Automatisierte Reduktion von reaktiver zu starker Bisimilarität

Zead Alshukairi

Agenda

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

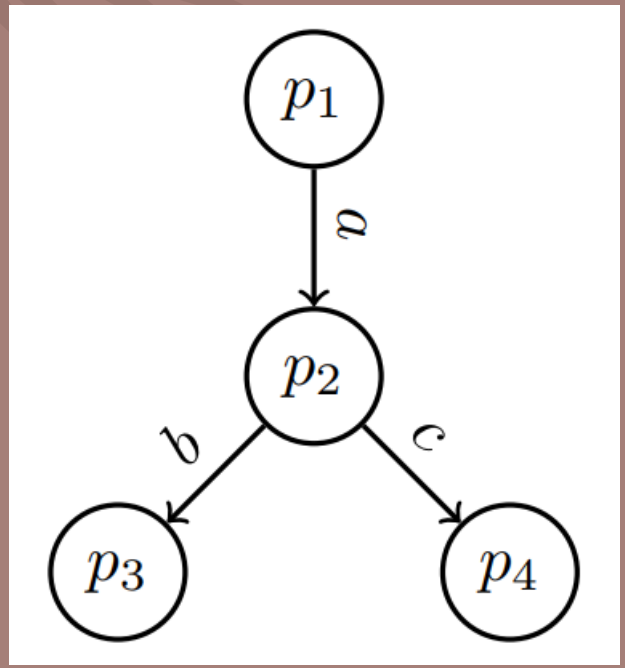
- Ziel meiner Arbeit
  - Bisimilaritäten
  - Kein Tool für reactive Bisimilarität
- Warum ist das wichtig
  - Timeouts in Anwendungen
  - Gegenseitiger Ausschluss
  - Kein Bedarf für manuelle Überprüfung
- Lösungsbestandteile
  - Starke Bisimilarität
  - Reaktive Bisimilarität
  - mCRL2-Projekt
  - Vordefinierte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

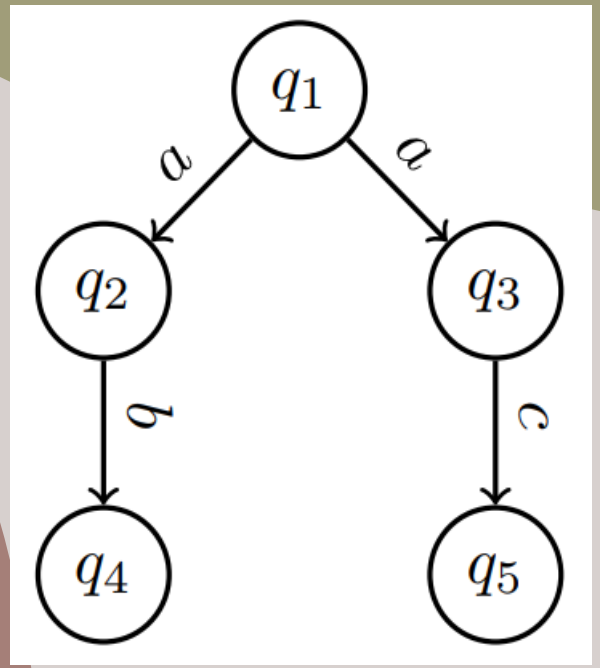
Grundlagen

- Starke Bisimilarität
- Reaktive Bisimilarität
- Die Reduktion von Max

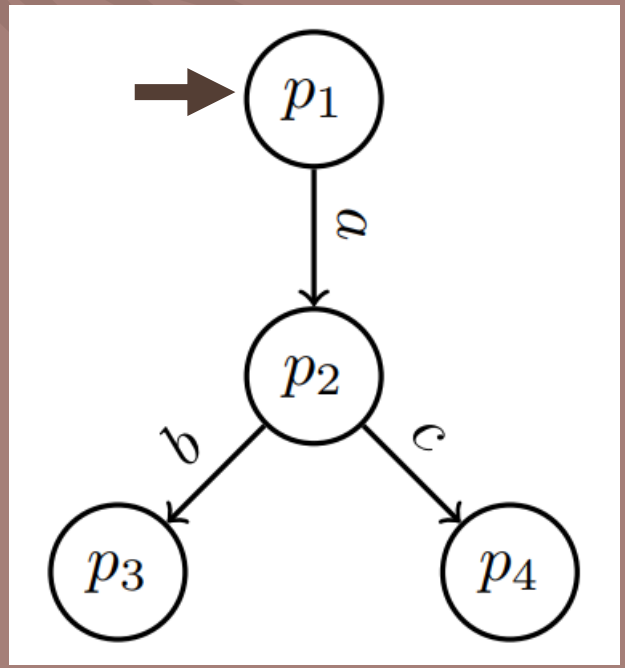
Starke Bisimilarität

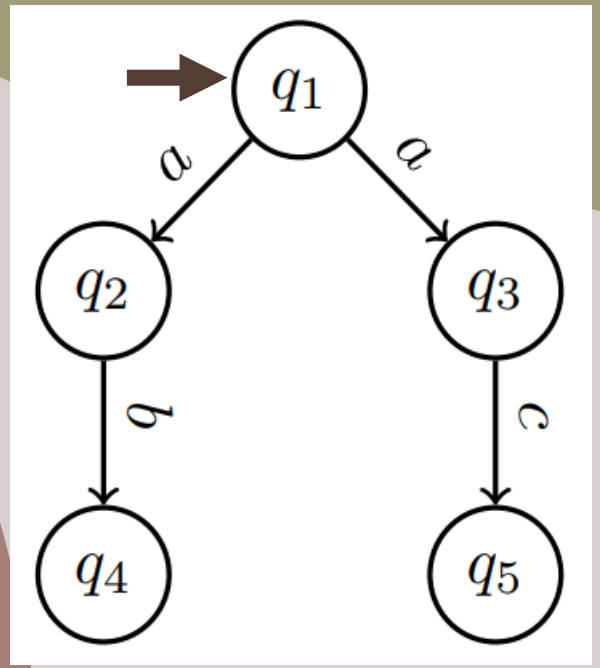
Starke Bisimilarität



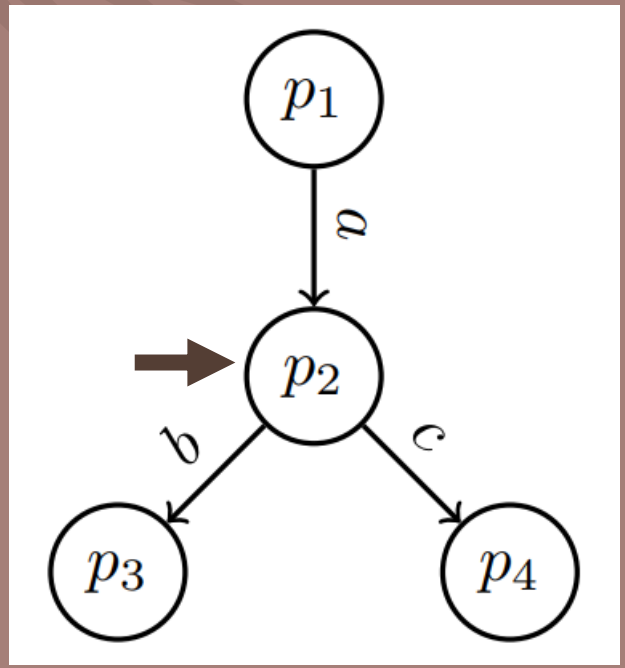


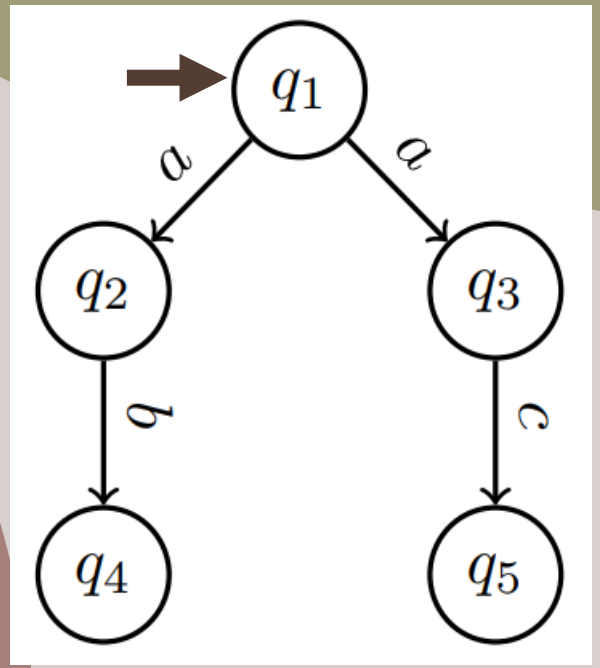
Starke Bisimilarität



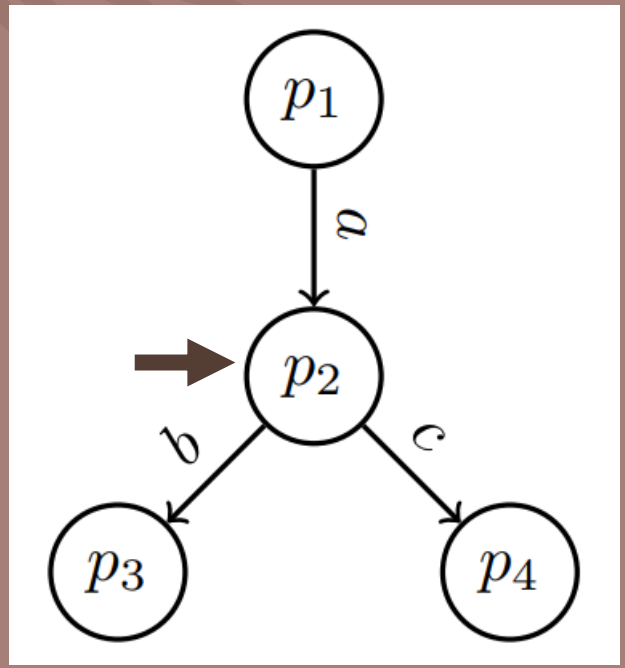


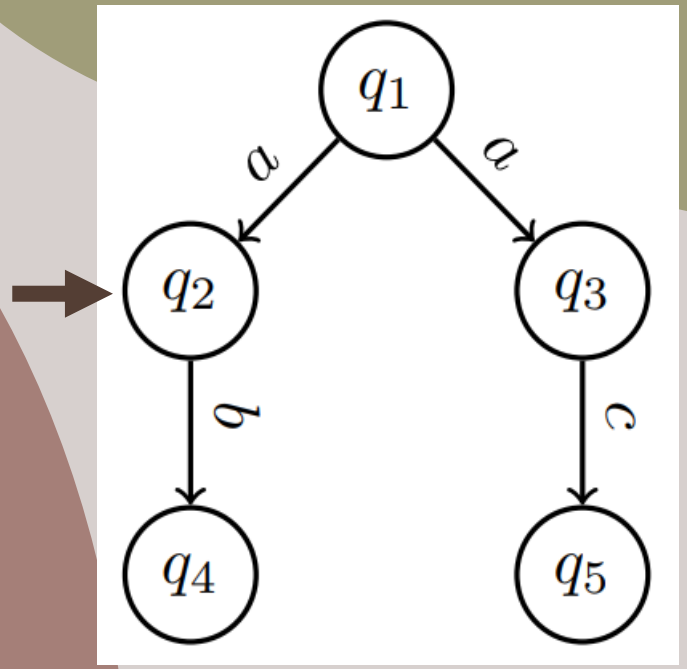
Starke Bisimilarität



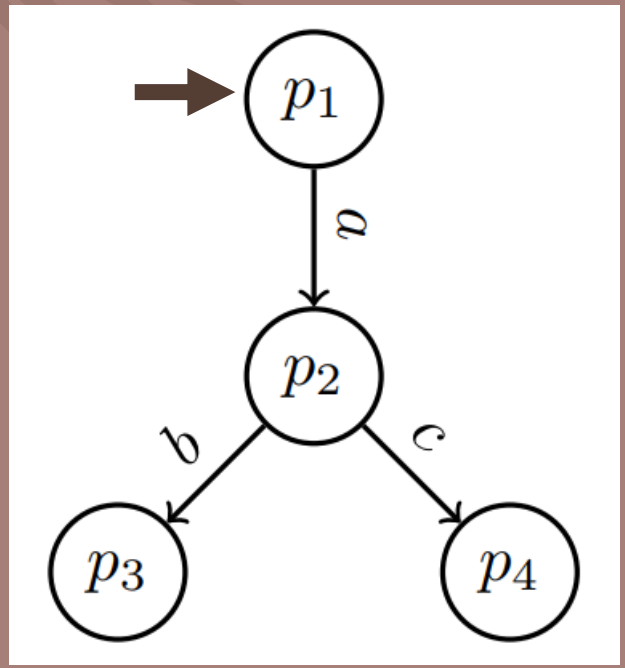


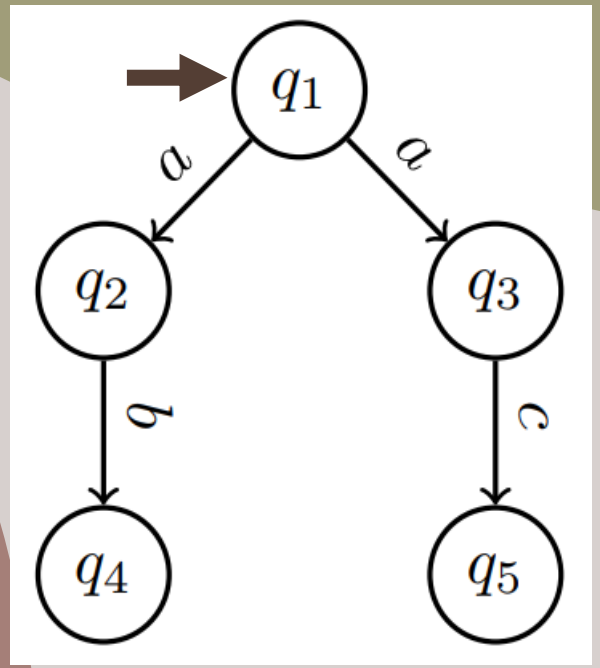
Starke Bisimilarität



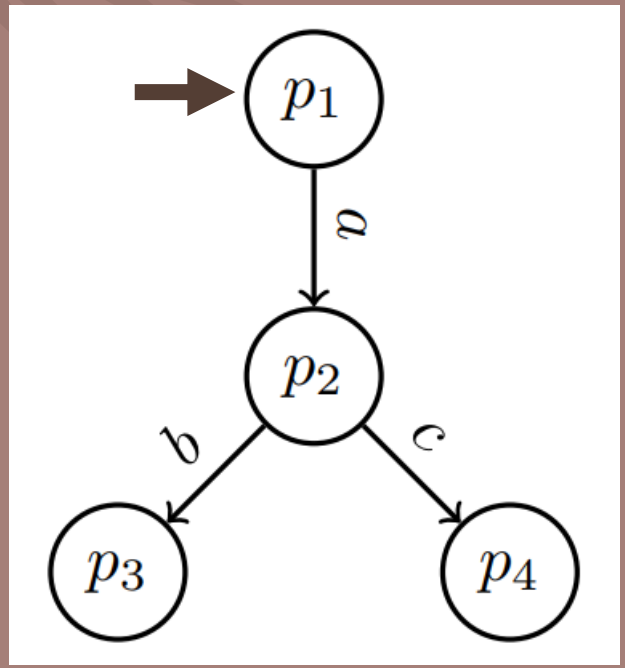


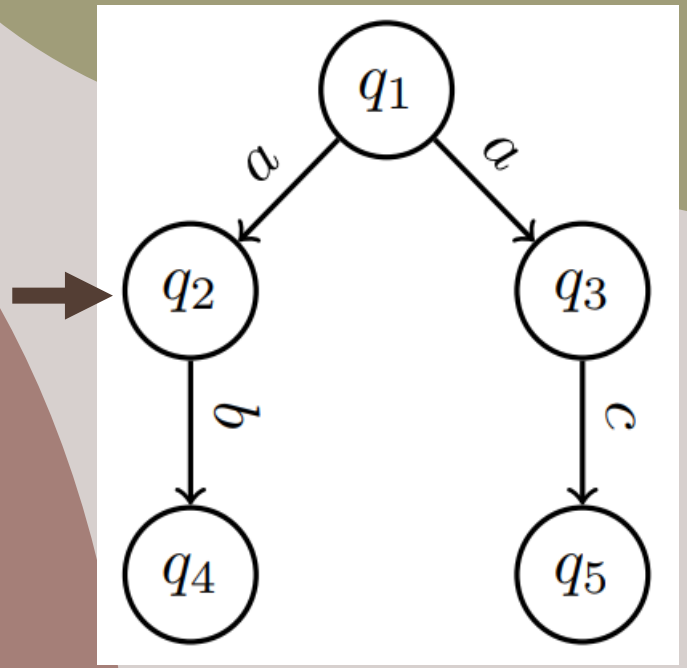
Starke Bisimilarität



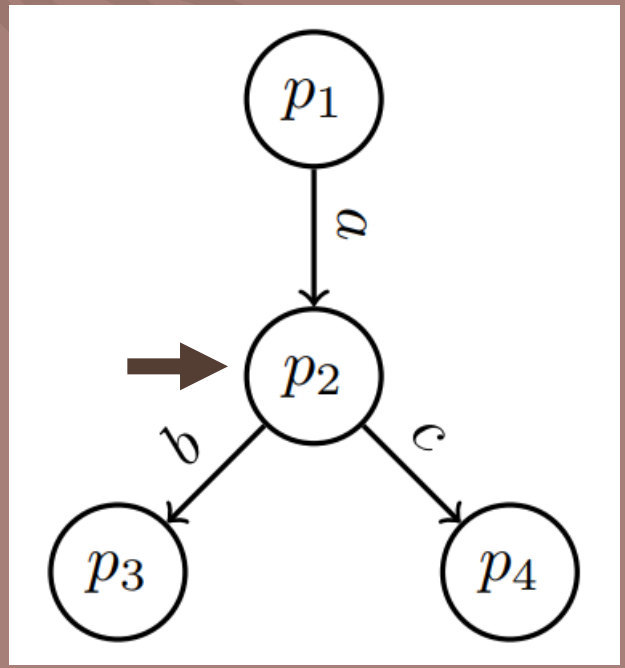


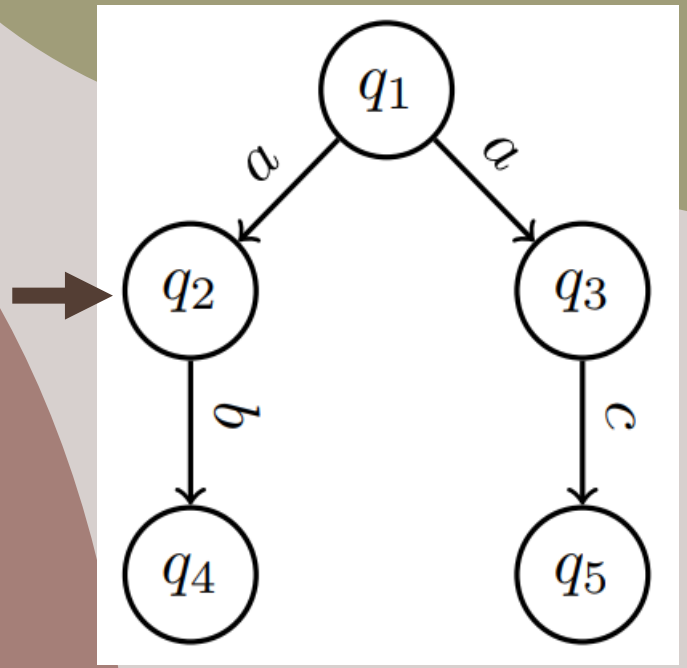
Starke Bisimilarität



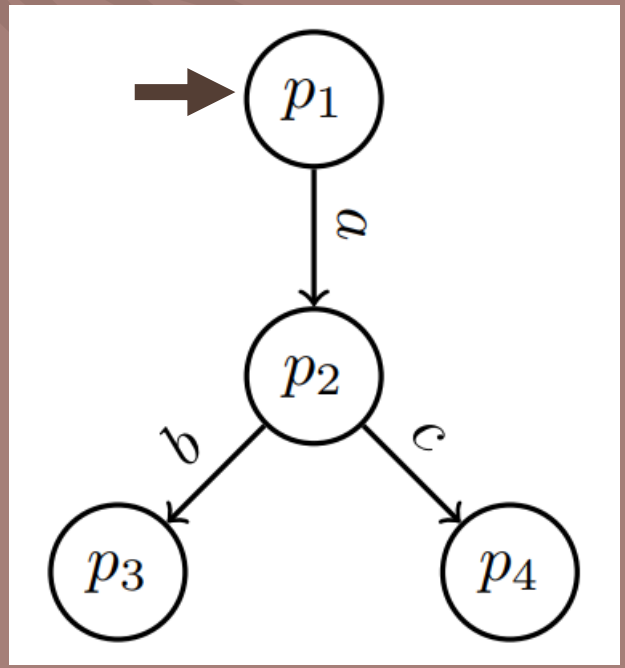


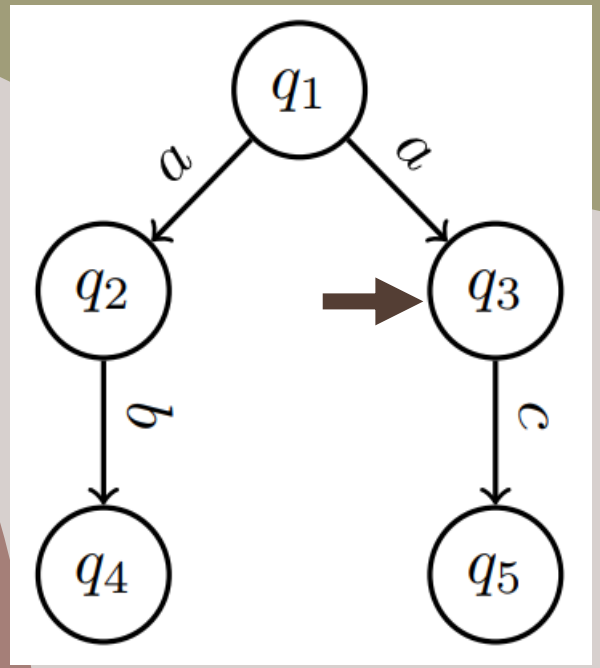
Starke Bisimilarität



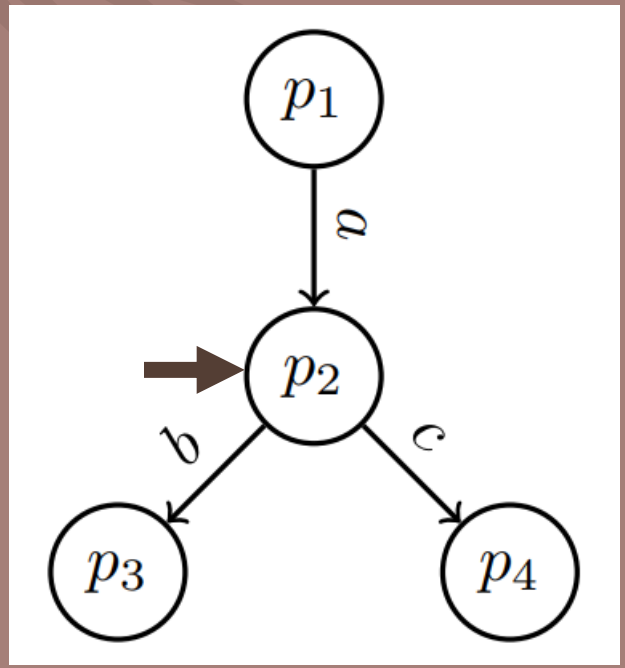


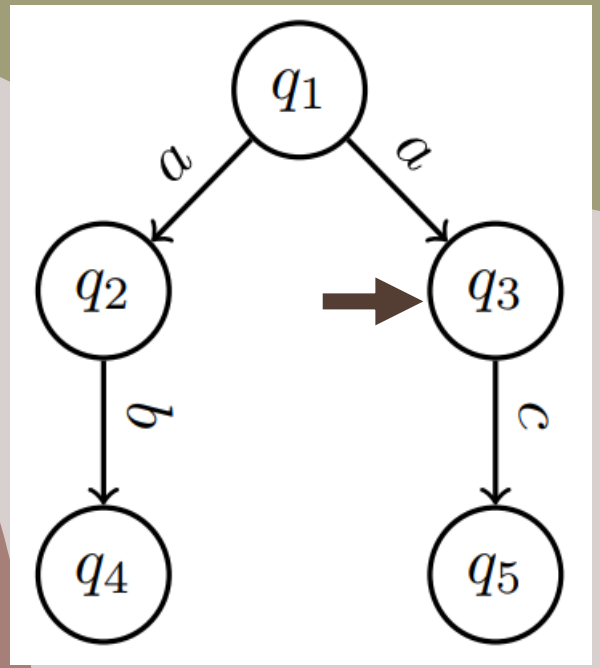
Starke Bisimilarität



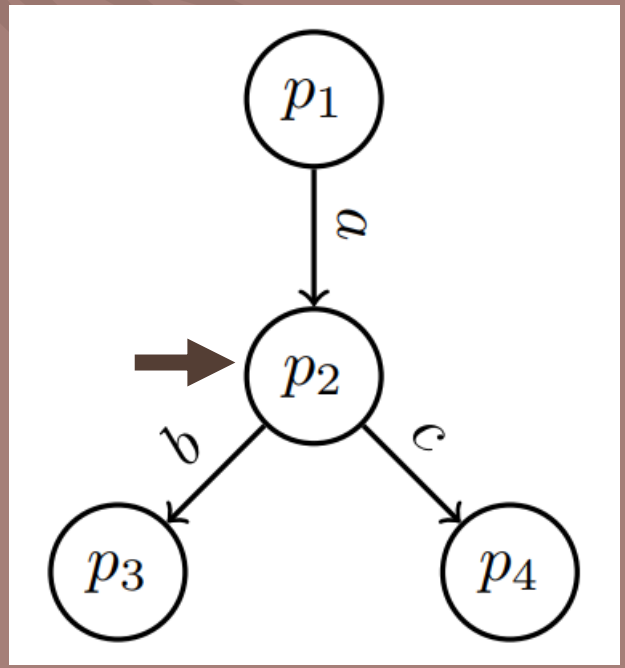


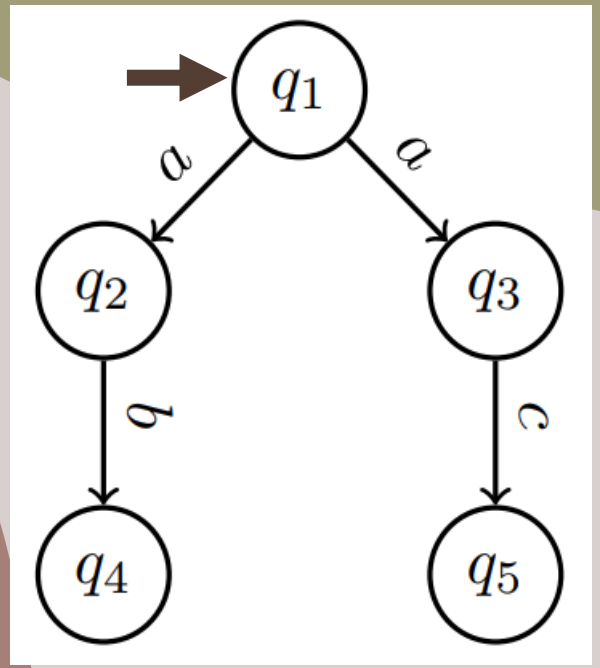
Starke Bisimilarität



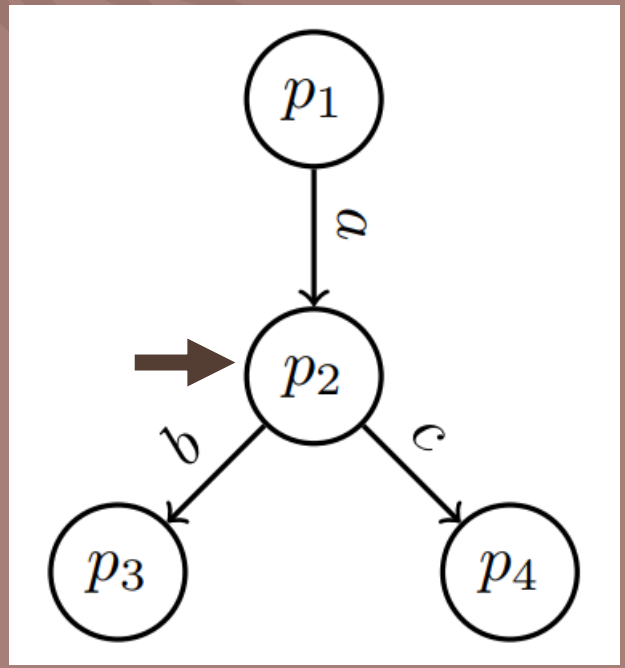


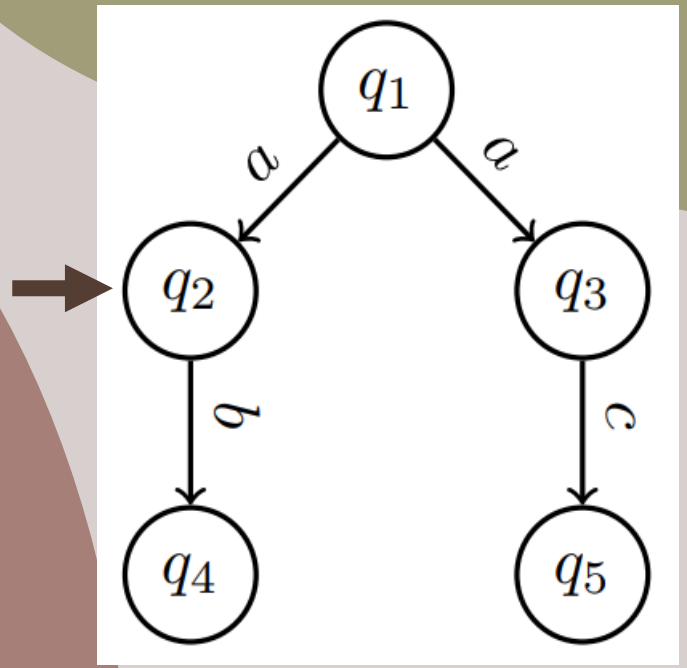
Starke Bisimilarität



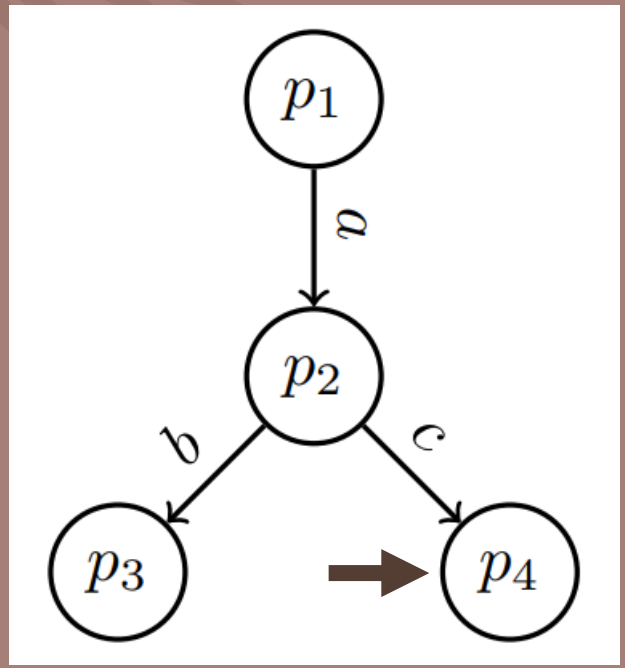


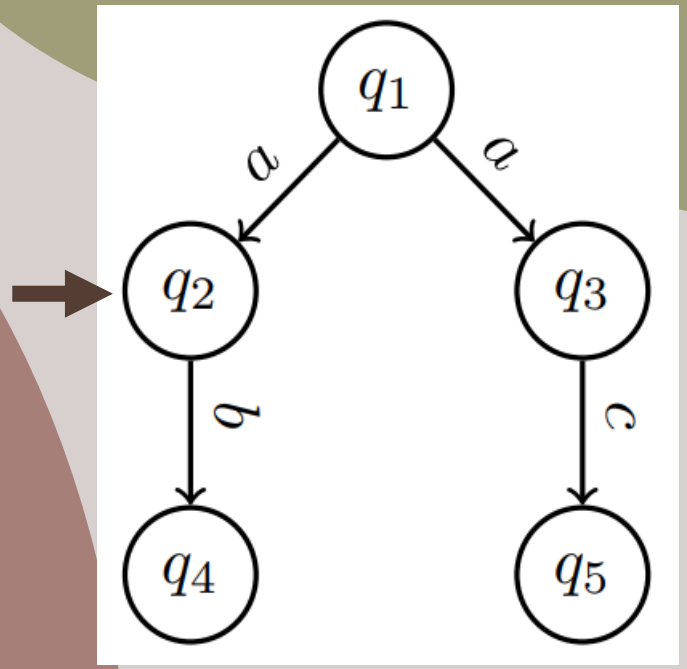
Starke Bisimilarität



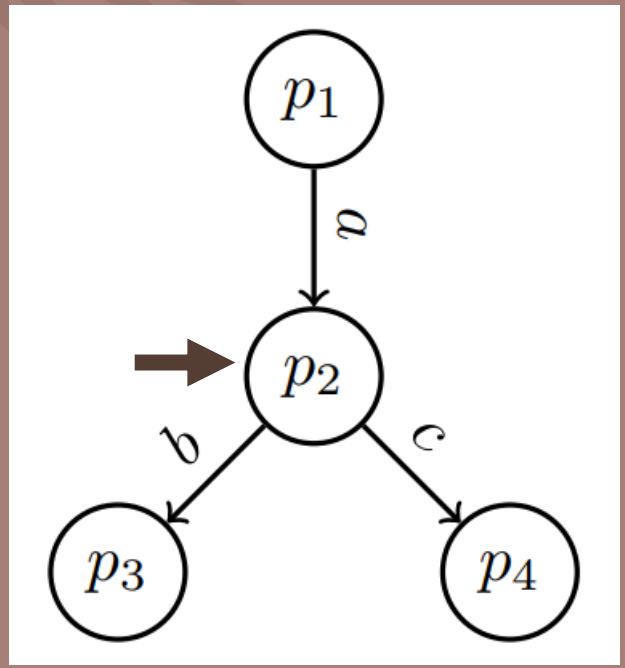


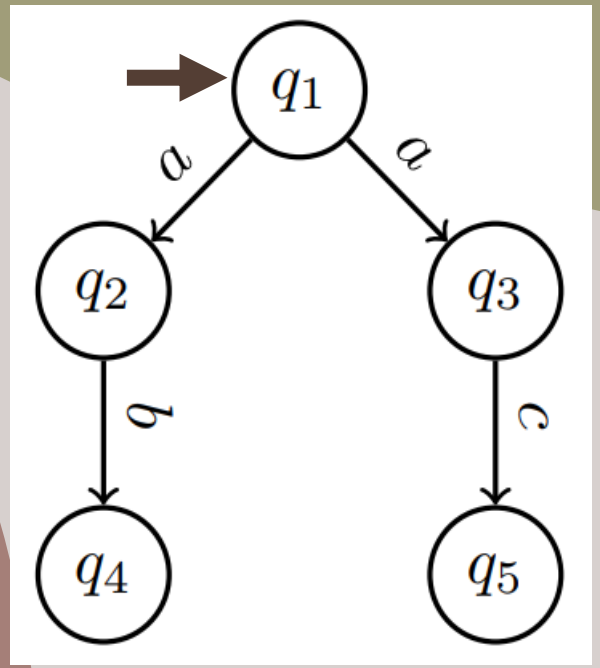
Starke Bisimilarität



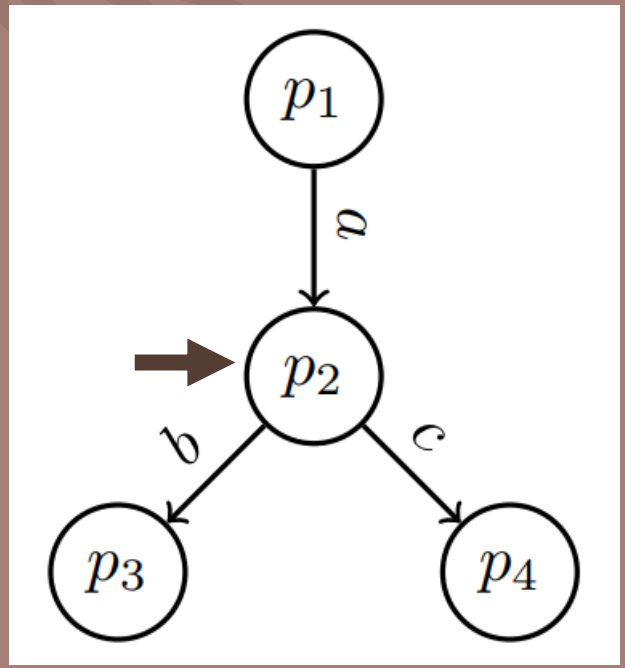


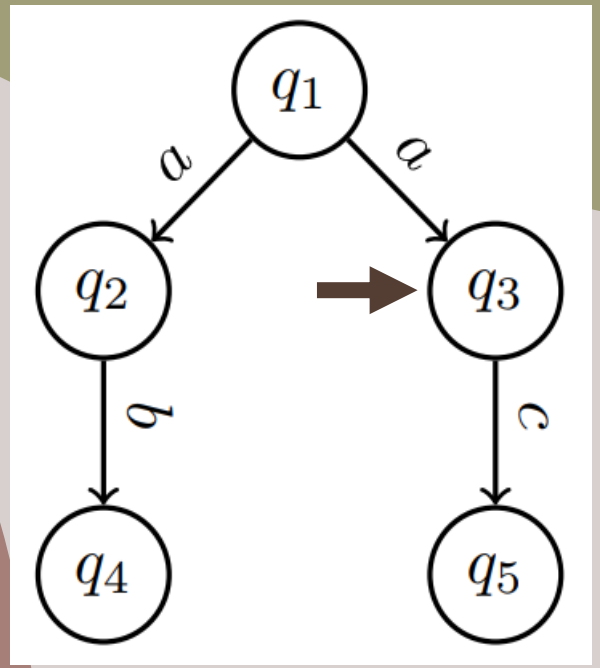
Starke Bisimilarität



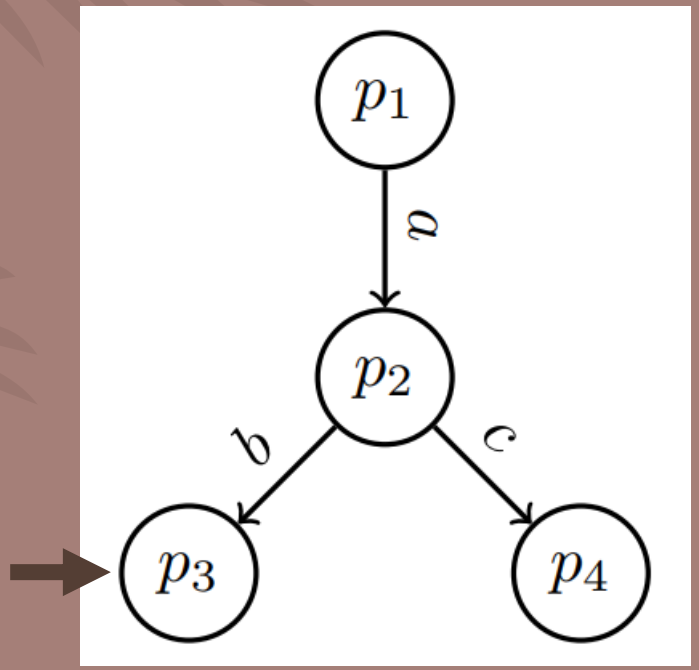


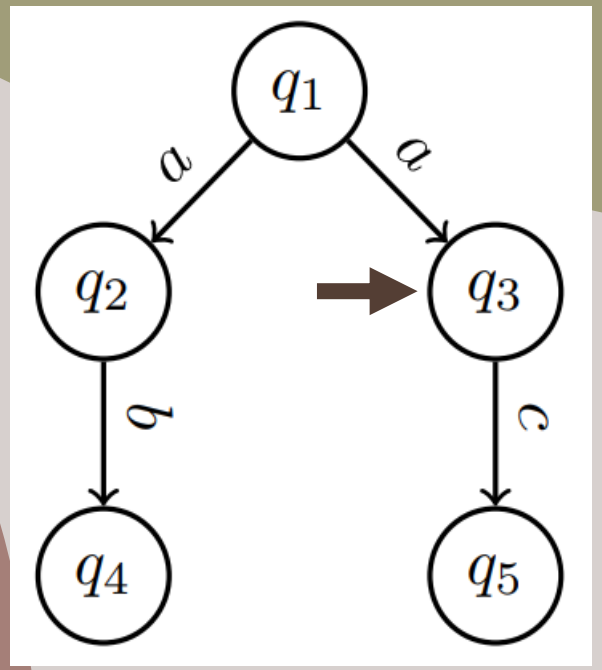
Starke Bisimilarität





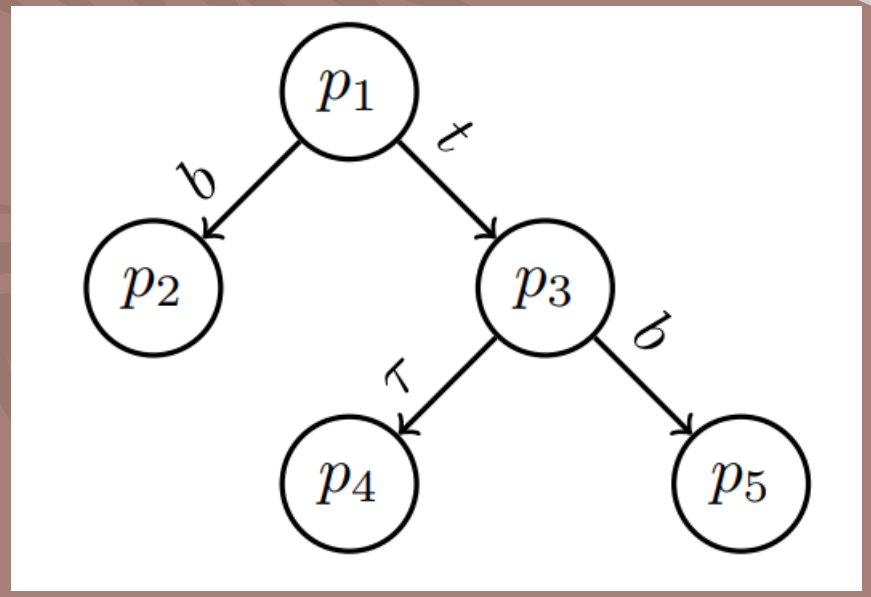
Starke Bisimilarität

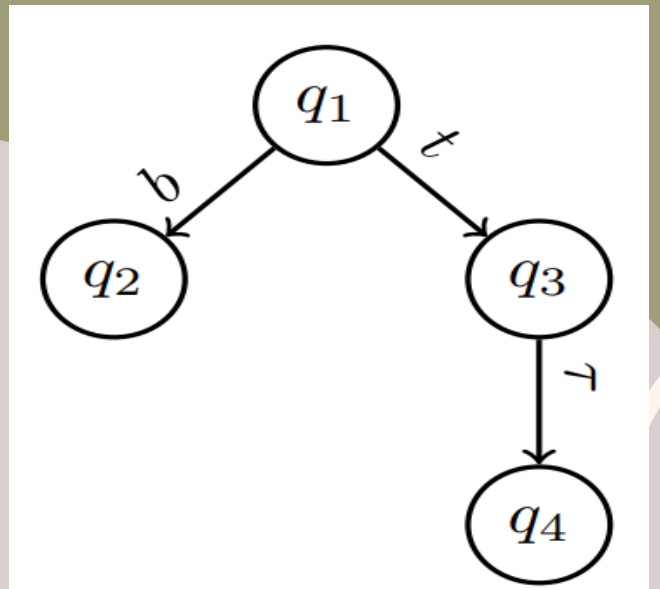




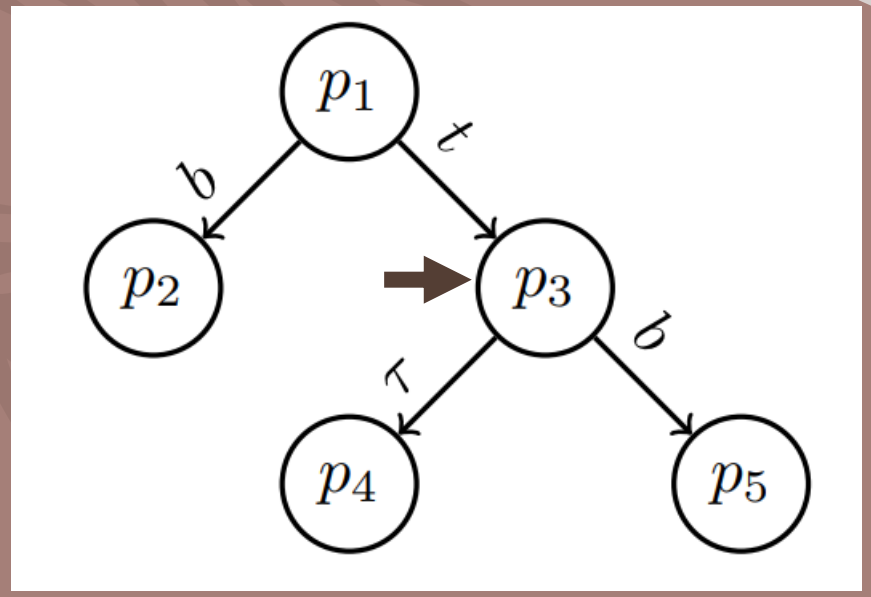
Reaktive Bisimilarität

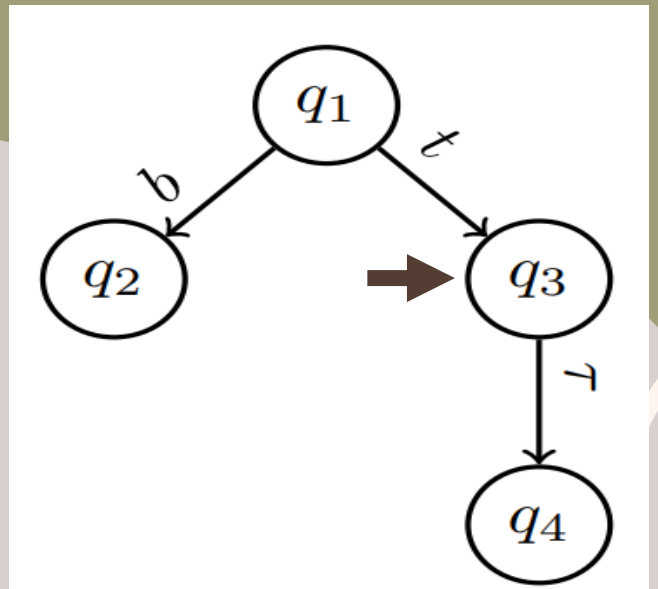
Reaktive Bisimilarität



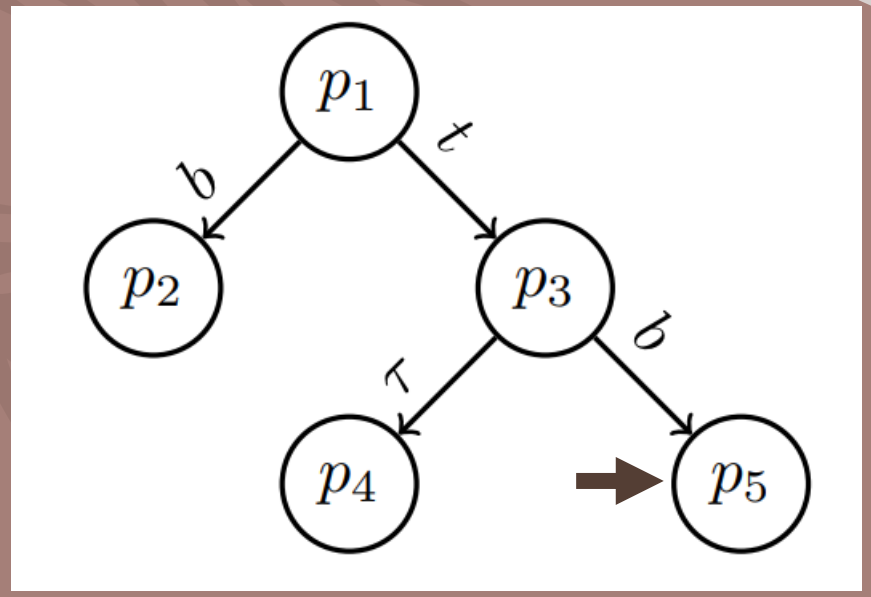


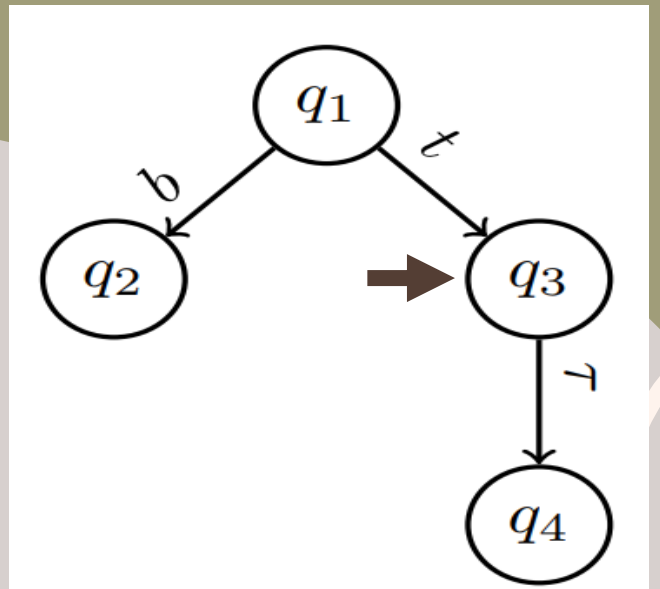
Reaktive Bisimilarität



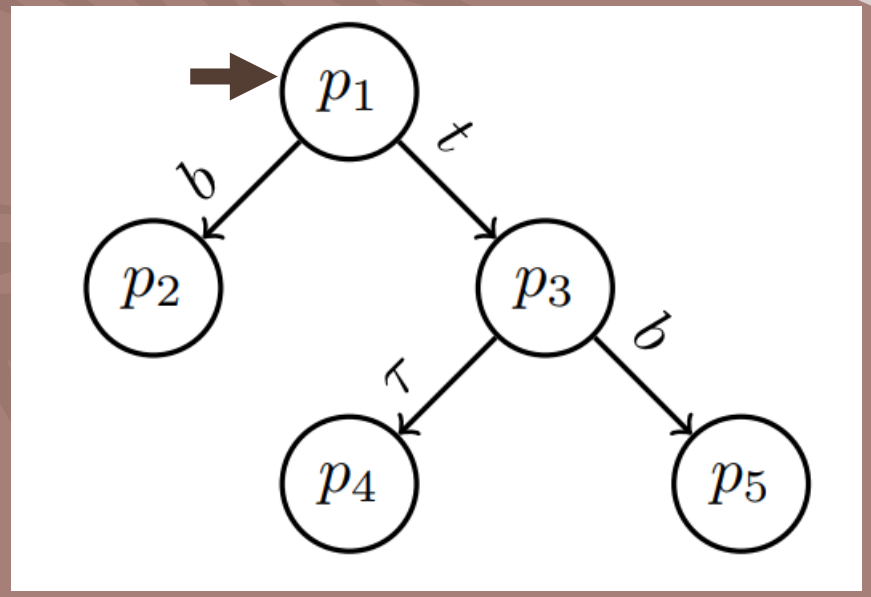


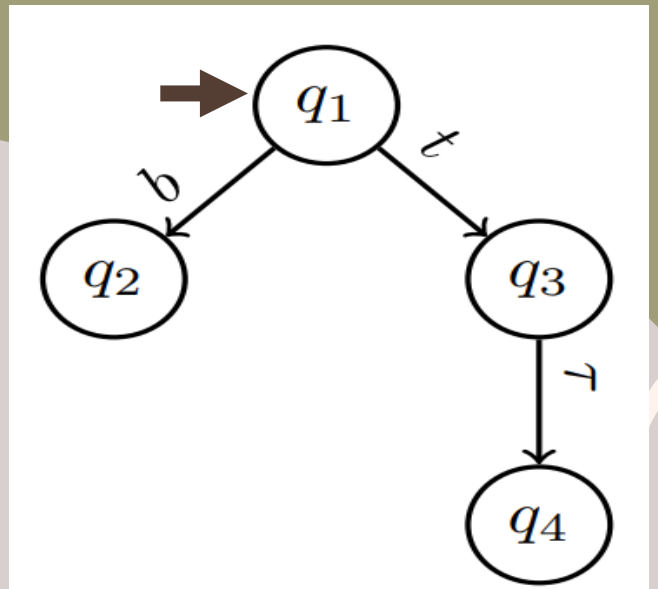
Reaktive Bisimilarität



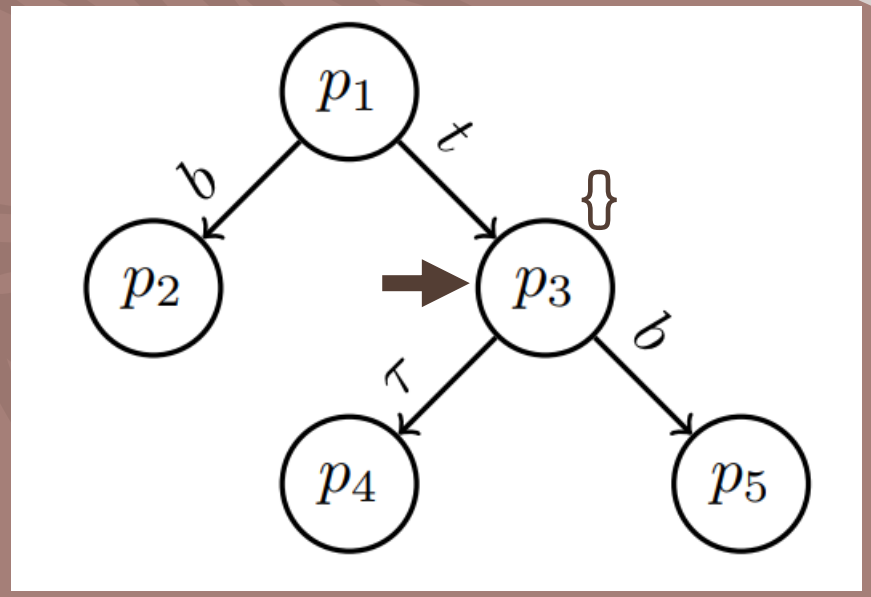


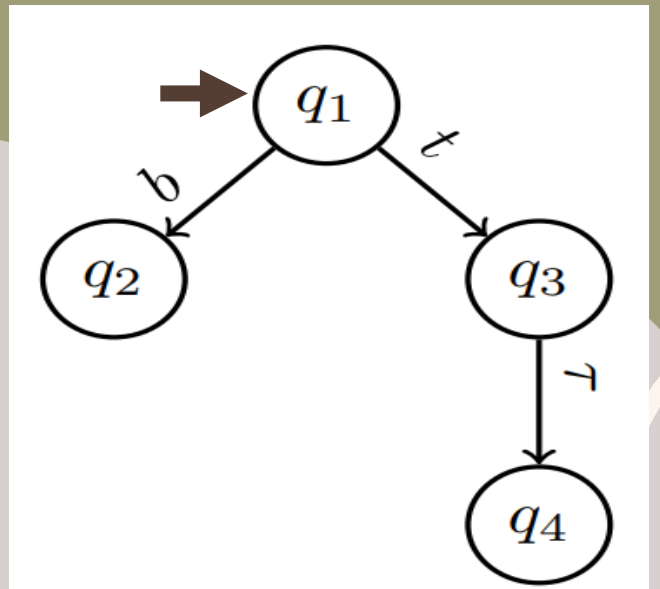
Reaktive Bisimilarität



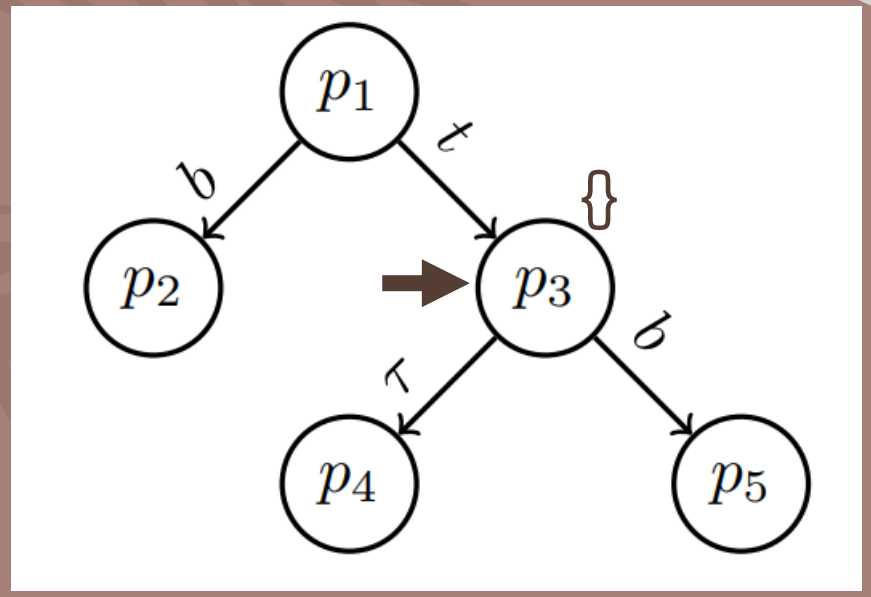


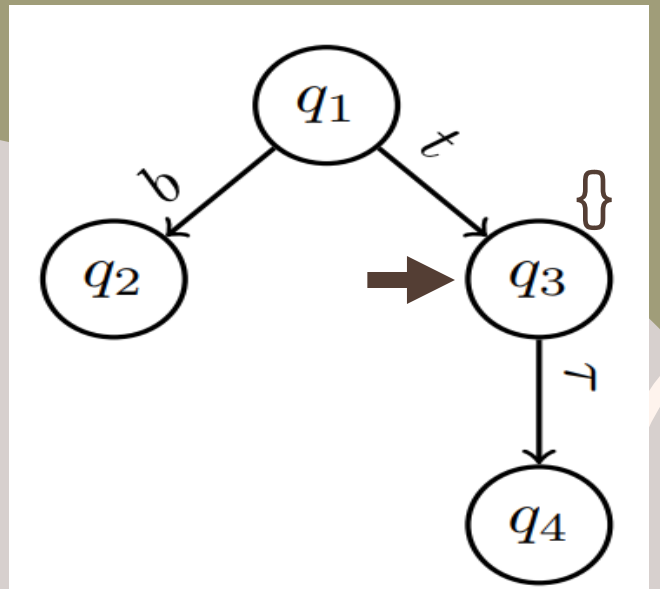
Reaktive Bisimilarität



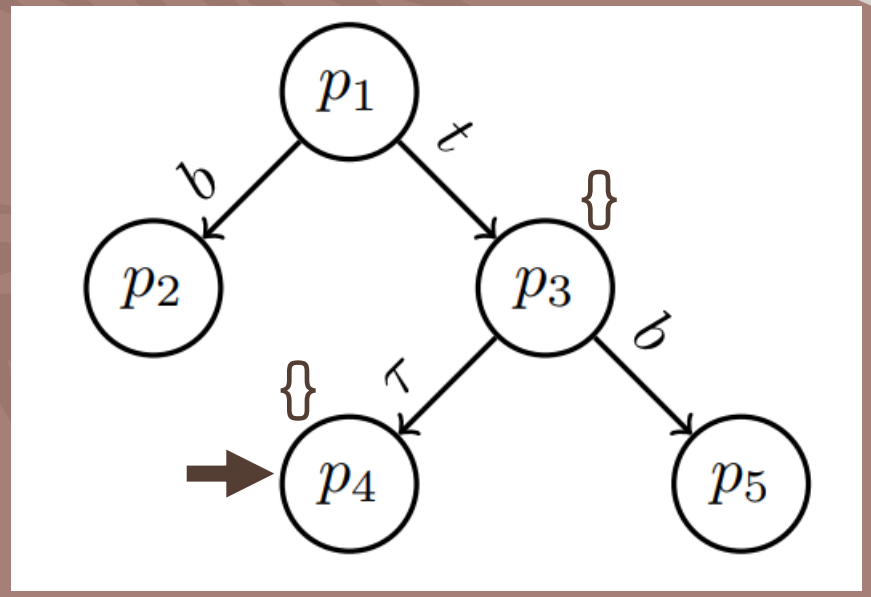


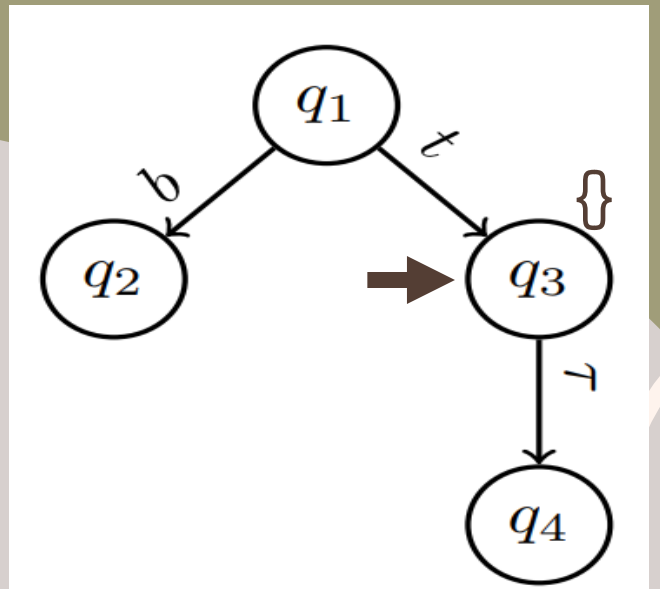
Reaktive Bisimilarität



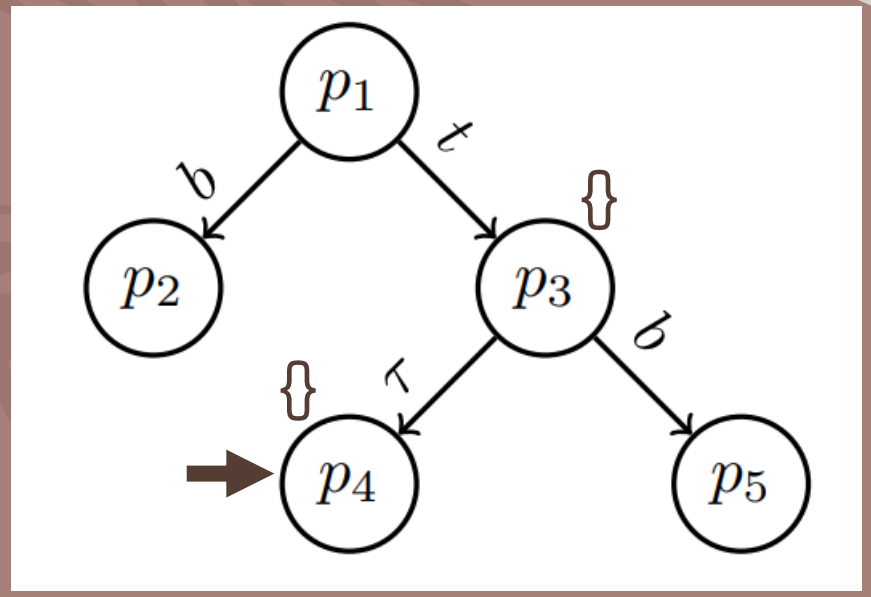


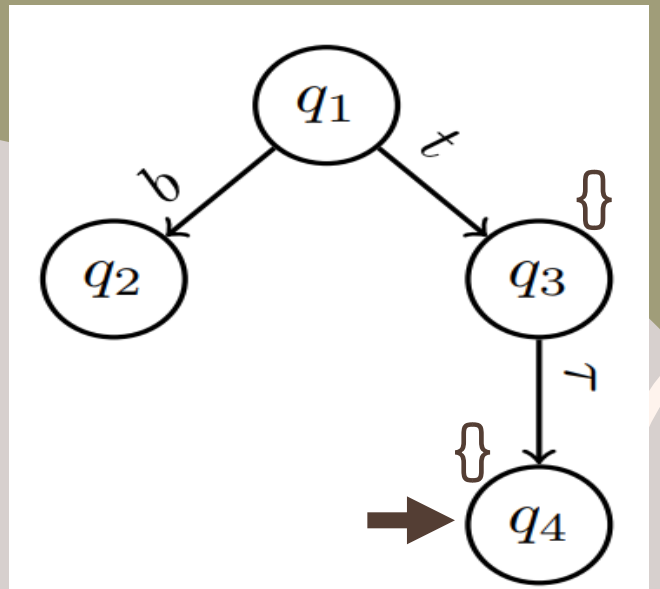
Reaktive Bisimilarität





Reaktive Bisimilarität





Die Reduktion von Max

Die Reduktion von Max

Reaktive Bisimilarität

Die Reduktion von Max

Reaktive Bisimilarität

Die Reduktion von Max

Die Reduktion von Max

Funktionsweise

Die Reduktion von Max

Funktionsweise

$$1)\ \frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_{\upsilon} \upsilon(p')} \qquad 2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')}\ a \in X \qquad 4)\ \frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_{\upsilon} \upsilon_X(p')}$$

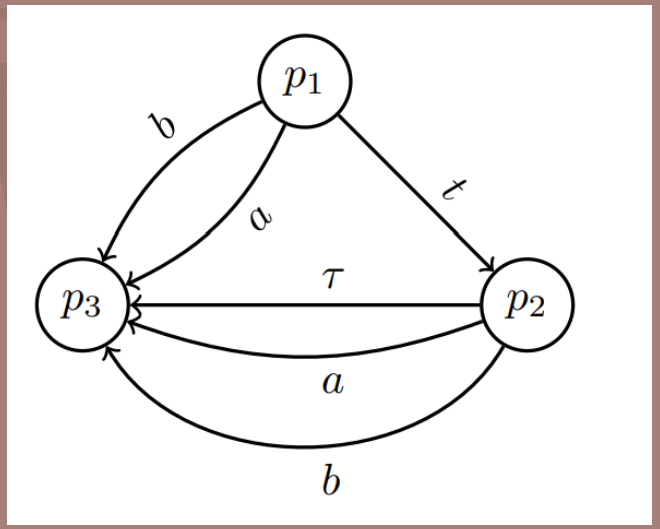
$$5)\ \frac{p \overset{\alpha}{\not\rightarrow} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_{\upsilon} \upsilon(p)} \qquad 6)\ \frac{p \overset{\alpha}{\not\rightarrow} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_{\upsilon} \upsilon_X(p')}$$

Die Reduktion von Max

Beispiel

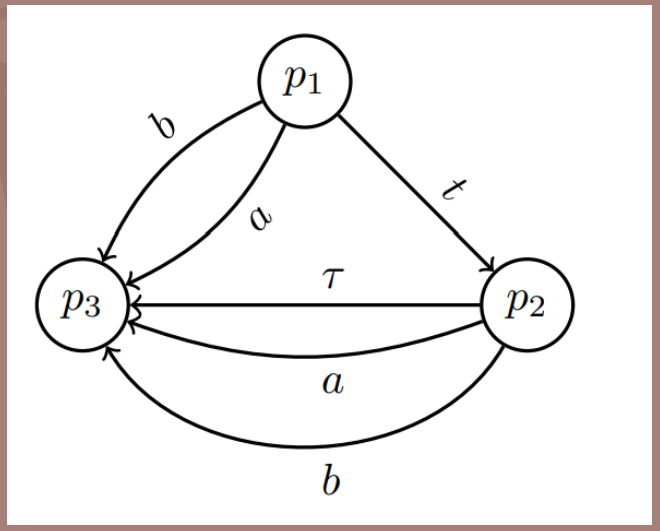
Die Reduktion von Max

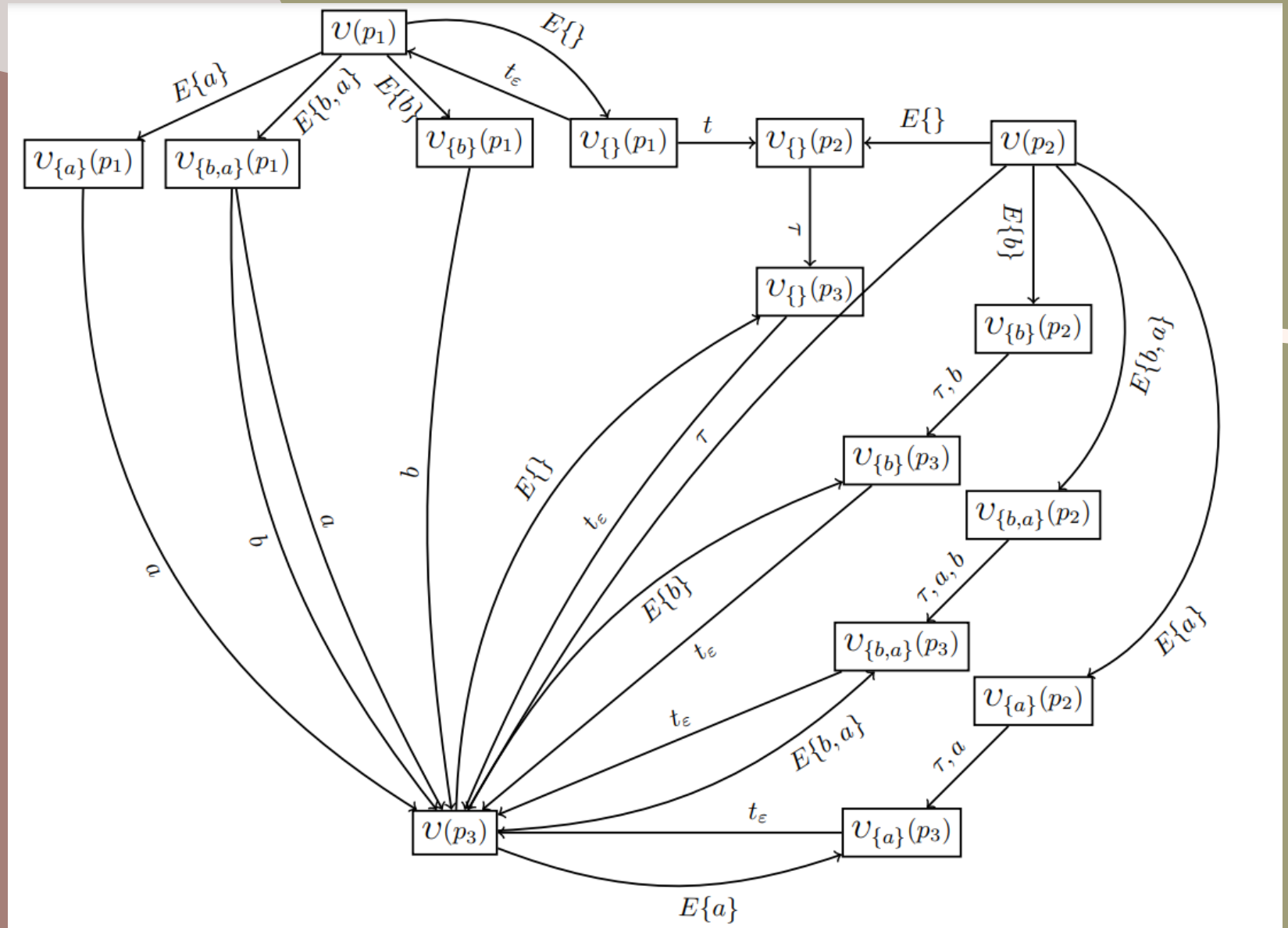
Beispiel



Die Reduktion von Max

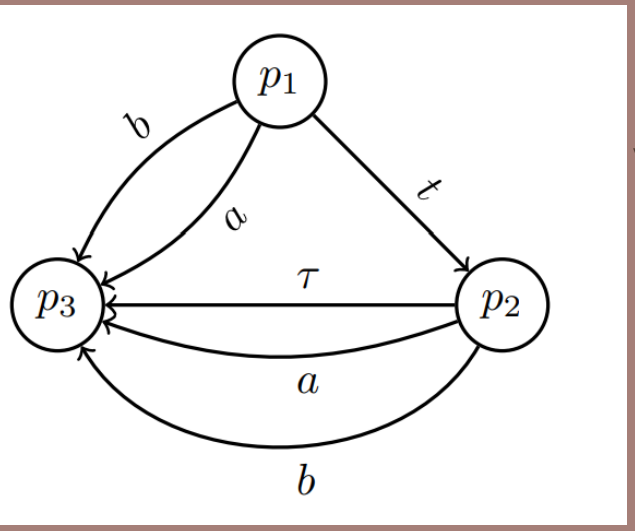
Beispiel



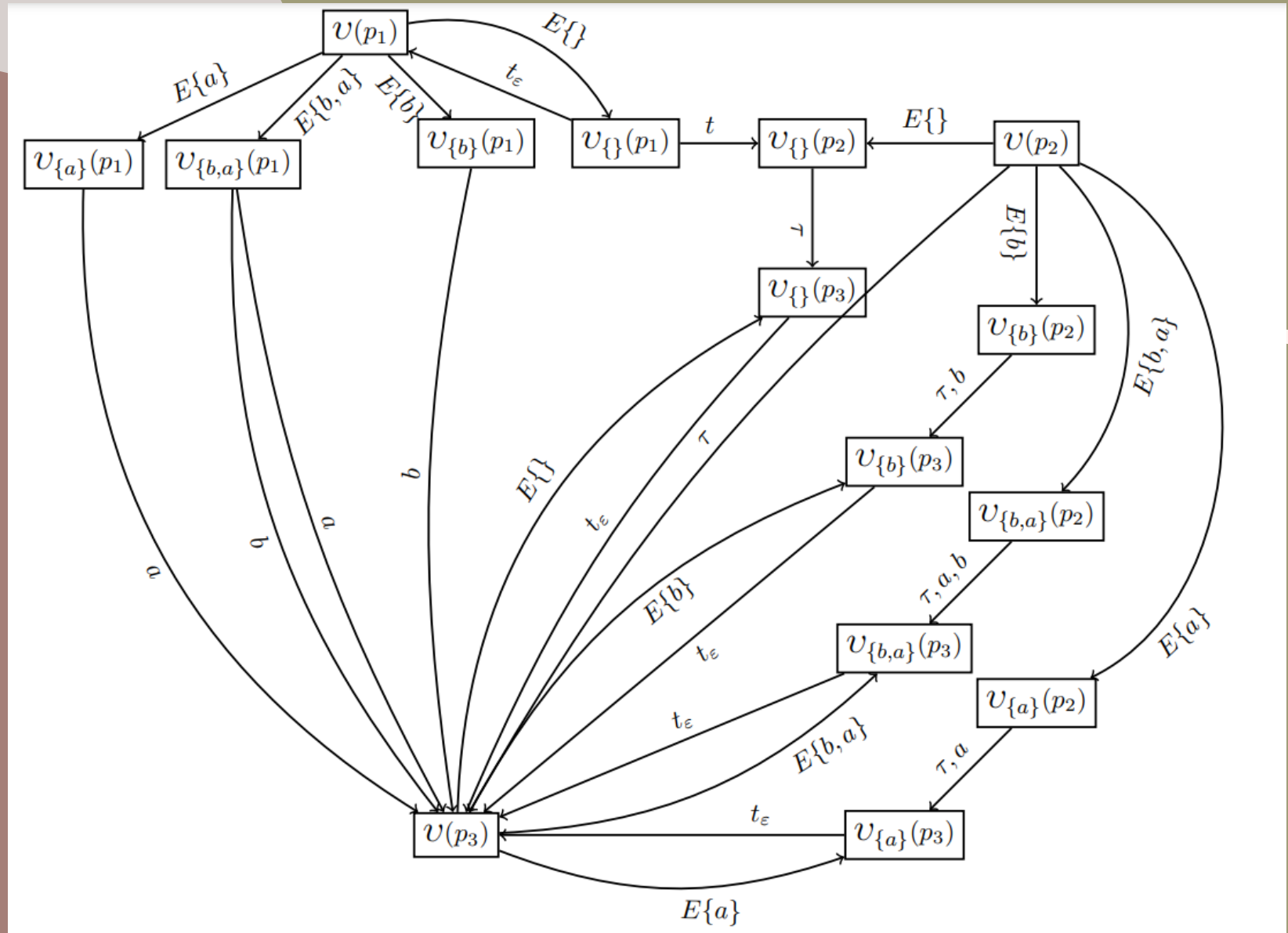


Die Reduktion von Max

Beispiel



WIE?



ARBEITSGEGENSTÄNDE

Kernidee
Ursprünglicher Algorithmus
Verbesserte Algorithmus
Einzige Umgebungsaktion
Überprüfung eines oder aller Paare

Kernidee

Kernidee

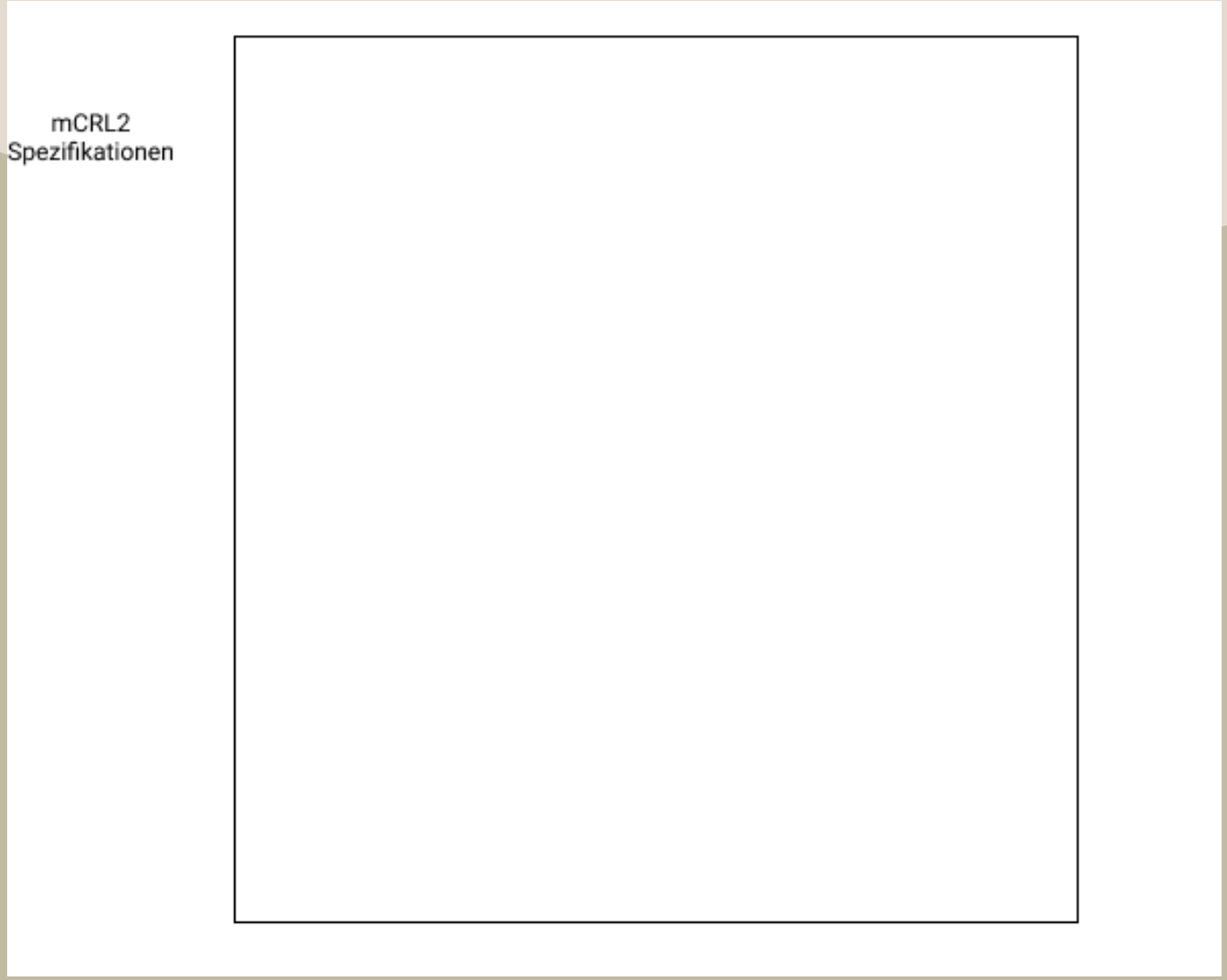
Kernidee

```
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
    );
```

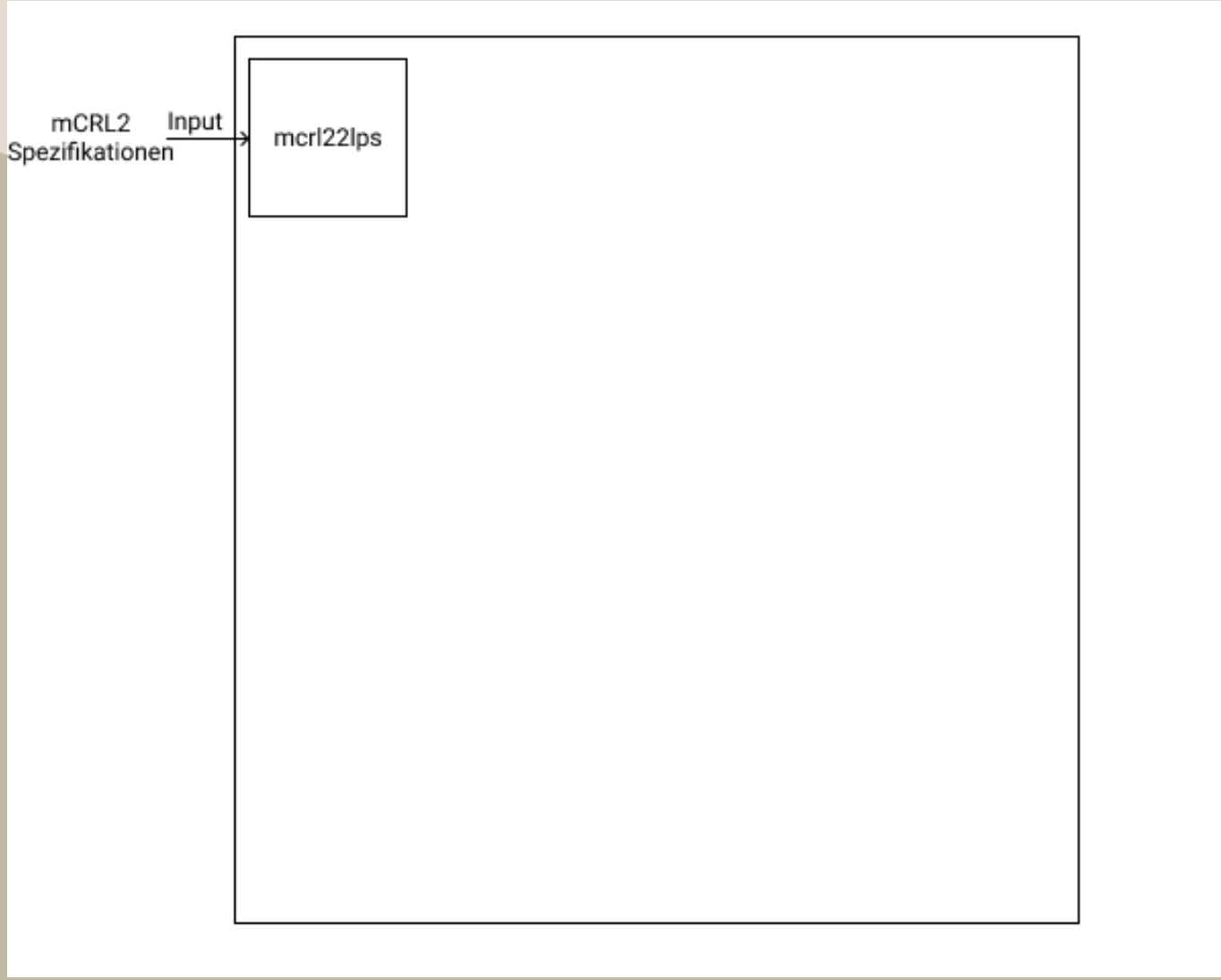
mCRL2
Spezifikationen

Kernidee



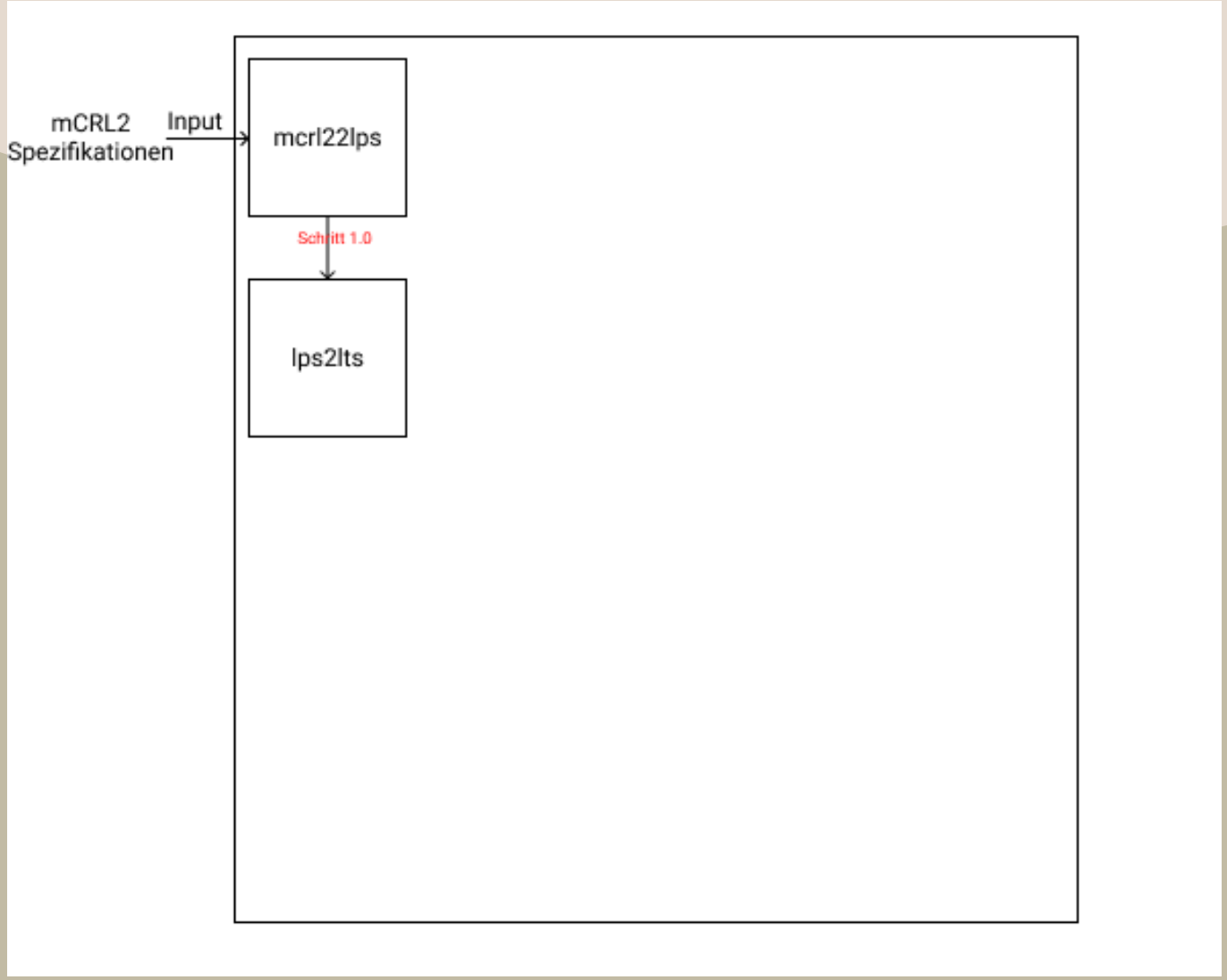
Kernidee

Lps Foto

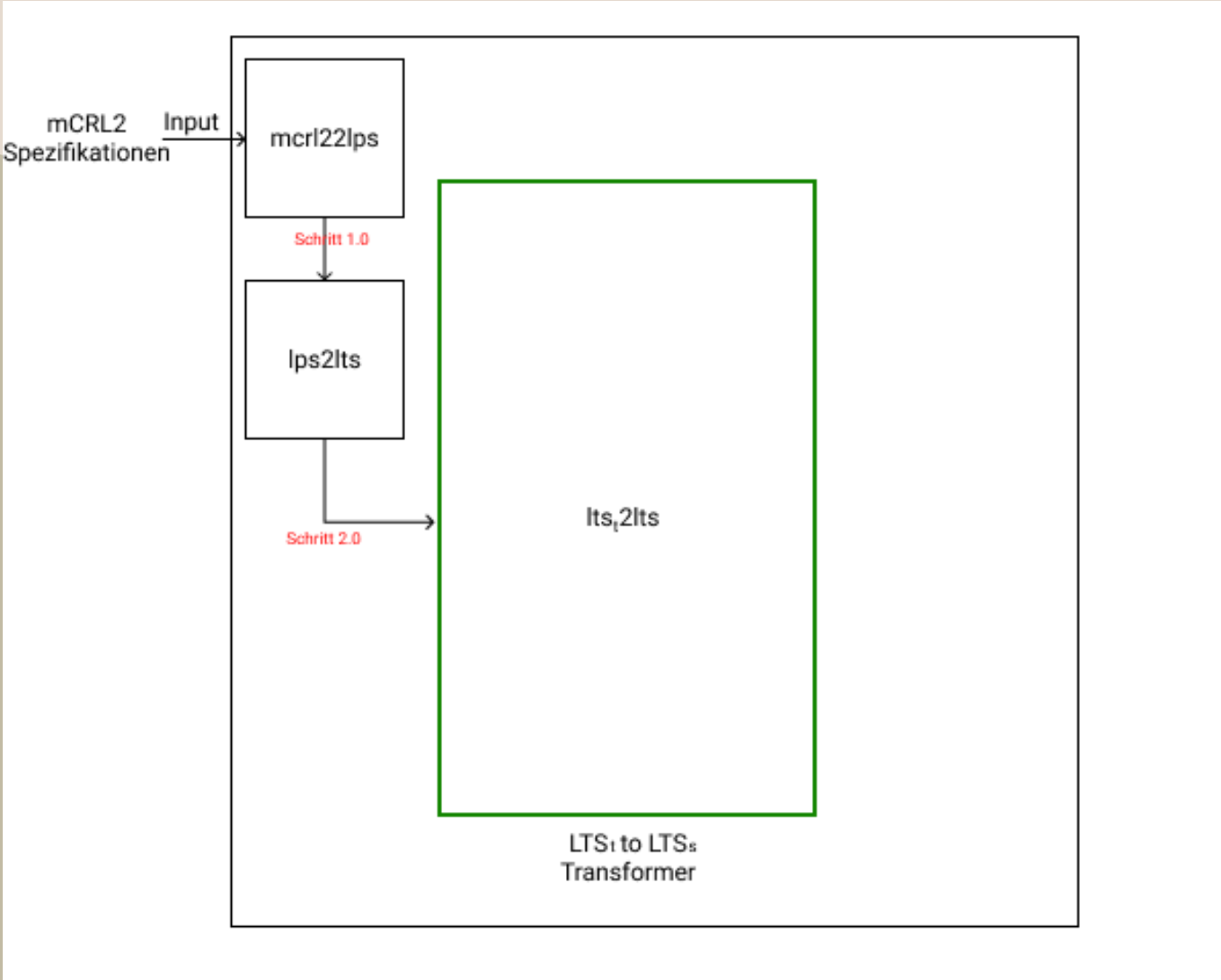


Kernidee

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

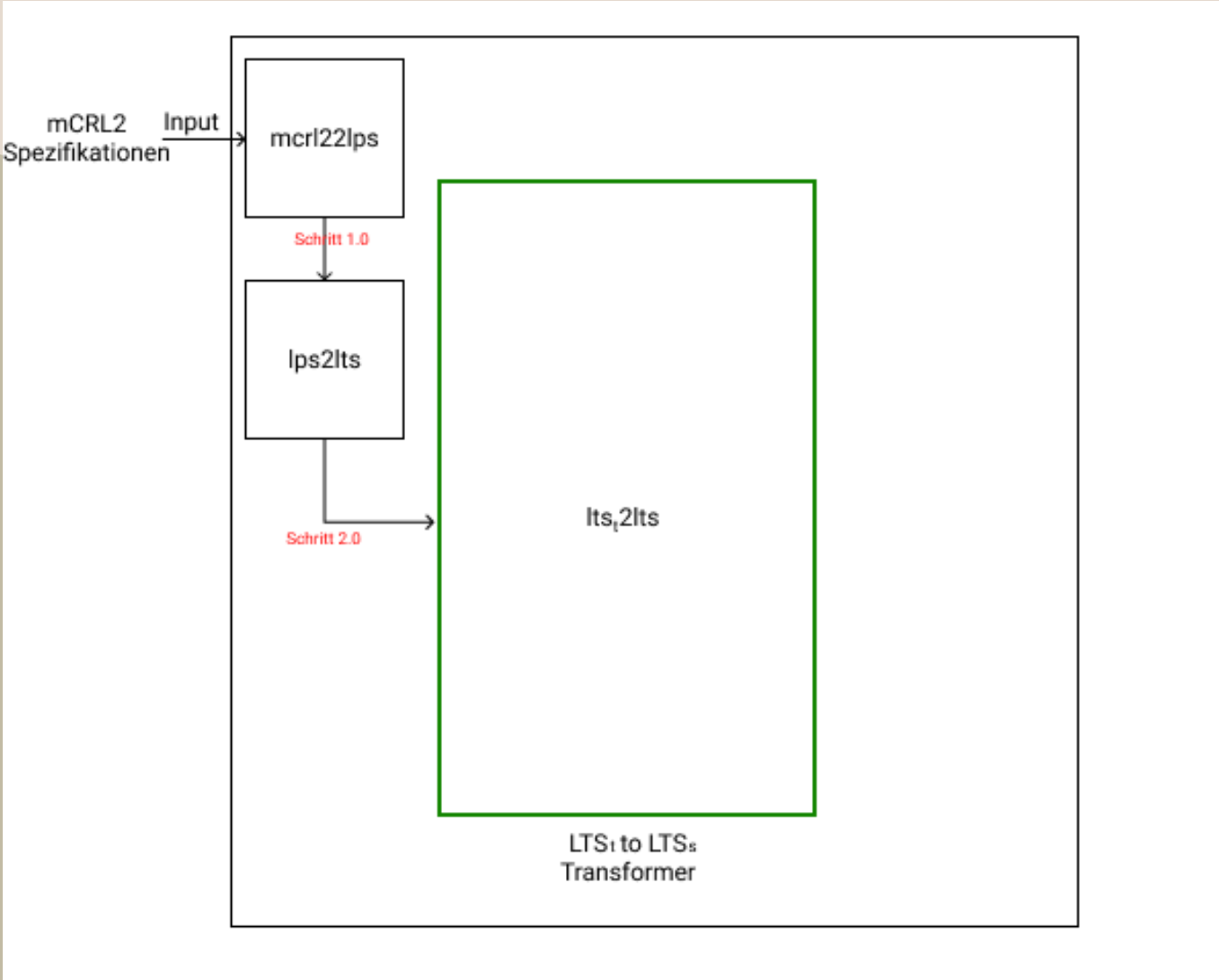


Kernidee

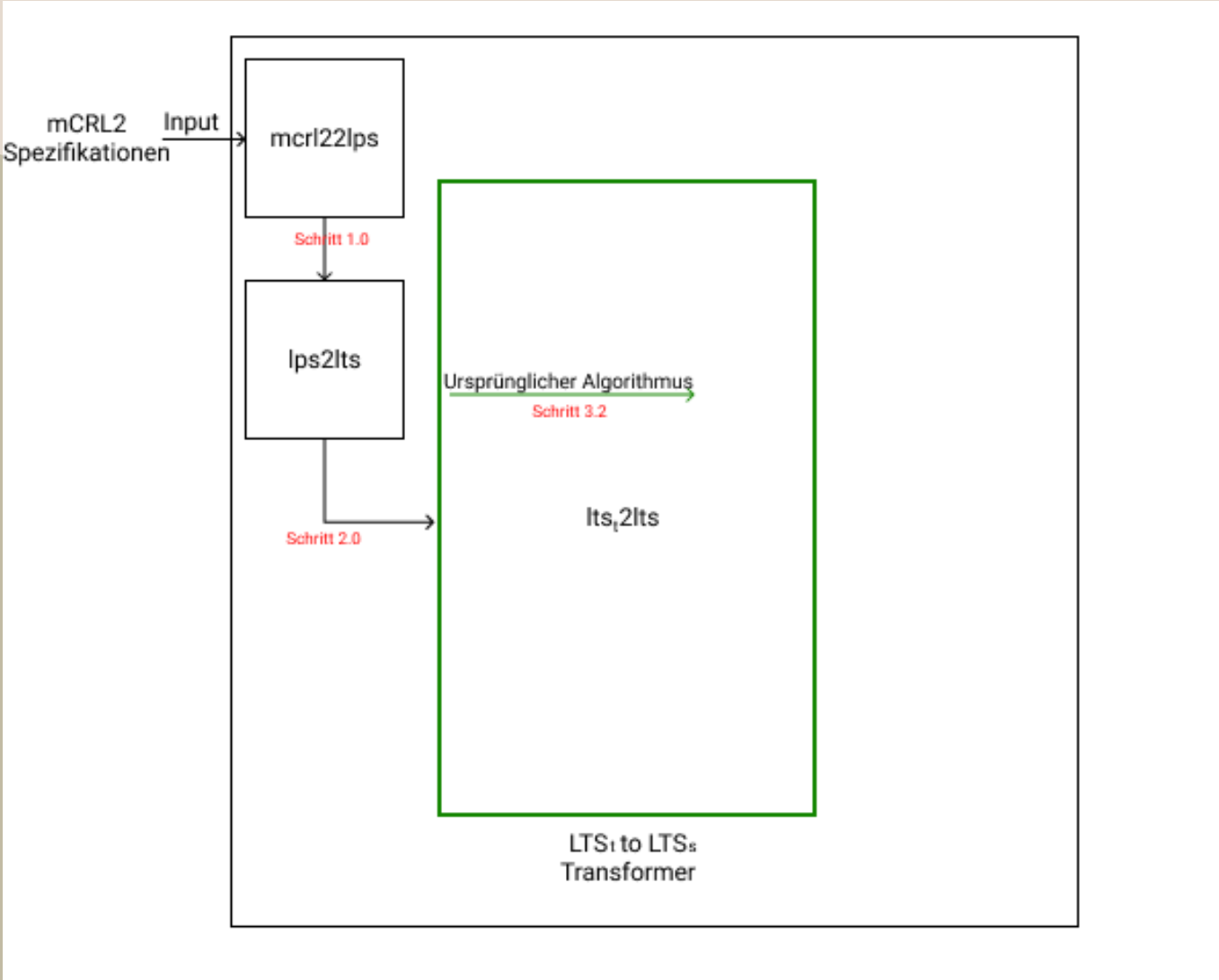

mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer

Kernidee

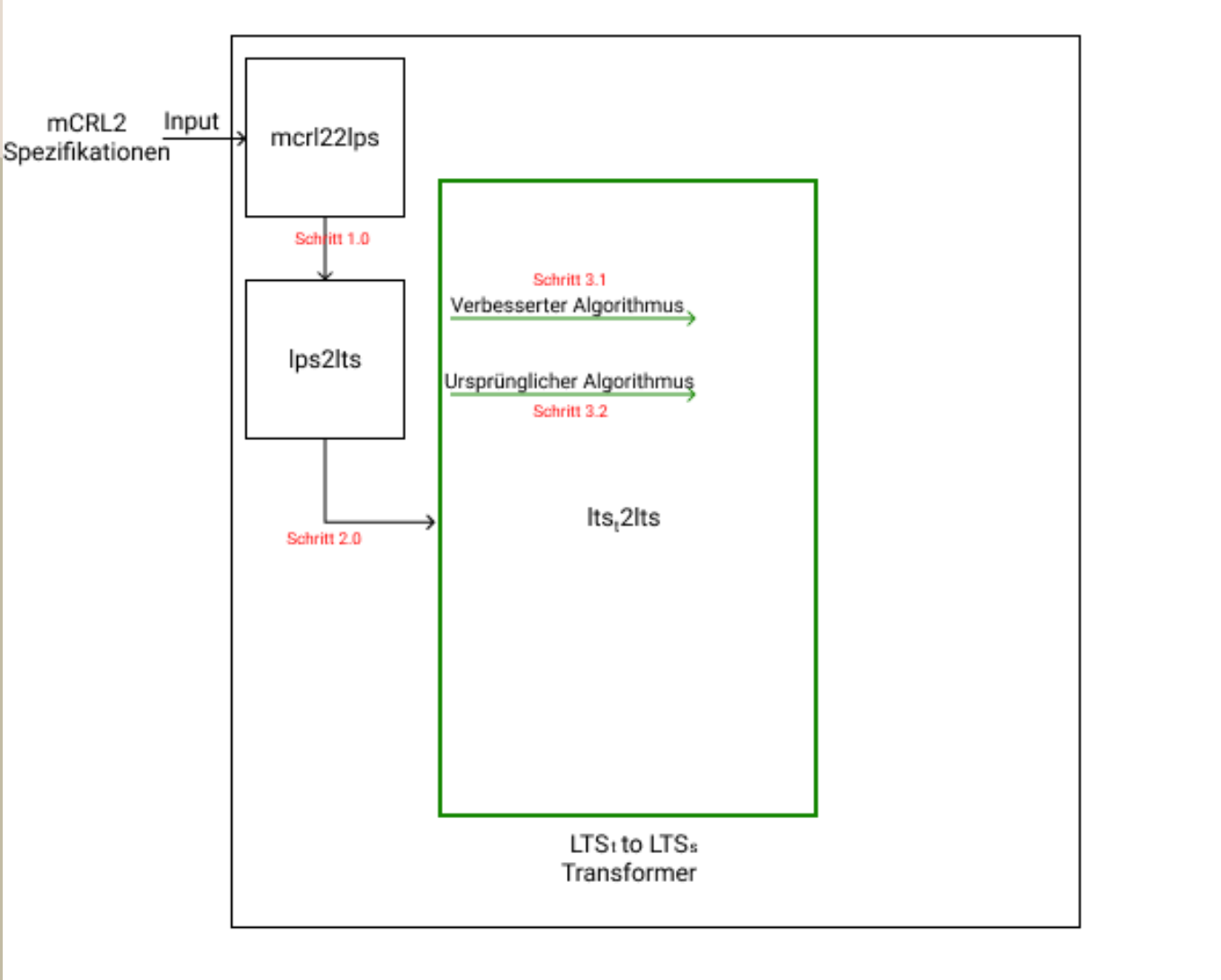
LTS_S Foto



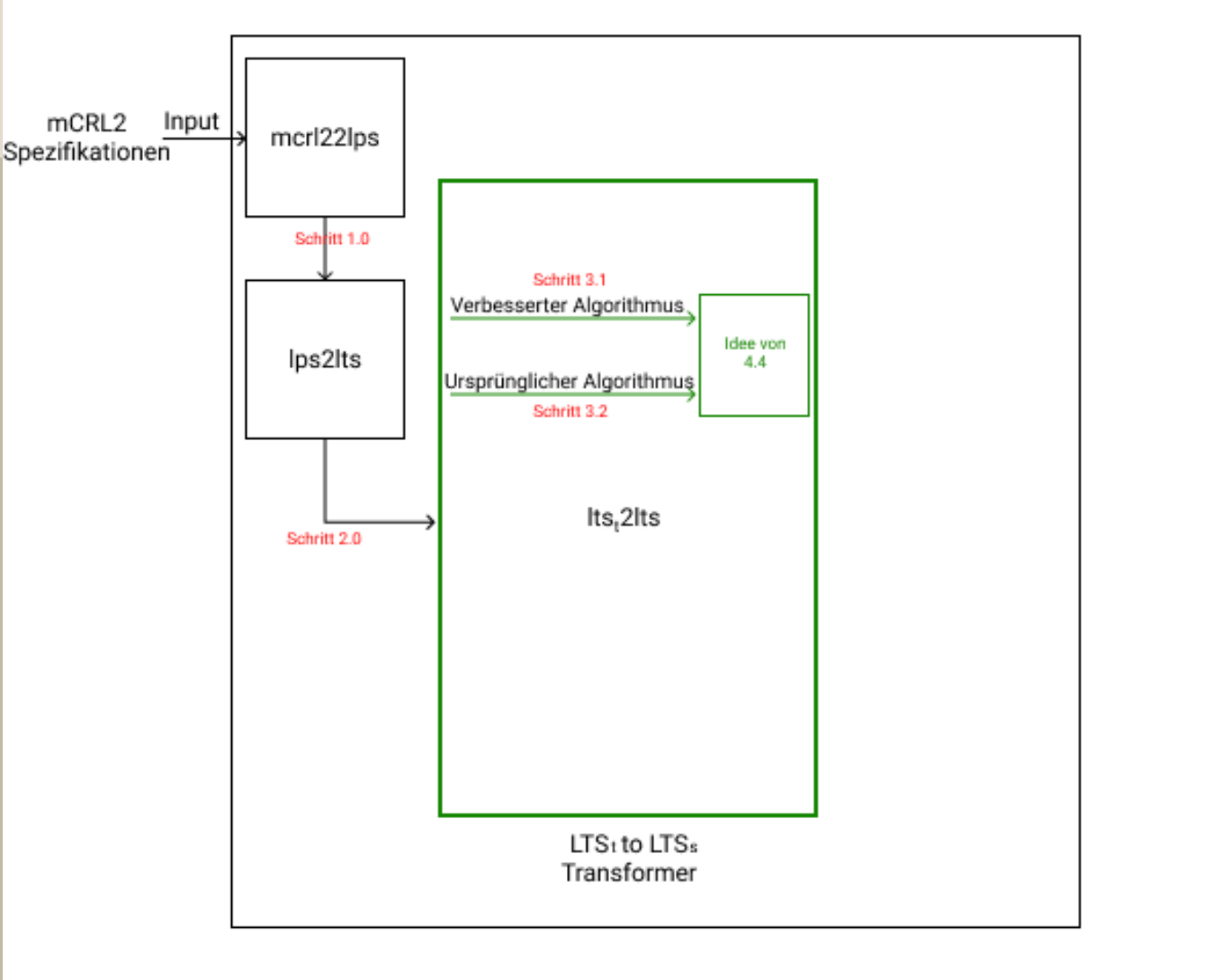
Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Ursprünglicher Algorithmus
Schritt 3.2
Schritt 2.0
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer

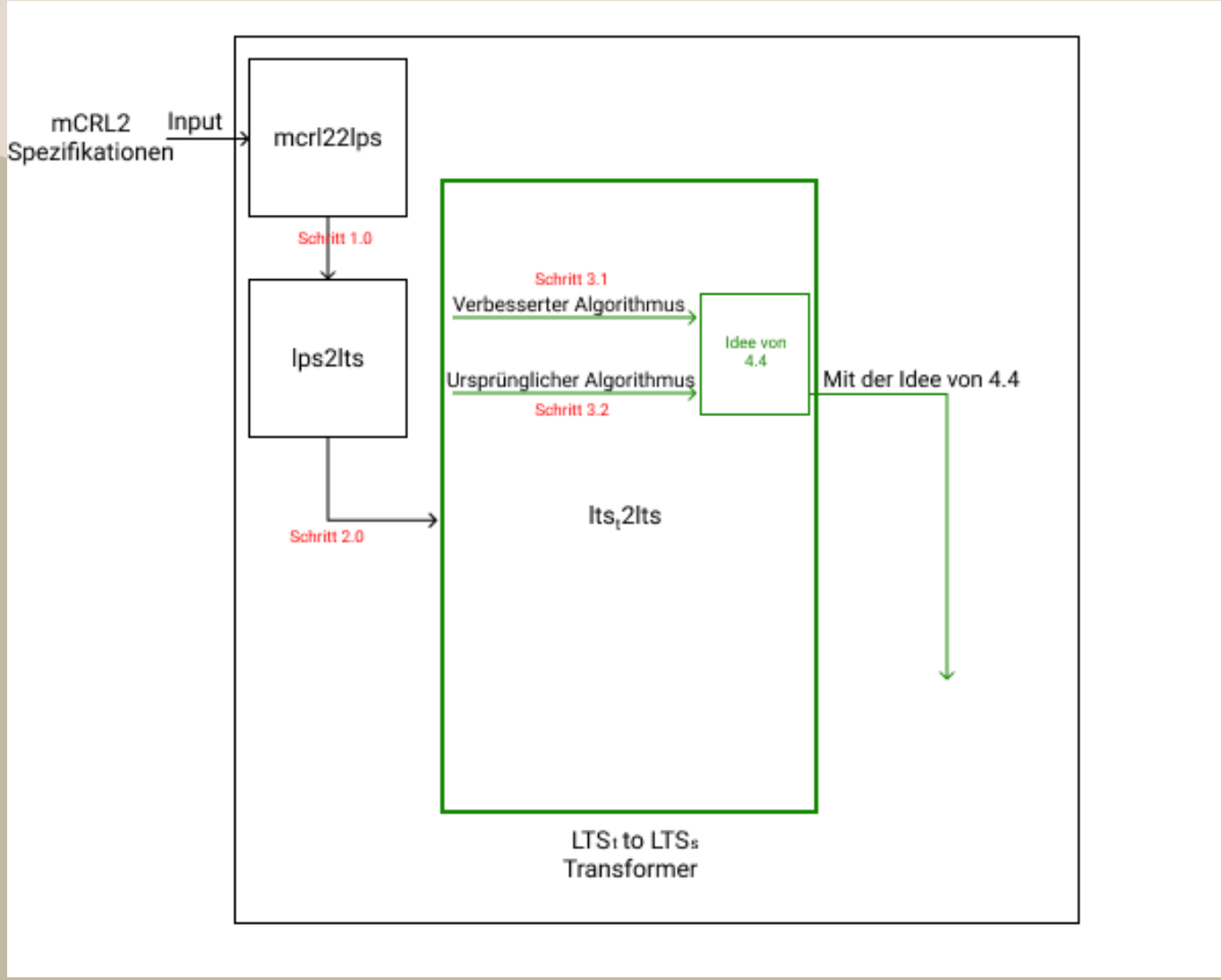
Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer

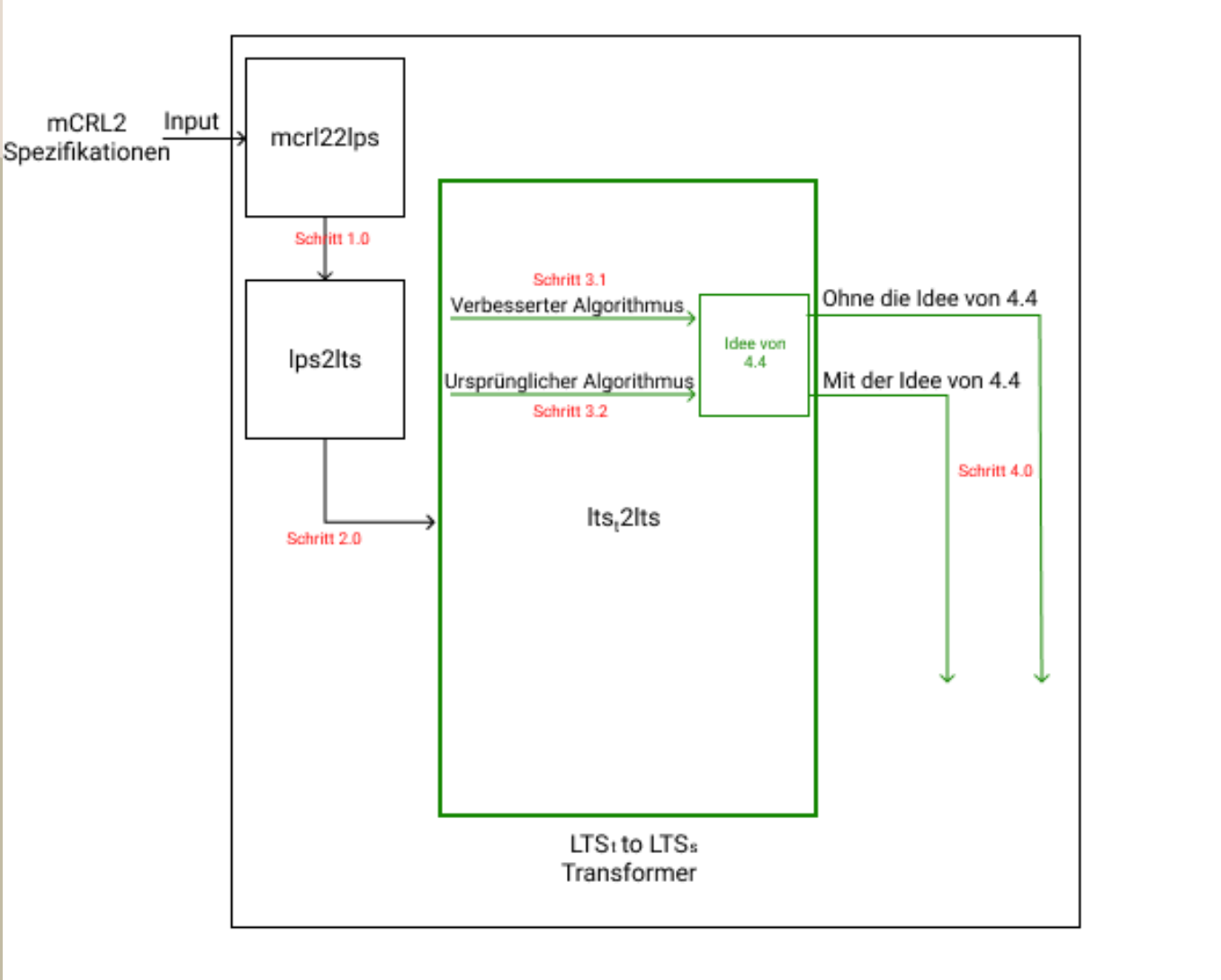
Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
Idee von
4.4
lts$_t$2lts
LTS$_I$ to LTS$_S$
Transformer

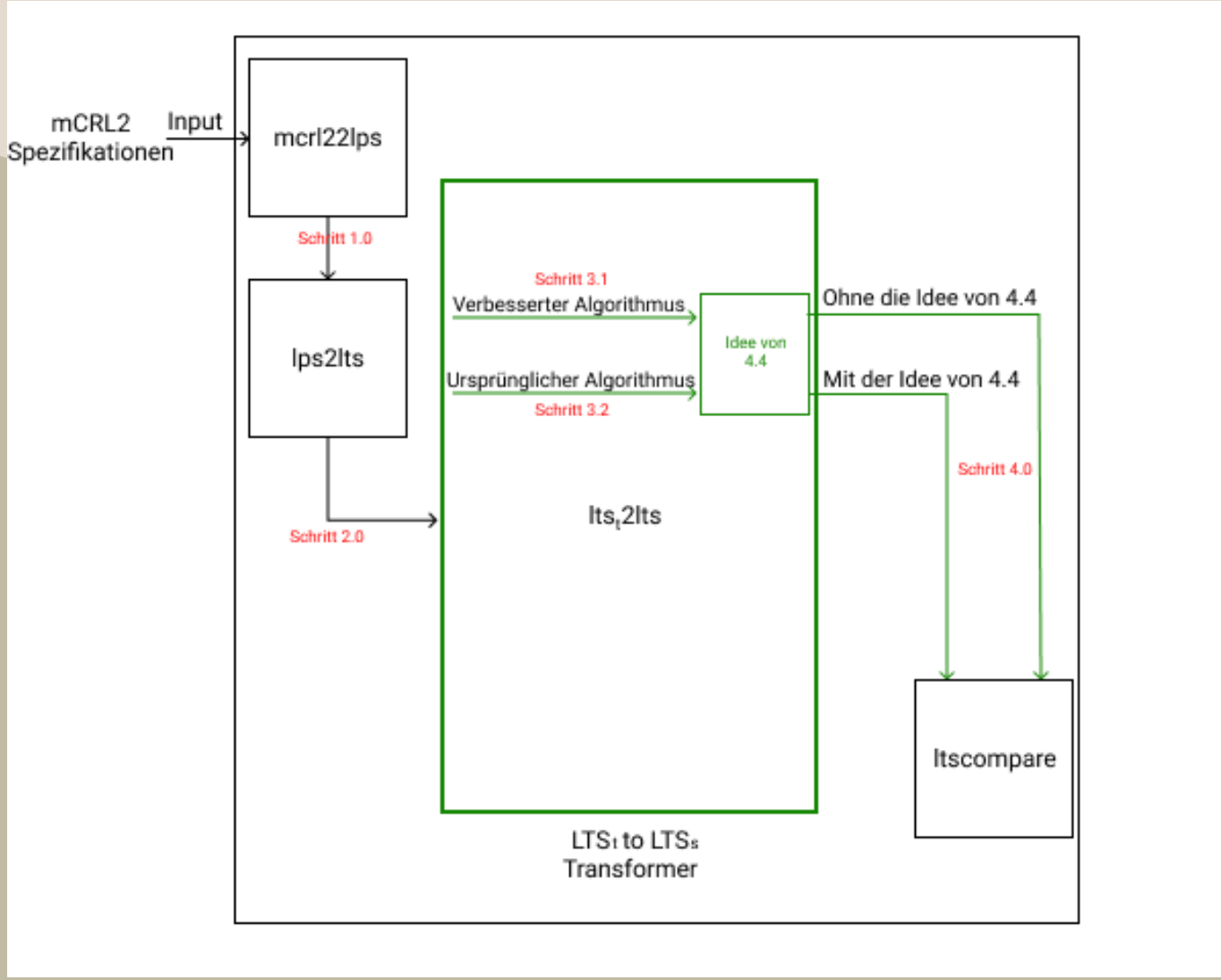
Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
Idee von 4.4
Mit der Idee von 4.4
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer

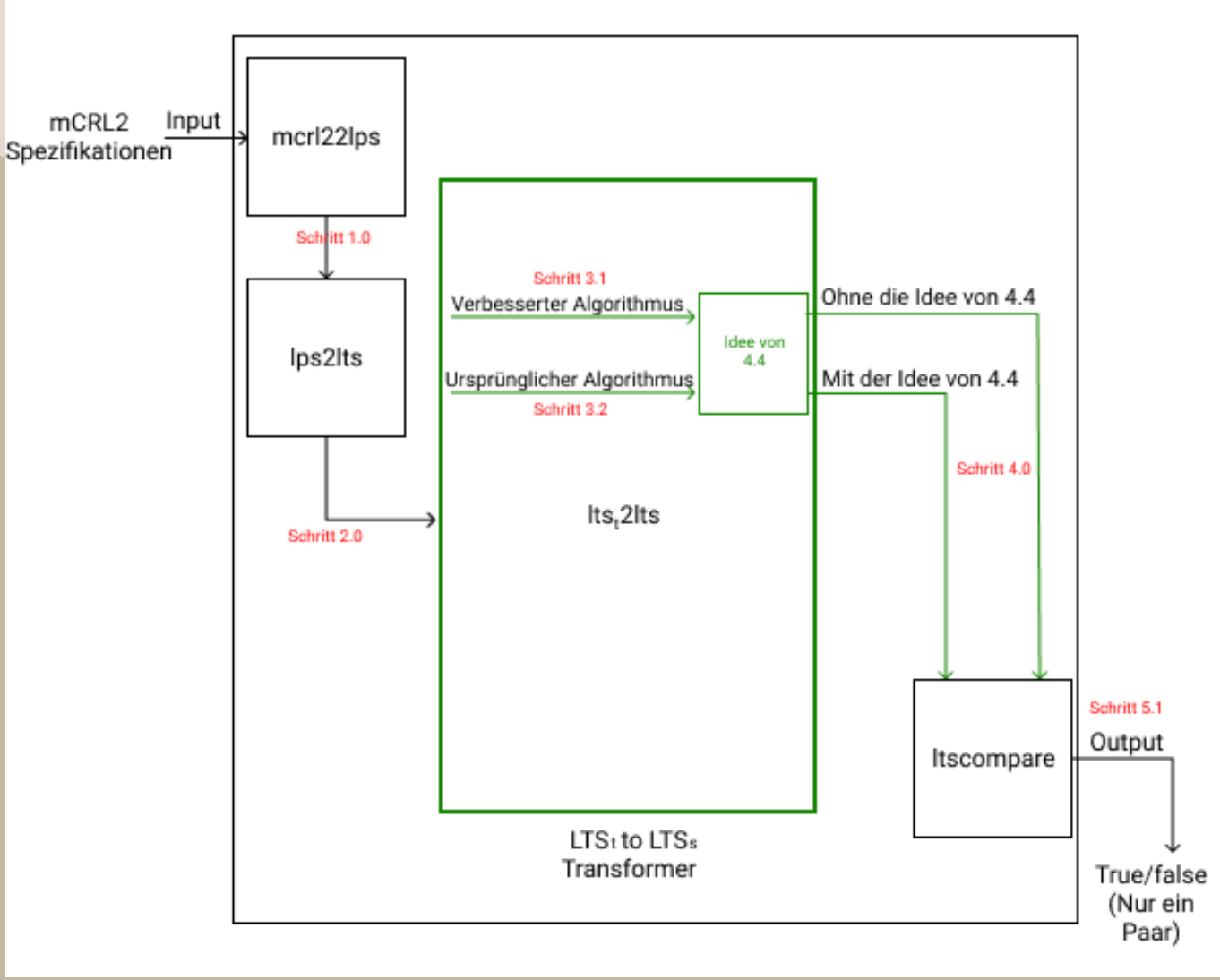
Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
Idee von 4.4
Ohne die Idee von 4.4
Mit der Idee von 4.4
Schritt 4.0
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer

Kernidee

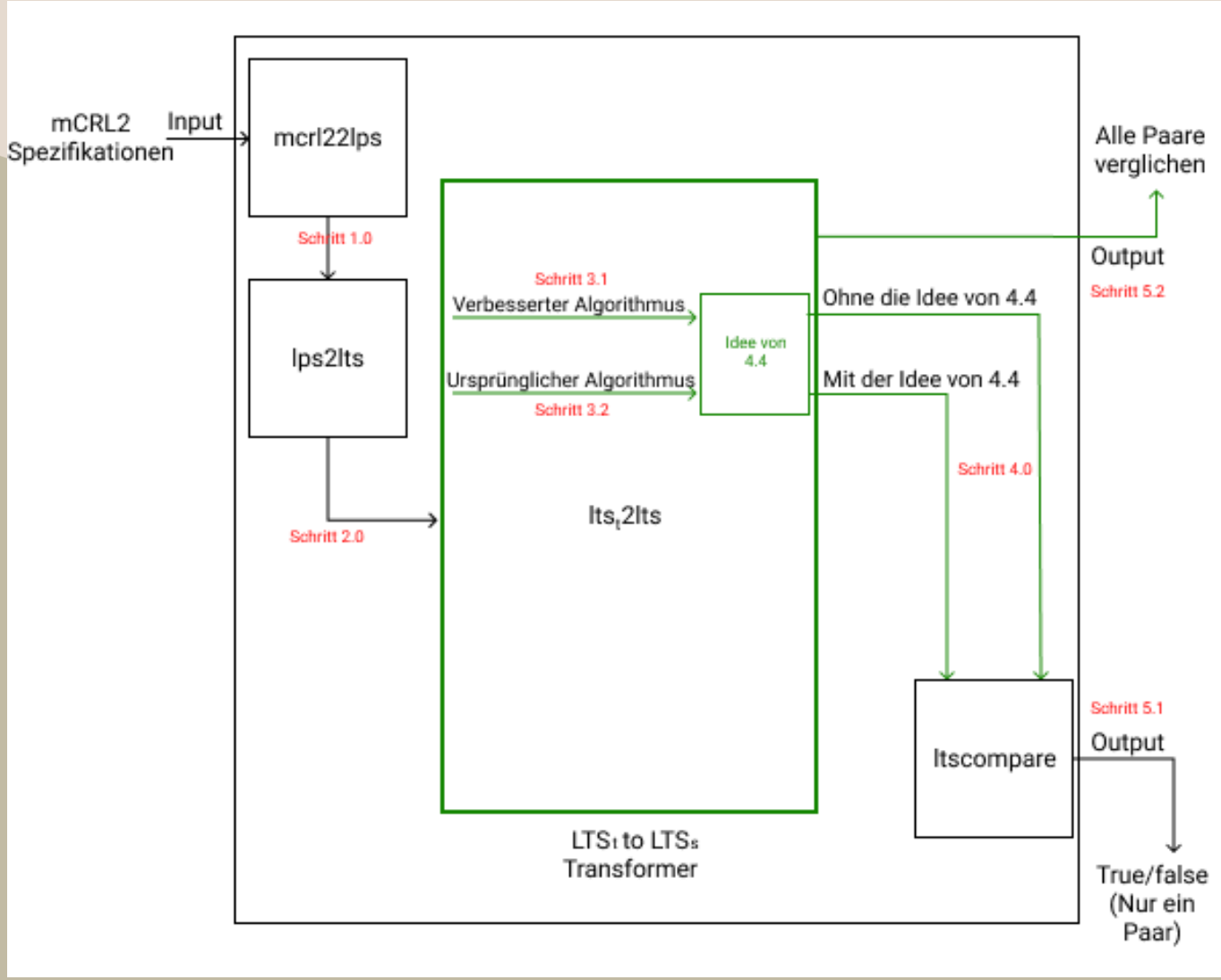

mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
Idee von 4.4
Ohne die Idee von 4.4
Mit der Idee von 4.4
Schritt 4.0
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer
ltscompare

Kernidee


mCRL2
Spezifikationen
Input
mcrl22lps
Schritt 1.0
lps2lts
Schritt 2.0
Schritt 3.1
Verbesserter Algorithmus
Ursprünglicher Algorithmus
Schritt 3.2
Idee von 4.4
Ohne die Idee von 4.4
Mit der Idee von 4.4
Schritt 4.0
lts$_t$2lts
LTS$_t$ to LTS$_s$
Transformer
ltscompare
Schritt 5.1
Output
True/false
(Nur ein Paar)

Kernidee

Ergebnis Foto



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: $ApplayRule3(process, transition, environment)$
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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$2)\ \frac{}{\mathcal{U}(p) \xrightarrow{E_X}_{\upsilon} \mathcal{U}_X(p)}\ X \subseteq A$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$1)\ \frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_{\upsilon} \upsilon(p')}$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$4)\ \frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_{\upsilon} \upsilon_X(p')}$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')}\ a \in X$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$5)\ \frac{p \overset{\alpha}{\not\mapsto} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$6)\ \frac{p \overset{\alpha}{\not\rightarrow} \forall \alpha \in X \cup \{\tau\} \quad p \overset{t}{\rightarrow} p'}{\upsilon_X(p) \overset{t}{\rightarrow}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus

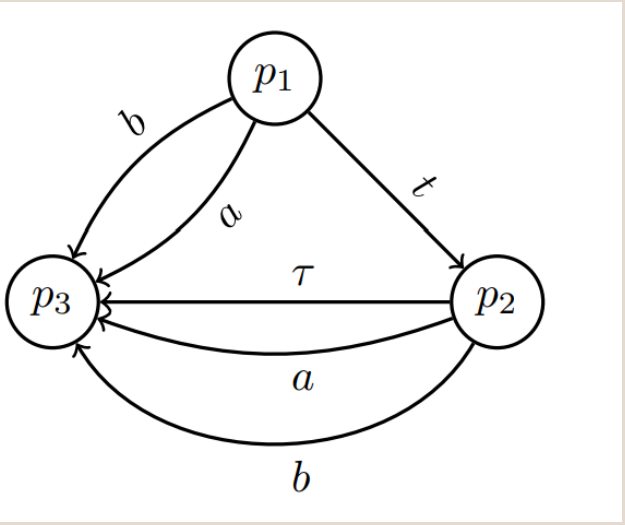
Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

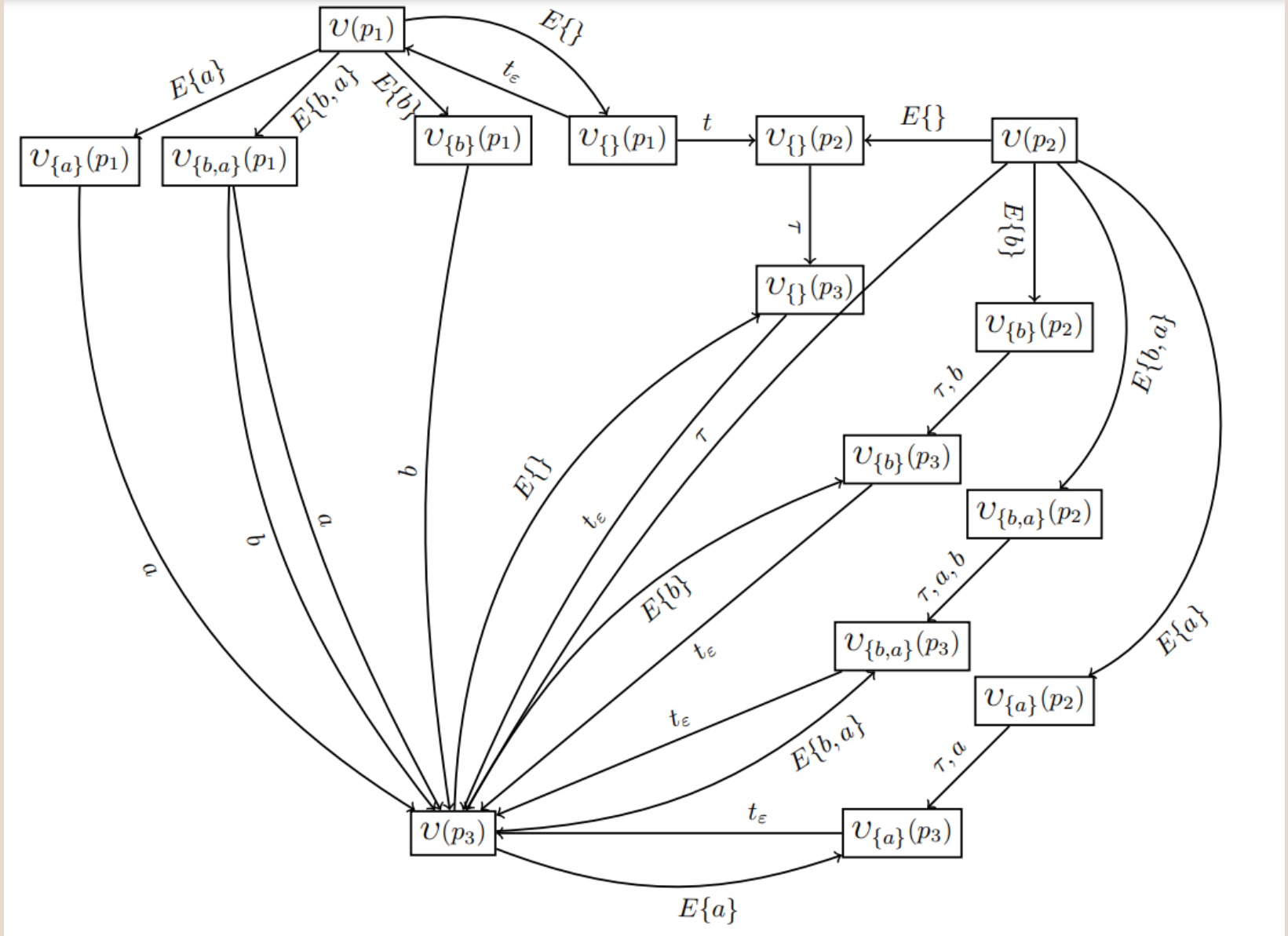
1: $processes, allSharedEnvironments, processSharedEnvironment$
2: $newProcesses \leftarrow emptyList$
3: **for** $process \in processes$ **do**
4: $processSharedEnvironment.add(process.getName(), allSharedEnvironments)$
5: $ApplyRule2(allSharedEnvironments, process)$
6: **for** $transition \in process.getOriginalTransitions()$ **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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$5)\ \frac{p \overset{\alpha}{\not\mapsto} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

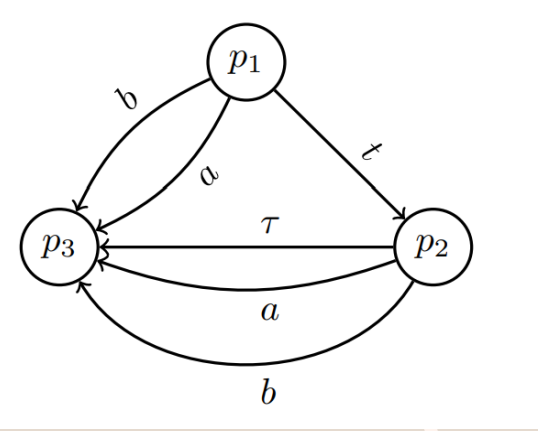
Ursprünglicher Algorithmus



WIE?



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: $ApplayRule3(process, transition, environment)$
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}}

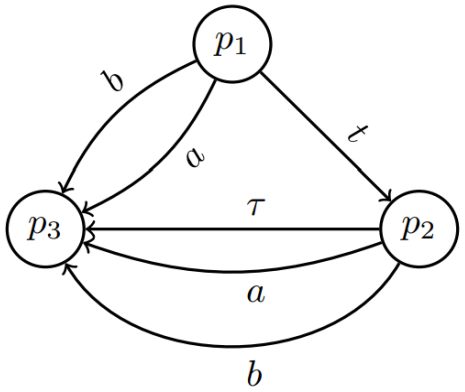
Ursprünglicher Algorithmus

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

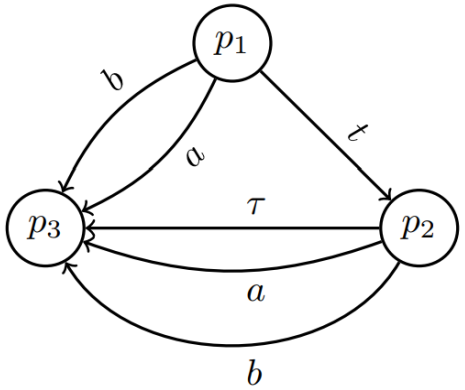
{{}, {a}, {b}, {a, b}}



$$2)\ \frac{}{\mathcal{U}(p) \xrightarrow{E_X}_{\upsilon} \mathcal{U}_X(p)}\ X \subseteq A$$

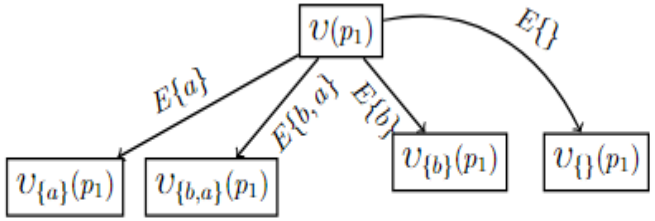
Ursprünglicher Algorithmus

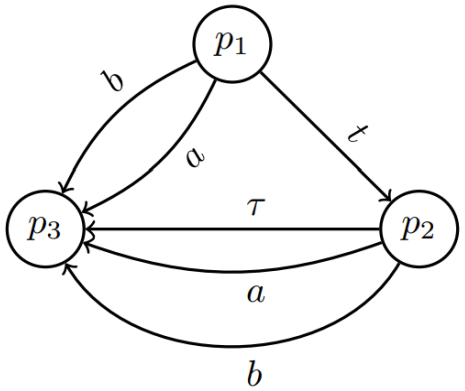
$\boxed{\upsilon(p_1)}$



$$2)\ \frac{}{\mathcal{U}(p) \xrightarrow{E_X}_{\upsilon} \mathcal{U}_X(p)}\ X \subseteq A$$

Ursprünglicher Algorithmus





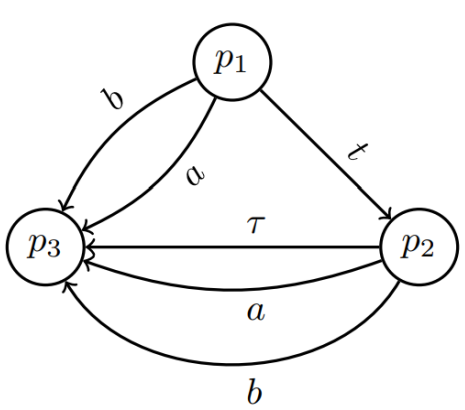
$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

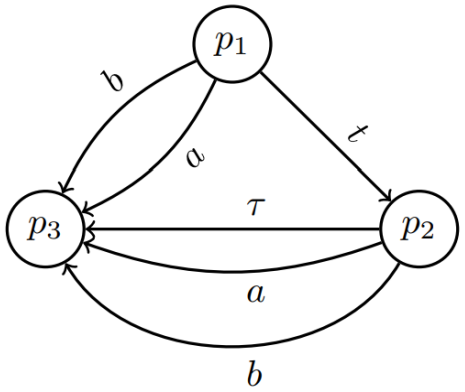
Require:

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$



$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

Ursprünglicher Algorithmus



2) $\dfrac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

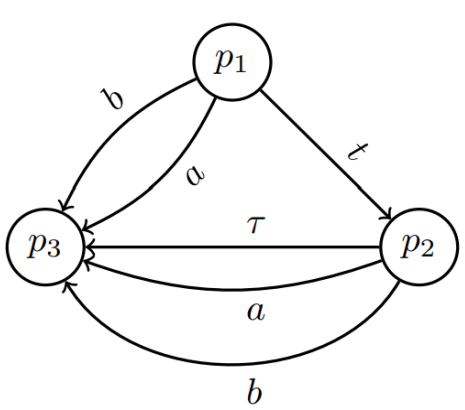
```
1: processes, allSharedEnvironments, processSharedEnvironment
2: newProcesses ← emptyList
3: for process ∈ processes do
4:     processSharedEnvironment.add(process.getName(), allSharedEnvironments)
5:     ApplyRule2(allSharedEnvironments, process)
6:     for transition ∈ process.getOriginalTransitions() do
7:         processEnvironments ← processSharedEnvironment.getValueOf(process.getName())
8:         for environment ∈ 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:                ApplayRule3(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 ← getNewProcesses()
```

Ursprünglicher Algorithmus

Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

Require:

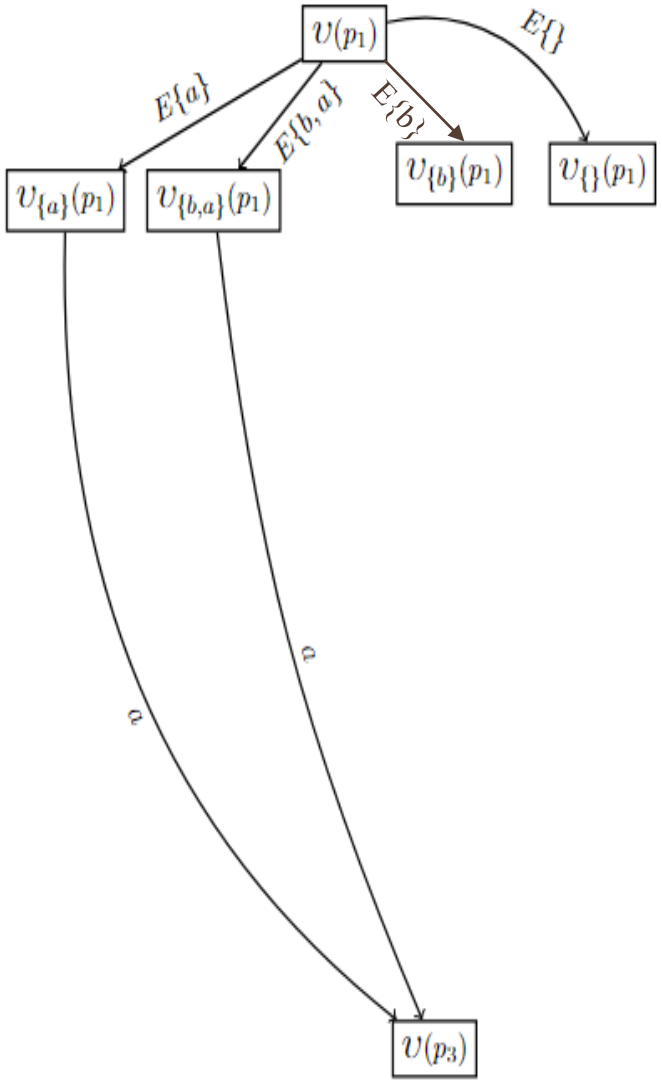
1: $processes, allSharedEnvironments, processSharedEnvironment$
2: $newProcesses \leftarrow emptyList$
3: **for** $process \in processes$ **do**
4: $processSharedEnvironment.add(process.getName(), allSharedEnvironments)$
5: $ApplyRule2(allSharedEnvironments, process)$
6: **for** $transition \in process.getOriginalTransitions()$ **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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

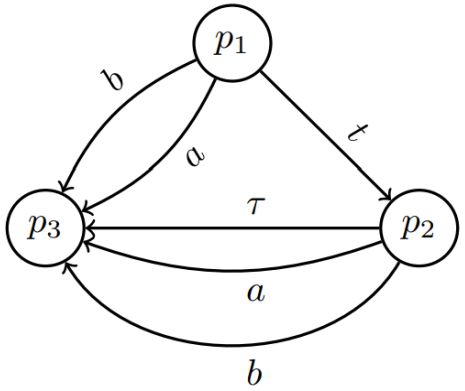


$$2)\ \frac{}{\mathcal{U}(p) \xrightarrow{E_X}_{\upsilon} \mathcal{U}_X(p)}\ X \subseteq A$$

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

Ursprünglicher Algorithmus

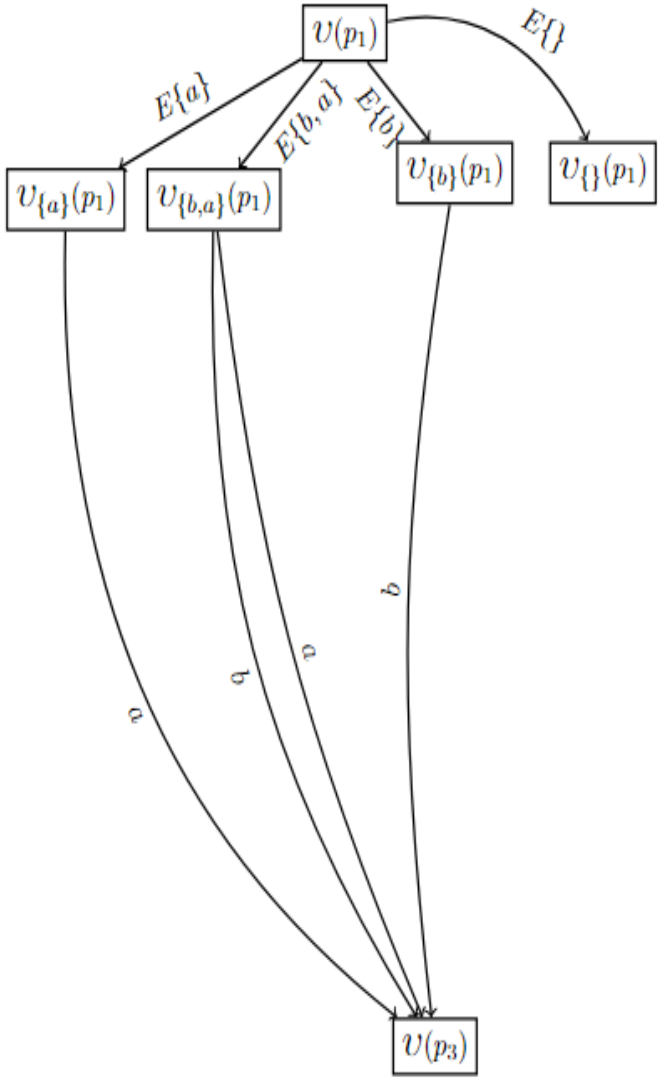


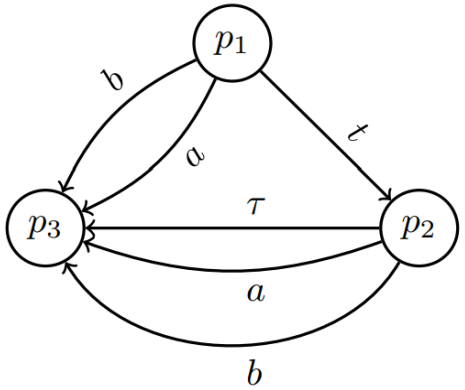


$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')}\ a \in X$$

Ursprünglicher Algorithmus





$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

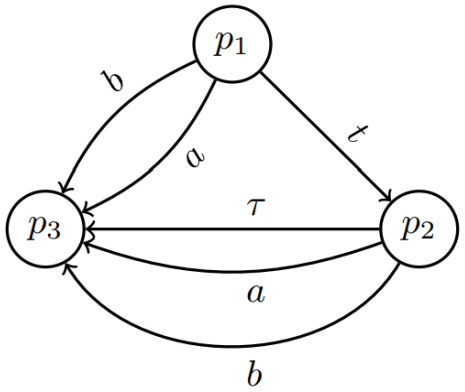
$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')}\ a \in X$$

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: $ApplayRule3(process, transition, environment)$
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) $$\frac{}{\mathcal{U}(p) \xrightarrow{E_X}_{\upsilon} \mathcal{U}_X(p)} \; X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\mathcal{U}_X(p) \xrightarrow{a}_{\upsilon} \mathcal{U}(p')} \; a \in X$$

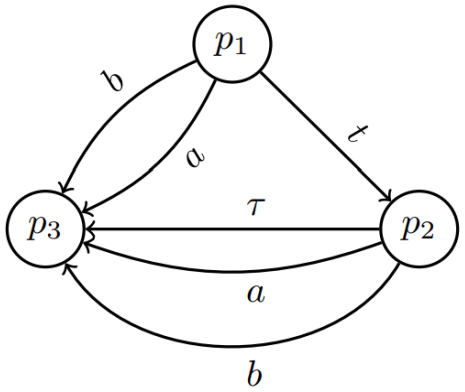
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\mathcal{U}_X(p) \xrightarrow{t_\varepsilon}_{\upsilon} \mathcal{U}(p)}$$

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$



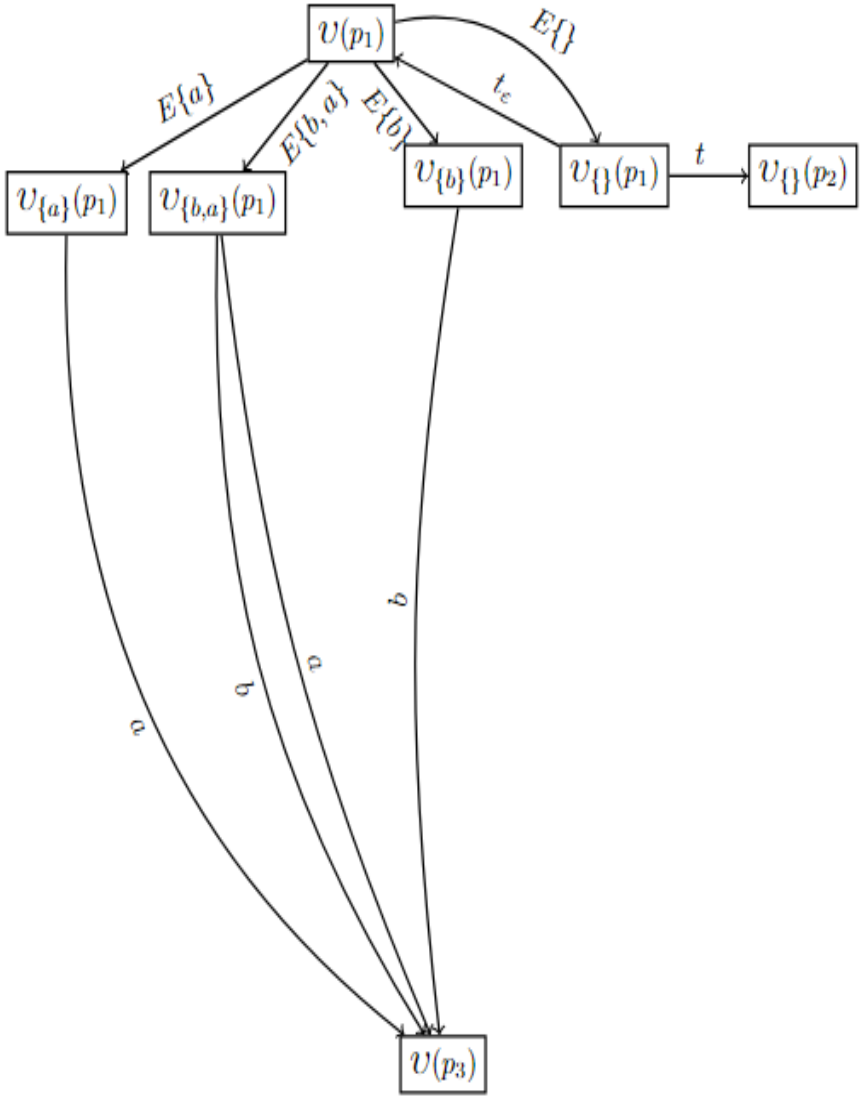
2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)} \; X \subseteq A$$

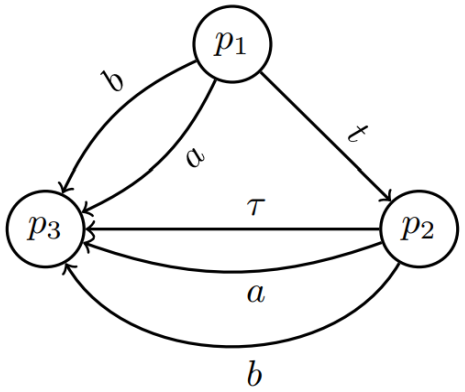
3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')} \; a \in X$$

5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_{\upsilon} \upsilon(p)}$$

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

Ursprünglicher Algorithmus





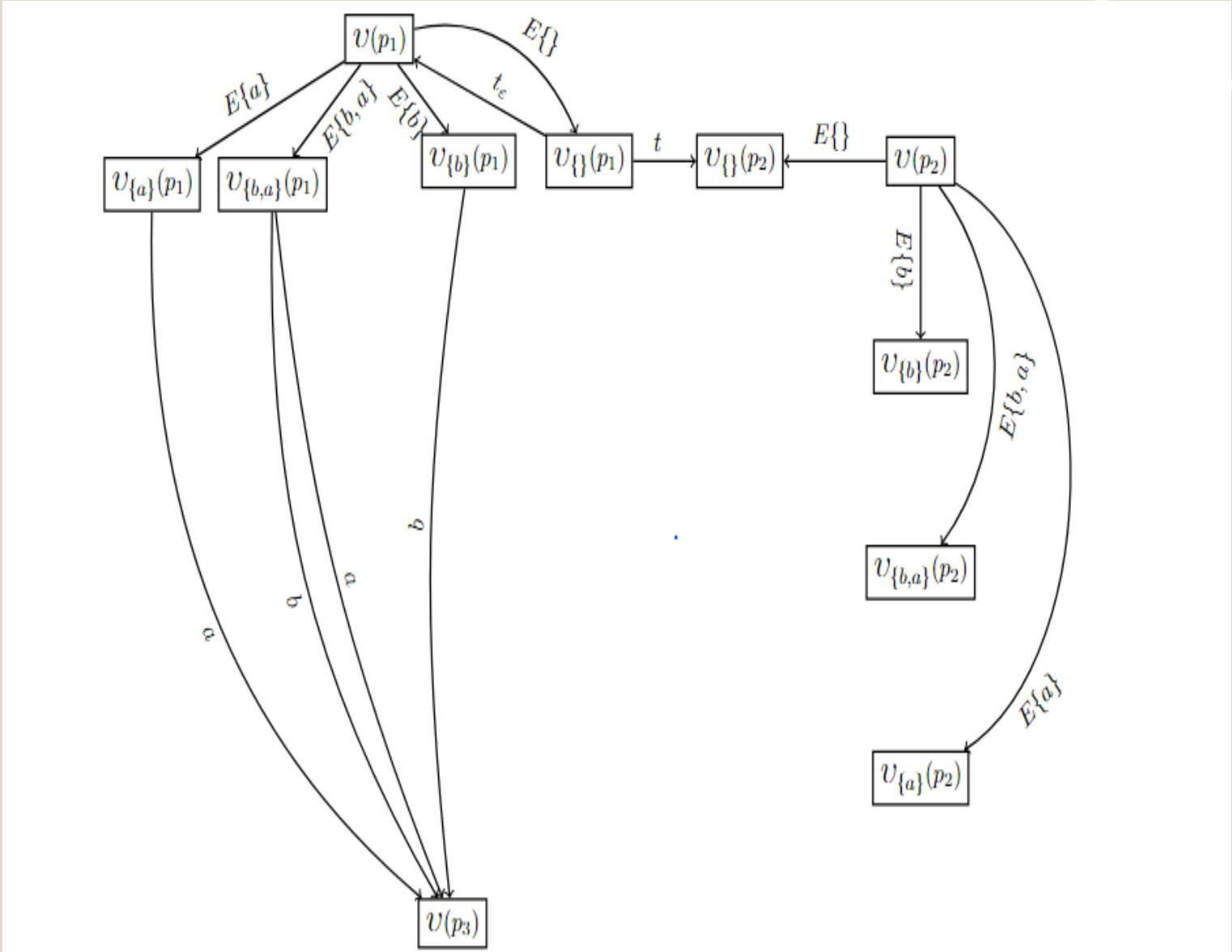
2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \; X \subseteq A$$

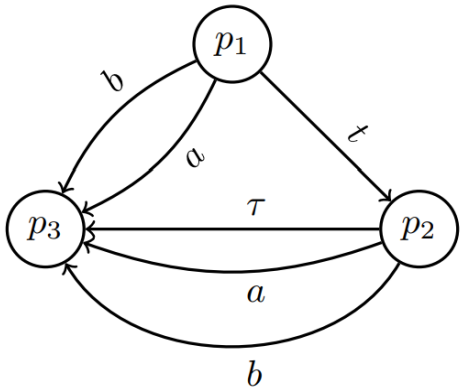
3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \; a \in X$$

5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

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

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)}\ X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')}\ a \in X$$

5) $$\frac{p \not\xrightarrow{\alpha}\ \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

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

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$1)\ \frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_{\upsilon} \upsilon(p')}$$

Ursprünglicher Algorithmus

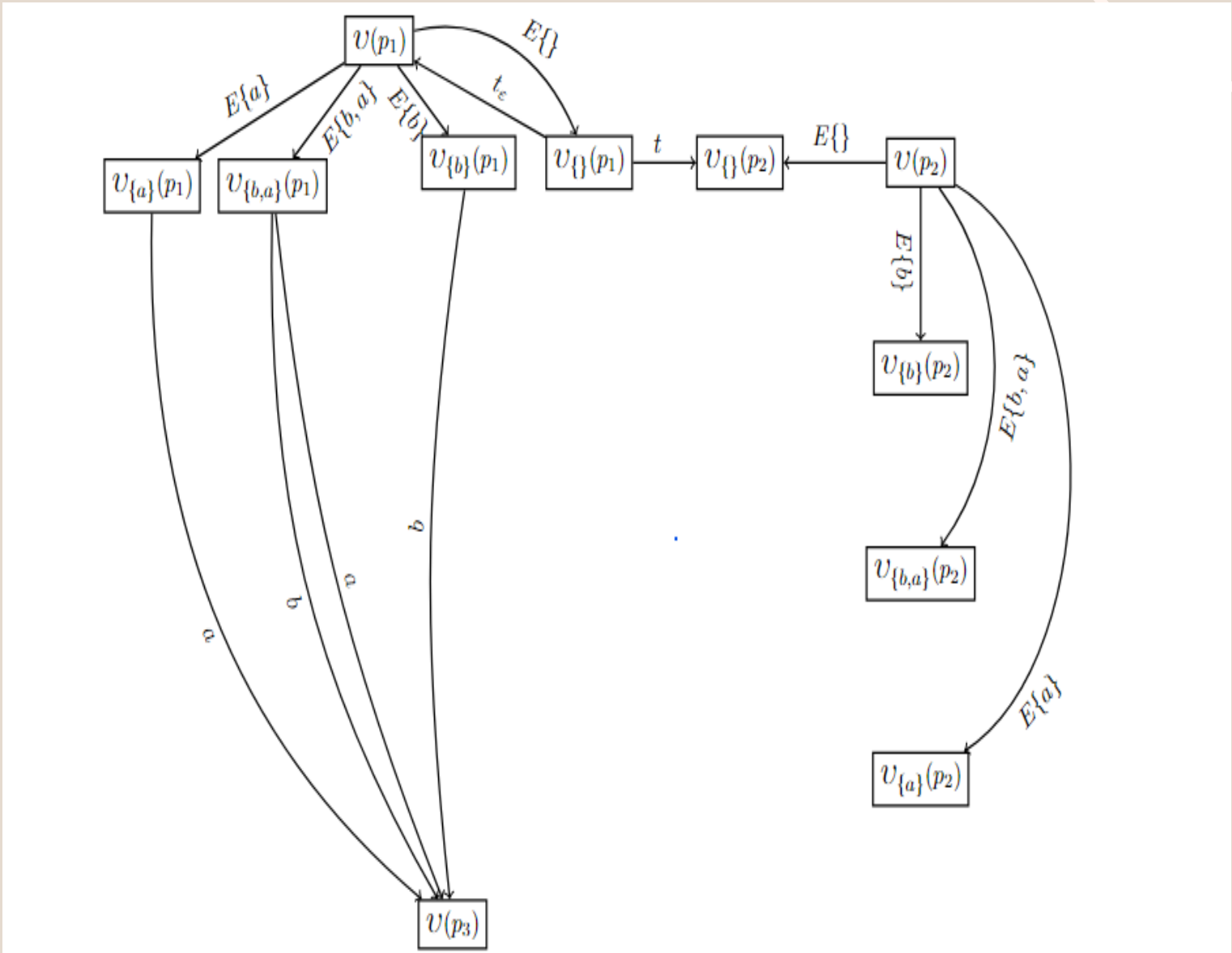
Algorithm 1 LTS_t zu LTS_s Transformationsalgorithmus ($getLTS_s$)

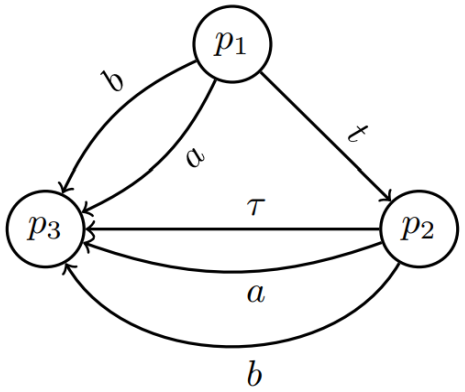
Require:

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: $ApplayRule3(process, transition, environment)$
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: $result \leftarrow getNewProcesses()$

$$4)\ \frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_{\upsilon} \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \; X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \; a \in X$$

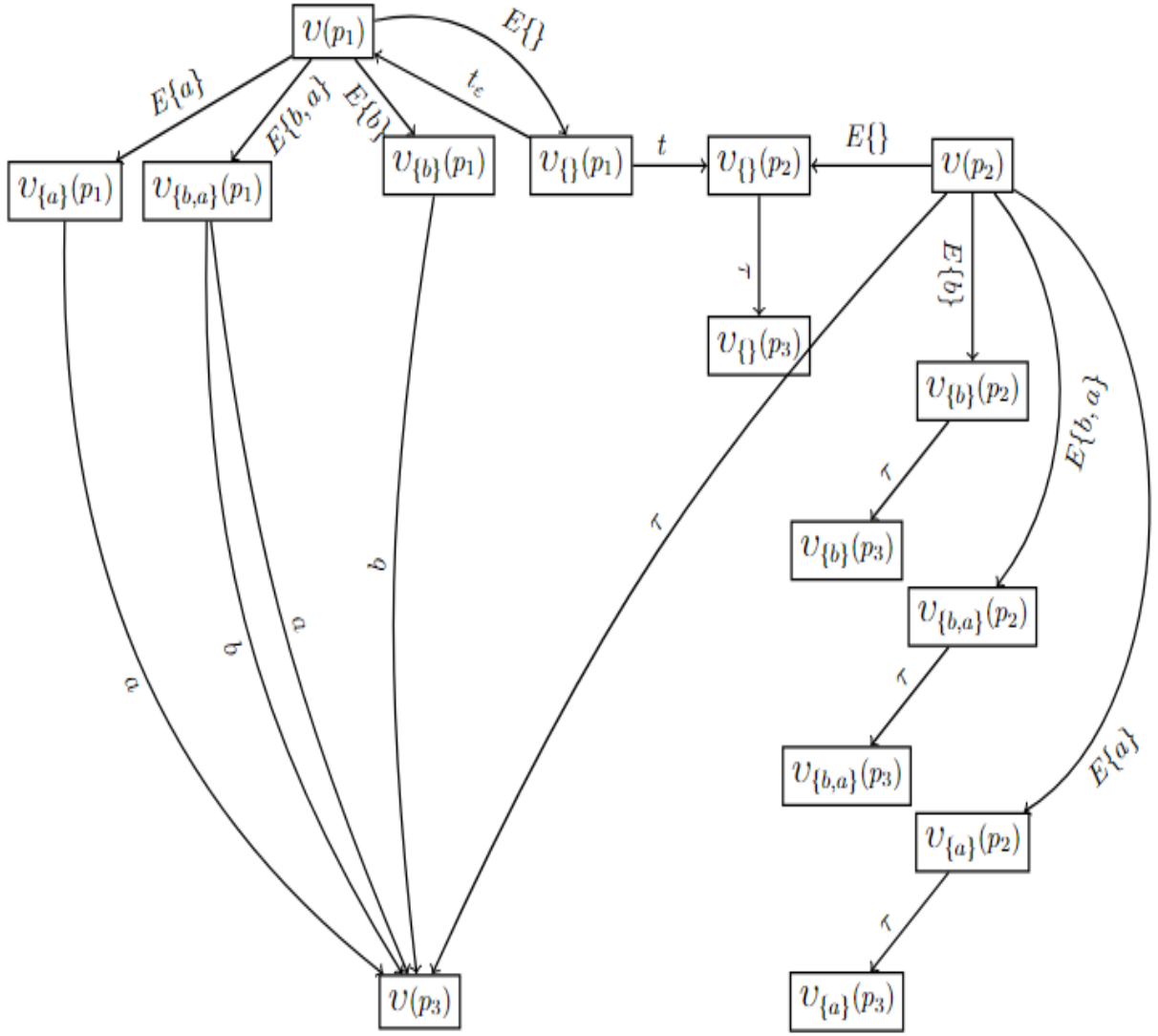
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

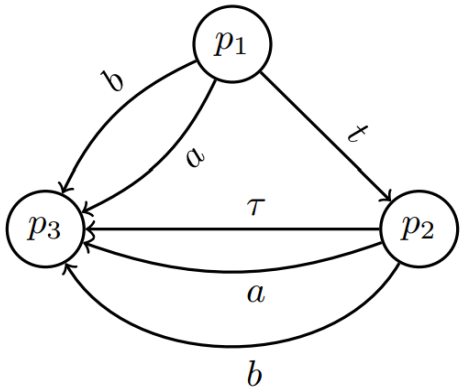
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \quad X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \quad a \in X$$

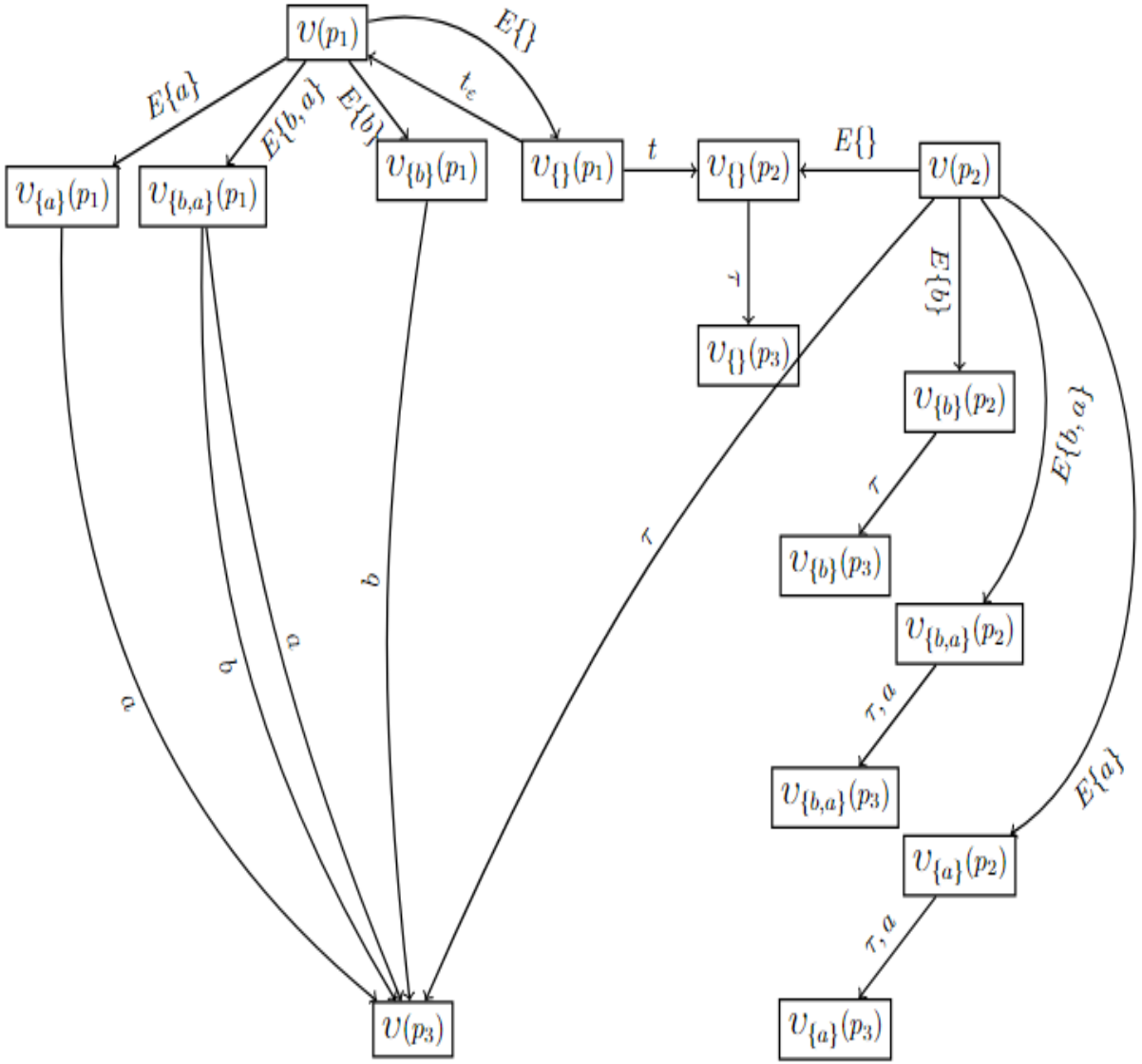
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

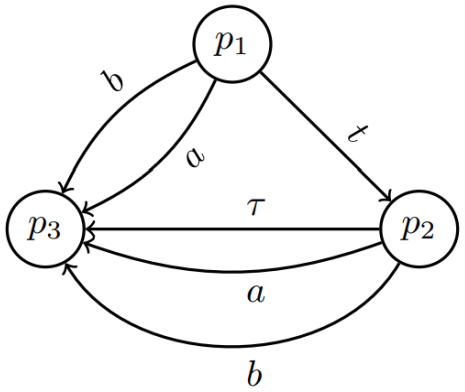
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \; X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \; a \in X$$

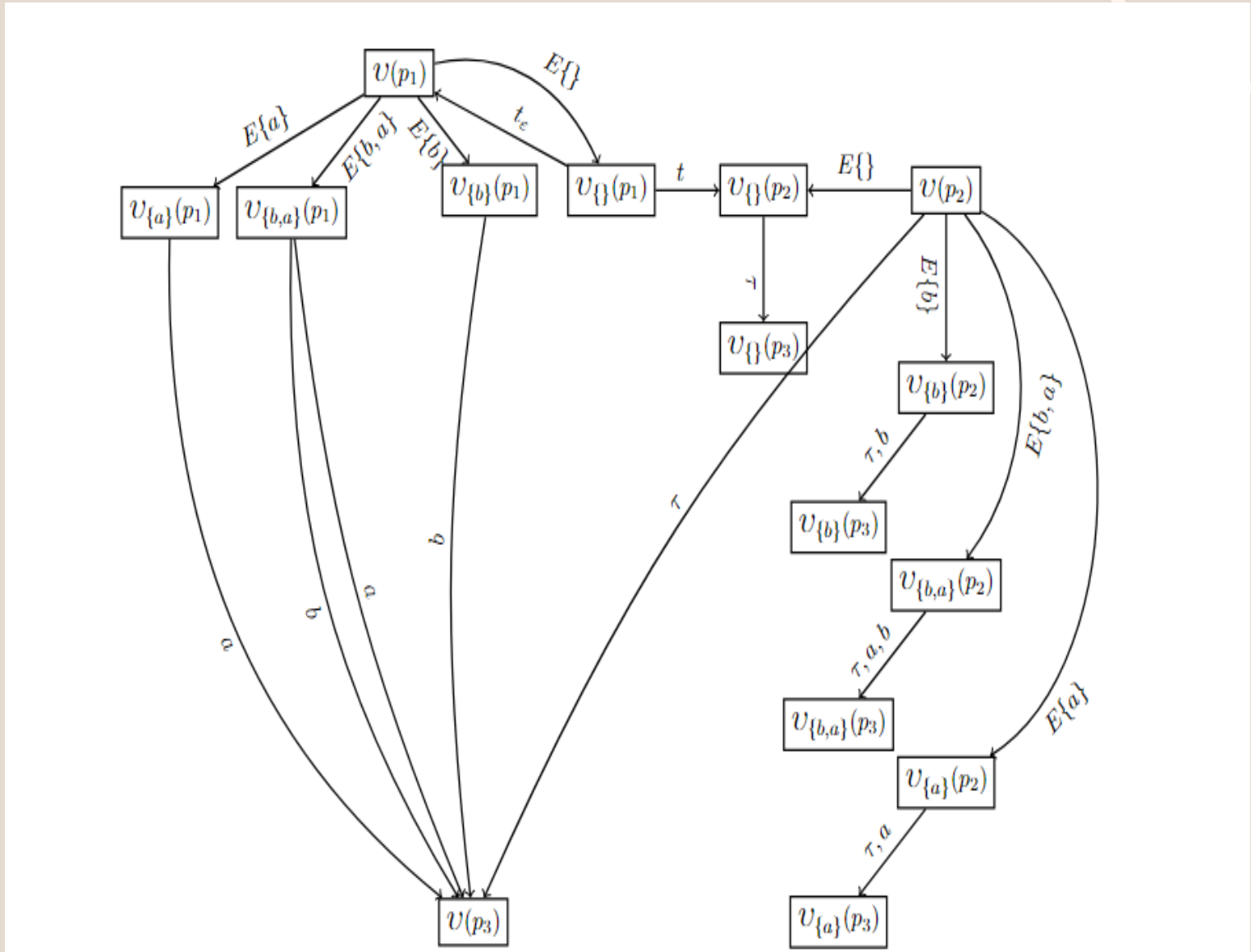
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

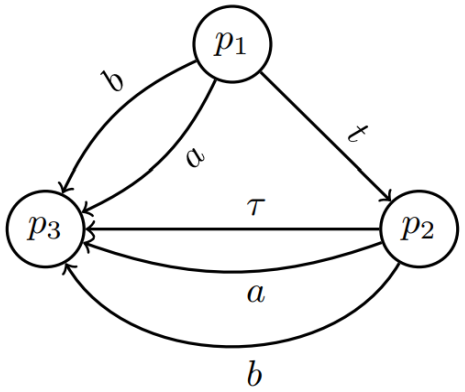
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \quad X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \quad a \in X$$

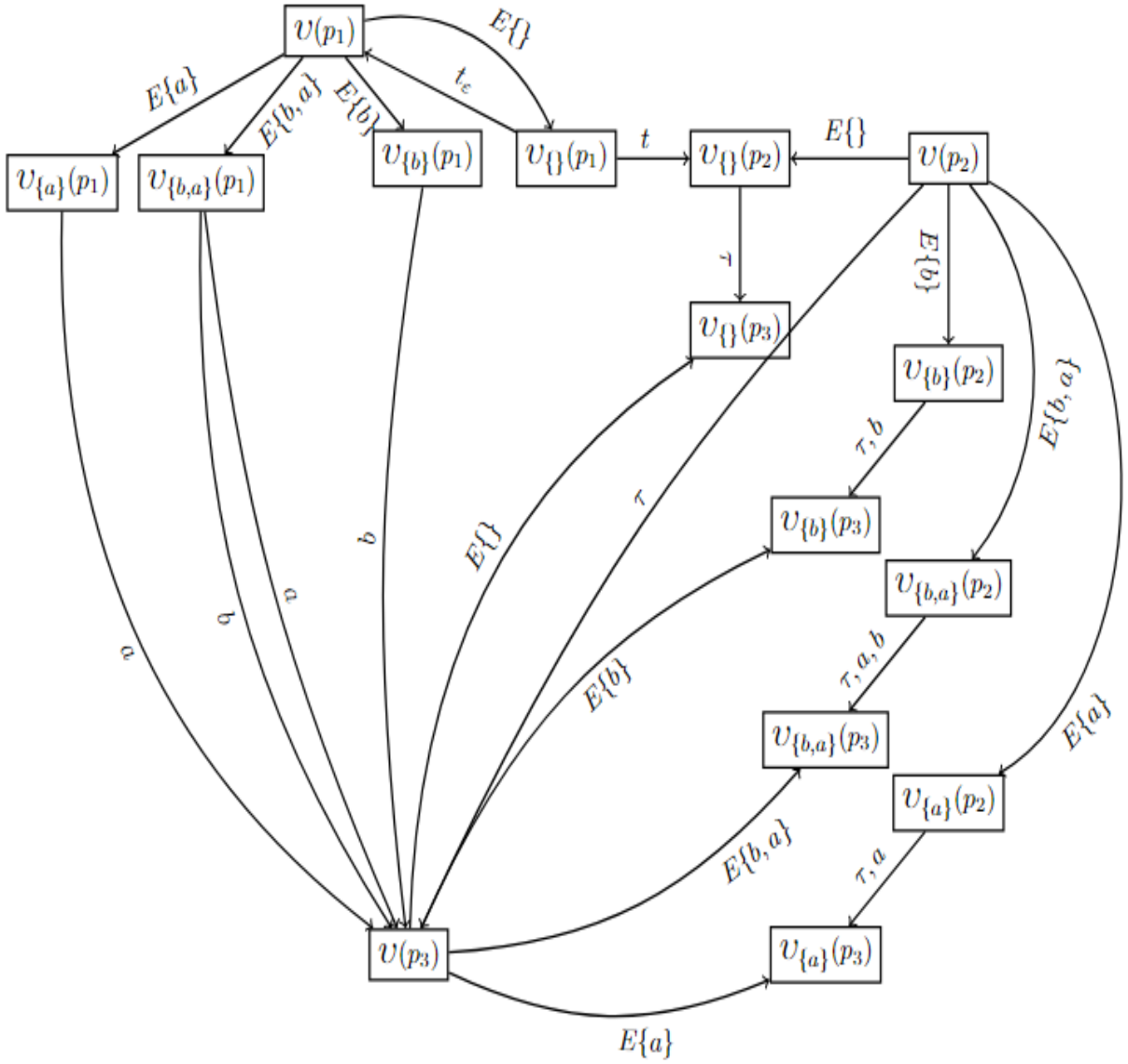
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

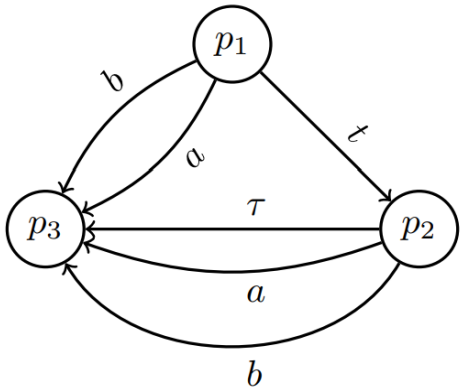
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \quad X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \quad a \in X$$

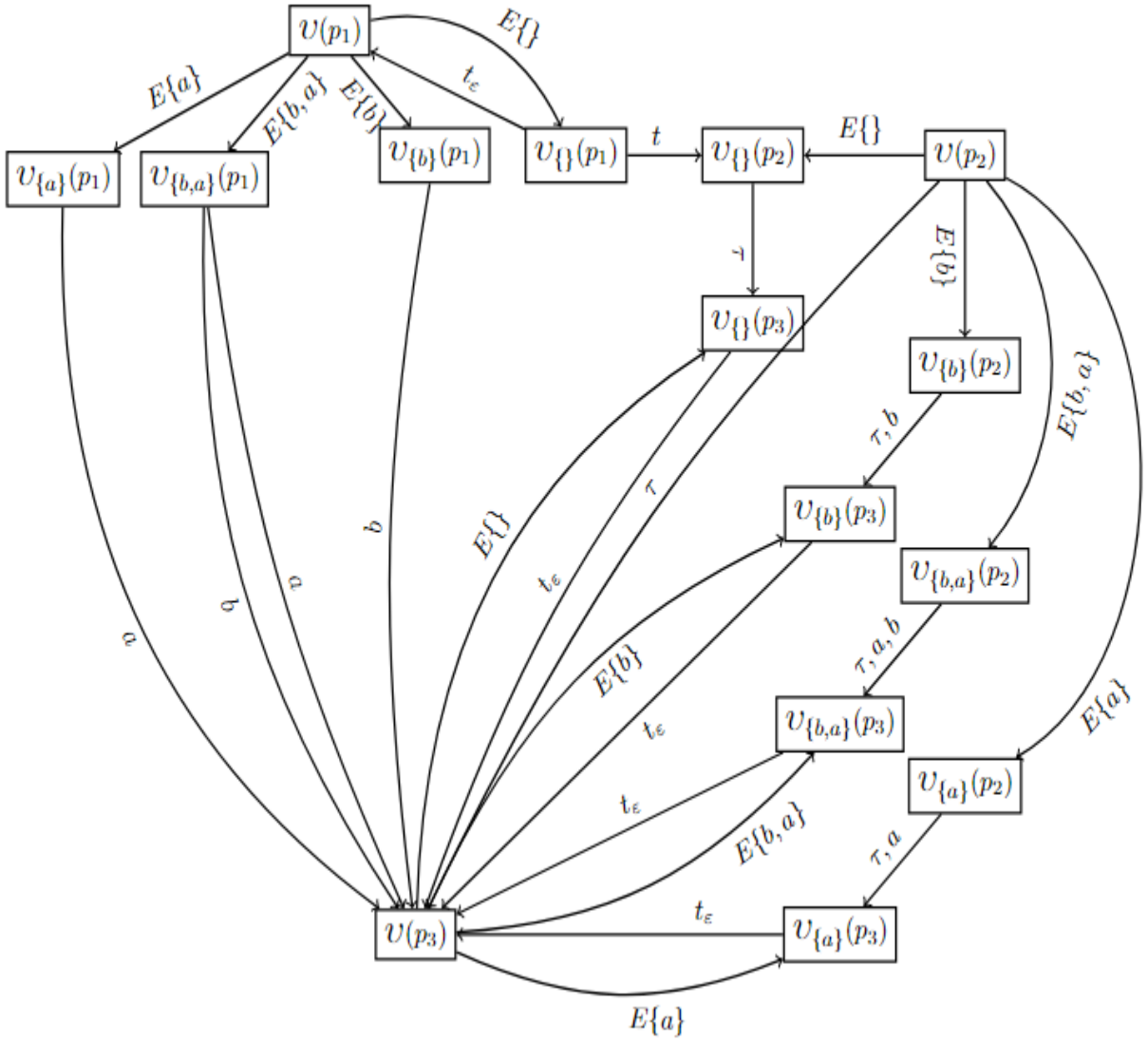
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

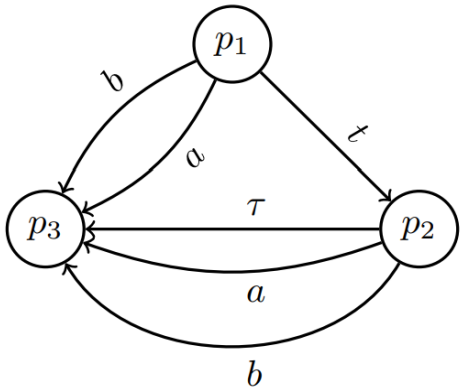
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Ursprünglicher Algorithmus





2) $$\frac{}{\upsilon(p) \xrightarrow{E_X}_\upsilon \upsilon_X(p)} \quad X \subseteq A$$

3) $$\frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')} \quad a \in X$$

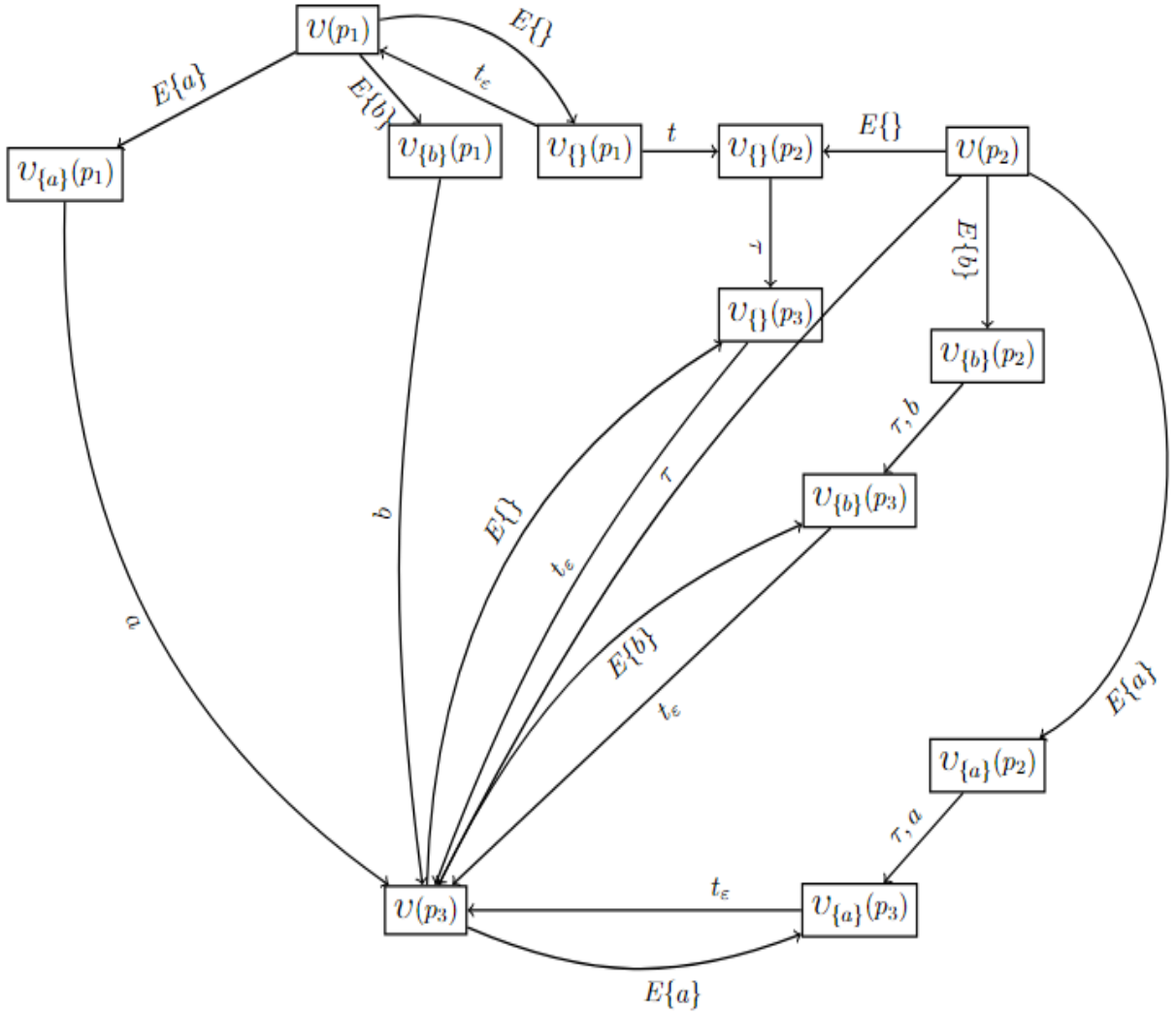
5) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

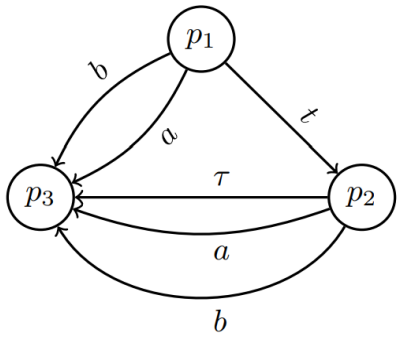
6) $$\frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\} \quad p \xrightarrow{t} p'}{\upsilon_X(p) \xrightarrow{t}_\upsilon \upsilon_X(p')}$$

1) $$\frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

4) $$\frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Verbesserte Algorithmus





$$\frac{}{\upsilon(p) \xrightarrow{Ex}_\upsilon \upsilon_X(p)}\ X \in U, \text{ mit}$$

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

$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_\upsilon \upsilon(p')}\ a \in X$$

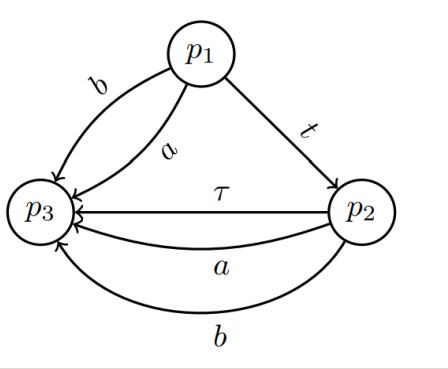
$$5)\ \frac{p \not\xrightarrow{\alpha} \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\varepsilon}_\upsilon \upsilon(p)}$$

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

$$1)\ \frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_\upsilon \upsilon(p')}$$

$$4)\ \frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_\upsilon \upsilon_X(p')}$$

Verbesserte Algorithmus



$$\frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \in U, \text{ mit}$$

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

$$3)\ \frac{p \xrightarrow{a} p'}{\upsilon_X(p) \xrightarrow{a}_{\upsilon} \upsilon(p')}\ a \in X$$

$$5)\ \frac{p \not\xrightarrow{\alpha}\ \forall \alpha \in X \cup \{\tau\}}{\upsilon_X(p) \xrightarrow{t_\epsilon}_{\upsilon} \upsilon(p)}$$

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

$$1)\ \frac{p \xrightarrow{\tau} p'}{\upsilon(p) \xrightarrow{\tau}_{\upsilon} \upsilon(p')}$$

$$4)\ \frac{p \xrightarrow{\tau} p'}{\upsilon_X(p) \xrightarrow{\tau}_{\upsilon} \upsilon_X(p')}$$

Einzige Umgebungsaktion

{{}, {a}, {b}, {a, b}}

Einzige Umgebungsaktion

Einzige Umgebungsaktion

File-Umgebung

Einzige Umgebungsaktion

{{}, {a}, {b}, {a, b}} → $E_{\{\ldots\}}$

File-Umgebung →

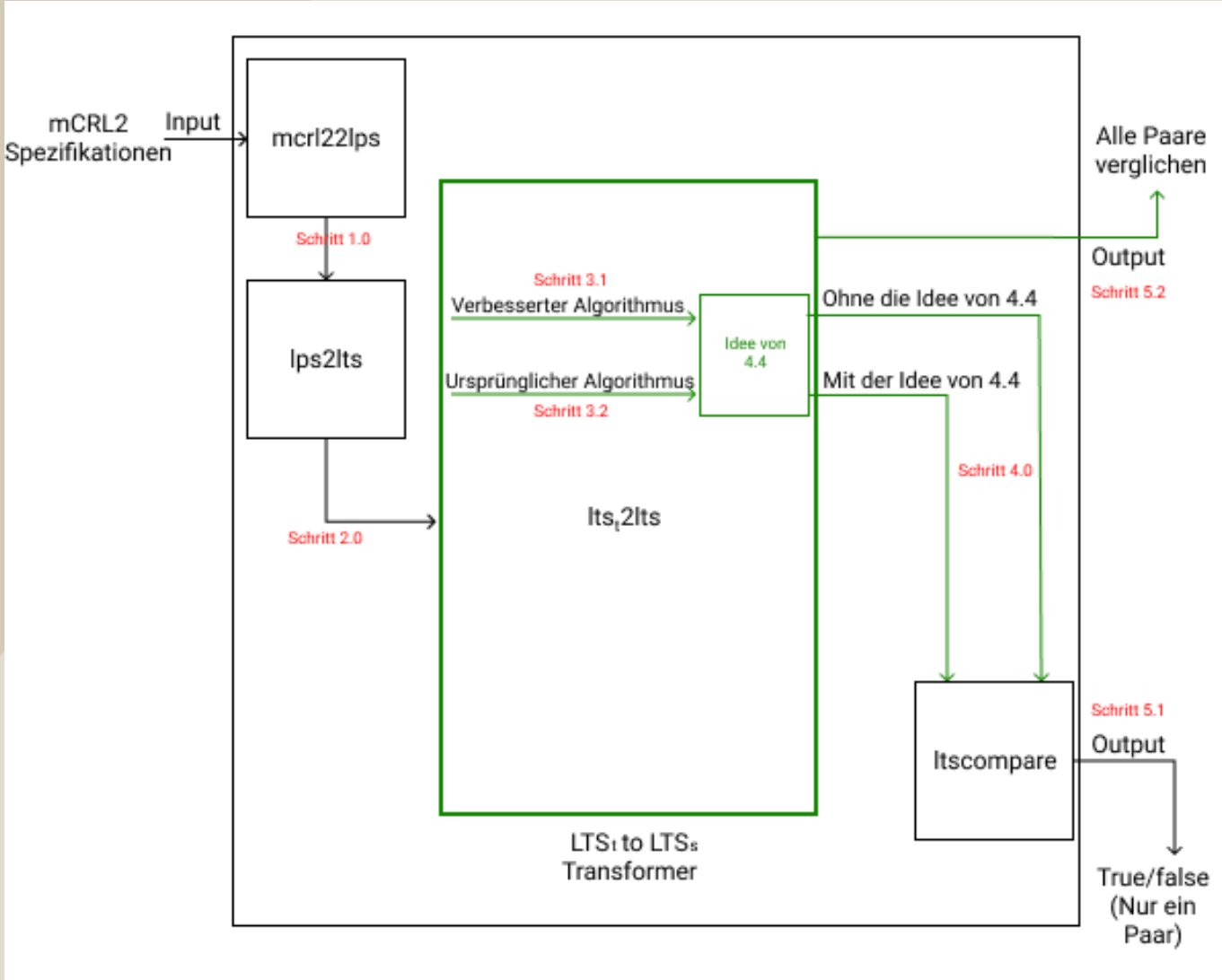
Einzige Umgebungsaktion

{{}, {a}, {b}, {a, b}} → $E_{\{...\}}$

File-Umgebung → File-Umgebung

Überprüfung eines oder aller Paare

Überprüfung eines oder aller Paare



Überprüfung eines oder aller Paare

Algorithm 2 Algorithmus von starker Bisimilarität

Require:

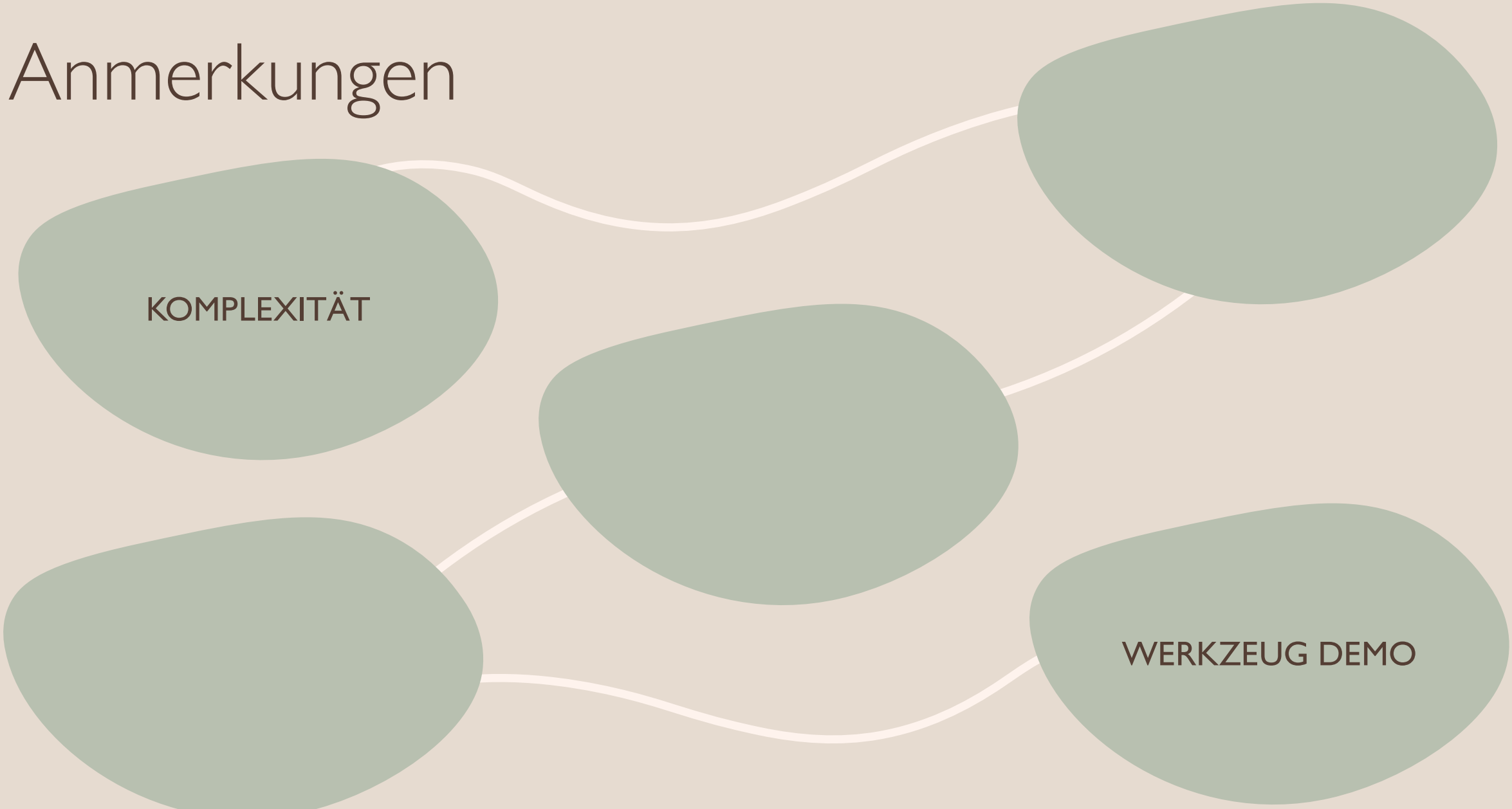
```
processList1
processList2
pairs ← all non-symmetric recursive pairs of processList1 and processList2
while true do
    pairs.removeIf(p →
        !checkSimulation(p2q, pairs, p.getFirst(), p.getLast())||
        !checkSimulation(q2p, pairs, p.getLast(), p.getFirst())
    )
    If pairs does not change beark
end while
return pairs
```

Algorithm 3 checkSimulation

Require:

```
leftToRight (check (p,q) or (q,p))
pairs
Prozess p from pair
Prozess q from pair
return p.getTransitions().allMatch(t1 →
            q.getTransitions().anyMatch(t2 →
                t1.getAction().getName().equals(t2.getAction().getName())
            &&
    areSuccessorsInR(leftToRight, pairs, t1.getSuccessor(), t2.getSuccessor())))
```

Anmerkungen



Komplexität

$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

Komplexität

$$2)\ \frac{}{\upsilon(p) \xrightarrow{E_X}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

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

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

Komplexität

$$2)\ \frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

$$\frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \in U, \text{ mit}$$

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

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

Komplexität

$$2)\ \frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

$$\frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \in U, \text{ mit}$$

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

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

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

Komplexität

$$2)\ \frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \subseteq A$$

$$\frac{}{\upsilon(p) \xrightarrow{Ex}_{\upsilon} \upsilon_X(p)}\ X \in U, \text{ mit}$$

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

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

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

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