

communicating and mobile systems:

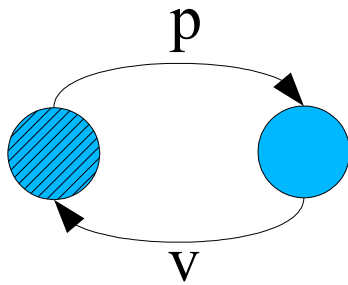
the  - calculus

chapter 7: observation equivalence: examples

Overview

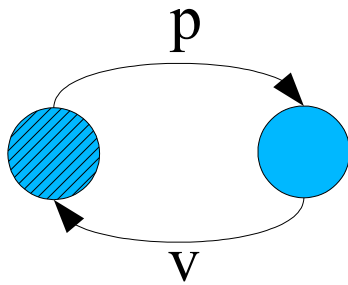
- Äquivalenz von Systemen (starke / schwache Äquivalenz)
- Beispiel: Lotterie
- Beispiel: Job Shop
- Beispiel: Schedule
- Beispiel: Buffer
- Beispiel: Counter

Starke Äquivalenz

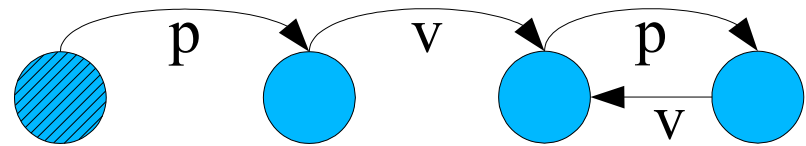


$$S = p.v.S$$

Starke Äquivalenz



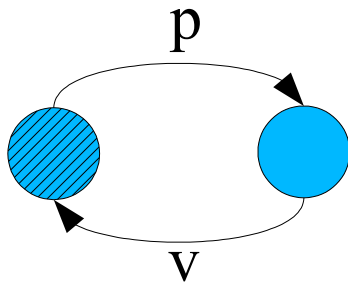
$$S = p.v.S$$



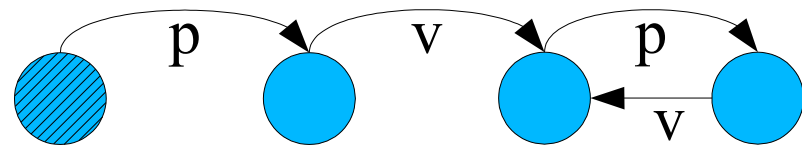
$$P = p.v.S$$

Starke Äquivalenz

Sind S und P stark äquivalent ?



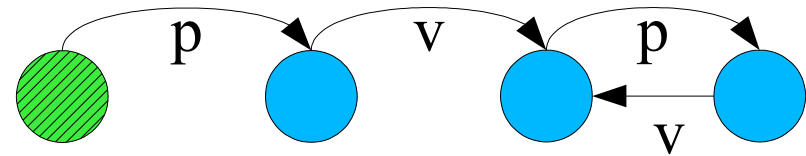
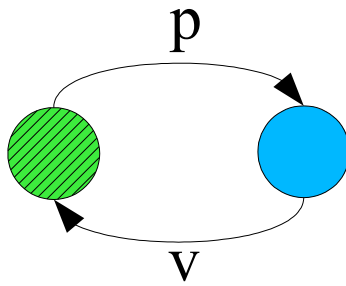
$$S = p.v.S$$



$$P = p.v.S$$

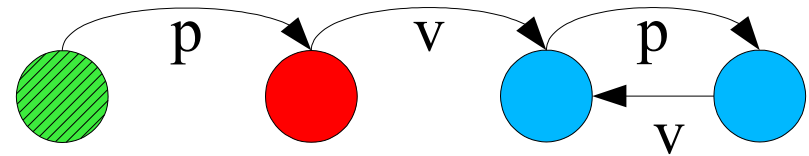
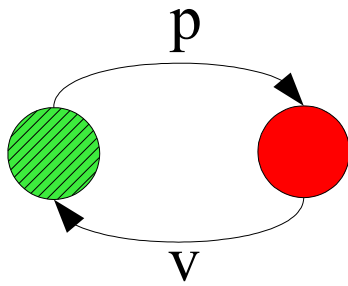
Starke Äquivalenz

Sind S und P stark äquivalent ?

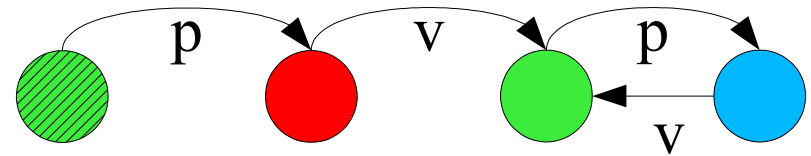
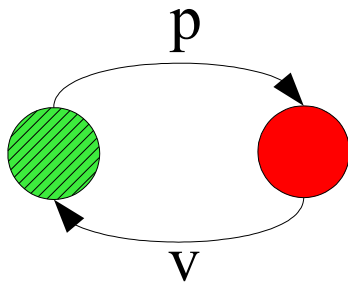


Starke Äquivalenz

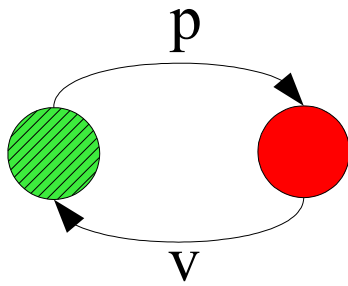
Sind S und P stark äquivalent ?



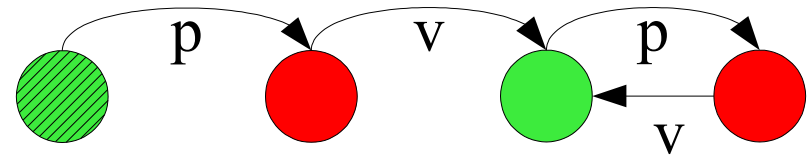
Starke Äquivalenz



Starke Äquivalenz

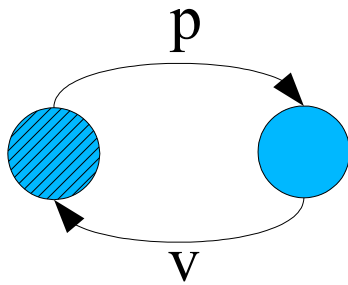


$$S = p.v.S$$

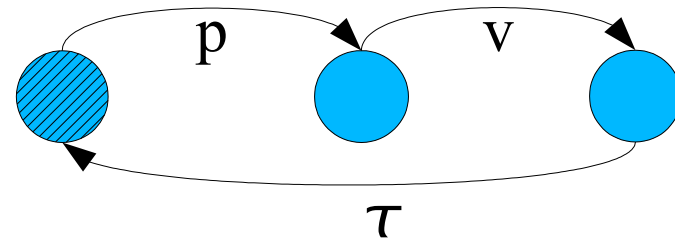


$$P = p.v.S$$

Schwache Äquivalenz



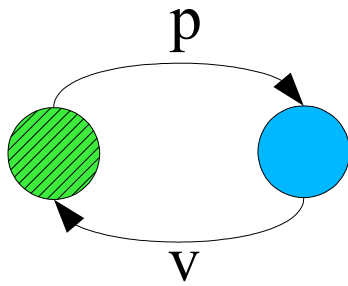
$$S = p.v.S$$



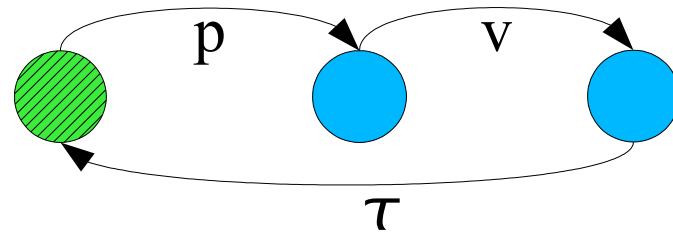
$$Q = p.v.\tau.Q$$

Schwache Äquivalenz

Sind S und Q stark äquivalent ?



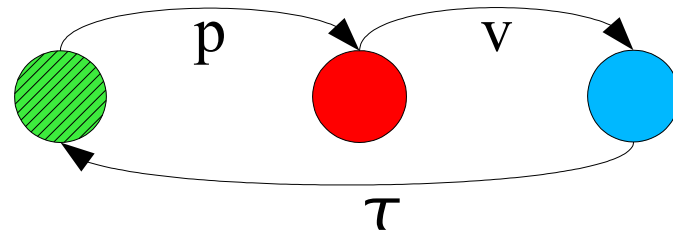
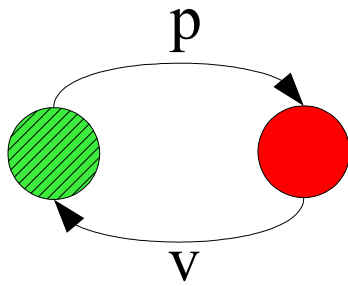
$$S = p.v.S$$



$$Q = p.v.\tau.Q$$

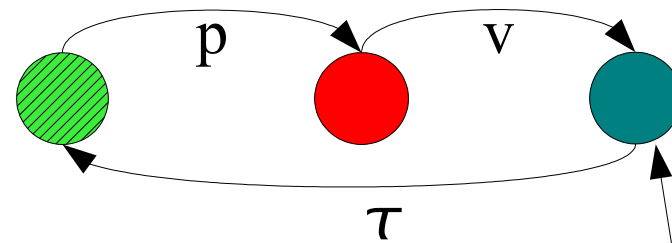
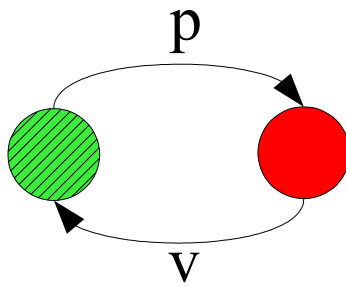
Schwache Äquivalenz

Sind S und Q stark äquivalent ?



Schwache Äquivalenz

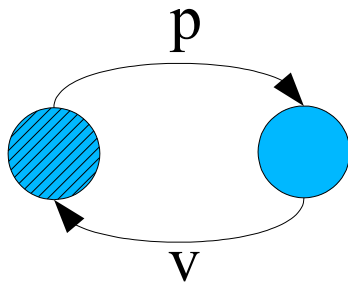
S und Q sind nicht stark äquivalent



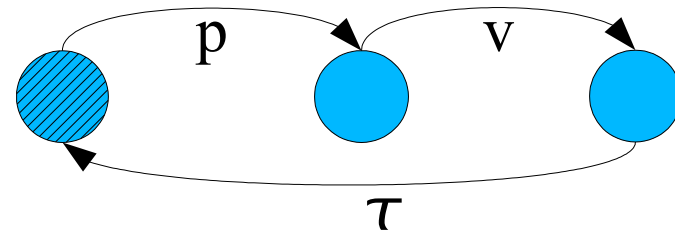
Für diesen Zustand gibt es
keinen entsprechenden im linken System

Schwache Äquivalenz

die Übergänge die man von Außen beobachten kann
sind in beiden Systemen die gleichen



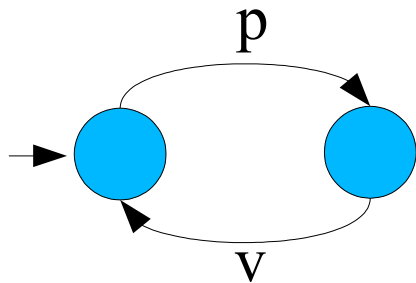
$$S = p.v.S$$



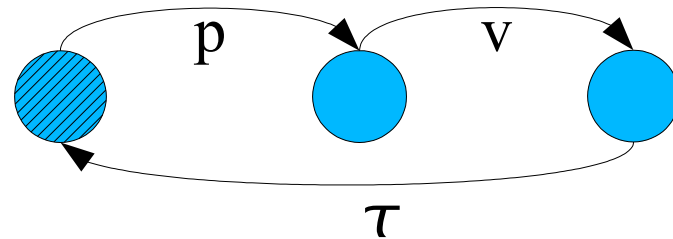
$$Q = p.v.\tau.Q$$

Schwache Äquivalenz

die Übergänge die man von Außen beobachten kann
sind in beiden Systemen die gleichen



$$S = p.v.S$$



$$Q = p.v.τ.Q$$

Deshalb wäre es sinnvoll S und Q ebenfalls als äquivalent anzusehen.

Schwache Äquivalenz

$$\textit{System} \approx \textit{Spezifikation}$$

Die Spezifikation wird durch einen Prozess dargestellt, der nur das Verhalten nach Außen darstellt, die Implementierungsdetails dabei vernachlässigt.

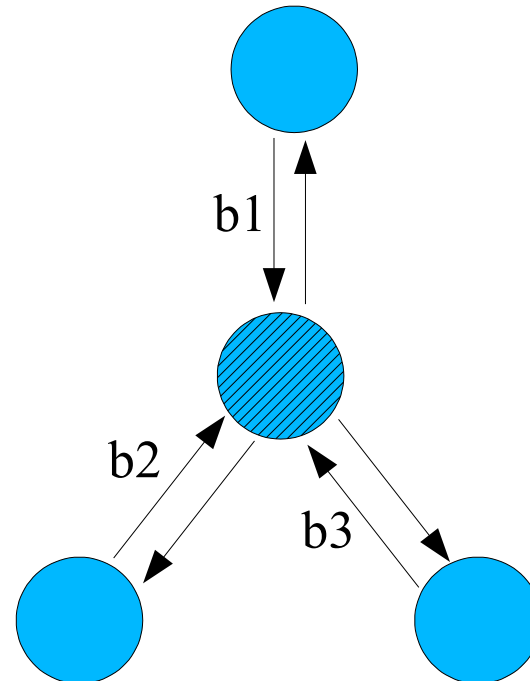
Das System stellt dann die konkrete Implementierung da.

Man möchte jetzt beweisen, dass sich System und Spezifikation äquivalent verhalten

Beispiel: Lottery

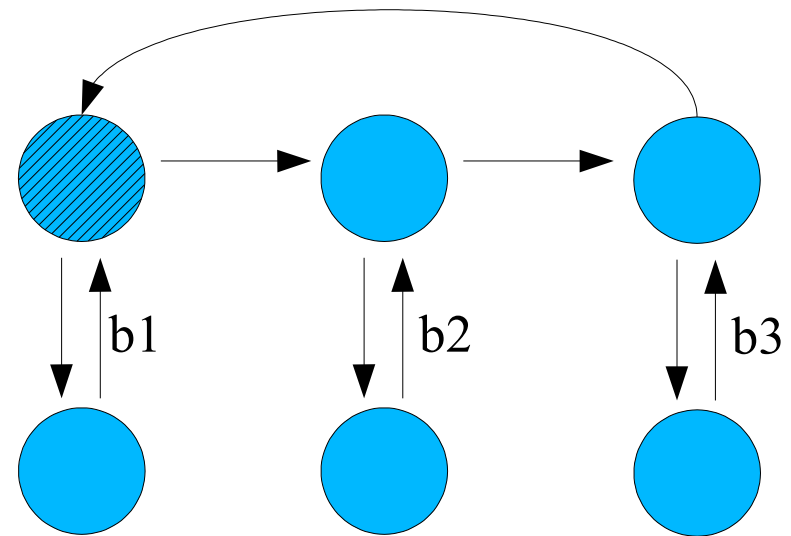
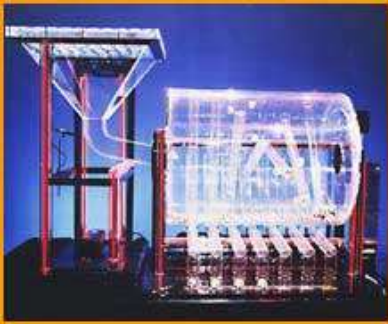


$$Lotspec = \tau \cdot b_1 \cdot Lotspec + \dots + \tau \cdot b_n \cdot Lotspec$$



Spezifikation einer Lotto-Trommel

Beispiel: Lottery

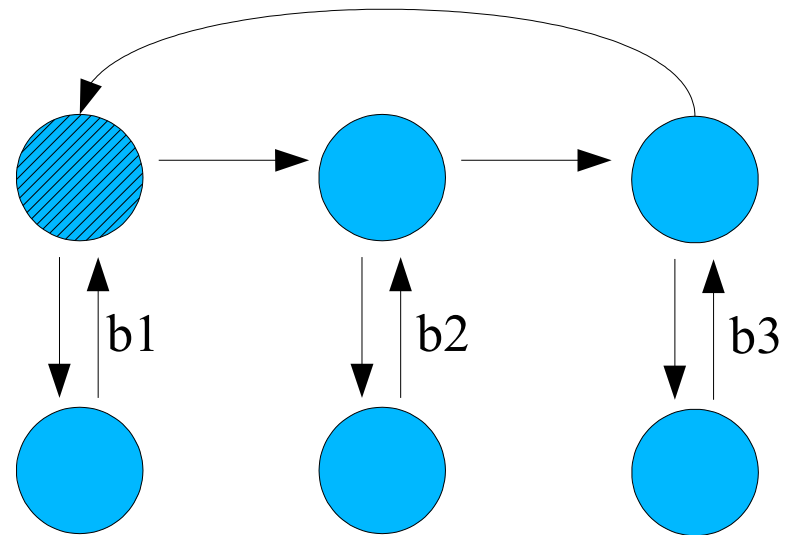
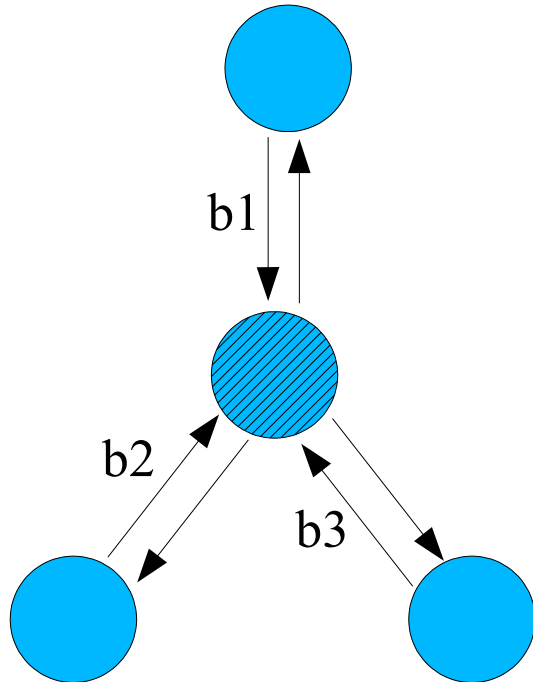


Implementierung einer Lotto Trommel

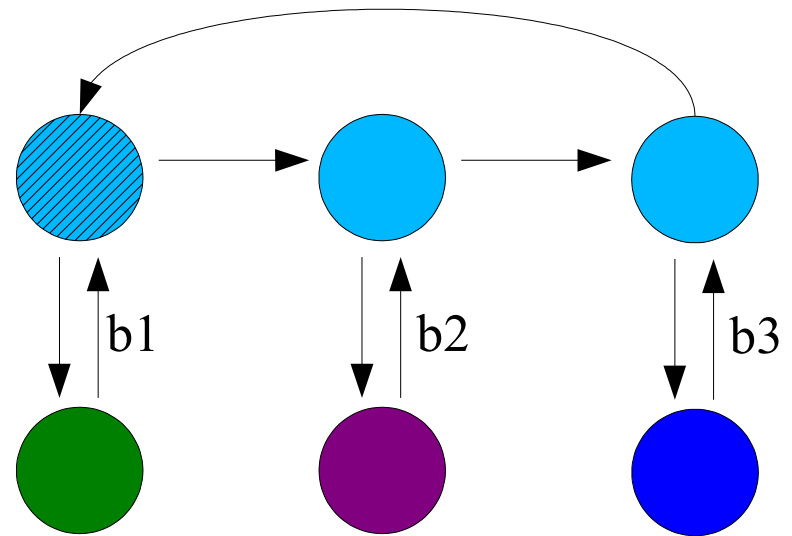
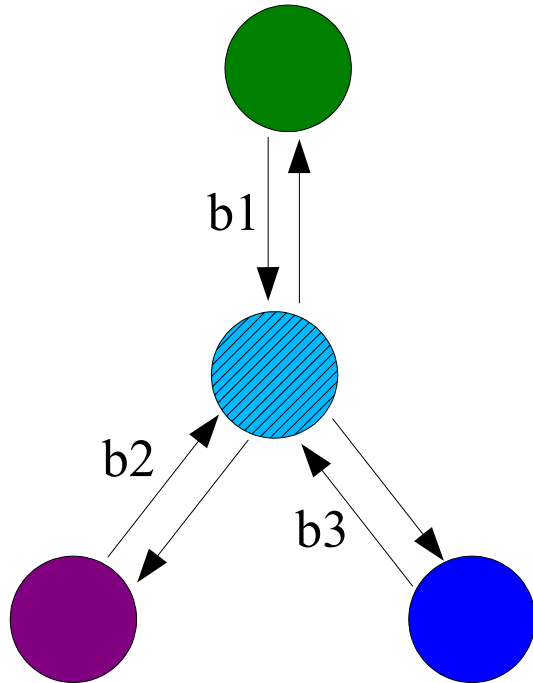
Beispiel: Lottery

Erfüllt Implementierung die Spezifikation?

System \approx Spezifikation

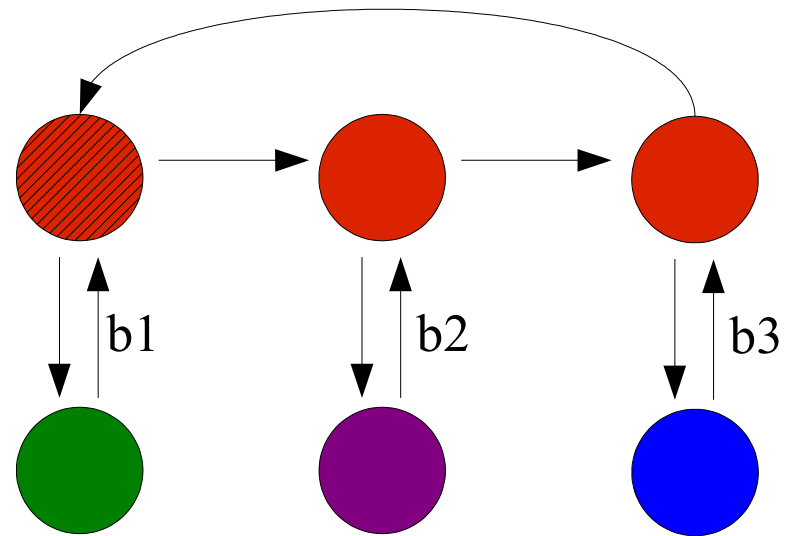
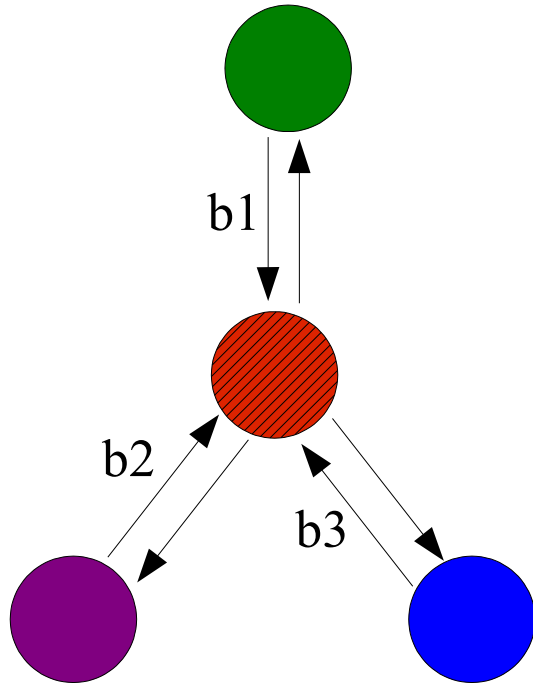


Beispiel: Lottery



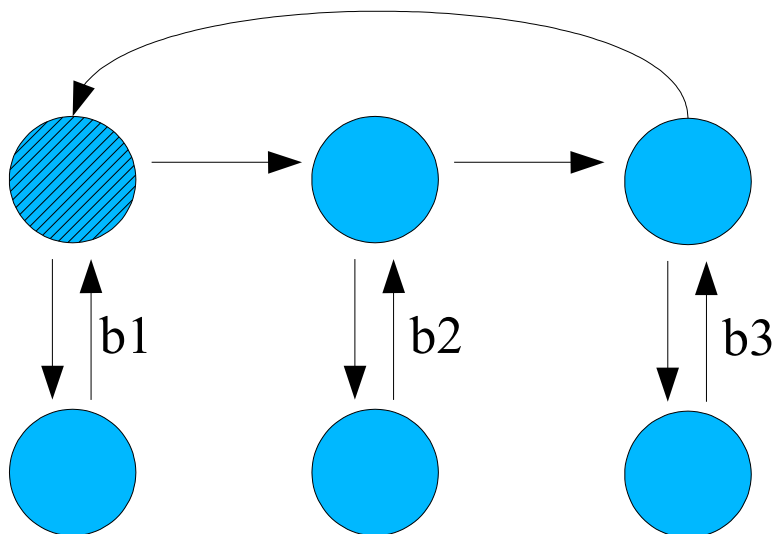
Beispiel: Lottery

Implementierung erfüllt die Spezifikation



Fairness

Die Implementierung beinhaltet eine mögliche Divergenz, d.h. die Trommel könnte sich unendlich lange drehen ohne eine Kugel auszuwerfen. Die Spezifikation hingegen kann nicht divergieren.



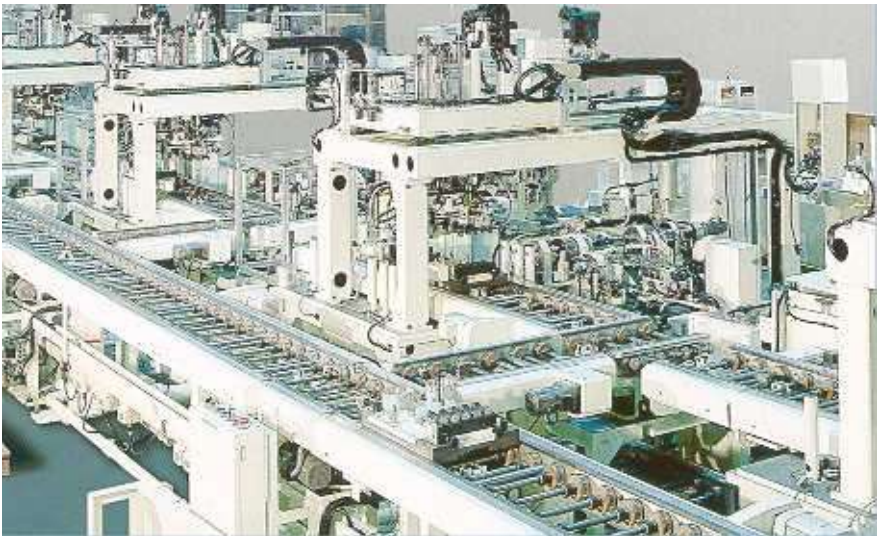
Die Äquivalenz der beiden Systeme, beruht auf der Fairness-Annahme, dass keine Transition unendlich oft vermieden wird.

Unter dieser Annahme sind die beiden Systeme äquivalent

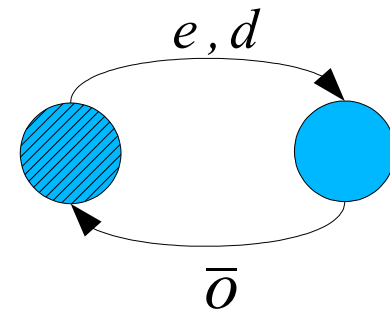
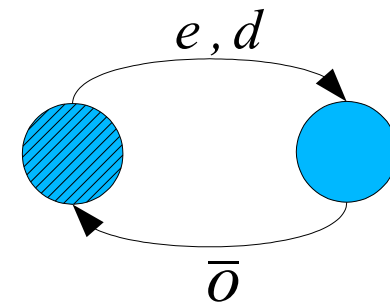
Schwache Äquivalenz

“Schwache Äquivalenz” oder “Beobachtbare Äquivalenz” macht eine Aussage darüber, ob sich zwei Systeme nach Außen hin gleich verhalten - für den Benutzer (Beobachter) gleiches Verhalten zeigen. Dies geschieht immer unter der Annahme, dass sich beide Systeme “fair” verhalten.

Beispiel: Job Shop



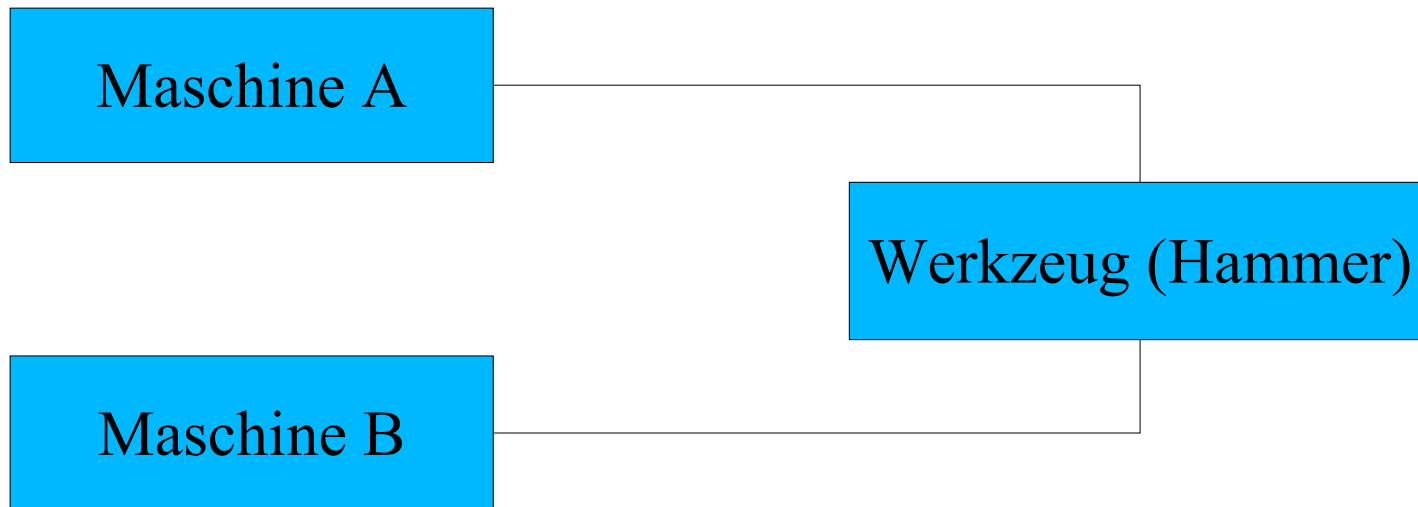
A | A



Spezifikation zweier paralleler
Maschinen

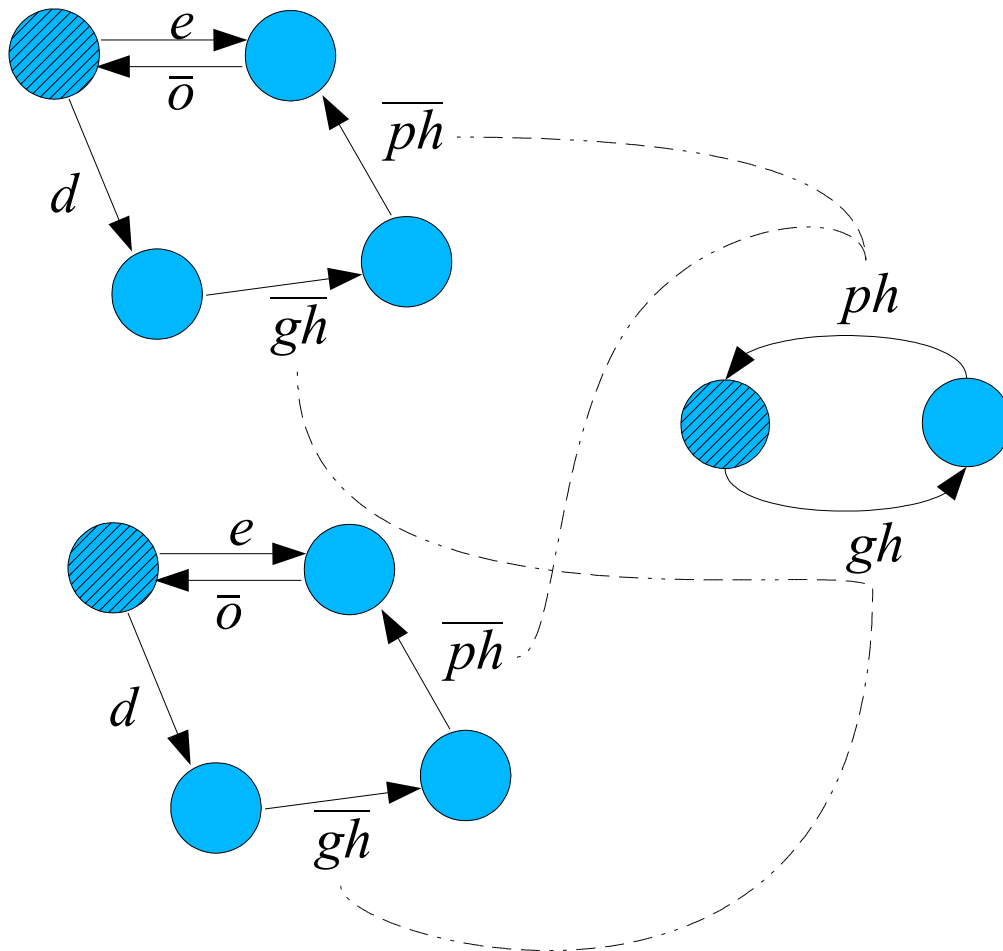
Beispiel: Job Shop

Implementierung



Beispiel: Job Shop

Implementierung

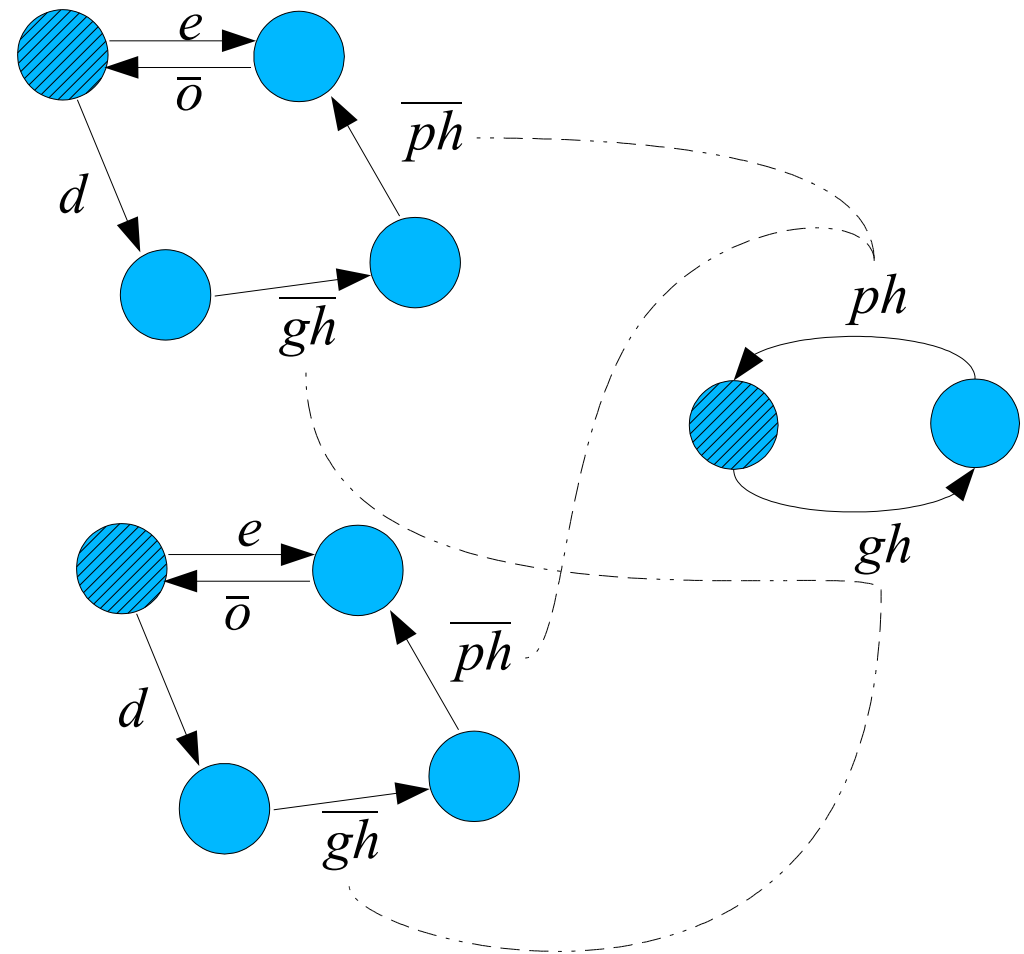
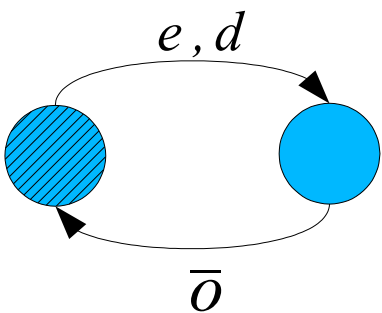
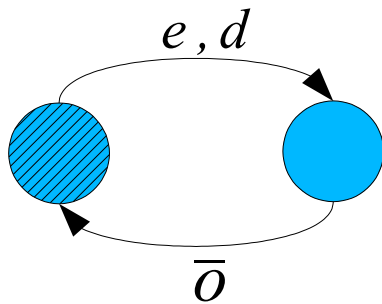


$$J = e.\bar{o}.J + d.\overline{gh}.\overline{ph}.\bar{o}.J$$

$$H = gh.ph.H$$

Jobshop = new ph gh (J|J|H)

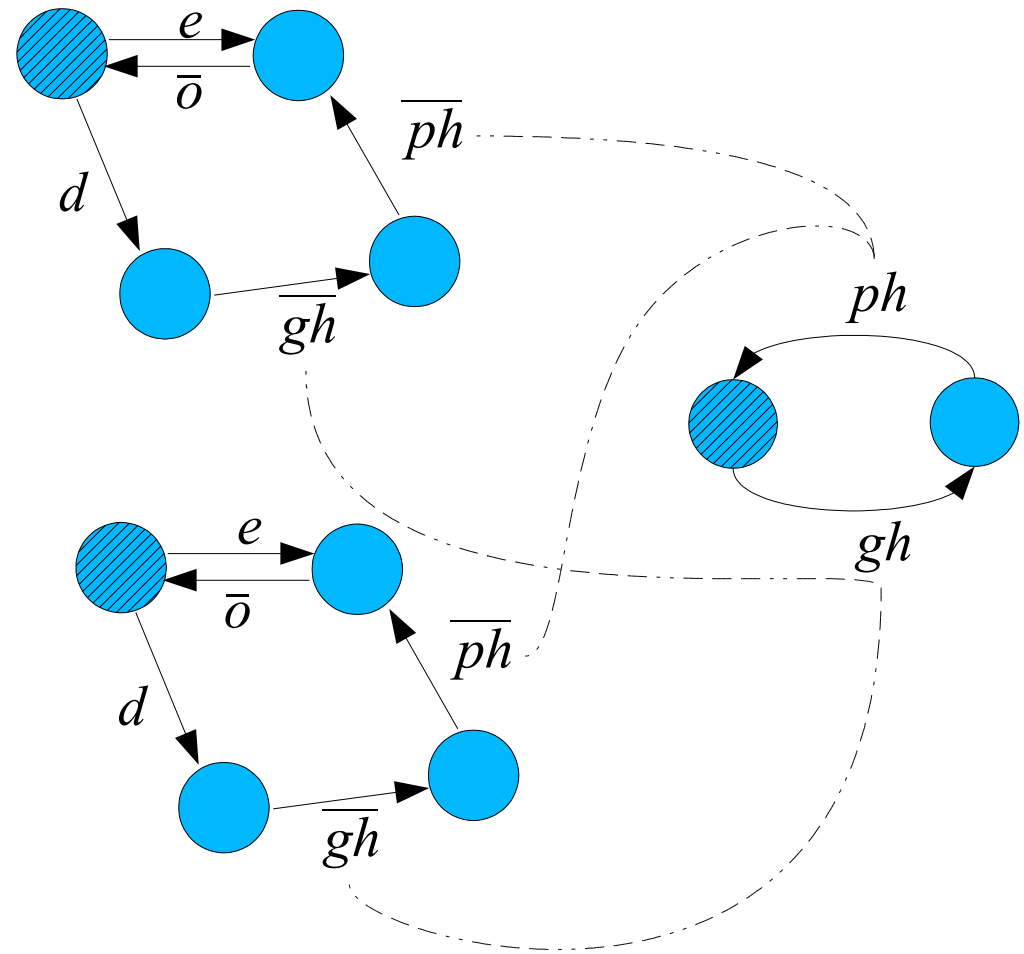
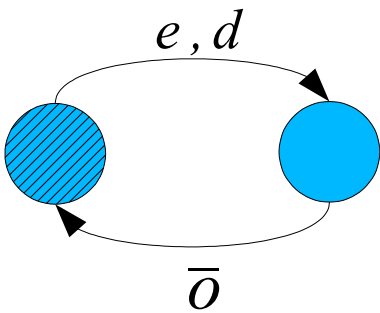
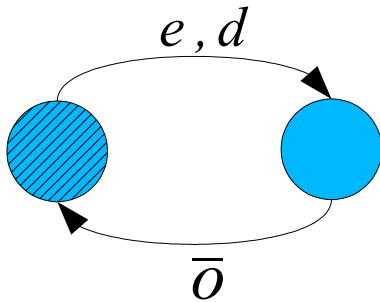
Beispiel: Job Shop



Erfüllt Implementierung die Spezifikation?

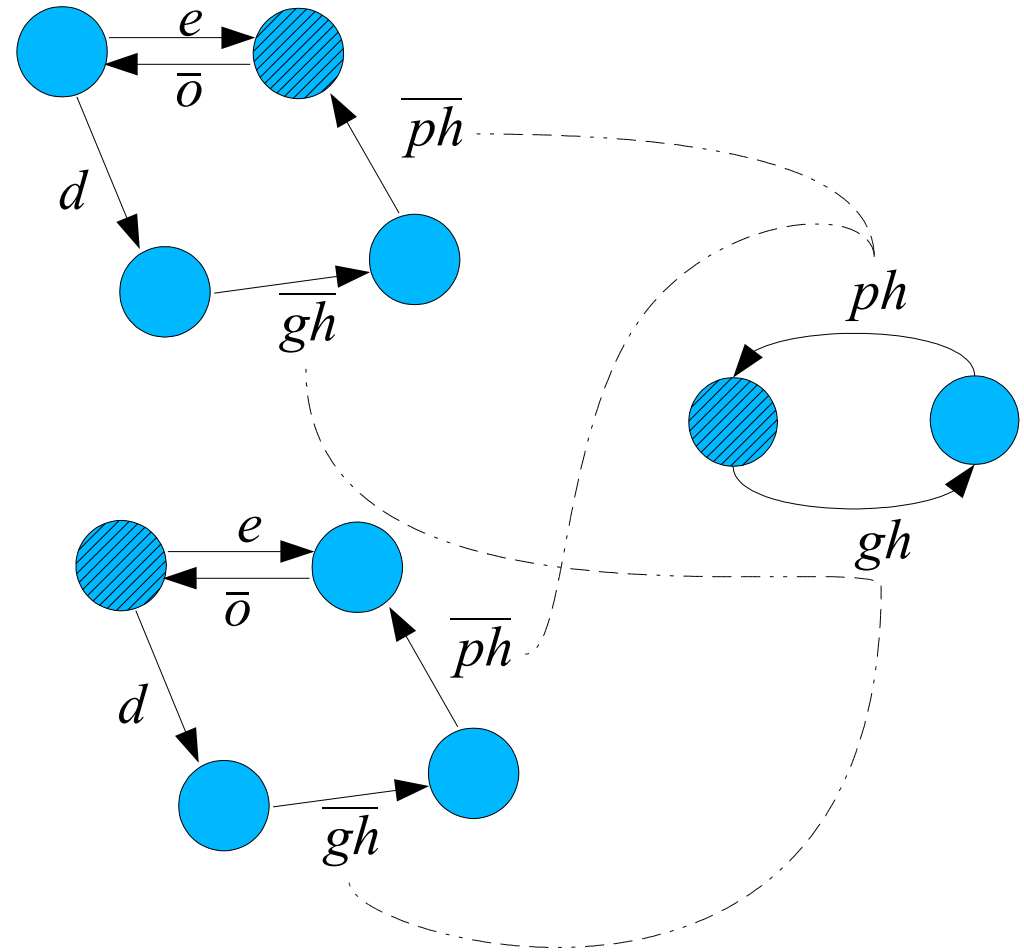
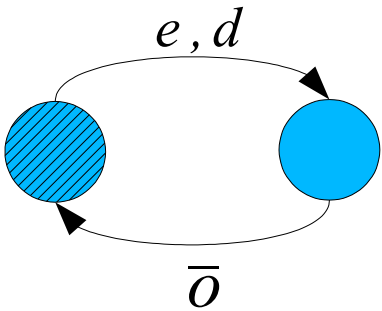
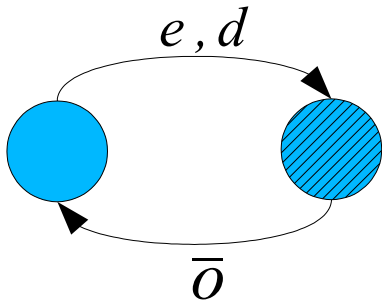
Beispiel: Job Shop

input: $e e e d d$



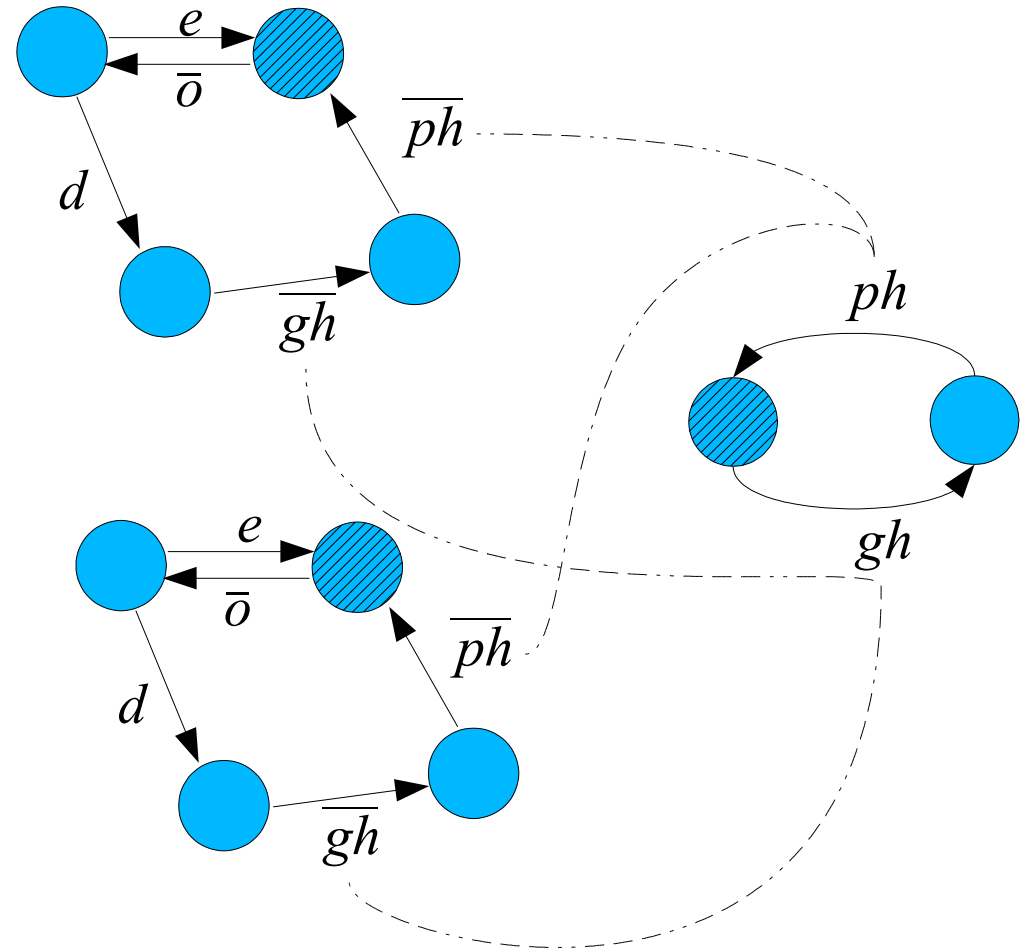
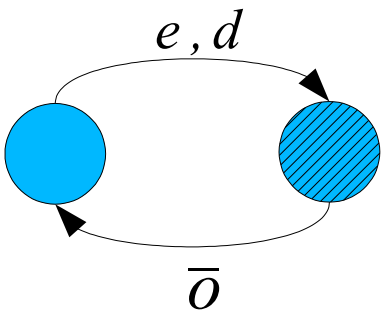
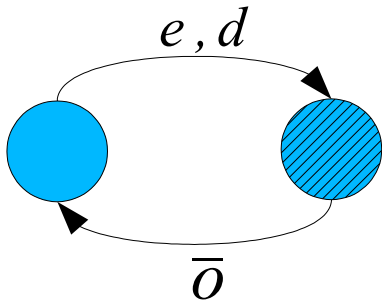
Beispiel: Job Shop

input: e e e d d



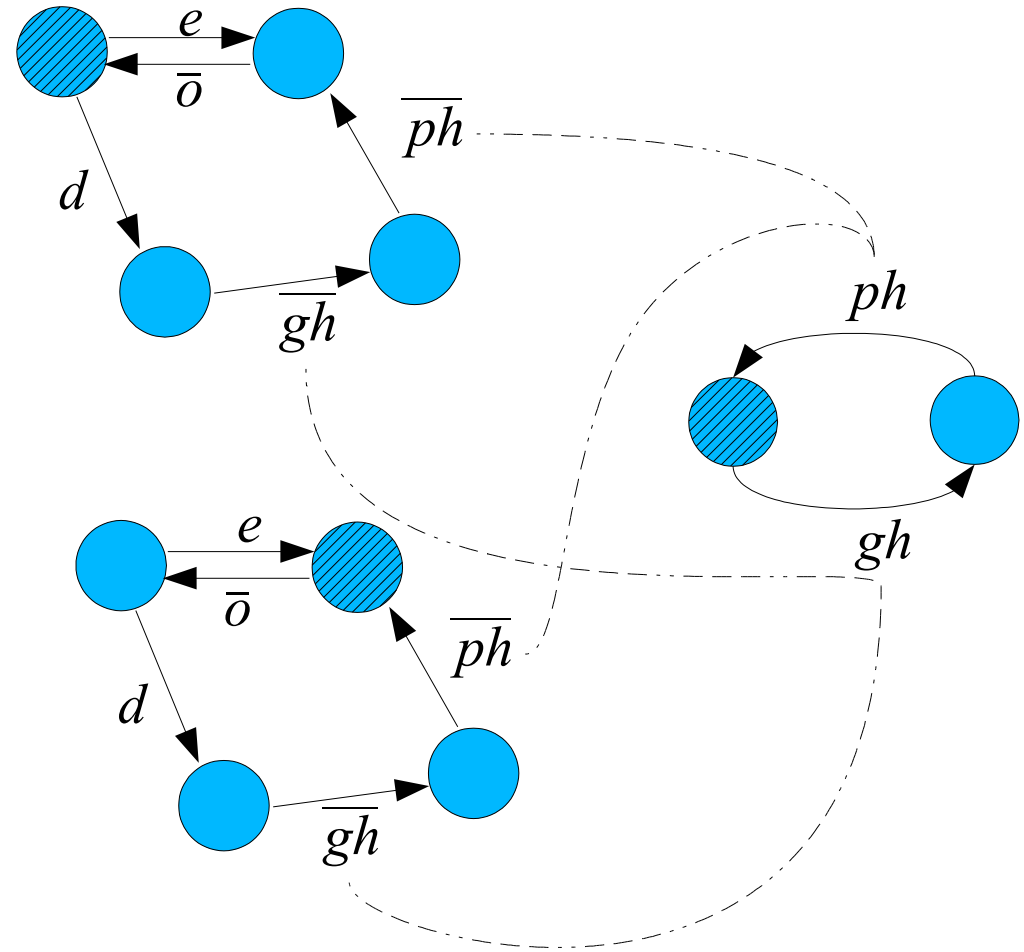
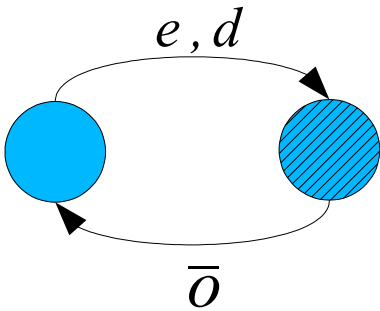
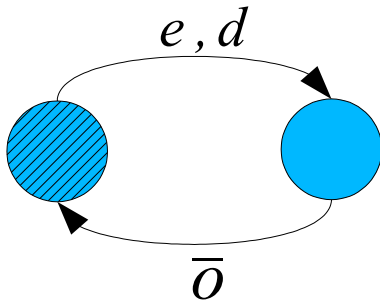
Beispiel: Job Shop

input: e e e d d



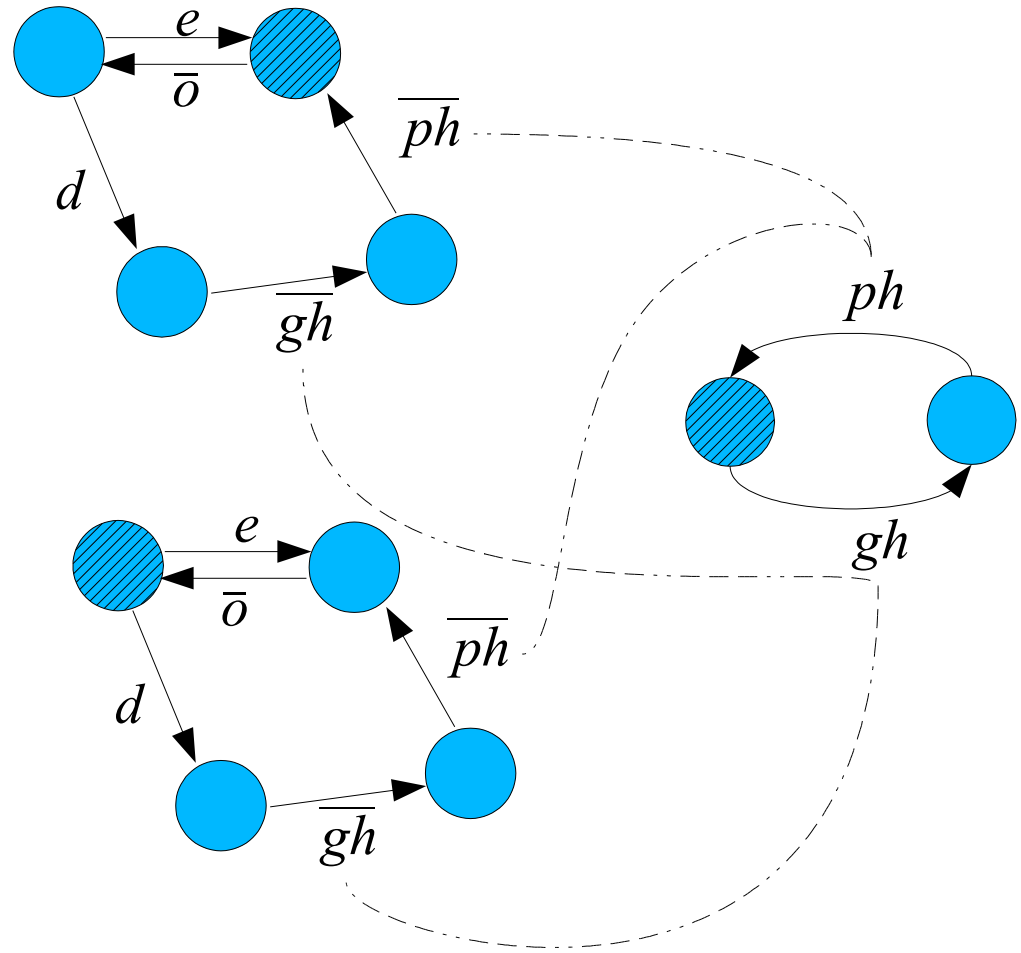
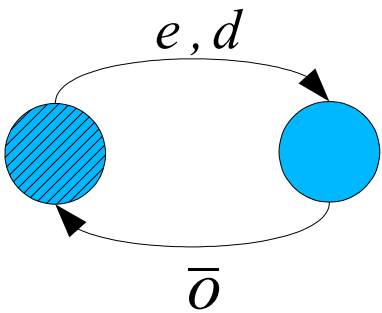
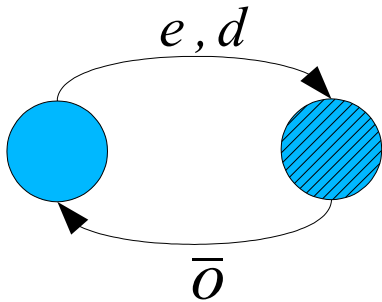
Beispiel: Job Shop

input: e e e d d



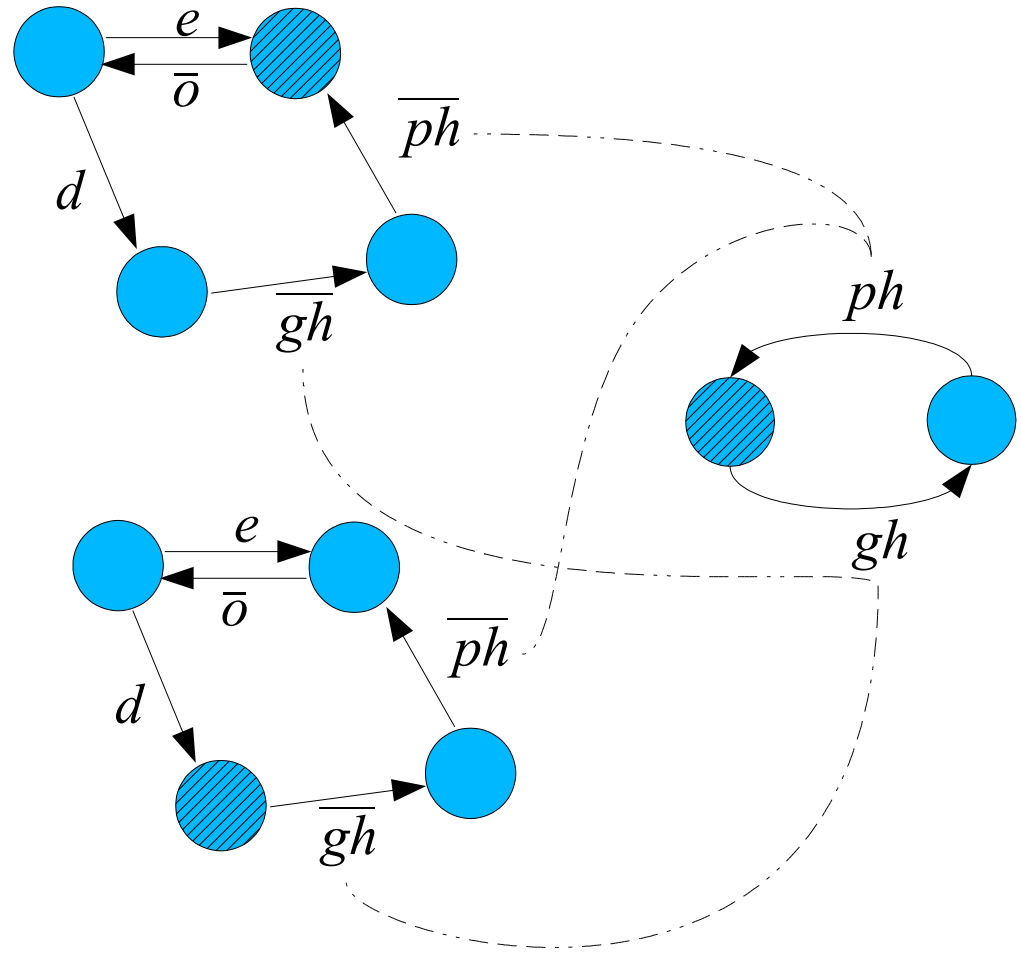
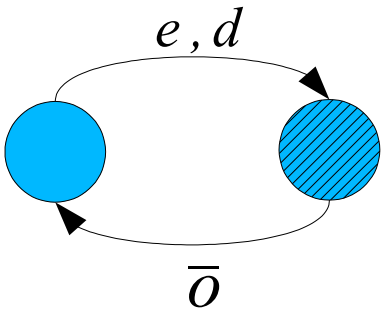
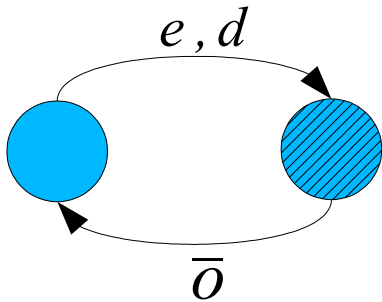
Beispiel: Job Shop

input: e e e d d



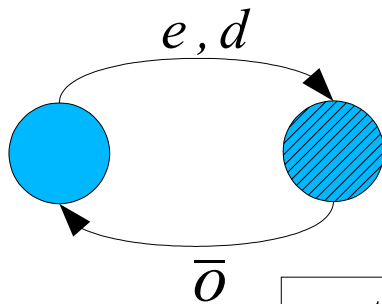
Beispiel: Job Shop

input: e e e d d

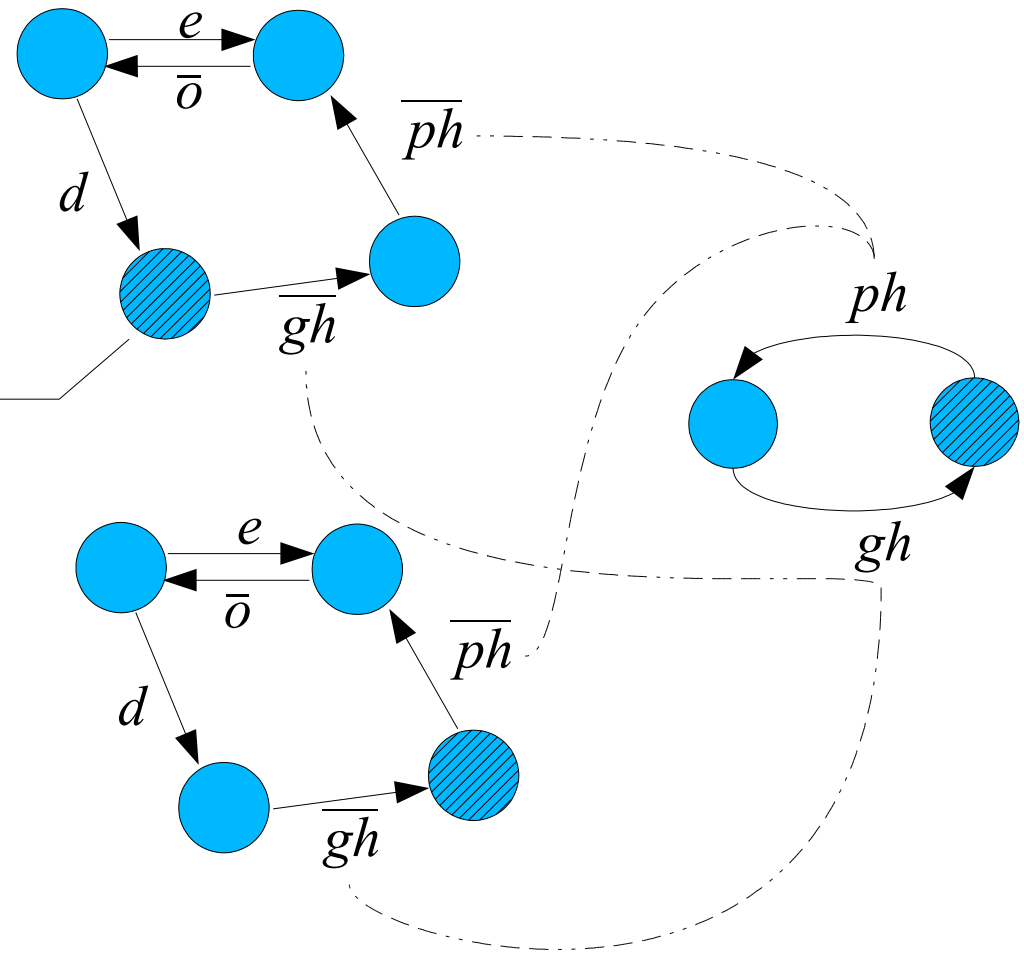
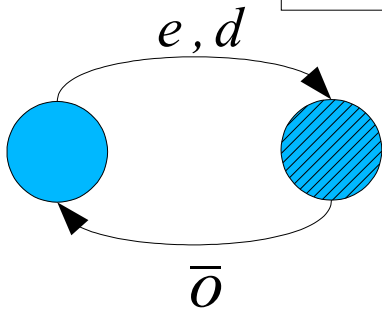


Beispiel: Job Shop

input: e e e d d

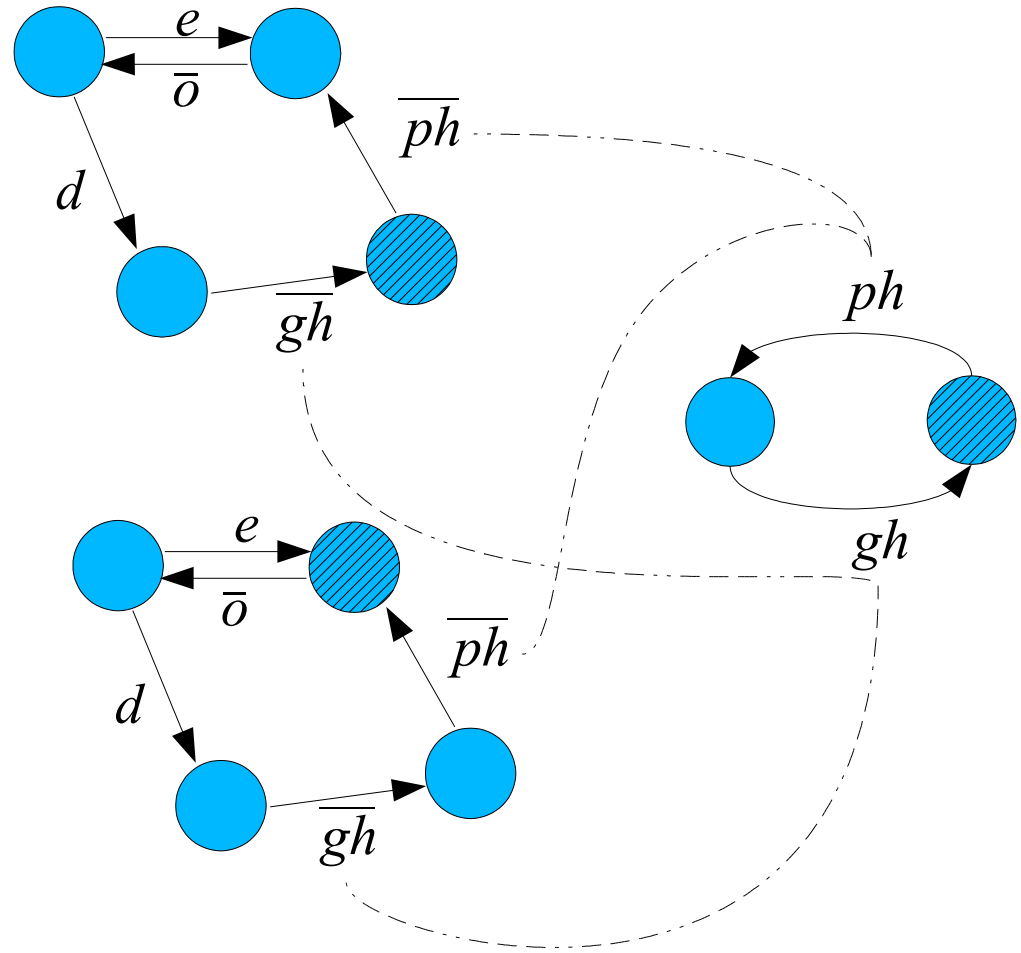
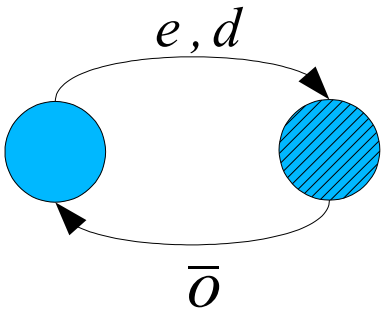
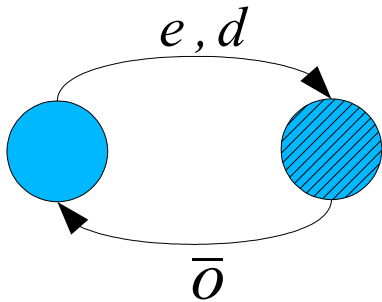


wartet auf Hammer



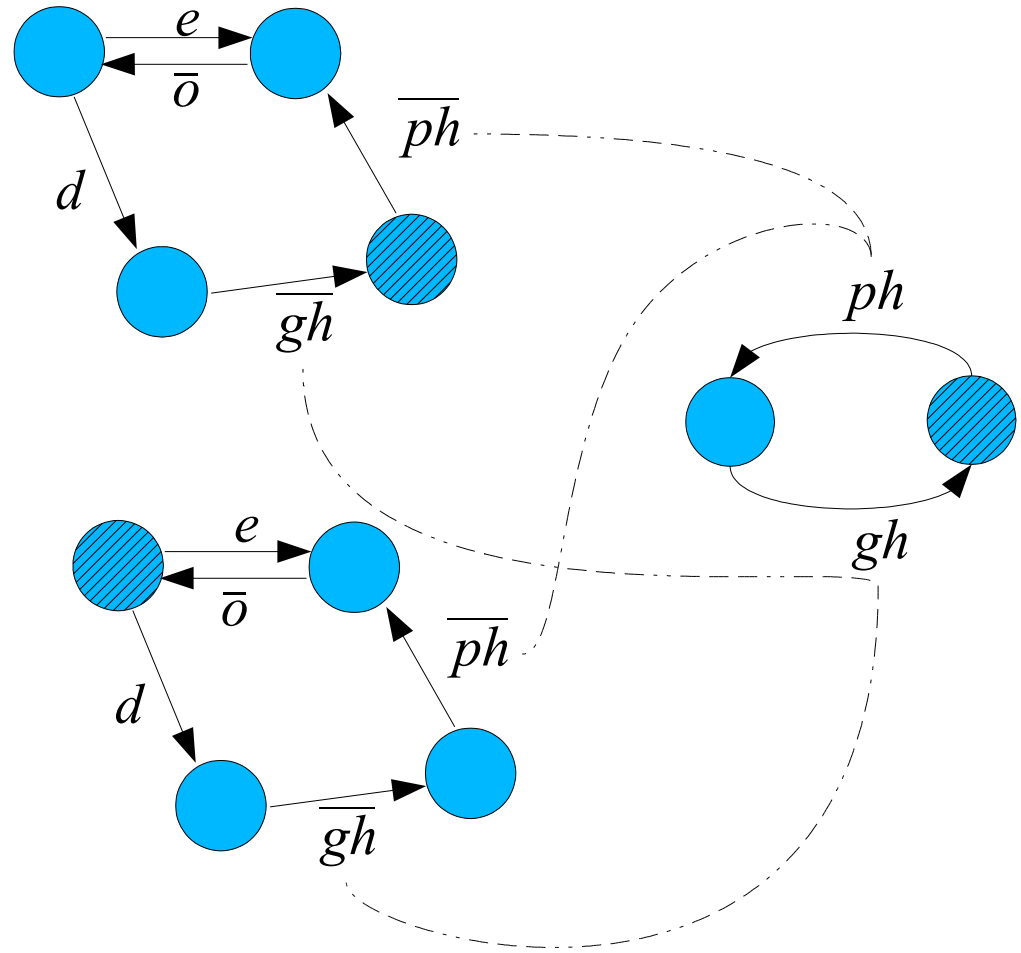
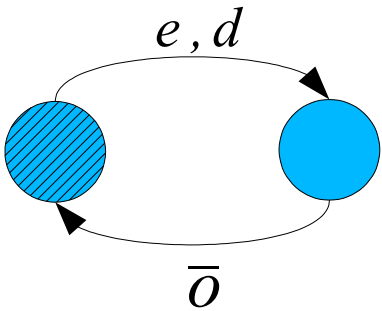
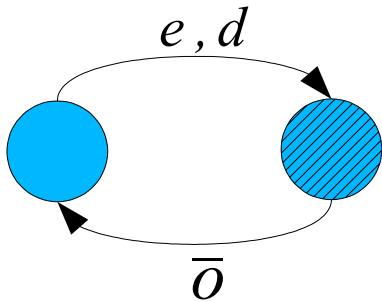
Beispiel: Job Shop

input: e e e d d



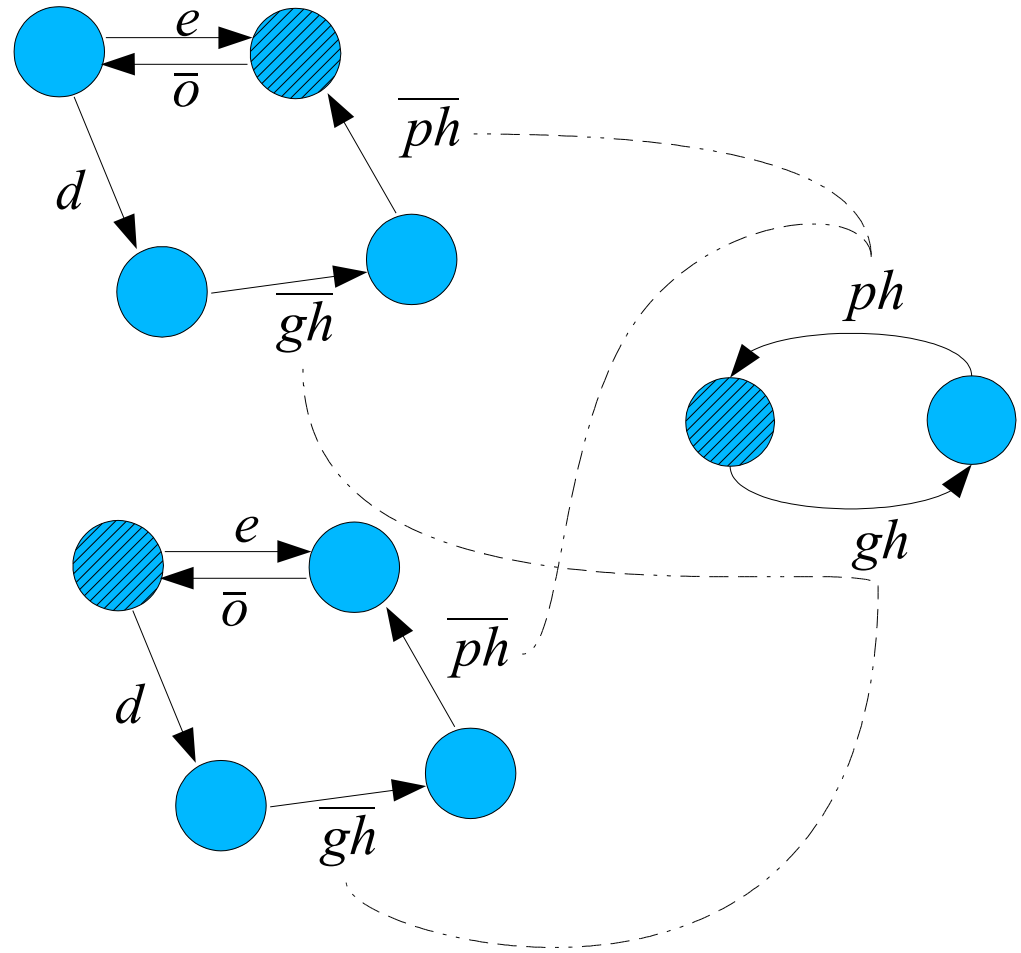
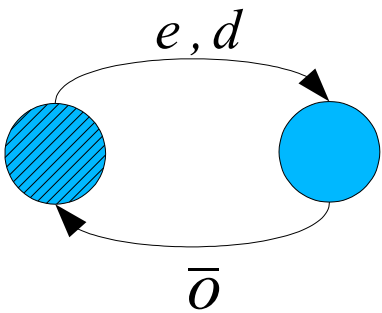
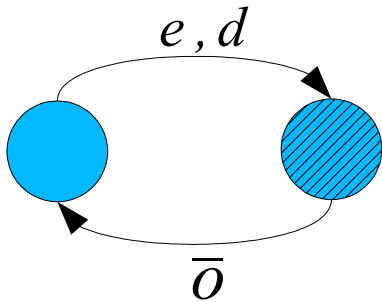
Beispiel: Job Shop

input: e e e d d



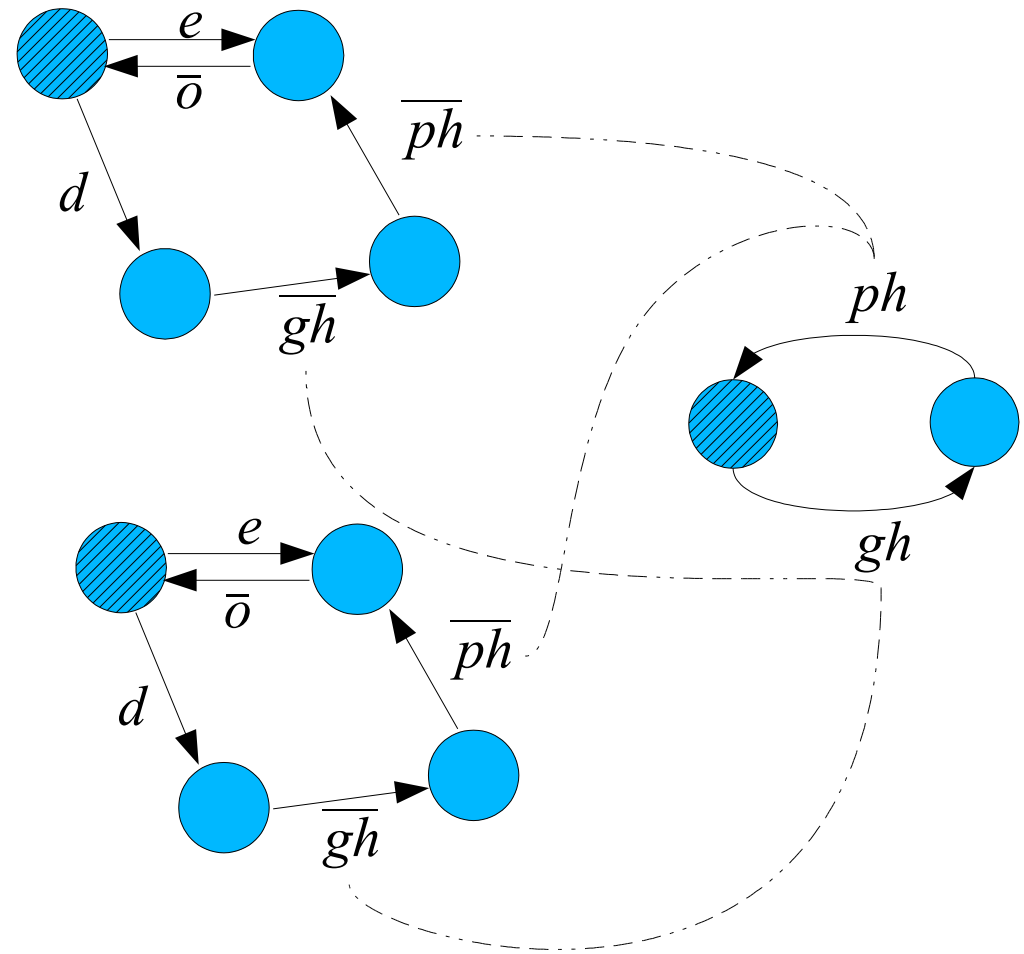
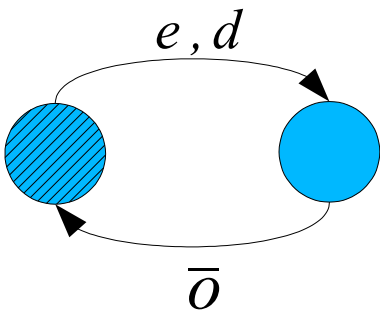
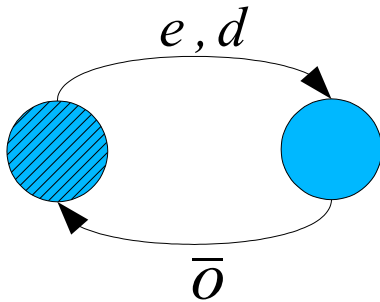
Beispiel: Job Shop

input: e e e d d

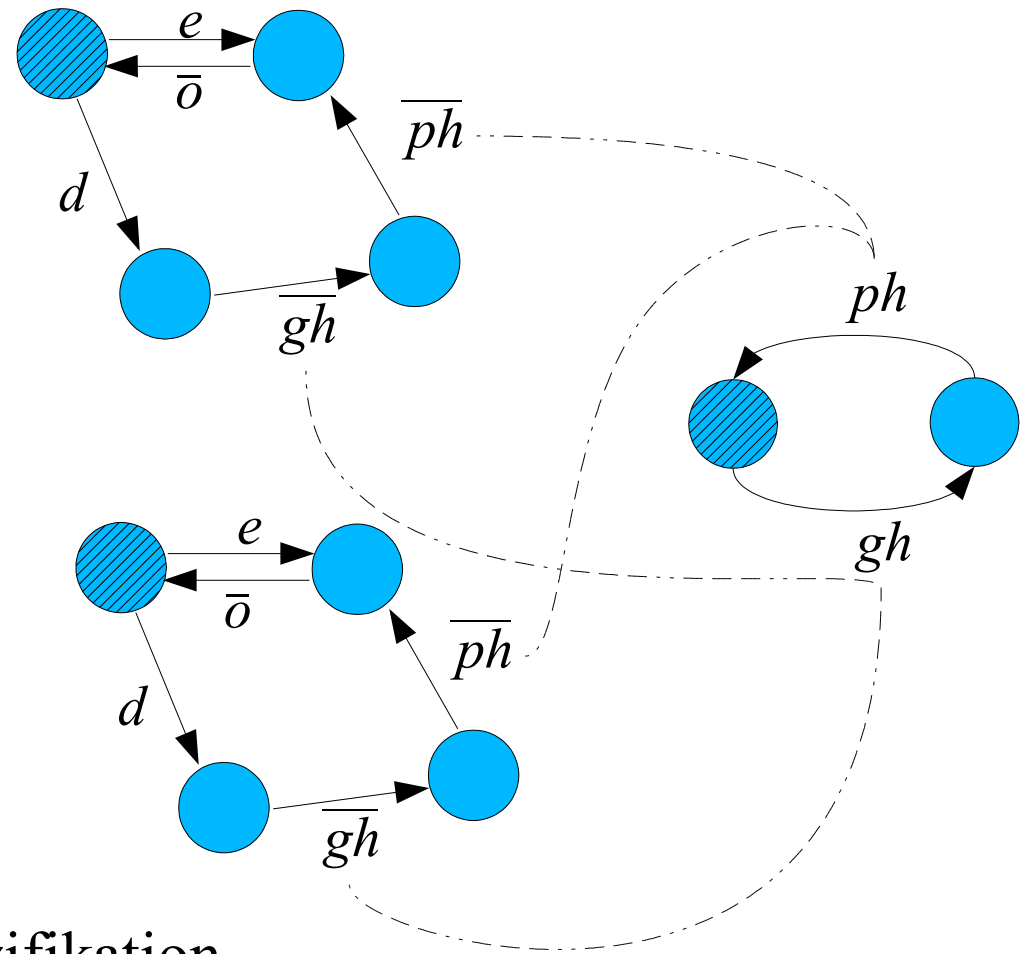
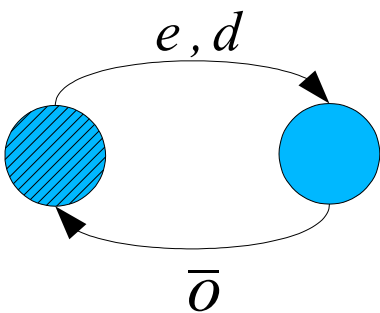
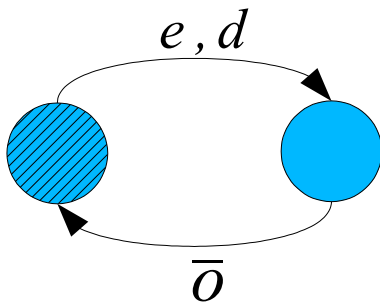


Beispiel: Job Shop

input: e e e d d

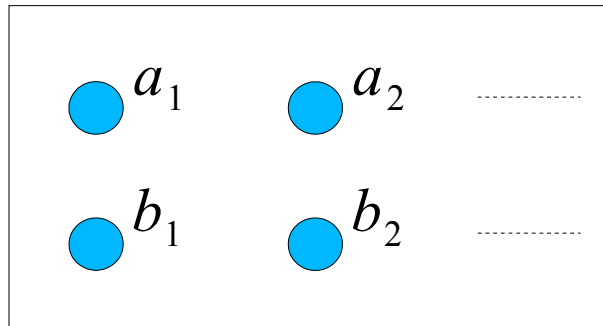


Beispiel: Job Shop



Implementierung erfüllt Spezifikation

Beispiel: Scheduler



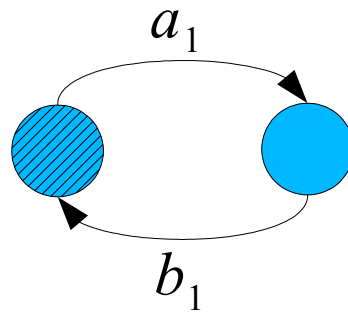
Starte job mit Drücken der Taste a_1 oder a_2

Beende Job mit Drücken der Taste b

a_1 und a_2 können jeweils nur abwechselnd gedrückt werden und nur wenn der Job mit einer b Taste beendet wurde.

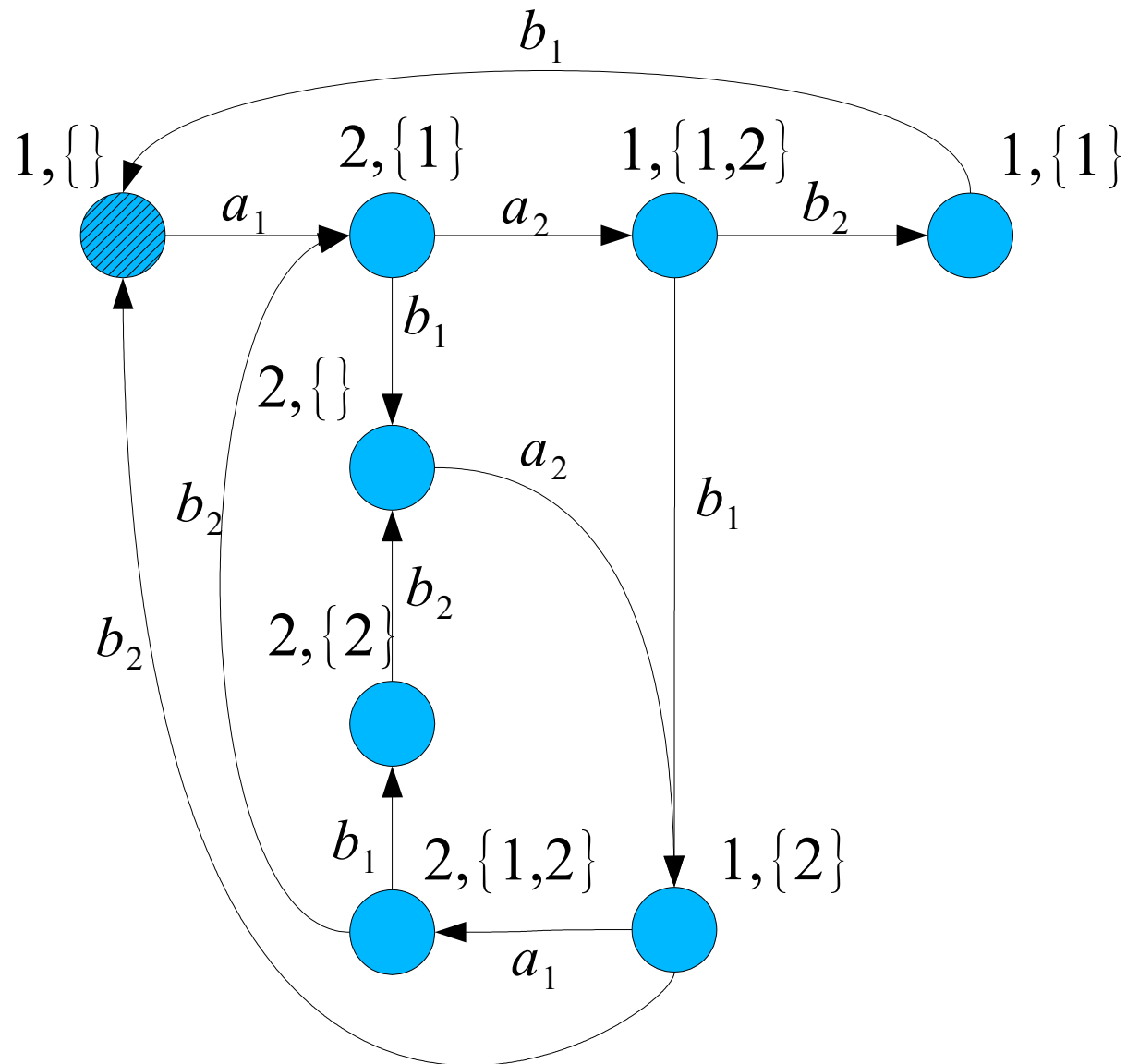
Beispiel: Scheduler

Spezifikation für $n = 1$



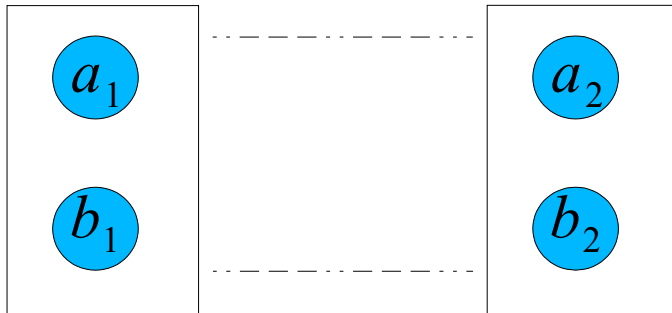
Beispiel: Scheduler

Spezifikation $n = 2$



Beispiel: Scheduler

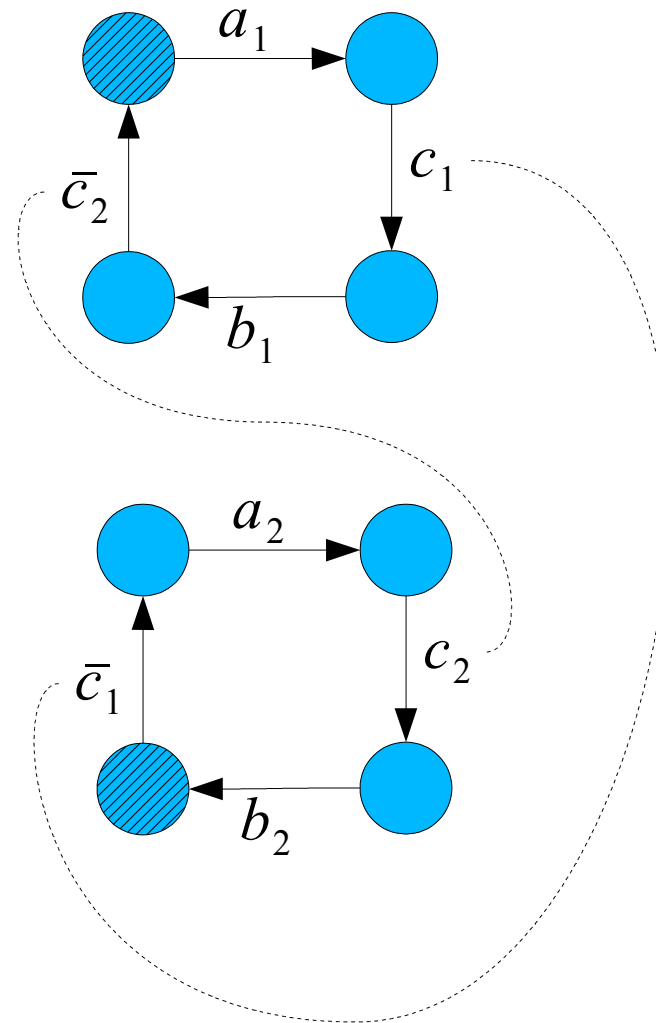
Implementierung für $n=2$:



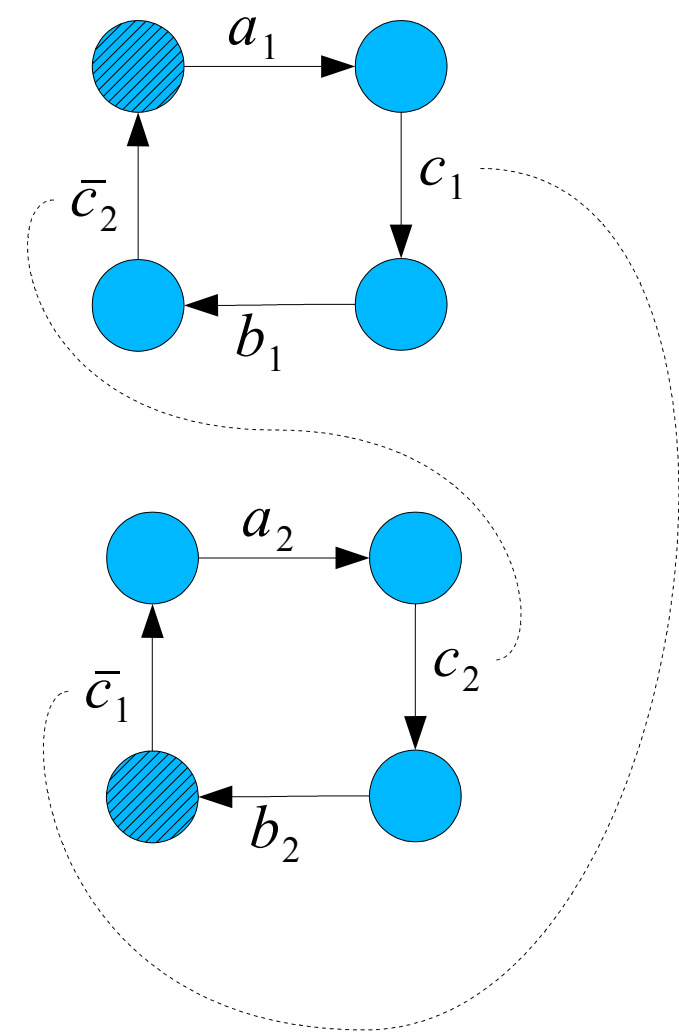
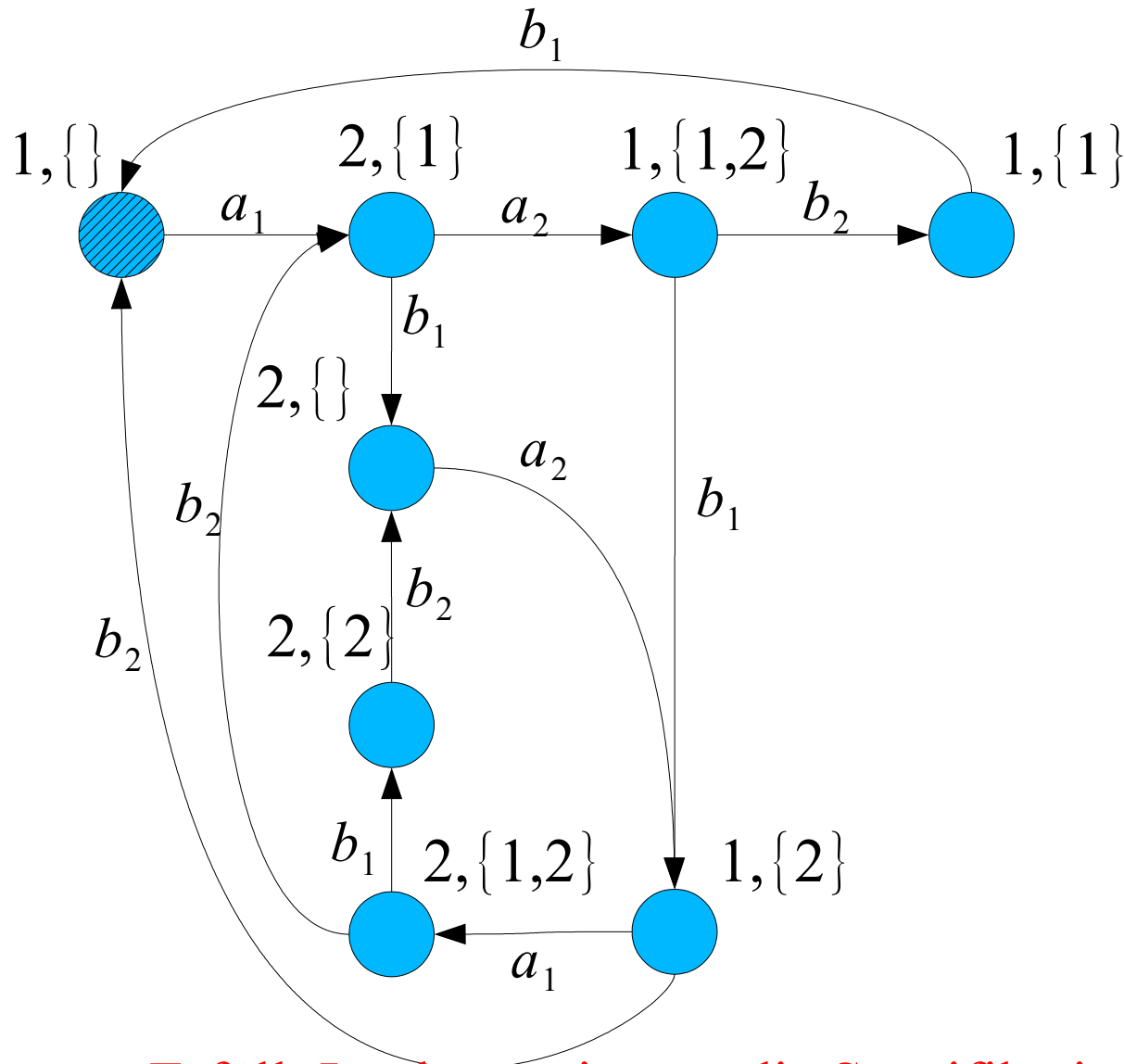
Zwei Prozesse, die durch “Leitungen” miteinander verbunden sind.

Beispiel: Scheduler

Implementierung für $n=2$:

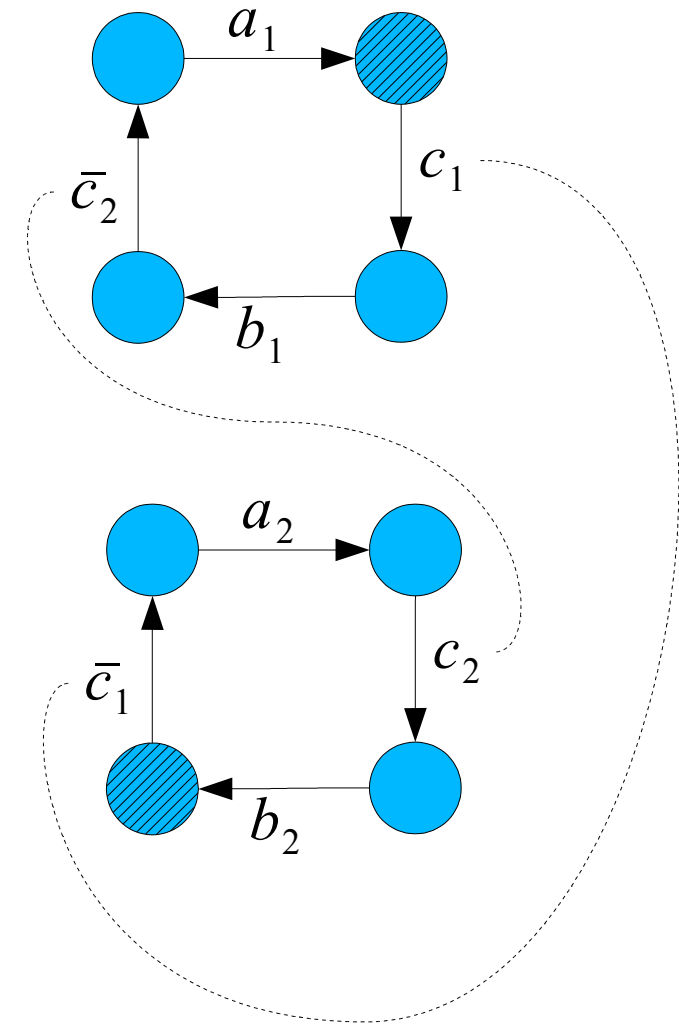
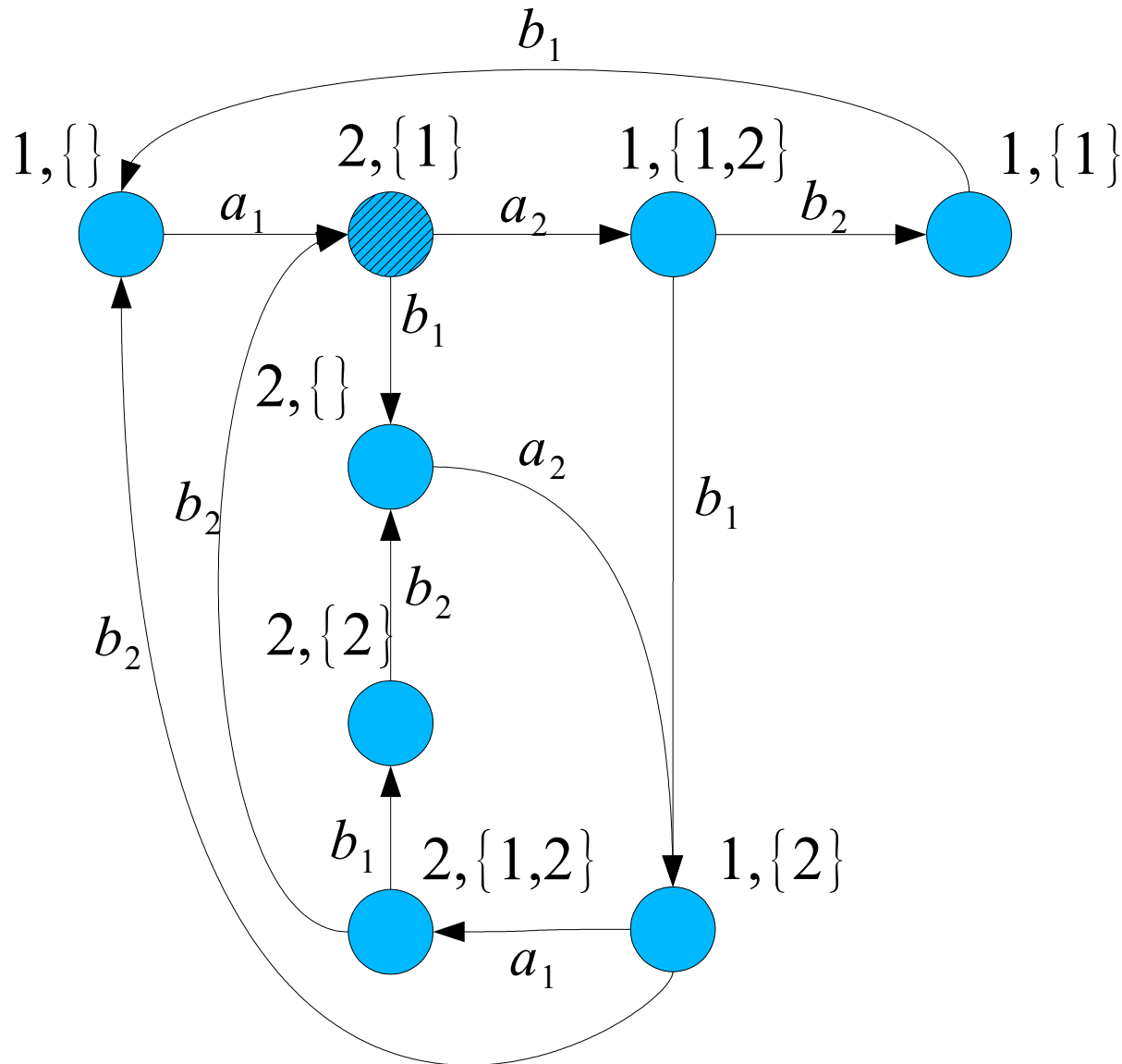


Beispiel: Scheduler

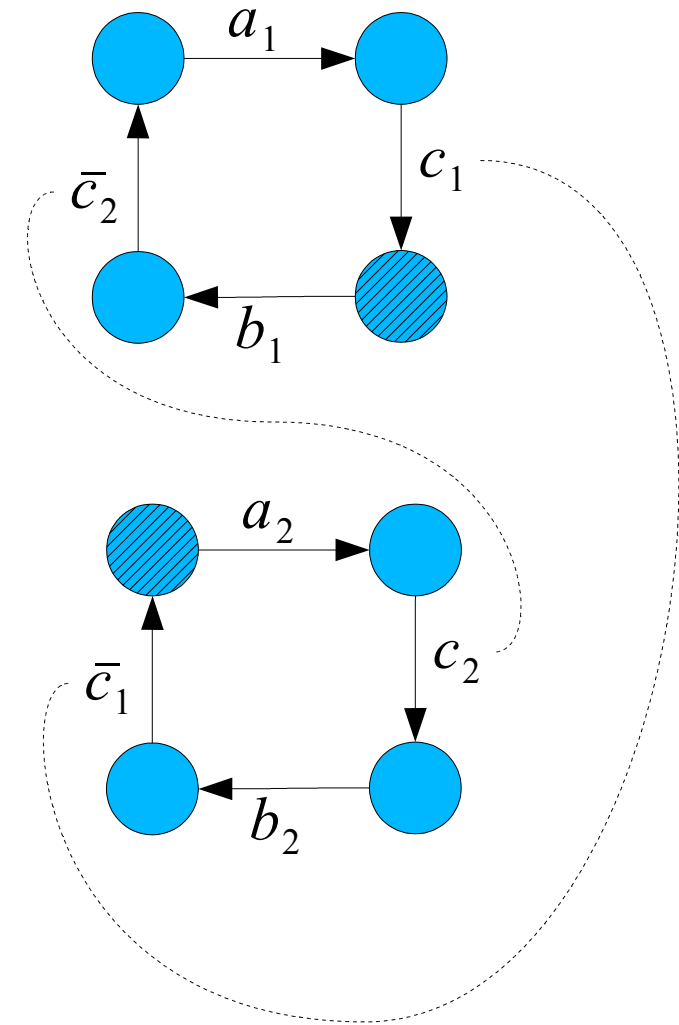
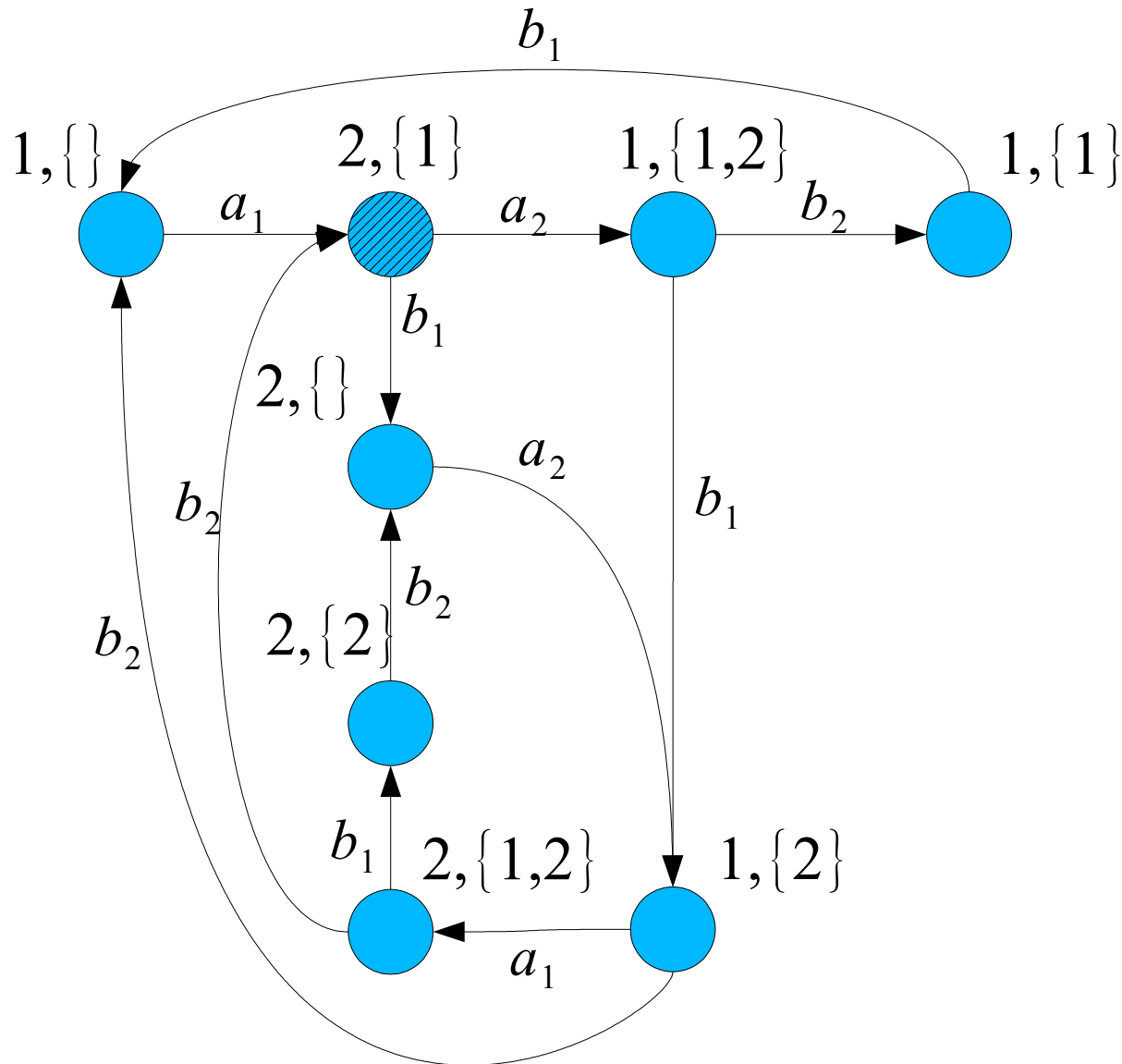


Erfüllt Implementierung die Spezifikation?

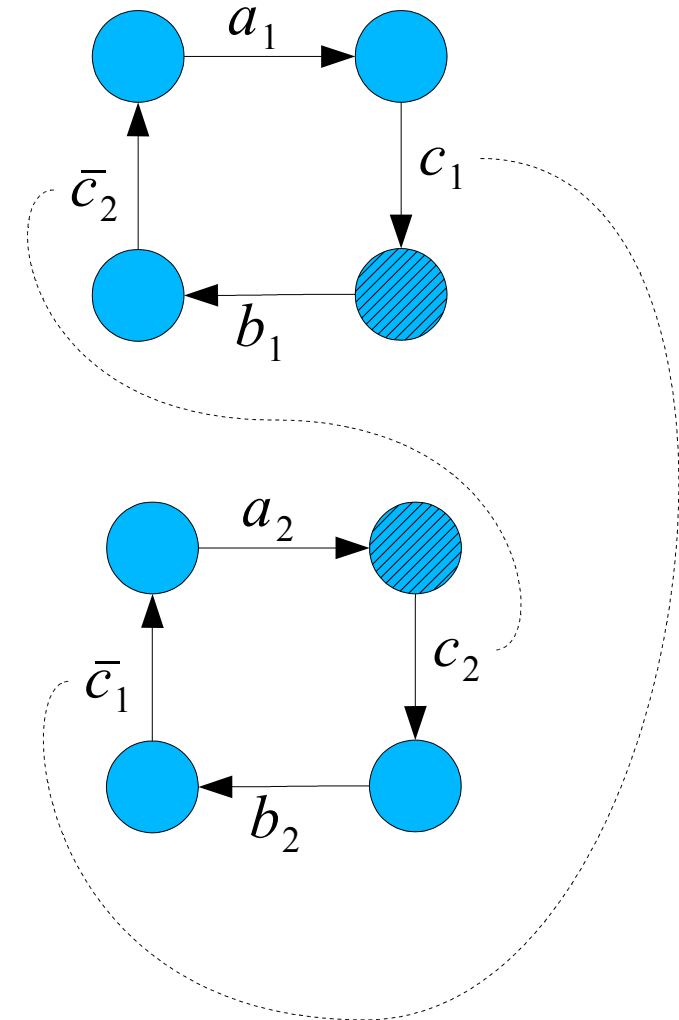
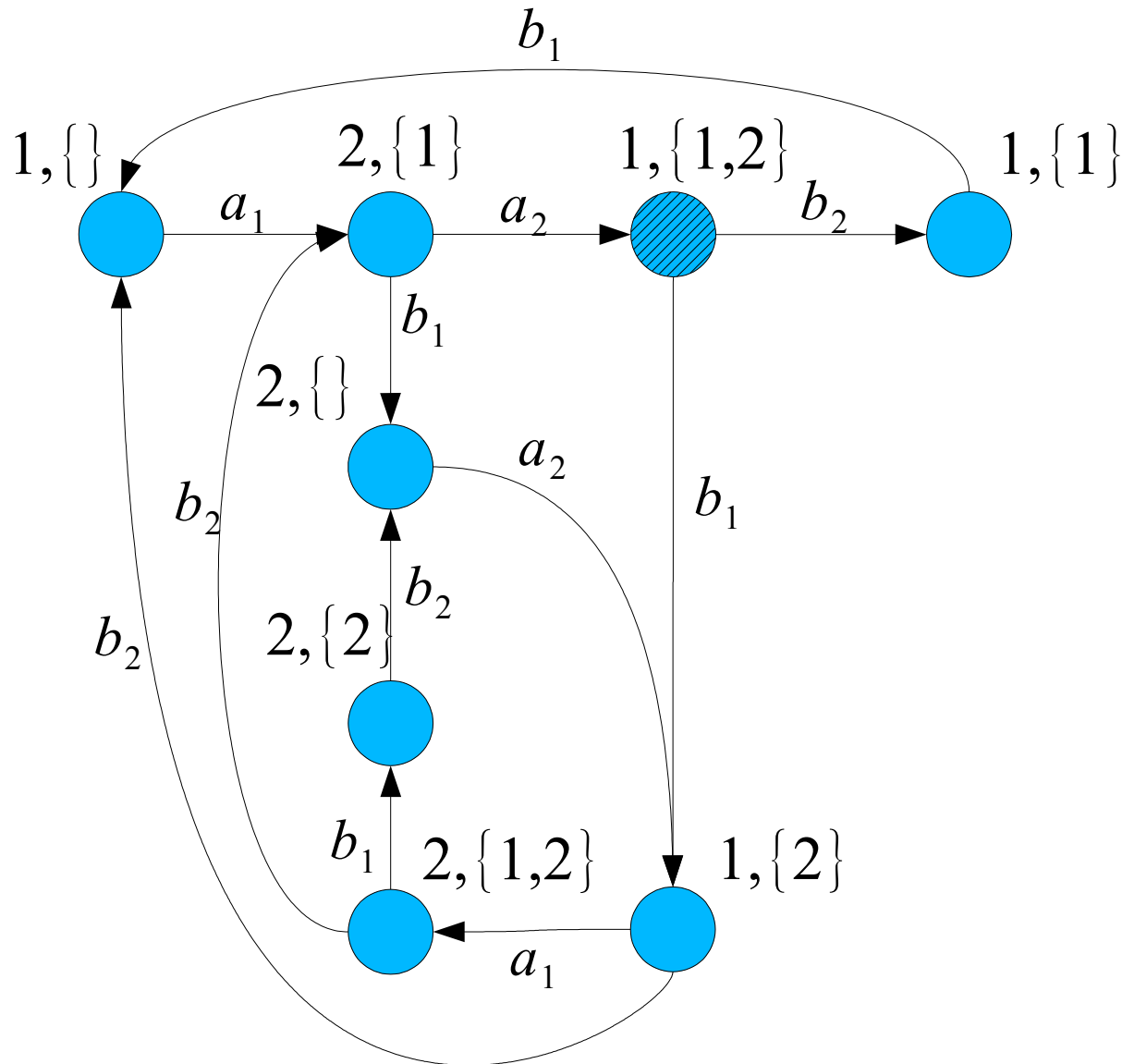
Beispiel: Scheduler



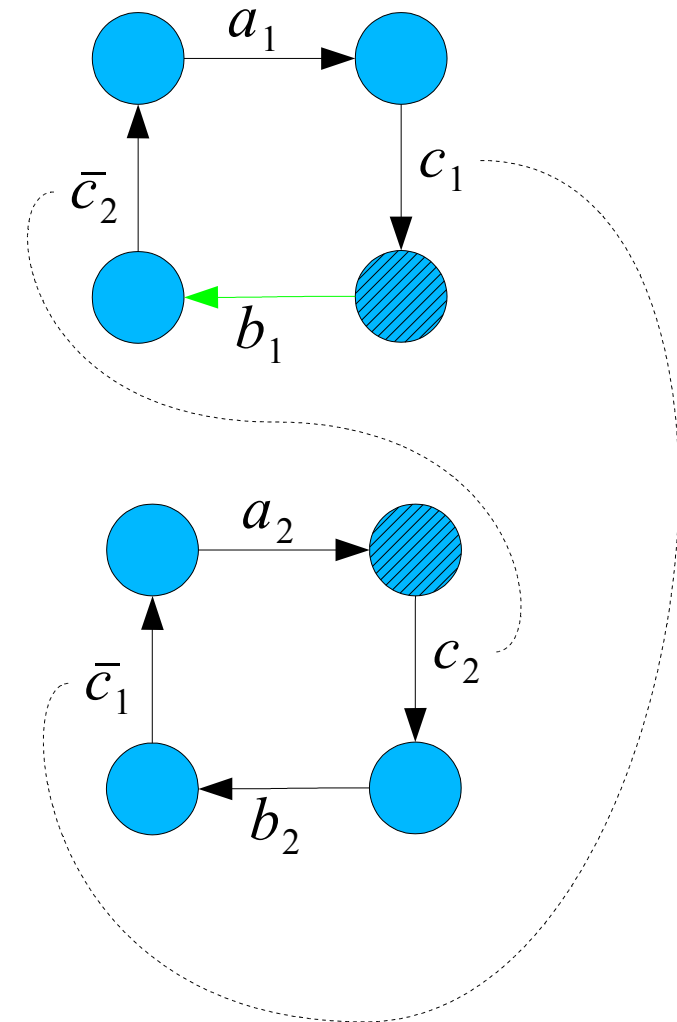
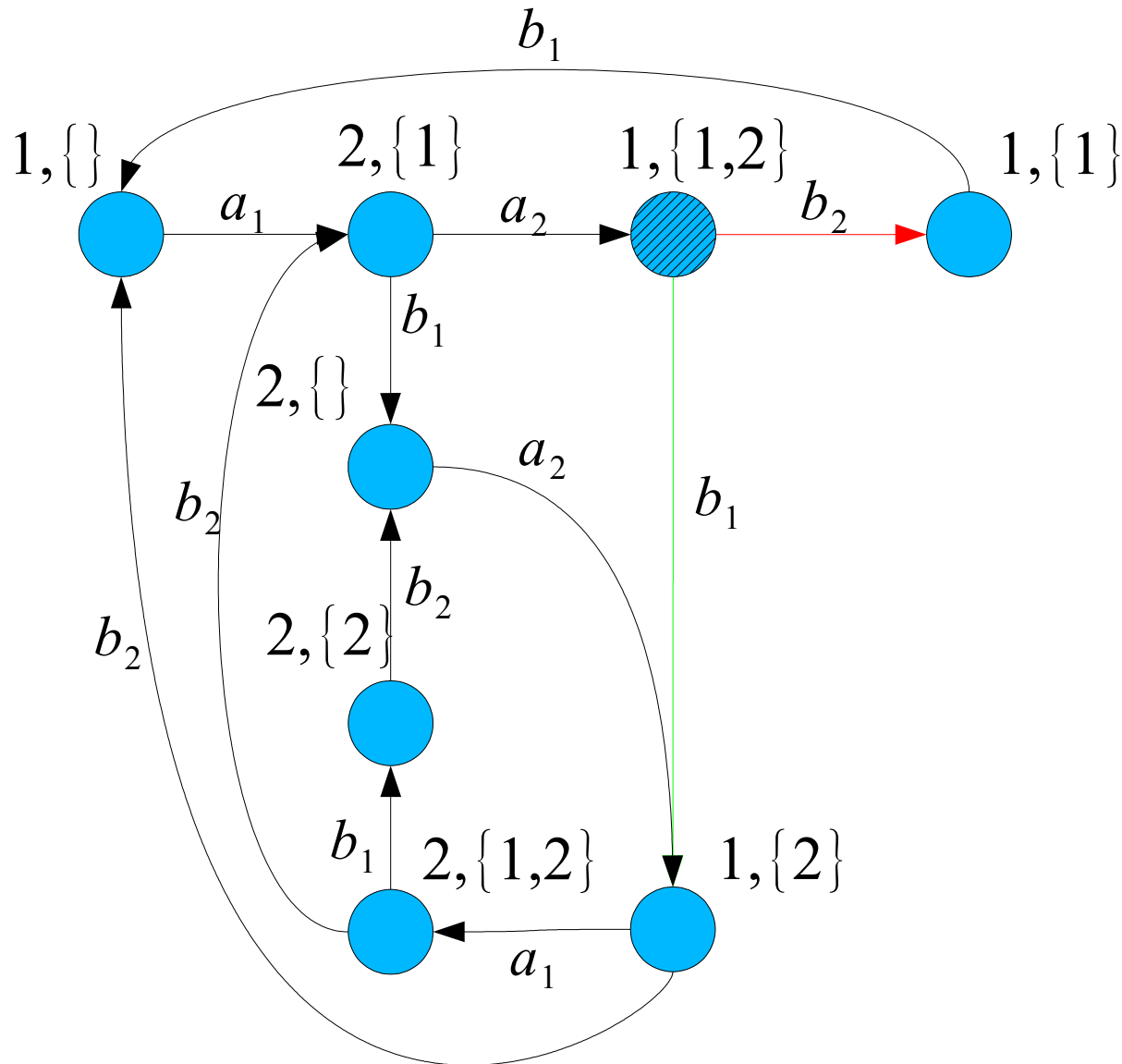
Beispiel: Scheduler



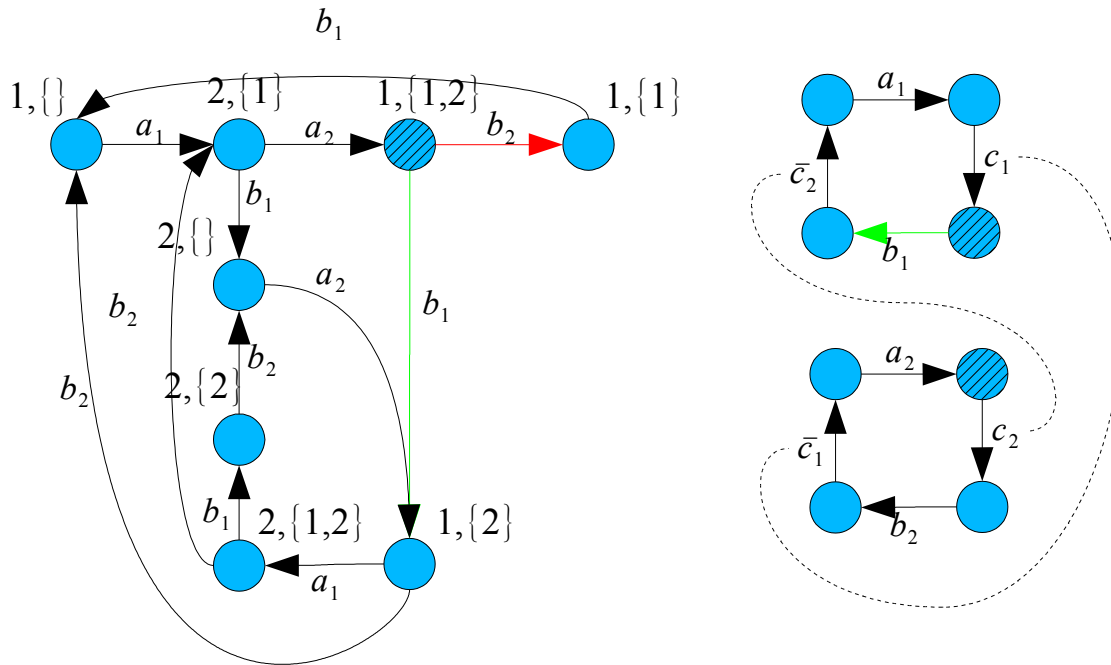
Beispiel: Scheduler



Beispiel: Scheduler



Beispiel: Scheduler



Die Implementierung erfüllt hier nicht die Spezifikation

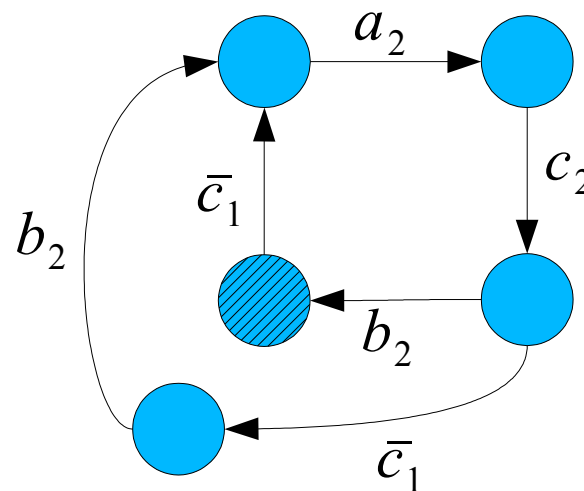
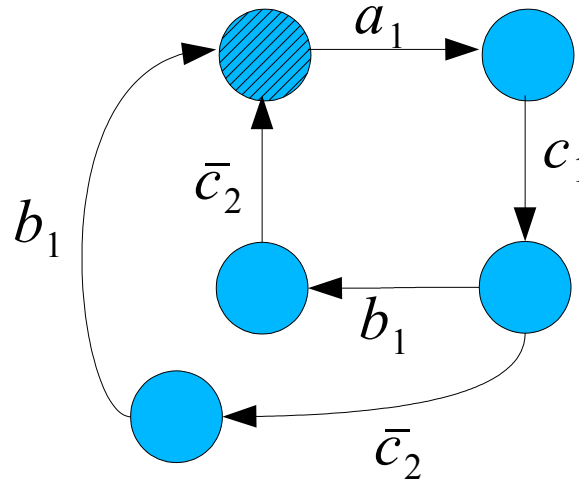
Für den Zustand $1, \{1,2\}$ im linken System existiert kein entsprechender Zustand im rechten System.

Links $(1, \{1,2\}) = b_2 \cdot (1, \{1\}) + b_1 \cdot (1, \{2\})$

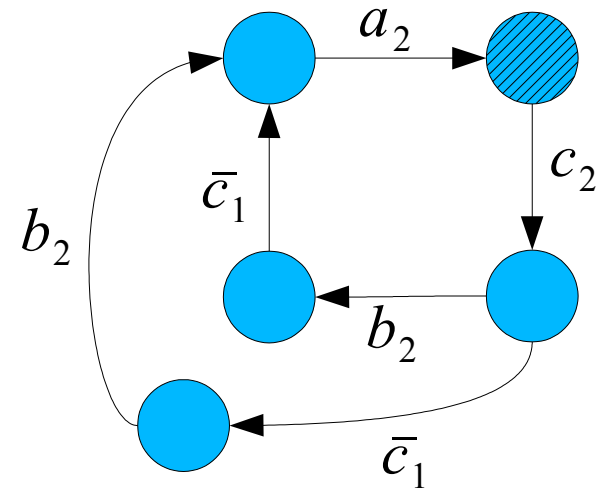
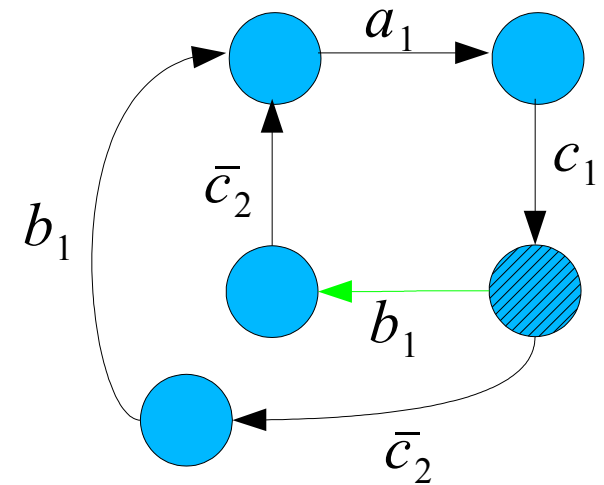
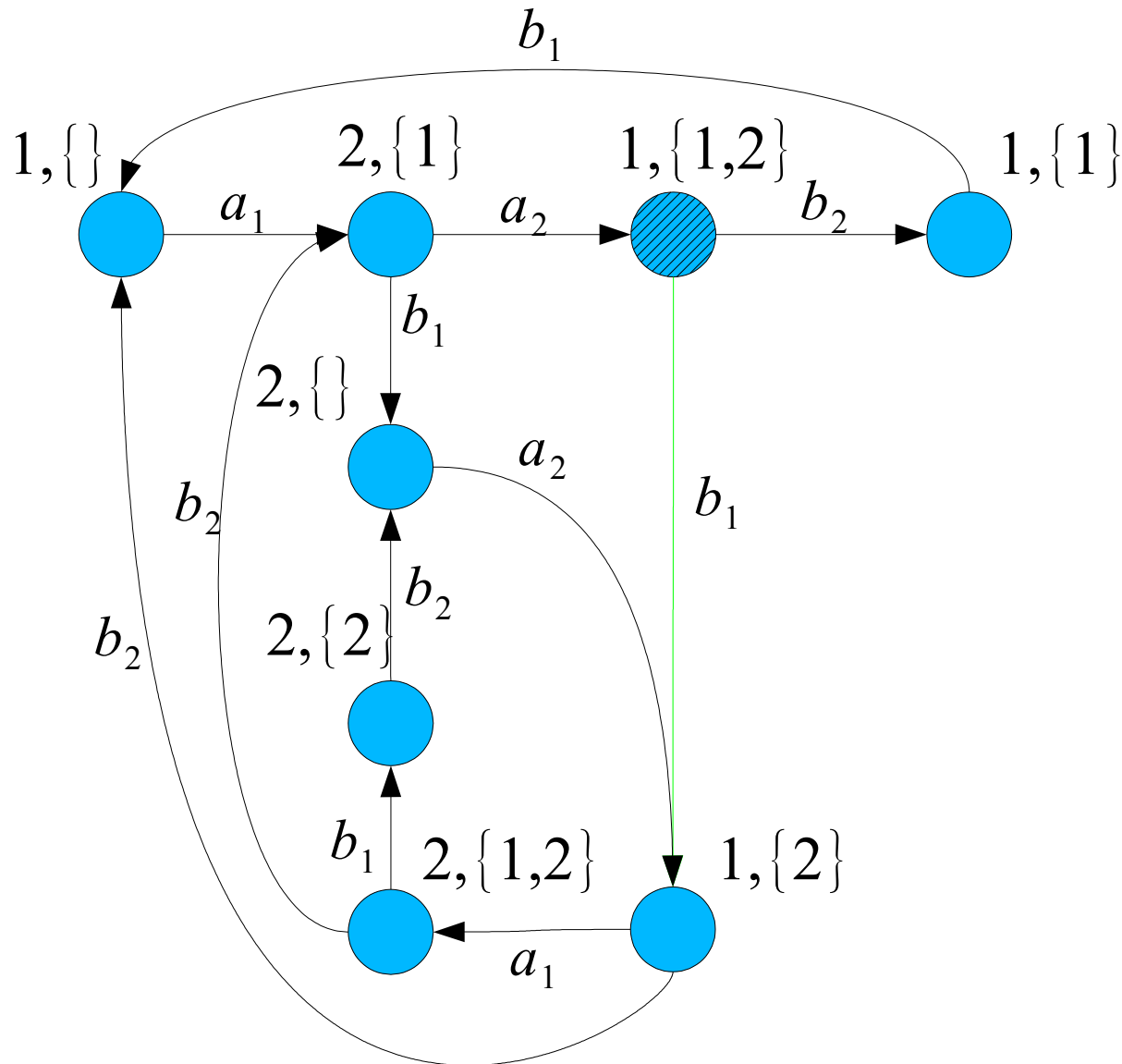
rechts $(1, \{1,2\}) = b_1 \cdot (1, \{2\})$

Beispiel: Scheduler

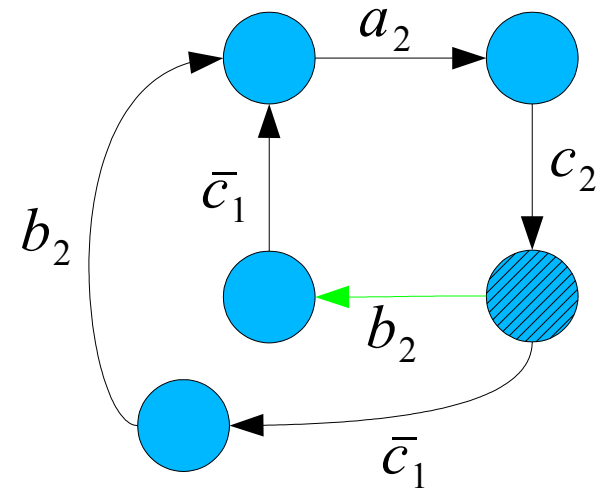
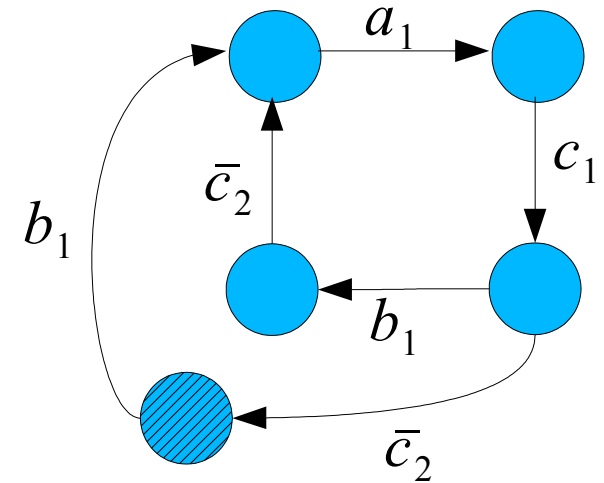
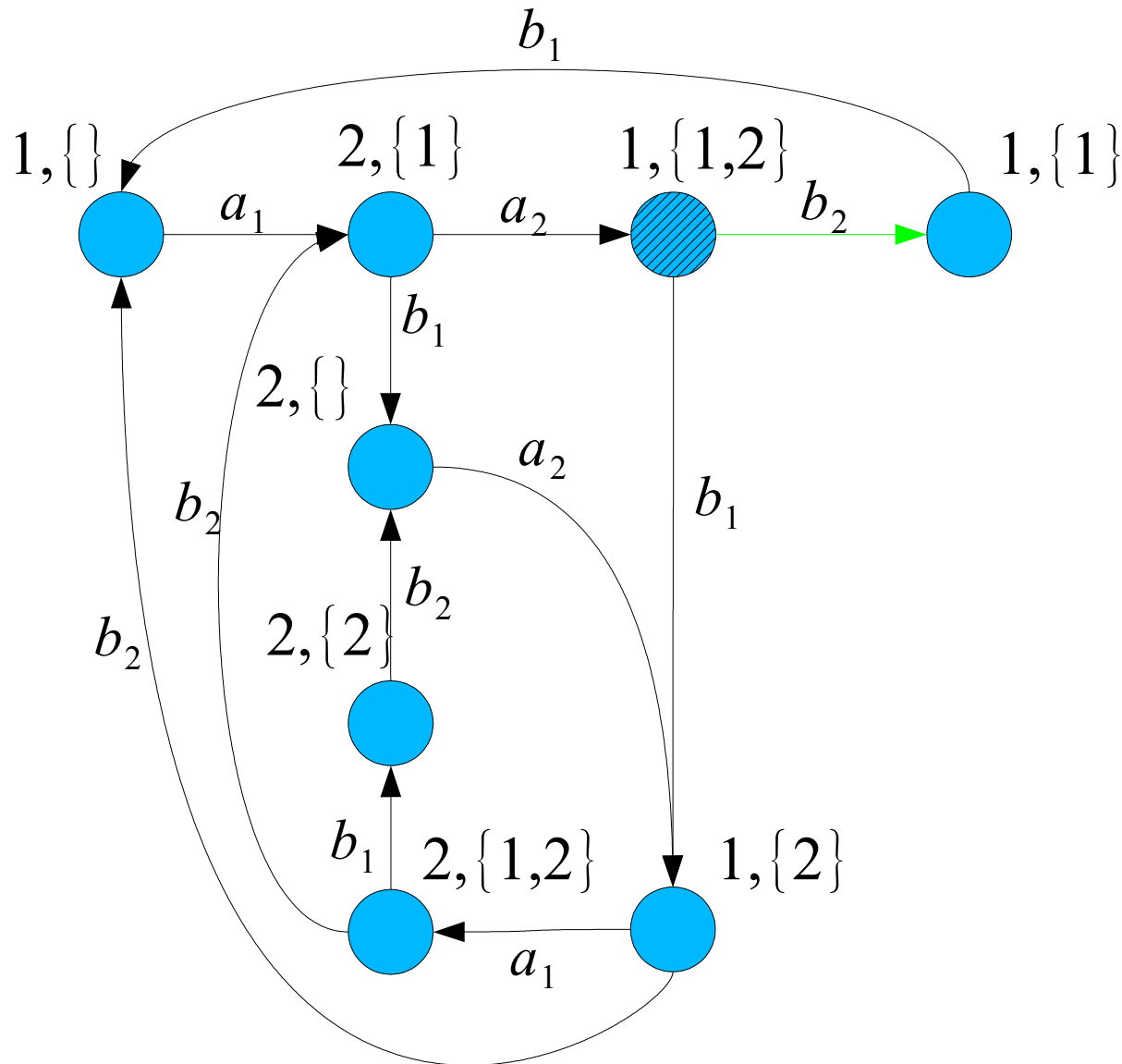
Ändern der Implementierung:



Beispiel: Scheduler

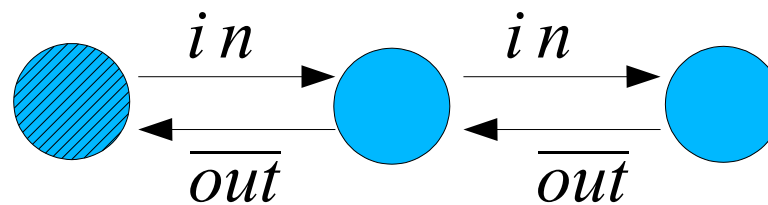
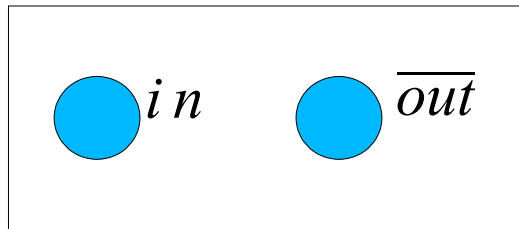


Beispiel: Scheduler



Beispiel: Buffer

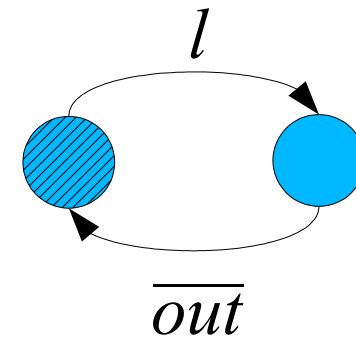
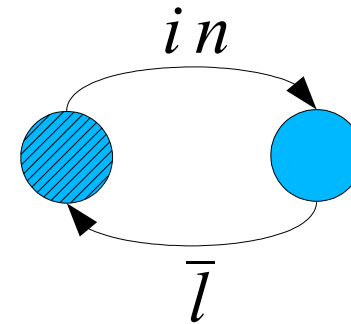
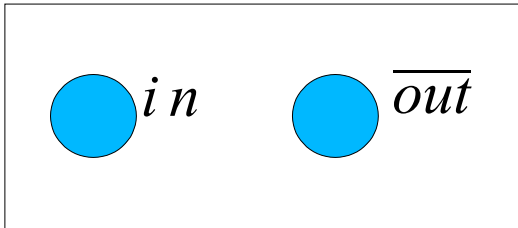
Unärer Buffer mit Kapazität 2



sequentiell

Beispiel: Buffer

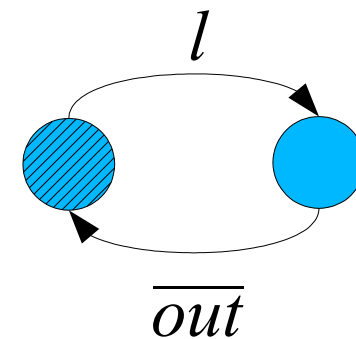
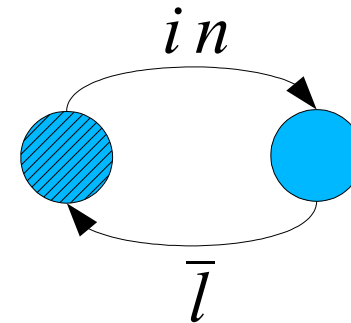
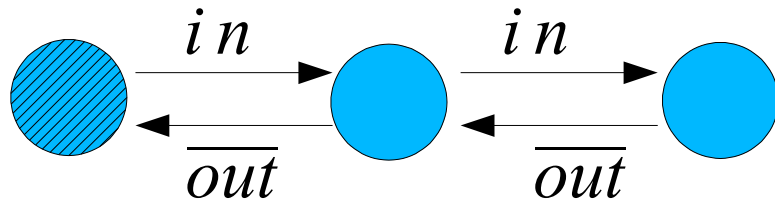
Unärer Buffer der Kapazität 2



parallel

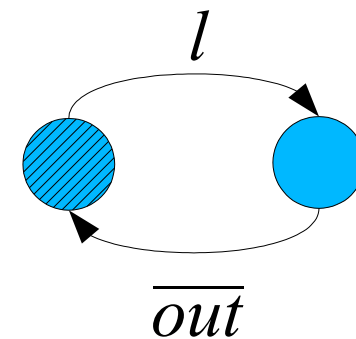
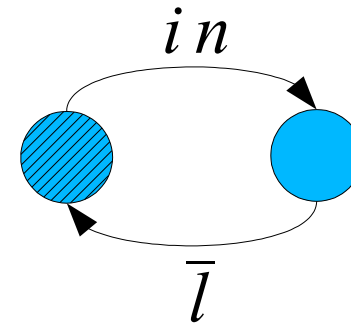
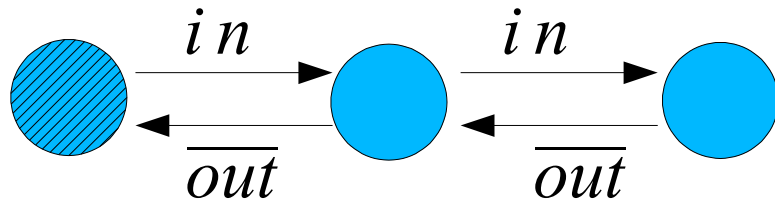
Beispiel: Buffer

Sind beide Systeme schwach äquivalent ?



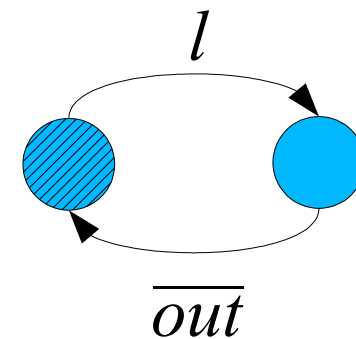
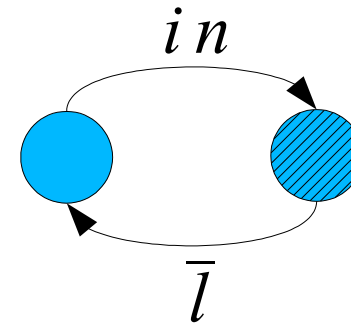
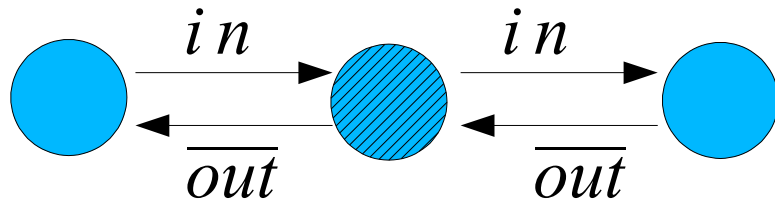
Beispiel: Buffer

Input: in in \overline{out} \overline{out}



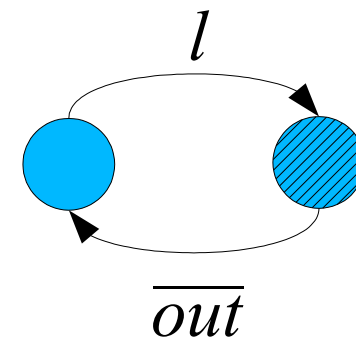
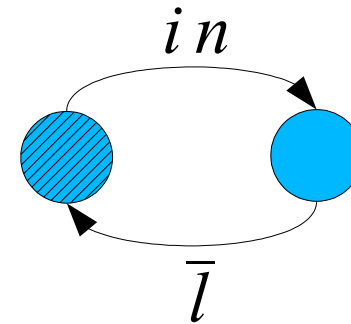
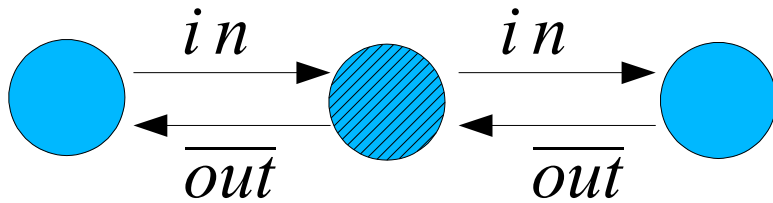
Beispiel: Buffer

Input: in in \overline{out} \overline{out}



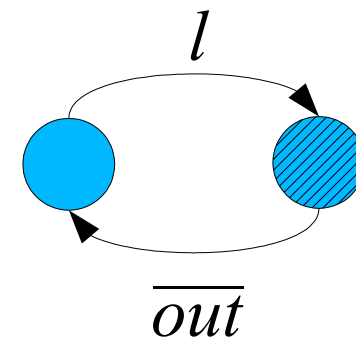
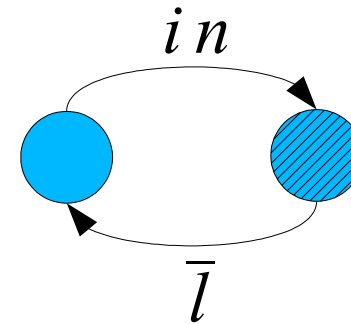
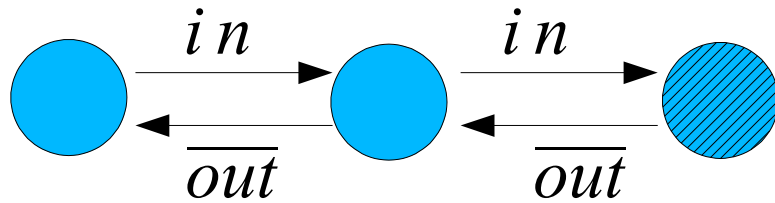
Beispiel: Buffer

Input: in in \overline{out} \overline{out}



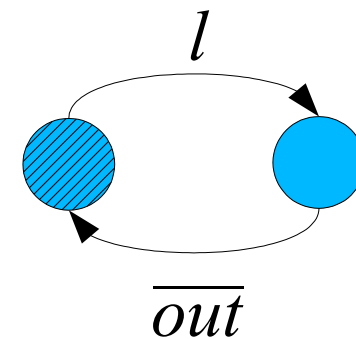
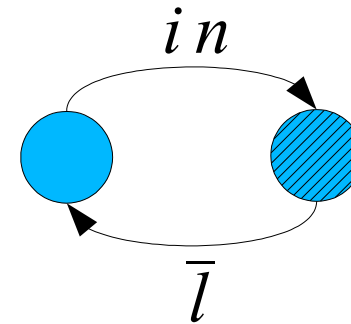
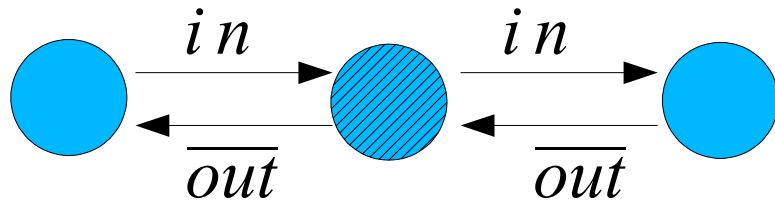
Beispiel: Buffer

Input: in in \overline{out} \overline{out}



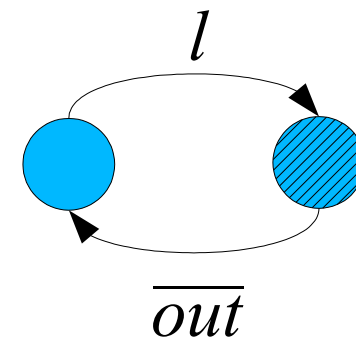
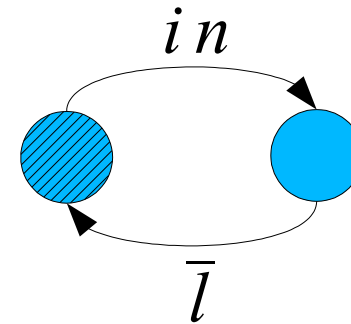
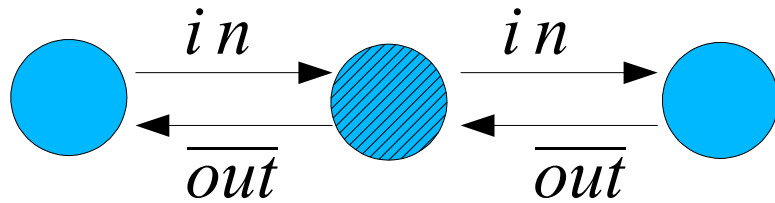
Beispiel: Buffer

Input: in in out \overline{out}



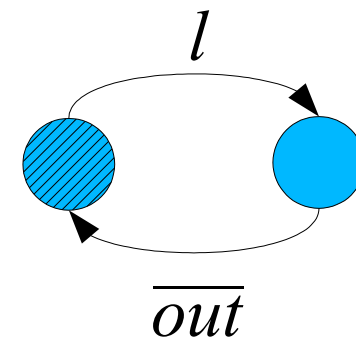
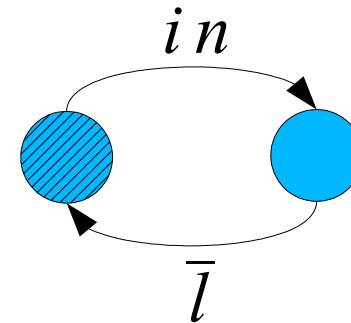
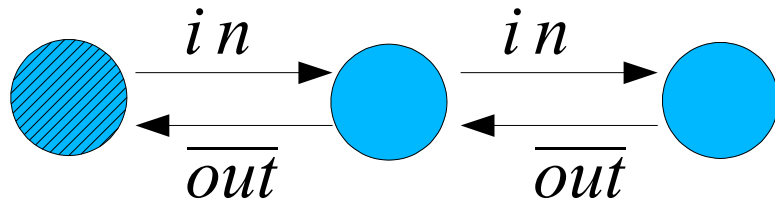
Beispiel: Buffer

Input: in in \overline{out} \overline{out}



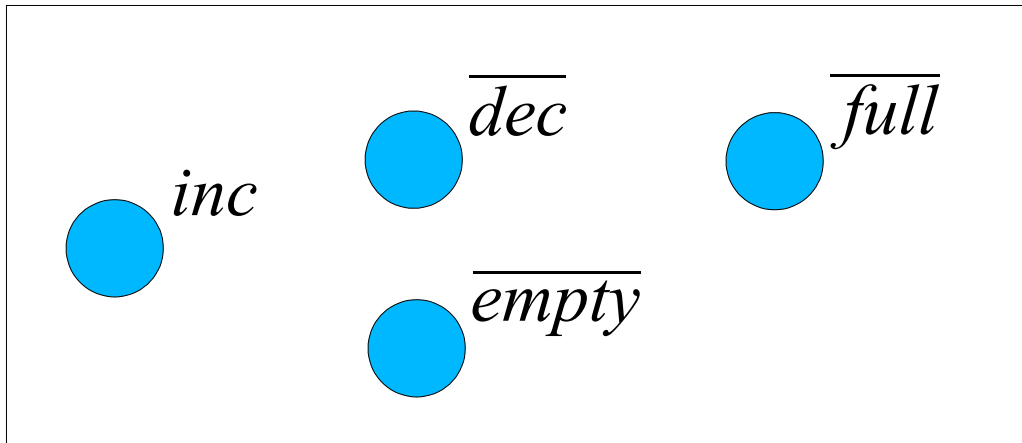
Beispiel: Buffer

Input: in in \overline{out} \overline{out}

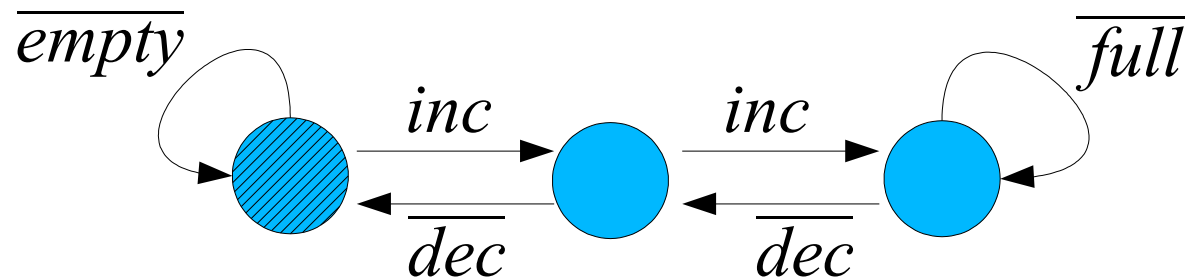


Beide Systeme sind schwach äquivalent.

Beispiel: Counter

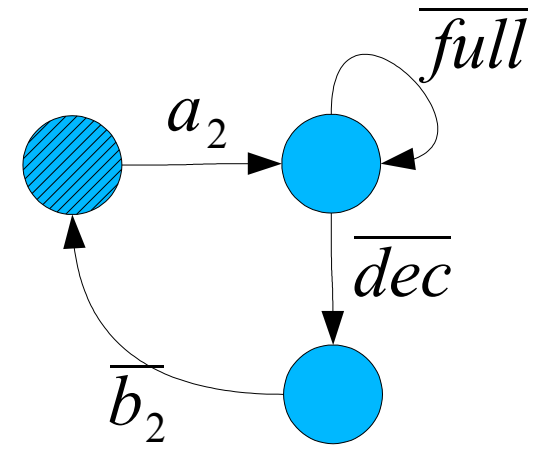
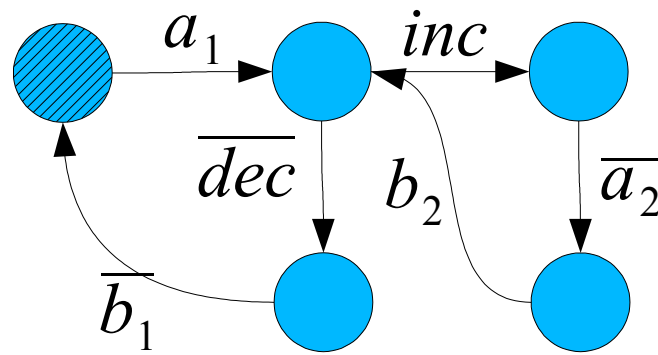
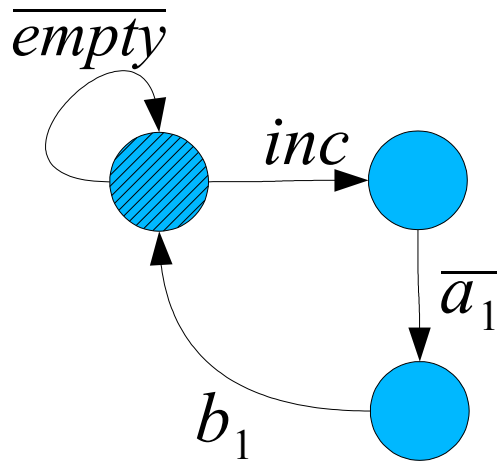


Spezifikation Counter



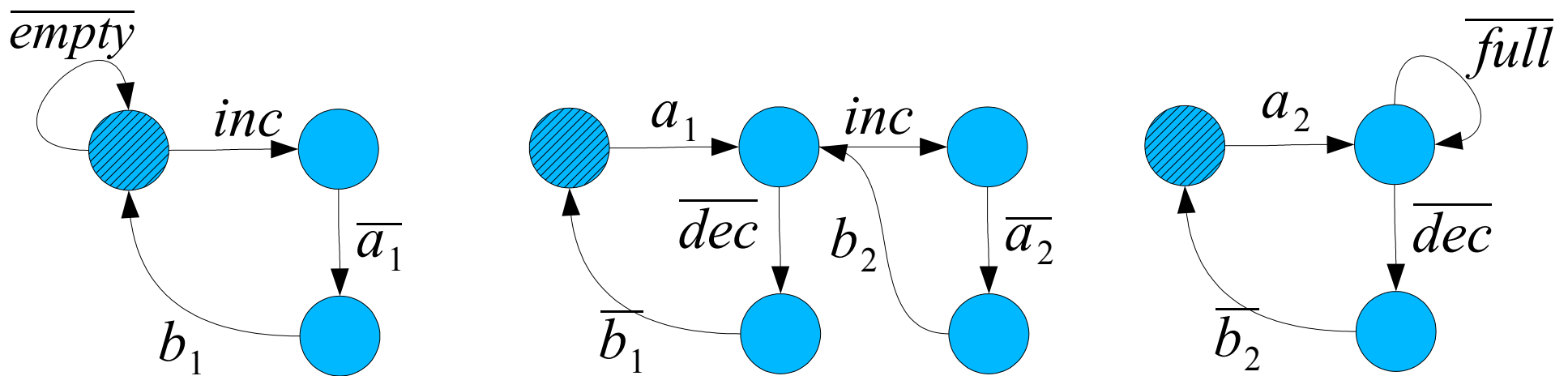
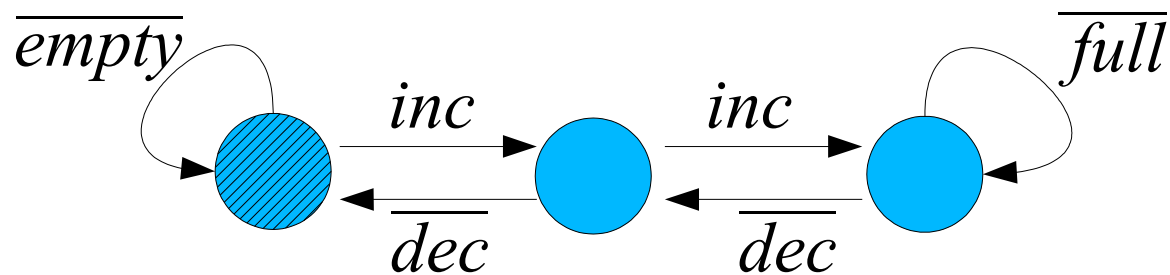
Beispiel: Counter

Implementierung Counter



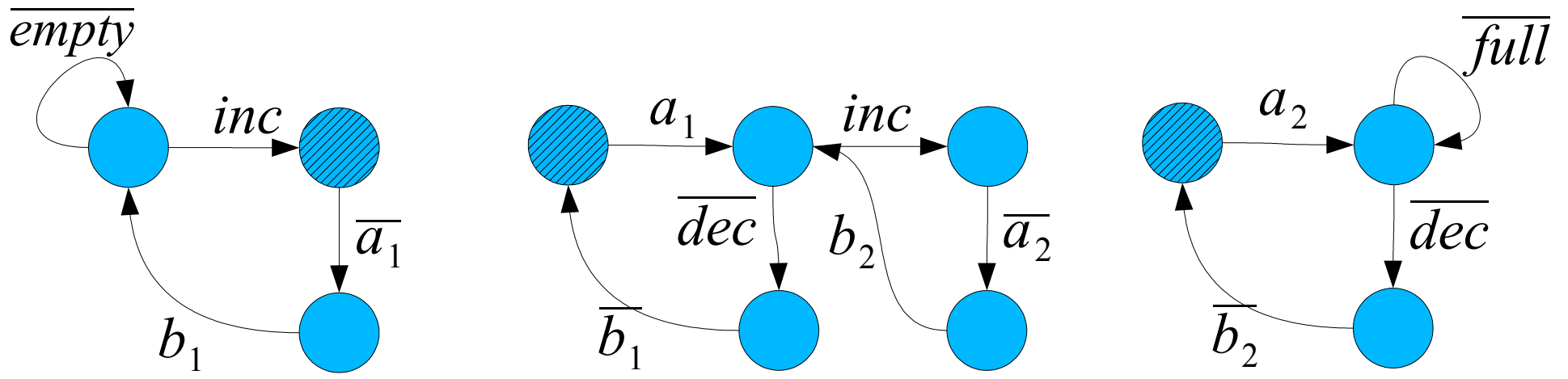
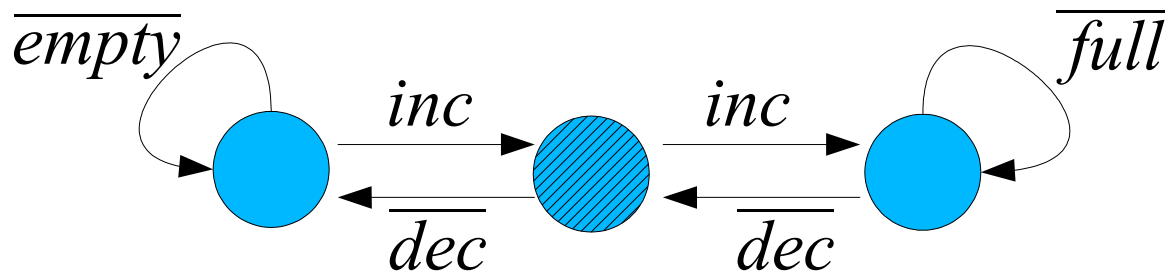
Beispiel: Counter

Input: inc inc \overline{dec} \overline{dec} \overline{empty}



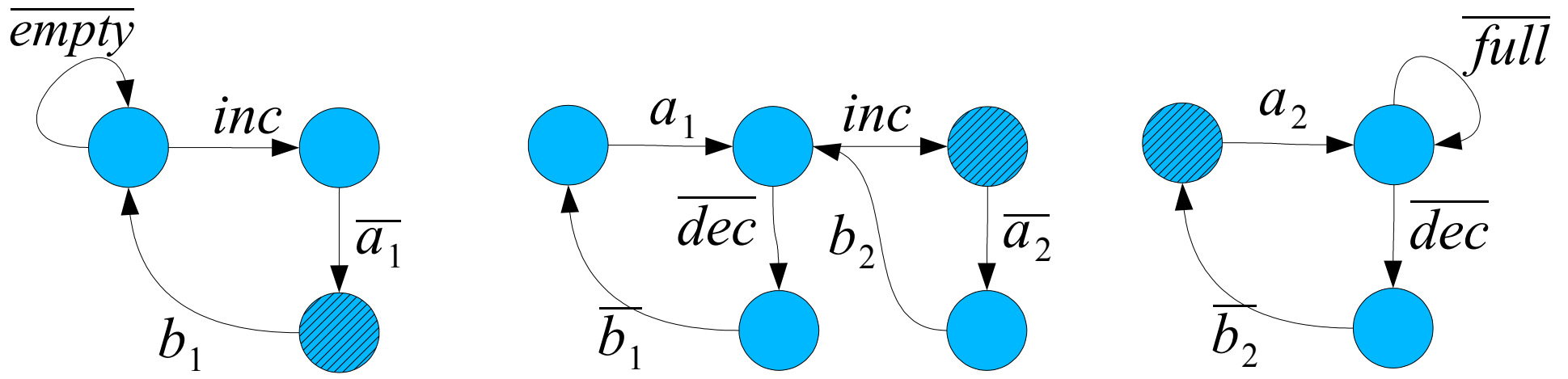
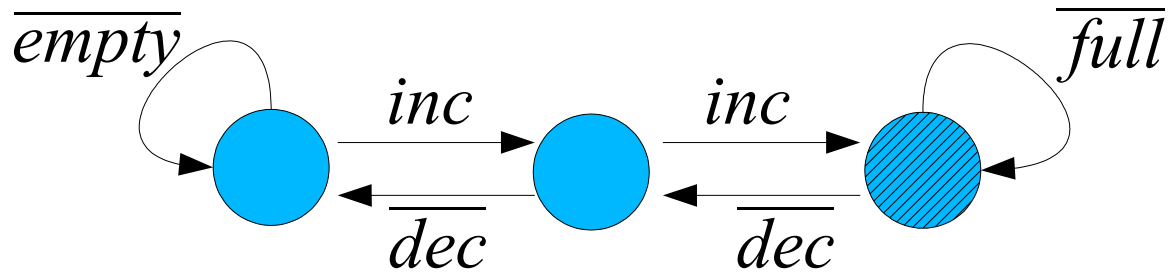
Beispiel: Counter

Input: inc inc \overline{dec} \overline{dec} \overline{empty}



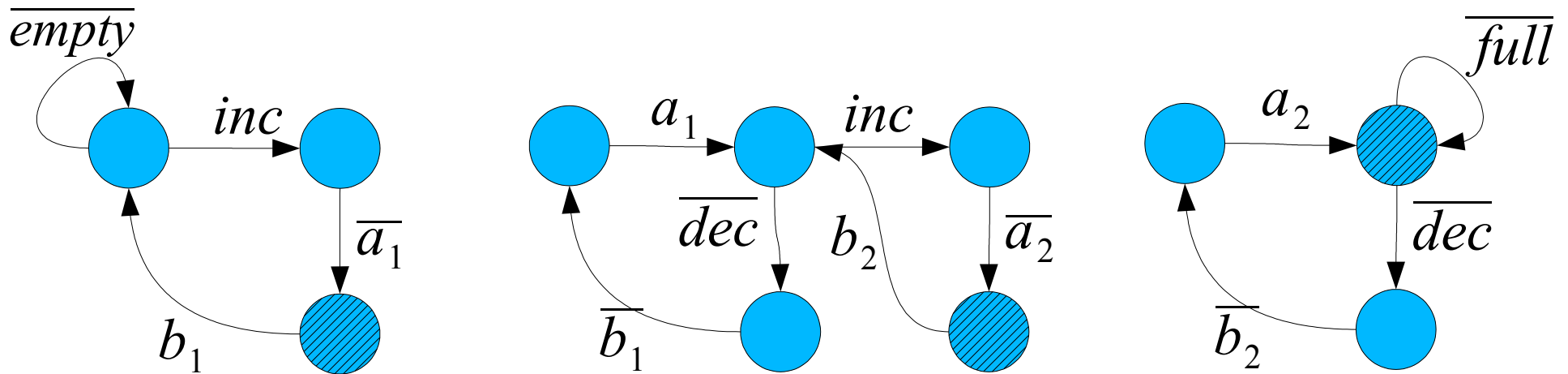
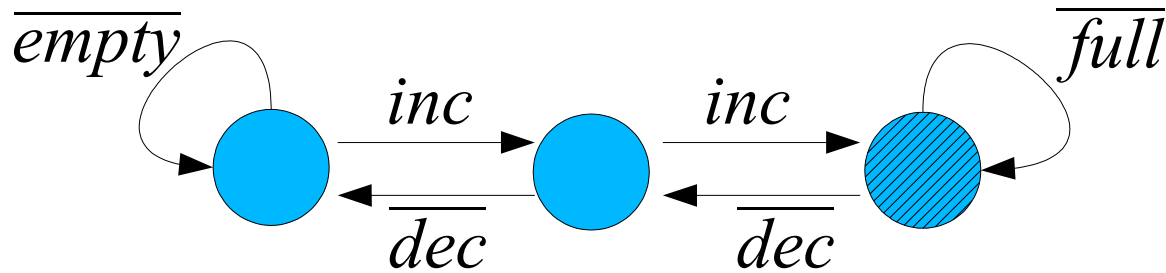
Beispiel: Counter

Input: *inc* inc \overline{dec} \overline{dec} \overline{empty}



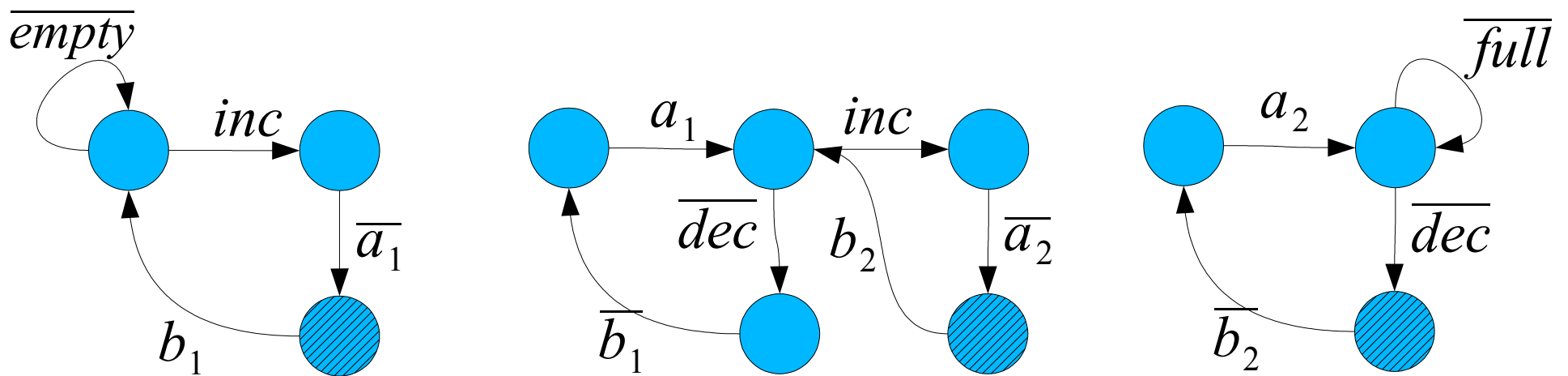
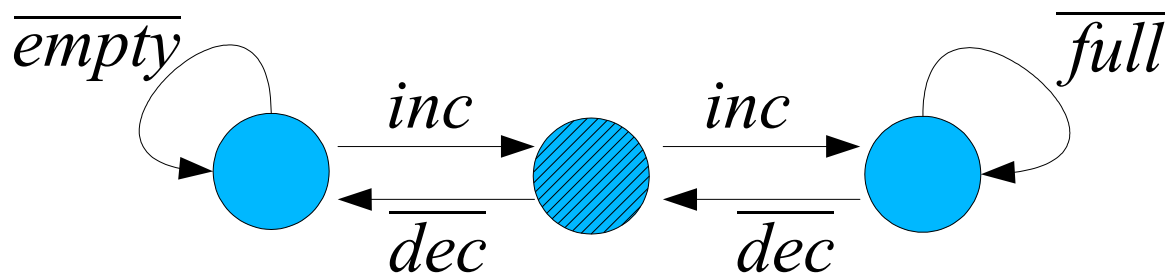
Beispiel: Counter

Input: *inc inc* dec \overline{dec} \overline{empty}



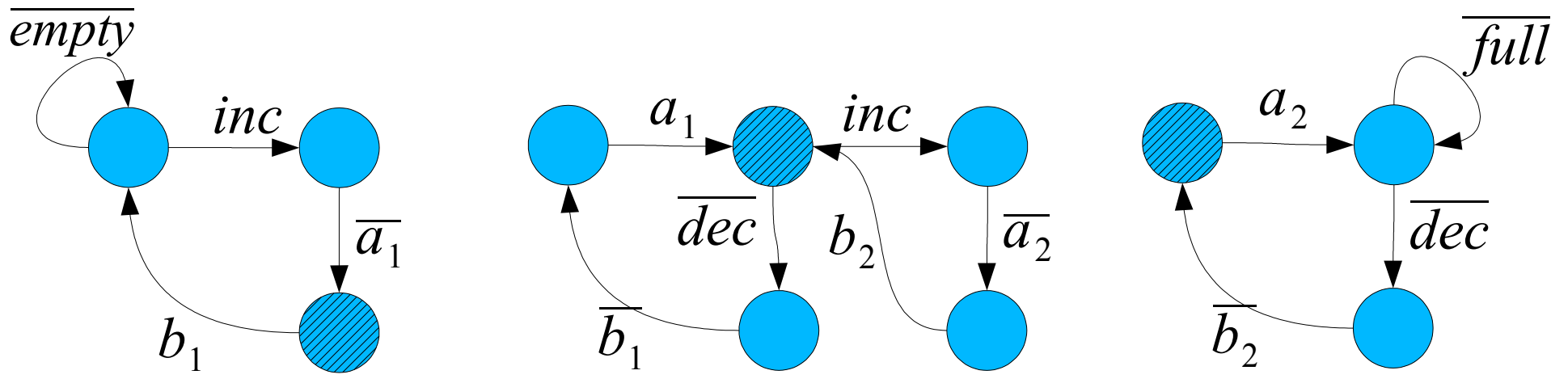
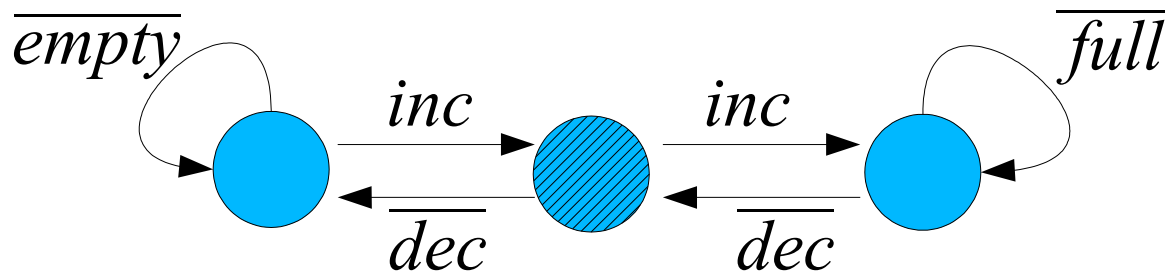
Beispiel: Counter

Input: *inc inc* dec \overline{dec} \overline{empty}



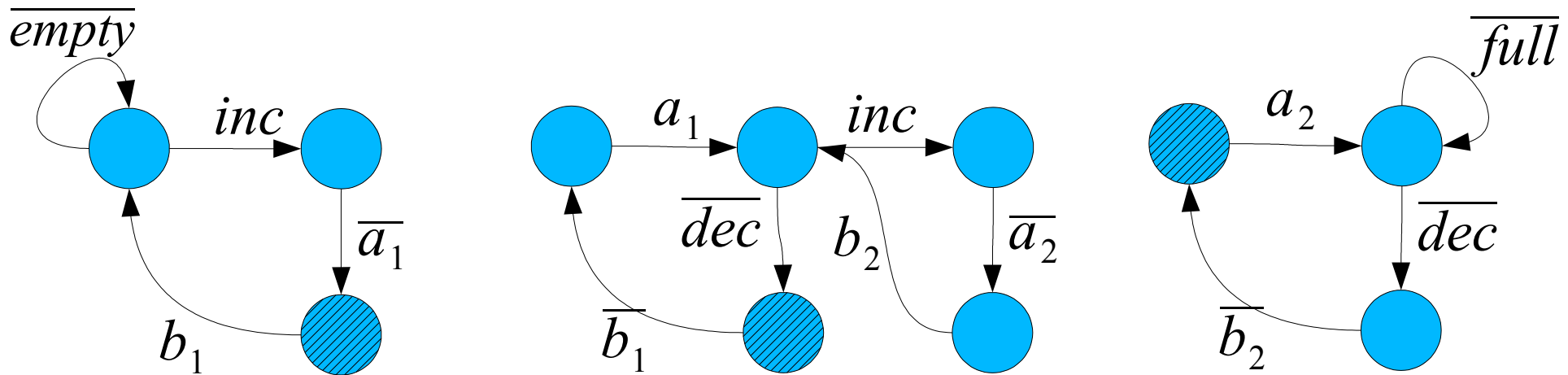
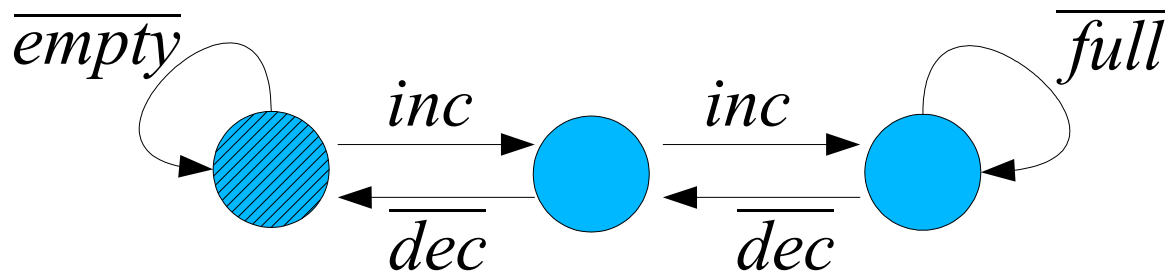
Beispiel: Counter

Input: inc inc \overline{dec} \overline{dec} \overline{empty}



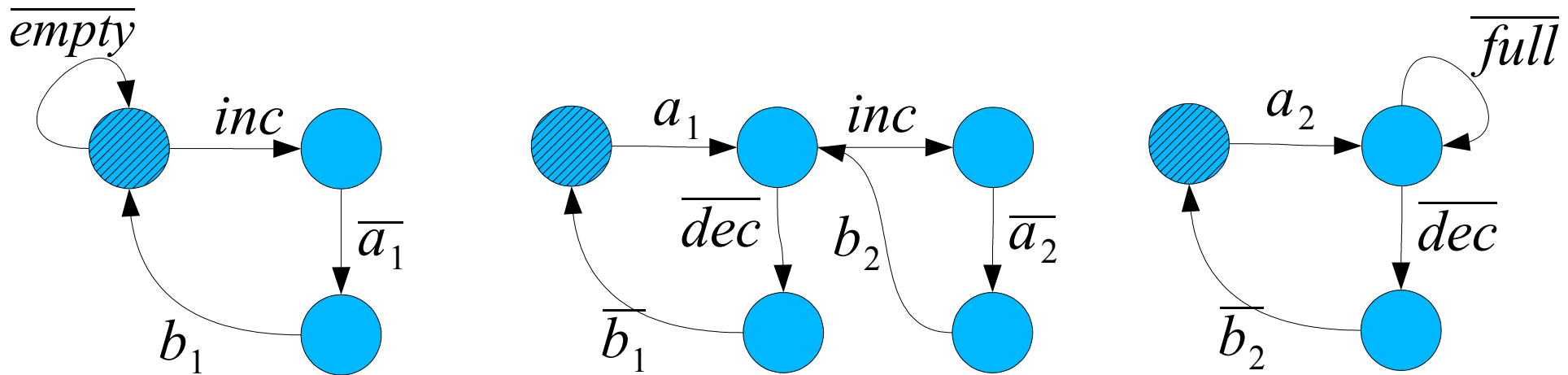
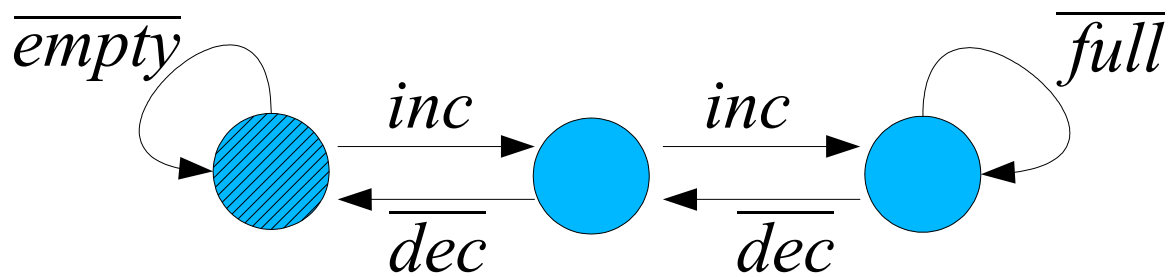
Beispiel: Counter

Input: inc inc \overline{dec} \overline{dec} \overline{empty}

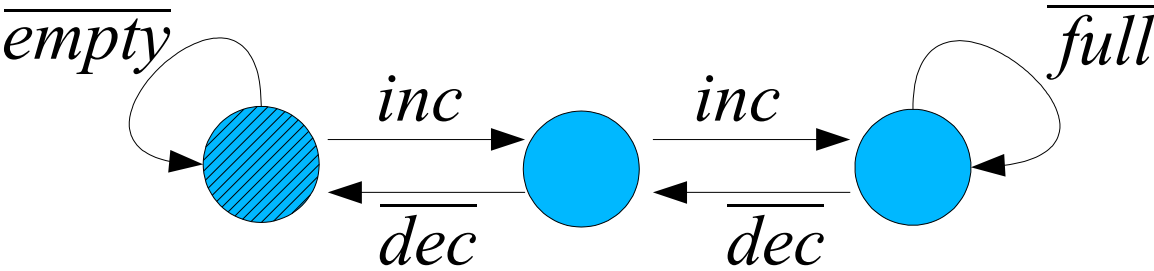


Beispiel: Counter

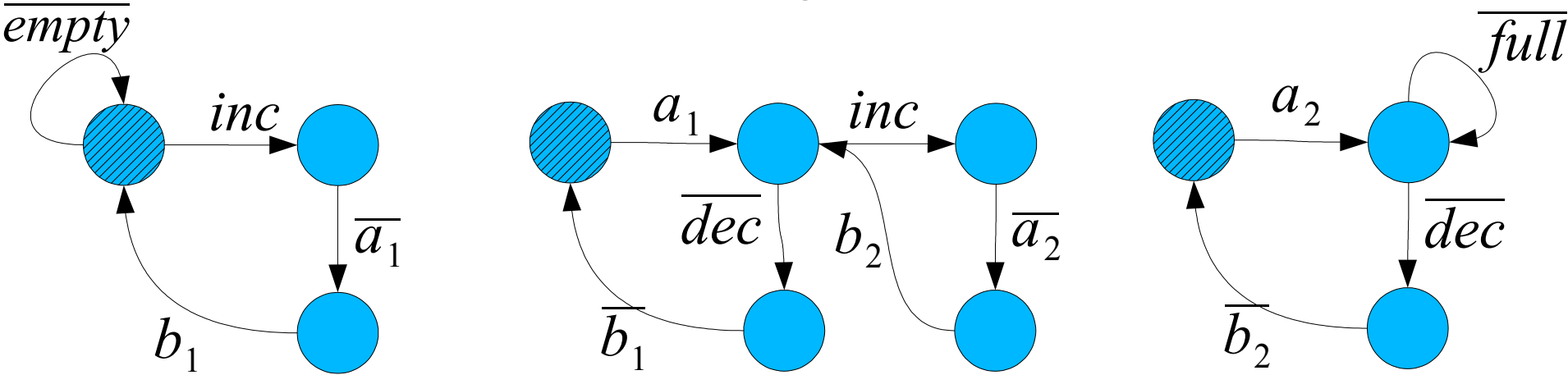
Input: *inc inc* \overline{dec} \overline{dec} \overline{empty}



Beispiel: Counter



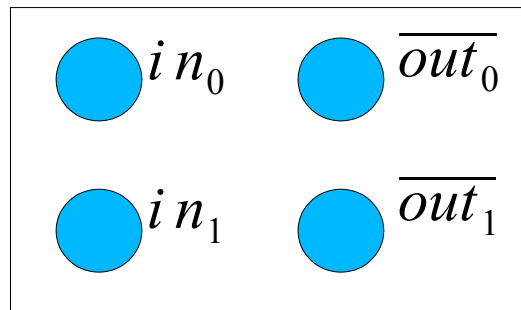
\approx



Referenzen

Milner, R., Communicating and Mobile Systems:
the Pi - Calculus, Cambridge University Press, 1999

Beispiel: Buffer



Binärer Buffer der Größe zwei:

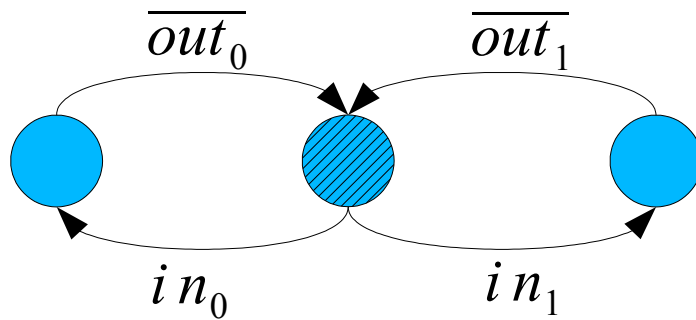
Er speichert binäre Werte und gibt sie in der Reihenfolge, in der sie gespeichert wurden wieder aus.

Kann folgende Werte speichern:

$\{0,1,10,00,11,01\}$

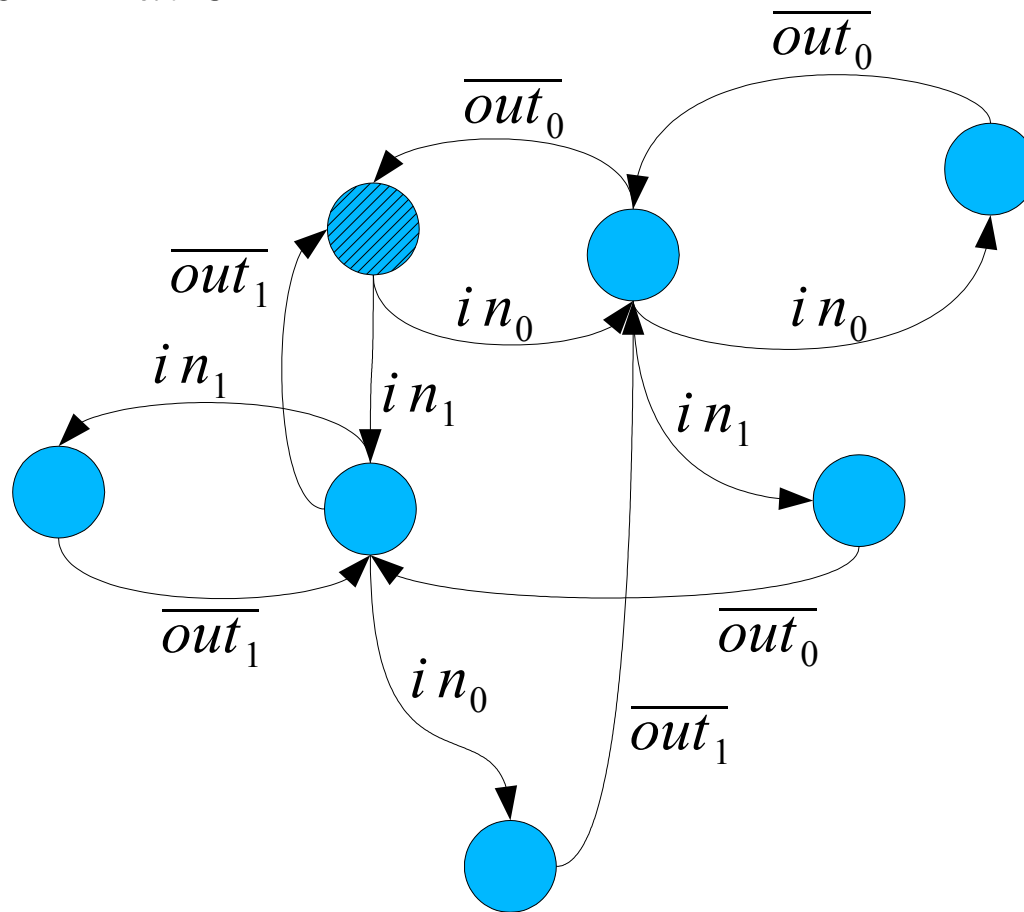
Beispiel: Buffer

Spezifikation $n=1$



Beispiel: Buffer

Spezifikation $n=2$



Beispiel: Buffer

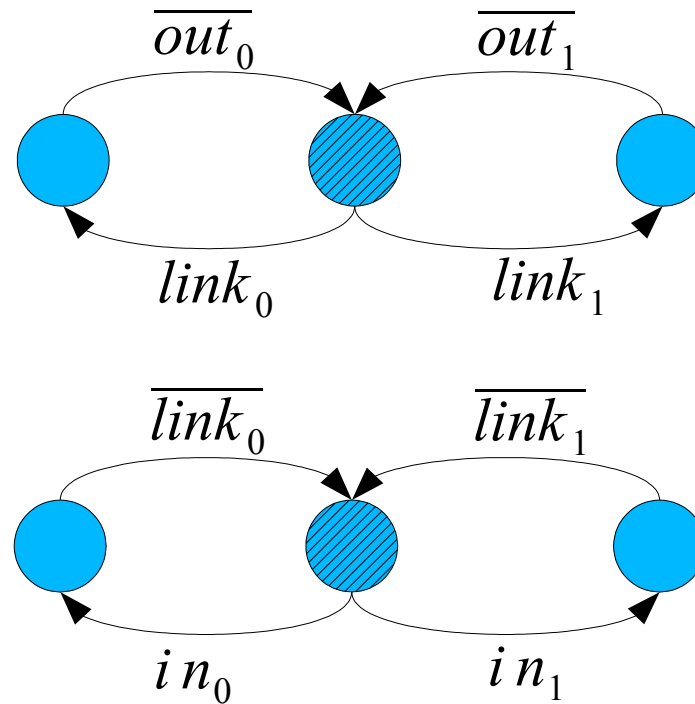
Implementierung für $n=2$



Zwei Speicherzellen von denen sich jeder eine 0 oder 1 merken kann

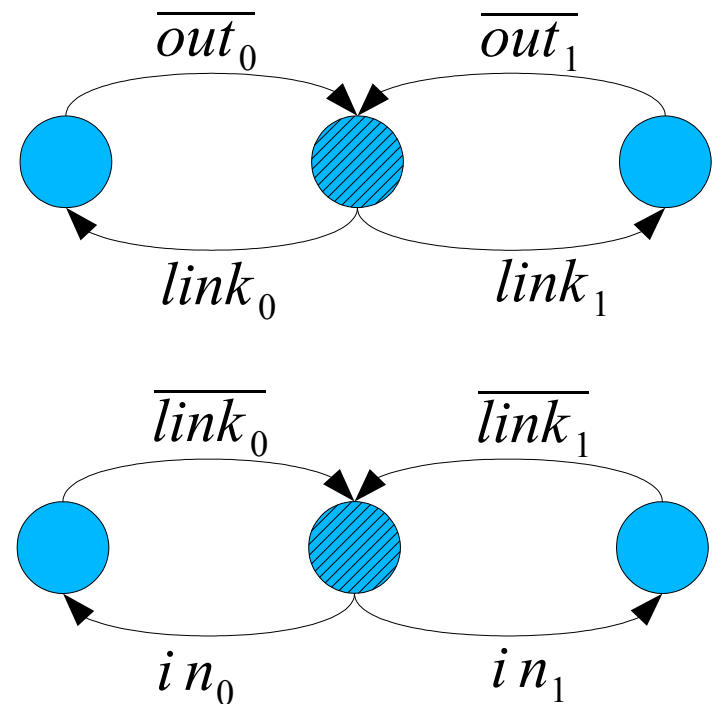
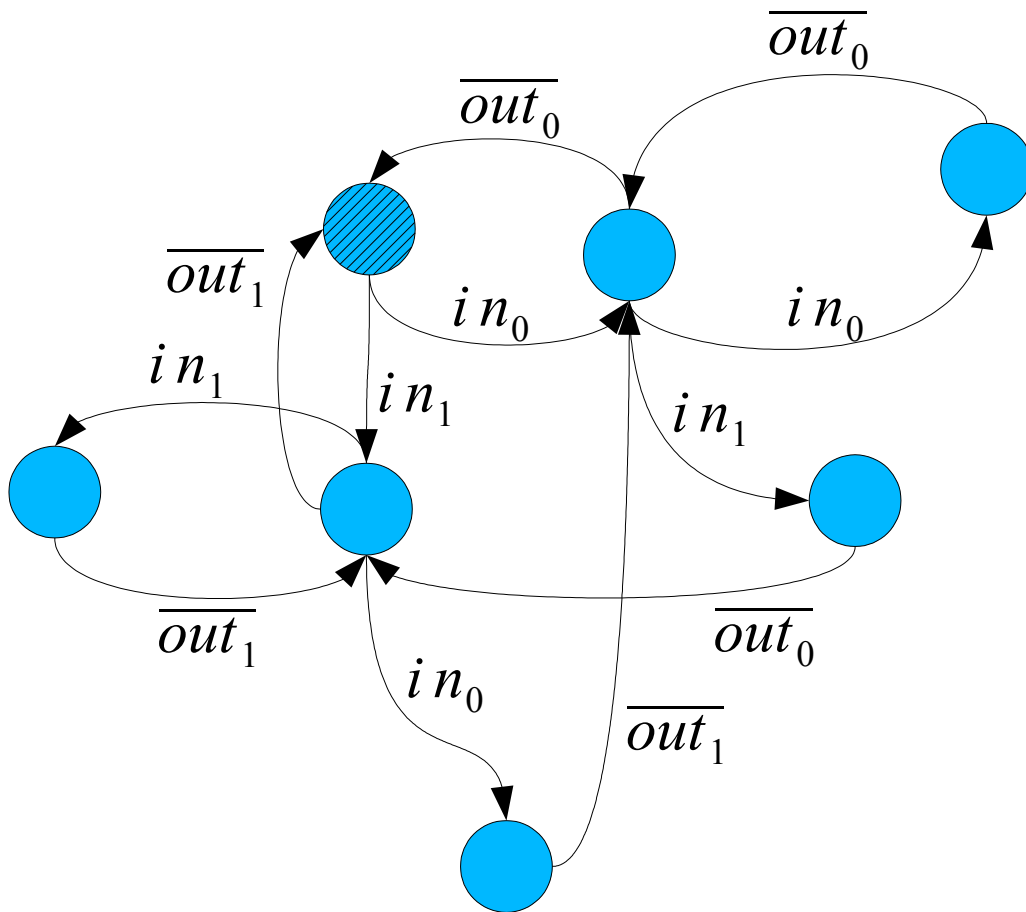
Beispiel: Buffer

Implementierung für $n=2$



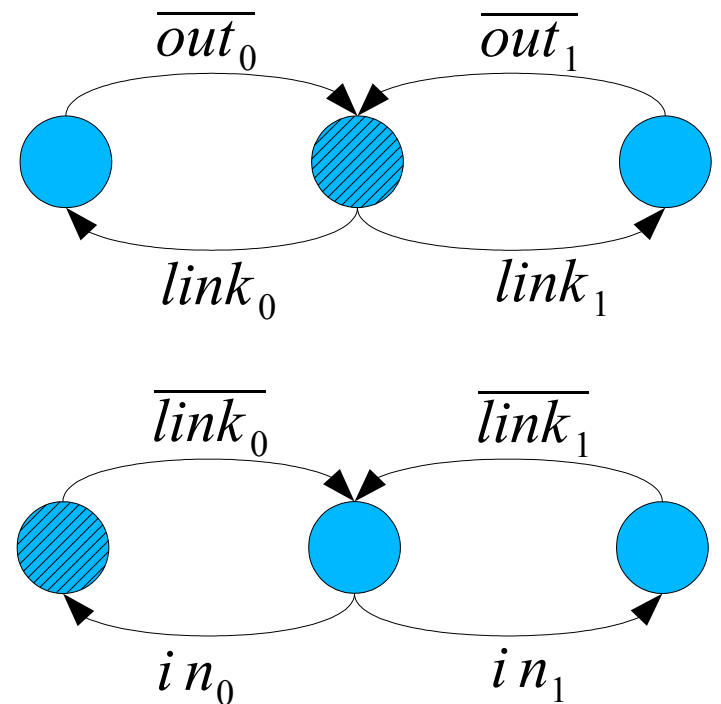
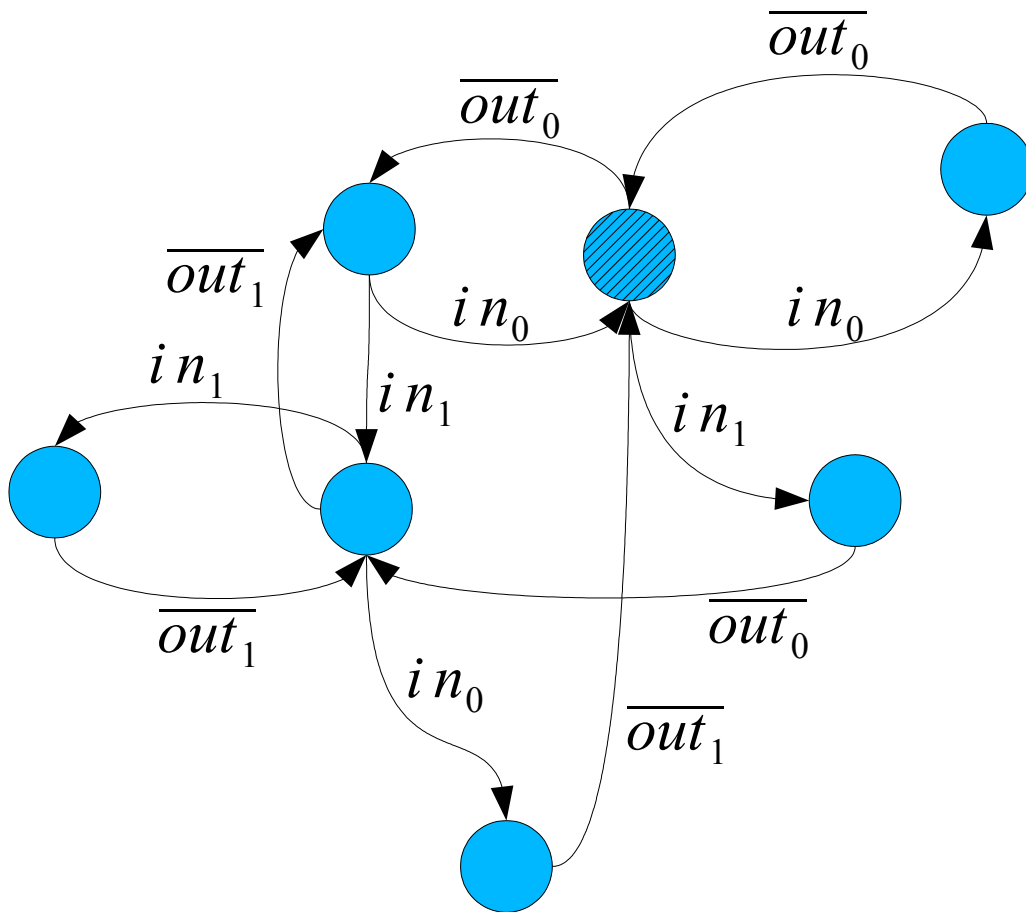
Beispiel: Buffer

Beispiel: speicher 01 in Buffer und lese diese Zahl wieder aus



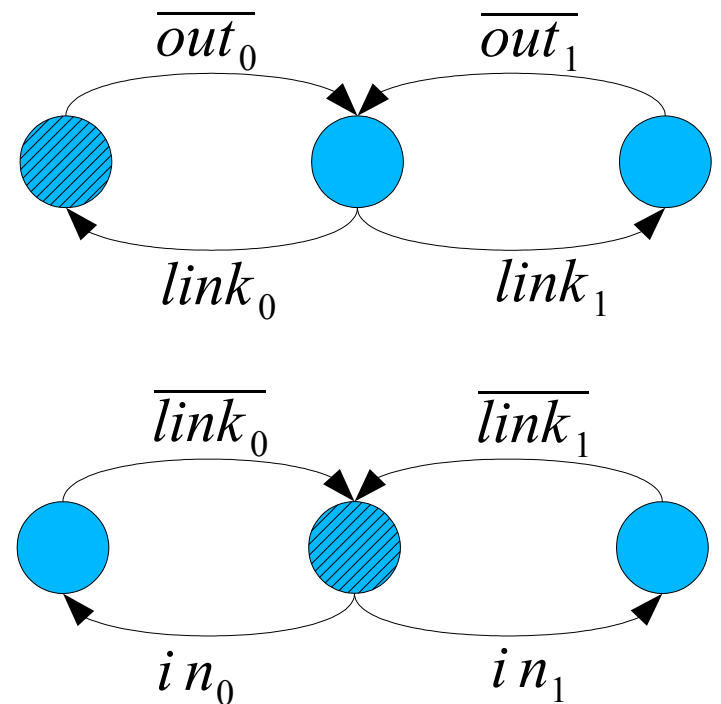
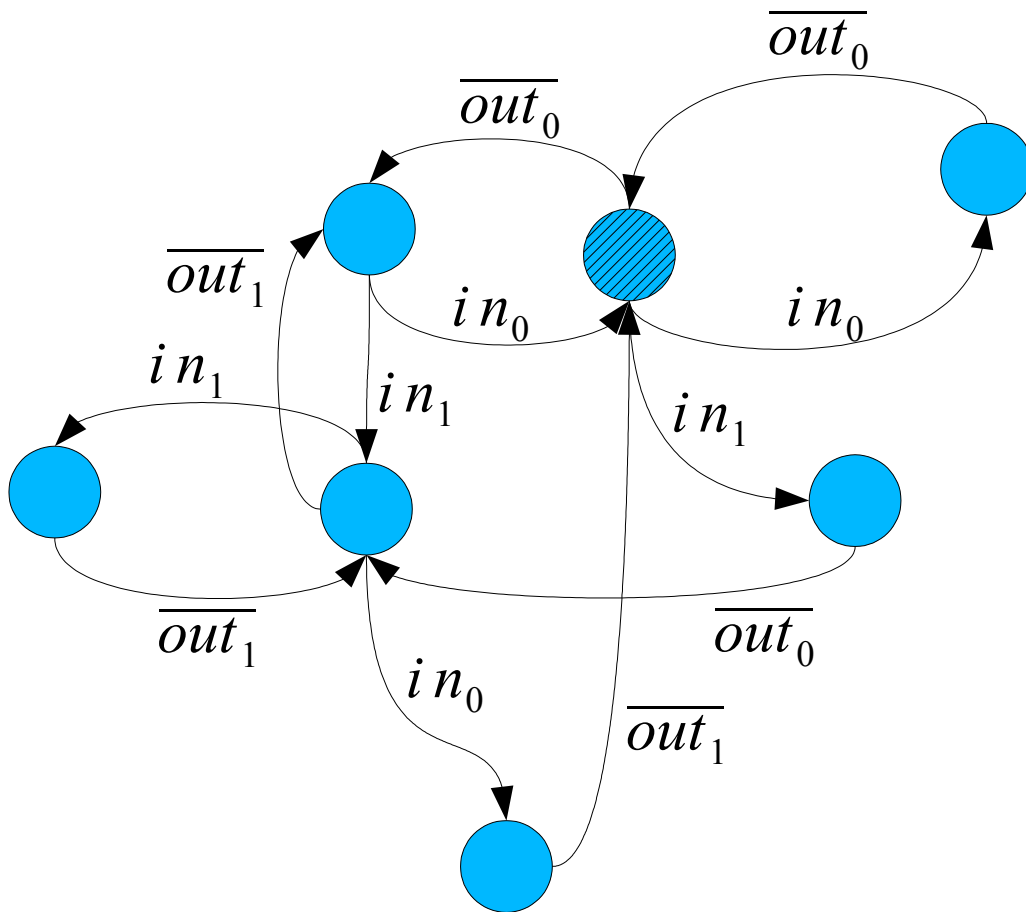
Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus



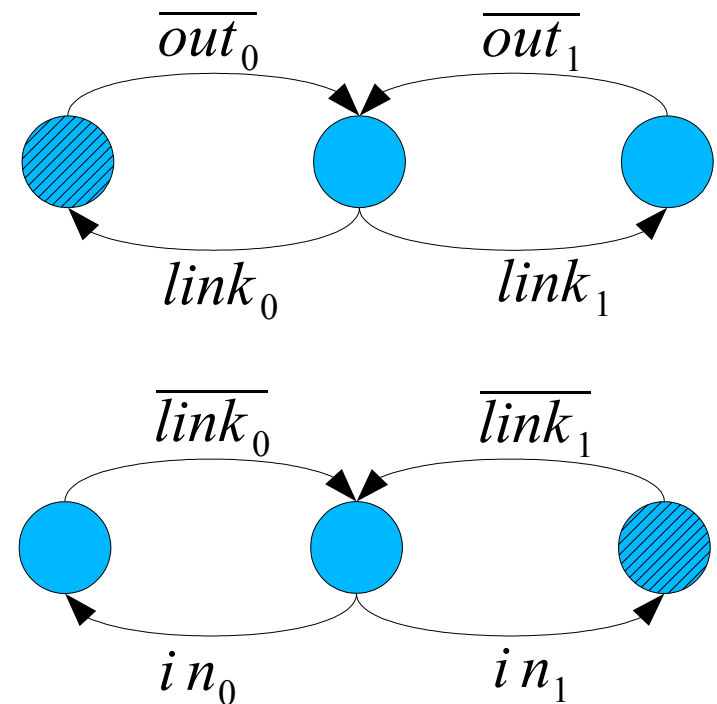
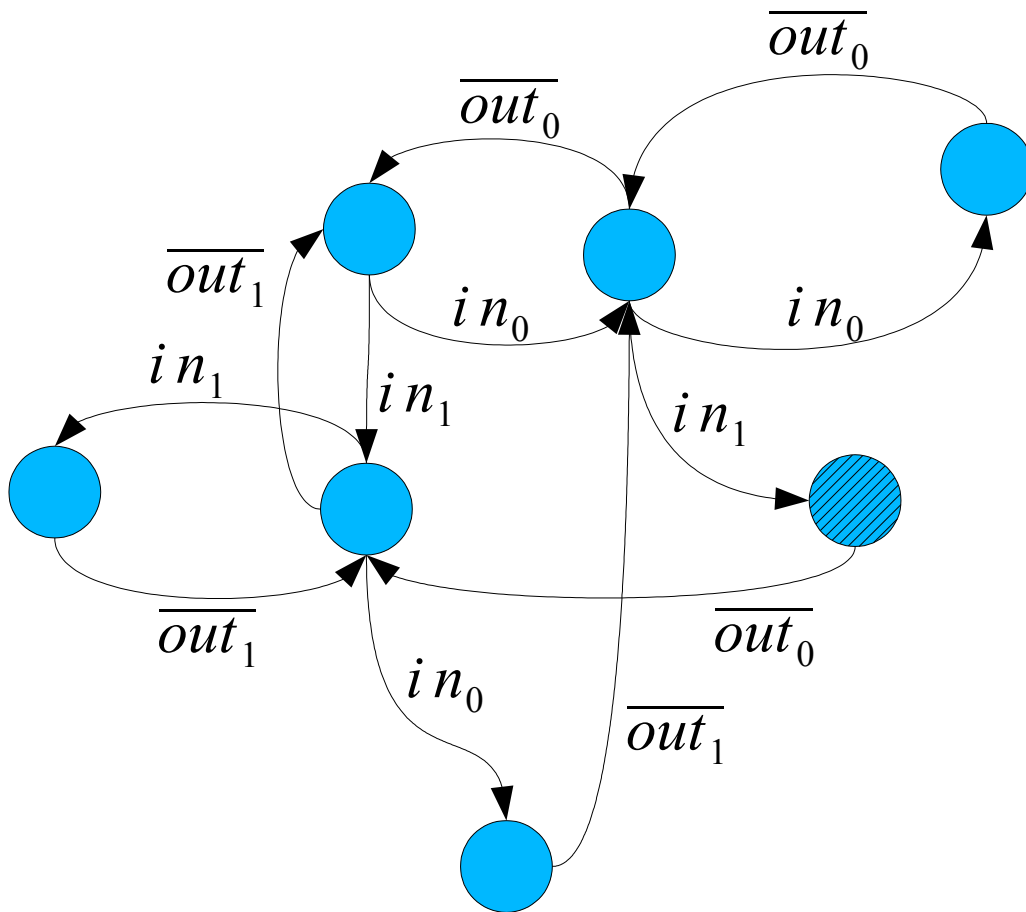
Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus



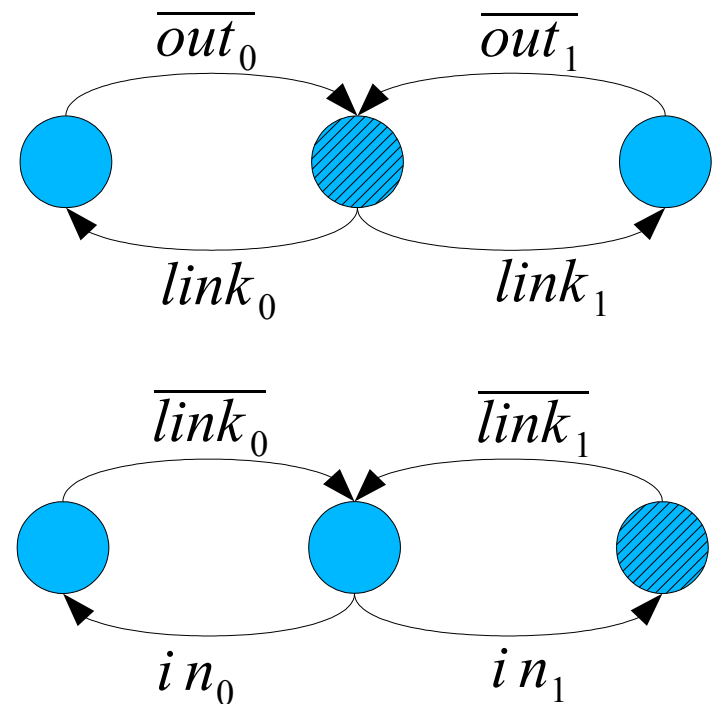
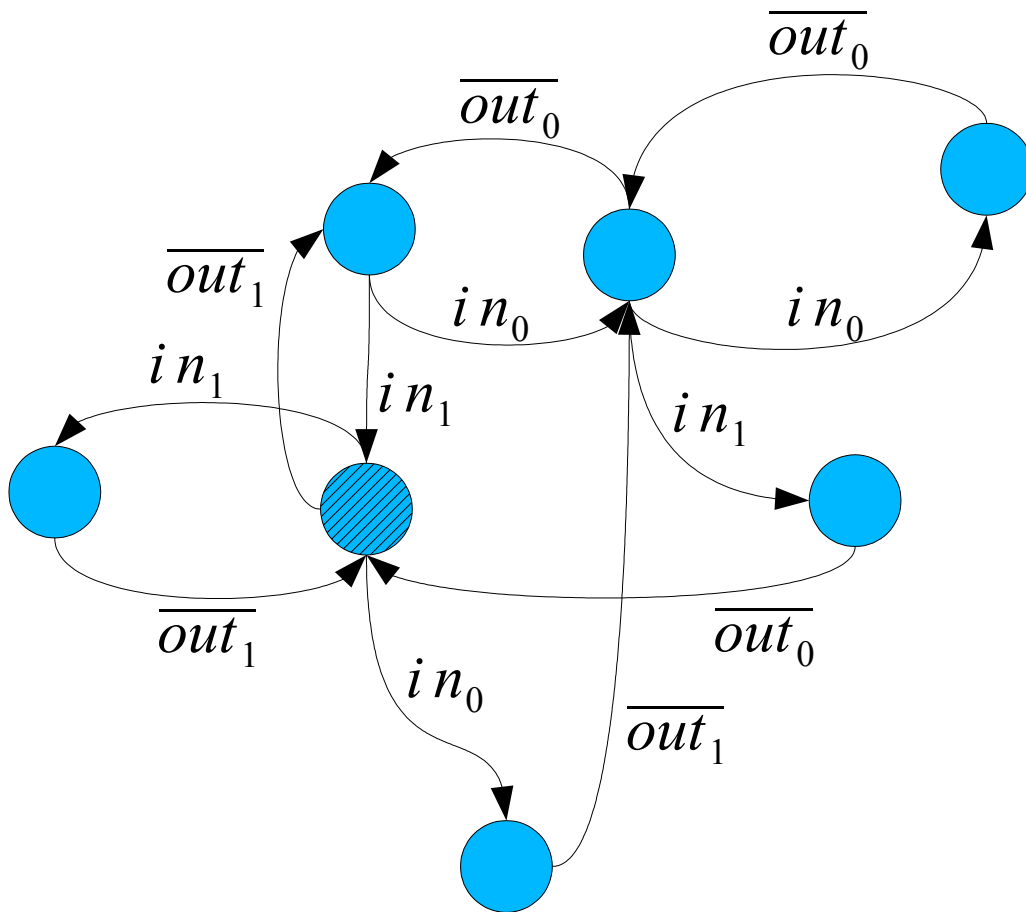
Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus



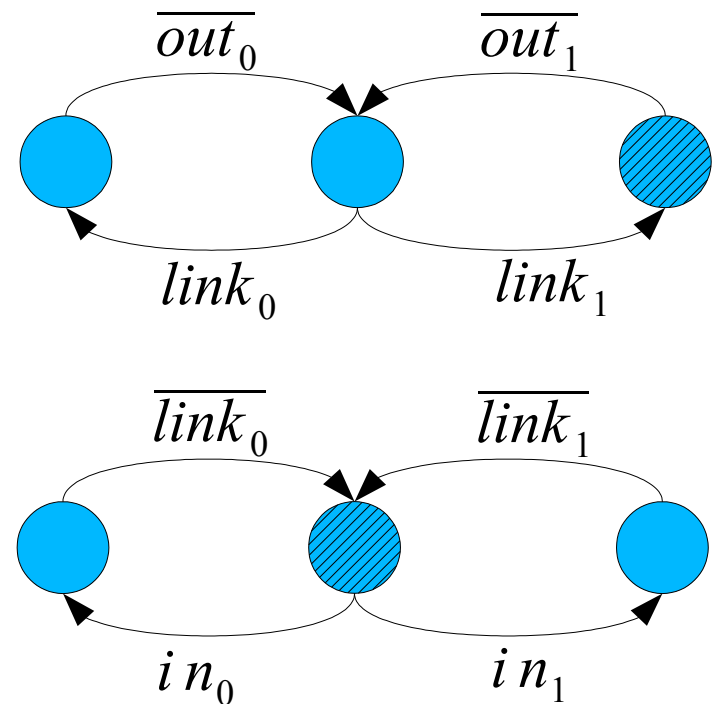
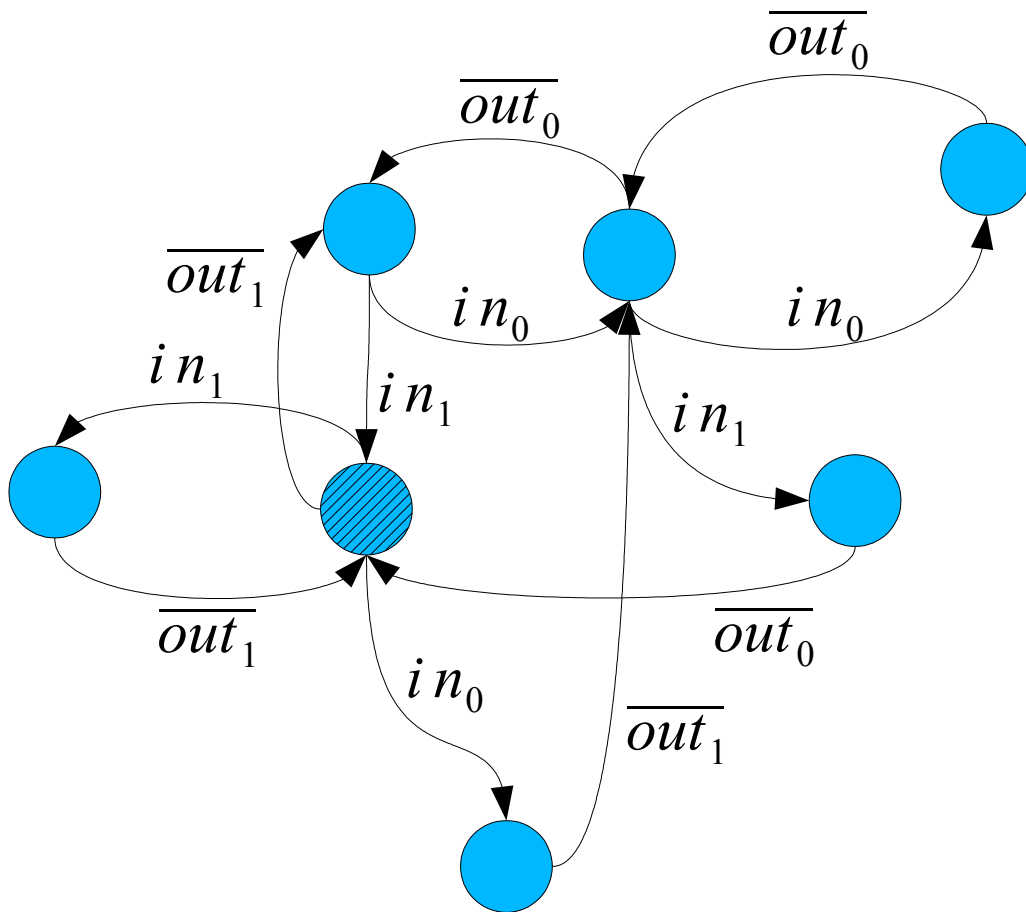
Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus



Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus



Beispiel: Buffer

Beispiel: speichere 01 in Buffer und lese diese Zahl wieder aus

