



Automatisierte Reduktion von reaktiver zu starker Bisimilarität

Zead Alshukairi



Agenda



EINFÜHRUNG
3

GRUNDLAGEN
23

ARBEITSGEGENSTÄNDE
50

ANMERKUNGEN
108

FAZIT
115

Einführung

- Ziel meiner Arbeit

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität
- Zusätzliche Features

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität
- Zusätzliche Features
 - Reduktion verbessert

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität
- Zusätzliche Features
 - Reduktion verbessert
 - Überprüfung eines Paar oder aller Paare

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität
- Zusätzliche Features
 - Reduktion verbessert
 - Überprüfung eines Paar oder aller Paare
 - Idee der einzigen Umgebungsaktion

Einführung

- Ziel meiner Arbeit
 - Bisimilaritäten
 - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
 - Timeouts in Anwendungen
 - Gegenseitiger Ausschluss
 - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
 - Starke Bisimilarität
 - Reaktive Bisimilarität
 - mCRL2-Projekt
 - Vordefinierte Reduktion von starker zu reaktiver Bisimilarität
- Hauptbestandteil meiner Arbeit
 - Automatisierung der Reduktion
 - Werkzeug zur Überprüfung reaktiver Bisimilarität
- Zusätzliche Features
 - Reduktion verbessert
 - Überprüfung eines Paar oder aller Paare
 - Idee der einzigen Umgebungsaktion
 - Ursprüngliche LTS's visualisiert

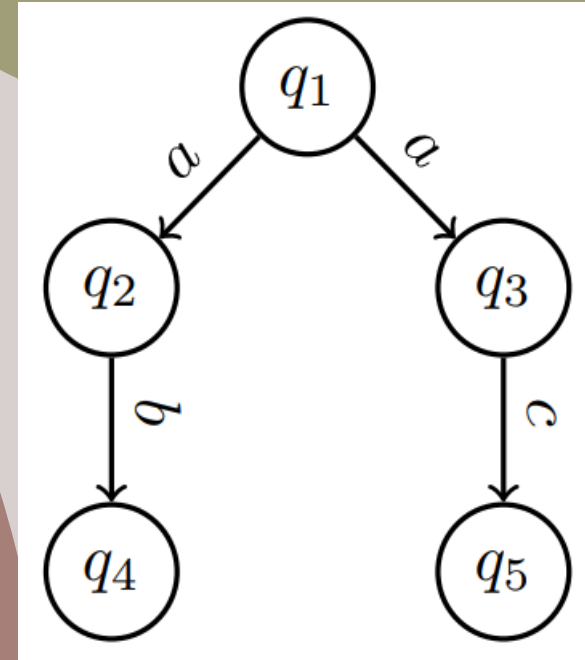
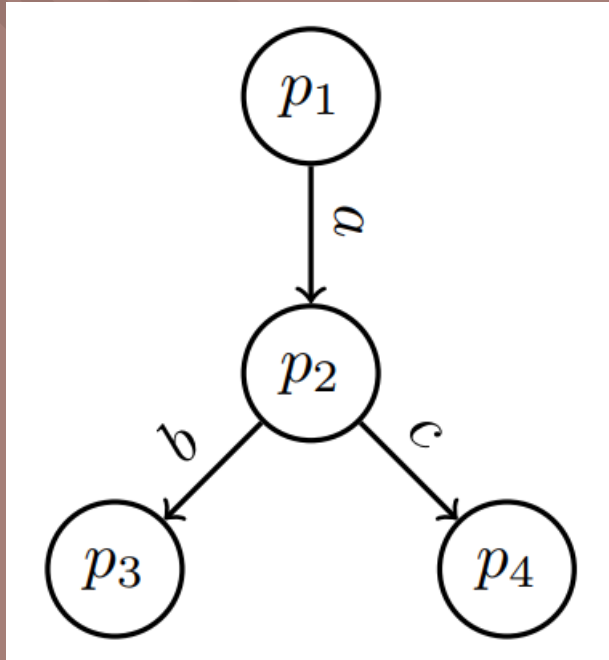
Grundlagen

- Starke Bisimilarität
- Reaktive Bisimilarität
- Die Reduktion von Max

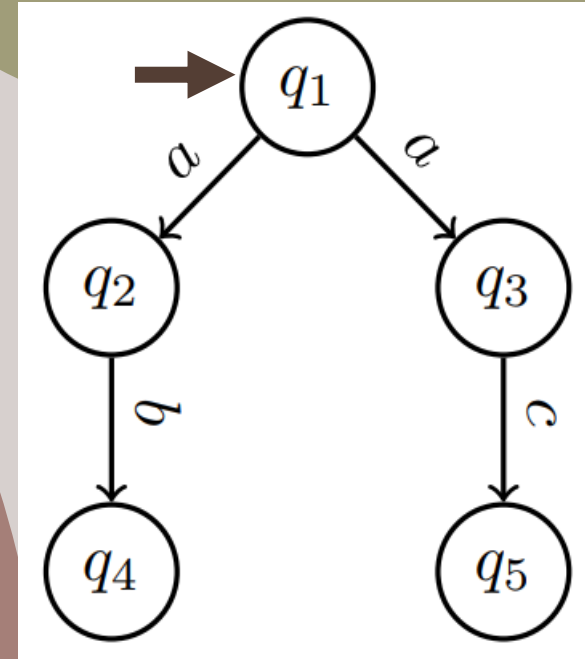
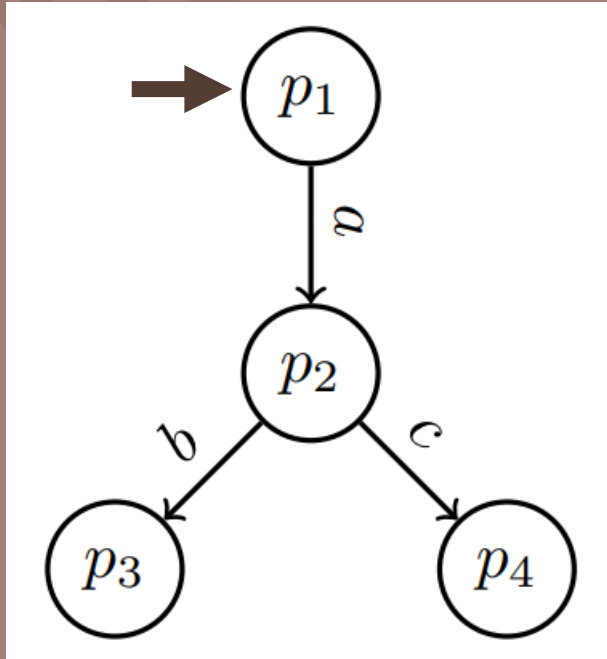
Starke Bisimilarität

The background features a light gray base with large, organic, overlapping shapes in muted olive green and dusty rose. In the top left, there are faint, stylized leaf patterns in a slightly darker shade of gray. A single, continuous white line curves across the lower right portion of the image, passing over the green and rose shapes.

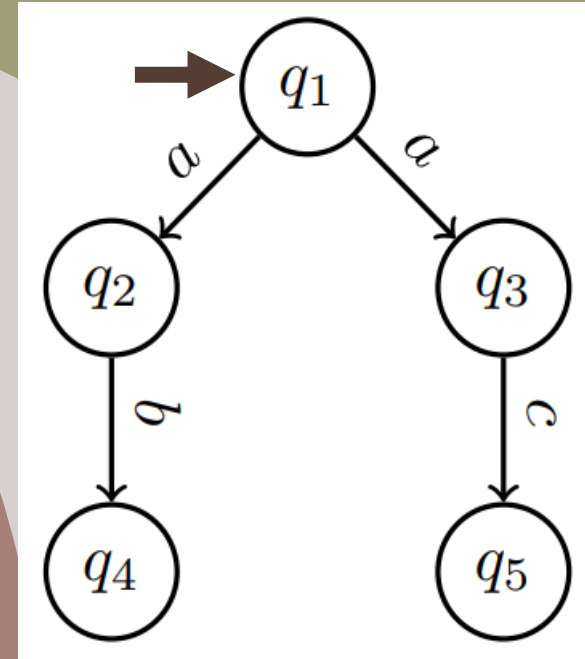
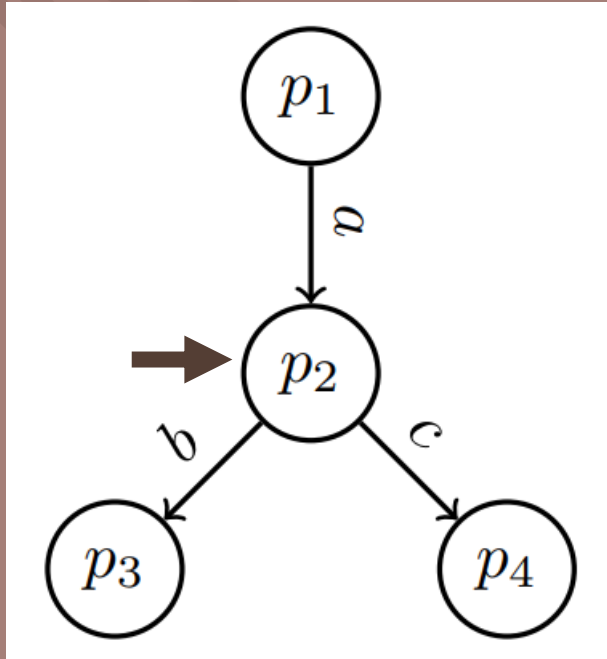
Starke Bisimilarität



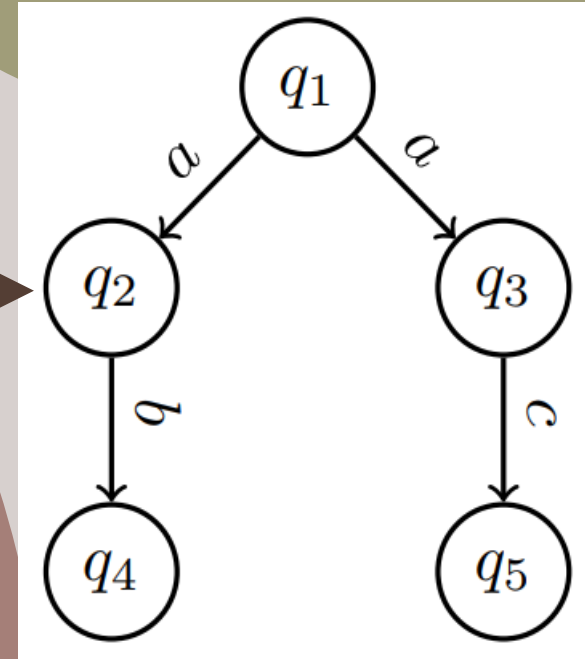
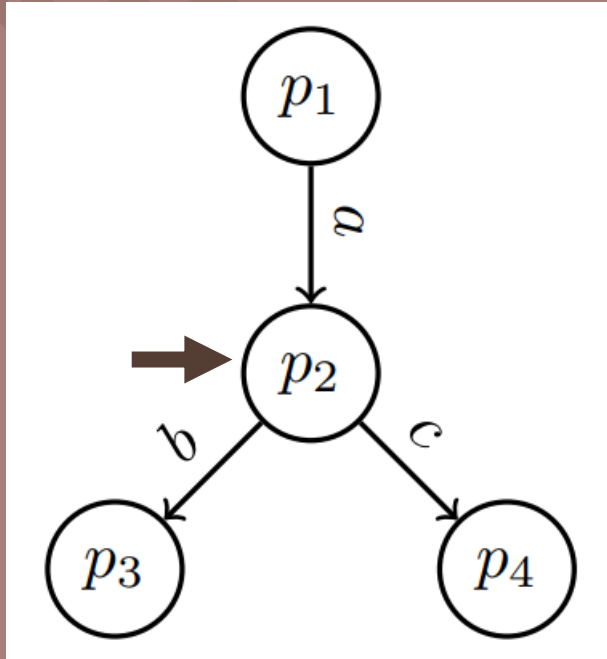
Starke Bisimilarität



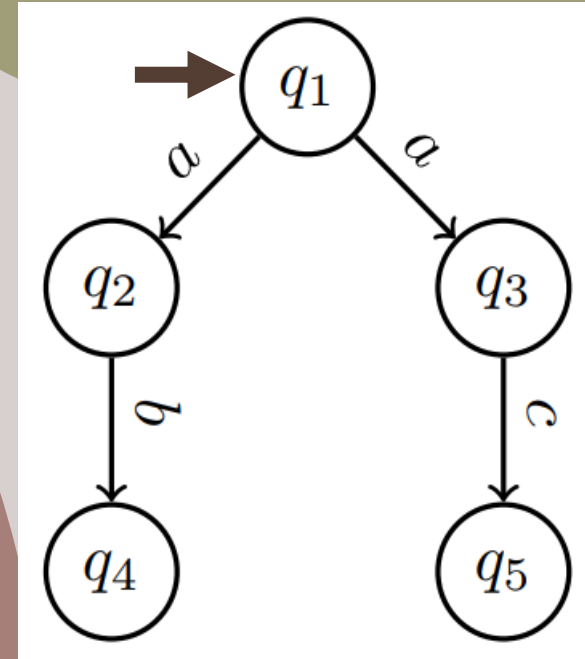
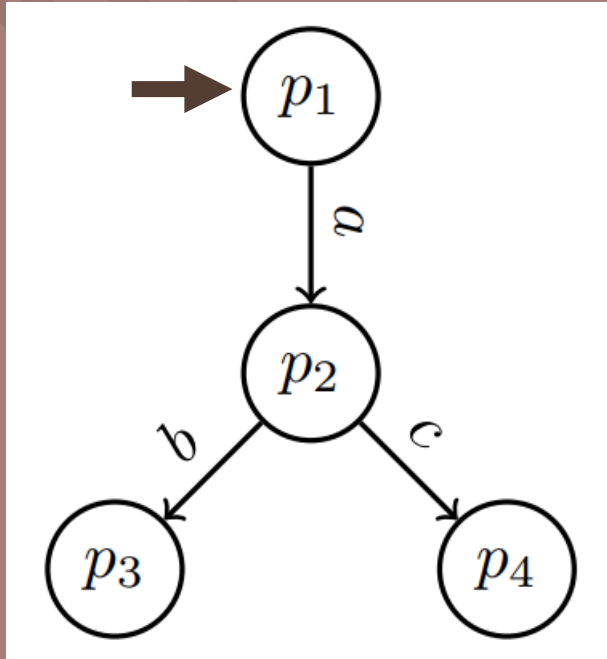
Starke Bisimilarität



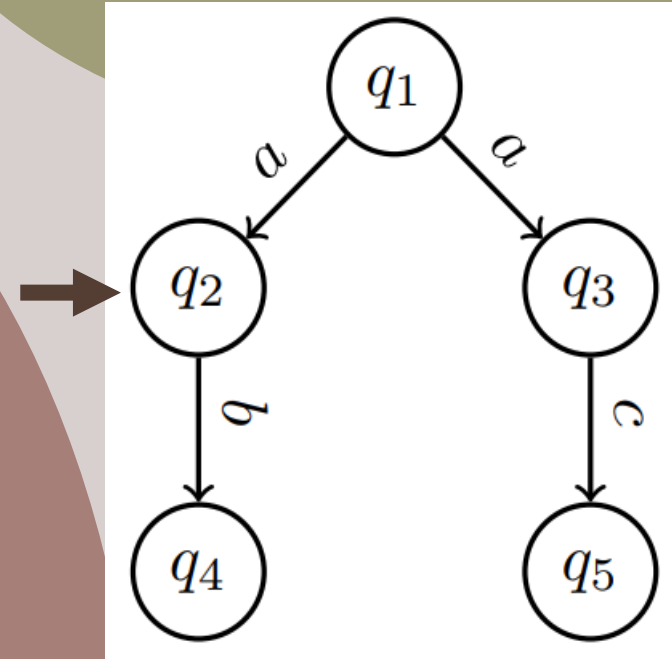
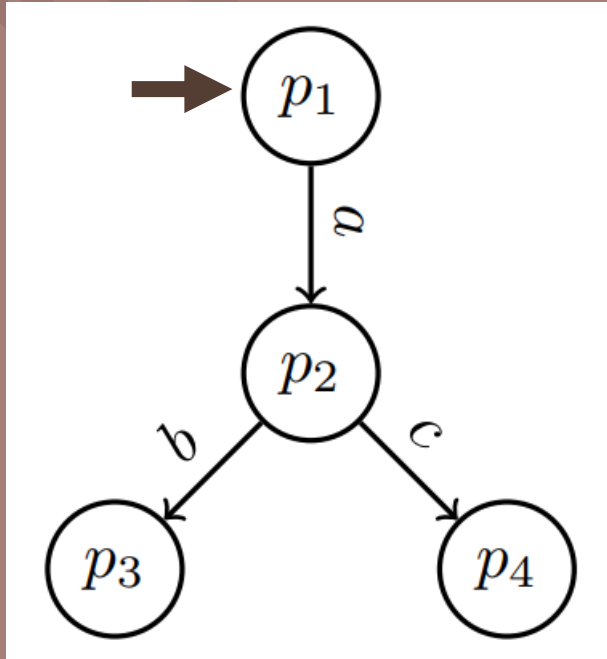
Starke Bisimilarität



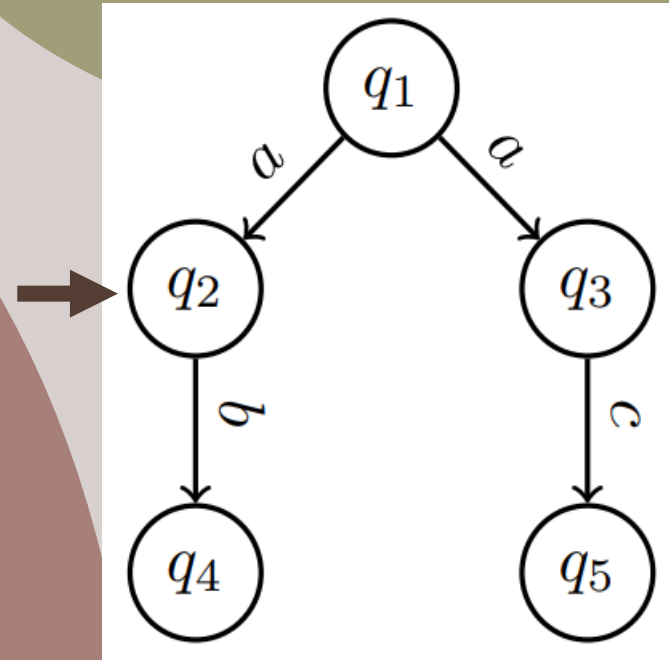
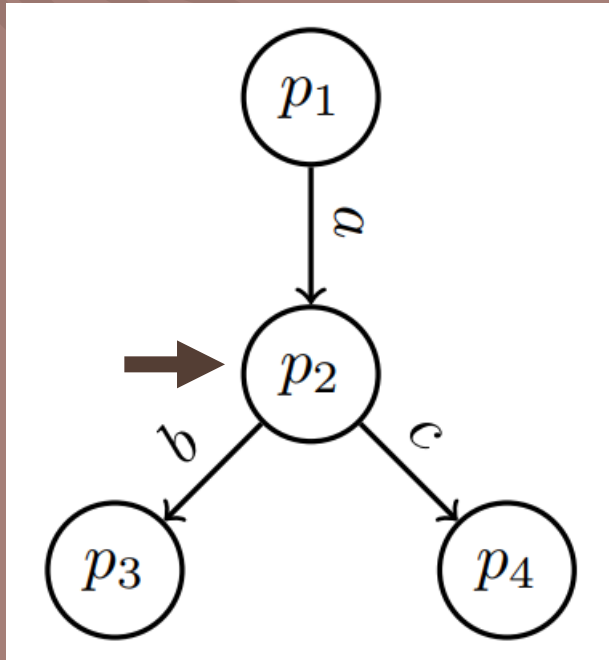
Starke Bisimilarität



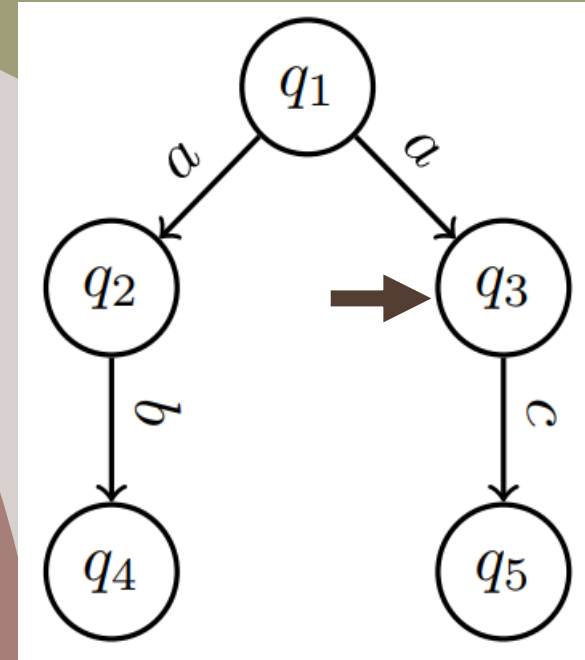
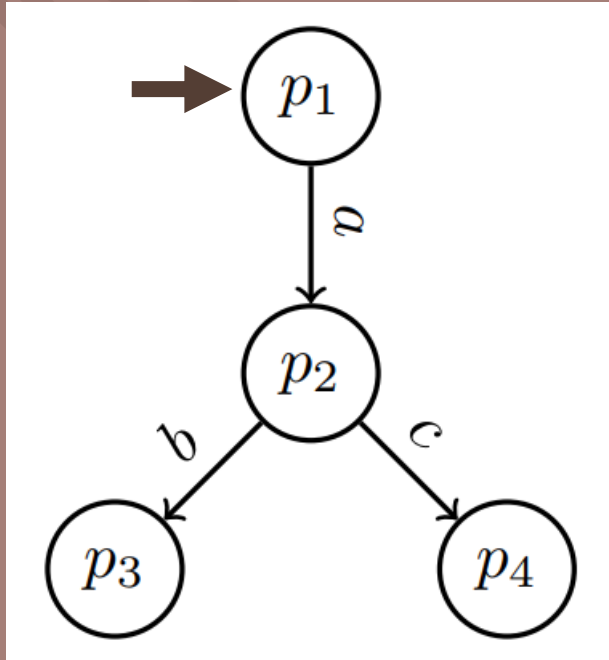
Starke Bisimilarität



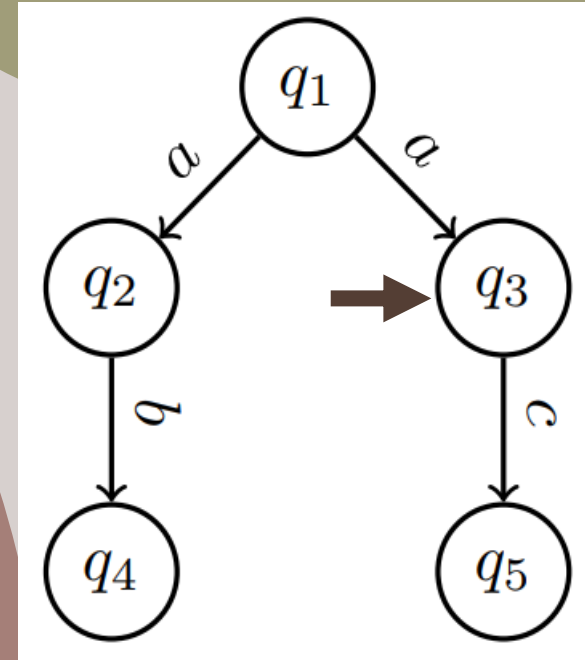
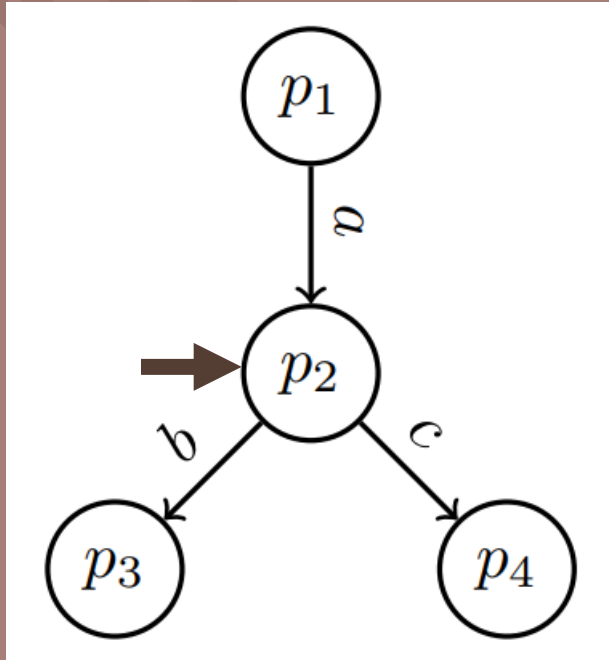
Starke Bisimilarität



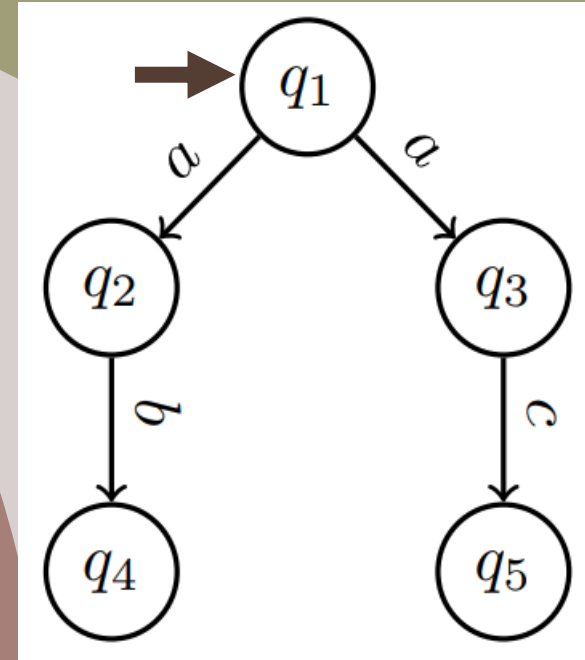
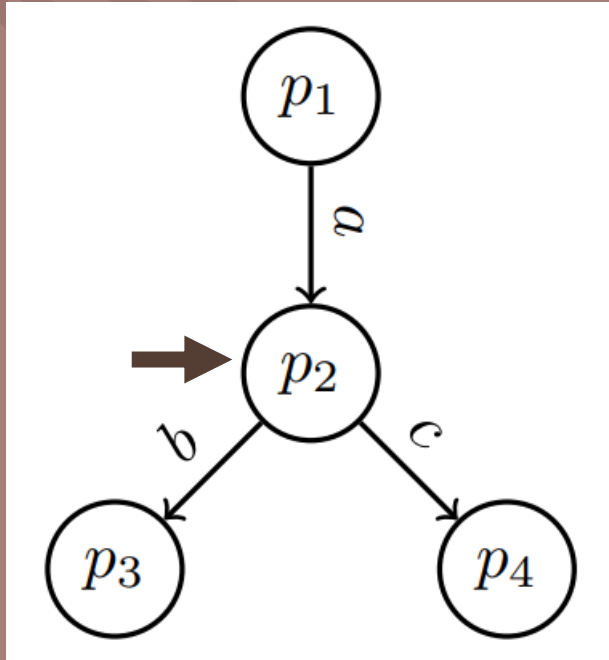
Starke Bisimilarität



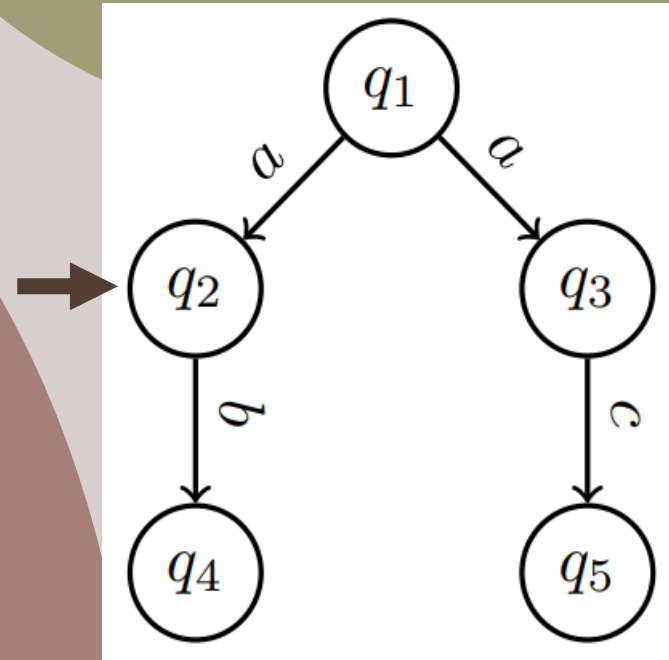
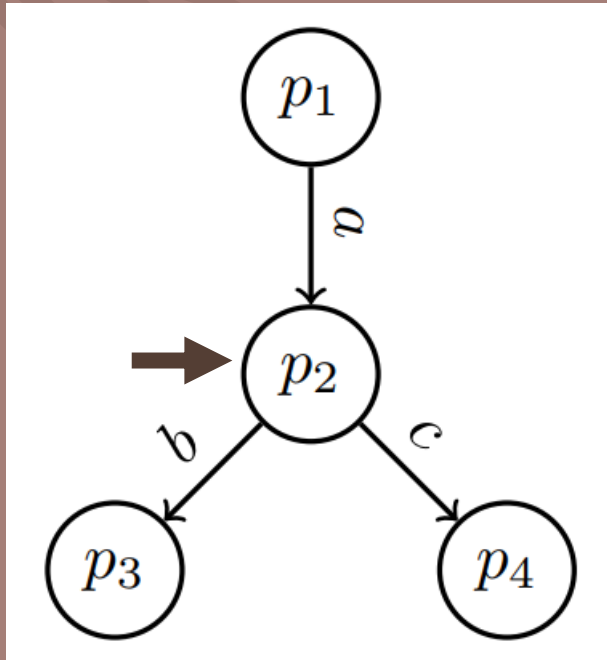
Starke Bisimilarität



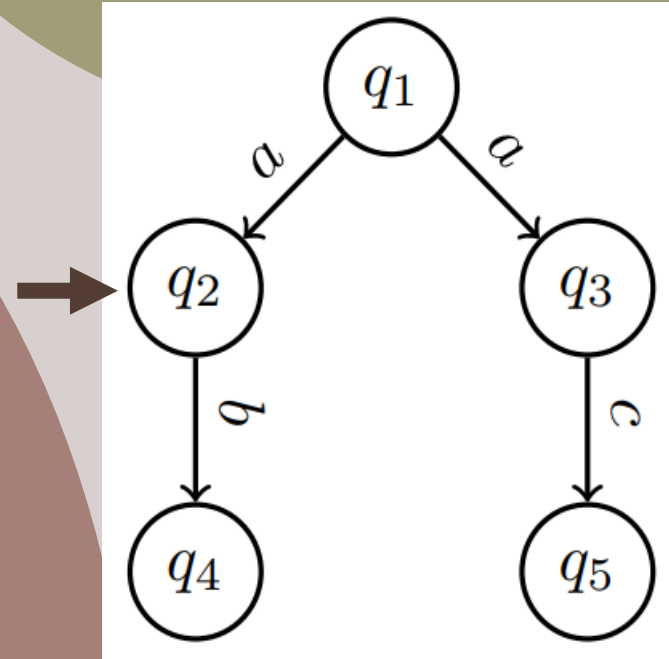
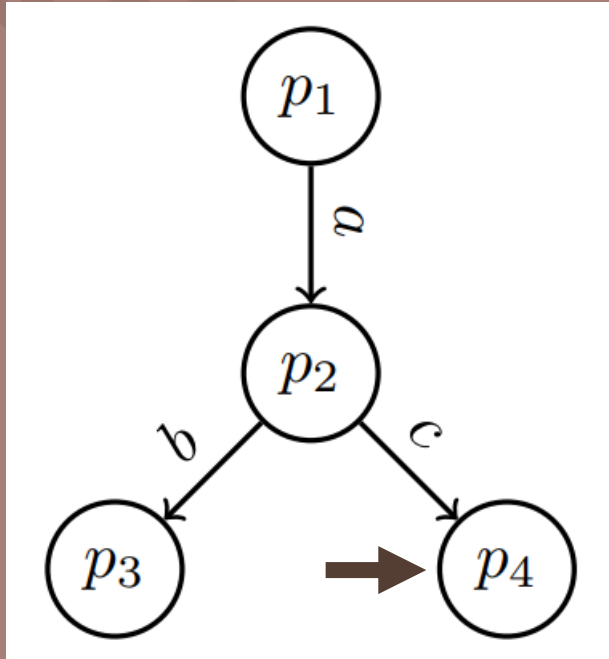
Starke Bisimilarität



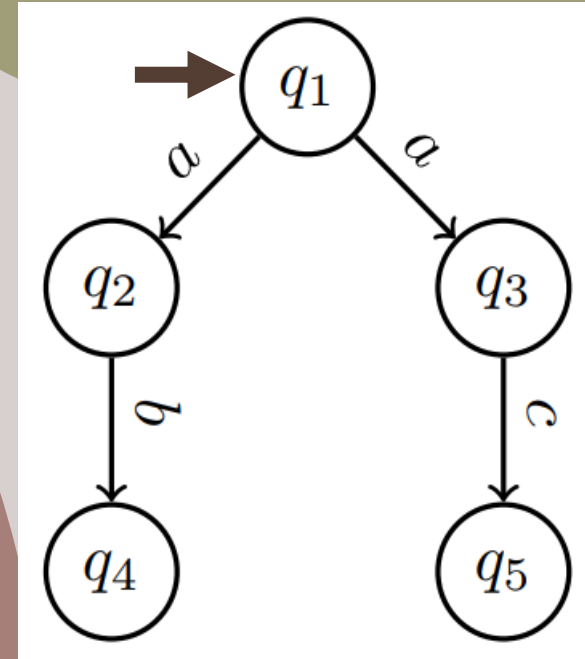
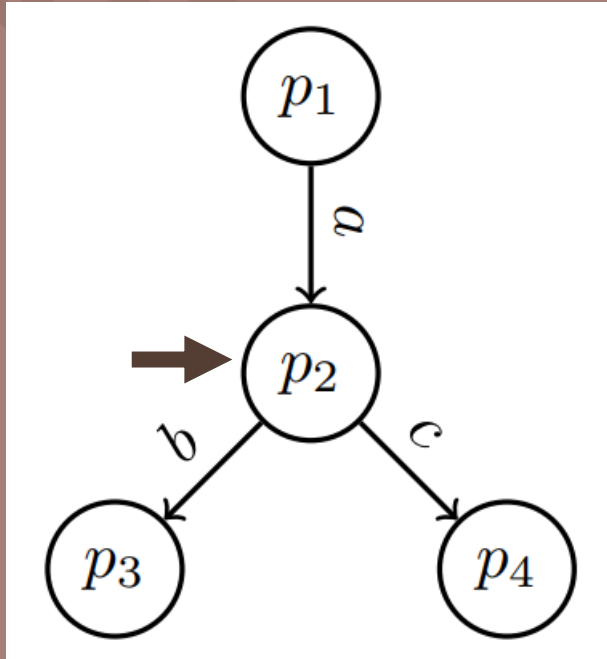
Starke Bisimilarität



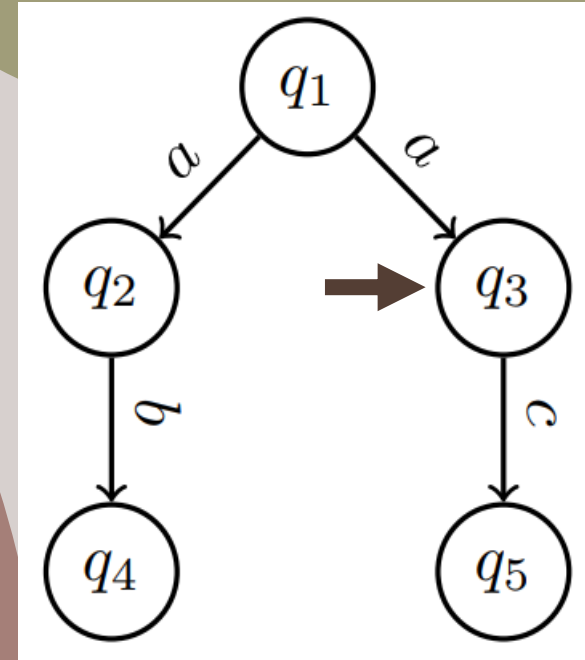
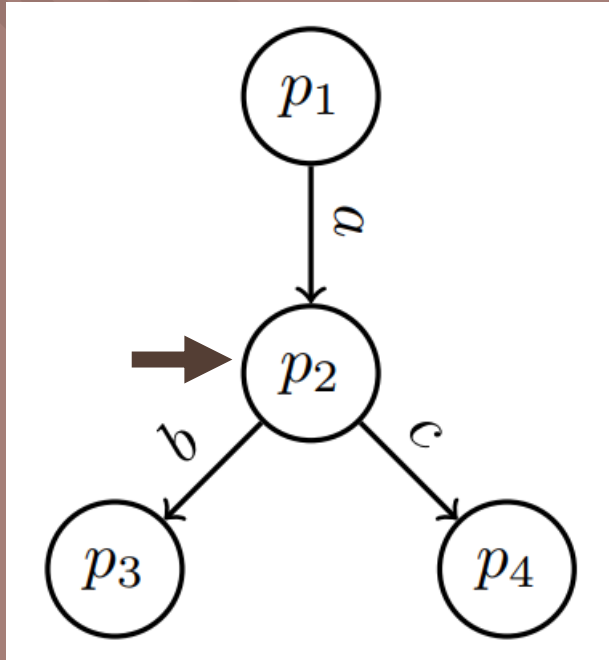
Starke Bisimilarität



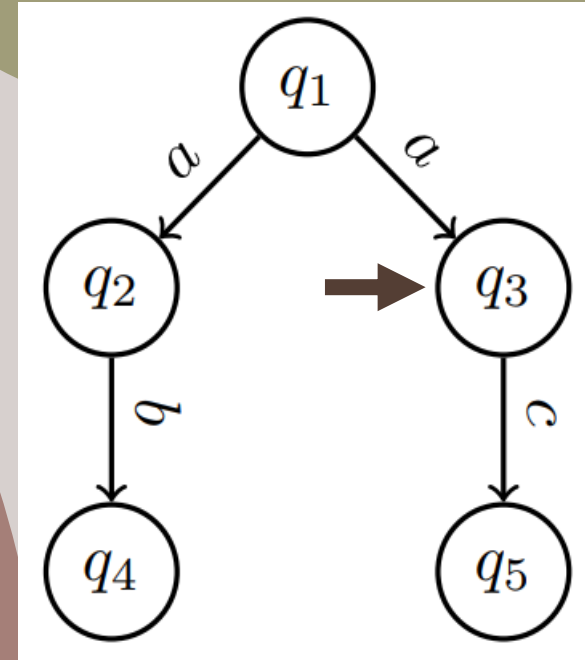
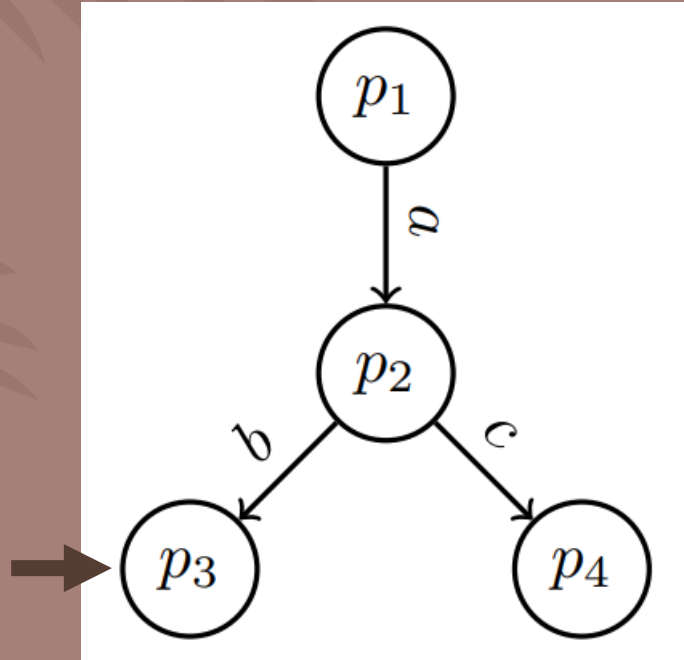
Starke Bisimilarität



Starke Bisimilarität



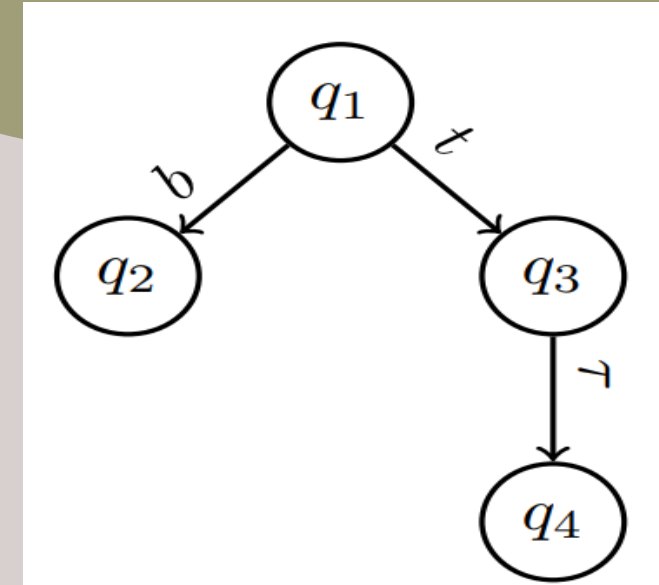
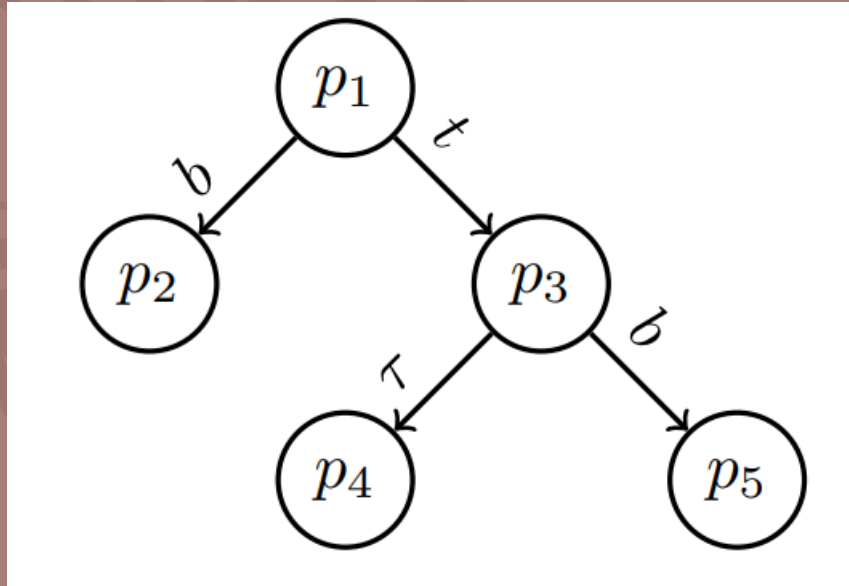
Starke Bisimilarität



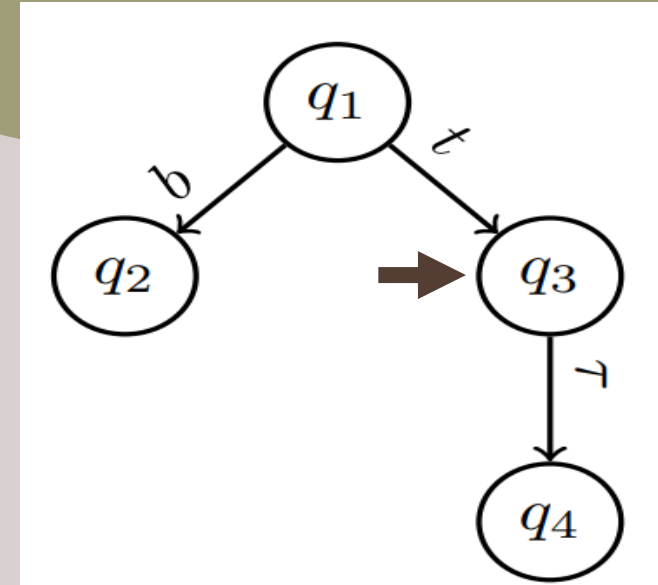
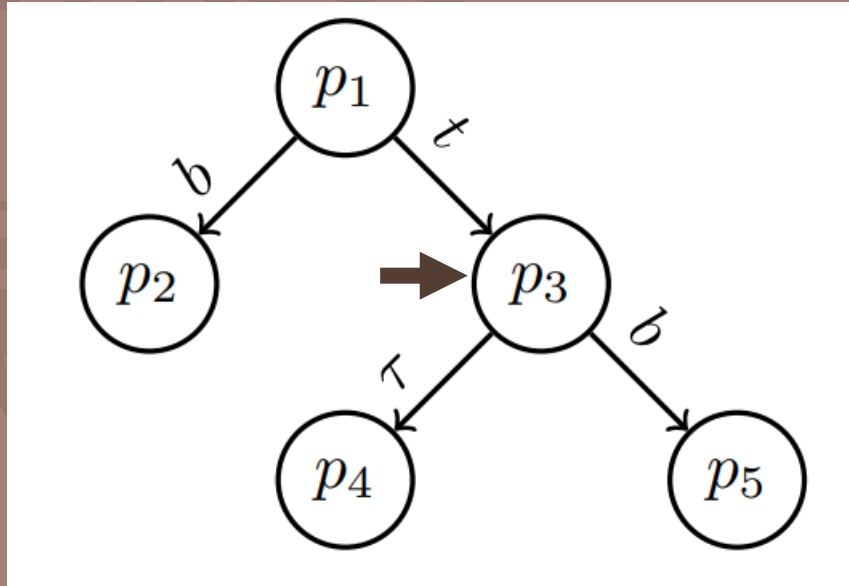
Reaktive Bisimilarität

The background features a light gray base with large, organic, overlapping shapes in muted olive green and dusty rose. A stylized, dark brown fern frond is positioned in the upper left corner. Two thin, white, flowing lines curve across the lower right portion of the image.

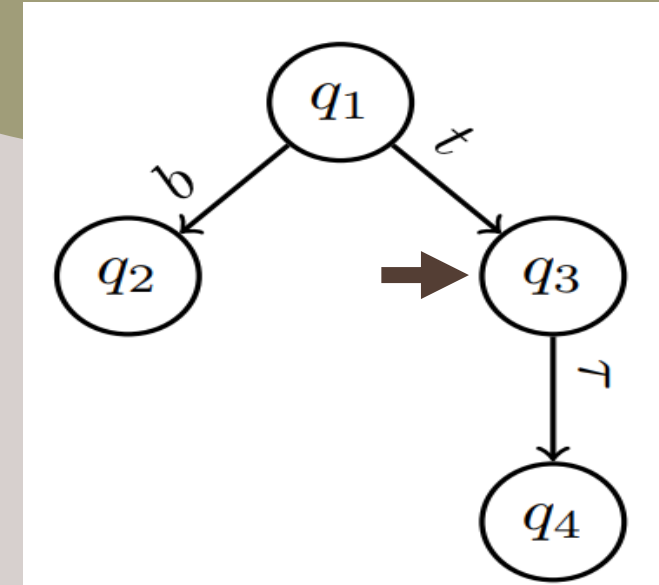
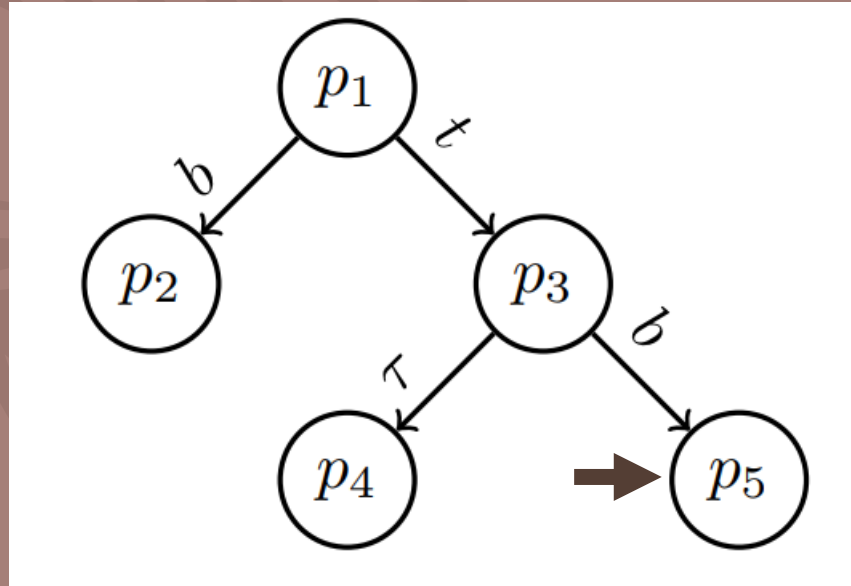
Reaktive Bisimilarität



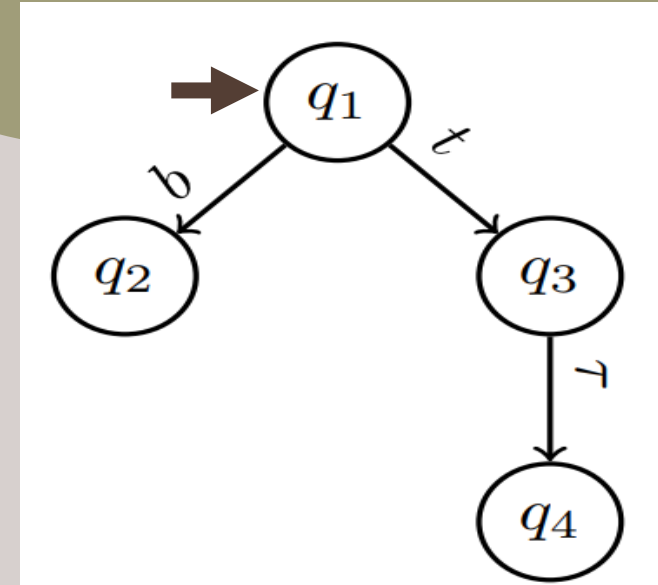
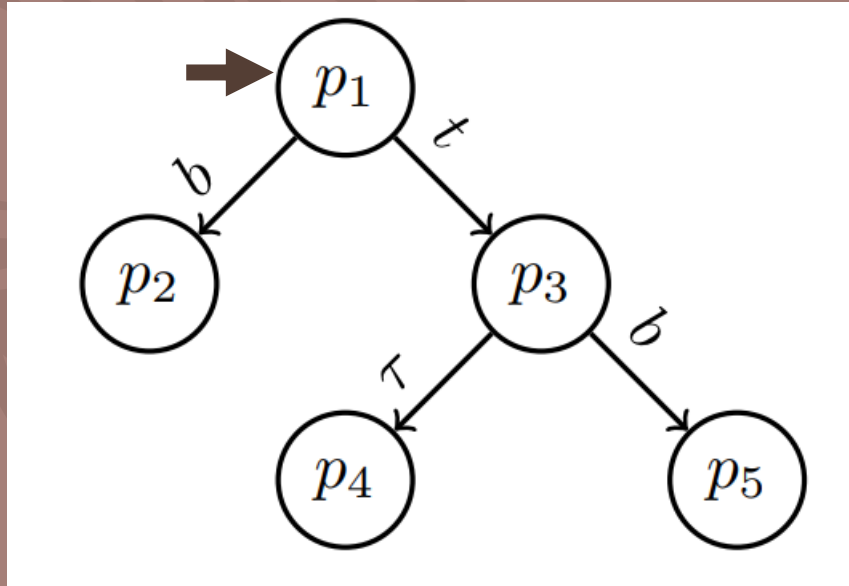
Reaktive Bisimilarität



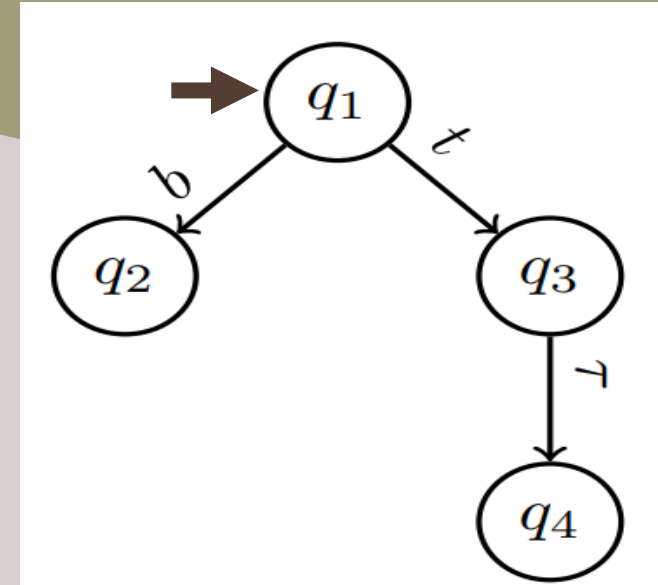
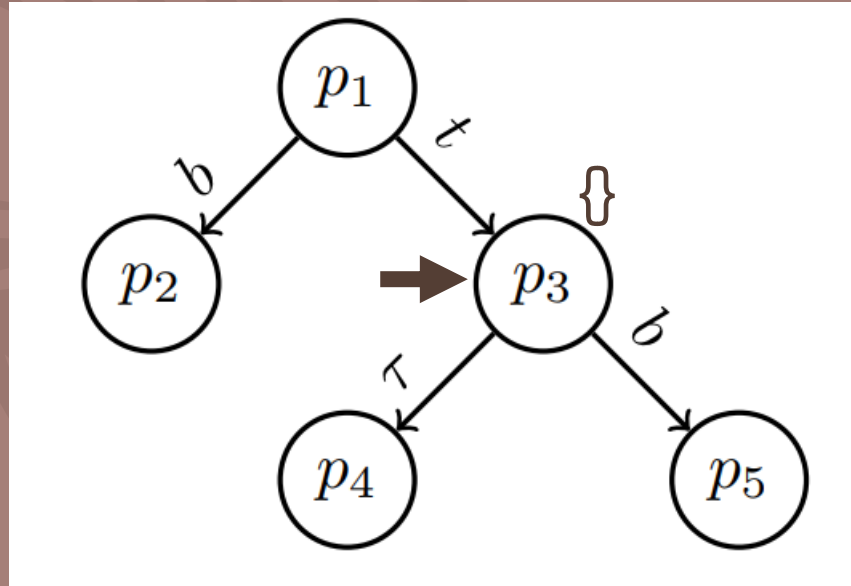
Reaktive Bisimilarität



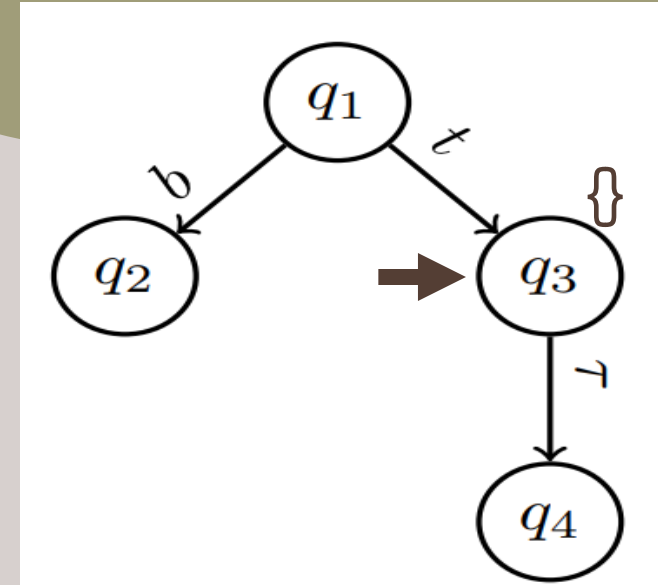
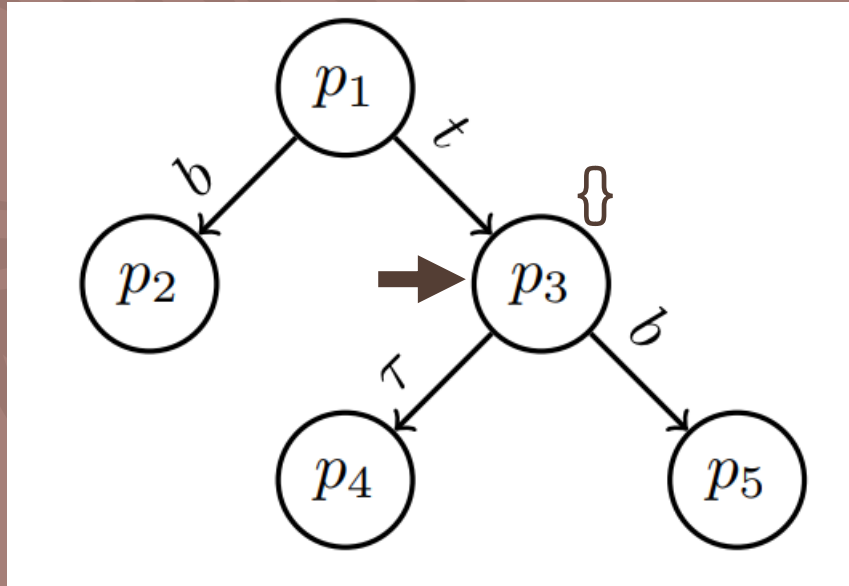
Reaktive Bisimilarität



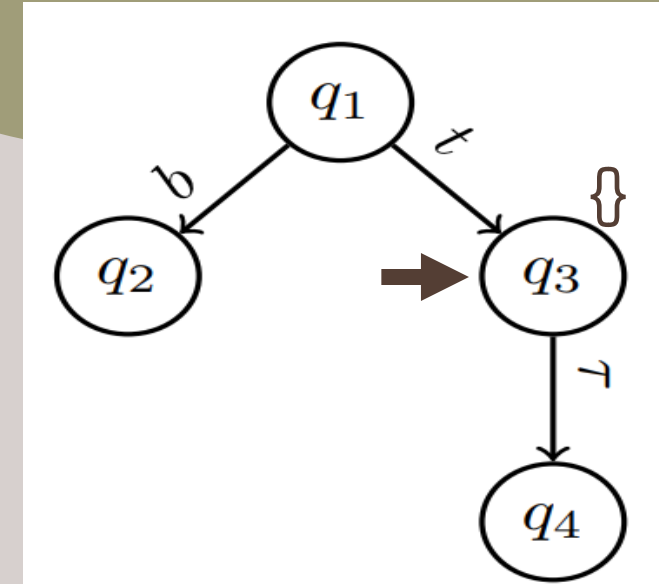
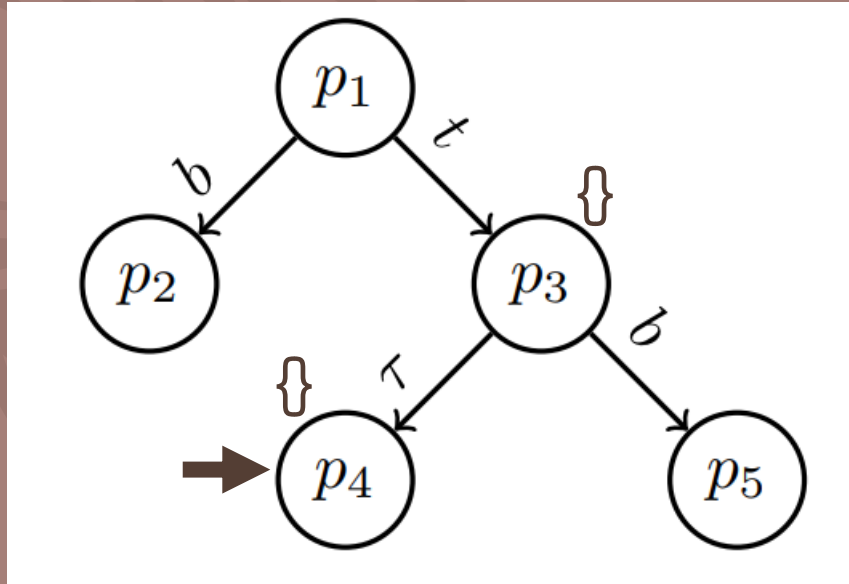
Reaktive Bisimilarität



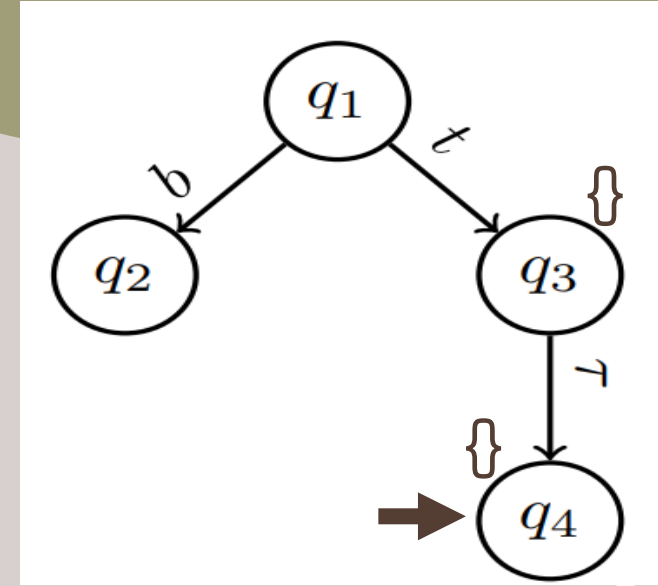
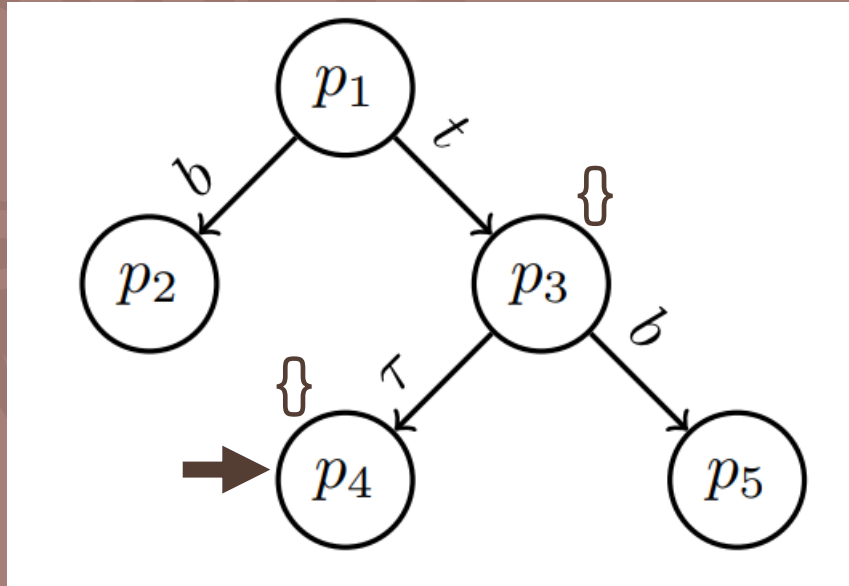
Reaktive Bisimilarität



Reaktive Bisimilarität



Reaktive Bisimilarität



Die Reduktion von Max

The background features a minimalist, abstract design. On the left, a large, solid brown shape curves upwards. To its right, a light gray shape with a wavy, organic border separates it from a solid olive-green area on the far right. In the top-left corner, a faint, stylized pine branch is visible. Two thin, white, curved lines sweep across the bottom of the image, adding a sense of movement.

Die Reduktion von Max

Reaktive Bisimilarität

Die Reduktion von Max

Reaktive Bisimilarität $\xrightarrow{\text{Reduktion}}$



Die Reduktion von Max



Die Reduktion von Max Funktionsweise

The background features a light gray base with large, organic, overlapping shapes in muted olive green and dusty rose. In the upper left, there is a faint, stylized silhouette of a palm tree or similar plant.

Die Reduktion von Max Funktionsweise

$$1) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_v \mathcal{V}(p')}$$

$$2) \frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{\mathcal{V}_X(p) \xrightarrow{a}_v \mathcal{V}(p')} a \in X$$

$$4) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}_X(p) \xrightarrow{\tau}_v \mathcal{V}_X(p')}$$

$$5) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\mathcal{V}_X(p) \xrightarrow{t_\varepsilon}_v \mathcal{V}(p)}$$

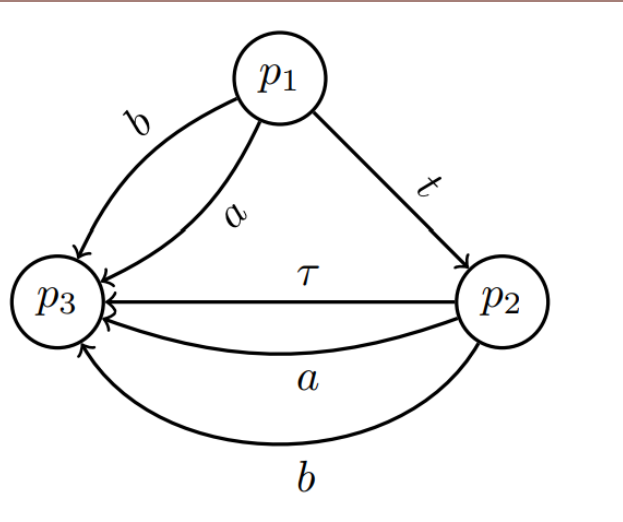
$$6) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\mathcal{V}_X(p) \xrightarrow{t}_v \mathcal{V}_X(p')}$$

Die Reduktion von Max Beispiel

The background features a light gray base with large, organic, overlapping shapes in muted olive green and dusty rose. Faint, stylized foliage patterns are visible in the upper left corner.

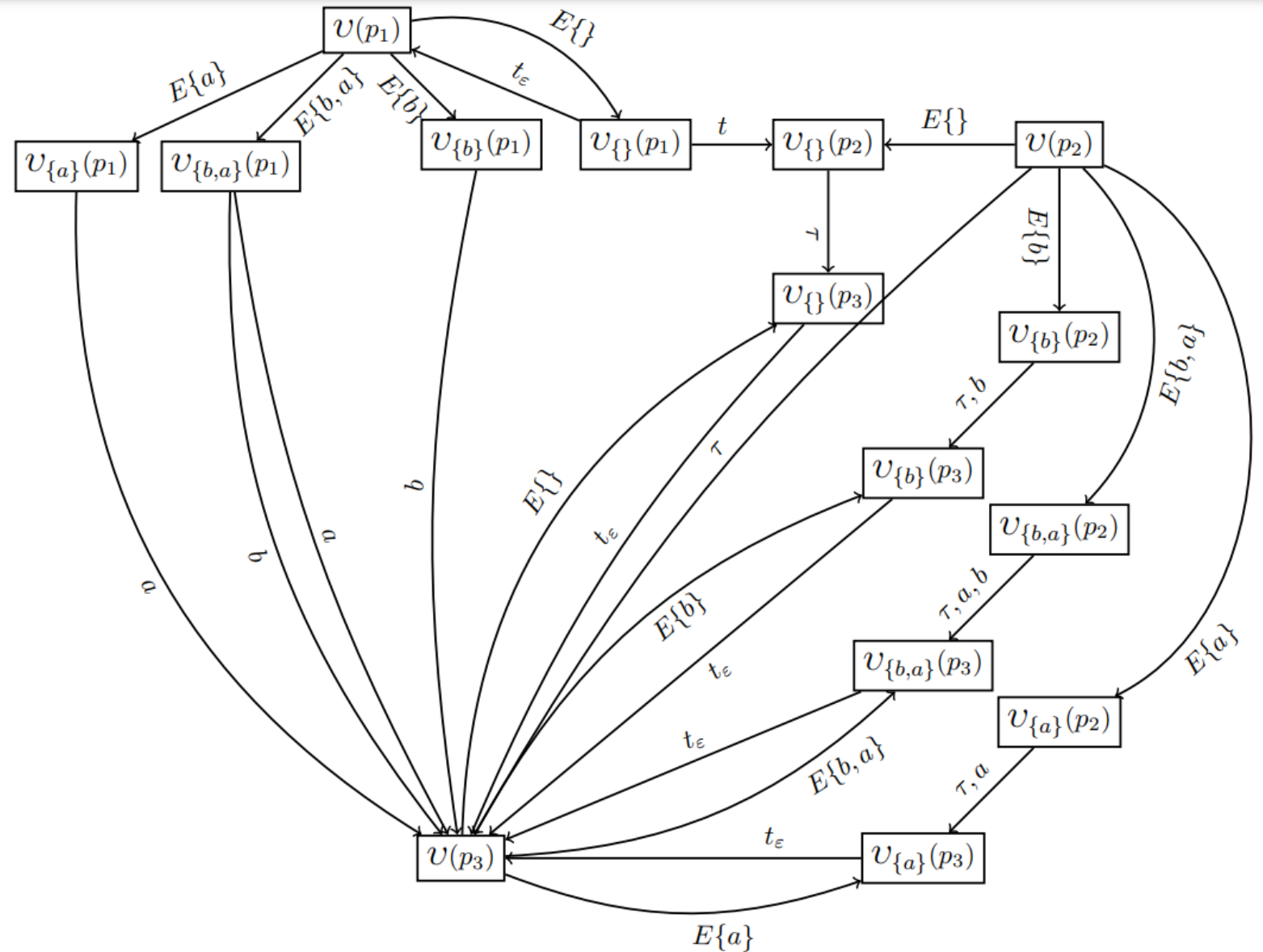
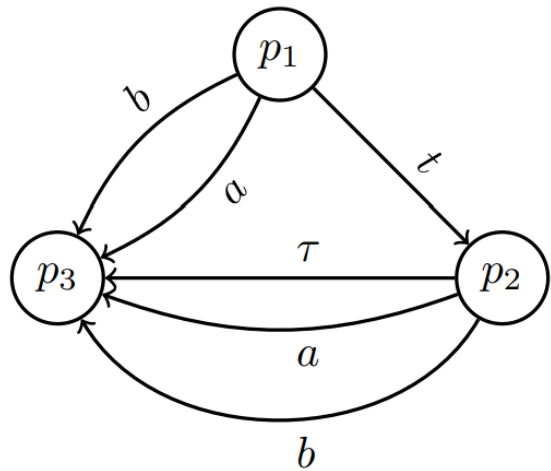
Die Reduktion von Max

Beispiel



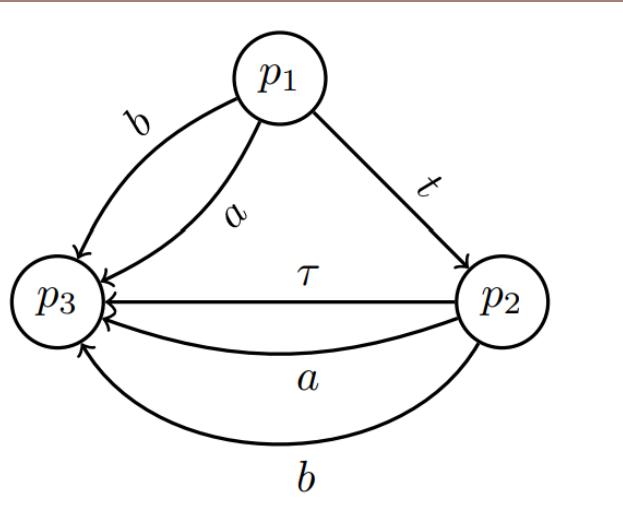
Die Reduktion von Max

Beispiel

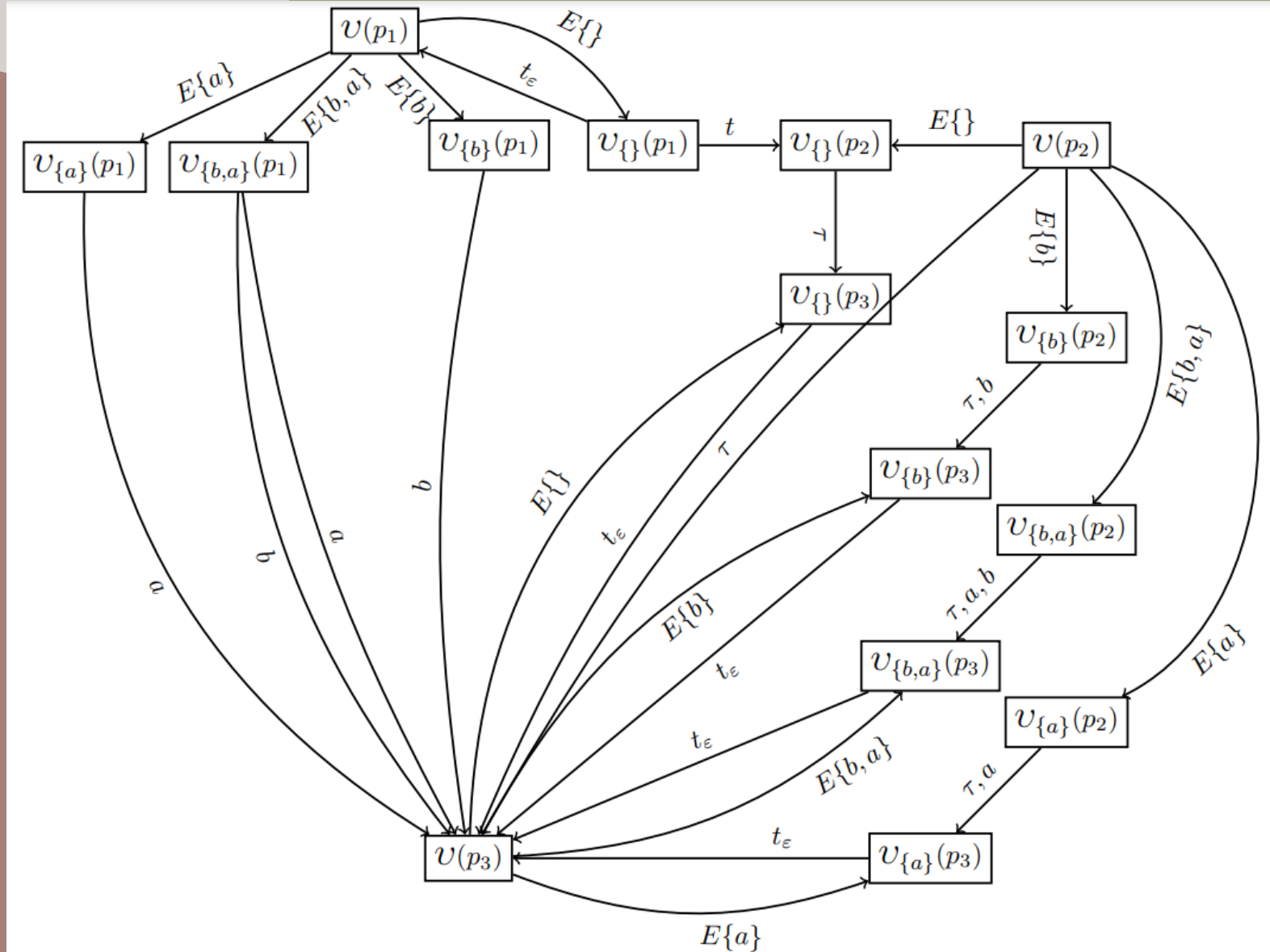


Die Reduktion von Max

Beispiel



WIE?





ARBEITSGEGENSTÄNDE

Kernidee

Ursprünglicher Algorithmus

Verbesserte Algorithmus

Einzig Umgebungsaktion

Überprüfung eines oder aller Paare

Kernidee

Kernidee

mCRL2
Spezifikationen

Kernidee

mCRL2
Spezifikationen

```
act
  a, b, f, g, c, time;

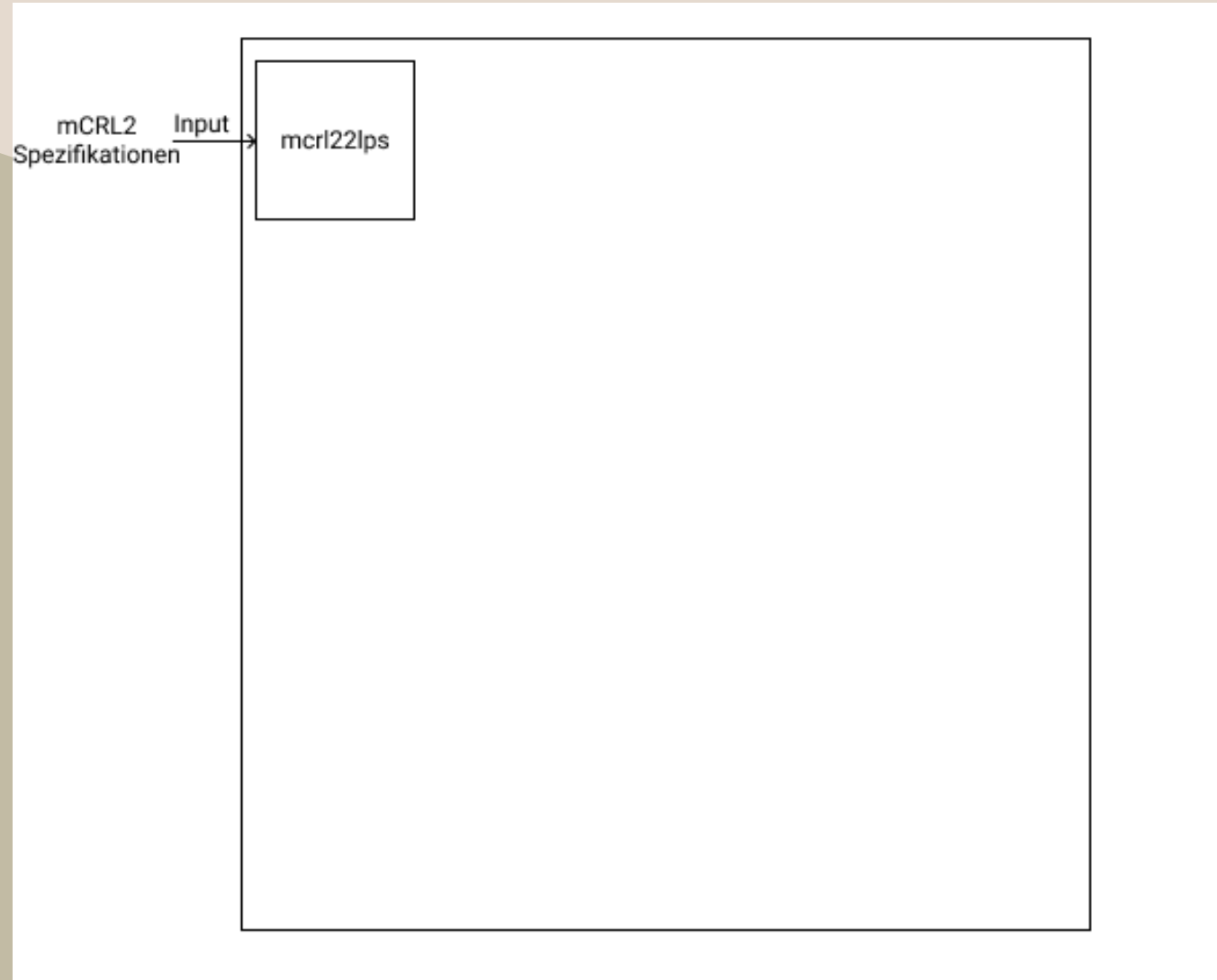
proc
  _0 = a._1 + time._2;
  _1 = b._3;
  _2 = c._4 + g._6;
  _3 = f._5 + time._2 + c._4;
  _4 = time._0;
  _5 = delta;
  _6 = delta;
init
  hide(
    {c}, _0
  );
```

Kernidee

mCRL2
Spezifikationen



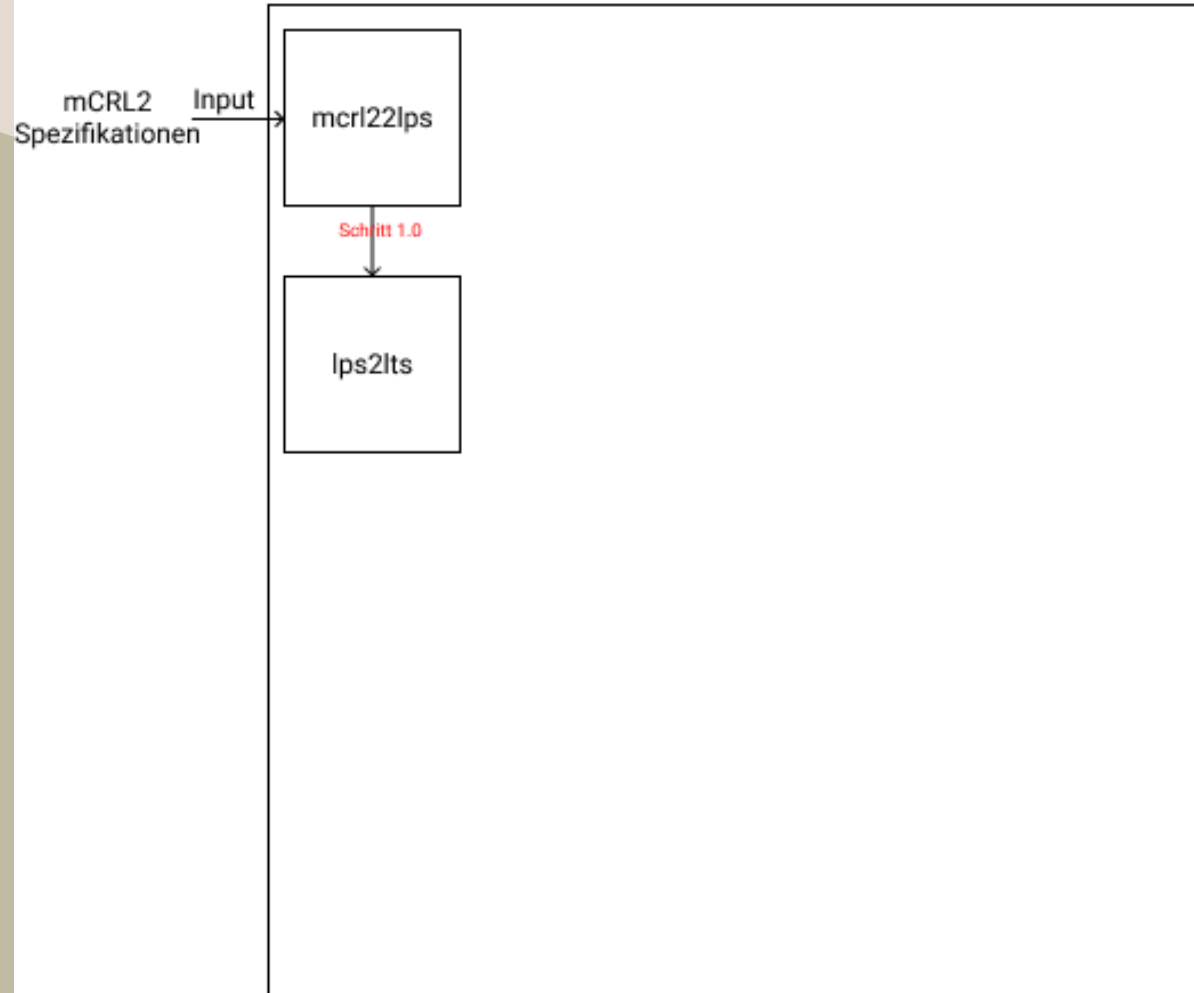
Kernidee



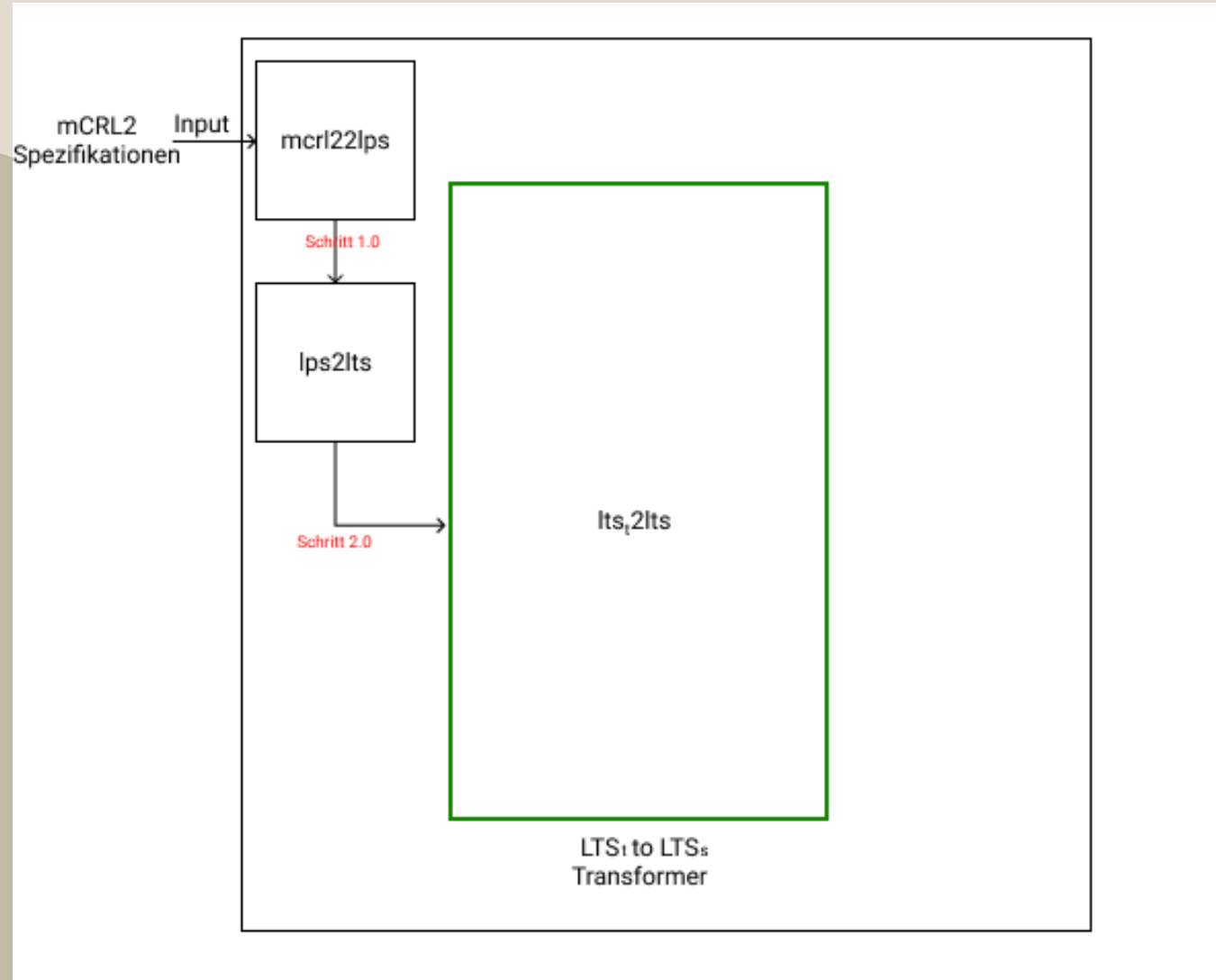
LPS ist in Binärformat

Kernidee

```
des (0,4,3)
(0,"time",1)
(0,"a",2)
(1,"tau",2)
(1,"a",2)
```

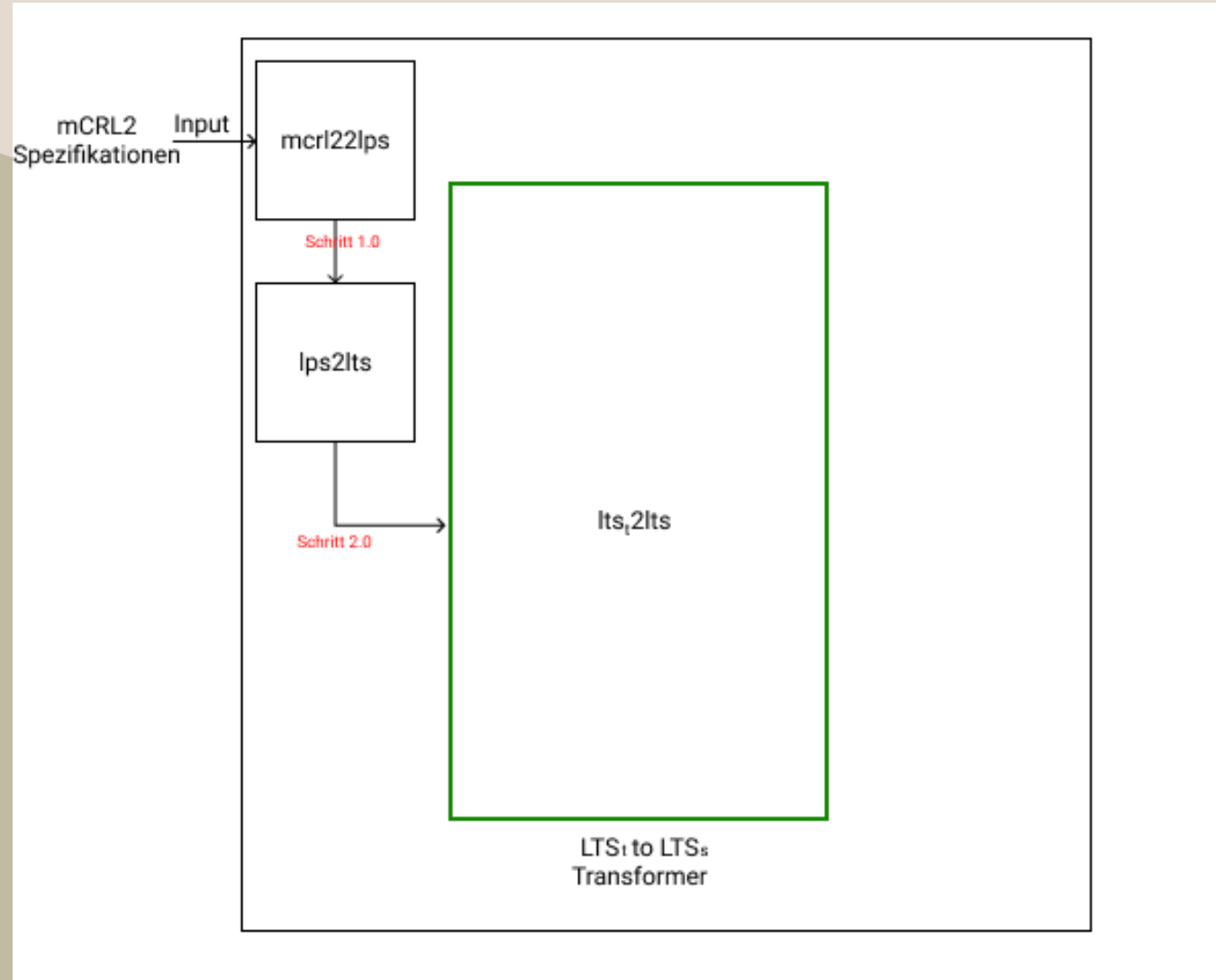


Kernidee

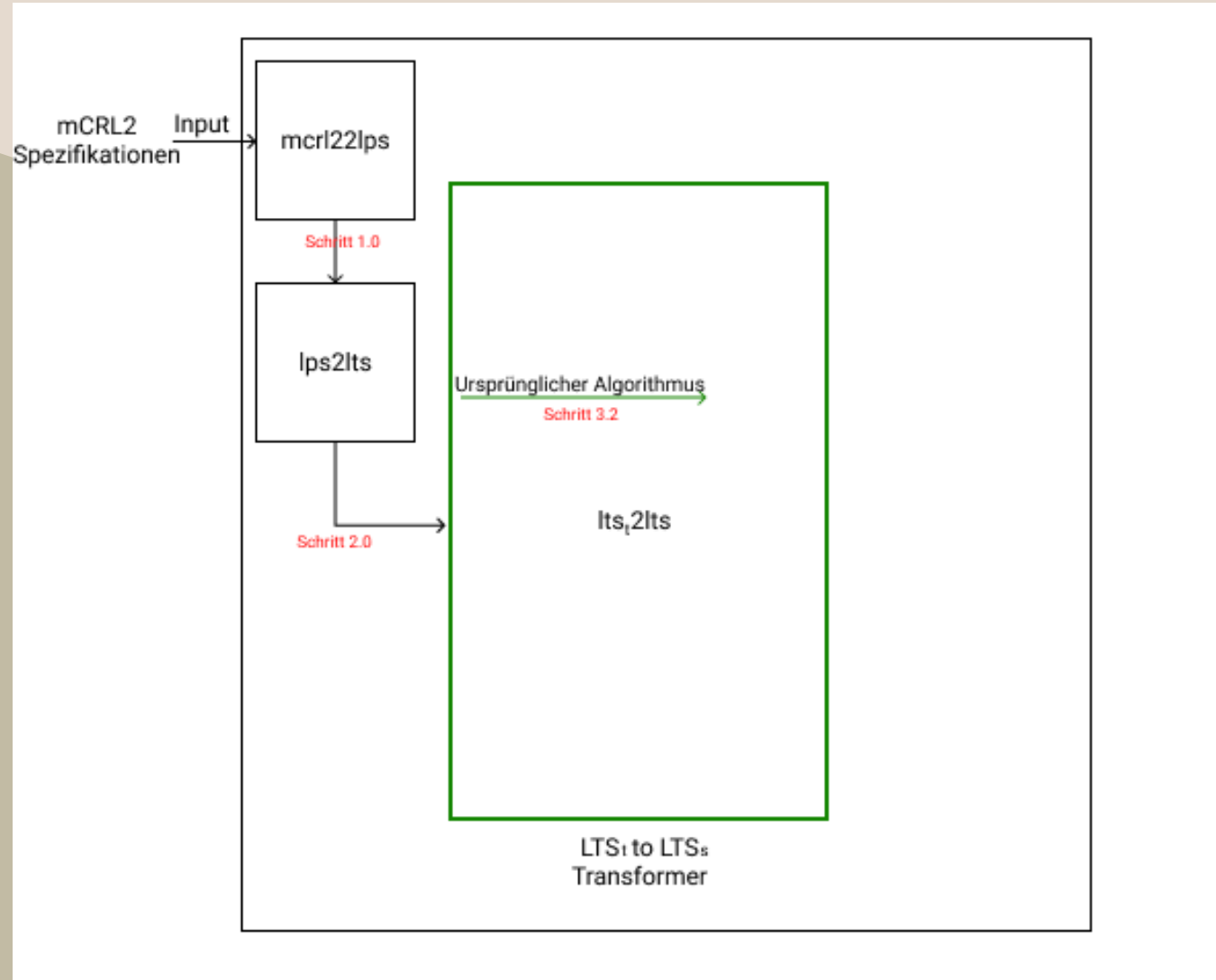


Kernidee

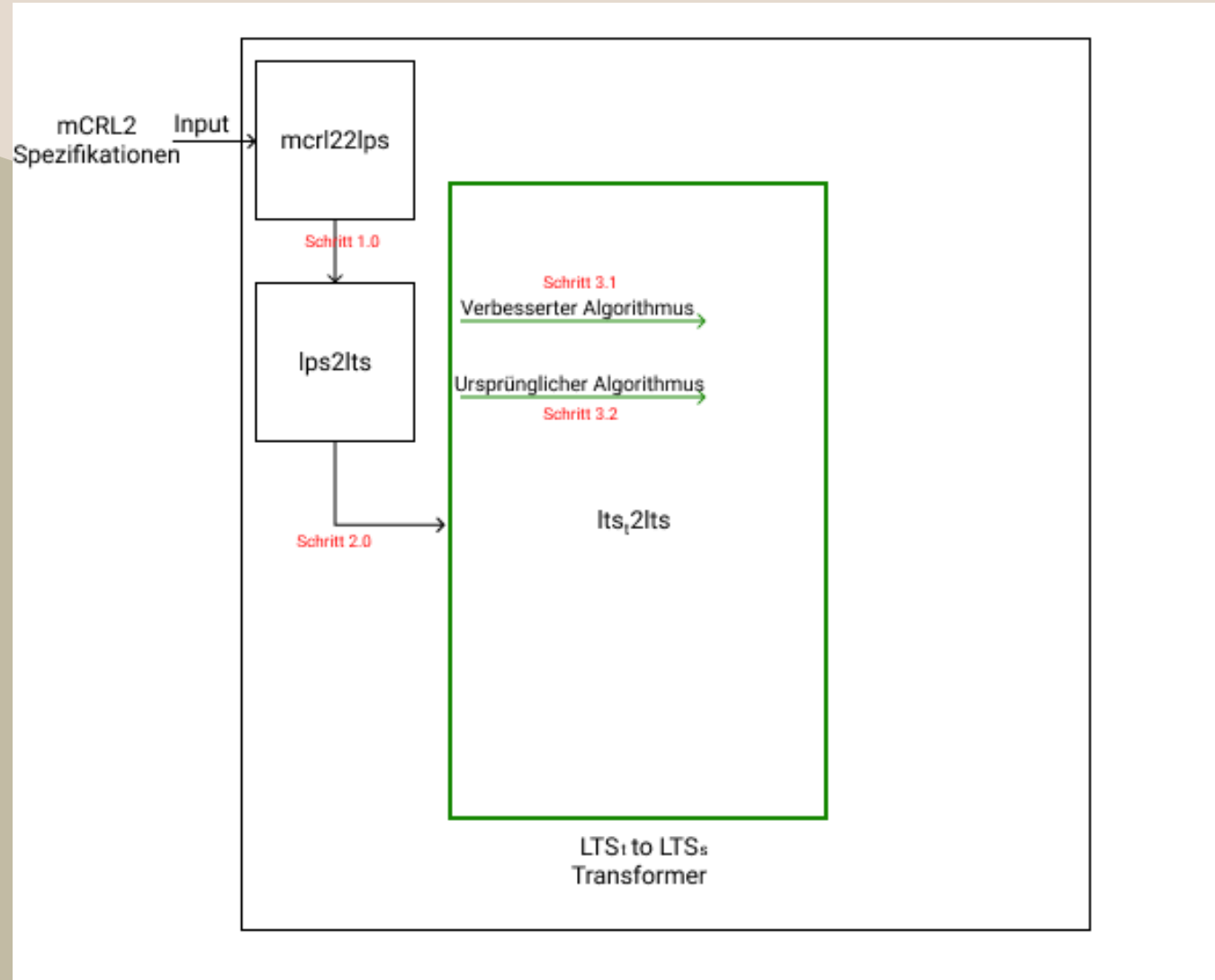
```
39 (15,"time",24)
40 (16,"t_e",0)
41 (16,"time",25)
42 (17,"noeE",26)
43 (17,"a",27)
44 (17,"a_b",28)
45 (17,"a_b_g",29)
46 (17,"a_b_g_f",30)
```



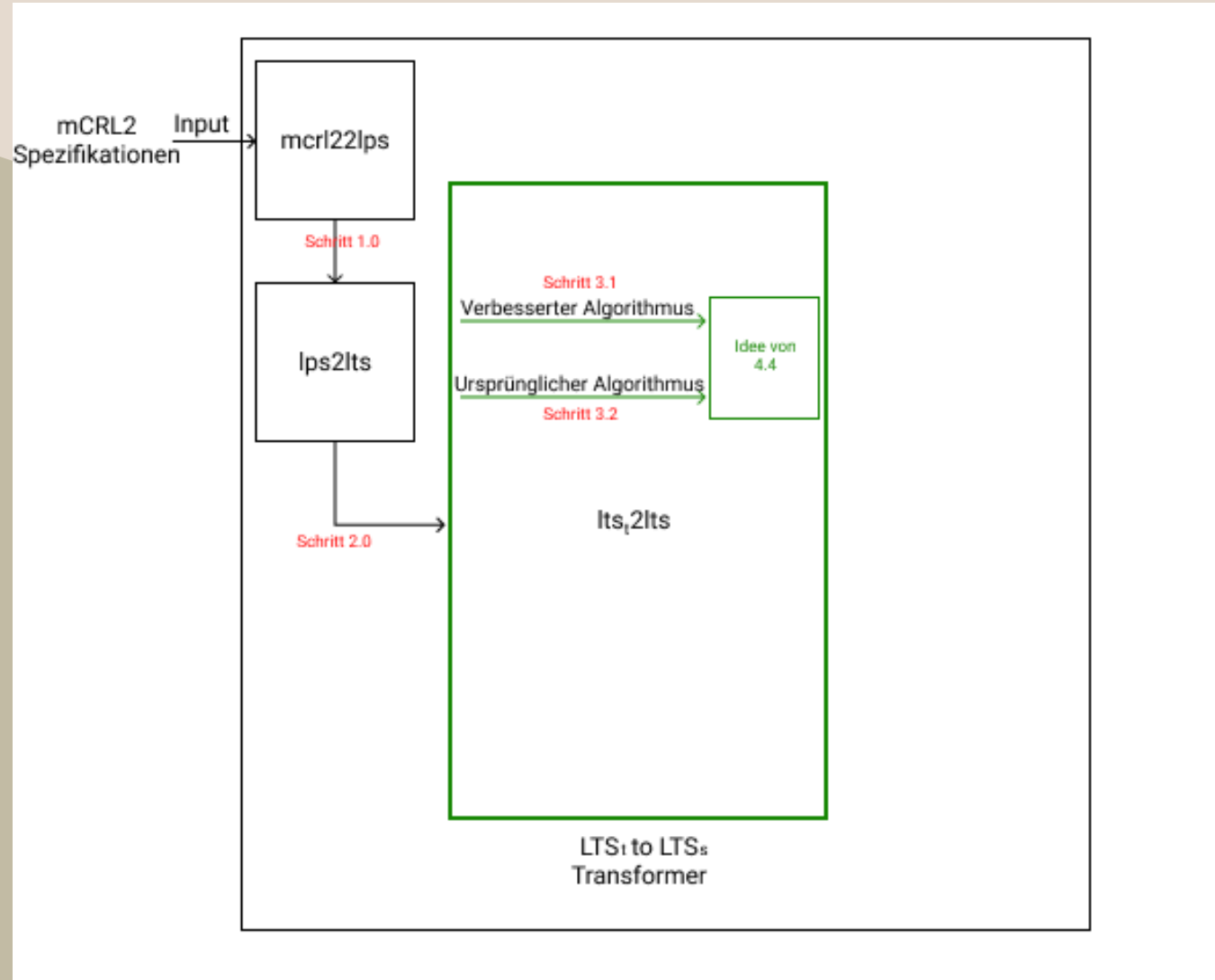
Kernidee



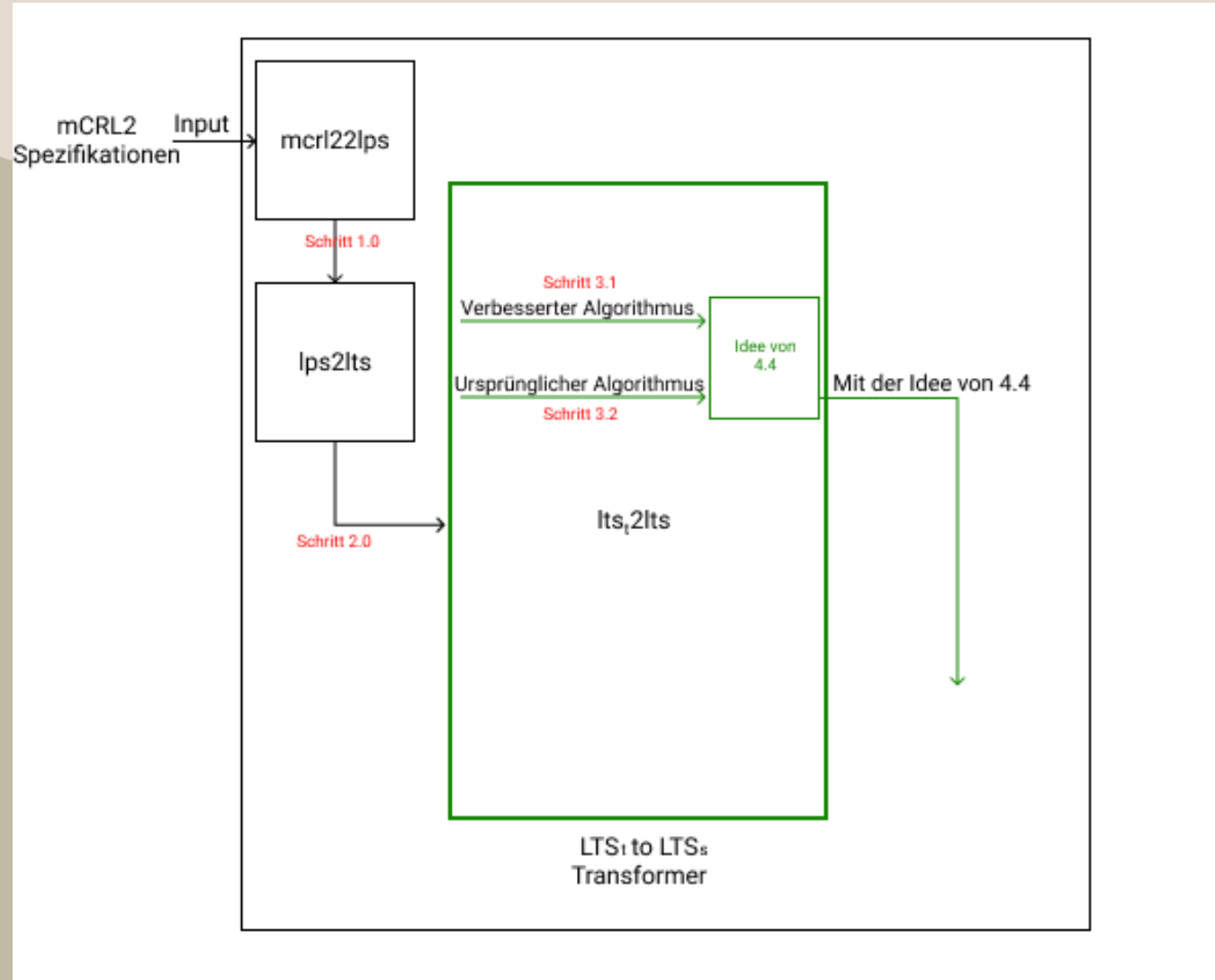
Kernidee



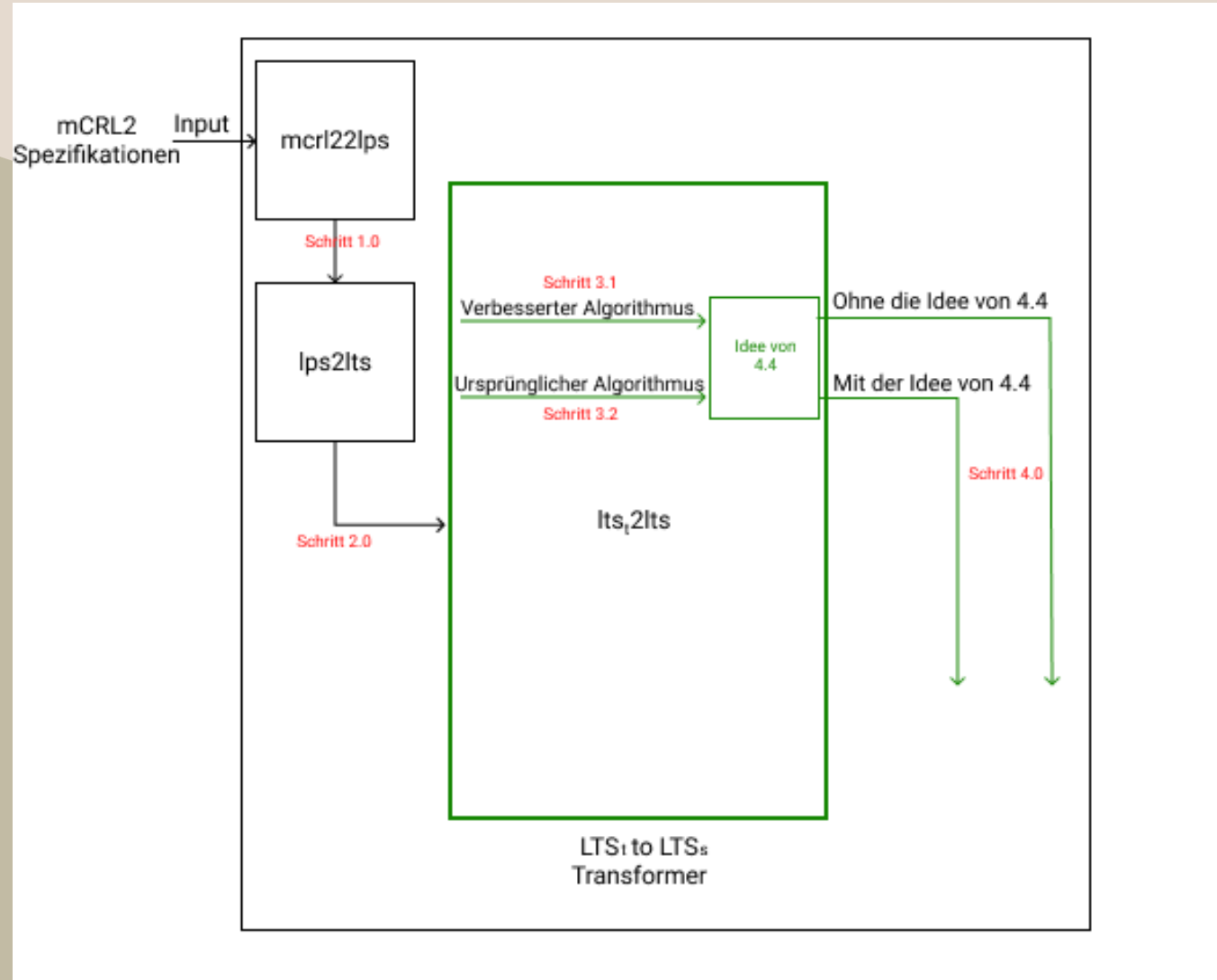
Kernidee



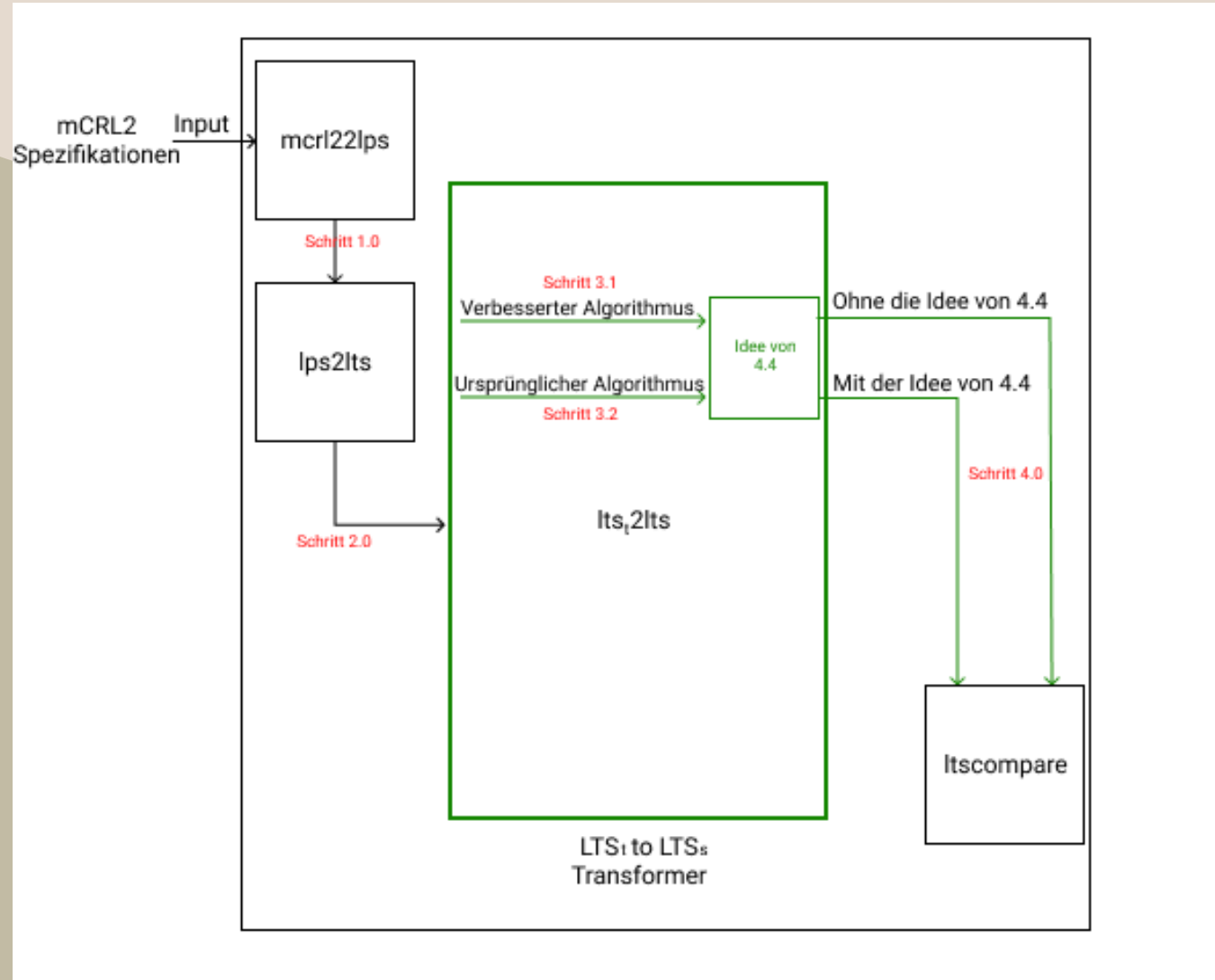
Kernidee



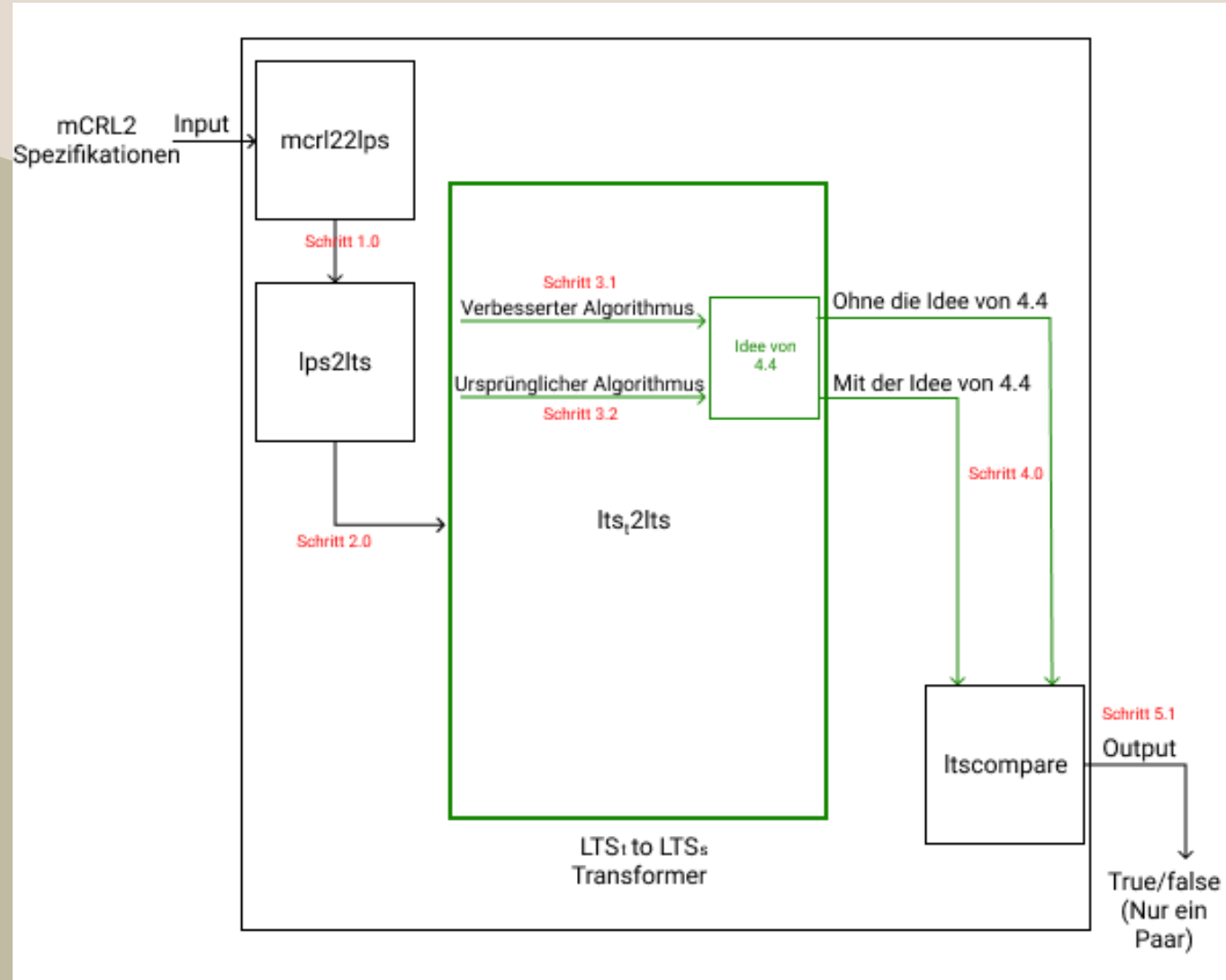
Kernidee



Kernidee

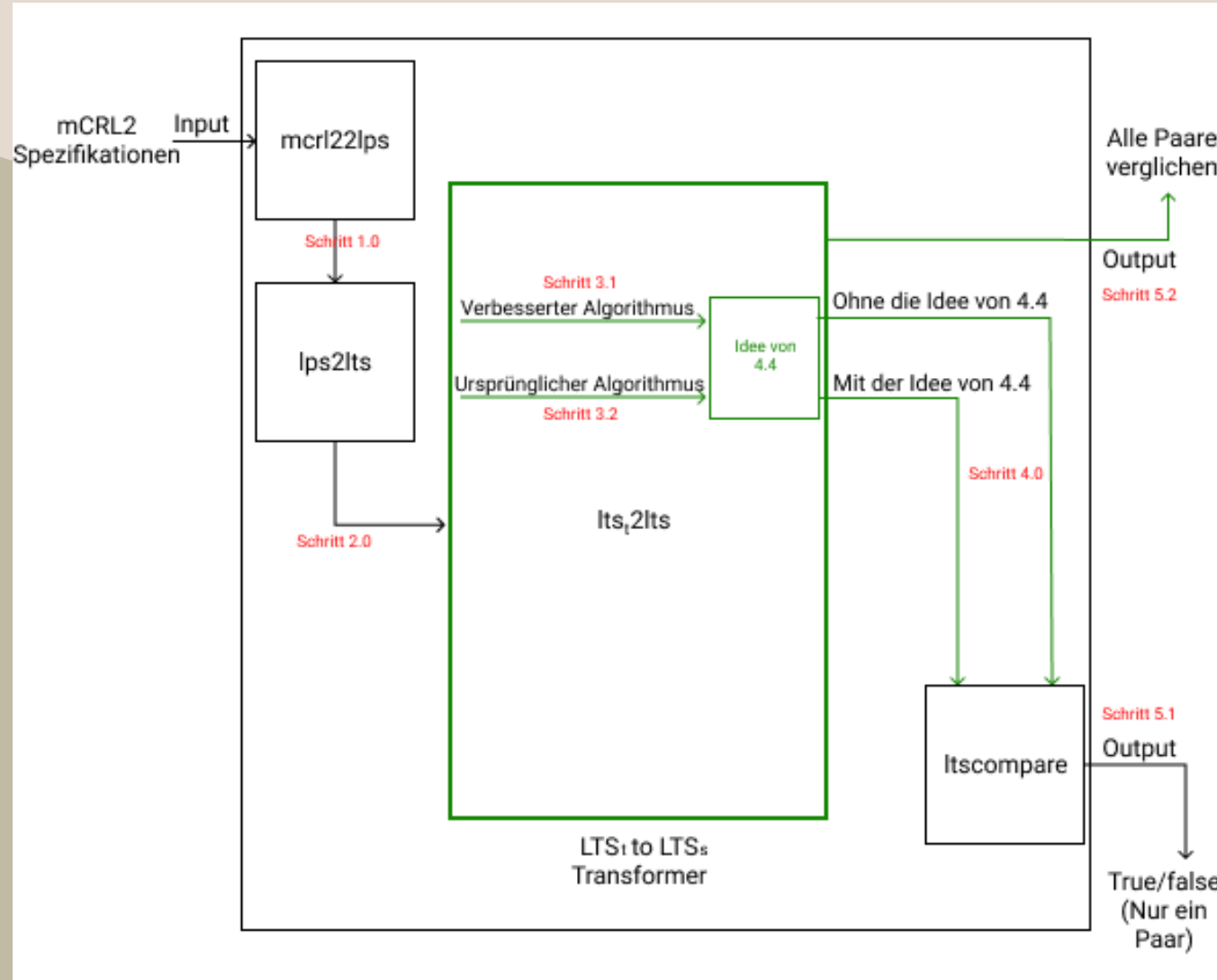


Kernidee



Kernidee

```
Checking if two mcr12 files exists...
Files exist.
Transforming the files into lts.t...
Transforming lts.t into lts.s...
Comparing the systems of reactive bisimilarity using strong bisimilarity definition...
(0,0)
(1,1)
(2,2)
(3,3)
(4,4)
Execution time: 2961998596956 - 296199816341 = 398 ms
```



Ursprünglicher Algorithmus



Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1:  $processes, allSharedEnvironments, processSharedEnvironment$ 
2:  $newProcesses \leftarrow emptyList$ 
3: for  $process \in processes$  do
4:    $processSharedEnvironment.add(process.getName(), allSharedEnvironments)$ 
5:    $ApplyRule2(allSharedEnvironments, process)$ 
6:   for  $transition \in process.getOriginalTransitions()$  do
7:      $processEnvironments \leftarrow processSharedEnvironment.getValueOf(process.getName())$ 
8:     for  $environment \in processEnvironments$  do
9:       if  $transition.getAction().isTauAction()$  then
10:         $ApplyRule1(process, transition)$ 
11:         $ApplyRule4(process, transition, environment)$ 
12:      else if  $transition.getAction().isNormalAction()$  then
13:         $ApplyRule3(process, transition, environment)$ 
14:      else if  $transition.getAction().isTimeAction()$  then
15:         $ApplyRule5(process, environment)$ 
16:         $ApplyRule6(process, transition, environment)$ 
17:      end if
18:    end for
19:  end for
20:   $applyRule5OfRemainingEnvironments(process, processSharedEnvironment)$ 
21: end for
22:  $result \leftarrow getNewProcesses()$ 
```

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$2) \frac{}{v(p) \xrightarrow{E_X}_v v_X(p)} \quad X \subseteq A$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$1) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_v \mathcal{V}(p')}$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$4) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}_X(p) \xrightarrow{\tau}_v \mathcal{V}_X(p')}$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a}_v v(p')} \quad a \in X$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$5) \frac{p \not\rightarrow \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon}_v v(p)}$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$6) \frac{p \not\rightarrow^q \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t}_v v_X(p')}$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$5) \frac{p \not\rightarrow \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon}_v v(p)}$$

A directed graph with three nodes labeled p_1 , p_2 , and p_3 . The nodes are arranged in a triangle. Directed edges are labeled with letters: an edge from p_1 to p_2 is labeled t , an edge from p_1 to p_3 is labeled v , an edge from p_2 to p_3 is labeled a , and an edge from p_3 to p_1 is labeled b . There is also a horizontal edge from p_2 to p_3 labeled τ .

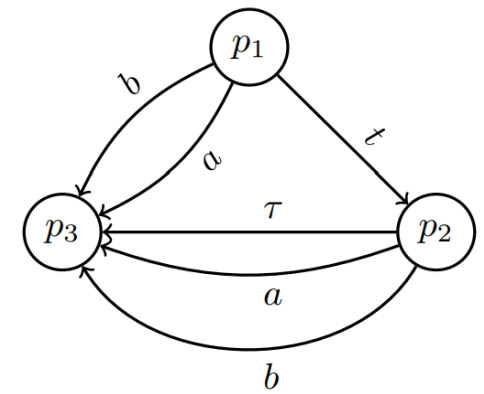
Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$\{\{\}, \{a\}, \{b\}, \{a, b\}\}$



Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

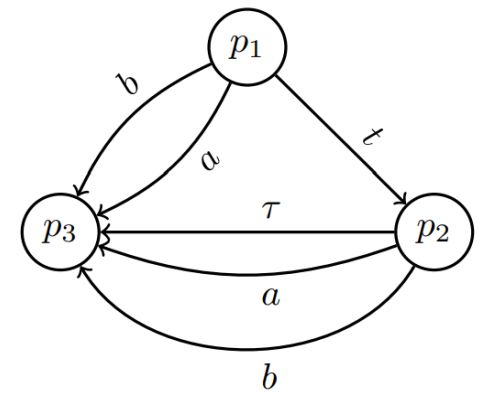
Require:

```

1:  $processes, allSharedEnvironments, processSharedEnvironment$ 
2:  $newProcesses \leftarrow emptyList$ 
3: for  $process \in processes$  do
4:    $processSharedEnvironment.add(process.getName(), allSharedEnvironments)$ 
5:   ApplyRule2( $allSharedEnvironments, process$ )
6:   for  $transition \in process.getOriginalTransitions()$  do
7:      $processEnvironments \leftarrow processSharedEnvironment.getValueOf(process.getName())$ 
8:     for  $environment \in processEnvironments$  do
9:       if  $transition.getAction().isTauAction()$  then
10:         $ApplyRule1(process, transition)$ 
11:         $ApplyRule4(process, transition, environment)$ 
12:      else if  $transition.getAction().isNormalAction()$  then
13:         $ApplyRule3(process, transition, environment)$ 
14:      else if  $transition.getAction().isTimeAction()$  then
15:         $ApplyRule5(process, environment)$ 
16:         $ApplyRule6(process, transition, environment)$ 
17:      end if
18:    end for
19:  end for
20:   $applyRule5OfRemainingEnvironments(process, processSharedEnvironment)$ 
21: end for
22:  $result \leftarrow getNewProcesses()$ 

```

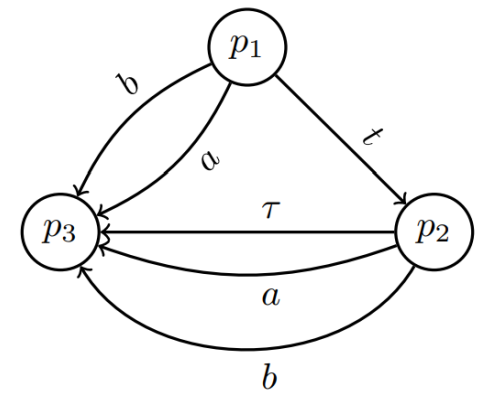
$\{\{\}, \{a\}, \{b\}, \{a, b\}\}$



$$2) \frac{}{v(p) \xrightarrow{E_X} v v_X(p)} X \subseteq A$$

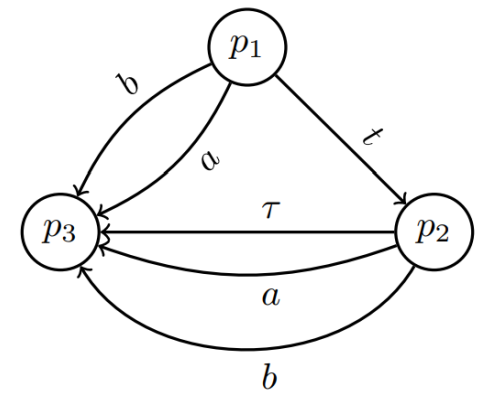
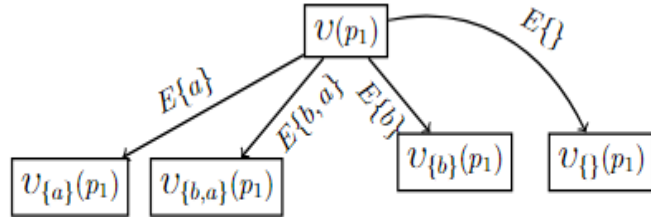
Ursprünglicher Algorithmus

$v(p_1)$



$$2) \frac{\quad}{v(p) \xrightarrow{E_X}_v v_X(p)} X \subseteq A$$

Ursprünglicher Algorithmus



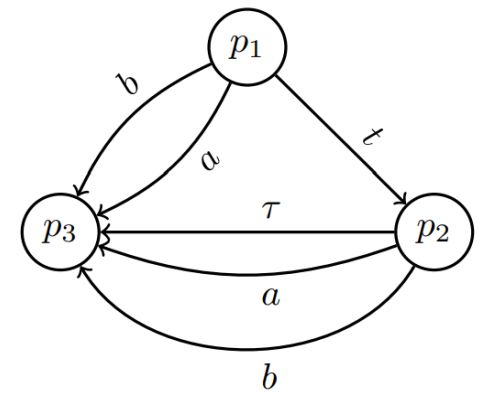
$$2) \frac{\quad}{u(p) \xrightarrow{E_X} u_X(p)} X \subseteq A$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```



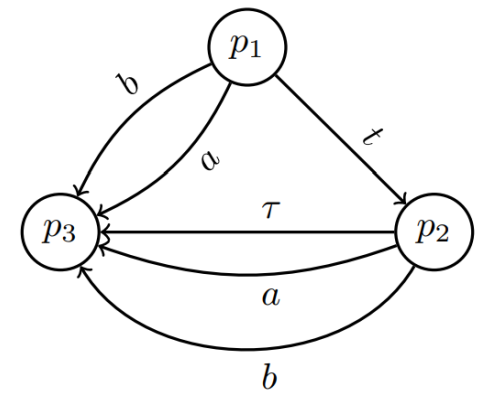
$$2) \frac{}{v(p) \xrightarrow{E_X} v v_X(p)} X \subseteq A$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```



$$2) \frac{}{v(p) \xrightarrow{E_X} v v_X(p)} X \subseteq A$$

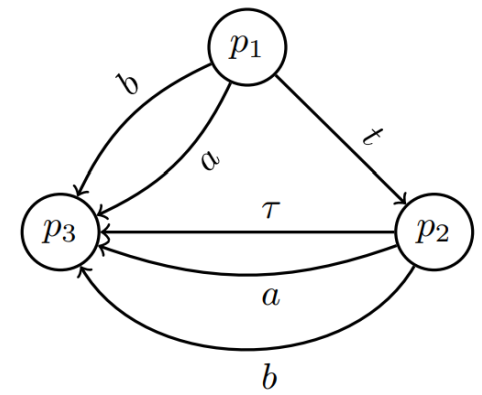
Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```

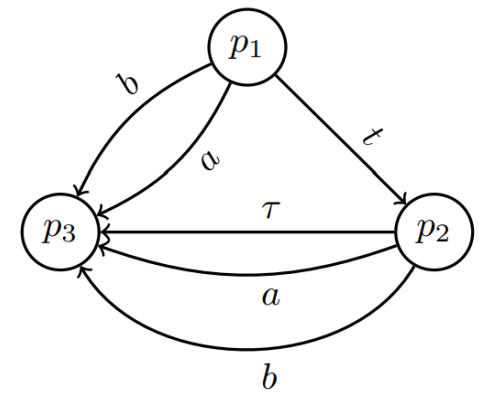
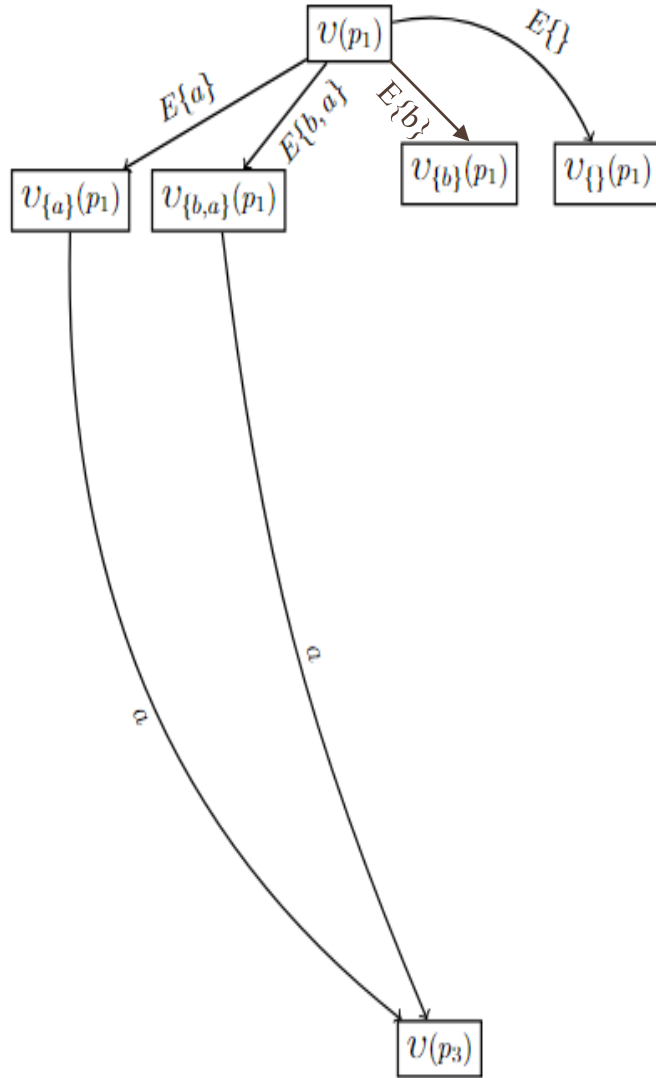
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
  
```



$$2) \frac{}{v(p) \xrightarrow{E_X}_v v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a}_v v(p')} a \in X$$

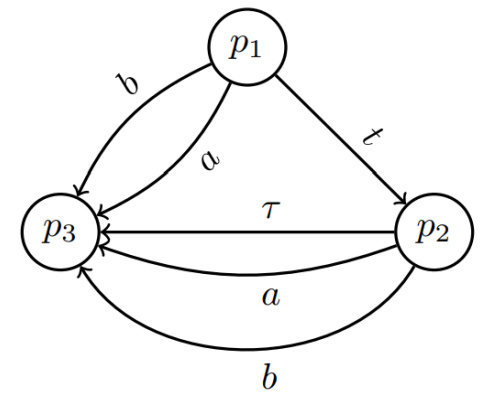
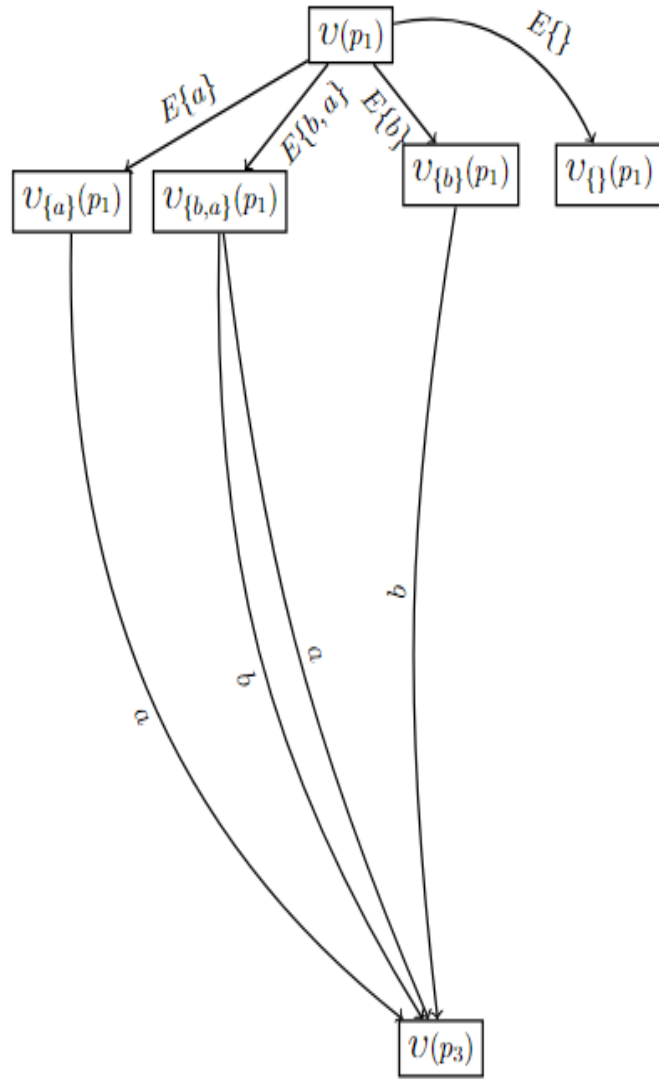
Ursprünglicher Algorithmus



$$2) \frac{}{v(p) \xrightarrow{E_X}_v v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a}_v v(p')} a \in X$$

Ursprünglicher Algorithmus



$$2) \frac{}{v(p) \xrightarrow{E_X} v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a} v(p')} a \in X$$

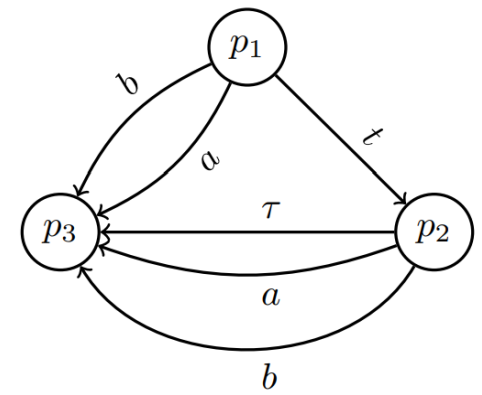
Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```

1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
  
```



$$2) \frac{}{v(p) \xrightarrow{E_X} v v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a} v v(p')} a \in X$$

$$5) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\varepsilon} v v(p)}$$

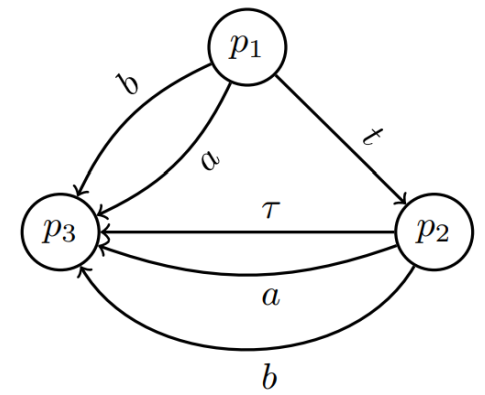
Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```

1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
  
```



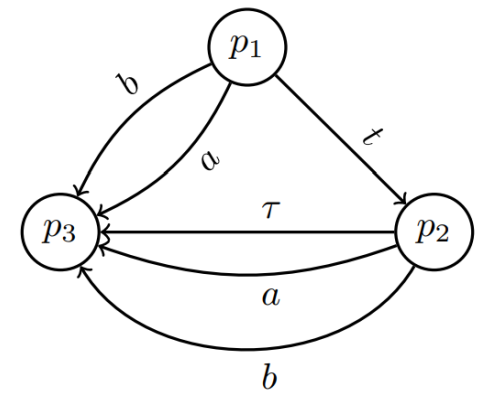
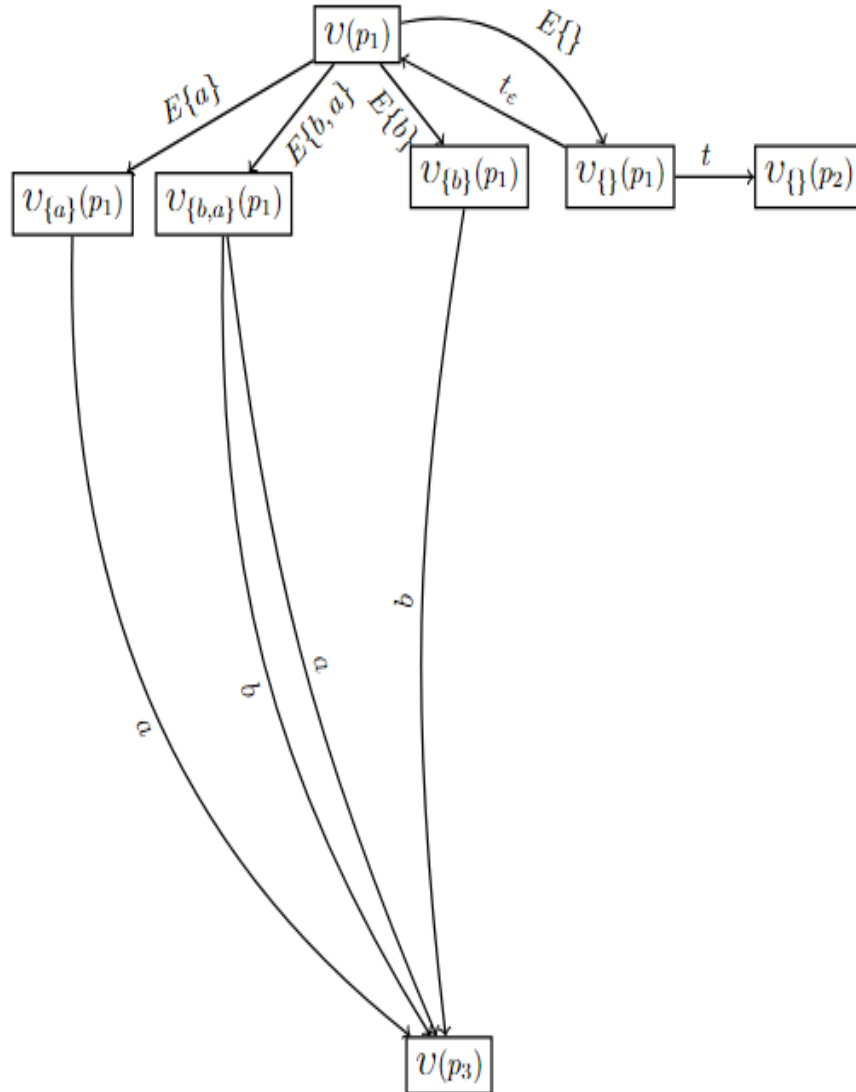
$$2) \frac{}{v(p) \xrightarrow{E_X}_v v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a}_v v(p')} a \in X$$

$$5) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon}_v v(p)}$$

$$6) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t}_v v_X(p')}$$

Ursprünglicher Algorithmus



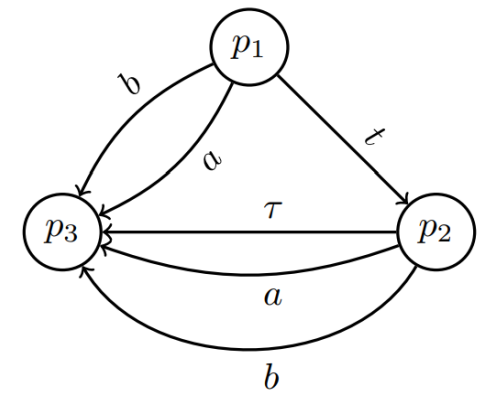
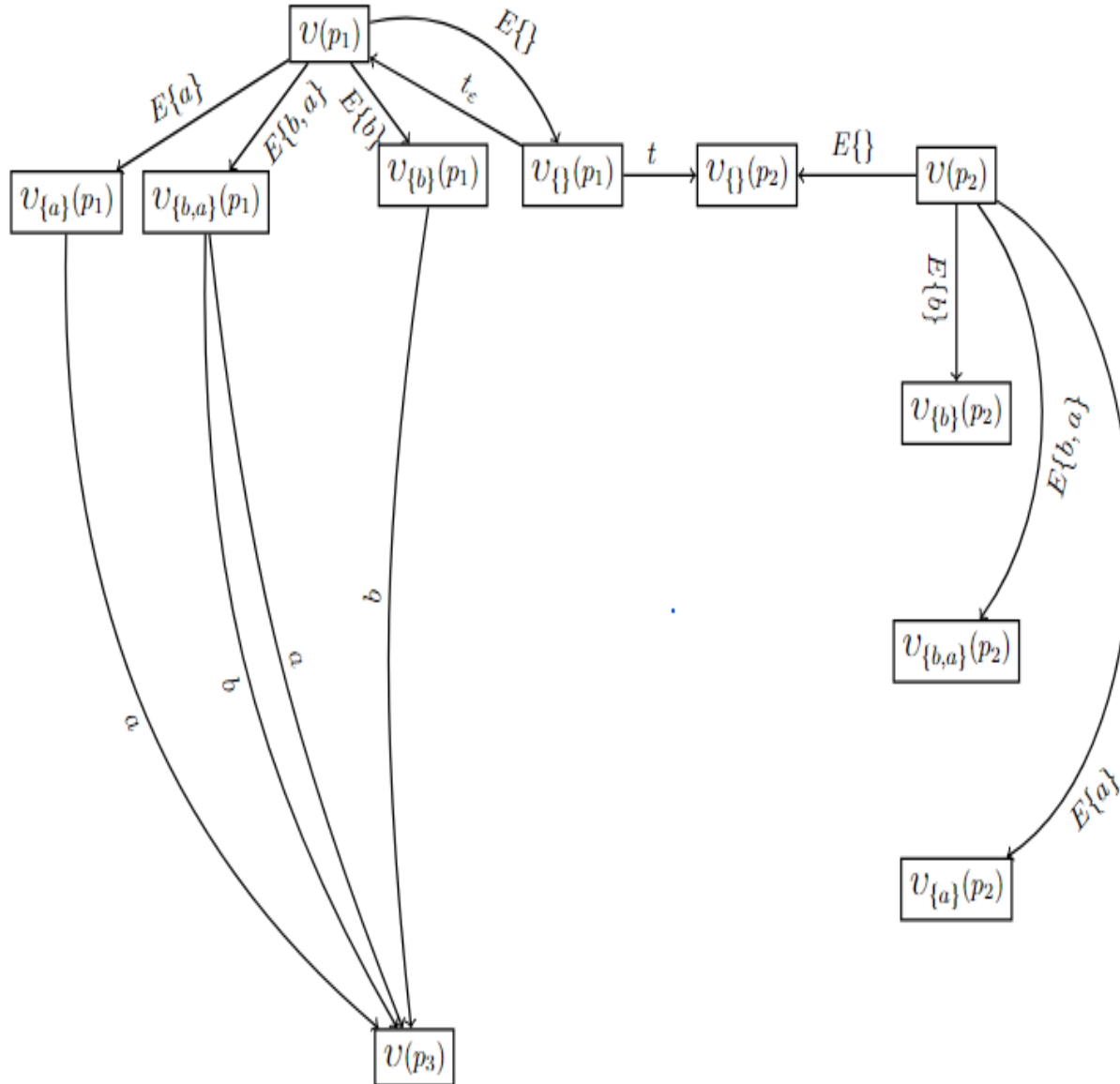
$$2) \frac{}{v(p) \xrightarrow{E_X}_v v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a}_v v(p')} a \in X$$

$$5) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon}_v v(p)}$$

$$6) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t}_v v_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{}{u(p) \xrightarrow{E_X \rightarrow_v} u_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{u_X(p) \xrightarrow{a \rightarrow_v} u(p')} a \in X$$

$$5) \frac{p \not\xrightarrow{\varphi} \forall \alpha \in X \cup \{\tau\}}{u_X(p) \xrightarrow{t_\varepsilon \rightarrow_v} u(p)}$$

$$6) \frac{p \not\xrightarrow{\varphi} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{u_X(p) \xrightarrow{t \rightarrow_v} u_X(p')}$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$1) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_v \mathcal{V}(p')}$$

Ursprünglicher Algorithmus

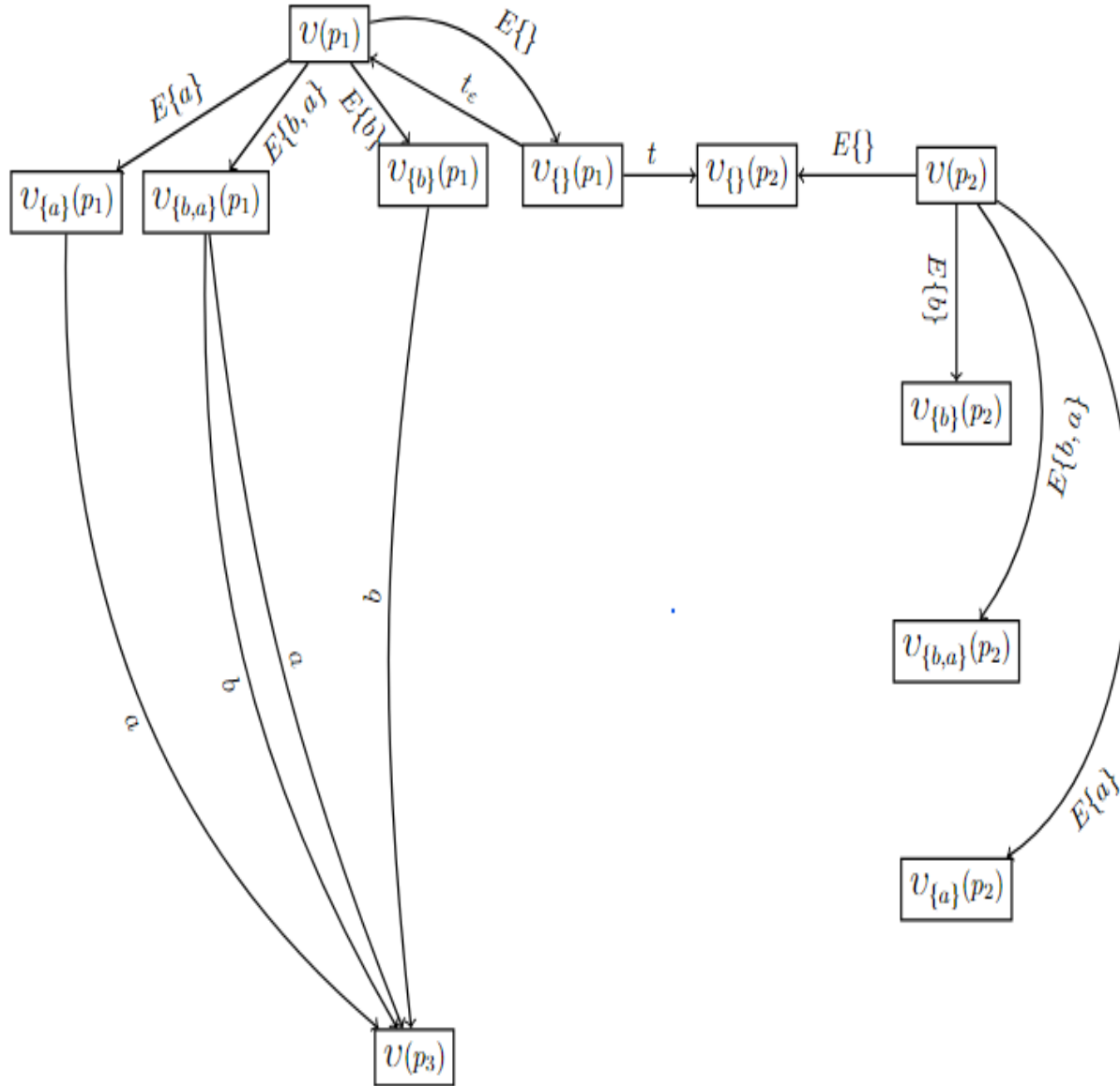
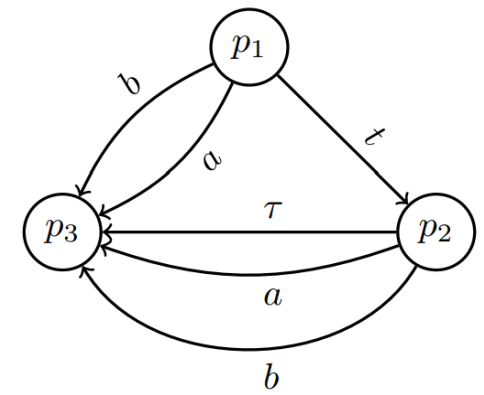
Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses  $\leftarrow$  emptyList
3: for process  $\in$  processes do
4:   processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:   ApplyRule2(allSharedEnvironments, process)
6:   for transition  $\in$  process.getOriginalTransitions() do
7:     processEnvironments  $\leftarrow$  processSharedEnvironment.getValueOf(process.getName())
8:     for environment  $\in$  processEnvironments do
9:       if transition.getAction().isTauAction() then
10:        ApplyRule1(process, transition)
11:        ApplyRule4(process, transition, environment)
12:      else if transition.getAction().isNormalAction() then
13:        ApplyRule3(process, transition, environment)
14:      else if transition.getAction().isTimeAction() then
15:        ApplyRule5(process, environment)
16:        ApplyRule6(process, transition, environment)
17:      end if
18:    end for
19:  end for
20:  applyRule5OfRemainingEnvironments(process, processSharedEnvironment)
21: end for
22: result  $\leftarrow$  getNewProcesses()
```

$$4) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}_X(p) \xrightarrow{\tau}_v \mathcal{V}_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{}{U(p) \xrightarrow{E_X \rightarrow_v} U_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{U_X(p) \xrightarrow{a \rightarrow_v} U(p')} a \in X$$

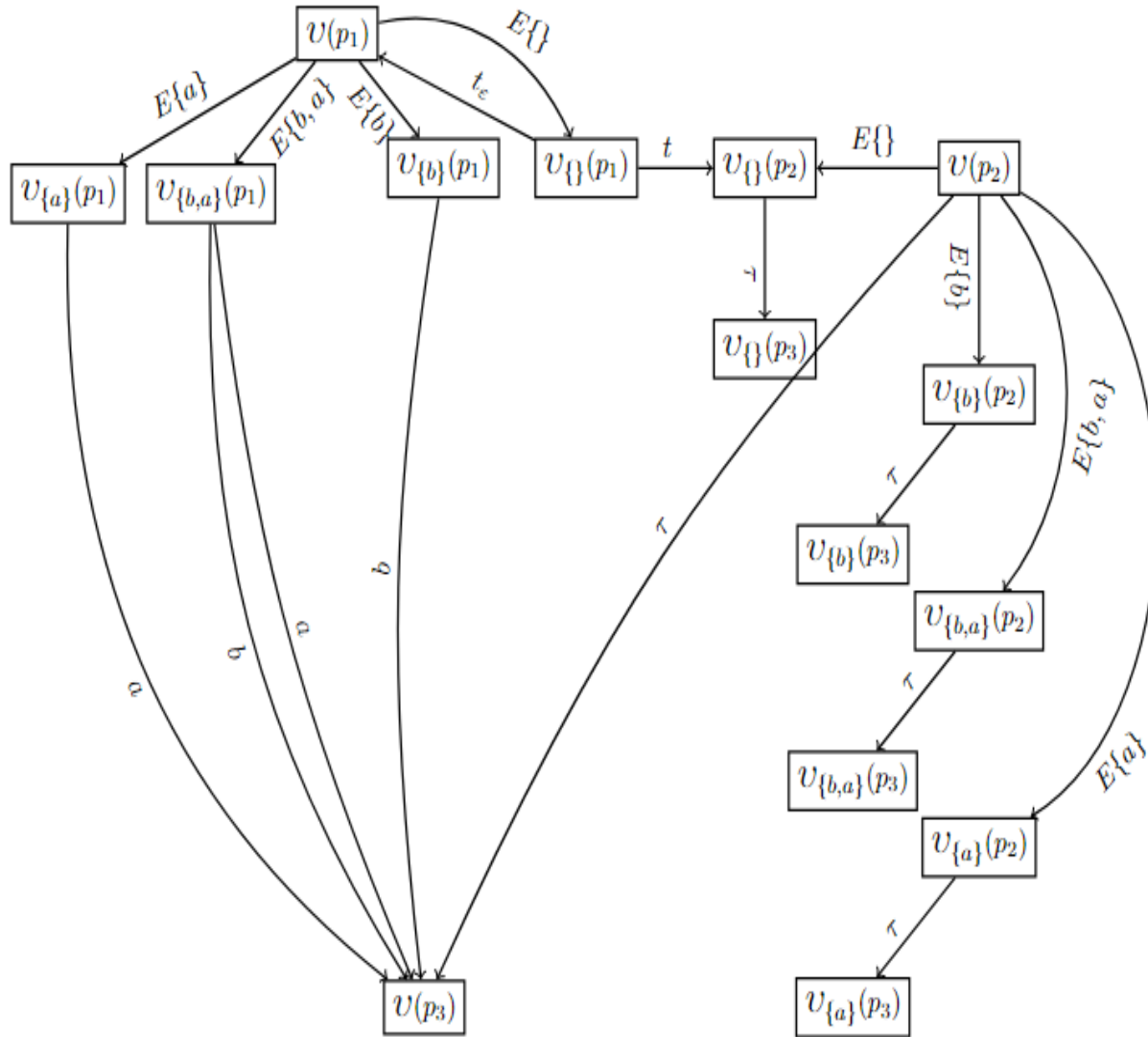
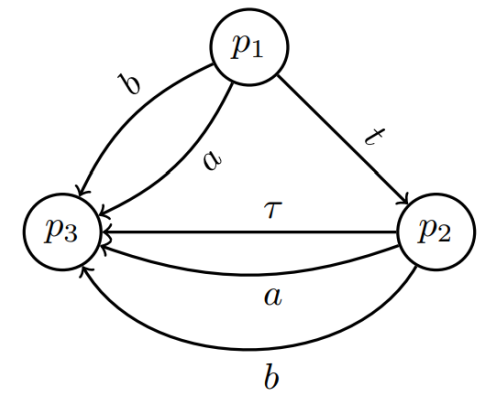
$$5) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\}}{U_X(p) \xrightarrow{t_\epsilon \rightarrow_v} U(p)}$$

$$6) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{U_X(p) \xrightarrow{t \rightarrow_v} U_X(p')}$$

$$1) \frac{p \xrightarrow{\tau} p'}{U(p) \xrightarrow{\tau \rightarrow_v} U(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{U_X(p) \xrightarrow{\tau \rightarrow_v} U_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{}{v(p) \xrightarrow{E_X \rightarrow v} v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a \rightarrow v} v(p')} a \in X$$

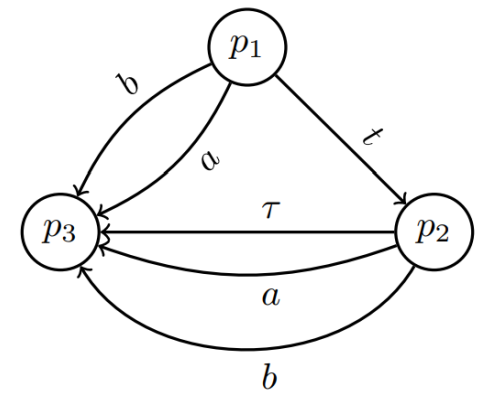
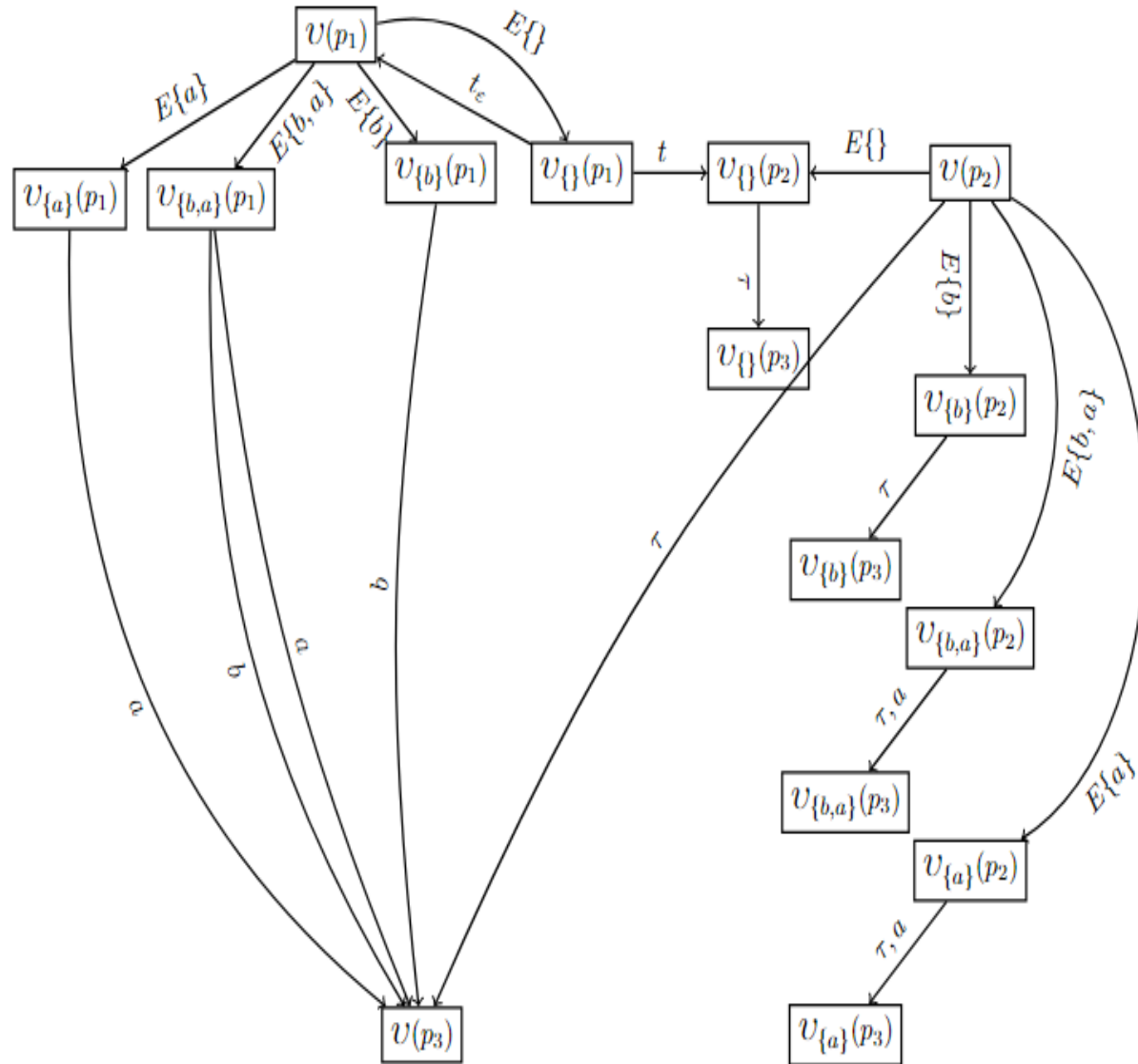
$$5) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon \rightarrow v} v(p)}$$

$$6) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t \rightarrow v} v_X(p')}$$

$$1) \frac{p \xrightarrow{\tau} p'}{v(p) \xrightarrow{\tau \rightarrow v} v(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau \rightarrow v} v_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{}{v(p) \xrightarrow{E_X \rightarrow_v} v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a \rightarrow_v} v(p')} a \in X$$

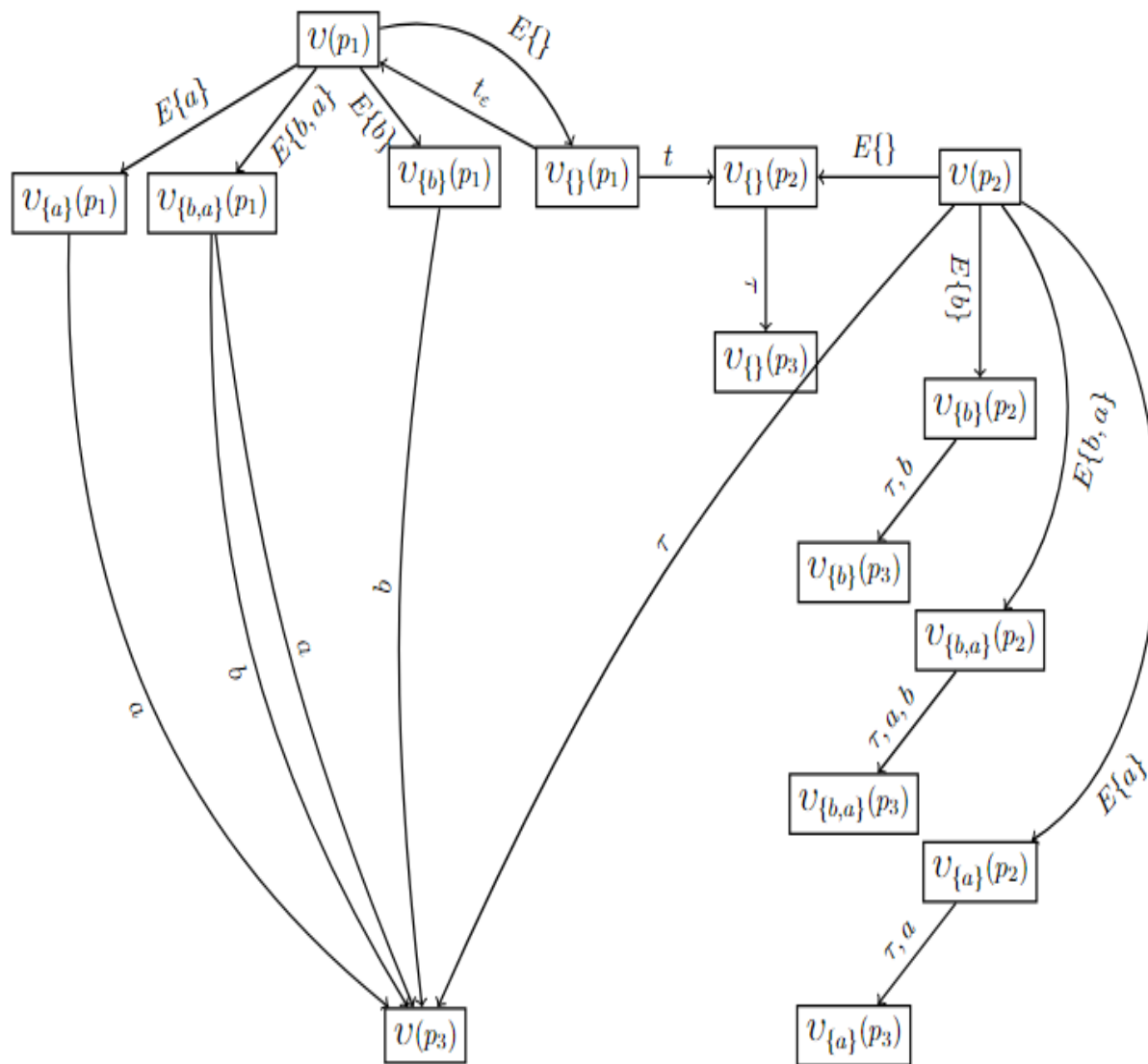
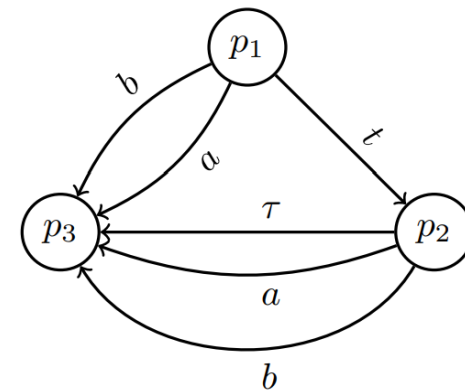
$$5) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon \rightarrow_v} v(p)}$$

$$6) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t \rightarrow_v} v_X(p')}$$

$$1) \frac{p \xrightarrow{\tau} p'}{v(p) \xrightarrow{\tau \rightarrow_v} v(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau \rightarrow_v} v_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{\quad}{\mathcal{V}(p) \xrightarrow{E_X} \mathcal{V}_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{\mathcal{V}_X(p) \xrightarrow{a}_{\mathcal{V}} \mathcal{V}(p')} \quad a \in X$$

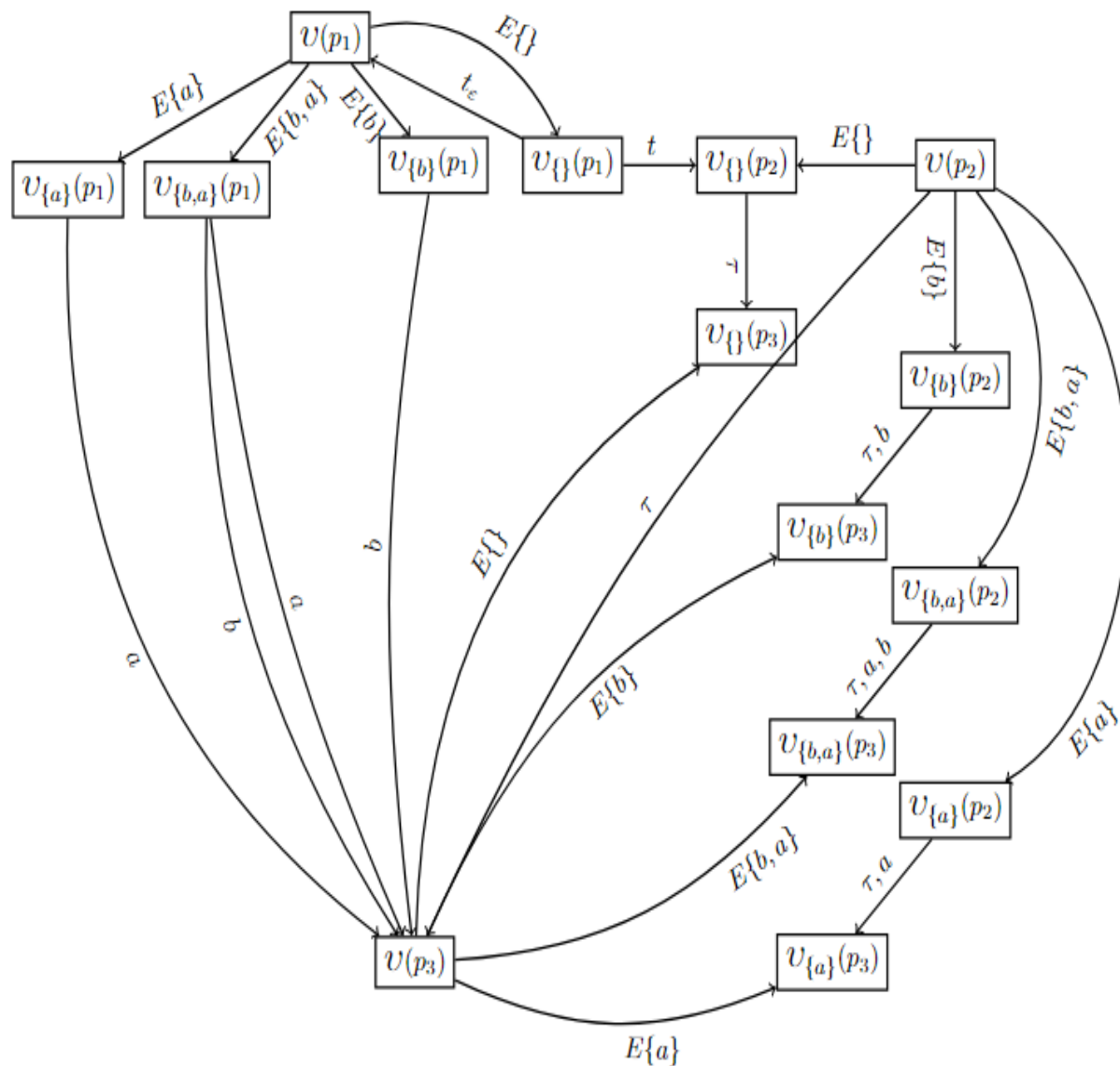
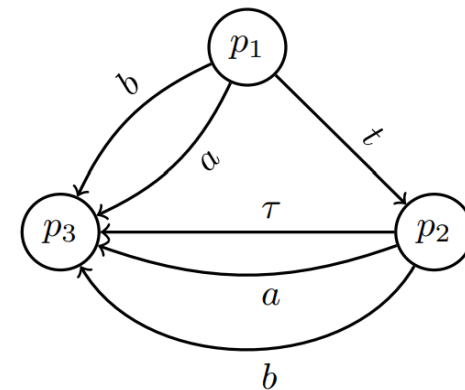
$$5) \frac{p \not\rightarrow^q \forall \alpha \in X \cup \{\tau\}}{\mathcal{V}_X(p) \xrightarrow{t_\varepsilon}_v \mathcal{V}(p)}$$

$$6) \frac{p \not\rightarrow^{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\mathcal{V}_X(p) \xrightarrow{t}_v \mathcal{V}_X(p')}$$

$$1) \quad \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_{\mathcal{V}} \mathcal{V}(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau}_v v_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{\quad}{\mathcal{V}(p) \xrightarrow{E_X} \mathcal{V}_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{\mathcal{V}_X(p) \xrightarrow{a}_{\mathcal{V}} \mathcal{V}(p')} \quad a \in X$$

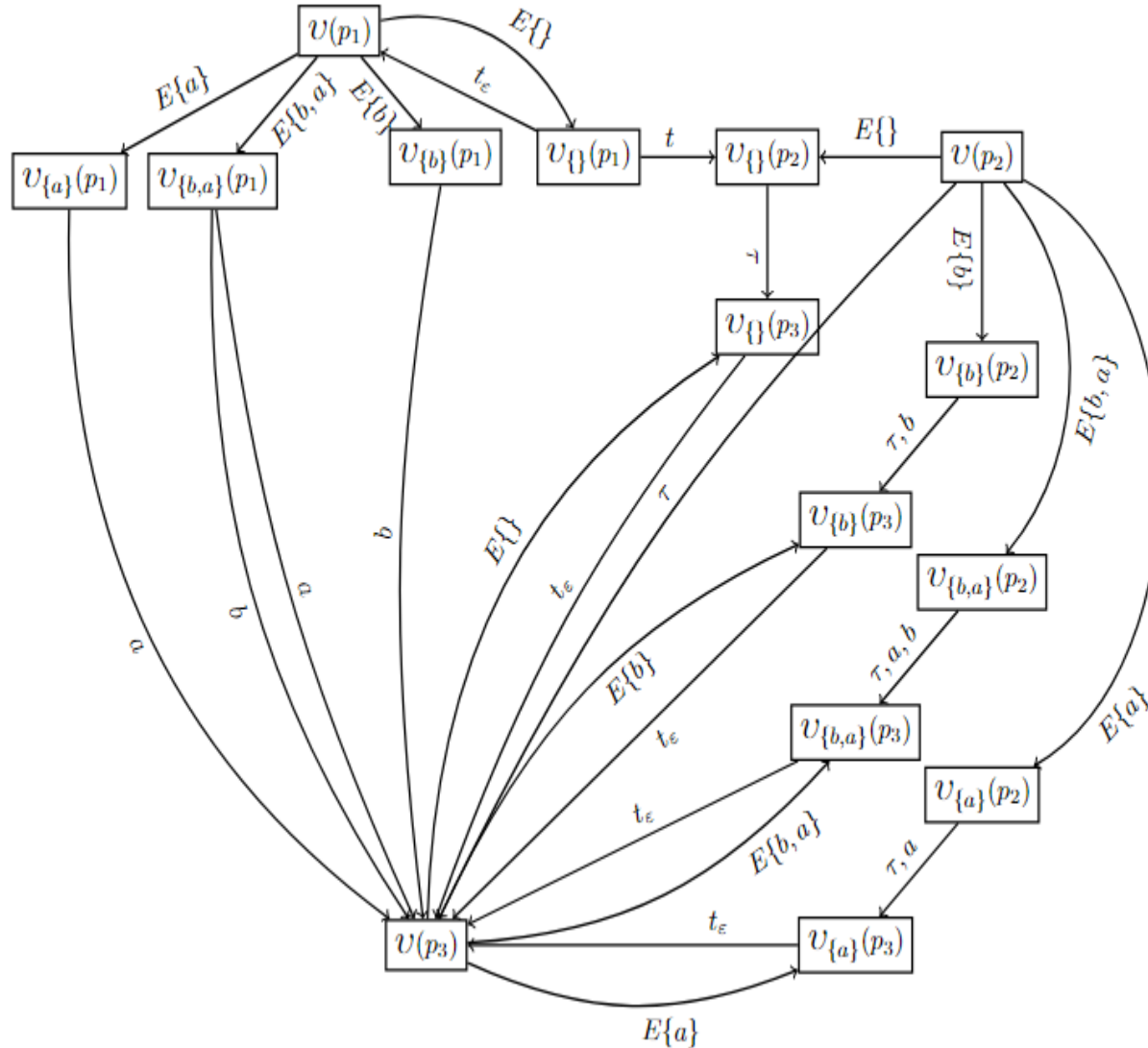
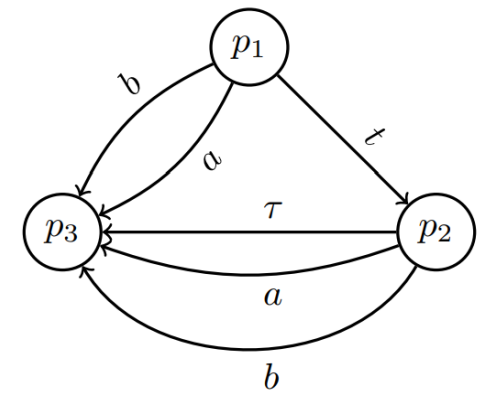
$$5) \frac{p \not\rightarrow^q \forall \alpha \in X \cup \{\tau\}}{\mathcal{V}_X(p) \xrightarrow{t_\varepsilon}_{\mathcal{V}} \mathcal{V}(p)}$$

$$6) \frac{p \not\rightarrow^{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\mathcal{V}_X(p) \xrightarrow{t}_v \mathcal{V}_X(p')}$$

$$1) \quad \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_{\mathcal{V}} \mathcal{V}(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau}_v v_X(p')}$$

Ursprünglicher Algorithmus



$$2) \frac{}{v(p) \xrightarrow{E_X \rightarrow v} v_X(p)} X \subseteq A$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a \rightarrow v} v(p')} a \in X$$

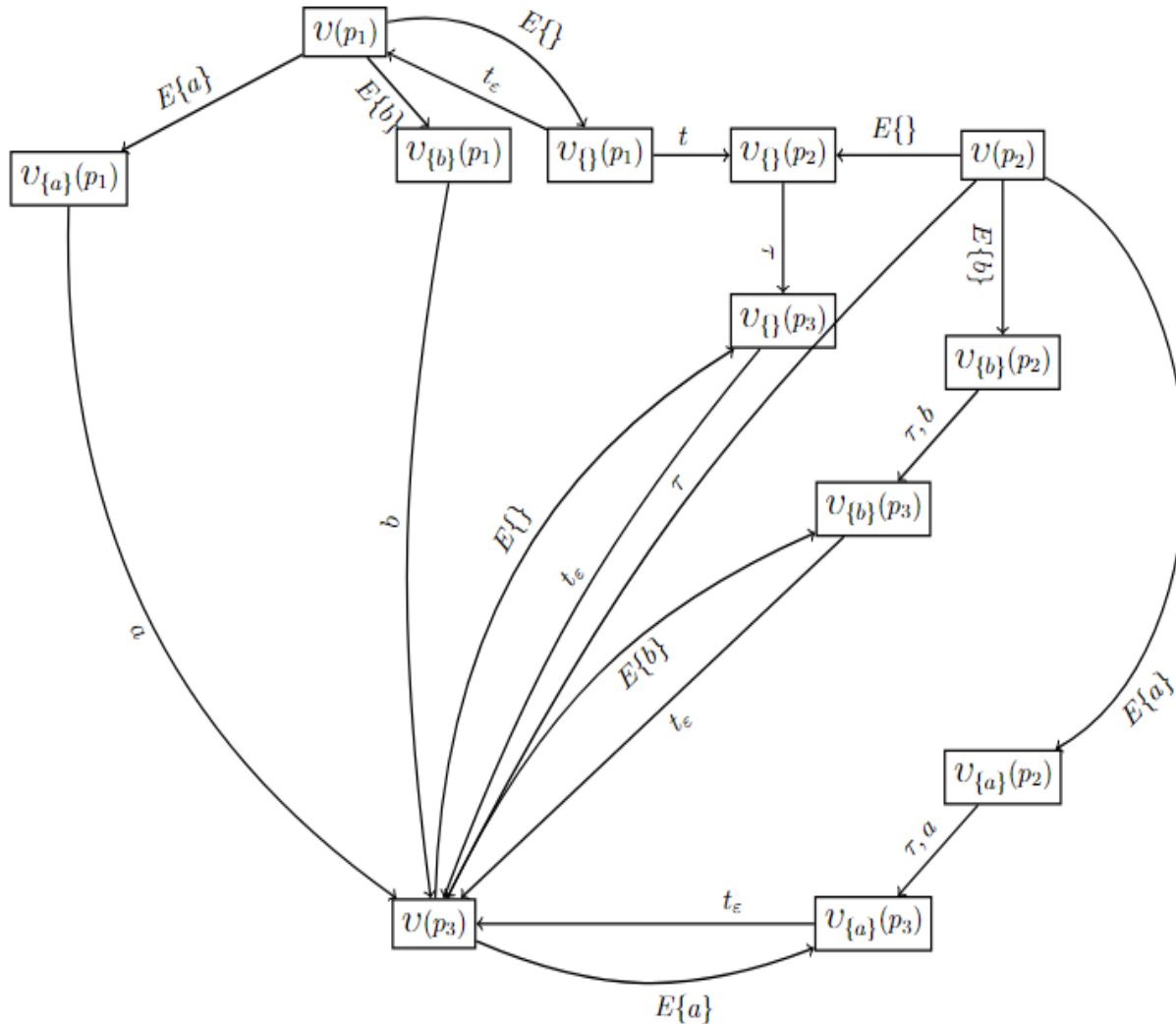
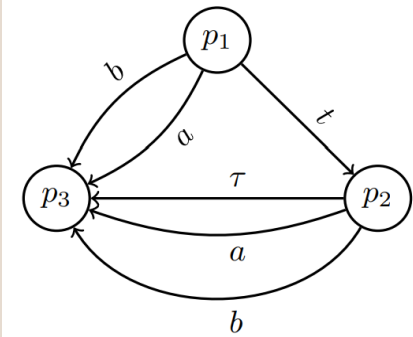
$$5) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\}}{v_X(p) \xrightarrow{t_\epsilon \rightarrow v} v(p)}$$

$$6) \frac{p \not\xrightarrow{\tau} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t \rightarrow v} v_X(p')}$$

$$1) \frac{p \xrightarrow{\tau} p'}{v(p) \xrightarrow{\tau \rightarrow v} v(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau \rightarrow v} v_X(p')}$$

Verbesserte Algorithmus


$$\overline{U(p) \xrightarrow{E_X} U_X(p)} \quad X \in U, \text{ mit}$$
$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t \in \mathcal{I}(p), p \in Proc \wedge t \text{ ist die Timeout-Aktion}\}$$

$$3) \frac{p \xrightarrow{a} p'}{v_X(p) \xrightarrow{a_v} v(p')} \quad a \in X$$

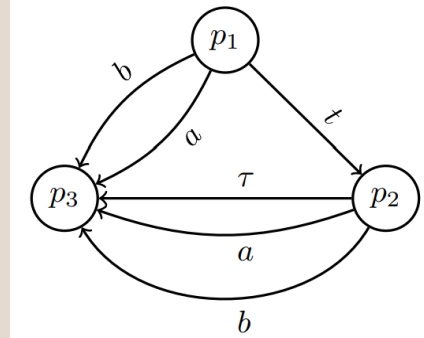
$$5) \frac{p \not\rightarrow^q \forall \alpha \in X \cup \{\tau\}}{\nu_X(p) \xrightarrow{t_\varepsilon}_v \nu(p)}$$

$$6) \frac{p \not\rightarrow^x \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{v_X(p) \xrightarrow{t}_v v_X(p')}$$

$$1) \quad \frac{p \xrightarrow{\tau} p'}{v(p) \xrightarrow{\tau}_v v(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{v_X(p) \xrightarrow{\tau_v} v_X(p')}$$

Verbesserte Algorithmus



$$\frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} \quad X \in U, \text{ mit}$$

$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t \in \mathcal{I}(p), p \in Proc \wedge t \text{ ist die Timeout-Aktion}\}$$

$$3) \frac{p \xrightarrow{a} p' \quad a \in X}{\mathcal{V}_X(p) \xrightarrow{a}_v \mathcal{V}(p')}$$

$$5) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\mathcal{V}_X(p) \xrightarrow{t_\varepsilon}_v \mathcal{V}(p)}$$

$$6) \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\mathcal{V}_X(p) \xrightarrow{t}_v \mathcal{V}_X(p')}$$

$$1) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}(p) \xrightarrow{\tau}_v \mathcal{V}(p')}$$

$$4) \frac{p \xrightarrow{\tau} p'}{\mathcal{V}_X(p) \xrightarrow{\tau}_v \mathcal{V}_X(p')}$$

Einzigste Umgebungsaktion

$\{\{\}, \{a\}, \{b\}, \{a, b\}\}$

Einzigste Umgebungsaktion

$$\{\{\}, \{a\}, \{b\}, \{a, b\}\} \longrightarrow E_{\{\dots\}}$$

Einzigste Umgebungsaktion

$\{\{\}, \{a\}, \{b\}, \{a, b\}\} \longrightarrow E_{\{\dots\}}$

```
42 (17,"noeE",26)
43 (17,"a",27)
44 (17,"a_b",28)
45 (17,"a_b_g",29)
46 (17,"a_b_g_f",30)
47 (17,"a_b_f",31)
48 (17,"a_g",32)
49 (17,"a_g_f",33)
50 (17,"a_f",34)
51 (17,"b",35)
```

Einzigste Umgebungsaktion

$\{\{\}, \{a\}, \{b\}, \{a, b\}\} \longrightarrow E_{\{\dots\}}$

```
42 (17,"noeE",26)
43 (17,"a",27)
44 (17,"a_b",28)
45 (17,"a_b_g",29)
46 (17,"a_b_g_f",30)
47 (17,"a_b_f",31)
48 (17,"a_g",32)
49 (17,"a_g_f",33)
50 (17,"a_f",34)
51 (17,"b",35)
```



Einzigste Umgebungsaktion

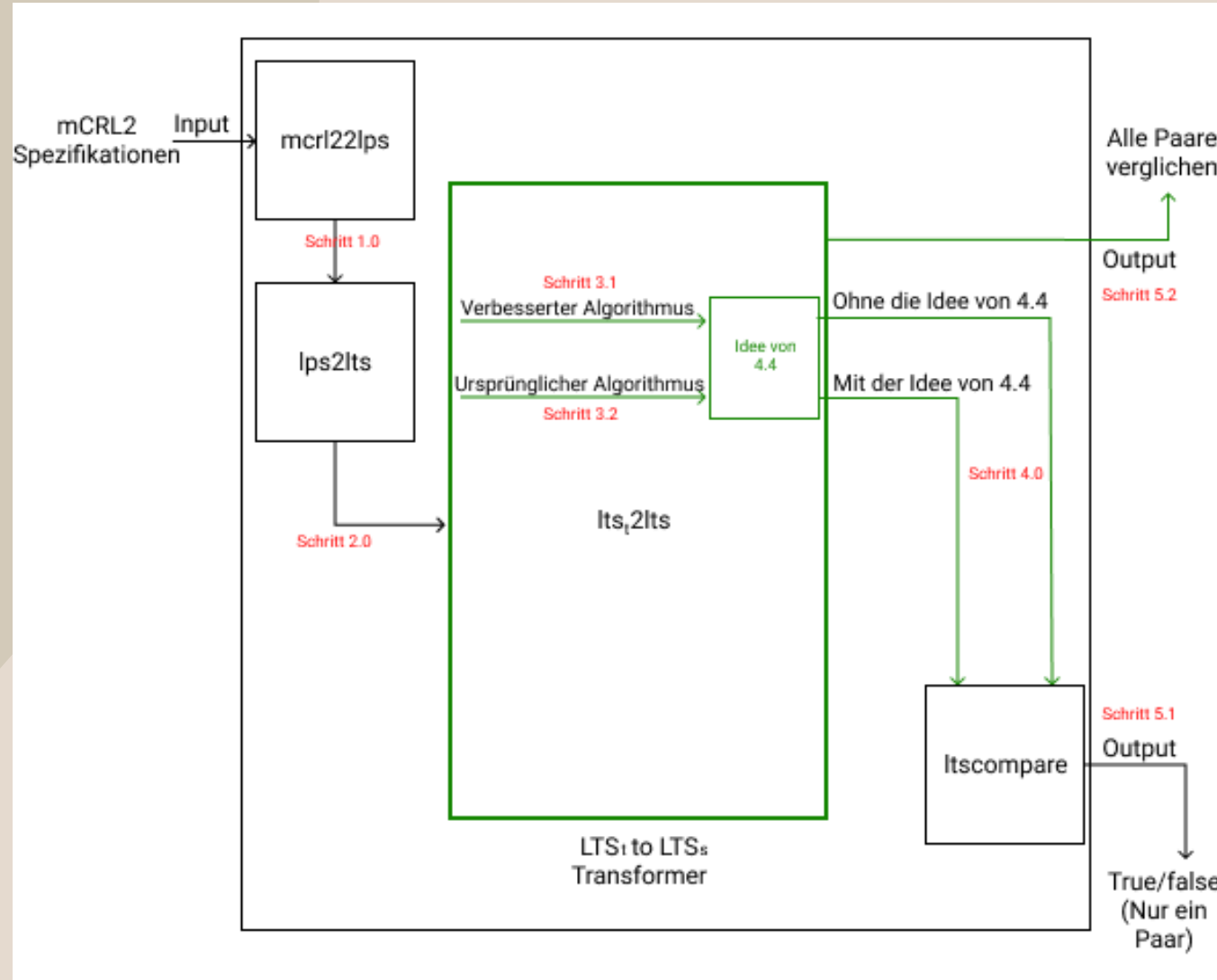
$\{\{\}, \{a\}, \{b\}, \{a, b\}\} \longrightarrow E_{\{\dots\}}$

```
42 (17, "noeE", 26)
43 (17, "a", 27)
44 (17, "a_b", 28)
45 (17, "a_b_g", 29)
46 (17, "a_b_g_f", 30)
47 (17, "a_b_f", 31)
48 (17, "a_g", 32)
49 (17, "a_g_f", 33)
50 (17, "a_f", 34)
51 (17, "b", 35)
```

```
41 (16, "time", 25)
42 (17, "E{...}", 26)
43 (17, "E{...}", 27)
44 (17, "E{...}", 28)
45 (17, "E{...}", 29)
46 (17, "E{...}", 30)
47 (17, "E{...}", 31)
48 (17, "E{...}", 32)
49 (17, "E{...}", 33)
50 (17, "E{...}", 34)
51 (17, "E{...}", 35)
52 (17, "E{...}", 36)
53 (17, "E{...}", 37)
54 (17, "E{...}", 38)
55 (17, "E{...}", 39)
56 (17, "E{...}", 40)
57 (17, "E{...}", 41)
58 (18, "tau", 53)
59 (19, "tau", 62)
60 (20, "tau", 63)
61 (20, "g", 69)
```

Überprüfung eines oder aller Paare

Überprüfung eines oder aller Paare



Überprüfung eines oder aller Paare

Algorithm 2 Algorithmus von starker Bisimilarität

Require:

```
1: processList1
2: processList2
3: pairs  $\leftarrow$  processList1  $\times$  processList2
4: while true do
5:   pairs.removeIf(p  $\rightarrow$ 
6:     !checkSimulation(p2q, pairs, p.getFirst(), p.getLast()) ||
7:     !checkSimulation(q2p, pairs, p.getLast(), p.getFirst())
8:   )
9:   If pairs does not change beark
10: end while
11: return pairs
```

Algorithm 3 *checkSimulation*

Require:

```
1: leftToRight (check (p,q) or (q,p))
2: pairs
3: Prozess p from pair
4: Prozess q from pair
5: return p.getTransitions().allMatch(t1  $\rightarrow$ 
6:   q.getTransitions().anyMatch(t2  $\rightarrow$ 
7:     t1.getAction().getName().equals(t2.getAction().getName())
8:   &&
9:   areSuccessorsInR(leftToRight, pairs, t1.getSuccessor(), t2.getSuccessor())))
```

Anmerkungen

KOMPLEXITÄT



```
graph TD; A(( )) --- B((KOMPLEXITÄT)); B --- C(( )); C --- D(( )); D --- E((WERKZEUG DEMO)); E --- F(( ))
```

WERKZEUG DEMO

Komplexität

$$2) \frac{}{\nu(p) \xrightarrow{E_X}_v \nu_X(p)} X \subseteq A$$

Komplexität

$$2) \frac{}{\nu(p) \xrightarrow{E_X}_v \nu_X(p)} X \subseteq A$$



$$\frac{}{\nu(p) \xrightarrow{E_X}_v \nu_X(p)} X \in U, \text{ mit}$$

$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t \in \mathcal{I}(p), p \in \text{Proc} \wedge t \text{ ist die Timeout-Aktion}\}$$

Komplexität

$$2) \frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} X \subseteq A$$



$$\frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} X \in U, \text{ mit}$$

$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t = \mathcal{I}(p). p \in Proc \wedge t \text{ ist die Timeout-Aktion}\}$$



$$|Proc_v| = |Proc| \cdot (1 + 2^{|A|})$$

Komplexität

$$2) \frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} X \subseteq A$$



$$\frac{}{\mathcal{V}(p) \xrightarrow{E_X}_v \mathcal{V}_X(p)} X \in U, \text{ mit}$$

$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t = \mathcal{I}(p). p \in Proc \wedge t \text{ ist die Timeout-Aktion}\}$$



$$|Proc_v| = |Proc| \cdot (1 + 2^{|A|})$$



$$|Proc_v| = |Proc| \cdot (|A| + |P_t| + 1)$$

Komplexität

$$2) \frac{}{\nu(p) \xrightarrow{E_X}_\nu \nu_X(p)} X \subseteq A$$

$$\frac{}{\nu(p) \xrightarrow{E_X}_\nu \nu_X(p)} X \in U, \text{ mit}$$

$$U = \{\{a\} \mid a \in A\} \cup \{A \setminus \mathcal{I}(p) \mid t = \mathcal{I}(p), p \in Proc \wedge t \text{ ist die Timeout-Aktion}\}$$

$$|Proc_\nu| = |Proc| \cdot (1 + 2^{|A|})$$

$$|Proc_\nu| = |Proc| \cdot (|A| + |P_t| + 1)$$

Werkzeug Demo

Fazit

- Werkzeug zur Überprüfung reaktiver Bisimilarität
- Reduktion verbessert aber nicht bewiesen
- Ein Paar oder alle Paare
- Idee der einzigen Aktion



Vielen Dank

Zead Alshukairi

alshukairi@campus.tu-berlin.de