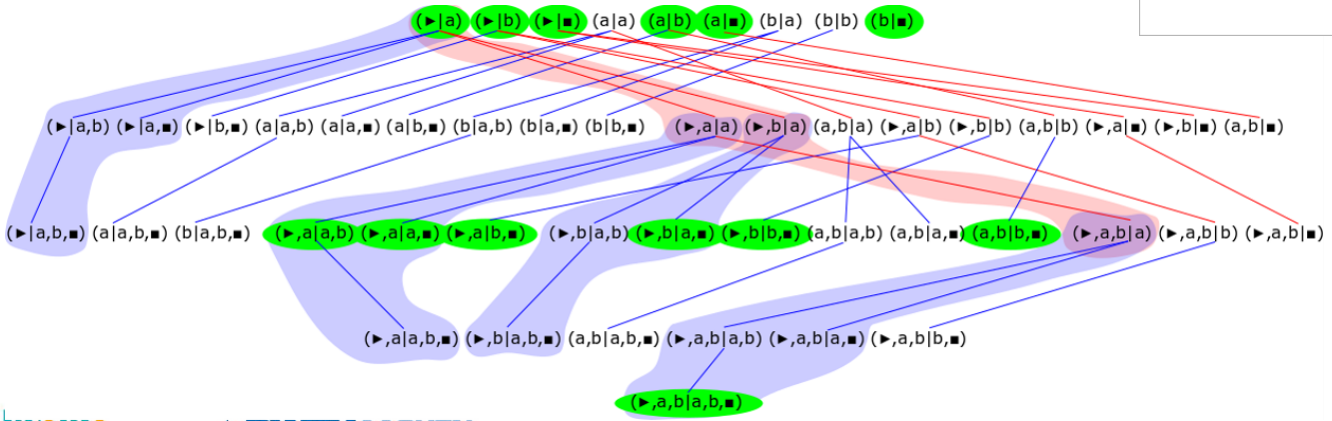


Our Approach – Running Example

L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}

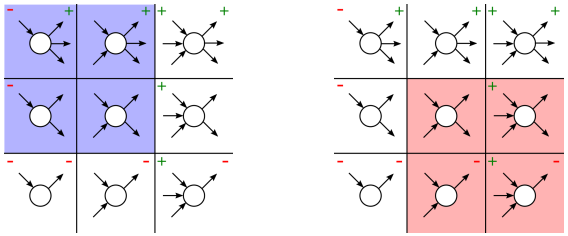


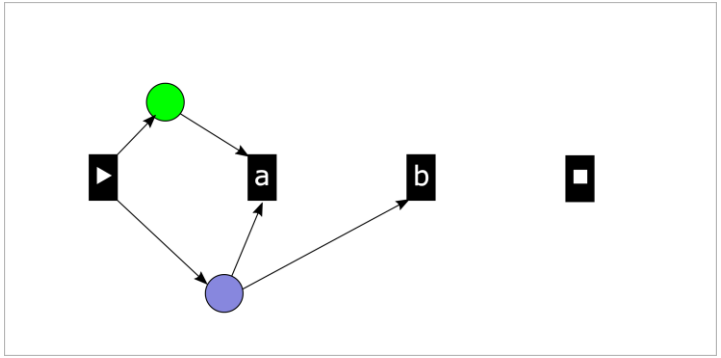


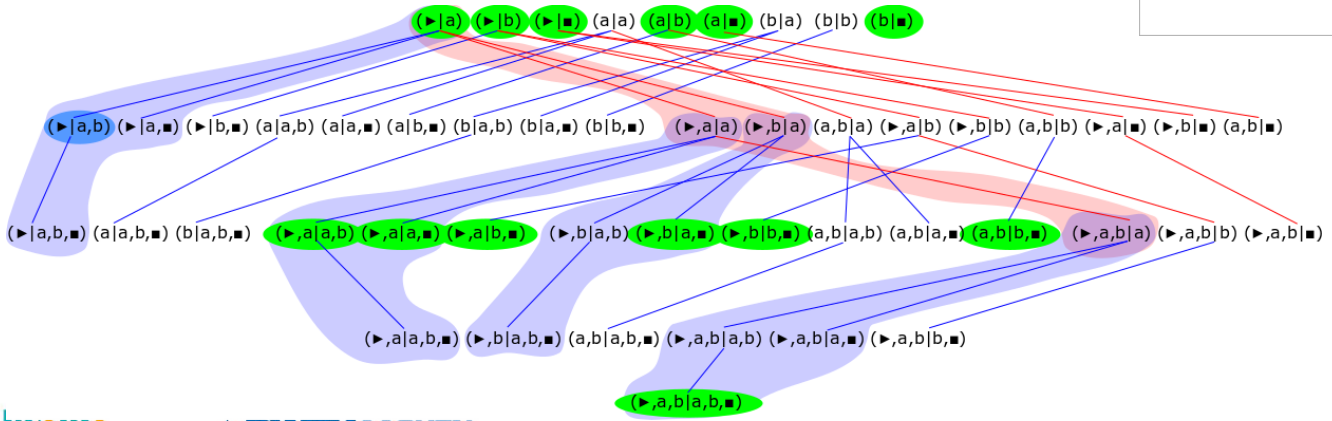


Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$



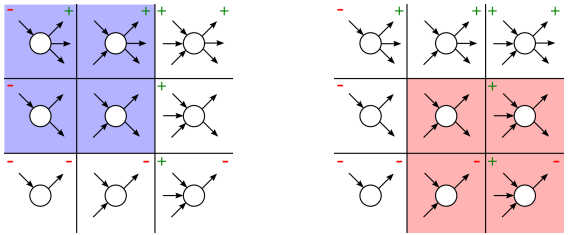


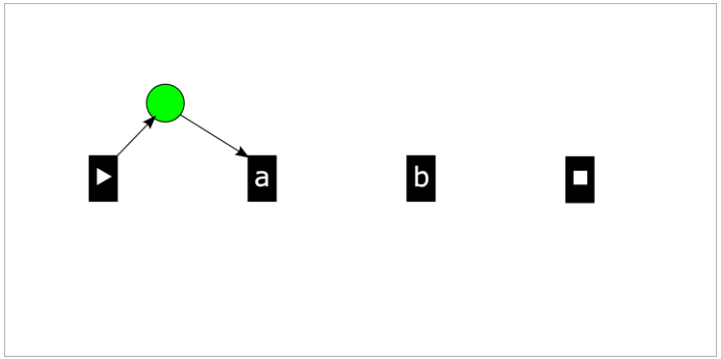


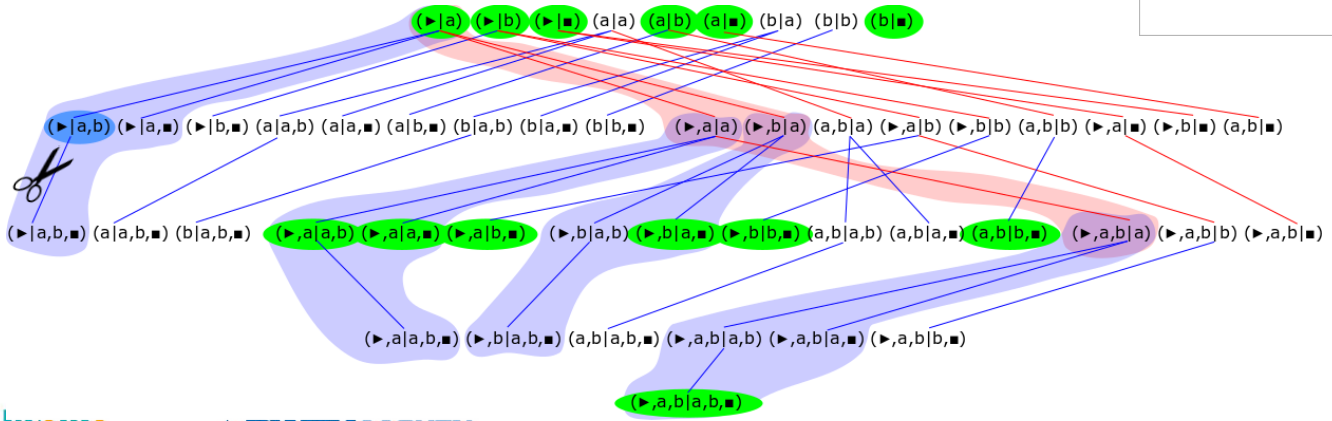
Chair of Process and Data Science | RWTH AACHEN UNIVERSITY

Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$

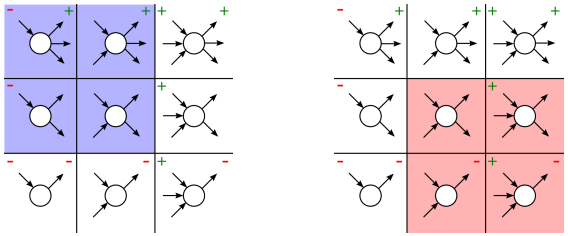


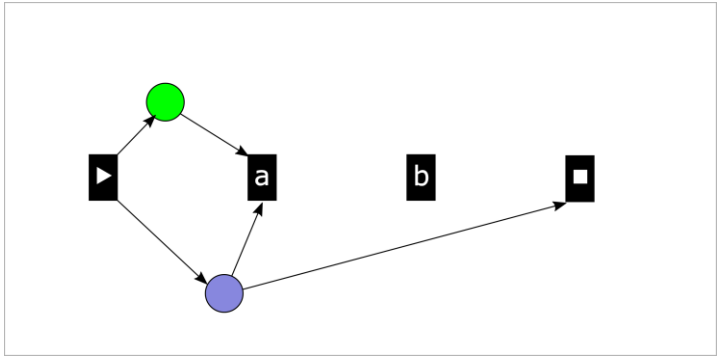


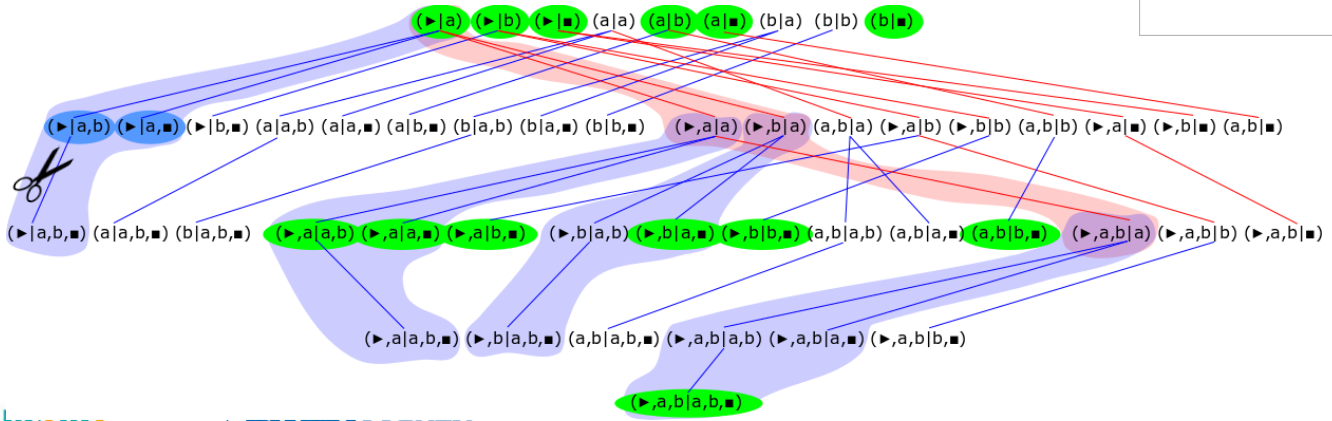


Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$

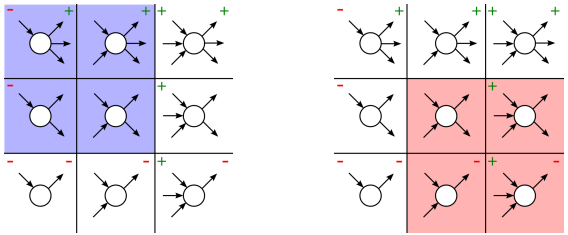


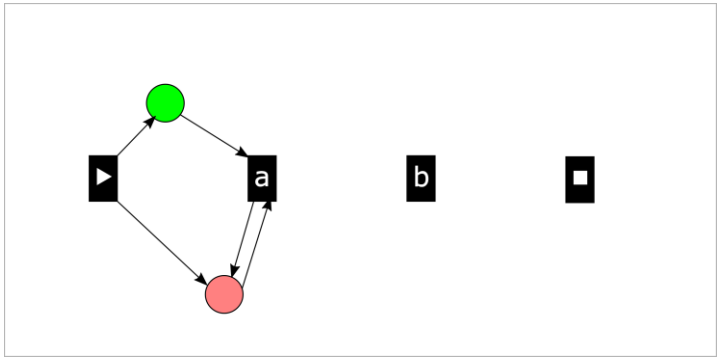


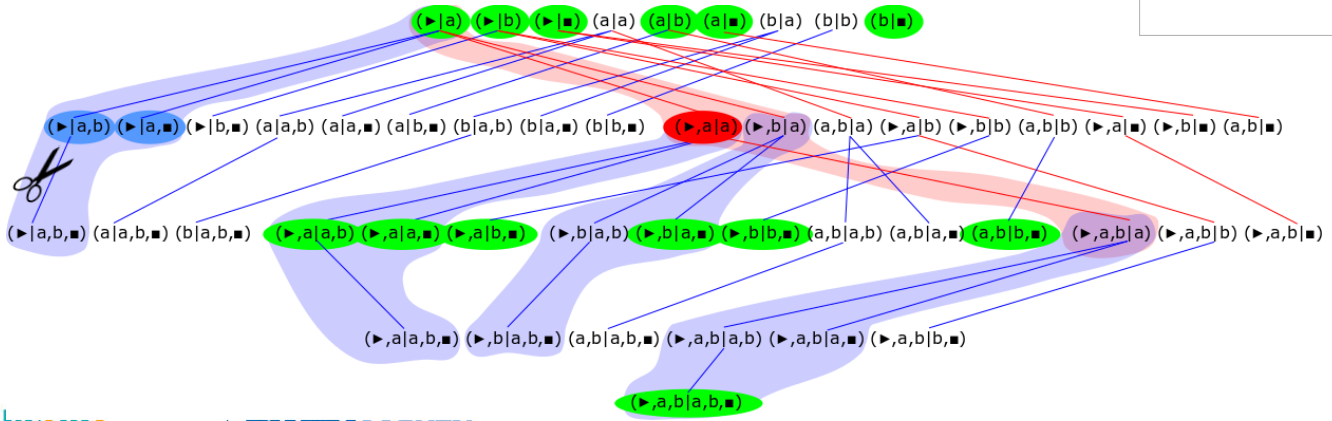


Our Approach – Running Example

L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}

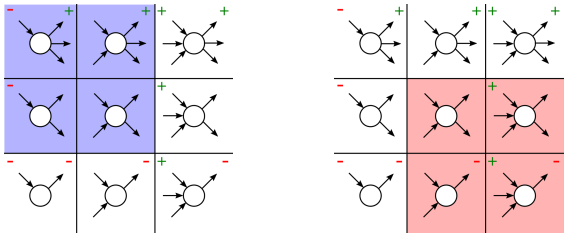


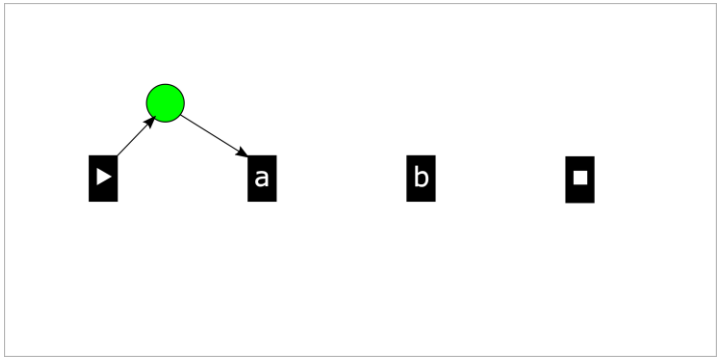


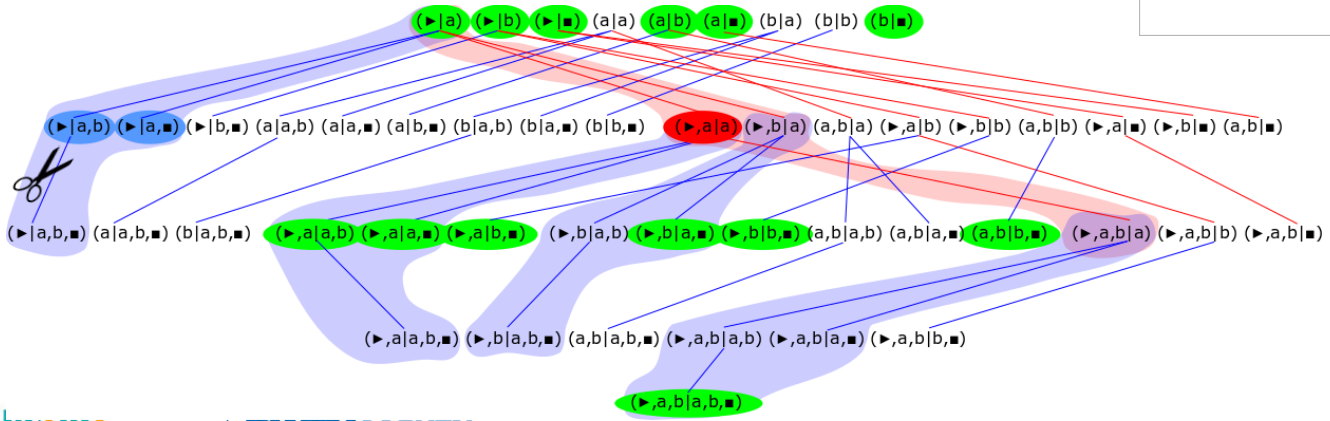


Our Approach – Running Example

L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}

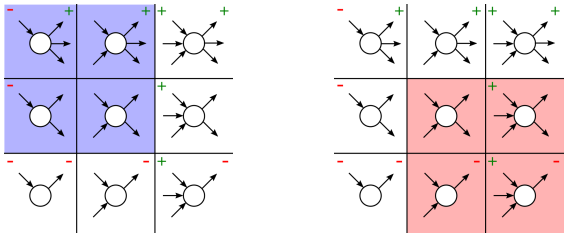


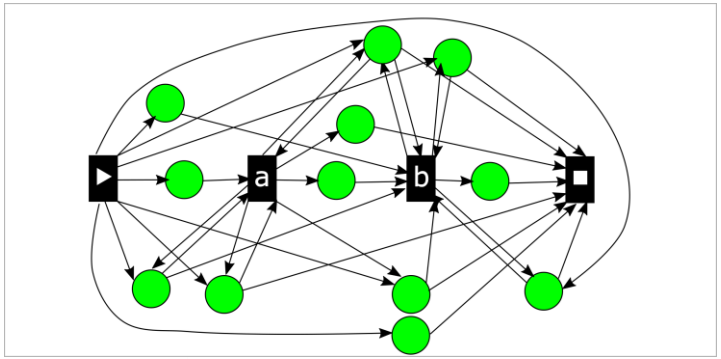


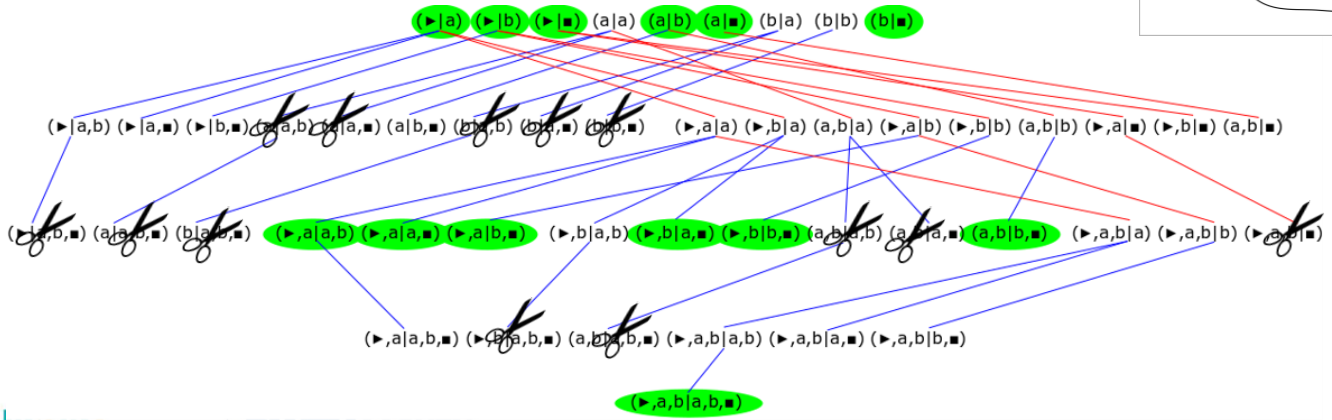


Our Approach – Running Example

L = {{ (►,*a*,*b*,■) }} → Transitions = {►,*a*,*b*,■}



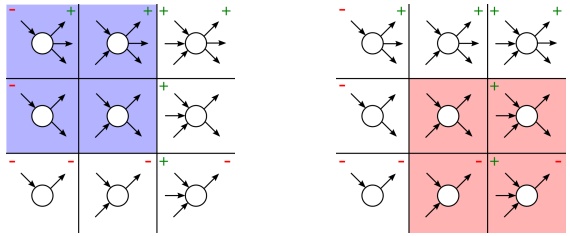


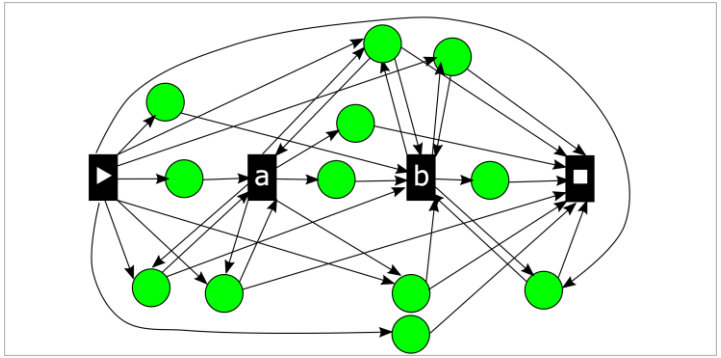


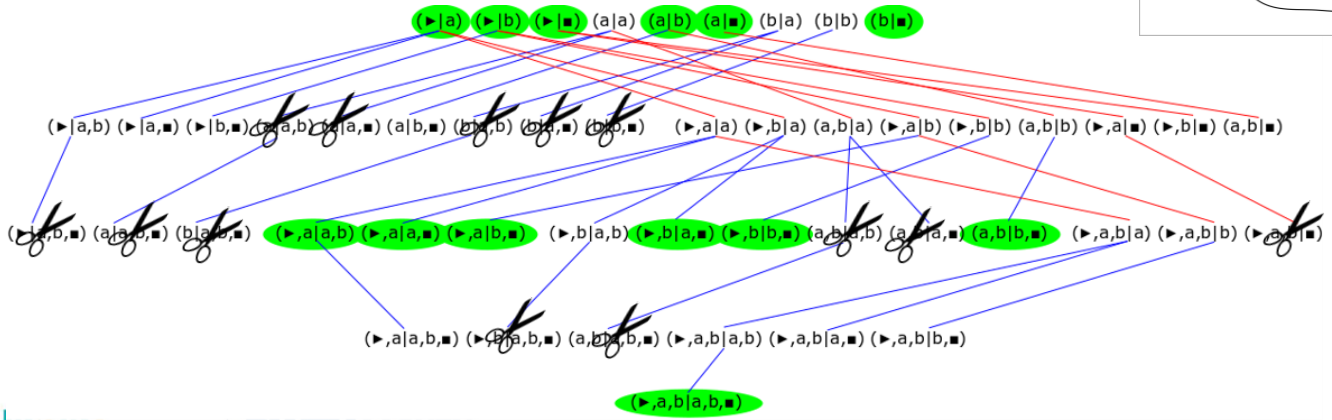
Chair of Process and Data Science | RWTH AACHEN UNIVERSITY

Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$





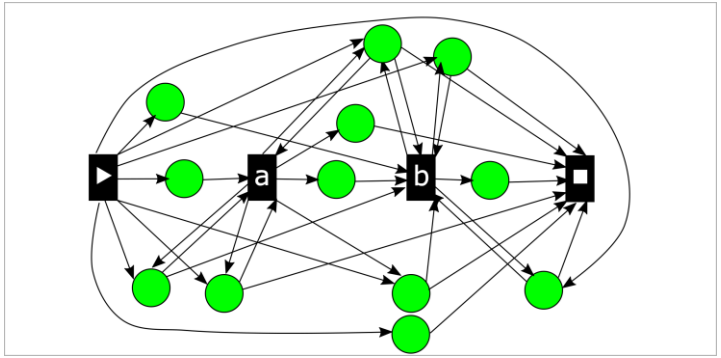


Deterministic DFS based on activity ordering

→ ordering determines position of place in tree

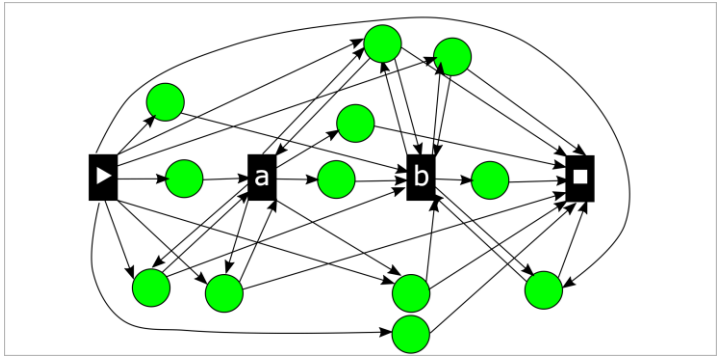
Our Approach – Running Example

L = {{ (▶,*a*,*b*,■) }} → Transitions = {▶,*a*,*b*,■}

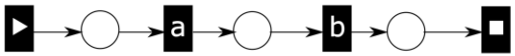


Our Approach – Running Example

$L = \{\{ (\blacktriangleright, a, b, \blacksquare) \}\}$ → Transitions = $\{\blacktriangleright, a, b, \blacksquare\}$



Post-processing:
removal of implicit places
(existing approaches)



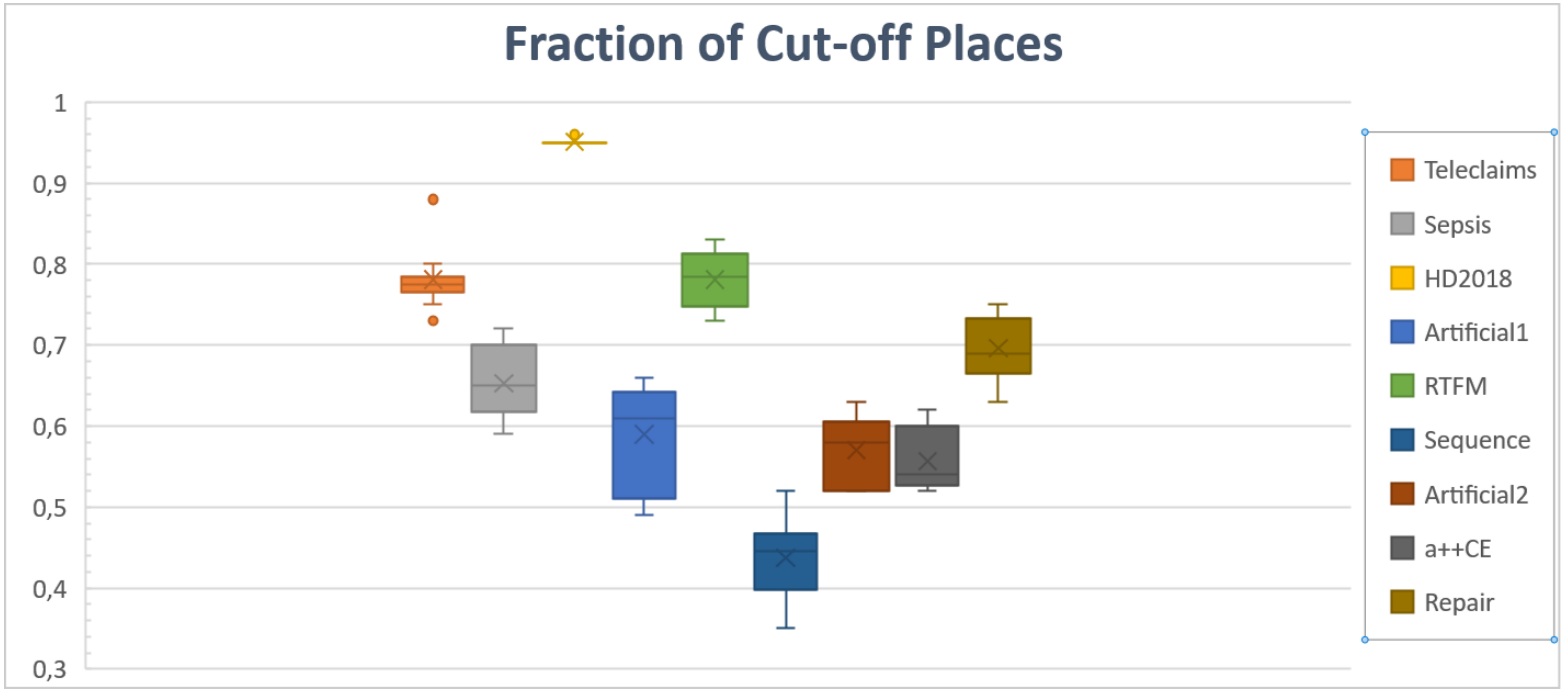
Our Approach

Evaluation

Our Approach – Evaluation

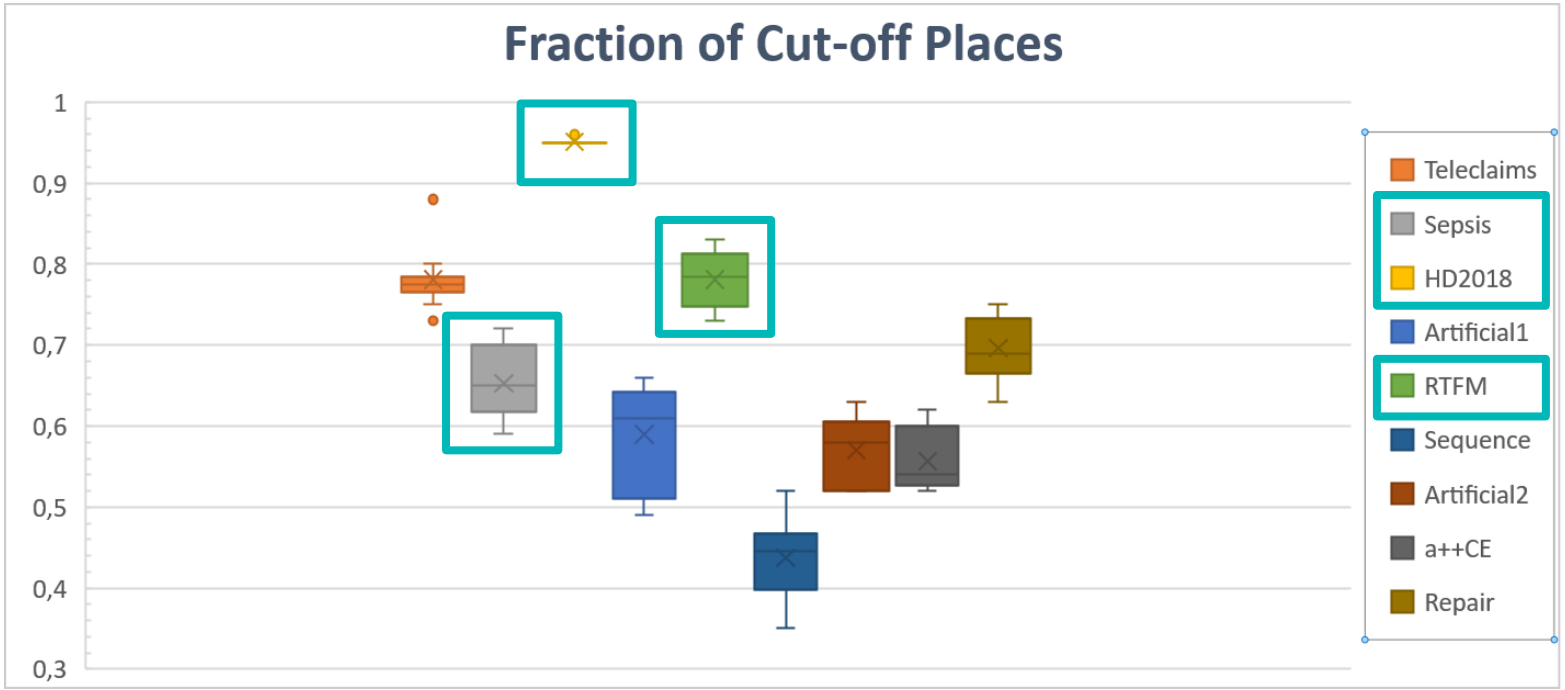
- Implemented in ProM, using Java
- **Focus on computation of fitting places**
  (post-processing based on existing results)

Our Approach – Evaluation



(experiments using randomized activity orderings, noise threshold = 1)

Our Approach – Evaluation



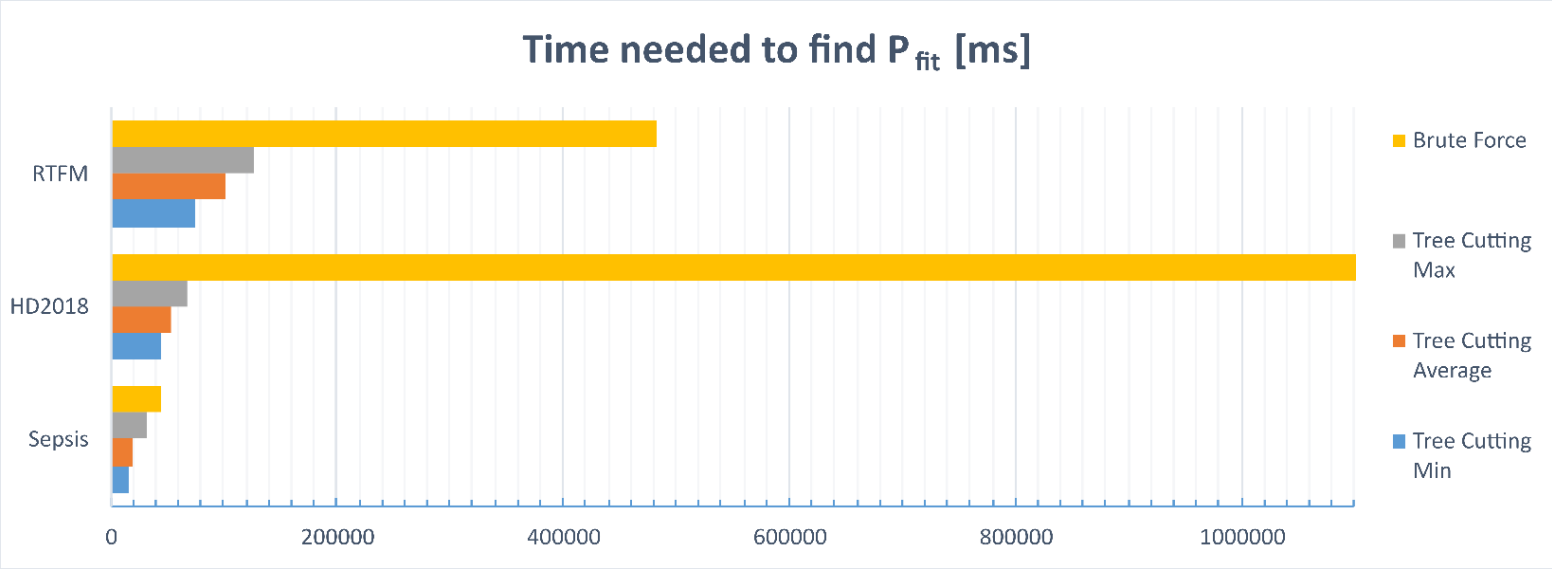
(experiments using randomized activity orderings, noise threshold = 1)

Our Approach – Evaluation

Cut-off: 0.8
#Activities: 13
#Traces: 231

Cut-off: 0.95
#Activities: 11
#Traces: 595

Cut-off: 0.6
#Activities: 11
#Traces: 27



(experiments using randomized activity orderings, noise threshold = 1)

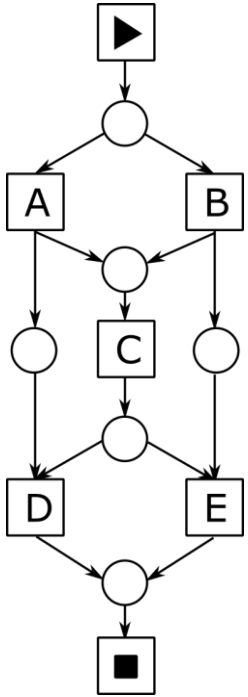
Our Approach – Conclusion

Current Approach

- High Fitness & Precision
- Complex Structures
- Improved Noise Handling
- Much faster than Brute Force

Future Work

- Improve Efficiency
- Improve Simplicity
- Explore extensions of the concept



Fitness

Precision

Simplicity

Noise Handling

Time Efficiency

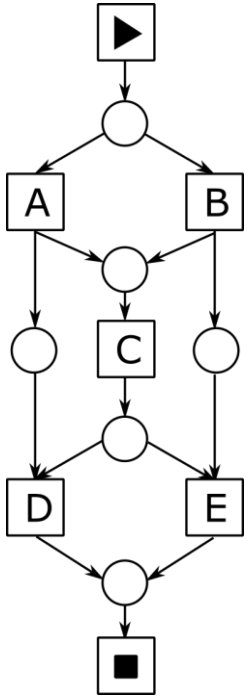
Our Approach – Conclusion

Current Approach

- High Fitness & Precision
- Complex Structures
- Improved Noise Handling
- Much faster than Brute Force

Future Work

- Improve Efficiency
- Improve Simplicity
- Explore extensions of the concept



Fitness

Precision

Simplicity

Noise Handling

Time Efficiency

Thank your for your attention!

Questions?